Learning robust control policies for real robots

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M.Sc. Miroslav Bogdanovic
aus Belgrad, Serbien

Tübingen
2021

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	14.01.2022
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Ludovic Righetti
2. Berichterstatter:	Prof. Dr. Martin V. Butz

Abstract

In this thesis we deal with the problem of using deep reinforcement learning to generate robust policies for real robots. We identify three key issues that need to be tackled in order to make progress along these lines. How to perform exploration in robotic tasks, with discontinuities in the environment and sparse rewards. How to ensure policies trained in simulation transfer well to real systems. How to build policies that are robust to environment variability we encounter in the real world.

We aim to tackle these issues through three papers that are part of this thesis. In the first one, we present an approach for learning an exploration process based on data from previously solved tasks to aid in solving new ones. In the second, we show how learning variable gain policies can produce better performing solutions on contact sensitive tasks, as well as propose a way to regularize these policies to enable direct transfer to real systems and improve their interpretability. In the final work, we propose a two-stage approach that goes from simple demonstrations to robust adaptive behaviors that can be directly deployed on real systems.

Contents

1 Overview of publications

This thesis encompasses work from three separate papers, two accepted publications and one submitted manuscript. We list the papers below and detail individual contributions in each work.

Accepted publications

Learning to explore in motion and interaction tasks [BR19]
Miroslav Bogdanovic, Ludovic Righetti
Published at 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Accompanying video available at https://youtu.be/lTX4ZdlYgXE.

- Author contributions: M.B., and L.R. designed research; M.B. performed numerical simulations; M.B. analyzed results; M.B. wrote the paper with support of L.R.

Learning variable impedance control for contact sensitive tasks [BKR20]
Miroslav Bogdanovic, Majid Khadiv, Ludovic Righetti
Published in IEEE Robotics and Automation Letters (RA-L)
Accompanying video available at https://youtu.be/AQuuQ-h4dBM.

- Author contributions: M.B., M.K., and L.R. designed research; M.B. performed numerical simulations; M.B. performed hardware experiments with contributions from M.K.; M.B. analyzed results; M.B. wrote the paper with support of L.R. and M.K.

Submitted manuscripts

Model-free reinforcement learning for robust locomotion using trajectory optimization for exploration [BKR21]
Miroslav Bogdanovic, Majid Khadiv, Ludovic Righetti
Submitted to IEEE Robotics and Automation Letters (RA-L)
Accompanying video available at https://youtu.be/lDPJzpVzLIk.

- Author contributions: M.B., M.K., and L.R. designed research; M.B. performed numerical simulations; M.B. performed hardware experiments with contributions from M.K.; M.B. analyzed results; M.B. wrote the paper with support of M.K. and L.R.

2 Introduction

2.1 Deep reinforcement learning in robotics

One of the overarching goals in the field of robotics is designing controllers that will allow robots to act in unstructured, real world environments. This is something that has repeatedly proven challenging for classical control approaches. Successfully being able to act in the real world usually requires extensive engineering to account for the variety that occurs in real world environments.

Deep reinforcement learning (DRL) seems very well suited for this task. In recent years, it has shown major successes across varied application areas and the idea of being able to train control policies in simulation to account for various aspects it could encounter in the real world seems attractive. Despite this, successes of DRL in robotics remain limited.

There has been a range of approaches showing impressive emergent behavior in simulation on robotic problems. Legged robots have been shown to learn to navigate complex environments with little guidance [HTS+17, PBYVDP17] or perform highly dynamic acrobatic maneuvers based only on simple demonstrations [PALvdP18]. However, achieving the same on physical robots has proven difficult.

Works such as [LFDA16] and [ABC+20] show impressive results for learning manipulation behaviors directly taking image data as input. However, the actual tasks performed in those works remain simple and the majority of complexity is limited to learning the vision aspect of the task.

One of the most impressive recent results has been [LHW+20]. There, policies trained exclusively in simulation enable a quadruped robot to robustly navigate highly uneven real world terrain. However, even that work is limited to comparatively slow motions and needs to utilize a highly structured control to be able to achieve those results.

There are several difficulties that are holding back DRL applications in robotics. We identify three key issues that prevent us from generating robust policies for real robots using DRL and throughout this thesis we aim to address each one. Firstly, the problem of exploration – how do we find good solutions in the first place in complex, discontinous environments with sparse reward signals. Second, how do we ensure that the policies trained in simulation preserve their performance when deployed on the real system. Third, how do we generate policies that are robust to the extensive variability that we can observe in real world environments.

2.2 Problems of exploration

Exploration is a key issue in almost any application of reinforcement learning. It is a particular issue in robotic tasks where there are discontinuities in the environment (arising from contact interactions) and the reward functions are often sparse. With the increase in the number of degrees of freedom of the system,

moving from simple single-leg systems to bipeds, quadrupeds and even full humanoid robots, it becomes more and more difficult to find even nominal solutions for most tasks.

One of the simplest approaches to improve exploration on control tasks has been to adapt the exploration process to produce trajectories more correlated in time. The idea behind it being that independently sampled controls at each timestep produce control signals that are not appropriate for most robotic tasks [Waw15]. An example of this approach can be seen in the Deep Deterministic Policy Gradient algorithm (DDPG, [LHP+15]), which uses an Ornstein-Uhlenbeck process [UO30] to generate time-correlated noise to use for exploration.

An often used method to circumvent exploration issues, with long history of applications in robotics, is to exploit demonstrations [ACVB09]. One promising way of utilizing demonstrations has been to train policies in simulation to track a demonstration trajectory as well as possible [PALvdP18, PCZ+20]. Rather than just learning to imitate exactly the controls given by the demonstration, these approaches enable finding of solutions even when the demonstrations are not perfect and behavior needs to be adapted at certain points. Alternatively, there are approaches that try to learn the residual control with respect to the demonstration [TZC+18]. While potentially simpler, this type of approach produces a time-dependent final policy. This is in conflict with the goal of using reinforcement learning methods to build adaptive policies, as timing of the behavior is fixed and cannot be adapted.

Another key element in dealing with exploration issues in robotic tasks is the choice of action space for the policy to act in. While directly controlling joint torques for the robot might give the most freedom to the policy, it often makes exploration very difficult. Approaches utilizing PD control to control desired positions for each joint instead have been shown to be less likely to get stuck in local minima [PvdP17]. There is still a lot of work in finding what is the best parametrization to be used for different robotic tasks, in order to gain most benefits in exploration by encoding strong structure, without sacrificing the ability to represent a diverse range of behaviors.

In this thesis, we will propose a new method for learning an exploration process based on data from previously solved tasks to aid in solving new, more complicated ones. Additionally, we will see how we can utilize demonstrations to aid in exploration while being able to adapt behavior away from what was demonstrated in order to account for environment uncertainty.

2.3 Transferring policies from simulation to real systems

In finding policies that will perform well on a real robotic system, the best approach would be to perform entire training on that same hardware. However, learning on real robots is expensive and potentially dangerous. The number of samples needed for finding policies for even the simplest tasks using reinforcement learning approaches can easily equal days or even weeks of training on the real system. This is ignoring practical aspects that such continuous running of a robotic system would entail. Apart from rare cases, like solving manipulation

problems with access to an array of robotic arms, doing the entire learning process on the real system like this is implausible.

Additionally, executing exploratory behaviors directly on hardware is inherently dangerous and can easily result in breaking the robot. A lot of guardrails need to be put in place to enable safe learning and it is difficult to make a good tradeoff between safety and giving enough freedom for the learning to happen.

For these reasons a wide range of reinforcement learning approaches relies on doing majority or entirety of the learning process in simulation and only then deploying those policies to real systems. Issue that needs to be addressed when utilizing those approaches is the discrepancy between simulation and the real world. Even the best simulators we have access to do not exactly mirror reality, especially when dealing with contact interactions. Simply performing training in simulation and expecting it to directly work on the real system is going to fail on all but most simple examples. Additionally, most precise available simulators are often not going to be used, as the goal of using the simulator is to significantly speed up training, so we are going to be limited to even less fidelity. Finally, even with very precise simulations, if we give the algorithm access to exact state information we can easily end up with incredibly fragile policies that exploit the noiseless deterministic nature of the simulation.

A direct approach to resolving this issue by trying to identify aspects of the system that are not well modeled inside the simulation. Usually, kinematics of the robot is something that we know with high precision. The element that is not as certain is the actuation model. It has been shown that learning this part of the system and replacing those elements of the simulation with the learned model improves the transfer of the policy to the real system [HLD+19].

Another, widely used approach for dealing with this issue has been domain randomization [TFR+17]. In particular, an aspect that proves key for transfer of policies learned in simulation to the real system is dynamics randomization [PAZA18], where the parameters of the robot are randomized. Instead of solving the problem for one specific model of the robot, which might not exactly match reality, learning is performed in such a way that the resulting policy performs the task on a range of slightly different robot models. This diversity makes it more likely to capture aspects of the real system and for the policies to retain their performance when deployed on those systems.

In addition to the previously discussed effects on exploration, the choice of action space also makes an impact on how well the policies can be transferred to real systems. In particular, imposing structure to the control of the robot makes it more likely for what is learned in simulation to be applicable in the real world. For example, utilizing PD control builds in feedback on the desired next position in the learned control policies, allowing it to account for some discrepancies between simulation and the real world and enabling the desired motion given by the policy to still be properly tracked.

Rather than relying on approaches such as domain randomization, in this thesis we focus on regularizing the trained policies to produce desired joint trajectories that can be effectively tracked by the closed-loop controller. We propose a novel regularization approach that we utilize across multiple works to produce

policies that can be successfully deployed to real systems without any additional training.

2.4 Building robust policies

Creating policies that are robust to varied range of environments the robot can act in is one of the main reasons for applying deep reinforcement learning methods to robotics problems.

A predominant way for achieving these robustness characteristics when performing training in simulation is to vary the environment the policy is acting in during training in some way [HTS+17]. These approaches have produced complex emergent behaviors in simulation, with robots learning to balance over uneven terrain or jump over obstacles and gaps.

Particularly fruitful direction of research has been how to perform these randomizations of the environment. While we would like to be robust to significant variability in the environment, starting training with sampling from such diverse set might lead to the reinforcement learning algorithm failing to converge. One approach that has been showing potential in dealing with this is usage of curriculums of environments of increasing complexity [HTS+17, YTL18, WLCS19, XLKvdP20]. This way the complexity can increase slowly, potentially taking into account current performance of the policy, so that the reinforcement learning algorithm always receives the right difficulty of the problem to solve.

For a lot of these approaches however there is a gap between what we see in simulation and the types of behaviors we know are possible on real systems. There has been some application of this general approach recently with actual deployments to real systems [LHW+20]. However, that work has both been limited to slower motions as well as having prior structure imposed on the policy. Being uncertain about a location of a contact is a significantly easier complication to deal with in a case when the robot has no flight phase and the endeffector can slowly move until it encounters contact. It becomes more difficult when the robot motion has a flight phase, contact is encountered at high velocity and needs to be prepared for.

A large amount of work however remains to be done in this area, in particular related to building policies that will be robust when deployed on real systems. The way to structure the learning process as well as which aspect of the environment to randomize are key to achieve policies that can act in real world environments.

Building robust policies for real robots is a key focus of this thesis. We examine the influence of the choice of action space on the robustness of learned policies and propose training variable gain policies in order to find better performing policies in the presence of contact uncertainties. We examine how to structure environments the robot is acting in during training and how to combine randomization of aspects of the environment with specific reward terms to produce robust behavior that carries over to real systems. Finally, we propose a two-stage method, where we utilize demonstrations to aid in learning, but are still able to fully adapt the policies in order to find truly robust solutions.

2.5 Thesis overview

This thesis covers work from three separate papers, addressing overlapping subsets of the three issues presented above. In Chapter 3 we first give an overview of the contributions of each paper. We then go over simplifying assumptions we make across our work and give details of the robot hardware used in Chapter 4. We then group topics from different papers based on the three key issues they are dealing with as follows:

1. **Learning to explore in motion and interaction tasks [BR19]** – This paper presents a way of learning how to explore when performing reinforcement learning on robotic tasks, based on data from a set of simpler tasks. We cover this in Chapter 5.1.

2. **Learning variable impedance control for contact sensitive tasks [BKR20]** – This work is concerned with the choice of action space for producing robust policies in contact sensitive tasks. In particular, it proposes learning variable gain policies to improve performance on these tasks, we cover this in Chapter 7.2. It also introduces aspects that we reuse in the next paper. First, a way to regularize the policies being learned to impose physical meaning to them and ensure their effective transfer to real systems, which we address in Chapter 6.1. Second, which aspects of the environment need to be randomized and in which way to ensure high level of robustness of policies when deployed to real systems, covered in Chapter 7.1.

3. **Model-free reinforcement learning for robust locomotion using trajectory optimization for exploration [BKR21]** – In this work we propose a two-stage approach for building robust policies starting from simple demonstrations. We cover the conflict between tracking time-based trajectories and being robust to variable environment conditions and how our approach resolves this in Chapter 7.3.

After, we discuss where the overall research has led us to and what next steps can be taken along those directions in Chapter 8. Finally, we give concluding remarks for the thesis in Chapter 9.

3 Individual paper contributions

3.1 Learning to explore in motion and interaction tasks

In this paper, we show how we can utilize data from previously solved tasks to build an exploration model that can improve learning of new, more complicated ones. We examine the performance of the approach in simulation, on a set of manipulation tasks of increasing complexity. We show how using the proposed approach improves both learning speed and reliability of finding solutions in new tasks. We also demonstrate how this method can be used in incremental solving of a set of tasks with increasing complexity, by using the solutions of each new task to expand the capability of the exploration model.

This work, while preliminary, shows an interesting potential for the design of robot-centric exploration strategies. While, for example, utilizing demonstrations to aid in learning can be simpler, approaches like the one we propose should find their place in more complex tasks, where good demonstrations are hard to come by and we need an alternative way to bootstrap learning.

3.2 Learning variable impedance control for contact sensitive tasks

There are two key contributions we make in this work. The first one is the proposed regularization scheme driving the policies to output joint trajectories that can be well tracked by the PD controller. This allows us to produce policies that can be deployed on real systems without need for dynamics randomization during training. It also allows us to produce policies that are more interpretable, with each output having its original physical meaning.

Our second contribution is the use of variable gain policies that can be learned using the above regularization scheme. We show how we can find better solutions more reliably utilizing this control structure on both fixed and floating base robots. We also show how this performance carries over to the real system.

Overall, we see no degradation in learning performance from increase in the number of outputs of the policy in the case of variable gain control. We are also able to avoid the issue of having many combinations of outputs resulting in the same behavior, by having the regularization scheme incentivize those with proper physical meaning. These aspects combined make it reasonable to use more complicated control structures when applying deep reinforcement learning in robotic tasks and have potential to be extended to other control schemes appropriate for specific tasks.

3.3 Model-free reinforcement learning for robust locomotion using trajectory optimization for exploration

The final paper in this thesis goes furthest in the overarching task of achieving robust behavior on real robots. The general approach presented there allows us

to go from single demonstration trajectories to robust policies that we can deploy on real robots without the need for any additional training.

The main advantage of the approach is its simplicity and generality. We only require a single demonstration trajectory for a given task. While we utilize trajectory optimization methods to arrive at those trajectories in our work, the approach is agnostic to the source of the demonstration. This trajectory only needs to encode some basic behavior on the task. It does not need to optimize any aspect of the performance, as that can be later done in the second stage of training. It only needs to give us a hint of the kinematic motion needed to complete the task. We do not even make use of the actions from the demonstration, just of the sequence of states. This makes it so that we can easily use a different controller during training than what was used for generating the demonstrations.

On the other side, as using demonstrations allows us to overcome exploration issues for the most part, the task reward we give in the second stage can be kept quite simple. We can have it encode directly the aspect of the task we are interested in optimizing, without the need for any extensive reward shaping.

Using this approach we are able to produce highly robust behavior on a real quadruped robot for dynamic hopping and bounding tasks. The behavior is robust to a wide range of environment variability (like uneven and soft ground) as well as external perturbations (pushes during execution). As a result of generality and simplicity of the approach we are hopeful that it can be further extended to a wide range of tasks and to different robotic systems.

4 Simplifying assumptions and hardware details

Before examining specific contributions we make on each of the three key issues described, we give details on the simplifications we make to be able to focus on those issues. Additionally, we give specifics of the hardware used that is key for enabling this type of work.

4.1 Simplifying assumptions

Overall we do not make any use of high-dimensional observations, like vision or touch sensor data, as part of the policy input. Instead, we generally restrict ourselves to direct measurements of the robot state (joints and base state). Additionally, we do not make any observations of the environment and instead aim to be robust to a set of variable environment configurations.

We make these simplifications to be able to focus on the behavioral aspect of the policies and in particular the interactions of the robot with the environment. Deep reinforcement learning approaches that learn direct mapping from high dimensional sensory data to controls have shown impressive results [LFDA16, ABC+20]. However, most of the complexity in those approaches remains in the observational part of the system.

In our work, we focus on how the policy acts in the environment, the interactions of the robot with the world and how we can create robust policies than can act in a variable, uncertain environment. Simplifications we make are crucial to be able to examine these aspects. However, there should be nothing preventing all the results presented here from being utilized as parts of a more complicated system, utilizing diverse sets of complex observations.

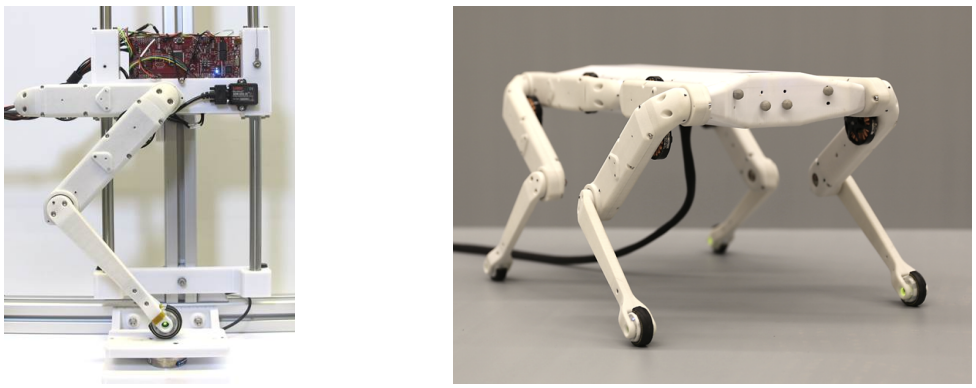
We also do not use any extended state history in the input to the policies or use policies with any internal memory. The one exception is the recurrent neural network based exploration model in [BR19]. But even in that work, the policies that are trained using that exploration model have no memory and are only based on the current state.

Policies with some memory capabilities have been used in robotic tasks, mostly in combination with domain randomization [PAZA18]. In those cases, the policy having access to longer state history or internal memory can allow it to implicitly identify either dynamics parameters or some aspects of the environment and act accordingly. We do not randomize robot parameters in our works and instead rely on imposing structure to the policies to ensure effective transfer to real systems. This removes one reason for having memory in the policies, as with a single instance of robot dynamics there is nothing to identify in that respect.

On the other hand, as for the environment variability, we explicitly do not want the policies to be able to identify specific aspects of the current configuration. Instead, we want them to be robust to a range of environment configurations that can be encountered. This adds more robustness into the policies, as they are able to account for environment changing during execution of the policy. For example, hopping policies in [BKR20] being able to continue hopping as the ground

height changes or policies in [BKR21] being able to deal with uneven ground and external pushes.

4.2 Robot hardware



(a) One-leg hopper (b) 8-DoF quadruped

Figure 1: Two hardware platforms used in this work. Images taken from [GMK+20].

Another key element for the success of the work covered in this thesis that cannot be disregarded is the hardware used for experiments. For all real hardware experiments we used lab-built robots based on the Open Dynamic Robot Initiative (ODRI) architecture [GMK+20]. It represents a simple, low-cost framework for building a range of robots with widely available parts.

Common hardware design between different robots allowed us to quickly move between them, reusing results and knowledge gained from working on one when moving to the next. The hopping experiments on a single leg setup used in [BKR20] provided a starting point for performing a more difficult task of robust hopping on a quadruped robot in [BKR21]. This also gives hope for the work presented here to be used on other robots based on the same design, with the two currently available ones being a 12-DoF quadruped and a 6-DoF biped.

Additionally, the hardware platform proved very well suited for deploying reinforcement learning policies to real systems. Safety is one of the main concerns in evaluating policies learned in simulation on real systems. Potential for expensive hardware failures can make such work exceedingly difficult. The platform we used proved robust and durable, with failures rarely resulting in hardware breaking. More importantly, with the modular 3D printed design of the robots, in the cases of parts of the robot breaking, repair procedure was simple and quick.

Finally, the open source aspect of the hardware framework and relative simplicity of reproducing it should aid in the reproducibility of the presented results, something that is often a key difficulty in robotics research.

5 Resolving exploration issues

In each work that is part of this thesis we approach exploration in a different way. In [BKR20] we examine comparatively simpler tasks for which policies can be learned from scratch, without any special approach for exploration. In [BKR21] we use demonstrations to bootstrap learning and then focus on dealing with issues that arise when learning to imitate given trajectories. In [BR19] on the other hand we deal exclusively with the problem of exploration.

5.1 Prior knowledge from simpler tasks

Most approaches to resolving exploration issues rely on prior knowledge in some form. In [BR19] we examine how we can use knowledge gained from solving a set of tasks on a robotic system to aid us in solving new, potentially more complicated tasks that share some characteristics with the original ones. For example, if we know how to solve a task where we use a robotic arm to exert a force on a specified point on a flat surface and a task where we do circular motion of the endeffector in free space, can this aid us in solving a task where we are asked to exert a force on a table while performing a circular motion?

Our proposed approach uses data collected from previously solved tasks to build an exploration model, one that can be used to generate random behaviors with characteristics seen in those tasks. When solving a new task, we then rely on samples from this model to explore rather than relying on random noise. We propose that utilizing these samples of more sensible behavior would improve exploration and allow us to find solutions to new tasks faster and more reliably.

The key concept we rely on in building this exploration model is estimating the combined distribution over actions arising from all the previously seen tasks at some point during an episode. In a sense, determining at some point which actions make sense based on what we have done previously and which do not. First, we examine this combined distribution over actions when conditioned on a current state of the system. For each action, we take the sum of probabilities for that action from each available policy, weighed by the likelihood of occurrence of the state in that task:

$$\pi_{EXP}(\mathbf{a_t}|\mathbf{s_t}) = \frac{1}{Z}\sum_{i=1}^{N}\mu_i(\mathbf{s_t})\pi_i(\mathbf{a_t}|\mathbf{s_t}) \tag{1}$$

To formulate it in another way, if we take a collection of (state, action) pairs from execution of all these tasks, we evaluate which actions are taken when the system is in state s_t in this combined dataset. We could now use samples from this distribution to explore, rather than relying on random noise.

The issue with this approach is that it will often prove uninformative, as it fails to capture any correlations across time. Conditioned only on a single state, a wide range of actions might be plausible across the examined set of tasks. Generating trajectories by taking consecutive samples from this distribution would result in behaviors similar to those seen when performing exploration using ran-

dom noise. We would not observe any coherent exploratory motions of the system we are trying to achieve.

This is why we aim to estimate the same distribution over actions, but instead conditioned on a longer history of preceding states:

$$\pi_{EXP}(\mathbf{a_t}|\mathbf{s_{t-K+1}},\ldots,\mathbf{s_t}) = \frac{1}{Z}\sum_{i=1}^{N}\mu_i(\mathbf{s_{t-K+1}},\ldots,\mathbf{s_t})\pi_i(\mathbf{a_t}|\mathbf{s_{t-K+1}},\ldots,\mathbf{s_t}) \tag{2}$$

This way we preserve all the key correlations across time. Sampling from this model gives us trajectories that smoothly continue the behavior preceeding the current state – they continue moving in a similar direction or continue applying force if in contact.

One additional benefit from utilizing this approach is that it allows us to build a curriculum of more and more complex tasks on a system. As the data used to train the exploration model can come from any source, it can also come from policies trained from previous application of the same approach. With this we can take the simplest tasks we already know how to solve (moving a robotic arm to a point in space), train an exploration model on that data and use it to solve a next set of tasks (moving the arm along a trajectory or going to a surface and applying a force) and then train a new model using the combined data to help on an even more complex task (performing a motion while applying a force). The approach is also modular, as we can choose the data containing the types of behaviors we will have on a new task in training an exploration model, making it more suited for the particular thing we currently care about.

6 Building transferable policies

Throughout our work we do not rely on approaches such as dynamics randomization to ensure transferability of learned policies to real systems. Here we show how instead we structure the policies to make them inherently transferable.

6.1 Learning structured control laws

When learning control policies for robotic tasks using reinforcement learning we often have the policy output parameters for a specific controller, rather than directly outputting control torques for each joint. Most commonly, we use a PD controller and have a policy give desired positions (and potentially velocities) for the controller. The expectation when we do this is that the policy will output a trajectory of desired positions in time, that is then properly tracked by the controller. Unfortunately, just by structuring the control like this there is nothing to ensure that that is what actually happens. What we often see in practice is that the policy can output highly non-smooth signals, which when put through the control law results in success on the relevant task. We cannot however successfully deploy these policies on real robots as any differences between simulation and the real world cause a control output like this to produce completely different behavior resulting in task failure and potentially even being unsafe.

What we want is a way to ensure that the policy outputs a trajectory that can be tracked by the controller as much as possible. In [BKR20] we propose a way of achieving this, by adding a new reward term during training. We design this reward term to penalize the difference between the desired position value given by the policy at timestep t and the actually achieved position and timestep $t+1$:

$$r_{tt} = -k \left\| \mathbf{q}^{\mathbf{t}}_{\mathbf{des}} - \mathbf{q}^{\mathbf{t+1}} \right\|^2 \tag{3}$$

This drives the policy to produce desired position values that can actually be tracked well by the PD controller.

The effect of the proposed reward term is particularly interesting when combined with the variable gain control law. With the variable gain control we have the reinforcement learning policy output both the desired joint positions and the values of the gains for the PD controller to track these desired positions. For simplicity, we will look only at the P part of the control law here:

$$\boldsymbol{\tau} = \mathbf{K_P}(\mathbf{s})(\mathbf{q_{des}}(\mathbf{s}) - \mathbf{q}) \tag{4}$$

With the reward term forcing the policy to output trajectory that is actually tracked, this leaves a free term $\mathbf{K_P}(\mathbf{s})$ to ensure that this tracking is actually performed. What this results in, as we show in [BKR20], is that the gains go low when the robot is performing free space motion and not a lot of torque is needed to track the trajectory, but go high in contact when that is needed to ensure that the trajectory is tracked properly.

We can also compare our approach to simply having a reward term incentivizing smooth outputs of the policy, which is sometimes used in dealing with

the same issue of noisy outputs. Rather than driving policy towards any smooth trajectory equally, the reward term we propose biases it towards the one that can be realized on the system – one that is in agreement with the physics of the system. It also gives no penalty for any case of outputs that can be well tracked, allowing for fast motions if the system allows for it.

We can see how the addition of this reward term changes the resulting policy output in Fig. 5 in [BKR20]. We can observe how the tracking of the outputted trajectory improves and how in the case of variable gain control this results in more sensible values for control gains – going down to ensure soft contact and increasing to allow force application when in contact. In the same work we also note how addition of this reward term allows hopping policies to successfully be deployed to the real system, with no dynamics randomization during training and no additional training on the real system.

7 Achieving robustness on real robots

Our main approach for finding policies that will provide robust behavior on real systems is to introduce variability in the environment during training for the policies to learn to account for. Below, we will first cover general aspects of how we introduce these variabilities and specific aspects we aim to achieve by doing so. Next, we discuss how the choice of action space, in particular learning variable gain policies, can make it easier for us to find policies robust to these uncertainties. Finally, we deal with the issue of how we can still generate fully robust policies while utilizing demonstration trajectories to bootstrap the initial learning process.

7.1 Environment variability during training

The main way we aim to achieve robust policies is to introduce variability during training for the policies to learn to account for. What is randomized and how proves to be key in actually achieving robustness on real robots with this approach.

The most important variability we introduce during training is that of contact location. That, combined with a reward for soft contact transitions, proves to be key in finding policies that interact well with the environment and that can because of it be deployed on physical systems. It also illustrates one of the benefits of the reinforcement learning approaches in these types of tasks. We can find emergent behaviors that are sometimes difficult to predict. For example, with the above defined setup of uncertain contact location and requirement for soft contact transition the expectation would be for the policy to be soft in preparation for contact and to absorb the contact when it happens. While these aspects are crucial and they are achieved with this type of training, what we additionally observe is lifting of the foot throughout the range of locations where the contact could perceivably happen. This reduces the impact velocity and additionally contributes to the contact transition being smooth.

Other important aspect in the search for generally robust policies is that we do not need to randomize exactly the specific aspects we expect to be variable in the real world environment. We can observe this best on the quadruped behaviors in [BKR21]. The main randomization we perform in that work is still that of contact location. However, we simply randomize the vertical position of a flat ground surface, resulting in same ground height under each foot. In addition to that, we randomize the initial state of the robot base, where we change the height and angle we start the motion from during training.

These two aspects in combination produce more than simply robustness to specific variability in those two values. When deploying trained policies to the real robot, we observed that the behaviors exhibit significant robustness to uneven and soft ground, as well as external pushes, none of which they were explicitly trained for. Having not been explicitly trained for it the policies do not account for this type of variability right away. They do not adapt the leg configuration to account for different contact height for each foot, something that

could allow the base to potentially stay perfectly horizontal throughout the motion. External pushes are also not absorbed, which could reduce the perturbation they cause. What happens is that the perturbation causes the base configuration to go off the nominal trajectory, however what the training has produced is robust recovery behavior for such cases. With the randomization of the initial base configuration, the state the base ends up in is not novel for the policy and it knows what to do to go from there back to the nominal motion cycle. In combination, these two aspects, soft contact transitions and robust recovery behaviors, produce highly robust motions on the real system with no additional training required.

7.2 Learning variable gain policies

Across the works included in this thesis, a key interest we have is building policies that interact with the environment. One of the key elements of this is the ability of the policy to smoothly go in and out of contact, particularly when there is uncertainty in contact locations.

Simplest way of having the reinforcement learning policy controlling the system is by having it directly output desired torque values for each joint.

$$\boldsymbol{\tau} = \boldsymbol{\tau}(\mathbf{s}) \tag{5}$$

This approach imposes no structure to the solution, which has its benefits, but also drawbacks. All other control structures that can be imposed would still result in torque being some function of the system state, and direct torque control can in theory learn to replicate any of these more complicated control laws. However, this might require a much more complicated function to be learned.

Other commonly used approach is employing PD control with fixed, pretuned gain values:

$$\boldsymbol{\tau} = \mathbf{K_P}(\mathbf{q_{des}}(\mathbf{s}) - \mathbf{q}) - \mathbf{K_D}\dot{\mathbf{q}} \tag{6}$$

This structure makes exploration easier and it has been shown that it makes the learning problem easier [PvdP17]. However, it is a bad choice for controlling the interactions with the environment, where even small deviations in the environment can result in large forces being applied and even potential damage to the system.

As an alternative to both of these approaches we propose using variable-gain control, where the policy controls both the desired position for each joint and corresponding gain values:

$$\boldsymbol{\tau} = \mathbf{K_P}(\mathbf{s})(\mathbf{q_{des}}(\mathbf{s}) - \mathbf{q}) - \mathbf{K_D}\dot{\mathbf{q}} \tag{7}$$

Benefits of using a variable gain control law are maybe best seen in the example shown in Fig. 9 in [BKR20]. There we examine a task of performing a circular motion on a table while applying a specified force to it. What is shown are forces applied to the table when different control laws are used. Unlike direct torque control or PD control with fixed gains, variable gain control is able to preserve contact throughout the motion. The other two control schemes either lose contact at various points or perform repeated impacts to the table trying to stay in

contact. Utilizing variable gain control makes solving the task well simpler, as gains can be reduced appropriately so that the contact behavior such as this is simple for the policy to learn.

7.3 Time-dependent demonstrations and robustness

In our work with variable gain policies, we performed all the learning from scratch, which was viable for comparatively simpler setups used in that work. Moving to more complex tasks, we need to utilize some approach to circumvent the exploration issues that arise. We already presented one way to approach the problem utilizing knowledge from simpler tasks we know how to solve (5.1). As an alternative, in our approach in [BKR21], we utilize demonstration trajectories to resolve the same issues.

Learning to track demonstration trajectories as well as possible inside the simulation has proven effective in learning complex control policies [PALvdP18, PCZ+20]. It proved equally effective in the tasks we examine in [BKR21]. When demonstration trajectories are readily available it is a simple and general way to generate policies performing some nominal behavior on a task.

What we are concerned in our work is what happens if we want these policies to be robust to a wide range of environment parameters. It turns out that having a policy try to track a time-based demonstration trajectory during training while accounting for significant environment uncertainties is not effective. We can examine this on an example task. Let us look at a problem where a robot is trying to learn how to hop in an environment where ground location is uncertain (similar to the setup we have in [BR19]). We assume we have access to a demonstration trajectory of a hopping behavior with some nominal ground position and that we are trying to reproduce that behavior as close as possible.

With different ground locations, the contact of robot with the ground will happen sooner or later than it is expected in the demonstration trajectory. This causes desynchronization between the policy and the trajectory it is trying to track. The policy now needs to wait for the demonstration (if the contact happened sooner than expected) or to speed up the motion to catch up with the demonstration (if contact happened later). Neither of which results in a smooth adaptive behavior we desire. Being locked into the timing of the demonstration trajectory makes proper adaptation to varied environment conditions hard and sometimes even impossible.

This is the problem we tackle in [BKR21]. We would like to be able to use demonstrations as a simple and general way to avoid exploration issues in complex tasks, while being able to freely adapt the behavior (in particular the timing of it) to account for environment uncertaities. We achieve this with a two-stage approach. In the first stage, we try to imitate the demonstration trajectory in simulation as well as possible to get a nominal policy for the task to bootstrap learning. In the second stage, we perform further training of this policy, but fully eliminate the imitation reward and replace it with a direct time-indepent task reward. At the same time, we introduce variability in the environment that the policy is now able to account for without conflict with the reward for tracking

the demonstration.

This proves to be a simple and powerful approach for learning highly robust policies for dynamic tasks, that can crucially successfully be deployed on real systems. When analyzing data from the execution of learned policies on the real system, we see exactly the kind of adaptation of timing to account for environment variability. The hopping policy for the quadruped robot when initialized by dropping the robot from different heights, lands softly regardless of the height and smoothly transitions to nominal behavior through several hops. ([BKR21], Fig. 3)

8 Discussion and future work

8.1 Utilizing learned exploration models

Our work on learning exploration models showed promising results, however it only scratched the surface on research in this direction. One topic that definitely deserves more attention is how we use the trained models to learn on a new task, in particular how we make tradeoffs between exploration and exploitation. While the simple approach we use in our work proves effective, it should be possible to explicitly combine the distributions over action space arising from the exploration model and the current distribution from the policy being learned. This would require a change to the reinforcement learning algorithm being used as the one we employ (DDPG, [LHP+15]) uses a deterministic policy and therefore does not provide us with the required probability distribution.

8.2 Learning with alternate control laws

The regularization term we propose in [BKR20] is in fact quite general and not restricted to just learning with the control laws presented in that work. In particular, an interesting control law to take into account is a PD control law with an additional feedforward term:

$$\boldsymbol{\tau}(\mathbf{s}) = \mathbf{K_P}(\mathbf{q_{des}}(\mathbf{s}) - \mathbf{q}) + \mathbf{K_D}(\dot{\mathbf{q}}_{\mathbf{des}}(\mathbf{s}) - \dot{\mathbf{q}}) + \boldsymbol{\tau}_{FF}(\mathbf{s}) \tag{8}$$

The idea behind a control law like this is to have the feedforward part, $\boldsymbol{\tau}_{FF}(\mathbf{s})$, realize the nominal motion, while the feedback part, $\mathbf{K_P}(\mathbf{q_{des}}(\mathbf{s})-\mathbf{q})+\mathbf{K_D}(\dot{\mathbf{q}}_{\mathbf{des}}(\mathbf{s})-\dot{\mathbf{q}})$, handles disturbances and ensures that the desired trajectory is tracked. However, by just having a policy with these three outputs, $\mathbf{q_{des}}(\mathbf{s})$, $\dot{\mathbf{q}}_{\mathbf{des}}(\mathbf{s})$, $\boldsymbol{\tau}_{FF}(\mathbf{s})$, there is nothing to ensure that this is the case. In fact, in the case of a control law with a complex structure such as this one, the output we get becomes even more chaotic, with practically no interpretability and no chance of transferring to the real system.

We can apply our proposed regularization scheme to this control law with two separate reward terms. First one being the same as we have used in previous work, penalizing the difference between desired joint position at timestep t and actual position at timestep $t+1$:

$$r_{tt} = -k \left\| \mathbf{q}^{\mathbf{t}}_{\mathbf{des}} - \mathbf{q}^{\mathbf{t+1}} \right\|^2 \tag{9}$$

The second would be an equivalent one for joint velocities, penalizing different between desired value at timestep t and actual value at timestep $t+1$:

$$r_{vt} = -k \left\| \dot{\mathbf{q}}^{\mathbf{t}}_{\mathbf{des}} - \dot{\mathbf{q}}^{\mathbf{t+1}} \right\|^2 \tag{10}$$

Now, we have three policy outputs ($\mathbf{q_{des}}(\mathbf{s})$, $\dot{\mathbf{q}}_{\mathbf{des}}(\mathbf{s})$ and $\boldsymbol{\tau}_{FF}(\mathbf{s})$) and a regularization terms on two of them (desired position and velocity). This leaves the feedforward torque term free to actually enable that the trajectory is tracked.

Our initial experiments in simulation with control laws such as this one showed potential. We were able to learn structured policies with proper tracking for both position and velocity, with the feedforward term increasing during contact phases to enable proper tracking.

8.3 Accounting for even more environment variability

One straightforward extension to the work we propose in [BKR21] is training policies to account for even more environment variability. For example, we expect even better performance of hopping and bounding policies we showed in that work if they would have been explicitly trained to account for uneven ground. It would be interesting to see how far we can push the uncertainties and if and how the policy could account for them.

However, from our initial experiments applying this amount of uncertainty from the start of the second stage of training results in the RL algorithm failing to find a solution. The policy fails in an overwhelming percent of the new random environment and the training cannot proceed.

One approach that shows potential for resolving this is using curriculums of environments of increasing complexity during the second stage of the training. Curriculum learning has been shown to aid in learning of challenging tasks [XLKvdP20]. There is still work to be done on how best to extend and adapt the difficulties during training, but these types of approaches seem like the exact thing needed to take our proposed approach further.

8.4 Goal conditioned policies

Generally, the policies we have used across our work have been designed to solve one specific instance of one task. For example, we had a policies that applied force to a point on a table, which worked for one specific, pre-defined point or hopping policies, performing continuous hopping with some specific height. The main issue with this is that if we want to solve a different instance of the same task we would need to repeat the entire training.

A simple solution for this issue would be an addition of an input to the policy for selecting or parametrizing task goal or some other aspect of the motion. This however comes into conflict with the approach we have been utilizing of employing demonstrations to aid in learning. We would now require a new demonstration for each task configuration which, depending on the source of the demonstrations, could easily prove infeasible.

One hopeful aspect for the approach we propose is that we can generate a wider range of behaviors from a single demonstration. For example, attaining different hopping height or different bounding cycles. With this there is some hope that we can start from one or a small number of demonstrations, for some select task configurations, and through the second stage of the training adapt them to cover the full possible range of task instances.

9 Concluding remarks

Deep reinforcement learning has great potential in many different robotic applications. Training control policies to account for a wide set of potential environment configurations can allow us to learn robust behaviors, able to recover from wide range of perturbations and succeed in completing the tasks. With this type of approach we can find policies that not only immediately respond to any perturbations, but even adapt behavior in advance to account for different possibilities.

In this thesis we dealt with several issues that need to be addressed for this to become reality. Exploration is likely to remain one of the dominant problems when attempting to find policies for difficult robotic tasks. While it is an issue in most reinforcement learning applications, it is particularly present in robotics, due to discontinuities in the environment and often sparse rewards. Employing demonstrations to circumvent these issues has proven successful in various tasks and it allows us to explore other aspects needed for solving the overarching problem. However, it is likely that more structured approaches for exploration will be needed for the most complex tasks. There are reasons to be hopeful about work in this direction. There is a lot of structure encoded in the physics of the robot, environment and their interaction and we should be able to exploit some aspects of that to vastly reduce the search space of policies, like we attempt to do in [BR19].

Another aspect that is proving to be crucial is how we structure the control law that the reinforcement learning policies are utilizing. This is particularly important when we are concerned with deploying learned policies on real systems. While a lot of tasks could in theory be done with policies directly controlling the desired torques for the robot, alternative parametrizations have the potential to make learning much easier without sacrificing any expressivity. Even more than that these control laws give us some additional structure apart from just what can be learned in simulation, as well as allowing for much better interpretability of what the learned policies are actually doing – something that is a key in long term robotic research. In our work we addressed a general way of enabling proper learning of structured control laws. What exact control law should be used for what problem is a difficult question, likely without a simple answer, but research in this direction is well positioned and likely to provide fruitfull results.

Finally, something we spent a lot of time on in this thesis is generating policies that are robust to variable environments. Basic approach of training policies on a varied set of environment configurations has shown a lot of early success, but it is also the aspect where there is most work left to be done. What and how precisely is randomized and how the training of the policies is structured to be able to find policies robust to it are key questions with a lot of room for future research.

10 References

[ABC+20] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[ACVB09] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[BKR20] Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Learning variable impedance control for contact sensitive tasks. *IEEE Robotics and Automation Letters*, 5(4):6129–6136, 2020.

[BKR21] Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Model-free reinforcement learning for robust locomotion using trajectory optimization for exploration. Manuscript submitted for publication, 2021.

[BR19] Miroslav Bogdanovic and Ludovic Righetti. Learning to explore in motion and interaction tasks. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2686–2692. IEEE, 2019.

[GMK+20] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020.

[HLD+19] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[HTS+17] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[LFDA16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[LHP+15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[LHW+20] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[PALvdP18] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[PAZA18] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[PBYVDP17] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[PCZ+20] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[PvdP17] Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13, 2017.

[TFR+17] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[TZC+18] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[UO30] George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[Waw15] Pawel Wawrzynski. Control policy with autocorrelated noise in reinforcement learning for robotics. *International Journal of Machine Learning and Computing*, 5(2):91, 2015.

[WLCS19] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

[XLKvdP20] Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. Allsteps: Curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, volume 39, pages 213–224. Wiley Online Library, 2020.

[YTL18] Wenhao Yu, Greg Turk, and C Karen Liu. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

A Accepted publications

Learning to Explore in Motion and Interaction Tasks

Miroslav Bogdanovic[1] and Ludovic Righetti[1,2]

Abstract— **Model free reinforcement learning suffers from the high sampling complexity inherent to robotic manipulation or locomotion tasks. Most successful approaches typically use random sampling strategies which leads to slow policy convergence. In this paper we present a novel approach for efficient exploration that leverages previously learned tasks. We exploit the fact that the same system is used across many tasks and build a generative model for exploration based on data from previously solved tasks to improve learning new tasks. The approach also enables continuous learning of improved exploration strategies as novel tasks are learned. Extensive simulations on a robot manipulator performing a variety of motion and contact interaction tasks demonstrate the capabilities of the approach. In particular, our experiments suggest that the exploration strategy can more than double learning speed, especially when rewards are sparse. Moreover, the algorithm is robust to task variations and parameter tuning, making it beneficial for complex robotic problems.**

I. INTRODUCTION

Deep reinforcement learning has attracted a lot of attention for robotic applications where full robot models can be difficult to identify, especially for contact dynamics, and lead to computationally challenging planning and control problems. In particular, it can be successful in producing robust behaviors with ability to handle uncertainties in the environment and quickly adapt to changes [1].

However, these algorithms suffer from several important issues that can limit their applicability. The most salient issue is related to sampling efficiency and exploration strategies. Indeed, exploration strategies required to generate new samples are often limited to simple noise models, which can drastically increase the number of required samples for policy convergence. This problem is especially acute in robotics, due to difficulties in obtaining large amounts of training data, discontinuities in the interactions with the environment, as well as complex, multi-part, potentially sparse reward functions. Quite often algorithms suffer from local minimums and flat surfaces in the reward space. In this work, we investigate a novel exploration strategy to alleviate these issues by leveraging previously learned tasks to better explore when learning novel ones.

One attempt to tackle the issue of exploration when applying deep reinforcement learning in robotics problems has been made in [2], by proposing the application of correlated noise. Correlated noise, more specifically Ornstein-Uhlenbeck (OU) process [3], is also used to improve exploration in [4]. While this exploration strategy can in theory explore the state space more rapidly, it tends to create very high changes in the control sequence while it would be preferable in robotic applications to have smoother motions with proper velocity profiles.

Exploration in reinforcement learning has been explored in more general settings. [5], for example, proposes exploration in the space of the parameters of the policy instead of the space of actions. Other approaches base exploration on some measure of novelty while moving through the state space, as is the case in research on intrinsic motivation [6], [7], [8]. These approaches mostly try to explore novel regions of the state space, but do not necessarily use knowledge from previously learned tasks. In this paper, we take a complementary approach where we leverage these previous tasks to generate better exploration strategies for novel ones.

Transfer learning, which seeks to exploit knowledge from previously learned tasks to accelerate learning new tasks is also relevant to our problem. In [9], authors propose learning a "distilled" policy from several tasks capturing the common behavior among them and constraining the individual policies to be close to it. The exploration process we learn bears some resemblance to this shared policy, the key difference being that in continuous action spaces, which is the setup we investigate, having such distribution only be dependent on the current state does not prove informative enough as we explain in detail in Section II-A. In [10], low level control on a locomotion system is learned and used to solve high level tasks. However, it requires that this type of separation exist as well as enough knowledge about it to be able to design a two-level structure with capability of learning it. Auxiliary tasks can also be used to improve learning when rewards are sparse, as in [11], however this requires an ad-hoc setup to define these tasks and how they should be interleaved with the main task to be learned. Our approach does not setup auxiliary tasks but still assumes that a set of tasks of increasing complexity is available. Our work also bears some connection to research on motion primitives [12], [13]. While we do not explicitly try to directly extract any motion primitive, the exploration process we learn can be thought of as generating basic action primitives.

In this work, we present a novel exploration strategy for deep reinforcement learning. We propose to learn a generative model of basic action primitives capturing the motion patterns seen in previously learned tasks. Extensive experiments on a set of simulated motion and contact

[1]Movement Generation and Control group, Max-Planck Institute for Intelligent Systems, Tübingen, Germany. Email: first.lastname@tuebingen.mpg.de

[2]Tandon School of Engineering, New York University, USA.

This work was supported by New York University, the Max-Planck Society and the European Unions Horizon 2020 research and innovation program (grant agreement No 780684 and European Research Councils grant No 637935).

interaction tasks for a robot manipulator demonstrate the capabilities of the approach. In particular, our approach shows significant learning speed-up compared to other state of the art algorithms, especially for tasks with sparse reward. We also show that the algorithm is robust to parameter changes and task variations, reducing the need for parameter tuning.

II. PROPOSED APPROACH

Our goal in this work is to use the knowledge we gain in solving several tasks with a given robot to facilitate learning new tasks for the same robot. In all the tasks we discuss in the paper we use the same robot and vary the tasks and the environments, with increasing complexity. More specifically, we want a method that uses good behaviors learned from previous tasks to help the exploration process when learning a new task. Our idea is to learn a function that generates random behaviors that resemble behaviors seen in previous tasks. In the following, we first describe our approach to build a model that generates random behaviors that retain similar characteristics as the behaviors learned in the previous tasks.

A. Learning the exploration model

We start by presenting a simple example illustrating the requirements for our approach and the difficulties inherent to learning good exploration strategies from previous tasks. In particular, we want to explain why longer trajectories need to be taken into account to learn proper exploration strategies. We consider a point mass moving in the plane with a fixed velocity magnitude. The control input consists in choosing a direction of motion, independently at each state. The desired task is to reach a desired location (B) from a randomly selected starting point in the plane (A) as shown in Figure 1(a).

We collect successful policies π_i for many instances of such tasks (i.e. for different goal positions). We then choose an arbitrary state s and evaluate the actions that all the different policies would take at that state $a_i = \pi_i(s)$ (Figure 1(b)). Considering that for each policy, the random goal positions can be distributed anywhere around the chosen state s, we can expect that given enough policies, we will be able to find a policy that would move in any chosen direction. Therefore, if we were to calculate a combined distribution π_c over the action space arising from all the policies:

$$\pi_c(a \mid s) \approx \sum_i \pi_i(a \mid s), \tag{1}$$

we would likely find a uniform distribution (i.e. no preferred direction of motion) which would be completely uninformative (Figure 1(c)). This illustrates that we cannot naively combine previous actions at a given state because the resulting distribution is likely to be of little interest to create sensible exploratory motions.

However, if we condition such distribution on a longer history of preceding states $s_{1:t}$, then there will exist only a few policies with a similar state history when arriving at the state (Figure 1(d)) and it is likely that their subsequent actions would be very similar. The combined probability distribution over actions:

$$\pi_c(a \mid s_{1:t}) \approx \sum_i p(s_{1:t} \mid \pi_i)\pi_i(a \mid s_{1:t}), \tag{2}$$

where $p(s_{1:t} \mid \pi_i)$ is the weighting factor equal to the probability of the policy π_i resulting in a state trajectory $s_{1:t}$, will be much more focused (Figure 1(e)). Taking consecutive samples from such a distribution would now result in a behavior with the characteristics of the original policies and would be significantly more useful for what we are trying to achieve.

We propose in the following to learn a combined probability distribution of a diverse set of policies conditioned on trajectories (or sequences) of preceding states. To represent this learned exploration model (LEP), we use a recurrent network, more specifically a Long Short Term Memory (LSTM, [14]) network. We start by collecting trajectories $\{(s_{1:T}, a_{1:T})_i\}$ from all the available policies. We note here that we only need trajectories and not the full policies for training the model, so our approach could work even if that is all we have access to (for example as a result of doing trajectory optimization [15], [16] or learning by demonstration [17]).

Importantly, we do not train our model on the full length of the collected trajectories, as we are interested in general basic characteristics of good behaviors, not behaviors that solve specific tasks. On longer time scales the characteristic associated to solving a specific task would become dominant, leading our network to overfit to solutions to individual tasks. Because of that we limit trajectory samples to a shorter timescale h and randomly sample sections of the trajectories of length h to build the training dataset.

The function approximated by the LEP network takes as input a history of robot states and outputs a distribution over the corresponding sequence of actions. We use diagonal Gaussian distribution as the output and train the network to maximize the log likelihood of the action sequences given the corresponding states (in the same way as in, for example, [18]).

B. Reinforcement learning with an exploration function

We now describe how the exploration model can be included in a reinforcement learning algorithm. In our experiments, we use Deep Deterministic Policy Gradient (DDPG, [4]), but any off-policy algorithm with independent noise could be used instead. DDPG is an actor-critic method that simultaneously learns a state-action value function $Q(s, a)$ and a deterministic policy $\pi(s)$ that optimizes it:

$$\pi(s) = \arg\max_a Q(s, a) \tag{3}$$

By keeping a constant estimate of the action with the largest Q value for each state it avoids the problem that arises in continuous action spaces, where calculating this value online requires solving an optimization problem.

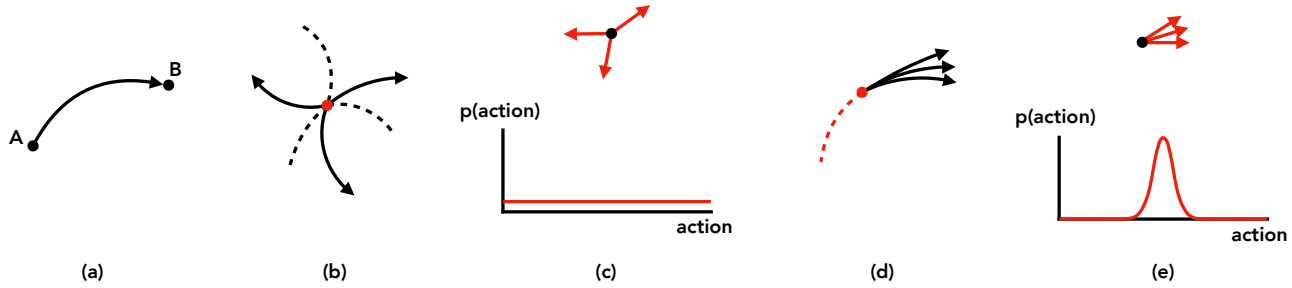


Fig. 1: Point mass example illustrating our approach (cf. Section II-A for details): (a) Task: moving from one point to another; (b) Policy trajectories conditioned on the selected state; (c) Combined action distribution conditioned on the selected state; (d) Policy trajectories conditioned on a state history; (e) Combined action distribution conditioned on a state history.

The full algorithm consists of interchanging steps of:

1) Gathering data by executing the current policy in the environment with an added output of an external noise process $\mathcal{N}$:

$$\begin{aligned} a_t &= \pi(s_t) + \epsilon \\ \epsilon &\sim \mathcal{N} \end{aligned} \tag{4}$$

2) Taking random samples from the gathered data and updating the value of the Q function using the Bellman equation and the policy π corresponding to the gradient of the current Q function estimate with respect to the action.

As we have seen, DDPG explores the spaces by adding exploration noise to an existing deterministic policy (Equation 4). We replace the exploration noise in DDPG with the output of the LEP (a generative model for motions with good properties). With that, the action that is taken at each time step during training is equal to the sum of current output of the deterministic policy and a sample from the exploration model:

$$a_t = \pi(s_t) + \epsilon_{LEP} \tag{5}$$

We reset the internal state of the LSTM network to its initial value every h steps, matching the sequence length it has been trained on. Thereby, the action distribution of the exploration is conditioned on the past $t \mod h$ states, where t is the current time step:

$$\begin{aligned} \epsilon_{LEP} &\sim p_{LEP}(a \mid s_{t_r:t}), \\ t_r &= t - t \bmod h \end{aligned} \tag{6}$$

Apart from this, we keep all the other aspects of the training exactly as they are in [4], including not reducing the exploration noise as the training progresses.

Remark 1: Note that while our model might produce sensible behaviors for the system there is no guarantee that the sum with output of the current policy will do the same. However, this simple approach works very well in practice. The DDPG policy is initialized to produce output values close to zero at the beginning of the training. Because of that, the initial samples in our case will for the most part be pure samples from the generative exploration model. As the policy changes during learning this might no longer be the case, but as we will later see in our experiments, the algorithm shows no trouble finely converging to the desired behavior.

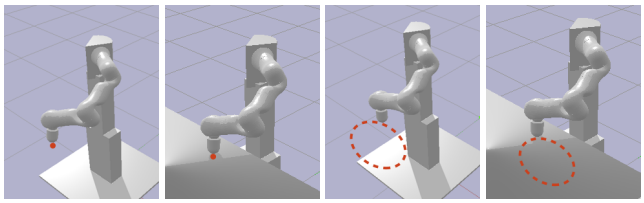


Fig. 2: Illustration of the four tasks tested in the experiments.

C. Continuous learning

One more aspect of our approach is its ability to be continuously applied, enabling solving of more and more complex tasks each time. As we have stated, we can use data from any source in training of the exploration model. This also includes policies resulting from applications of some previously trained exploration model of the same type. This way we can train the model on the data available to us initially, use it to learn on a new task, add the data from these new policies to our training set and then repeat the process. We can keep doing this as many times as necessary to get to a point where we can solve some complex task we are interested in.

III. EXPERIMENTAL SETUP

In this section we describe the experimental setup used to evaluate the performance of the approach.

A. Environment

We test the algorithm on a simulated KUKA LWR, a 7-DoF robotic arm. We choose tasks of increasing complexity and in particular tasks involving contact interactions with a table. All our experiments are implemented using the Bullet physics simulator. We keep the state and action spaces same across all the tasks. The state space consist of joint position, joint velocities and measured 3D contact forces at

the end-effector, for a total of 17 dimensions. The action space consists of torques applied to each joint, which is 7 control dimensions. We also add gravity compensation to the torque command as it is automatically added by the on-board controllers on the real KUKA LWR robot.

B. Motion and contact interaction tasks

Each of our different tasks is characterized completely by the defined reward function (and the presence or absence of the table in the environment, which we add in tasks with interaction aspects). We test four types of tasks of increasing complexity: a reaching task, a contact task, a periodic motion and a periodic motion while interacting with the table. We describe the tasks and the cost functions in detail in the following. Note that our cost functions are rather straightforward and do not specially seek to facilitate learning.

1) Task 1: Reaching a desired target: This task consists of getting the end-effector to a desired position and orientation in space. The reward function consists of two parts: the distance to the goal and the orientation error. In this task, the desired orientation always points straight down and we only vary the desired goal position.

2) Task 2: Stationary force application: The goal of this task is to apply a desired normal force on a desired location on the table. The reward function in this case will consists of three parts, the same two costs used in Task 1 for the position and orientation of the end-effector and a cost measuring the error between the desired and measured normal contact forces and adding a constant bonus term whenever the end-effector is in contact with the table to incentivize contact behaviors. Note that the contact cost is sparse as most robot configurations lead to no contacts.

3) Task 3: Periodic motion along a closed curve: Here we require the end-effector to move along a given circular path in space (while keeping a specified orientation). The reward function has three parts, based on position, velocity and orientation of the end-effector. The reward is based on the distance to the circle (as opposed to a fixed target location as in Task 1) and desired velocity based on the tangential velocity vector that the end-effector should have on the point on the circle currently closest to it. The orientation reward is kept the same as in previous tasks.

4) Task 4: Periodic motion with contact force regulation: Combining aspects of all the previous tasks, here the goal is to move along a given circular trajectory on the surface of the table while applying a constant normal force to it. Reward function is a combination of the trajectory reward given in Task 3 and the force reward used in Task 2.

C. Defining success during learning

It is not sufficient to find a high reward policy, we also need to check that the robot is indeed achieving the desired task. For each task, we empirically define a reward value for which we consider the task solved. In order to find this value, we analyzed many instances of the behavior on the task. We determine the value such that all behaviors with higher scores perform all the aspects of the task in a satisfactory way. For example, in Task 4, we make sure that policies performing only three out of the four aspects of the behavior we desire (moving along the trajectory without making contact with the table, applying force while being stationary on a single point on a trajectory, etc.) never reach this threshold for the cumulative reward.

IV. Results

We now present the results of our simulations. In particular, we demonstrate how tasks involving complex contact interactions can be learned efficiently with our approach. We also systematically compare the results with other state of the art reinforcement learning algorithms and test the robustness of the approach to random initialization. In all of our experiment, we compare our method with the normal DDPG algorithm and with an on-policy reinforcement algorithm, Proximal Policy Optimization (PPO) [19]. For both these algorithms, we use the implementations from OpenAI Baselines [20].

A. Same task type for training and testing

First, we evaluate how our method performs when it is trained on the same type of tasks it is later tested on. We investigate this behavior with the reaching task (Task 1), which is the simplest of all tasks. To generate the initial policies to train the exploration model we use PPO. The reason to use this algorithm instead of DDPG is that the vanilla DDPG (e.g. without a good exploration strategy) did not produce policies as good as PPO for this simple experiment. All subsequent training and improvement of the exploration model are done using data generated by our approach on previous tasks. We use 100 policies trained on instances of the task with varying goal positions for this initial data collection.

We generate new instances of the same task with different goal locations and compare the performance of our approach (DDPG + LEP) with DDPG and PPO. We explore various noise setups for DDPG (Gaussian noise and Ornstein-Uhlenbeck processes, with a range of values for the variance), as well as different subsequence lengths for training of the LEP. We present results for the best configuration for each algorithm averaged over 6 different goal positions in Figure 3.

We notice that all the algorithms converge to a good behavior (above the gray line), which is expected for this simple task. PPO, being an on-policy algorithm, converges noticeably slower than the other two. While standard DDPG implementation and the one using our exploration process both converge relatively quickly, the average for our approach reaches the desired task value more than two times faster than the one using standard noise and has a visibly higher percentage of satisfactory solutions at the end. While for this task the comparison can be biased as our RNN was trained on other instances of the same task (i.e. other desired positions), it is important to notice that our approach does not exhibit bias towards goal positions seen in the training

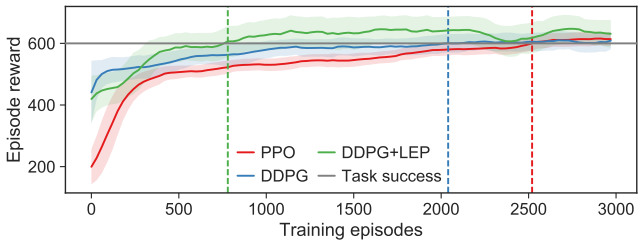


Fig. 3: Learning results of the best performing policies for Task 1: average cumulative reward (bold lines) and variance (shaded area) across all task instances. Our method (green) converges more than twice faster than the other algorithms (vertical dashed lines).

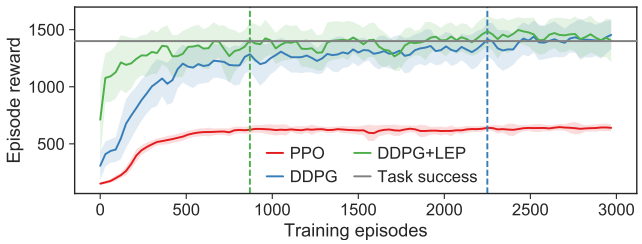


Fig. 4: Learning results of the best performing policies for Task 2: average cumulative reward (bold lines) and variance (shaded area) across all task instances. Our method (green) converges more than twice faster than DDPG (vertical dashed lines), PPO is not able to find a solution.

set and manages to converge to the desired target with as good or better accuracy than the other two methods.

B. Application to a new task and dealing with sparse reward

Next, we investigate how our approach performs on a previously unseen task. We are also interested in how the approach deals with a sparse reward signal. We first use Task 2 with the same exploration process trained in the previous experiment, using data from policies reaching random points in space. We expect that our exploration model contains more informative motions for the end-effector resulting in a speedup in learning despite the very different nature of the tasks the RNN was trained on. Indeed, this task contains a reaching component which was seen before and a contact regulation component that was not seen in Task 1. As before, we compare our approach with regular DDPG and PPO. The results, again averaged over 6 different task instances, are shown in Figure 4.

Unlike the previous set of experiments, in this task PPO is not capable of finding a policy that achieves the desired behavior. The reward information about force interaction in this case is very sparse and is only present when the end-effector is in contact with the table. Even though the position part of the reward guides the policy to such states, a broader exploration is then required to find rewarding types of interaction. PPO, lacking this, fails to find solutions for the task and optimizes only for the position and orientation parts of the reward. The other two approaches both converge to

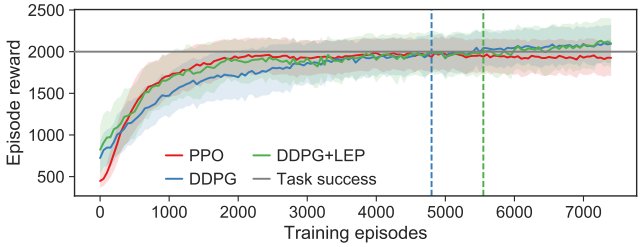


Fig. 5: Learning results of the best performing policies for Task 3: average cumulative reward (bold lines) and variance (shaded area) across all task instances. Our method (green) converges slightly slower that the normal DDPG (vertical dashed lines), PPO is not able to find a solution.

satisfactory behaviors, but again our methods does so more than twice as fast as the normal DDPG algorithm.

We repeat this experiment with Task 3 requiring a motion along a closed curve in free space. This task requires a periodic motion, which is inherently different than reaching motions in terms of expected position and velocity profiles. It is therefore an inherently more difficult task for our exploration model, but one with a non-sparse reward, providing guiding information throughout the state space. Here we vary the circle radius and center for each new task instance. Results, in this case averaged over 18 different instances, are shown in Figure 5.

Our approach and DDPG take a significantly longer time to converge than in the previous tasks. While PPO never reaches the threshold for which we consider that the appropriate behavior is achieved, its final behavior is not very far from being acceptable. We notice that all three algorithms converge similarly, with the normal DDPG being initially slower. This experiment suggests that for tasks that require very different movement profiles than the movements our exploration model was trained on, our exploration method will not necessarily significantly improve convergence, yet it is still not detrimental to the learning process, which is an important aspect to afford generalization to other tasks.

From these two experiments we can see that, as expected, the advantage for using our approach comes when the reward information is sparse and simple exploration is no longer sufficient, however the required basic movement profiles need to share similar characteristics in order to benefit from a significant speedup.

C. Complex interaction task and continuous learning

In the last experiment, we would like to demonstrate other important aspects of our approach: that it can scale to significantly more complex tasks and that the exploration model can be extended with previously learned motions, allowing continuous learning of richer exploration strategies as we discussed in Section II-C. To do so, we update our exploration model by training it with a combination of data from Task 2 and Task 3. We collect 100 policies from each of the two tasks.

For testing we use Task 4, which requires concurrent force

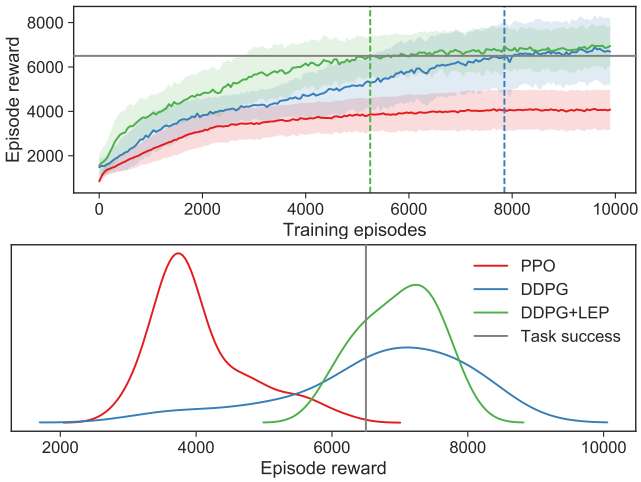


Fig. 6: Learning results of the best performing policies for Task 4. (Top) average cumulative reward (bold lines) and variance (shaded area) across all task instances. (Bottom) Final reward distribution.

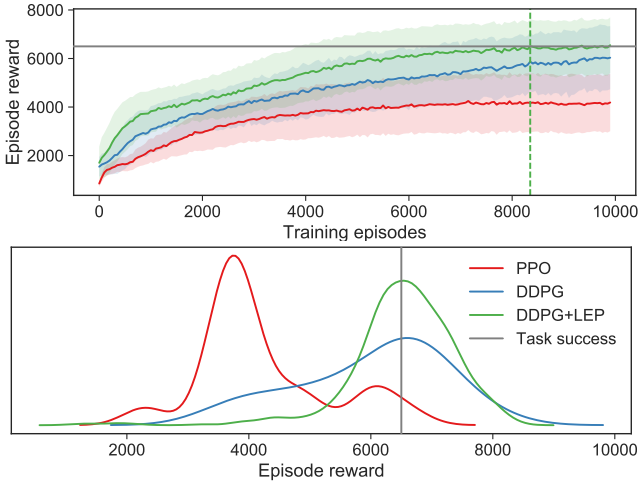


Fig. 7: Robustness to parametrization results for Task 4. (Top) average cumulative reward (bold lines) and variance (shaded area) across all task instances and all parametrizations. (Bottom) Final reward distribution.

regulation and motion along a circular path on the table and contains aspects from all previously encountered tasks. It means that we could decompose the task as a combination of all previous three tasks (Task 1 to reach the table, and Task 2 and 3 to perform the motion on the table). First, we compare our approach with other algorithms in the same way as before. We still use the best performing parametrizations of each algorithm to compare the results and in this case use 50 different task instances for the comparison (Figure 6). With the table interaction aspect again present, PPO fails to find satisfactory solutions. Both standard DDPG and the one using our exploration process do converge, but with our method doing so almost twice as fast. In addition to the learning curves, Figure 6 also shows the full distributions of the cumulative rewards for the policies for all three algorithms at the end of learning. This figure clearly suggests that our approach both has a higher percentage of satisfactory solutions as well as practically no policies with really bad scores. This is in contrast with PPO which finds mostly poorly performing policies and DDPG that has a more elongated distribution of results: for certain task instances it finds solutions but fails to do so for others. These results additionally support that our approach can speed-up learning but more importantly, it suggests that learning performance becomes more consistent and repeatable across task instances.

D. Robustness

Finally, we would like to demonstrate robustness and the lack of need for tuning parameters in our approach. This is especially important because oftentimes reinforcement learning algorithms are very sensitive to parameter tuning and the same experiments with random initial conditions for the system can lead to very different results [21].

In all previous experiments, we presented results using the best parametrization for each algorithm (between different noise parameters for standard DDPG and different subsequence lengths used for training the LEP). Now we present results for learning Task 4 for all the parametrization we tested, without making any such choices. The results are presented in Figure 7. The difference between each algorithm is very clear. The results for our method are only slightly worse than those we presented for the best parameter configuration, demonstrating its robustness to parametrization. The average policy performance at the end of the training especially is only slightly affected. Without any tuning our method still produces a majority of satisfactory policies for this complex task. That is not the case with either standard DDPG or PPO. DDPG becomes significantly worse in this evaluation, with the average not reaching satisfactory value at the completion of the training. PPO was already not performing the tasks with the best parametrization. These results support the idea that our exploration strategy can not only speed-up learning, but also improve the robustness of the algorithms to parameter tuning, which can be a significant gain when deploying such algorithms on novel tasks and robots.

V. DISCUSSION

The goal of learning the exploration process is to make the behaviors needed to solve a new task more likely to occur during exploration. This is done in an effort to speed up learning, as well as to achieve complex behaviors that might otherwise be missed. In the absence of exploration capable of doing so, we are usually forced to add guiding terms to reward functions to lead the policies in desired direction during training. Such terms not only require additional tuning, but are also not representative of actual aspects we would like to achieve. As tasks become more and more complex, with many-part reward functions, the process of adding this guiding information becomes untenable. This is why we aim

to relieve some of this effort by having a good exploration process, freeing up the reward function to just encode the task at hand.

Some of the main questions to be considered when using an approach like the one we present in this work are related to ways in which data from one task can be useful in solving a different one. Here, that is reflected in the choices we make in selecting data to train the exploration process, as well as more generally how we structure a curriculum of tasks with a goal of generating more and more complex behaviors on the system.

First, it is worth pointing out that using all the data we have access to, even if it is extensive and varied, might not necessarily be a bad idea. Barring issues with the model not being able to fit the data correctly, the only downside would be that our model encodes a wider distribution, covering behaviors that might not be directly useful for the current task. In that case, some of the exploratory behavior might not be relevant, but that should not prevent us from gaining benefits from the rest of it.

In the same way having too much data might not cause issues, lacking data for some part of the behavior needed to solve the task might not be detrimental either. Using an exploration model trained only on behavior needed for one aspect of the task will not necessarily prevent us from learning how to solve all the other aspects as well. For example, as can be seen in Figure 4, using an exploration process trained only using free space motions causes no issues in learning on a task where force also needs to be applied. What is more, learning is significantly faster on the new task than it is with the standard methods.

Taking the two previous points into account we still want to make the best choices we can in building an exploration process with a goal of solving a new task. For achieving that we should take a look at what kind of motion and interaction behavior is expected in the new task and choose already known tasks exhibiting some parts of that behavior. The goal being to decompose the task into as many elements that are already contained in some of the known policies. The same idea applies when building an entire curriculum of tasks to solve on a system, where we should start from the ones easiest to solve and slowly add new aspects as we progress in generating more and more complex behaviors.

VI. CONCLUSION

In this paper, we presented a novel approach to learn an exploration process for reinforcement learning using previously learned tasks. The system is built such that as novel tasks are learned, the exploration model can be improved and facilitate learning more complex tasks. This is particularly useful for robotics problems where such hierarchy of tasks (from simple to complex) naturally arises. In our future work we intend to demonstrate the learned policies on a real robot and extend the approach to more complex manipulation tasks.

REFERENCES

[1] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[2] P. Wawrzynski, "Control policy with autocorrelated noise in reinforcement learning for robotics," *International Journal of Machine Learning and Computing*, vol. 5, no. 2, p. 91, 2015.

[3] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Phys. Rev.*, vol. 36, pp. 823–841, Sep 1930. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.36.823

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015. [Online]. Available: http://arxiv.org/abs/1509.02971

[5] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," *CoRR*, vol. abs/1706.01905, 2017. [Online]. Available: http://arxiv.org/abs/1706.01905

[6] A. G. Barto, "Intrinsically motivated learning of hierarchical collections of skills," *International Conference on Developmental Learning and Epigenetic Robotic*, pp. 112–119, 2004.

[7] A. Laversanne-Finot, A. Péré, and P.-Y. Oudeyer, "Curiosity driven exploration of learned disentangled goal spaces," *arXiv preprint arXiv:1807.01521*, 2018.

[8] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-Driven Exploration by Self-Supervised Prediction," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, pp. 488–489, 2017.

[9] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," *CoRR*, vol. abs/1707.04175, 2017. [Online]. Available: http://arxiv.org/abs/1707.04175

[10] N. Heess, G. Wayne, Y. Tassa, T. P. Lillicrap, M. A. Riedmiller, and D. Silver, "Learning and transfer of modulated locomotor controllers," *CoRR*, vol. abs/1610.05182, 2016. [Online]. Available: http://arxiv.org/abs/1610.05182

[11] M. A. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. V. de Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing - solving sparse reward tasks from scratch," *CoRR*, vol. abs/1802.10567, 2018. [Online]. Available: http://arxiv.org/abs/1802.10567

[12] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[13] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] I. Mordatch, Z. Popović, and E. Todorov, "Contact-Invariant Optimization for Hand Manipulation," in *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, 2012, pp. 1–8.

[16] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On Time Optimization of Centroidal Momentum Dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia: IEEE, May 2018, pp. 5776–5782. [Online]. Available: https://arxiv.org/abs/1709.09265

[17] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," *Springer handbook of robotics*, pp. 1371–1394, 2008.

[18] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[20] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.

[21] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "How many random seeds? statistical power analysis in deep reinforcement learning experiments," *arXiv preprint arXiv:1806.08295*, 2018.

Learning Variable Impedance Control for Contact Sensitive Tasks

Miroslav Bogdanovic[1], Majid Khadiv[1] and Ludovic Righetti[1,2]

***Abstract*—Reinforcement learning algorithms have shown great success in solving different problems ranging from playing video games to robotics. However, they struggle to solve delicate robotic problems, especially those involving contact interactions. Though in principle a policy directly outputting joint torques should be able to learn to perform these tasks, in practice we see that it has difficulty to robustly solve the problem without any given structure in the action space. In this paper, we investigate how the choice of action space can give robust performance in presence of contact uncertainties. We propose learning a policy giving as output impedance and desired position in joint space and compare the performance of that approach to torque and position control under different contact uncertainties. Furthermore, we propose an additional reward term designed to regularize these variable impedance control policies, giving them interpretability and facilitating their transfer to real systems. We present extensive experiments in simulation of both floating and fixed-base systems in tasks involving contact uncertainties, as well as results for running the learned policies on a real system (accompanying videos can be seen here).**

***Index Terms*—Reinforcement Learning; Compliance and Impedance Control; Motion Control.**

I. INTRODUCTION

MANY interesting robotic applications necessitate complex physical interactions with the environment. During locomotion, intermittent contacts and force modulation enable the robot to keep balance and move forward. Multi-contact interactions are also central to the efficient manipulation of objects. Establishing and breaking contact is especially hard because it causes a switch in the dynamics of the system which can rapidly lead to failures if not controlled properly. Unforeseen changes in the contacts location and properties (friction, stiffness, etc) can also dramatically degrade the robot behavior and remain a fundamental challenge in robotic manipulation and locomotion.

Deep reinforcement learning has shown a lot of promise in recent years for robotic applications. However, in an effort to learn end-to-end policies the focus has often been on the complexity in the observation part of the task, specifically vision, and not necessarily on the physical interaction part of the problem. One important aspect, that we investigate in the paper, relates to the choice of a policy parametrization that affords efficient learning of policies robust to contact uncertainties.

It was shown that position control with fixed pre-tuned gains can have better learning performance than pure torque outputs [1], and such policies have been successfully transferred to physical robots interacting with the environment [2]. These results are achieved by guiding properly the exploration using desired position and stabilizing the system around that using pre-tuned feedback gains. However, previous work has demonstrated the importance of some form of force control when learning interaction tasks, either explicitly [3] or by learning time-varying control gains [4], [5], [6], [7], state-varying feedforward and feedback gains [8], or unified motion and gains varying strategy [9], [10], [11].

Recent results [11], [12] suggest that learning impedance schedules in task space can significantly speed up learning of manipulation tasks. The operational space representation used in those works has the advantage of abstracting the robot kinematics, but with the potential drawback of fixing the redundancy resolution scheme which can limit the range of possible behaviors. Indeed, the ability to vary nullspace resolution schemes is critical to enable the concurrent execution of several tasks necessary to achieve complex behaviors, e.g. avoiding an obstacle while reaching for an object or taking a step while maintaining balance [13], [14]. One can also argue that methods that require solving an inverse problem suffer from numerical instability near singularity, or rule out a significant space of possible motions achievable without predefining a task-space. Moreover, there is evidence that the best choice for a task-space may vary across and within tasks [15]. For these reasons, in this paper we focus on joint space policy learning despite potential training speedup that can be achieved by doing the same in a predefined task space.

The main goal of this paper is to investigate the effect of policy parametrization on reinforcement learning for robotic tasks involving complex contact interactions and hard impacts. We provide empirical evidence that control policies concurrently generating desired positions and joint impedance tend to produce more robust behaviors. We present both extensive numerical simulations and real hardware experiments. In particular, we find that the resulting policies are robust to various types of contact uncertainties (friction, stiffness and contact location). Additionally, we propose a reward term regularizing these variable gain policies and giving them interpretability, allowing for direct transfer to a real robot. We perform an extensive analysis on two very different systems: a single-

Manuscript received: February 24, 2020; Revised June, 1, 2020; Accepted July 2, 2020.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by New York University, the European Union's Horizon 2020 research and innovation program (grant agreement No 780684 and European Research Council's grant No 637935) and a Google Faculty Research Award.

[1]Max-Planck Institute for Intelligent Systems, Tübingen, Germany {mbogdanovic, mkhadiv, lrighetti}@tue.mpg.de

[2]Tandon School of Engineering, New York University, USA

Digital Object Identifier (DOI): see top of this page.

leg hopper (floating-base) creating intermittent contacts with hard impacts on the ground and a manipulator (fixed-base) performing a delicate force control task. In both cases we show that variable gain control outperforms a wide range of learned fixed gain or direct torque control policies, especially in the presence of contact uncertainty.

II. CONTROL POLICY PARAMETRIZATION

In this work, we compare several ways to parameterize control policies when using reinforcement learning in robotics tasks. For each examined control parametrization we represent the policy with a neural network. The inputs of the network stay the same in each case (the state of the system, $\boldsymbol{\xi}$), while the output corresponds to the parameters of the individual controller (torque, desired positions, gain parameters, etc). Based on these parameters the controller produces the torque, $\boldsymbol{\tau}$, that is then applied to the system. We now present each of the controller parametrization that will be examined.

Direct torque control. In the first parametrization, the neural network directly outputs desired torque commands (Figure 1(a)) and the resulting control law is simply:

$$\boldsymbol{\tau} = \boldsymbol{\tau}(\boldsymbol{\xi}) \tag{1}$$

This parametrization imposes no structure, which provides some benefits. As there is no imposed structure, any control that is a function of the variables present in the state can be expressed using this parametrization, given that the neural network is capable of approximating it. Additionally, such control parametrization may have a more direct control of interaction forces than position controllers and may provide benefits in that respect, at least in static contact situations.

However, having no imposed structure also has several downsides. With this parametrization, the function to be learned may be unnecessarily complex. For example, for a motion task, the policy needs to learn something akin to PD control as well as a part that generates desired positions, all inside one function. This representation does not explicitly separate the feedback (how to correct from errors) from the feedforward (what is the desired behavior) pathways. Further, small variations in commanded torques can lead to very different movements and therefore to different task costs. This will also raise issues for generating meaningful exploration, which might be more difficult than doing the same in the space of desired joint positions.

Fixed gain PD control. The second parametrization considers a PD controller with fixed gains. In this case, the policy outputs desired joint positions and torques are computed using a PD controller with some pre-defined gains (Figure 1(b)). The control law is then

$$\boldsymbol{\tau} = \boldsymbol{K_p}(\boldsymbol{q_{des}}(\boldsymbol{\xi}) - \boldsymbol{q}) - \boldsymbol{K_d}\dot{\boldsymbol{q}} \tag{2}$$

Again, there are positive and negative aspects to such a parametrization. It is often easier to find solutions in this setup [1], due to a simpler action space to explore and a policy easier to encode. Indeed, small variations in desired positions lead to small variations in task execution, at least in contact-free motions. It is in fact, as we will also see in our experiments later, the best choice for tasks involving only free space motions.

On the other side, achieving a desired behavior in interaction with the environment becomes more difficult, especially when uncertainties are present, even when the gains are well tuned for the specific task. The policy can to some extent control the interaction with the environment by changing desired joint positions, but as we will see later, finding such solutions becomes increasingly difficult.

Variable gain PD control. The last parametrization is a PD controller with variable gains, i.e. the policy modulates both the desired position and impedance of each joint (Figure 1(c)). The control law is written as

$$\boldsymbol{\tau} = \boldsymbol{K_p}(\boldsymbol{\xi})(\boldsymbol{q_{des}}(\boldsymbol{\xi}) - \boldsymbol{q}) - \boldsymbol{K_d}(\boldsymbol{\xi})\dot{\boldsymbol{q}} \tag{3}$$

Note: In our experiments, we use a single output to control both $\boldsymbol{K_p}$ and $\boldsymbol{K_d}$ by imposing a fixed relationship between damping and stiffness, similar to the scaling of critical damping. The neural network outputs the $\boldsymbol{K_p}$ gain and $\boldsymbol{K_d}$ is varied with the square root of that value.

In contrast to the previous two controllers, we have introduced an extra degree of freedom (the gain modulation) for each joint. In theory, this added degree of freedom allows for an explicit separation between the feedforward path, i.e. the desired behavior encoded in the desired position, and the feedback path, i.e. the response to unforeseen events encoded in the feedback gain.

This parametrization will preserve the ease of exploration characteristic of PD control with fixed gains. Moreover, with additional control dimensions to use, the functions the policy needs to learn may become even simpler than in the pure position control case. Finally, key to contact sensitive tasks we are particularly interested in, the policy has finer control over contact interactions with the environment. Robustness to environment uncertainty might be easier to encode in the feedback path while preserving the feedforward one to encode the ideal, unperturbed, behavior.

An obvious drawback is that we doubled the size of the action space the policy is acting in. But, as we will see later, this rarely causes loss in learning performance.

Remark: Both fixed and variable gain PD control we discuss in this work are different than the ones commonly used in robotics, where the control follows a predefined time-based trajectory [2]. All controllers we examine are state-based without any notion of time, and as such are capable of handling uncertainty in contact location and time.

III. EVALUATION PROCEDURE

Evaluation goals. In our evaluations we focus on setups where proper interaction with the environment is crucial for task success. In contact sensitive problems, planning and optimization-based approaches often struggle and reinforcement learning has the potential to generate solutions which cannot be easily found otherwise.

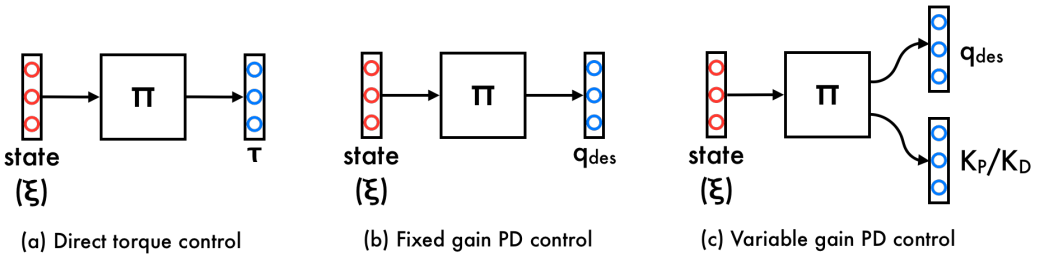
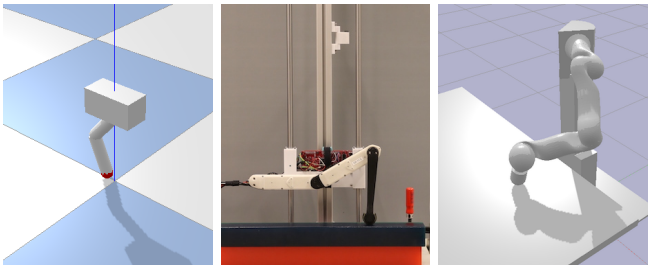


Fig. 1: Structures for the three control policies used.



(a) (b) (c)

Fig. 2: Evaluation environments used: A floating-base system: a hopper jumping on a surface in simulation (a) and on real hardware (b); A fixed-base system: a manipulator interacting with the environment in simulation (c).

A central aspect of this work is to evaluate how controller parametrizations influence task performance in presence of different contact uncertainties in the environment. This is an important aspect in order to generate motions that can transfer to real physical systems. To be able to successfully transfer policies from simulation to real systems we need to find solutions that are capable of handling variation in critical environment parameters. Therefore, the parametrization best able to find solutions in such cases is more likely to produce good results when applied on the physical system.

Evaluation environments. To study these aspects, we use two different environments, a single leg hopper which is a floating-base system and a manipulator fixed to the ground. We seek to show that our results are consistent across two very different environments, with contact interactions of completely different nature. The manipulation task requires fine interaction while the hopping task requires soft, reactive landing and quick force exertion. Both contain movements that alternate between free-space motions and contact interaction phases. The simulations are implemented in the PyBullet simulator [16]. We also show direct transfer of the learned policy for the hopping environment to the real system.

Reinforcement learning algorithm. For training the control policies we use Deep Deterministic Policy Gradient (DDPG, [17]). We chose an off-policy algorithm to reduce the issue of local minima, especially present here arising from combination of learning to control in joint space, discontinuities in the dynamics arising from contact interaction, and complex, multi-part reward functions. However, we do not make use of any particularities of DDPG in our approach. We therefore expect that the results we present here will remain consistent when using other learning algorithms.

IV. CONTROLLING A FLOATING-BASE SYSTEM

A. Task description

Setup. The first setup we use in our evaluations consists of a floating-base robot hopper with a two degrees of freedom leg [18] and a solid surface beneath it. We restrict the base to only move along the Z-axis which eliminates the falling down effect while still capturing the base motion and intermittent contacts during continuous jumping.

Task. The task is to achieve stable periodic hopping motions. We penalize hard impacts on the ground, as it is not something that would be acceptable on the real system. We are interested in motions where the system smoothly lands and pushes off, without any discontinuities in its velocity. To produce policies that are robust to contact switch uncertainty we randomly change ground surface height during episode execution in a range between $-5\,\mathrm{cm}$ and $5\,\mathrm{cm}$. This corresponds to ground variations of 31% of the total hopper height in the fully stretched configuration ($32\,\mathrm{cm}$).

We set the state of the system to consist of the joint positions and velocities for the two leg joints as well as position and velocity of the base. We do not explicitly provide to the system any information about contact.

Reward function design. To generate hopping motions we intentionally keep the reward function as simple as possible. The main part of the reward is based on the height of the robot base at every timestep, with an increase for values that cannot be reached without leaving the ground. This term, on its own, is enough to produce consistent hopping motions. However, regardless of the controller design, policies trained on such a reward produce exactly the excessive impacts on the ground we are looking to avoid. In order to prevent impact forces that can produce damage on the real system we penalize large forces applied to the robot. Finally, to avoid high frequency control command we introduce a torque smoothness penalty.

Even though in this case we are dealing with a comparably simple system, this reward design creates a challenging learning problem. It is relatively easy for policies to get stuck in a local minima where the system is just held upright with its leg fully extended and not reach any hopping motion in their exploration. The addition of the large force penalty makes the problem even more difficult as initial hopping motions found during training are bound to result in penalty for bad landings larger than the reward received for jumping.

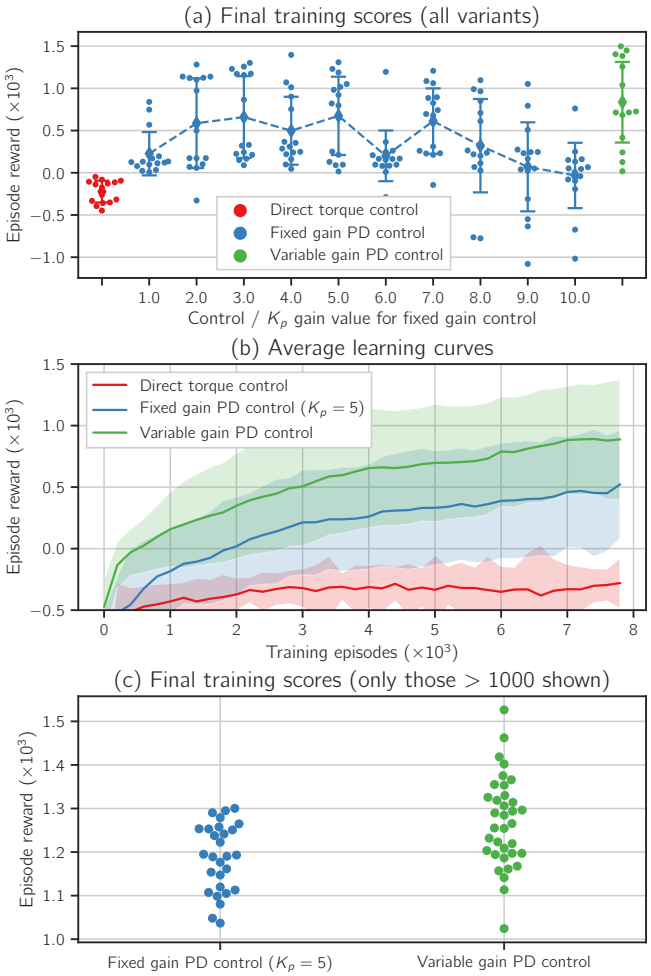


Fig. 3: Results for training in simulation on the hopping environment: (a) Final training scores for all controller parametrizations. For the fixed gain policy results are shown for a wide range of gain values; (b) Averaged learning curves for the three controller parametrizations. Results are averaged over 100 training instances for each parametrization; (c) Closer look at the best performing policies at the end of the training (only those with score greater than 1000).

B. Simulation results

Quantitative comparison. We run several training instances for each of the three controller parametrization. For fixed gains control we examine performance over a range of different gains. In Figure 3(a) we show final scores at the end of each training. We can see that direct torque control policies completely fail to achieve the task. For fixed gain control we can observe a drop in performance when gains are too low and when they are too high, with a medium value of $K_p = 5.0$ having the best average performance.

For reliably comparing fixed and variable gain control on this task we repeat the experiment using 100 training instances for each controller parametrization, using the best performing gain value for fixed gains control. In Figure 3(b) we present learning curves averaged over all the experiments for each controller parametrization. We can see that average performance of variable gain control exceeds that of fixed gain one, a result of more of those policies converging to adequate solutions as well as doing so faster.

Being more likely to converge is important, as it allows us to find policies we can use in a limited number of training runs. However, at the end we are always going to pick the best performing policy to deploy on the real system. We are therefore interested in understanding how good the *best policies* are. In Figure 3(c) we show the individual final performances from the previous experiment for all the policies above a score of 1000. Here we can see that it is not only the case that the variable gain policies are more likely to converge or that they do so faster on average, but that they find solutions strictly better than any that are found with the fixed gain parametrization.

Qualitative analysis. In Figure 4 we show sample of behavior of the best performing variable gain policy. Examining the gain profiles, we can note that they go to lowest possible value in advance of making contact, before spiking up to allow the system to first slow down, and then push itself off the ground. The torque values stay close to zero except during landing and push-off when they go to maximum values that can be exerted. The contact interaction is such that the impact is absorbed and the force is smoothly applied to first slow down and then accelerate the system upwards, without losing contact at any point.

C. Experiments on the real system

Transferring policies to real systems. When learning in simulation the policies can learn to exploit idealized rigid-body dynamics. Indeed, simulations do not include robot flexibility or difficult to model dynamic effects such as Coulomb friction, drive train dynamics, etc. Furthermore, they typically assume infinite bandwidth control authority, without any delays, which is known to be an issue to compute optimal policies [19], [2]. As a result, optimized policies can quickly change control outputs which results in very good behavior in simulation but excites the dynamics of the real robot in problematic ways, for example creating unwanted and possibly destabilizing oscillations in the motion. This is an effect that we observed in our initial experiments.

An approach often taken in sim-to-real research is to randomize robot parameters in the simulation to capture various types of unmodeled dynamics [20]. However, there is no guarantee that this randomization will capture the unmodeled dynamics of interest since the simulation does not explicitly capture this dynamics. Further, dynamic randomization might prevent finding appropriate solutions that could transfer to the real robot by generating samples that do not apply to the real robot. In this paper, we explore a different approach that exploits our control parametrization.

Trajectory tracking reward term. To address the aforementioned issue, we introduce an additional reward term during learning of the policies in simulation, which aims to force the policy to generate desired positions that can be effectively tracked. The reward term penalizes the difference between the desired position given by the policy at time step

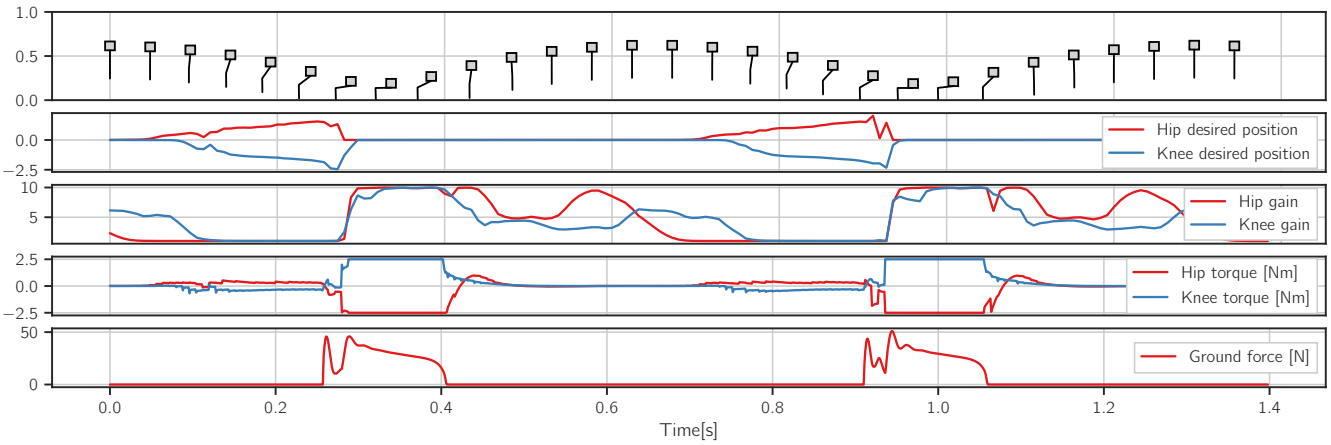


Fig. 4: Example of a learned behavior of the variable gain policy in simulation. The gains are at minimum just before landing and then increase quickly to enable push-off.

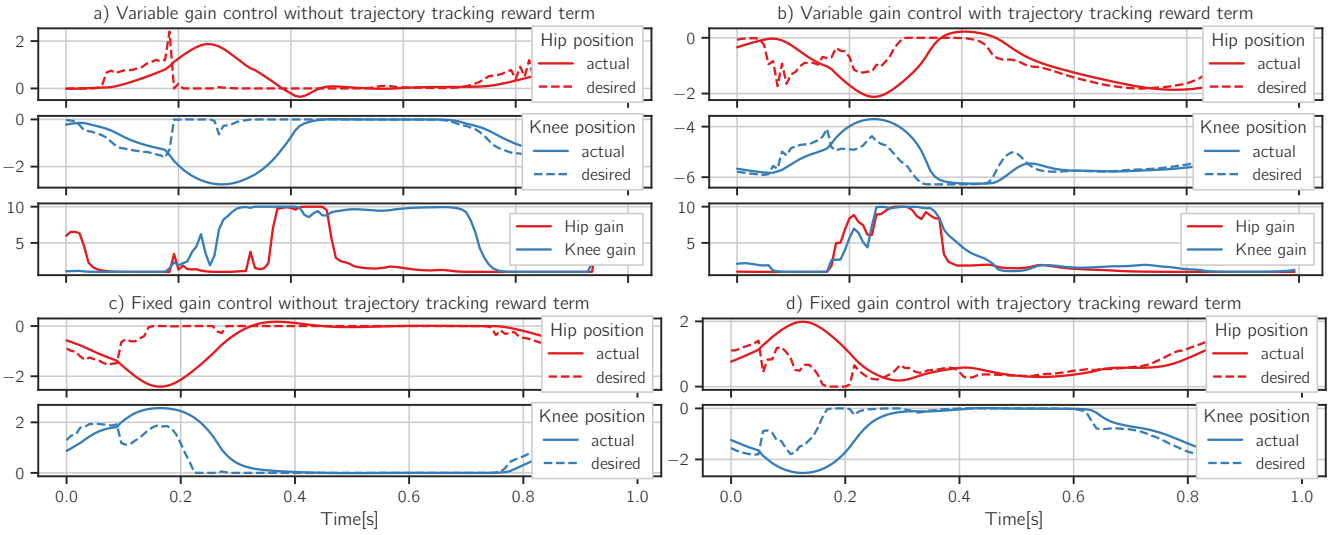


Fig. 5: Effect of introduction of the trajectory tracking reward term on outputs of fixed and variable gain policies: (a), (b) Comparing the effect on resulting trajectory and gains for the variable gain policy; (c), (d) Comparing the effect on resulting trajectory in the case of fixed gain policy.

t, q^t_{des} and the actual position achieved in the next time step $t+1$, q^{t+1} as follows:

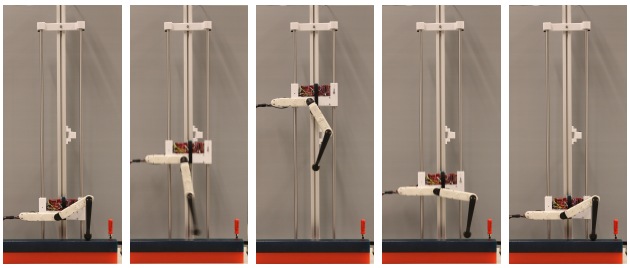
$$r_{tt} = -k \left\| q^t_{des} - q^{t+1} \right\|^2$$

With this reward term, we favor desired joint positions that can be tracked by the closed loop system in simulation. Without such a reward term, the variable gain policy can make a choice between various desired position and gain pairs that results in the same applied torque at each time step. For example, instead of generating desired positions that can be meaningfully executed, it can just give desired position values far from the actual robot trajectory with smaller gains to achieve the same torques and therefore the same behavior. However, this would lead to behaviors that could be very sensitive to variations in the dynamics. As an additional benefit, this term forces the policy output to remain interpretable as desired position and feedback gains, clearly separating feedback and feedforward control paths.

We should note the difference between this term and a term penalizing desired positions that move away from the *current* state of the system. In the second case any desired position different than the current one receives some penalty and the system is incentivized not to move. With the reward term we propose here if the position is reached in the next time step, zero penalty is given. Even in the case when the desired position is not reached, penalty is only given on the remaining distance to it. Only the desired values that cannot be reached are penalized, all motions where the trajectory can be tracked receive zero penalty.

In order to evaluate the effect of this addition to the reward on the resulting policies, we repeat the training process described previously with the trajectory tracking penalty enabled for both fixed and variable gain policies. We find that the best scoring policies for both these controller parametrizations still generate hopping with similar performance.

The outputs of both policies, with and without trajectory



(a) (b) (c) (d) (e)

Fig. 6: Example of a variable gain policy behavior on the real system: (a) Push-off position on the ground; (b) Pushing off the ground; (c) Reaching maximum height; (d) Getting into position in preparation for the landing; (e) Landing and starting the cycle again.

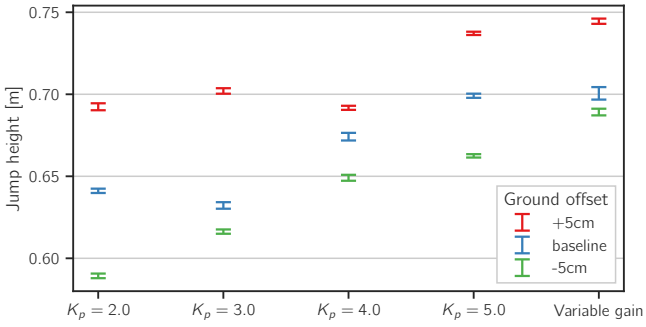


Fig. 7: Jumping heights for fixed and variable gain policies deployed on the real system. We use three ground positions where the baseline is the normal ground height and the other two ($-5\,\mathrm{cm}$, $5\,\mathrm{cm}$) are the ends of the range over which the policies are trained.

tracking reward, are shown in Figure 5. We can see that before adding this penalty term the variable gain policy gives desired position output far from the current trajectory when realizing the force needed to push off from the ground. However, when training with this new reward term the desired positions for both joints track the actual trajectory more closely. We can see that this also results in interpretable gain results – going as low as possible before contact and then spiking up to realize the force that is needed. On the other hand the output of the fixed gains policy practically does not change at all (the plots differ as a result of the opposite knee orientation found between the two solutions). When gains are fixed, the only manner to control a desired contact force is to go off the trajectory with the desired position values. On the other hand, the resulting force can be controlled by varying the gains in the other parametrization.

Evaluating policies on the real system. We evaluated the best performing policies in simulation for each gain value for the fixed gain controller and the one best performing variable gain policy. Torque control did not produce any sensible policy worth evaluating on the real system. For the fixed gain policies, having position gain values of $K_p = 1.0$ was insufficient for hopping so that is omitted from the following comparison. Gain values of $K_p = 6.0$ and higher resulted in unstable behavior so those are omitted as well.

Apart from this, the transfer of policies trained in simulation in the way we described to the real system is very robust. Out of the 6 policies we evaluated all succeeded in reasonable jumping performance, for many consecutive jumps, on the first try. The achieved jump heights are presented in Figure 7.

Importantly, the variable gain policy outperforms all of the fixed gain ones, i.e. it is the policy that saw the least decrease in performance. Moreover, the resulting jumping height is higher than what was demonstrated using model-based control in [18]. It is in fact the highest jumping we were able to generate on this platform to date. Also, even though its gains can, and do, go as high as $K_p = 10.0$ it shows none of the instability present in the fixed gain policies with those same gain values. The reason for this being that it increases the gains to maximum values only when necessary (e.g. during push-off). Indeed, the maximum admissible gains in contact will always be larger than during the flying phase due to the changing reflected inertia at the joints in loaded/unloaded conditions.

V. CONTROLLING A FIXED-BASE SYSTEM

A. Task description

Setup. We use a simulation of a 7 degree of freedom KUKA LWR manipulator (Figure 2(a)). In all the simulations, gravity is compensated with a feedforward term, which is the default behavior on the real robot.

Task. The task we are interested in consists of doing a circular motion with the endeffector touching a table in front of it, while applying a desired constant vertical force. This task is relevant for many applications that require sliding contacts, such as cleaning a surface, using a tool on an object, etc. Tasks involving sliding contacts are especially difficult to optimize in general. The task is designed such that the robot starts from a random initial position. It should be able to reach the table, establish a safe contact and exert a desired vertical force. We consider three types of uncertainties for the table: stiffness (i.e. how soft is the contact), friction, and height. These uncertainties are relevant for real-world applications as changes in contact properties and location can easily destabilize controllers and lead to failures. Moreover contact stiffness and friction cannot be known precisely before interaction in an unknown environment.

For this task, the states of the system consist in the positions and velocities of all 7 joints of the arm, as well as the total force measured by the endeffector.

Reward function design. To learn a policy for achieving the desired task we define a reward function consisting of several parts (1-5). We use two terms to drive the circular motion along a desired trajectory: (1) the current distance from the endeffector to the closest point on the circle and (2) the difference between the current velocity vector and the desired tangential velocity on the closest point on the trajectory so as to achieve a motion with constant angular velocity. The three other terms are: (3) a reward based on the orientation of the endeffector, (4) a constant reward for any interaction between the endeffector and the table and a further reward based on the difference with desired contact force and (5) a penalty term

for any interaction between the table and any part of the robot other than the endeffector.

While the reward function might seem complex, each term directly encodes one aspect of the task and we pay particular attention not to incentivize any specific behavior in solving it. The apparent complexity is precisely the result of this, as we use multiple terms to define the circular motion along the trajectory, instead of simply having one specific point to track, explicitly to ensure time-invariance of the resulting policies.

B. Simulation results

Quantitative comparison. We examine the robustness of our approach to variability present in the environment. We consider in three separate simulations uncertainties on table height, friction, and stiffness. For each of the three variables we define a possible range of values and uniformly sample a new environment in each episode during training. We vary the table height in a $20\,\mathrm{cm}$ range from $0.8\,\mathrm{m}$ to $1.0\,\mathrm{m}$ and Coulomb friction coefficients in a range from 0 (no friction) to 1. We also vary the rigidity of the surface (which can easily influence the stability of a controller) with stiffness values from $50\,\mathrm{N/m}$ to $500\,\mathrm{N/m}$.

We perform each policy training for a fixed, predefined number of episodes. For each controller we repeat the training 6 times, with different circular trajectories to track. In Figure 8 we present the combined results, showing the mean and standard deviation for the learning curves across these individual trainings.

We can see that variable gain control outperforms both of the other two parametrizations. Splitting the control into motion and impedance parts makes it crucially easier for a good behavior to be found. One control term can handle the circular motion, while the other, depending on the experiment, can manage contact location uncertainty or compensate for unknown friction of the surface.

Qualitative analysis on contact transition. Further comparison between different cases in Figure 8 reveals that the performance gap between the variable gain policies and the other two (fixed gains and direct torque) is more obvious when there is uncertainty in the contact location. Since the dynamics of the system changes before and after contact, transition between the two modes (i.e. free motion vs. in-contact) has a critical impact on the task achievement. Hence, the policy that is able to tolerate uncertainty in the mode transition can outperform other ones drastically. To investigate qualitatively the behaviour of different policies in this case, we plotted the corresponding normal interaction forces (Figure 9) for a representative experiment. We can clearly see that the applied force from variable gain policy is smooth without loosing contact. On the other hand, direct torque control looses contact frequently and the fixed gains policy generates forces with high frequency oscillations. The variable gain policy leads to smoother contact forces which could be realistically applied on a real robot.

VI. DISCUSSION

Trajectory tracking term. The trajectory tracking term was crucial in our sim-to-real transfer, as it prevents the policy from varying the desired position with a high frequency. In other words, the policy is incentivized to change the desired position in a way that is consistent with the system dynamics and constraints (as much as it does not degrade achieving the desired task). As a side benefit, the trajectory tracking term also gives interpretability to the output of the policy, i.e. the sequence of desired positions over time can be seen as the desired feasible trajectory and the multiplier to the error replicates the feedback gains. As a result, if we have a variable gain policy, we can find a desired trajectory and an optimal set of feedback gains for that desired trajectory. This interpretability can yield insight about the optimal impedance modulation for contact-rich tasks, which is still an open problem in the field.

Action space parametrization alternatives. When proposing any new structure in the action space of the policy, in addition to standard considerations on the control side, we suggest to take into account two additional factors from the learning perspective. First, good exploration is critical for fast convergence and to avoid getting stuck in undesired local minima. This is where position control based policies come ahead of direct torque control, but they in no way completely solve the problem and there is further research to pursue in that area. Second, if we entirely decouple the feedback path from the feedforward one (e.g. $\tau = \tau_{ff}(\xi) - K_p(\xi)x - K_d(\xi)\dot{x}$), the learned policy may realize the entire control through a single term, mostly ignoring the other one (our experiments with such control law formulations resulted in precisely that type of behavior). This would also strip the control law terms of any physical meaning we tried to impose. When using any such control law in a policy learning setting, where the same behavior can be produced by different combinations of control terms, there needs to be something incentivizing one choice over another. Only then can the individual terms have the intended physical meaning, giving us the interpretability we desire.

Stability of variable impedance policy. Varying impedance as a function of state can cause instability of the control. As discussed in [21], reasonable varying stiffness profiles show no destabilization tendencies. In this paper, first we found a range of joint stiffness and damping that does not cause any instability on the real system for a wide range of motions. Then we let the policy find a state-dependent impedance within this range. With this strategy, we never experienced any instability when we applied the learned policy directly on the real robot.

VII. CONCLUSION

In this paper, we have investigated the effect of action space representation on the performance and robustness in contact-rich tasks in the presence of uncertainties. On both a floating-base hopping task and a fixed-base table wiping one we demonstrate that variable impedance control allows us to find better performing policies and to do so more reliably. Additionally, we showed how we can use a regularization term to impose the original physical meaning to desired trajectory and impedance, giving interpretability to these policies. Finally, we

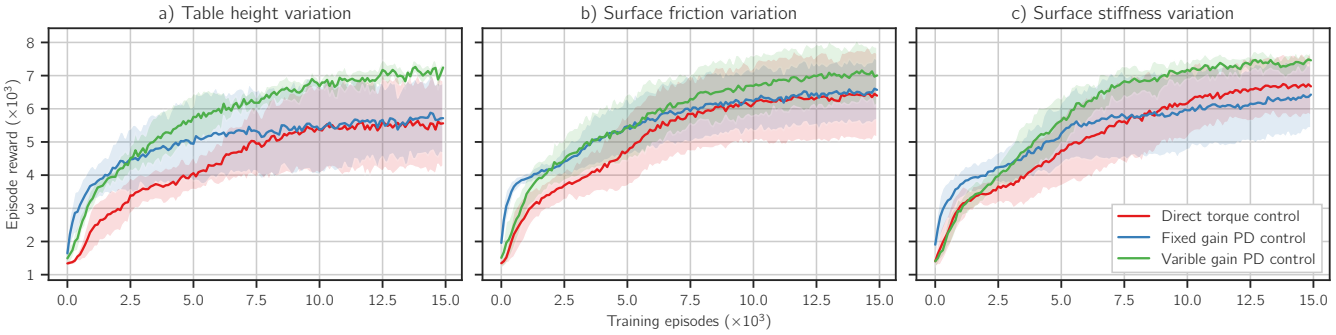


Fig. 8: Robustness evaluations for the fixed-base setup. Results for environments with varying table height (a), table surface friction (b), and table surface stiffness (c).

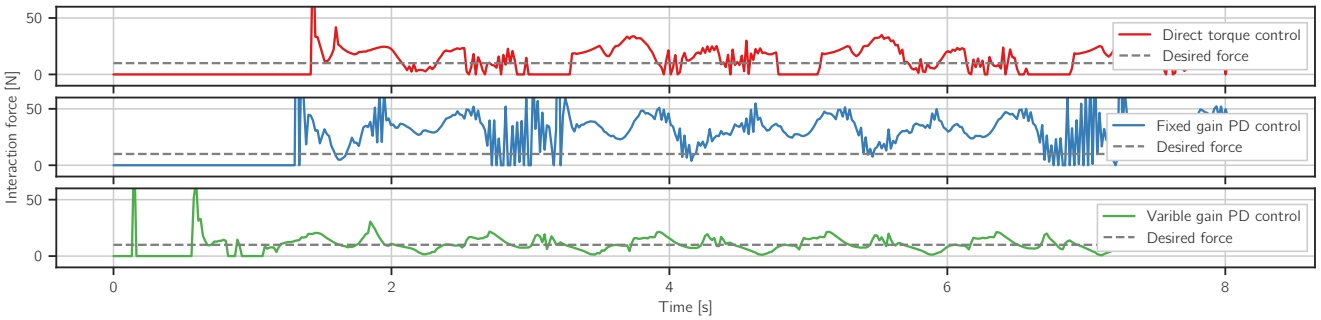


Fig. 9: Table force interactions during circle tracking task for learned policies for different controllers.

demonstrated how the policies can then directly be deployed on a real system, preserving performance and robustness.

REFERENCES

[1] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p. 12, ACM, 2017.

[2] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[3] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4639–4644, IEEE, 2011.

[4] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[5] F. Stulp, J. Buchli, A. Ellmer, M. Mistry, E. A. Theodorou, and S. Schaal, "Model-free reinforcement learning of impedance control in stochastic environments," *IEEE Transactions on Autonomous Mental Development*, vol. 4, no. 4, pp. 330–341, 2012.

[6] K. Kronander and A. Billard, "Learning compliant manipulation through kinesthetic and tactile human-robot interaction," *IEEE transactions on haptics*, vol. 7, no. 3, pp. 367–380, 2013.

[7] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," *arXiv preprint arXiv:1903.01066*, 2019.

[8] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4326–4332, IEEE, 2011.

[9] S. M. Khansari-Zadeh and O. Khatib, "Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors," *Autonomous Robots*, vol. 41, no. 1, pp. 45–69, 2017.

[10] J. Viereck, J. Kozolinsky, A. Herzog, and L. Righetti, "Learning a structured neural network policy for a hopping task," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4092–4099, 2018.

[11] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," *arXiv preprint arXiv:1906.08880*, 2019.

[12] P. Varin, L. Grossman, and S. Kuindersma, "A comparison of action spaces for learning manipulation tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6015–6021, Nov 2019.

[13] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Soueres, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, 2013.

[14] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[15] J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell, "Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 464–470, IEEE, 2015.

[16] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[18] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, 2020. Early access.

[19] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-aware model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1517–1524, 2019.

[20] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1–8, IEEE, 2018.

[21] K. Kronander and A. Billard, "Stability considerations for variable impedance control," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1298–1305, 2016.

B Submitted manuscripts

Model-free Reinforcement Learning for Robust Locomotion Using Trajectory Optimization for Exploration

Miroslav Bogdanovic[1], Majid Khadiv[1] and Ludovic Righetti[1,2]

***Abstract*— In this work we present a general, two-stage reinforcement learning approach for going from a single demonstration trajectory to a robust policy that can be deployed on hardware without any additional training. The demonstration is used in the first stage as a starting point to facilitate initial exploration. In the second stage, the relevant task reward is optimized directly and a policy robust to environment uncertainties is computed. We demonstrate and examine in detail performance and robustness of our approach on highly dynamic hopping and bounding tasks on a real quadruped robot.**

I. INTRODUCTION

Deep reinforcement learning (DRL) has recently shown great promises to control complex robotic tasks, e.g. object manipulation [1], quadrupedal [2] and bipedal [3] locomotion. However, exploration remains a serious challenge in RL, especially for legged locomotion control, mainly due to the sparse rewards in problems with contact as well as the inherent under-actuation and instability of legged robots. Furthermore, to successfully transfer learned control policies to real robots, there is still no consensus among researchers about the choice of the action space [4] and what (and how) to randomize [5] in the training procedure to generate robust policies.

Trajectory optimization (TO) is a powerful tool for generating stable motions for complex and highly constrained systems such as legged robot [6], [7], [8]. However, re-planning trajectories through a model predictive control (MPC) scheme is still a challenge, because the computation time for solving a high-dimensional non-linear program in real-time is highly expensive. Furthermore, apart from recent works explicitly taking into account contact uncertainty to design robust control policies [9], [10], the inclusion of robustness objectives in trajectory optimization can quickly end up in problems that cannot be solved in real time for high-dimensional systems in multi-contact scenarios.

In this work, we propose a general approach allowing for the generation of robust policies starting from a single demonstration. The main idea is to use TO to generate trajectories for different tasks that are then used for exploration for DRL. The approach allows us to generate policies robust to environmental, and particularly contact transition, uncertainties.

It is crucial to highlight that simply trying to optimize the policy to track the demonstration as well as possible is highly problematic in the presence of environmental uncertainties. As an example, let us say we want to produce a hopping policy that can account for large uncertainty in ground height. Instead of just going directly into a baseline hopping motion after making contact with the ground, the policy would need to force itself to get back into phase with the demonstration to have any luck to complete the task. This might not even be possible as the policy would also need to have some information about the current phase of the demonstration trajectory – which, without having time in the input state, might not be possible. Hence, losing the time dependence from the demonstration trajectories in the final feedback policy is the key in our approach to provide robustness with respect to contact timing uncertainties.

A. Related work

Demonstrations have long been used in dealing with exploration issues in reinforcement learning for robotic tasks [11], [12], [13]. Recently, demonstrations have been used as one of the main ingredients to learn complicated locomotion behaviors in a DRL setting [14], [15]. In these works, a policy is trained using reinforcement learning to reproduce a given demonstration trajectory as well as possible in simulation. In [15], those policies are transferred to the real system relying on domain randomization methods. The main difference between those works and our approach is that we do not restrict ourselves to tracking a demonstration trajectory. Instead, we only use the demonstration to avoid exploration issues and get a reasonable initial policy. Then, we continue training with a time-independent, direct task reward instead of the reward for tracking the demonstration trajectory. This gives the policy the freedom to optimize directly for task performance instead of having to reproduce the demonstration. Additionally, it allows for better recovery behaviors, as the policy is able to adapt the timing of the motion to environmental changes.

Several contributions have used reinforcement learning to compute policies in simulation before deploying them on a real quadruped robot [2], [16]. However, these results are typically limited to not very dynamic behaviors. Additionally, in [16], strong prior structure is imposed on a policy by having it output foot position residuals and motion frequencies, instead of directly doing control in joint space. These kinds of approaches become less feasible in building truly robust

[1]Max-Planck Institute for Intelligent Systems, Tübingen, Germany {mbogdanovic, mkhadiv}@tue.mpg.de

[2]Tandon School of Engineering, New York University, USA {ludovic.righetti@nyu.edu

This work was supported by New York University, the European Union's Horizon 2020 research and innovation program (grant agreement 780684) and the National Science Foundation (grants 1825993, 1932187 and 1925079).

policies for dynamic tasks, with fast contact transitions and longer flight phases.

B. Contributions

We propose a two-stage method for computing robust adaptive policies for real robots starting from a single demonstration trajectory. The approach allows the policy to find a robust solution without being hindered by the time-depended aspect of the demonstration trajectory or be biased by it in general.

We use this approach to produce highly dynamic motions on a quadruped robot. By training the control policy to handle varied initial conditions, as well as contact locations and timings, we produce robust policies with smooth contact switching behavior at high speed. Additionally, trained policies prove robust to elements they have not been explicitly trained for, like uneven terrain, soft terrain or external perturbations.

For each task, the procedure starts from a single demonstration trajectory which does not even need to be fully physically realizable. We further show how we can use this single demonstration to produce a varied set of policies for the real robot.

II. TWO-STAGE ALGORITHM

Our proposed algorithm proceeds in two stages as shown in Fig. 1. In the first one we train a policy to track a time-based demonstration trajectory as well as possible inside a physical simulation. In the second stage we further adapt the resulting policy, making it now optimize for a time-independent task reward instead. We also introduce variability in the environment and further reward terms to regularize the behavior in this stage. As a final result we get a robust policy that can be directly deployed on the real system without any additional training.

A. Stage 1: Learning a policy to track a given trajectory

Initial demonstration. We assume we have access to a demonstration trajectory showing general kinematic motion required for completing the task, while not necessarily being fully physically realizable. In this work we use the trajectory optimization algorithm proposed in [8] to compute such demonstrations. This approach is general and does not consider any simplification in the dynamics or kinematics, hence it is capable of generating a variety of different gaits for legged robots. It is important to emphasize that any other trajectory optimization algorithm can be used in our approach, as long as it provides a set of full-body trajectories.

In order to utilize such demonstration trajectories it is not sufficient to simply learn to reproduce actions taken in the trajectory. The policy needs to learn when to go off the trajectory in order to complete the task. With our proposed approach the actions taken in the demonstration are never used, which among other things gives us freedom in the choice of how to parametrize the controller we are trying to learn.

Controller parametrization Throughout this work we use PD control in joint space to control the robots. The policy outputs the desired joint positions at each step and uses fixed P and D gains. Throughout all our stages of training we have an additional term incentivizing the policy to output values for desired joint positions that are actually tracked as well as possible. Specifically, we penalize the difference between the value given for the desired position by the policy at step t and the actual position achieved at next step $t+1$:

$$r_{tt} = -k_{tt} \left\| \mathbf{q}_{des}^{joint}(t) - \mathbf{q}^{joint}(t+1) \right\|^2 \tag{1}$$

This has been the crucial aspect in our previous work [17] for direct policy transfer to real robot without domain randomization of robot parameters.

Training procedure. We use Proximal Policy Optimization (PPO) [18] to optimize the policies, but we do not have many requirements in the choice of algorithm. As we use the provided demonstration to resolve exploration issues, we do not need a, potentially off-policy, reinforcement learning algorithm with strong characteristics in this regard. We instead choose an on-policy algorithm with good convergence properties.

In this stage, the optimized reward consists of two parts: a part for tracking the time-based demonstration and the above-defined regularization term that is a part of the controller parametrization:

$$\begin{aligned}
r_{s1} &= r_{ti} + r_{tt} \\
r_{ti} &= k_{ti1} \exp\Big(- k_{ti2} \|\mathbf{x}_{demo}^{base} - \mathbf{x}^{base}\| \Big) \\
&\quad k_{ti3} \exp\Big(- k_{ti4} \|\dot{\mathbf{x}}_{demo}^{base} - \dot{\mathbf{x}}^{base}\| \Big) \\
&\quad k_{ti5} \exp\Big(- k_{ti6} \|\mathbf{q}_{demo}^{base} \ominus \mathbf{q}^{base}\| \Big) \\
&\quad k_{ti7} \exp\Big(- k_{ti8} \|\boldsymbol{\omega}_{demo}^{base} - \boldsymbol{\omega}^{base}\| \Big) \\
&\quad k_{ti9} \exp\Big(- k_{ti10} \|\mathbf{q}_{demo}^{joint} - \mathbf{q}^{joint}\| \Big) \\
&\quad k_{ti11} \exp\Big(- k_{ti12} \|\dot{\mathbf{q}}_{demo}^{joint} - \dot{\mathbf{q}}^{joint}\| \Big) \\
r_{tt} &\text{ as defined in (1),}
\end{aligned} \tag{2}$$

where $\mathbf{x}^{base}$ is the position of the robot base, $\mathbf{q}^{base}$ the base quaternion, $\boldsymbol{\omega}^{base}$ the base angular velocity and $\mathbf{q}^{joint}$ the joint positions. k_{ti1}, ..., k_{ti12} represent individual weight and scale constacts for each term. We mark difference between two quaternions with $\ominus$.

Following [14], we make two further design choices that prove to be vital in making the training robustly work:

1) We initialize each episode at a randomly chosen point on the demonstration trajectory.
2) We terminate episodes early if the robot enters states that are not likely to be recoverable (based for example on tilt angle of the robot base) or just not conducive for learning (for example knees of the robot making contact with the ground).

Output. In the first stage, we aim to produce a policy that provides some nominal behavior on the task in simulation.

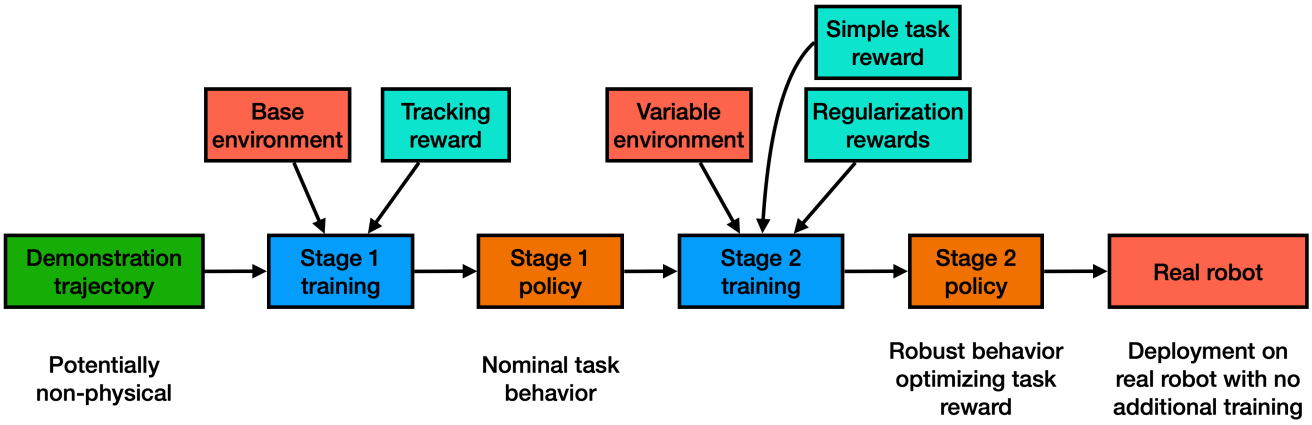


Fig. 1: Schematic of our proposed framework. We start with a single demonstration (which need not be physically realizable in simulation) and try in the first stage to learn a policy to track the demonstration trajectory that produces successfully nominal behavior in simulation. To enable transfer to the real robot, the second stage starts with the resulting policy from the first stage and tries to robustify the policy by randomizing contact and to optimize for performance by replacing demonstration tracking reward with task reward. We directly apply the output policy from the second stage to the robot without any domain randomization of robot parameters.

However, in our experiments, these policies failed to transfer to the real robot. They remain static, cause shaky behavior on the robot, or resulted in motions with severe impacts (examples in the accompanying video). To solve these problems, we need an additional stage of training that generates robust policies in simulation that transfer to the real robot.

B. Stage 2: Generating robust time-independent policy

To modify policies such that they are successfully transferable to the real robot, We continue training starting from the policies outputted from the first stage, introducing the following changes in the training procedure:

Initialization. We replace initialization on the demonstrated trajectories with initialization in a wider range of states. This allows us to better cover the range of states the policy might observe when deployed on the real system, allowing it to learn how to recover and continue the motion in those cases.

Environment uncertainties. We introduce uncertainties in the environment in order to produce more robust motions when deployed on the real system. In this work, we are mainly concerned with randomization of the contact surface heights.

Time-independent task reward. We replace the time-based demonstration tracking reward with a time independent, direct task reward. We have already noted the issues that can arise while trying to account for environment uncertainties while tracking a time-based demonstration trajectory. The policy is locked into trying to follow the specific time schedule regardless of the environment, whereas adapting it would produce much better recovery behavior. Switching to a time-independent reward, directly defining the task helps us deal with this.

Switching to this task reward has additional benefits. It allows us to directly optimize desirable aspects of the task, whereas the demonstration only needs to give us some nominal behavior on the task. The policy is free to change the behavior in a way that is needed to perform the task in the best possible way, without being penalized for not doing it in the same way as in the demonstration. We can also, as we will see later, produce varied behavior starting from a single demonstration by adapting this task reward.

Regularization rewards. Finally, we add additional reward terms in this stage to further regularize the behavior of the learned policies. We aim to incentivize desirable aspects of policies in the tasks, like torque smoothness and smooth contact transitions.

III. EVALUATION

A. Tasks setup

We evaluate our approach on two different dynamic tasks on a quadruped robot: hopping and bounding. In both cases we start from a basic demonstration trajectory that gives basic shape of the motion without necessarily being physically realizable. Starting from that we apply our two-step training procedure in simulation to produce robust policies that we then test on a real robot.

We perform experiments on the open-source torque-controlled quadruped robot, Solo8 [19] (see Fig. 2), which is capable of very dynamic behaviors. For simulating the system we use PyBullet [20].

We use exactly the same training procedure in both cases, with the only differences arising from the need to allow for base rotation around one axis in the bounding task. This is a particular benefit of the approach we present here – for a new task we only need a single new demonstration trajectory and a single simple reward term defining the task.

Stage 1 – Early termination. The only aspect in the first stage of training specific to the chosen tasks is how we perform early termination. We perform early stopping here

based on the current tilt of the robot base (we increase the range appropriately for the bounding task), as well as when any part of the robot that is not the foot touches the ground.

Stage 2 – Initialization. As noted in the method description, in the second stage of training we introduce a wider range of initialization states. In the two tasks we examine here, this consists of randomizing the initial height of the base of the robot, tilt of the base around x- and y-axis and randomness in the initial joint configuration. We preserve the early termination criteria from stage 1, only extending the range of allowed base tilt angles with the way it is increased in the initialization.

Stage 2 – Environment uncertainties. We also introduce uncertainties in the training environment. We randomize the ground position up and down in the range of $[-5\,\mathrm{cm}, 5\,\mathrm{cm}]$ (approximately 20% of the robot leg length). We also randomize the ground surface friction coefficient in the range $[0.5, 1.0]$. While we restrict ourselves only to this limited set of initial state and environment randomizations, as we will see in the later evaluations, this produces policies that are quite robust as they can also handle uneven ground or external perturbations.

Stage 2 – Reward structure (hopping task). By using a demonstration trajectory to deal with exploration issues, we can define the individual task rewards to be very simple, without the need for any reward shaping.

For the hopping task we use the following reward

$$r_{s2hp} = r_{hp} + r_{ps} + r_{ct} + r_{ts} + r_{tt} \tag{3}$$

We use the r_{hp} reward term to define the task

$$r_{hp} = \begin{cases} k_{hp} z^{base}, & \text{if } z^{base}_{min} < z^{base} < z^{base}_{max}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The reward at each timestep is proportional to the current height of the robot base (z^{base}), with constant weight k_{hp}. It is clipped to zero bellow a certain threshold (z^{base}_{min}), one that the robot can reach without leaving the ground. We additionally clip the value of this reward to be zero above a certain height threshold (z^{base}_{max}) to incentivize lower hops.

We also introduce several reward terms to incentivize different desirable aspects in the resulting behavior. They reward the base to be close to its horizontal default posture (r_{ps}) and smooth contact transitions (r_{ct}) and torque smoothness (r_{ts}).

We reward the policy for being static in all the base dimensions (positions (x^{base}, y^{base}) and Euler angles (θ^{base}_x, θ^{base}_y, θ^{base}_z)) except the one the motion is performed on (z-axis in this case). With k_{ps1}, ..., k_{ps10} being weight and scale constants.

$$\begin{aligned} r_{ps} = k_{ps1} \exp\left(-k_{ps2}|x^{base}|^2\right) \\ k_{ps3} \exp\left(-k_{ps4}|y^{base}|^2\right) \\ k_{ps5} \exp\left(-k_{ps6}|\theta^{base}_x|^2\right) \\ k_{ps7} \exp\left(-k_{ps8}|\theta^{base}_y|^2\right) \\ k_{ps9} \exp\left(-k_{ps10}|\theta^{base}_z|^2\right) \end{aligned} \tag{5}$$

This term is crucial as it drives the policy to stay at the default posture as much as possible. Without it the policy could perform the task well in the simulation while always being close to falling over – which would likely happen when it was transferred to the real system.

The second key reward term asks for smooth contact transition (r_{ct})

$$r_{ct} = \begin{cases} -k_{ct} \sum_{i=1}^{4} F_i^{foot}, & \text{if } \sum_{i=1}^{4} F_i^{foot} > F_{limit}^{foot}, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

We do so by simply penalizing any contact force values (F_i^{foot}) above a certain threshold (F_{limit}^{foot}), to penalize impact, with k_{ct} being a constant weight. Without this term we would have the feet hitting the surface hard on each landing – this is precisely what we observe in policies from stage 1 where this reward term is not present. This is not the kind of behavior desired on the real system and these impacts can cause actual damage to the robot. These types of policies also transfer less well between simulation and real world. They can learn to perform the task well in simulation by generating hard impacts, but doing so exploits the weaknesses of the contact model in simulation, resulting in a poor performance when transferred to the real system. Smooth contact transitions enable a better transition between simulation and the real system and so the policies where incentivize those end up transferring to the real system much better.

The third reward term (r_{ts}) prevents the policy to ask for a very quick change in the desired torque which is not realizable on the real robot with limited control bandwidth

$$r_{ts} = -k_{ts1} \exp\left(k_{ts2} \left\|\boldsymbol{\tau}(t) - \boldsymbol{\tau}(t-1)\right\|\right) \tag{7}$$

with τ being the joint torque and k_{ts1} and k_{ts2} weight and scale constants respectively.

Final reward term r_{tt} is the same one we use in the first stage of training (as defined in (1)).

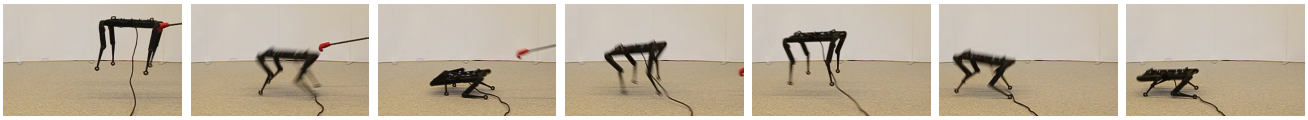
Stage 2 – Reward structure (bounding task). For the bounding task, we only make changes to the parts of the reward defining the task:

$$r_{s2bn} = r_{bn} + r_{cc} + r_{ps} + r_{ct} + r_{ts} + r_{tt} \tag{8}$$

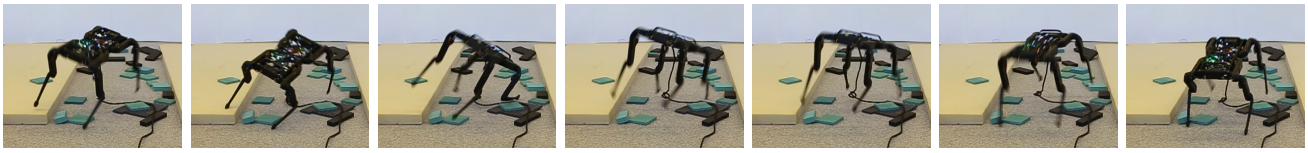
We define the task reward here in two parts, r_{bn} and r_{cc}. r_{bn} rewards the policy for being close to the path the demonstration takes in the $[z^{base}, \theta^{base}_y]$ space

$$r_{bn} = -k_{bn} \min_i \|(z^i - z^{base}, \theta^i_y - \theta^{base}_y)\| \tag{9}$$

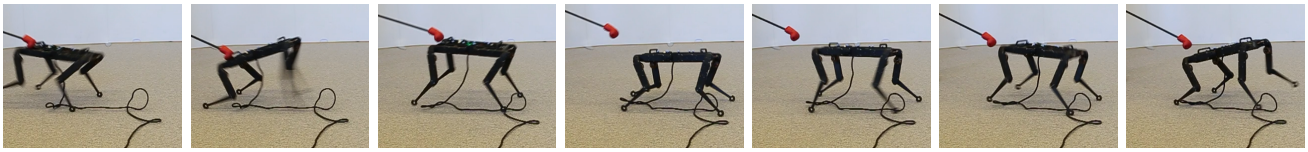
(a) Hopping on a surface comprised of a soft mattress and small blocks



(b) Hopping with push recovery



(c) Bounding on a surface comprised of a soft mattress and small blocks



(d) Bounding with push recovery

Fig. 2: Examples of the robustness tests carried out on the quadruped robot Solo8.

with k_{bn} being a constant weight. We do this with no concept of time in this case, by just taking the distance to the closest point. This gives the policy freedom to do this motion slower or faster, with different amplitude.

The second part of the task reward in this case, r_{cc}, is related to the contact state

$$r_{cc} = \begin{cases} k_{cc}, & \text{only front two legs in contact,} \\ k_{cc}, & \text{only back two legs in contact,} \\ k_{cc}, & \text{no legs in contact,} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

with k_{cc} being a constant reward. It incentivizes the policy to, when making contact with the ground, only do so with front or back legs at the same time. Without anything to incentivize the policy to do this, we have observed stage 1 policies reproducing the bounding motion while keeping all feet in contact with the ground. This reward part ensures appropriate contact states with flight phases in between.

We keep the other reward terms, ones used to incentivize desired aspects of the behavior, the same as in the hopping task (r_{tt} as defined in (1), r_{ct}, r_{ts} as defined in (7)). The one change we make is to the reward incentivizing the robot base staying close to the default posture, r_{ps}

$$\begin{aligned} r_{ps} = k_{ps1} \exp\left(-k_{ps2}|x^{base}|^2\right) \\ k_{ps3} \exp\left(-k_{ps4}|y^{base}|^2\right) \\ k_{ps5} \exp\left(-k_{ps6}|\theta_x^{base}|^2\right) \\ k_{ps7} \exp\left(-k_{ps8}|\theta_z^{base}|^2\right) \end{aligned} \tag{11}$$

We do not reward staying at default posture in the θ_y^{base} direction in this case, as that is the angle the robot is moving around while bounding.

B. Hopping task results

Figure 3 shows results when we drop the real robot from different heights to start the motion. We show the base height and estimated contact force for one of the legs in time. As we do not have force sensors in the feet, we estimate the contact forces based on the torques the robot applies, using $F_i^{foot} = (S_i J_i^T)^{-1} S_i \tau$, where S_i and J_i are the joint selection matrix of the leg i and Jacobian of the foot i, respectively. Note that this estimation ignores the energy dissipated through damping of the robot structure and drive system. We further align the plots based on the later part of the motion – the stable cycle the robot gets into.

First, we can note that regardless of the drop height the robot goes into the same stable hopping cycle. What is more, it does so very quickly, as we can see all the individual rollouts matching after only two hops. We can also note here the benefits of time independence of the policy. It is what allows us to be able to start the motion from this large range of initial heights. It is also what enables this fast stabilization, as we can see that the two initial hops are on a different cycle – one needed to stabilize the motion properly.

This test also allows us to show the general quality of contact interaction we are able to achieve with this approach. We can see that the impact forces, even on the highest drop ($1\,\mathrm{m}$ height), barely go over the force values for the stable hopping cycle ($50\,\mathrm{cm}$ height). This is purely learned behavior, as a result of impact penalties introduced in stage

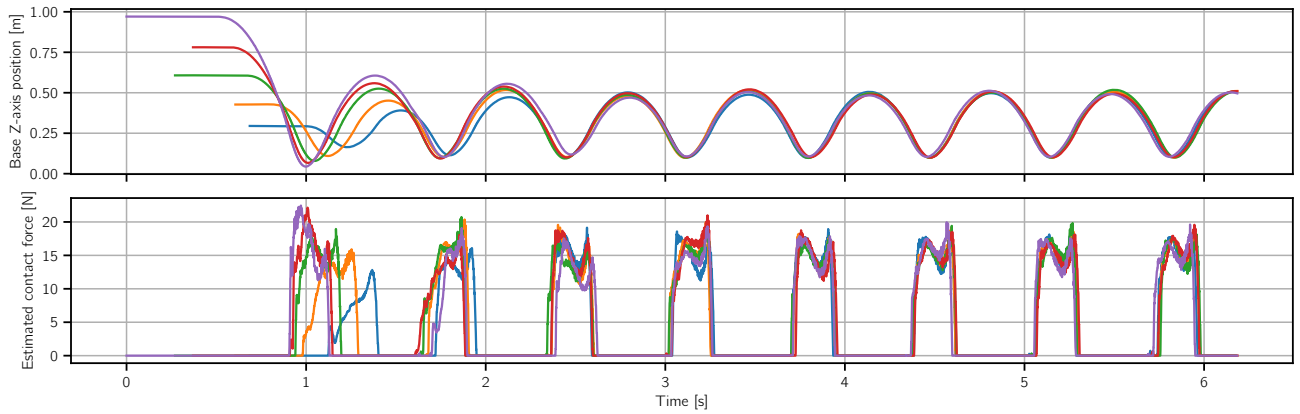


Fig. 3: Hopping experiments started from different initial heights. The top figure shows that after dropping the robot from a set of different heights ranging between 0.3-1 m, the robot quickly goes back to the nominal behavior which is hopping with the maximum height of 0.5 m. The bottom figure shows that, dropping from different initial heights, the robot is able to adapt its landing such that the impact forces remain very low and almost invisible in the estimated force.

2 of the training. It is not present in the demonstration and when we test stage 1 on the real system high impact forces are generated and the policies are very fragile. The general character of the smooth contact transition can also be well observed in the accompanying video.

In addition, the resulting behavior shows robustness to uneven terrain and external pushes, without ever being explicitly trained for either. The robot is able to recover from significant tilt of the base arising from either external push or landing on an uneven surface (Fig. 2a, 2b). More extensive examples of recovery behavior can be seen in the accompanying video.

Finally, we demonstrate the variety of robust behaviors that can be optimized from the same demonstration by doing repeated trainings with hopping reward being clipped at some maximum height – giving a value of zero above it. With this simple change in the task reward, starting from one demonstration, we can produce hopping behaviors at different heights. Examples of this can be seen in the accompanying video.

C. Bounding task results

In Fig. 4 we show results for a test where we drop the robot from different angles to start the motion. We perform the same test for two different final stage 2 policies for this task. We can see that the policies can handle a wide range of initial base angles – around 35 degrees in both directions. What is more, as was the case with the hopping task, we can see that here as well all the initialization end up in the same stable motion cycle.

The bounding motion also exhibits similar robustness to uneven terrain and external perturbations as the hopping motion (Fig. 2c, 2d). Same as with the hopping task, the policies rely on their knowledge of how to handle a varied set of base states to recover from anything that arises from these conditions even through it was not explicitly trained on them.

In this task, we would also like to demonstrate variety of behaviors we can generate from one single demonstration. Unlike in the hopping task, where we made simple changes in the task reward to achieve this, here we instead give more freedom to the task reward and examine variety of produced behaviors. As noted in the task reward definition, the policy has the freedom to produce slower or faster bounds with smaller or larger amplitudes. As seen in the accompanying video we arrive at a variety of bounding behaviors in this way, starting from the same initial demonstration trajectory.

IV. DISCUSSION

Simplicity and generality of the approach. One of the main benefits of the approach we presented is its generality and simplicity in applying it to new tasks. The only elements needed for each new task, as can be seen from the two task we presented here, are a single demonstration trajectory and a direct straightforward task reward.

The trajectory we use does not need to be perfect or even fully physically realizable, it only needs to provide the rough trajectory needed to complete the task in some basic way. It does not need to provide ideal performance on the task, as we can optimize for task performance in the later stage of training. It also only needs to provide us with a sequence of states, and not necessarily actions, which is simpler in some cases where it is not trivial to calculate the exact forces to realize motion, like it is actually the case in the trajectory optimization we use here.

As for the task reward, reward shaping is not needed, as the demonstration resolves any exploration issues that could occur as a result of sparse reward signal. We can, as we do here, just directly reward the aspect of the task we care about. We keep reward terms other than the task reward as general as possible, encoding characteristics of general good behavior of the robot and we expect those to be kept the same across a varied range of tasks.

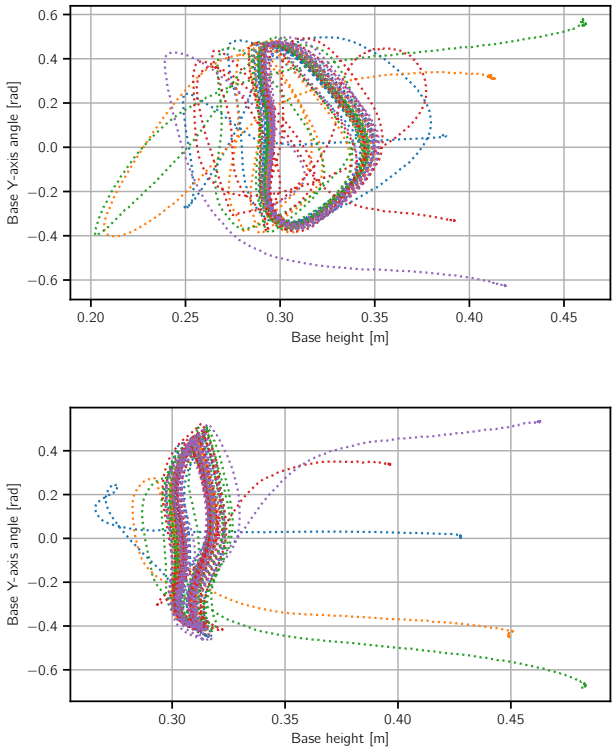


Fig. 4: Two different bounding behaviors on the robot with different initial conditions. Starting with a wide range of initial angles for the base in y-direction (roughly between -35 to 35 deg), the robot quickly converges back to the desired behavior.

Reinforcement learning perspective. From the reinforcement learning perspective our approach presents a simple and effective way to deal with exploration issues in robotic tasks. We also remove the trajectory tracking reward in the second stage of our training, so, as seen in our experiments, we are able to change the policy away from exact behavior defined in the demonstration.

Trajectory optimization perspective. From the trajectory optimization perspective, our approach proposes a systematic way to consider different types of uncertainty and find a robust control policy for robotic tasks, especially those with contact. We believe this is a practical way to combine the strength of trajectory optimization and reinforcement learning for continuous control problems; 1) Trajectory optimization is used to generate a desired behavior efficiently to achieve the task at hand 2) different types of realistic uncertainties are easily added to the simulation, e.g. contact timing uncertainty, and RL is used to produce a robust feedback policy.

V. CONCLUSION

In this work, we presented a general approach for going from trajectories gained by doing trajectory optimization to robust learned policies on a real robot. We showed how we can start from a single, not necessarily physically realizable, trajectory and arrive at a robust policy that can be directly deployed on a real robot, without any need for additional training. Through extensive tests on a real quadruped robot we demonstrated significant robustness in the behaviors produced by our approach. Importantly, we do so in setups, uneven ground and external pushes, for which the robot was not explicitly trained for. All this gives hope that approach like this could be used across varied robotic tasks to simply generate robust policies to be used on real hardware, bridging the gap between trajectory optimization and reinforcement learning in such tasks.

REFERENCES

[1] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[2] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[3] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning*, pp. 317–329, PMLR, 2020.

[4] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p. 12, ACM, 2017.

[5] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," 2021.

[6] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[7] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.

[8] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Transactions on Robotics*, pp. 1–19, 2021.

[9] A. Aydinoglu, V. M. Preciado, and M. Posa, "Stabilization of complementarity systems via contact-aware controllers," *arXiv preprint arXiv:2008.02104*, 2020.

[10] B. Hammoud, M. Khadiv, and L. Righetti, "Impedance optimization for uncertain contact interactions through risk sensitive optimal control," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4766–4773, 2021.

[11] S. Schaal *et al.*, "Learning from demonstration," *Advances in neural information processing systems*, pp. 1040–1046, 1997.

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

[13] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[14] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[15] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[16] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[17] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6129–6136, 2020.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[19] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[20] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." http://pybullet.org, 2016–2020.