DEEP REINFORCEMENT LEARNING FOR ADAPTIVE CONTROL IN ROBOTICS

By

Luke Bhan

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER of SCIENCE

in

Computer Science

May 13, 2022

Nashville, Tennessee

Approved:

Gautam Biswas, Ph.D.

Marcos Quinnones-Grueiro, Ph.D.

ACKNOWLEDGMENTS

I would like to thank my mom and my dad for their support. Additionally, I would like to thank my lab mates Ibrahim and Tim as well as my mentors Dr. Biswas and Dr. Quinones-Grueiro for their assistance with my development as a young researcher.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction and Motivation

1.1 Introduction

In recent years, the control of nonlinear systems using data-driven methods has garnered large interest due to a multitude of reasons. First, the amount of pure compute available in modern processors has increased. Second, advances in Deep Learning algorithms, particularly in reinforcement learning, have shown viability on both simulated and real-world tasks. However, there is still a series of significant challenges to overcome before robotics can robustly be deployed in real-world applications. In particular, data-driven policies struggle to generalize to real world environments due to the ever changing dynamics of system parameters. Furthermore, policies need to be robust to unforeseen state inputs and maintain ”reasonable” safety guarantees to avoid damaging nearby surroundings.

In this work, we present two approaches for controlling real-world robotic systems with vastly different environmental challenges. Our work is based on data-driven methods for control as these methods have shown great promise to capture the nonlinear dynamics of systems that cannot be represented physically. First, we propose a form of hierarchical reinforcement learning that combines system identification and policy blending for generalizing across different human-assitive environments. We are able to show sample efficiency over standard domain randomization and able to regularly complete an itching task that otherwise would require large training times (¿ week). Next, we showcase a different form of deep reinforcement learning by designing a hybrid-model based data-driven control architecture for flying an octorotor under motor faults. We demonstrate improved control over a model-based controller for both single and dual faults at high fault magnitudes. Furthermore, we validate the robustness of our approach by varying the system parameters fed to the controller and demonstrate consistent flight across a single trajectory task. In both the aforementioned scenarios we demonstrate a control scheme for data-driven methods that are both sample efficient and generalize across their respective tasks.

CHAPTER 2

Policy Blending for Assistive Control

This chapter is adapted from a conference paper published at the International Control of Robotics and Automation. Bhan et al. (2022)

2.1 Introduction

Over the last few years, there has been significant interest in developing models that are trained in simulation and then transferred to the real world Kaspar et al. (2020), Tobin et al. (2017), Tan et al. (2018). Despite progress in learning from simulated policies, these methods still suffer from long simulation times as they require large amounts of experience to handle the unknown environments present in the real world. Additionally, these policies struggle to generalize for complex situations as they may become unpredictable when faced with new challenges. As such, researchers have approached training these policies in two distinct ways. The first approach utilizes techniques, such as Kalman Filters to identify system parameters that can inform policies on how to respond in different environmental conditions Ljung (1986). However, these policies struggle to generalize and often require re-tuning Ljung (1986). The second approach involves randomizing sets of system parameters during training so that the policy learns to robustly handle a wide variety of situations. This approach - domain randomization Peng et al. (2018b) - requires the actual parameters to be in the set of random values and as such, the robustness of the policy is directly correlated to the range of the generated randomized parameters. Moreover, the resulting policies are overly conservative in action selection, especially when the parameter set is large. Finally, for complex tasks with many parameters, creating large ranges across multiple parameter values requires significant training time before a robust policy can be generated Matas et al. (2018).

Both of the above-mentioned approaches require a single policy that aims to 1) solve the task at hand for a single set of parameters and 2) can generalize the solution to a wide set of parameters. These policies struggle to generalize and fail for parameter spaces in complex tasks. As an alternative, we attempt to decouple the process of generating generalized system parameters to support the learning of policies that solve the task efficiently. We do this by utilizing blending techniques informed by system identification methods. With our approach, we train single policies that are efficient for a distinct sets of the parameter space and utilize a blending network to identify how best to combine the actions of these individual policies based on the estimated parameters of the system. We verify the validity of our approach for a collaborative robot and human itching task in which the human has different combinations of motor impairments. Our proposed approach works with the assumption that the space of each parameter is bounded, thus any combination of parameter values results in a convex space. This guarantees convergence of the blending network given a set of individual policies learned to control the system under single parameter variability.

2.2 Related Work

There have been many approaches to policy learning based on estimation of simulation parameters; however, to the author's knowledge, none combined system identification with a blending approach. For example, Martinsen et al. (2020) demonstrates the use of simultaneous system identification and policy training where they explore a series of predictive error methods to minimize the difference between the observed parameters and the estimated parameters for model predictive control (MPC). Additionally, domain randomization has been used for a motorized robotic control tasks Peng et al. (2018a). However, these tasks do not consider a collaborative environment nor do they handle multiple faults introduced by the interacting agents. Furthermore, OpenAI et al. (2019) solves a challenging Rubik's cube control task by automatic domain randomization that slowly increases the difficulty of the task, but can take significant time as it does not consider the integration of any real-world sampling. Lastly, Mehta et al. (2019) considers an adaptive domain randomization strategy where the framework attempts to identify domains that can create challenging environments for the policy. This approach is similar to ours, except that their approach is purely data driven based on the result of their policy. This requires significant training time due to potential sample inefficiency. In contrast, we utilize prior domain knowledge to identify environments that have a high potential of being challenging for the policy.

In addition to the large amount of research designed for efficient sim2real transfer, there have been a series of recent work that demonstrate the effectiveness of policy blending. Narita and Kroemer (2021) demonstrates the use of policy blending for simple tasks such as opening a cap, flipping a breaker, and turning a dial. However, their policy learns directly from sensor measurements and does not consider impairments in the agents. Furthermore, Dragan and Srinivasa (2013) has shown a policy blending technique between a human and robot policy for robot-assisted control to accurately assist the human with various tasks such as fetching a water bottle. However, this work does not consider training models using modern deep reinforcement learning (DRL) techniques. Given these approaches, it is worthwhile to combine robust policy blending with modern system identification as a new approach to generalized modelling.

2.3 Formalism

In this section, we first formalize the reinforcement learning problem for tasks with multiple varying parameters. Then, we outline three approaches presented in the literature to solve such tasks.

2.3.1 Robust Markov Decision Process

We model tasks with multiple varying parameters as a robust Markov decision process (R-MDP) defined using a tuple $(S, A, R, \mathscr{P}, \gamma)$, where S is the state space, A the action space, $R : S \times A \to \mathbb{R}$ represents a reward function, $\gamma \in [0,1]$ is called a discount factor, and $\mathscr{P}(s,a) \in \mathscr{M}(S)$ is an uncertainty probability set where $\mathscr{M}(S)$ represents a family of probability measures over next states $s' \in S$. The next state of the system is contingent on the conditional measure $p(s'|s,a) \in \mathscr{P}(s,a)$ where $s \in S$ is the current state and $a \in A$ is the action selected by an agent.

In this work, we adopt the assumption that $\mathscr{P}$ is structured as a Cartesian product $\otimes_{s\in S, a\in A}\mathscr{P}_{s,a}$, also known as the state-action rectangularity assumption Wiesemann et al. (2013). This implies that nature can choose the worst-transition independently for each state and action. Moreover, we assume that the uncertainty probability set is defined by the parameter space that characterizes the task $\lambda_t \in \Lambda$ such that $P(\mathbf{s}_{t+1} = \mathbf{s}'|\mathbf{s}_t, \lambda_t, \mathbf{a}_t, \kappa)$ $(S \times \Lambda \times A \rightarrow S)$.

In a standard MDP framework, a policy $\pi : S \rightarrow p(A)$ maps states to distributions over actions with the goal of maximizing the sum of the discounted rewards over the future. The optimization criterion is the following

$$\mathscr{J}(\pi^*) = \max_{\pi\in\diamond} \mathscr{V}^{\pi}(\int), \; \forall \int \in S. \tag{2.1}$$

where the value function, $V^{\pi} : S \rightarrow R$, defines the value of being in any given state s

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s,a)\right], \; \forall s \in S. \tag{2.2}$$

Traditional RL algorithms require that the system dynamics and reward function do not change over time to be able to find an optimal deterministic Markovian policy satisfying 3.2. This property is clearly not satisfied in the R-MDP case. Therefore, we propose an approach to decouple learning how the system dynamics change depending on the system parameters and learning how to optimally solve the task.

2.3.2 Domain Randomization

To effectively identify π^* in simulation, a set of parameters must be defined to model the environment. Domain randomization attempts to sample a set of some N parameters which we will denote ξ for which a reasonable range of potential values is constructed - usually from domain specific knowledge Tobin et al. (2017). In this paper, we will consider domain randomization of the uniform type such that the parameters are uniformly sampled within a feasible range. For example, the weakness of a certain human joint can be sampled uniformly between 0 and 1 where 0 invokes no mobility while 1 is a joint that is at full strength.

2.3.3 System Identification Via Parameter Estimation

System identification through parameter estimation is a well studied subject in which a estimator can consistently receive samples from a real world environment and generalize these samples into an estimated true value. In this work, we utilize the Unscented Kalman Filter (UKF) for our estimator Wan and Van Der Merwe (2000) and make the assumption that our real-world parameters can be measured with some confidence, but may be cluttered with noise. Although this is a strong assumption, it can be softened for systems with non-measurable parameters by using approximate models of the system plus general estimation methods like Particle Filtering. See for example Shamrao et al. (2018).

2.3.4 Autotuned Search Parameter Model (SPM)

When the environment's parameters cannot be measured neither estimated through physics-based estimation, we propose to utilize a new technique that can estimate the parameters by interacting with the environment as an agent. Recently, Du et al. formulated a new approach to system identification where they define a data-driven model that learns a map from observation-action-parameter estimate sequences to a probability distribution, i.e. $(o_{1:T}, a_{1:T}, \xi_{guess}) \implies (0,1)^N$ such that the parameter estimates ξ_{guess} are greater than, less than, or equal to the true parameters Du et al. (2021). The mapping works like a binary classifier that is continuously trained concurrently to the policy such that it slowly converges to the real world parameters by learning from its own policy interaction trajectories. Following this iterative search, we can then perform a level of system identification that does not completely rely on domain knowledge for our experiments.

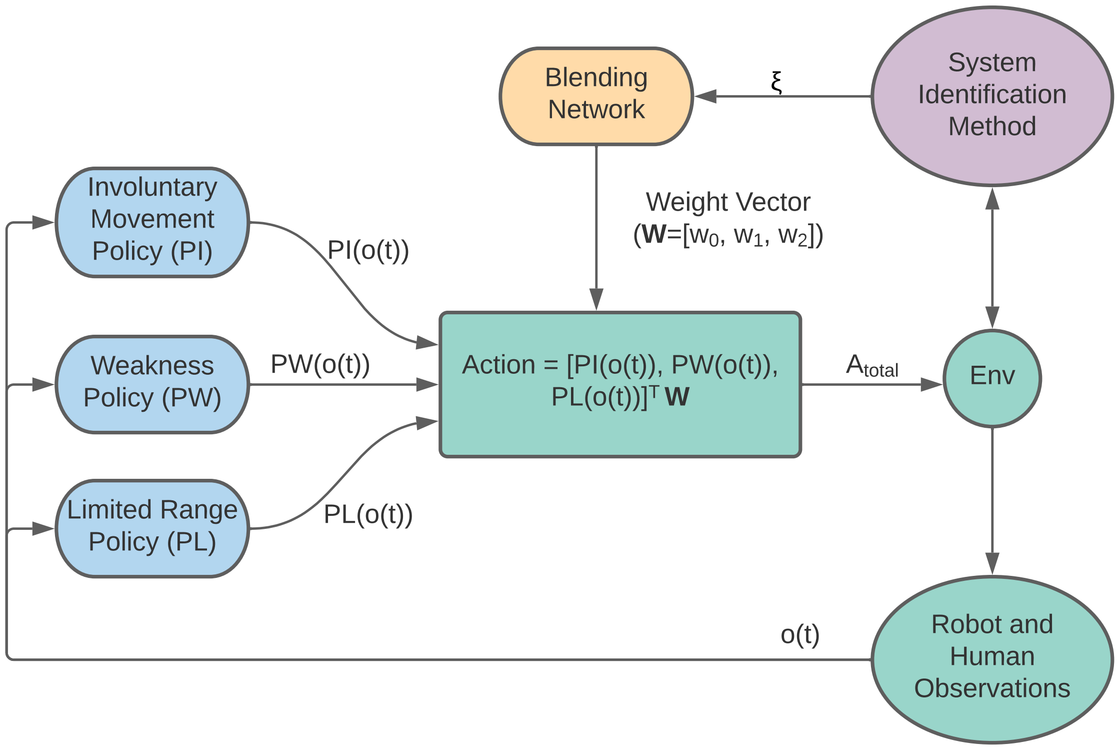
Blending Network
ξ
System Identification Method
Involuntary Movement Policy (PI)
PI(o(t))
Weight Vector (W=[w0, w1, w2])
Weakness Policy (PW)
PW(o(t))
Action = [PI(o(t)), PW(o(t)), PL(o(t))]T W
Atotal
Env
Limited Range Policy (PL)
PL(o(t))
o(t)
Robot and Human Observations

Figure 2.1: Policy blending with system identification for a human-robot collaborative task

2.4 Approach

In this section, we present the details of our solution scheme for solving collaborative tasks with multiple varying parameters. Figure 2.1 show the different components of the proposed approach instantiated for the case study presented in the next section. A set of individual policies learned for individual parameter changes is required. Each individual policy can be trained for a single parameter distribution and the total number of policies scales linearly with respect to the total number of parameters. Training each policy can be accomplished through Domain Randomization techniques. The core components of the proposed scheme - the blending network and system identification technique - are described next.

2.4.1 Blending Network Learning

To create a decoupled policy in which we can solve individual tasks while maintaining robustness to a variety of system parameters, we introduce a blending network in which we consider solely the N system parameters ξ as its state space. This policy then only needs to output the weights $\mathbf{W}$ of its sub-policies at each time step to generate the action for the environment. The sub-policies of this model are trained on a single set of constant system parameters in which a unique environment is identified through previous domain knowledge. We then define the action of the blending network as $a = \frac{1}{N}\sum_{i=0}^{N} w_i \pi_i(s_t)$ where $w_i \in \mathbb{R}$, N is the number of sub policies and $\pi_i(s_t)$ represents the action taken by the sub policy given a state at time t.

2.4.2 Concurrent System Identification

We then combine the blending network with a concurrent system identification scheme to obtain a generalized policy that is robust to different environmental changes, i.e. different combinations of parameter values. To do this, we let the state space of the blending network consist of only the estimated parameters and as such must learn to associate certain parameters with the appropriate sub-policies. In practice, every certain number of training steps, we utilize our system identification method to update the state space of the blending network with a more accurate set of estimate system parameters and continue training. We emphasize that our approach is independent of the system identification method chosen and thus can be tailored based on the available domain knowledge of the environment.

2.5 Experiments

2.5.1 Assistive Gym

For our experiments, we utilize a framework introduced by Erickson et al. (2019) which is an Open AI gym environment for collaborative human and robot interaction. Assistive gym is a realistic physics environment powered by PyBullet that enforces realistic human joints as well as it provides a series of robots for collective tasks. We utilize the Jaco robot as it performed best in the individual itching policies explored in Erickson et al. (2019) for our control task and define the success value of a task as the amount of force applied to the target itch position throughout the entire episode. Each episodes consists of 200 time steps equating to 20 seconds of real-world time.

2.5.2 Training of Sub-Policies

For demonstrating our model, we attempt to solve a collaborative itching task using assistive gym Erickson et al. (2019) where a robot is assisting an impaired human in itching. We consider 3 impairments similar to Clegg et al. (2020) for the human:

(a) Involuntary Movement: The first impairment is involuntary movement which is handled by adding noise normally distributed to the joint actions of the human. For this policy, we sample the noise according to a normal distribution where each joint in the arm has a mean of 0 noise and a standard deviation of 5 degrees of noise.

(b) Weakened Strength: The second impairment involves weakness in the ability for the human to move their arms which is introduced by lowering the strength factor in the PID controller of the joints. This value is also sampled normally with a mean of 0.66 and standard deviation of 0.2 with 1 representing full strength and 0 representing immobility.

(c) Limited Range: Lastly, we consider a limitation in the range of movement for each joint in the arm of the human. Like above, full joint movement is represented by 1 and immobile joints are represented by 0. As such, we sample the limited movement from a normal distribution with mean 0.75 and standard deviation of 0.1.

Initially, we begin by training a single policy for each individual impairment on 2 million time steps (5000 episodes) using Proximal Policy Optimization (PPO) Schulman et al. (2017). We designed a grid-search experiment to obtain the neural network architecture. The best configuration for each network obtained consists of 2 layers of 64 nodes. For all

Table 2.1: Trained policies and their respective observation and action spaces

Policy	Observation Space	Action Space
Involuntary Movement	34 human joint values 30 robot joint values	10 human joint values 7 robot joint values
Weakness	34 human joint values 30 robot joint values	10 human joint values 7 robot joint values
Limit Range of Motion	34 human joint values 30 robot joint values	10 human joint values 7 robot joint values
blending network	Only System Parameters: 1 for Estimate Weakness 1 for Estimated Range Limit 10 for Estimated Involuntary Movement Joints	3 weighted values for blending the policies

single impairment policies, we define a state space of 64 joints between the robot and human along with an action space of 17 joint targets. All policies use the same reward function as defined in Erickson et al. (2019). This reward function considers a weighted combination of the distance of the robot arm to the target itch position, a penalty for large actions, and the contact induced with the itch target: $R(t) = w_{distance} * ||pos_{target} - pos_{robot}|| + w_{action} * ||a(t)|| + w_{force} * F_{target}$ where w_i represents a weight for that term i, $||pos_{target} - pos_{robot}||$ is the Euclidean distance from the arm to the target, $||a(t)||$ is the Euclidean norm of the action vector taken by the robot and F_{target} is the current force applied to the target.

2.5.3 Training of Blending Network

Similar to the sub-policies, we consider the same reward and utilize a PPO model with 2 layers of 64 nodes each for training the blending network. However, for the blending network we train for 400k time steps and the state space only consists of the system parameters. Unlike the sub-policies, our blending network is trained on a human with all three impairments and as such must consider many more cases of how the robot needs to act. By training the policy on a general all three impairments, we allow our blending network to

Table 2.2: Trained policies and their respective observation and action spaces

Method	Policy Blending	State Space
Domain Randomization	No	Trained from Human and Robot Observation Space
UKF	Yes	12 Parameters Estimated By UKF Sampling Real World
Autotuned SPM	Yes	12 Parameters Estimated by Mapping Function of Interaction Between Policy and Real World
Perfect Parameters	Yes	Parameters are Passed as the State Space at the Start of Each Epsiode

become more robust to parameter identification and improve on the notion that training a single policy to handle all three impairments is complex and time-consuming due to sample inefficiency.

2.5.4 Training of Domain Randomization

To train the domain randomization model, we train on a human invoking exhibiting all three impairments. Similar to above, we use PPO with layers of 64 nodes each. However, these impairments are now sampled uniformly as such:

(a) Involuntary Movement: The noise for each joints angle is between $[-10, 10]$ degrees.

(b) Weakened Strength: We consider a weakness coefficient between $[0.25, 1]$.

(c) Limited Range: We consider range limitations between $[.5, 1]$ times the original motion.

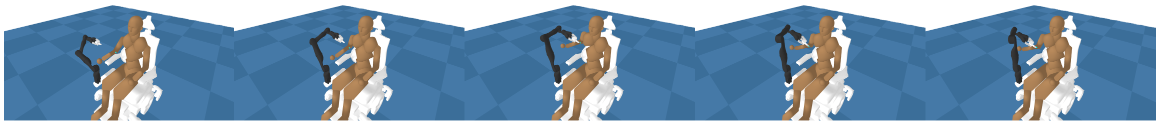
Figure 2.2: Example of our robot completing the itching task even when the human is dis-functionally moving its arm upward

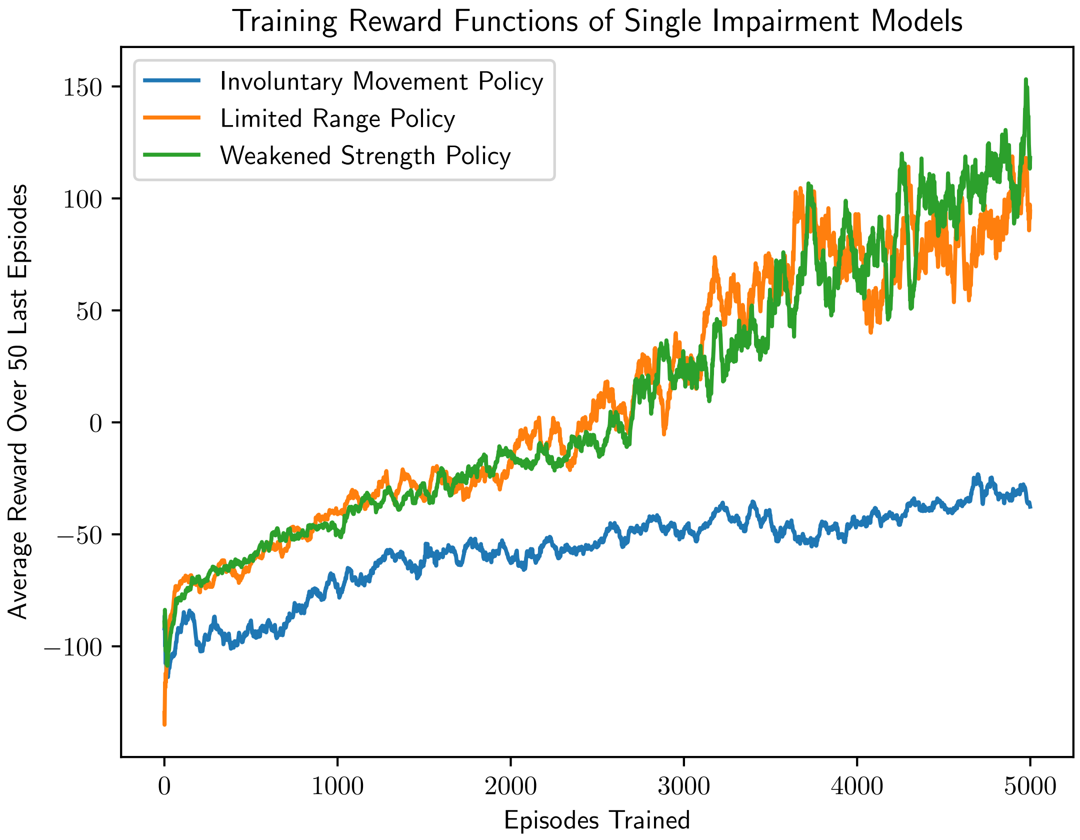

Figure 2.3: Training reward average over 50 episodes for single impairment policies. The rewards here are averaged over the training of 3 seeds.

2.6 Discussion and Results

For our initial sub-policies, we can see that the weakness and limit based policies can achieve a higher reward consistently over the involuntary movement policy in Fig. 2.3. For our blending network, we consider the best performing sub-policies and only train on humans with a combination of all three impairments. As such, in Fig. 2.5 can see that the rewards are much lower than those of the individual policies. Additionally, we notice that there is a significant advantage to using a blending-based policy with system identification over general domain randomization. Furthermore, we can see that the ability to estimate the real-world parameters enhances the policies overall convergence as the auto tuned policies struggle to achieve the same success as the UKF based or the baseline (system parameters are perfectly known to the blending network at each timestep). To further evaluate our policies, we define a testing experiment in which we undergo 100 episodes of our human exhibiting all three impairments in which the impairment values are sampled as above but in conjunction. We still utilize the given system identification method for estimating the state space of the blending network. Fig. 2.4 a) shows a box plot of the performance for the joint parameter variations. We consider experiments in which the human only enacts a single impairment and the results are shown in b), c), and d) of Fig. 2.4 and Table 2.3, respectively.

From the box plots, we can see that we outperform domain randomization for 100 separate episodes. Furthermore, there is a difference between the system identification methods as the UKF and the system fed with the correct parameters outperform the autotuned search approach. As such, we can determine two important things about our approach. First, the policy blending has a significant improvement over general domain randomization in terms of both sample efficiency and performance. Second, our design can successfully employ various types of system identification; however, those identification methods may significantly affect the overall performance of the policy and should be based on the maximum amount of domain knowledge available.

Given this, we must note limitation of our scheme is that we need to develop the sub-policies; however, these theoretically provide us stability and robustness when faced with unknown environments. Additionally, given that these sub-policies can be reused as they are now decoupled from the main blending network, different approaches can quickly be tested and tuned - a problem limiting current domain randomization methods. Furthermore, it is not guaranteed that a linear combination of weights from the blending network and the

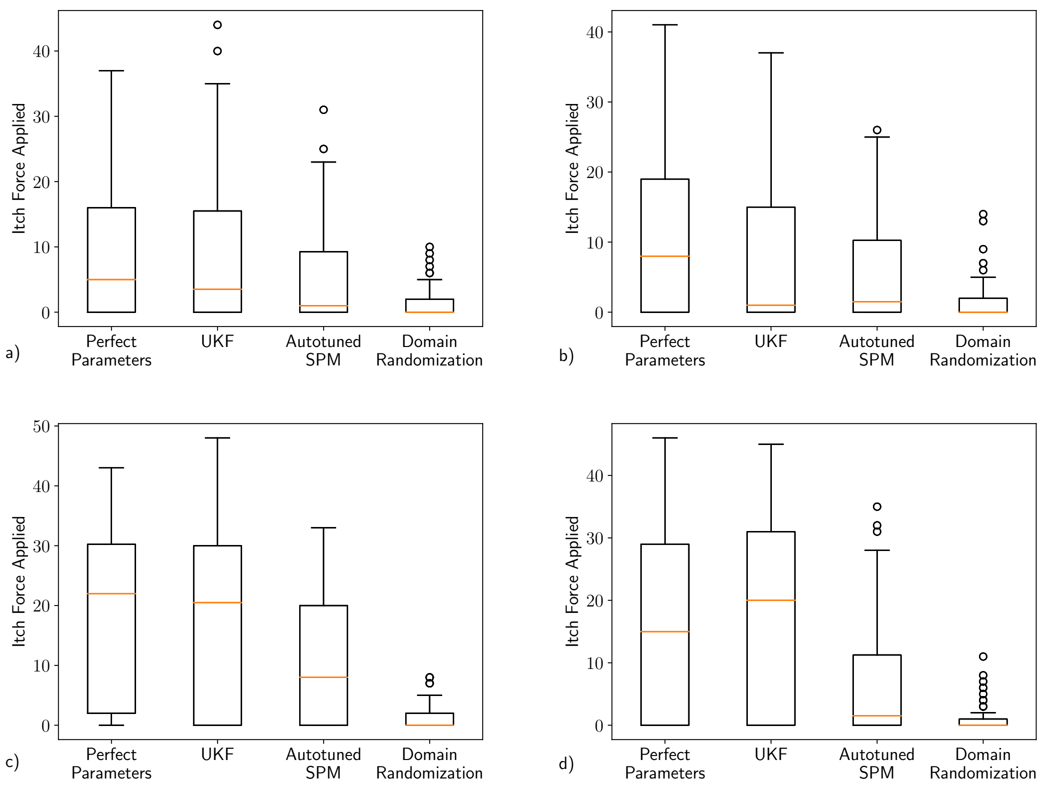

Figure 2.4: Application of the trained policy to a real environment for 100 separate episodes. We use the highest reward policy for each situation. a) Shows the itch force applied when the human has a combination of all three impairments. b), c), and d) show the force applied when the human has a single impairment in the form of limited range, weakness, or involuntary motion respectively.

action spaces of the blending policy is a good approximation other than from an empirical standpoint. Different methods could certainly be used for the policy blending and this is a future direction of exploration.

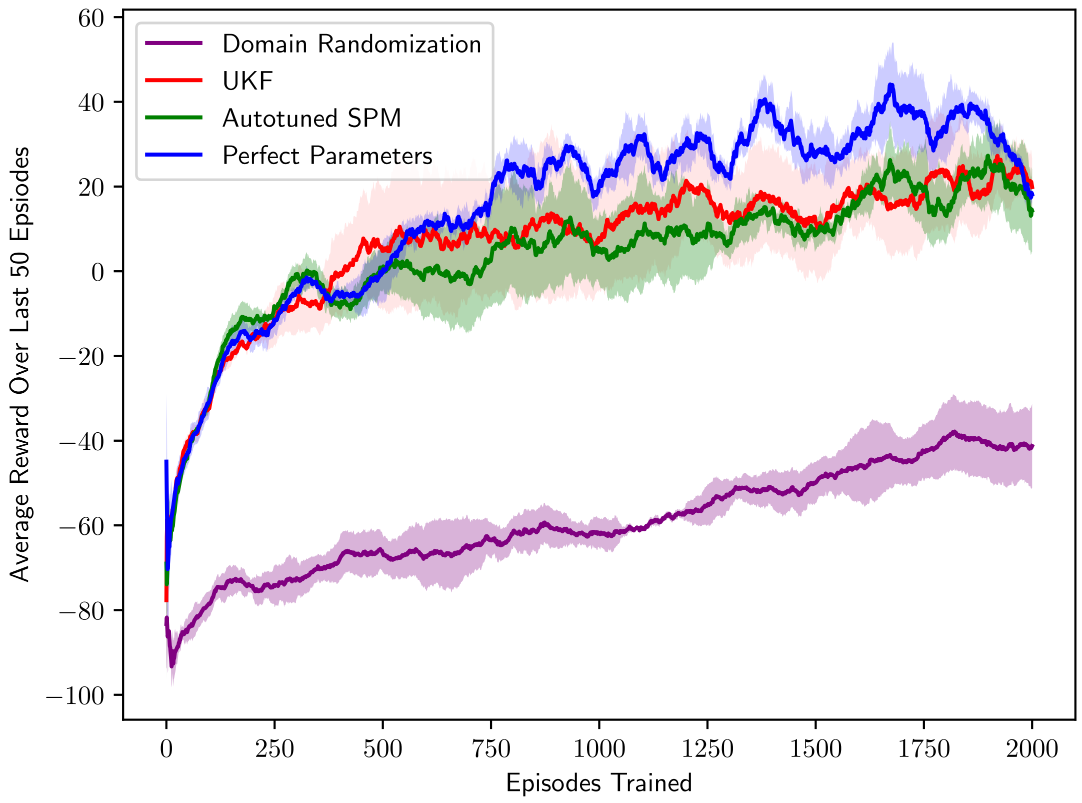

Figure 2.5: Training reward averaged over 50 episodes for our policy exploration methods. The rewards shown here are averaged over the training of 3 seeds

Table 2.3: Mean and STDEV of Each Method Given a Specific Impairment

	Combined Impairments		Involuntary Movement Impairment		Limited Range of Motion		Weakness in Joints	
Method	Mean	STDEV	Mean	STDEV	Mean	STDEV	Mean	STDEV
Domain Randomization	1.33	2.18	0.96	1.94	1.56	2.77	1.07	1.90
UKF	8.68	**10.58**	**18.14**	**14.62**	8.05	10.09	18.13	**14.49**
Autotuned SPM	5.26	7.35	6.34	8.67	5.71	7.35	10.42	10.32
Perfect Parameters	**9.03**	10.23	15.02	14.40	**11.2**	**11.71**	**19.0**	13.81

CHAPTER 3

Hybrid Model Predictive Control and Deep Reinforcement Learning for Fault Tolerant Control of Octorotors

This chapter is adapted from a conference paper published at Systems of Fault Tolerant Control. Bhan et al. (2021)

3.1 Introduction

Fault tolerance is imperative for real world cyber physical systems that can be altered, damaged, and even destroyed by their environment. In this work, we will present a fault tolerant control scheme using data-driven methods for UAV's (Octorotors) and showcase our schemes feasibility for large magnitude faults.

3.2 Background on Fault Tolerant Control

Fault tolerant control (FTC) approaches are split into two distinct solutions: active and passive. Active solutions require fault detection online that informs the controller about the characteristics and magnitudes of any potential faults. Passive solutions involved predefined faults in which there is less computational complexity; however, they are constrained by the designers ability to predict fault instances.

Furthermore, there are two main approaches to fault controllers. There are model-based and data-driven methods depending on the controller architecture. Model-based controllers are designed based off of the physical implications due to the fault dynamics. Meanwhile, data-driven controllers learn directly from the system data. The limitation of model-based controllers come from their assumption about the UAV dynamics and they lack an ability to handle unforseen faults. On the contrary, data-driven methods generally rely on a large amount of data to capture the non-linear dynamics of the system and thus are not always easy to develop.

The performance of model-based FTC methods depends on having accurate and comprehensive first-principle models. Complex models require enhanced control methods compromising the robustness of the controller. Data-driven FTC methods allow developing complex control strategies but they are sample inefficient requiring numerous experiments to achieve a satisfactory performance. As such, deep reinforcement learning (DRL) approaches attempt to minimize this sample inefficiency and have become very popular for solving continuous control tasks in the last few years Dulac-Arnold et al. (2019).

3.3 Related Work

Reinforcement learning has been applied for fault tolerant control in a series of different applications. Naug et al. (2020) Zhang and Gao (2020) Zhang et al. (2017). However, to the authors knowledge, it has never been applied to an octorotor. Moreover, it has been applied in numerous ways to a quadrotor. For example, Fei Et. Al. had proposed an approach that modifies the torque vector for recovering a quad rotor in the case of cyber attacks. Fei et al. (2020) Additionally, Ahmed Et. Al. showcased a hybrid RL FTC scheme for degraded state operation, but not a complete fault. Ahmed et al. (2020). Theoretically, an octorotor should be able to handle larger faults than that of a quadrotor due to its 8 motor shape. Therefore, it would be worthwhile to see if a data-driven FTC scheme is possible for Octorotors. Furthermore, Carlucho et al. (2020) has shown that altering the PID parameters of a model based scheme is very viable in the robotic setting. Given this, we will attempt to apply a PID altering approach to a UAV where we minimize the ”overshoot” of the initial MPC.

3.4 Formalism

3.4.1 Reinforcement Learning

Reinforcement learning (RL) aims to solve an optimal control problem through neural network-based methods. The control law is refined through the continuous interactions of a learning agent with an environment Sutton and Barto (2018). The control problem is formalized through the following definition: A **Markov decision process** is defined by a four tuple: $M = \{S, A, T, R\}$, where S represents the set of possible states in the environment. The transition function $T : S \times A \times S \rightarrow [0,1]$ defines the probability of reaching state s' at $t+1$ given that action $a \in A$ was chosen in state $s \in S$ at *decision epoch, t*

$$T = p(s'|s,a) = Prob\{s_{t+1} = s'|s_t = s, a_t = a\} \tag{3.1}$$

. The reward function $R : S \times A \rightarrow \Re$ estimates the immediate reward $R \sim r(s,a)$ obtained from choosing action a in state s. The objective of the agent is to find an optimal policy π^* that maximizes the following criteria $\forall s \in S$:

$$V^{\pi^*}(s) = \max_{\pi \in \Pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s, a_t = \pi(s)\right], \tag{3.2}$$

where $V^{\pi} : S \rightarrow R$ is called value function and it is defined as

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s\right], \forall s \in S, \tag{3.3}$$

where $0 < \gamma \leq 1$ is called the discount factor, and it determines the weight assigned to future rewards. The agent's objective is to find the policy that maximizes the expected sum of reward. Obtaining a policy with optimality guarantees requires the following two conditions to be satisfied

1. $|R \sim r(s,a)| \leq C < \infty, \forall a \in A, s \in S$

2. T and R do not change over time.

Systems subjects to faults undergo changes that cause their dynamic model, represented by the transition function T, to change over time Dulac-Arnold et al. (2019). Therefore, learning direct control with DRL for fault tolerance is not theoretically feasible. Given this, we propose a fault adaptive control scheme that avoids using DRL for direct control, but combines model based and DRL in the following sections.

3.5 Octorotor Design

3.5.1 UAV Model

We consider an octocopter dynamics model based on Newton-Euler equations of motion for a rigid body. octocopter's cascade control scheme is shown in Figure 1. This control approach allows for stabilization of the position and orientation of the octocopter with respect to a trajectory. A set of three PID controllers adjust the vehicle attitude, and a different set of three PID controllers adjust the position variables, together forming nested feedback loops.

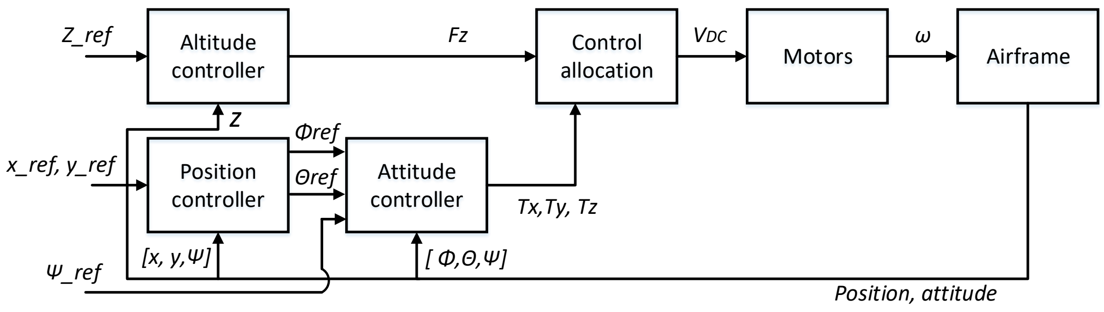

Figure 3.1: Cascade Control scheme for the octocopter

The reference trajectory is defined in terms of position and yaw angle $[x_t, y_t, z_t, z_t, \psi_t]$. The Altitude PID controller generates the required force in the z direction. The position PD controllers estimate, based on the current position of the vehicle and yaw angle, the reference for the pitch (θ) and roll (ϕ) angles. The attitude PD controllers generate the required torque in each direction. The control allocation block transforms the torques and forces into a reference voltage for each motor of the octocopter. Finally, each motor generates angular velocity according to brushless dc motor dynamics and we cap the input voltage to 11.1V to represent a realistic motor scenario. Osmić et al. (2016)

Table 3.1: Octocopter Parameters

Parameter	Value
Mass	2
Inertia Coefficient X	0.0429 kgm^2
Inertia Coefficient Y	0.0429 kgm^2
Inertia Coefficient Z	0.0748 kgm^2
Length	1m
Rotor Thrust Constant	$8.54858 * 10^{-6}$ Ns^2/rad^2
Drag Constant	$1.3678 * 10^{-7}$ Nms^2/rad^2
Nominal Motor Resistance	0.2371 Ω
Electrical Motor constant	0.0107 Vs/rad
Mechanical Motor Constant	0.0107 Nm/A

3.5.2 Designing Realistic Faults

The design of realisitc faults is of large importance to accurately represent the scheme in the real world. As such, we consider motors which are mechanically suspectible to degredaiton in the form of bearing wear and corrosion. To represent these faults, we chose to modify the resistance parameter in the following motor representation. We consider the following:

$$\dot{\omega} = \frac{1}{J_m}(K_e i_c - T_{load} - D_f \omega - T_f), i_c = \frac{1}{R_{eq}}(v_{DC} - K_e \omega_i), \tag{3.4}$$

where $R_{eq} = \frac{2}{3}\sum_{j=1}^{3} R_j$ is the equivalent electric resistance of the coils, K_e is the back electromotive force constant, ω is the angular velocity, T_f is the static friction torque, D_f is the viscous damping coefficient, J_m is the inertia along the z axis, v_{DC} is the input voltage control signal, i_c is the current demanded, and T_{load} represents the torque load generated by the propellers. When we increase the R_{eq} value, we are enhancing the resistance in the circuit which could be caused by dust or the aformentioned erosions. This will result in a loss of effectiveness for each motor. In this work, we consider single and dual motor faults according to Figure 2's motor location.

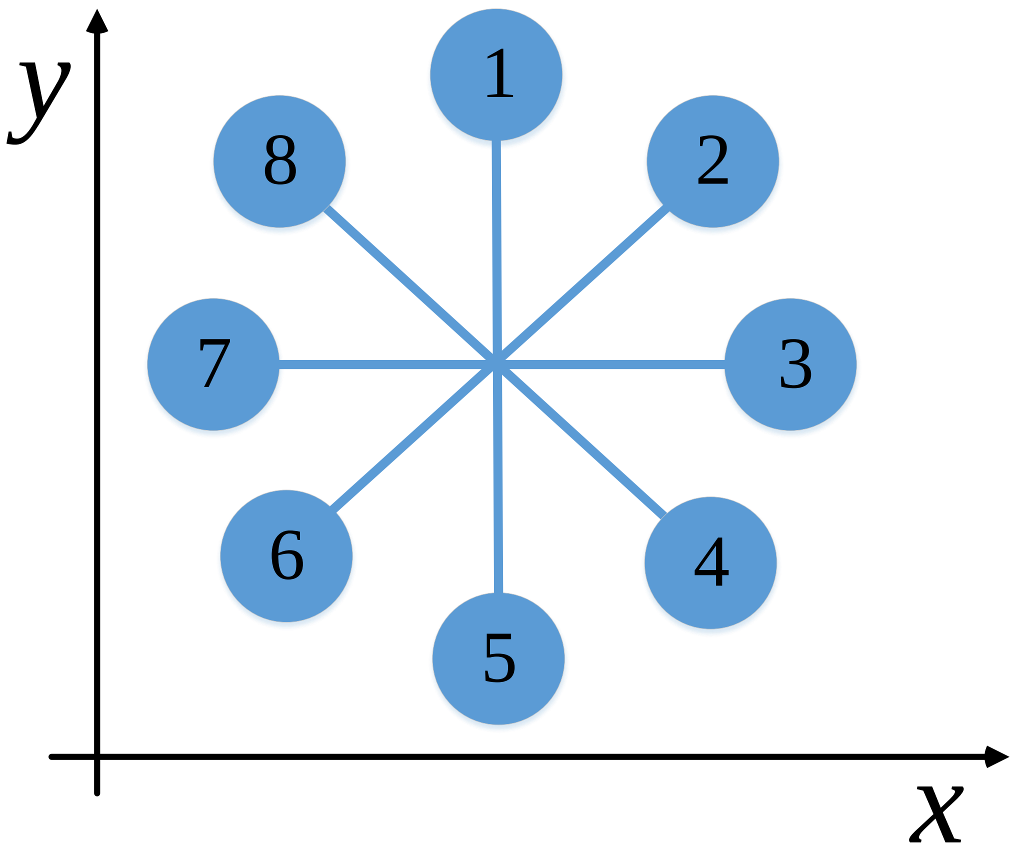

Figure 3.2: Octocopter motor fault configuration

3.6 Proposed Framework

In this section, we will formulate our FTC approach and discuss it's intuition. We propose the combination of model-based control schemes with DRL as a solution to the FTC problem presented in Figure 3. Model-based control methods like Proportional Integral Derivative (PID) Controllers remain dominant in real world industry applications thanks to their simple structure, ease of implementation, and wide variety of tuning methods Borase et al. (2021). Nonetheless, traditional tuning methods for PID control do not account simultaneously for multiple input-output systems and multiple PID controllers. We propose to extend the scheme proposed in Carlucho et al. (2020) to accommodate faults assuming they are not catastrophic- the system can continue to operate in a degraded manner and performance can be recovered to some extent by updating the PID parameters.

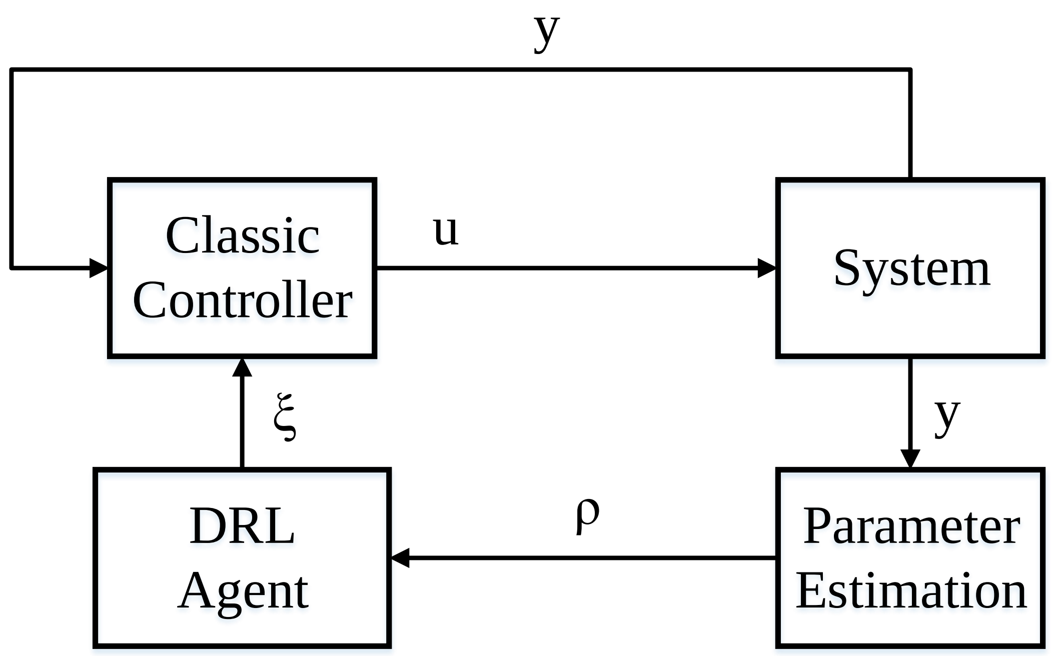

Figure 3.3: Fault Adaptive Control framework

The core of the proposed approach relies on the combination of parameter estimation techniques with DRL. We propose to update the PID controller when the value of the parameter(s) associated with faults affect the control performance. We assume that the faults can be accurately measured through widely studied estimation techniques like the Unscented Kalman Filter and the Particle Filter Daigle et al. (2012). The estimated parameters are then used as inputs to the DRL agent ($s = \rho$) and the action of the agent consists in a new set of parameters for the controller(s) ($a = \xi$).

3.7 Experiments and Results

3.7.1 Experimental Design

In this work, we considered training the DRL agent to learn how to adapt the parameters of PD position controllers. This implies a four-dimensional action space $a = \{K_p^x, K_d^x, K_p^y, K_d^y\}$. Additionally, the state space consists of the position vector, velocity vector, Euler angles, and their angular velocities $s = \{x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}\}$. The only information received by the agent is the motor resistance estimated and the reward function is defined by $R = (10 - error)/10$ where *error* is the Euclidean distance calculated between the position of the octocopter and the reference trajectory. We defined 10 meters as the maximum deviation allowed from the reference trajectory and we re-scale the reward between 0-1 as suggested for continuous control tasksHenderson et al. (2018). Other error functions were considered such as $R_t = max(0, 1 - ||\mathbf{x} - \mathbf{x}_{ref}||) - C_\theta ||\theta|| - C_\omega ||\omega||$ where $\mathbf{x}$ is the position vector, ω is the angular velocity and θ is the euler angle vector and C_θ and C_ω are constants that penalize the octorotor for spinning. However, we found that this error function did not perform well as the constants were hard to tune and the octorotor was not robust to noise.

We considered a change in the reference from $(x = 0, y = 0)$ to $(x = 5, y = 5)$ as the trajectory tracking task for simplification purposes and assuming that the resulting architecture will scale well as long as the changes in the reference are smaller than the one experience during training. Different fault magnitudes must be experienced by the agent to learn how to adapt the position controller parameters. However, we noticed that randomly selecting the fault magnitude for each episode resulted in no convergence. We thus defined a curriculum learning approach where we first expose the agent to the lower bound of the fault magnitude until it converges and then we generate for each episode with probability of 0.5 a fault with maximum magnitude. In this way, we avoid the catastrophic forgetting problem for the agent.

We demonstrate the approach on two different experiments. For both experiments, we use PPO as defined previously with the following parameters:

In the first experiment, we explore a single fault on motor 3 as defined in figure 2. We initially begin training the DRL agent on a 3x fault (3 times the nominal value of the parameter) and after convergence is reached, we then introduce a larger 8x where at the beginning of the episode, a choice of the fault is made between 3x and 8x with equal probability. We then test our approach by comparing the trajectory between the nominal PD

Table 3.2: PPO Parameters

Parameter	Value
Optimizer	ADAM
Batch Size	64
Discount Factor	0.99
Initial Learning Rate	0.0003
Time Steps of an Episode	2000
Total Number of Episodes	3000

controller and the DRL scheme when the DRL controller is given the exact magnitude of the 8x fault. Furthermore, we then evaluate the robustness of our controller by introducing variance in the estimated parameter such that the value given to the controller is biased. We sample the parameter from a normal distribution with mean value equal to the true value and a deviation of 0.5x the nominal resistance.

For our second experiment, we explore a dual motor fault on motor 5 and 6 as defined in figure 2. However, our controller is now given two values for the estimated motor resistances instead of the one above. For training, we set both motors to an equal 2x fault and allow the DRL agent to learn until convergence is reached. Then, following the same method as proposed above, we introduce a large fault of 4x in both motors such that a choice is made between the smaller and larger fault at the beginning of each episode. Following this, we then test our approach by comparing the trajectory where both motors have a 4x fault. However, to explore the robustness of our approach, we then explore setting motor 6 to a 4x fault and allow the fault of motor 5 to vary. We then introduce variance in the state estimator of both motors such that parameters fed are normally distributed with mean equal to the true value and a standard deviation of 0.5x the nominal resistance. This allows us to explore the effects of large state estimation errors in both of our motors. Furthermore, it also allows us to explore the realistic case of different fault magnitudes on the two different motors. In this exploration, it's worth noting that we are considering multiplicative faults where changes in the motor dynamics may affect the dynamics of the controllers and control allocation requirements. This results in a more complex scenario to handle and showcases the value of using a DRL agent for online adaptation. Furthermore, it is worth highlighting that we do not consider additive faults as they have been widely studied subject Zheng and Li (2019) and most likely will not need a DRL-based tuning approach.

3.7.2 Results For Single-Motor Faults

For our single-motor fault experiment, we can see in figure 3.4 that our reward function clearly converges by step 1000 and when the larger fault is introduced, we have a dip in performance, but then converge again by the time training ends. As such, this demonstrates that our scheme first minimizes the error with a small fault, and then builds on it's initial learning by minimizing the error of both the small and larger fault. From figures 3.5 and 3.6, we can see that our method outperforms the nominal PD controller in terms of a 8x fault for a single episode. In the X-direction, it is clear that our controller does not overshoot and converges closer to the reference value while the nominal PD controller significantly overcompensates and contains more error in it's convergence. Meanwhile, in the Y-direction, both controllers perform similarly due to the positioning of the motor fault. Furthermore, in 3.7, we can see that while the median error of smaller faults are outperformed by the nominal PD, when we move to larger faults, our approach can still accurately achieve convergence with a significantly lower error when compared to the nominal PD controller. As such, a parameter estimator can be used to measure the active states of each motor and when a large fault is detected, our controller can be invoked to ensure the stability of the octocopter. Additionally, figure 3.7 demonstrates that our controller is robust to large instability as only a marginal set of outliers ever surpass the PD controller's error in 7x, 7.25x, and 8x faults. However, this is just a 100 trial run at each fault magnitude and thus is only an empirical monte-carlo robustness figure and has *no* robutsness guarentees.

3.7.3 Results of Dual Motor Faults

Considering a multi-motor fault carries larger impact on the octocopter's performance, we only explore multi-motor faults up to 4x. Similar to above, we can see in figure 3.8 that our

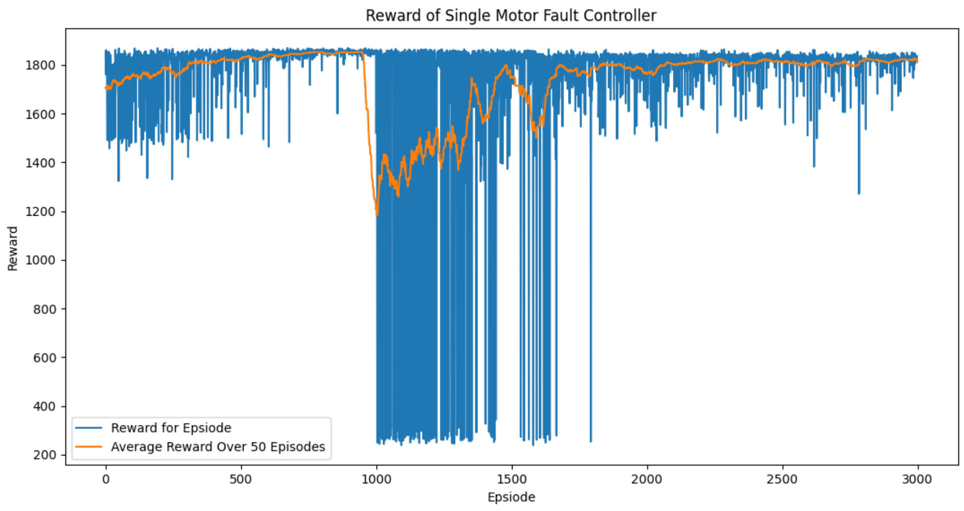

Figure 3.4: Reward Function for Training the DRL Agent on a Single Motor 3 Fault

controller first learns to handle the smaller dual fault and then struggles initially when the larger fault is introduced, but ultimately converges to minimize the larger and small fault errors. As such, in a single episode's trajectory, we can see in figure 3.9 that both the nominal controller and the reinforcement learning scheme converge to the correct X position. However, we can see that the nominal controller performs poorly as it initially becomes unstable and then overcompensates for the fault while our approach accurately compensates to ensure a much faster convergence. Furthermore, in the Y-direction show by figure 3.10, we can see that the reinforcement learning approach slightly overcompensates, but still converges equally fast as the nominal controller. We can see that the approach shown is not perfect in the Y-direction, but the improvements in the X-direction significantly outweigh the marginal overshoot in the Y-direction. Similar to figure 3.7, we can also see in 3.11 that the hybrid based control outperforms the sole PD controller at large faults when the PD controller begins to deteriorate in stability. However, in this case, we set a single motor to a 4x fault and vary the fault of the second motor. Despite the variance of the second motor's fault, we see that the controller is robust to parameter estimation noise as even the outliers in the dual motor fault experiment have smaller error than that of the nominal controller.

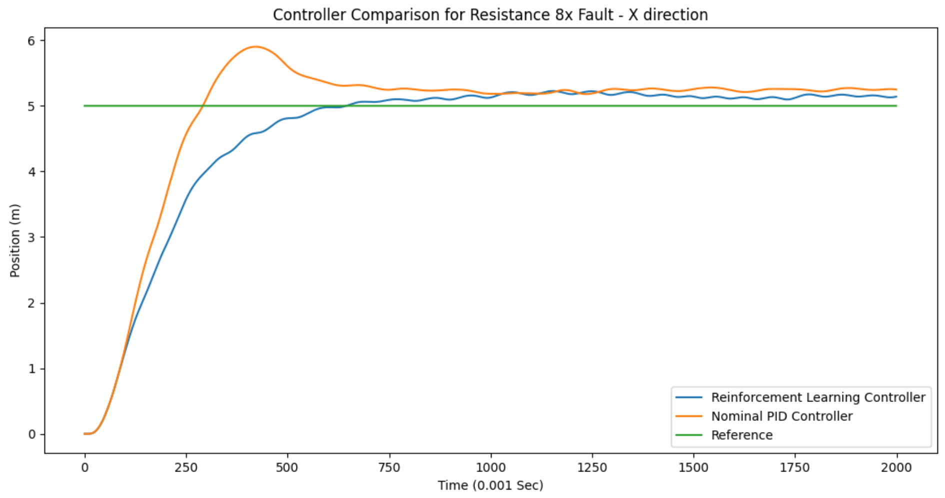

Figure 3.5: Comparison of X-trajectory Between PD Control and Hybrid Scheme for 8x Fault on Motor 3

Additionally, this demonstrates that our approach is viable for different magnitude faults in each motor as figure 3.11 demonstrates an equal or better performance then the PD control scheme at almost every single fault magnitude.

3.8 Limitations and Future Work

We present a hybrid MPC and RL scheme for fault tolerant control. We evaluated our scheme on a simple trajectory task and identified improvement over the nominal controller for large magnitude faults. In this section, we will quickly discuss the assumptions, limitations and potential future work. For our simulation we make a series of key assumptions:

- The proposed model is representative of a real world Octorotor.
- The integrator does not introduce numerical instability in cases where we would be able to fly.
- Our simple trajectory task from 0, 0 to 5, 5 is applicable to a real world trajectory as it will consist of a series of waypoints with smaller distance between the points.

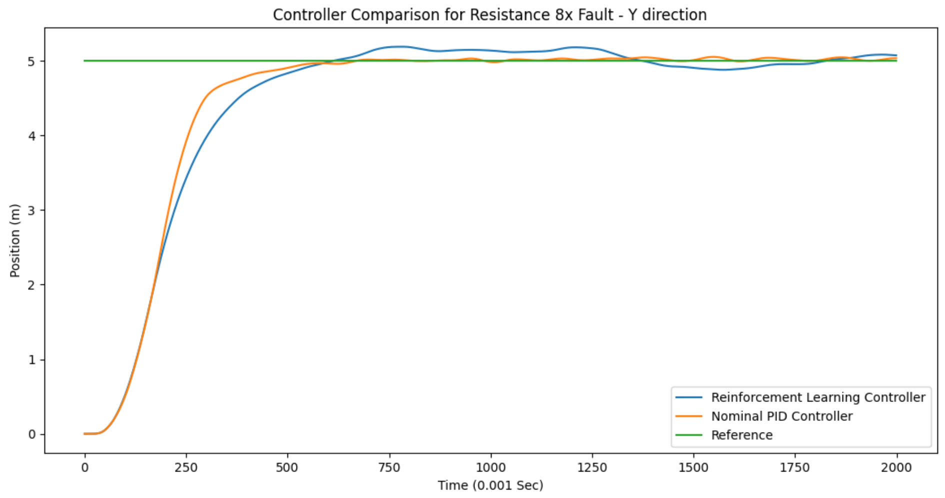

Figure 3.6: Comparison of Y-trajectory Between PD Control and Hybrid Scheme for 8x Fault on Motor 3

Here, we can see that our assumptions can already be relaxed as for example it would be interesting to see in the future if we can train our approach on complex trajectories. Furthermore, robustness is certainly a limitation of our work as we have no formal guarantees that the learning controller will stay stable. As such, it is imperative that we stress our work is only viable in situations where large magnitude faults would otherwise deem the nominal approach unstable. Furthermore, we have limitations in the fact that our approach was only tested in simulation. To create the simulation, we needed to use a variable time-step integrator and thus we are introducing some variance in our simulation with respect to the real world. Naturally, future work would then be to attempt our scheme on a small scaled real-world octorotor. Beyond this, there are also limitations in training resources and computation power for the network size and it would be interesting to see if we can develop a scheme that performs as well under nominal conditions and therefore always include the control rather than having to ”switch” it on under large faults. Lastly, it would be very interesting to see if a formal method approach could be developed for robustness guarantees as without this, we cannot utilize our controller in the real world.

Given these, there is a large amount of research to be done with UAVs and fault tolerant

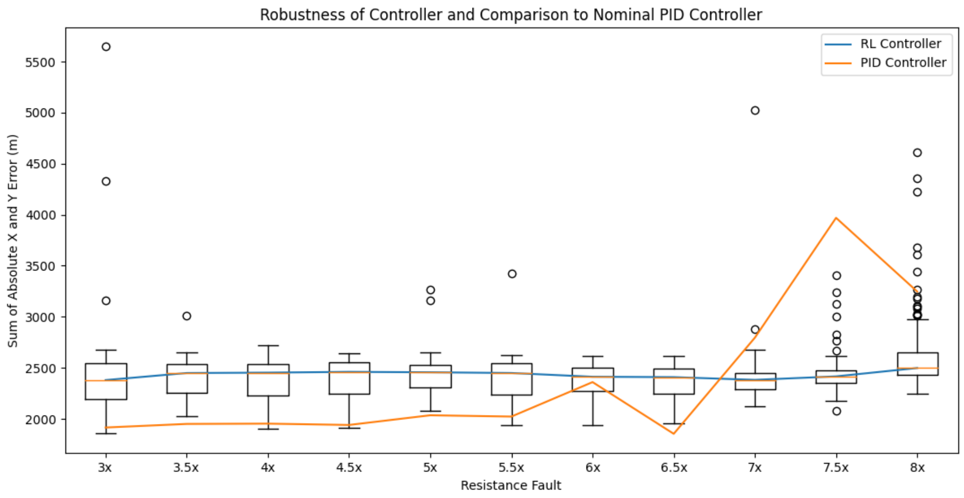

Figure 3.7: Robustness of Hybrid Scheme for Single Motor 3 Fault. The robustness is shown over a box-plot where the circles are outliers and the edges of the box represent the first and third quartiles of the data.

control. However, this work takes us one step closer to utilizing data-driven methods with UAVs and thus should be built upon for future exploration and research.

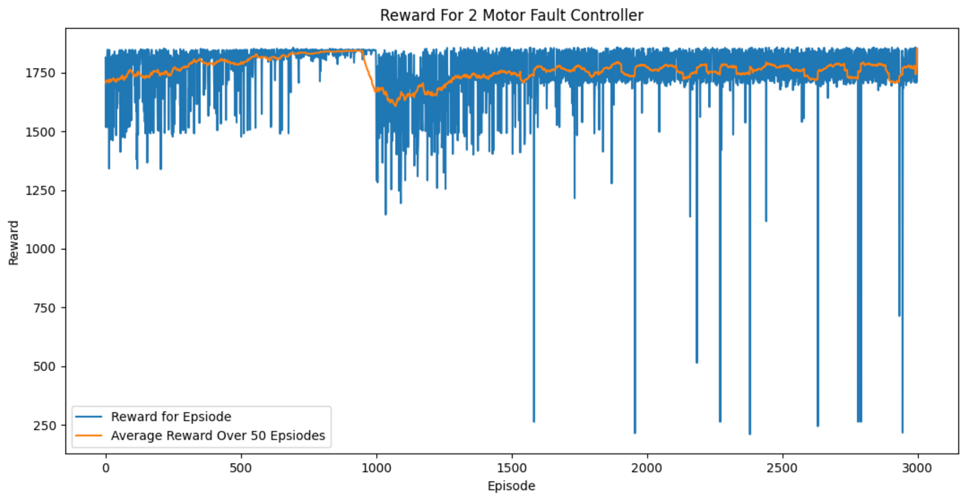

Figure 3.8: Reward Function for Training the DRL Agent on a Dual Motor 5 and 6 Faults

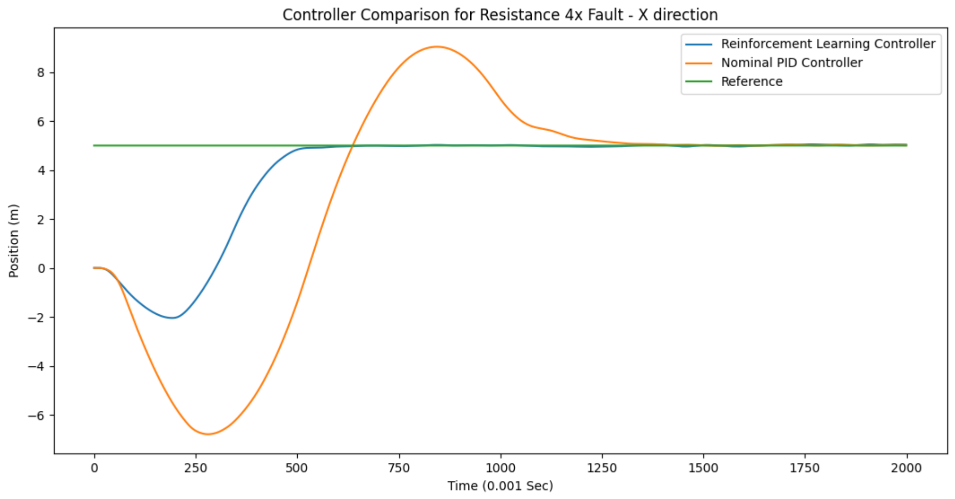

Figure 3.9: Comparison of X-trajectory Between PD Control and Hybrid Scheme for 4x Faults on Motor's 5 and 6

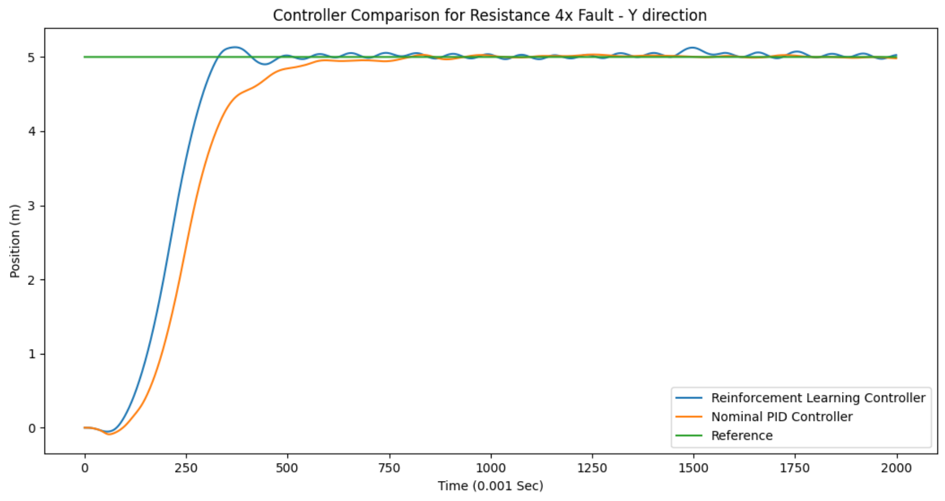

Figure 3.10: Comparison of Y-trajectory Between PD Control and Hybrid Scheme for 4x Faults on Motor's 5 and 6

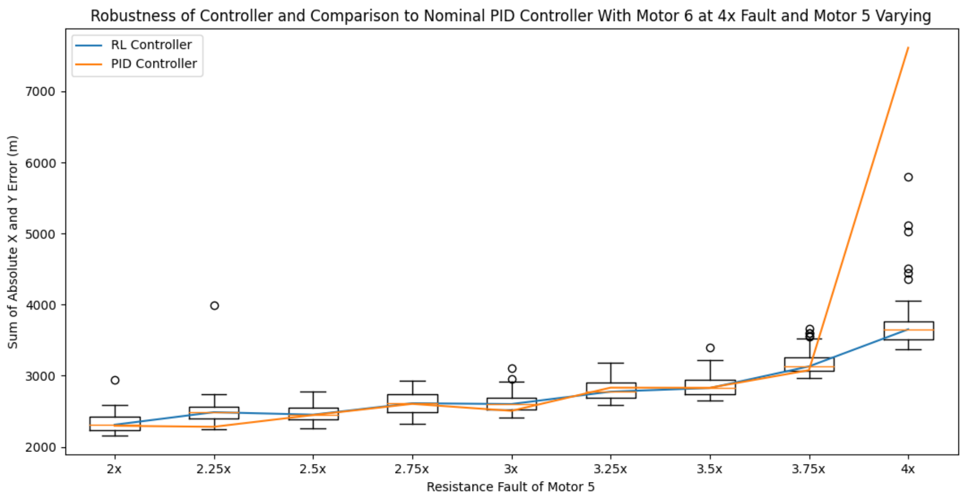

Figure 3.11: Robustness of Hybrid Scheme for Dual Motor 5 and 6 Faults

CHAPTER 4

Conclusion

Data-driven approaches for nonlinear robotics control are viable and generalize better to the real world than some physics based approaches. Particularly, we demonstrate this in two vastly different cases of robotics control. We first demonstrate a hierarchical reinforcement learning architecture that combines system identification and policy blending different human-assitive environments. We exhibit augmented sample efficiency over standard domain randomization and are able to regularly complete an itching task across a series of different human impairments.

Additionally, we demonstrate a different data-driven scheme for fault tolerant control of octorotors. We showcase our approach on both single and dual motor faults successfully navigating the ocotrotor over a PD control method. Furthermore, we validate the robustness of our approach by varying the system parameters fed to the controller and showcase very little change in controller behavior.

Lastly, we identified a series of oppurtunities for improving on our foundation. In particular, we considered applying a policy blending approach for fault-tolerant control in which we do not need the time-delay required by system diagnostic checks. Furthermore, our hierarchical reinforcement learning architecture shows great promise for multi-impairment systems that can be combined in nontrivial ways. Lastly, we mentioned that the nature of policy blending does not require linearity and other approaches for combining parameter weights may be worthwhile to explore. Ultimately, this work lays a foundation for applying data-driven models to generalized nonlinear control processes in a sample efficient manner.

References

Ahmed, I., Quiñones-Grueiro, M., and Biswas, G. (2020). Fault-tolerant control of degrading systems with on-policy reinforcement learning.

Bhan, L., Quiñones-Grueiro, M., and Biswas, G. (2021). Applying reinforcement learning for fault tolerant control in octorotors. *Systems of Fault Tolerant Control (Systol).*

Bhan, L., Quiñones-Grueiro, M., and Biswas, G. (2022). Concurrent policy blending and system identification for generalized assistive control. *International Conference of Robotics and Control(ICRA).*

Borase, R. P., Maghade, D. K., Sondkar, S. Y., and Pawar, S. N. (2021). A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2):818–827.

Carlucho, I., De Paula, M., and Acosta, G. G. (2020). An adaptive deep reinforcement learning approach for MIMO PID control of mobile robots. *ISA Transactions*, 102:280–294.

Clegg, A., Erickson, Z., Grady, P., Turk, G., Kemp, C. C., and Liu, C. K. (2020). Learning to collaborate from simulation for robot-assisted dressing. *IEEE Robotics and Automation Letters*, 5(2):2746–2753.

Daigle, M., Saha, B., and Goebel, K. (2012). A comparison of filter-based approaches for model-based prognostics. In *2012 IEEE Aerospace Conference*, pages 1–10.

Dragan, A. and Srinivasa, S. (2013). A policy blending formalism for shared control. *International Journal of Robotics Research*, 32(7):790 – 805.

Du, Y., Watkins, O., Darrell, T., Abbeel, P., and Pathak, D. (2021). Auto-tuned sim-to-real transfer.

Dulac-Arnold, G., Mankowitz, D. J., and Hester, T. (2019). Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901.

Erickson, Z., Gangaram, V., Kapusta, A., Liu, C. K., and Kemp, C. C. (2019). Assistive gym: A physics simulation framework for assistive robotics.

Fei, F., Tu, Z., Xu, D., and Deng, X. (2020). Learn-to-recover: Retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks. In *IFAC*, pages 7358–7364.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep Reinforcement Learning That Matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Kaspar, M., Osorio, J. D. M., and Bock, J. (2020). Sim2real transfer for reinforcement learning without dynamics randomization. *CoRR*, abs/2002.11635.

Ljung, L. (1986). *System Identification: Theory for the User*. Prentice-Hall, Inc., USA.

Martinsen, A. B., Lekkas, A. M., and Gros, S. (2020). Combining system identification with reinforcement learning-based mpc. *IFAC-PapersOnLine*, 53(2):8130–8135. 21st IFAC World Congress.

Matas, J., James, S., and Davison, A. J. (2018). Sim-to-real reinforcement learning for deformable object manipulation. *CoRR*, abs/1806.07851.

Mehta, B., Diaz, M., Golemo, F., Pal, C. J., and Paull, L. (2019). Active domain randomization.

Narita, T. and Kroemer, O. (2021). Policy blending and recombination for multimodal contact-rich tasks. *IEEE Robotics and Automation Letters*, 6(2):2721–2728.

Naug, A., Quiñones-Grueiro, M., and Biswas, G. (2020). A relearning approach to reinforcement learning for control of smart buildings. *CoRR*, abs/2008.01879.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. (2019). Solving rubik's cube with a robot hand.

Osmić, N., Kurić, M., and Petrović, I. (2016). Detailed octorotor modeling and pd control. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002182–002189.

Peng, X., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018a). Sim-to-real transfer of robotic control with dynamics randomization. pages 1–8.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018b). Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Shamrao, Padmanabhan, C., Gupta, S., and Mylswamy, A. (2018). Estimation of terramechanics parameters of wheel-soil interaction model using particle filtering. *Journal of Terramechanics*, 79:79–95.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. (2018). Sim-to-real: Learning agile locomotion for quadruped robots.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world.

Wan, E. and Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158.

Wiesemann, W., Kuhn, D., and Rustem, B. (2013). Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183.

Zhang, D. and Gao, Z. (2020). Fault tolerant control using reinforcement learning and particle swarm optimization. *IEEE Access*, 8:168802–168811.

Zhang, D., Lin, Z., and Gao, Z. (2017). Reinforcement-learning based fault-tolerant control. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 671–676.

Zheng, J. and Li, P. (2019). Fault tolerant control of actuator additive fault for quadrotor based on sliding mode observer. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 5908–5913.