Intelligence for Morphing Aircraft

by

Kevin P.T. Haughn

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2023

Kevin PT. Haughn

kevpatha@umich.edu

ORCID iD: 0000-0002-8219-8983

For you

ACKNOWLEDGEMENTS

The work presented in this dissertation would not have been possible, or at least not nearly as enjoyable, without the support of many friends, family, and colleagues who have impacted my life in a variety of ways. Much of that impact occurred long before I began pursuing my doctoral studies, and for this reason the following list of thanks is not a complete representation of the people in my life for whom I am grateful.

First and foremost, thank you to my advisor Dr. Inman. Thank you for agreeing to meet with me at the end of my undergraduate studies, and seeing my pole vault career as a benefit as opposed to a detriment. You offered me the freedom to pursue my interests while pushing me out of my comfort zone to grow as an independent researcher. More importantly, you exemplify the type of academic and mentor that I strive to become. With your success you actively lift others up. At each conference and grant review you exude positivity that is noticed by all in attendance. I get a sense of pride with each smile I receive when introducing myself as your student. Your kindness, humility, and brilliant sense of humor are only surpassed by your ability to prioritize the well being of your students. This is evident in the lab environment you cultivate, and appreciated throughout the academic community.

Much of my support came from within the AIMS lab, and I am grateful for each member and their support from studying for the preliminary exam through writing and defending this dissertation. A few of my labmates have been particularly helpful during this journey. Lawren Gamble, thank you for your invaluable mentorship during the formative years of my research career. You introduced me to the world of research, and showed me that it can be both rewarding and fun. Christina Harvey and Piper Sigrest, I learned a lot from and with you two, and I am thankful we embarked on this journey together. You filled a traditionally stressful experience with plenty of laughter and good times. I look forward to future collaborations and seeing where our careers take us.

I also have many people to thank outside of the academic environment. My parents, Melissa and Ken Haughn, who fostered my creativity and problem solving through childhood and continue to support me in all goals I choose to pursue. Thank you to my grandparents, Ardie and Ed Haughn, and Pat and Jim Howard, for the endless support and giving me food and a place to rest while completing this dissertation. Thank you to my friends from back home, Travis Vanderveen and Nick VanDeWalker, for pushing me to pursue my goals when I was unsure where life would take me. Thank you to my friends who made Ann Arbor a home for me during my undergraduate and graduate studies, including Nick, Emma, Tori, Plow, Dan, Shean, and Hannah. You guys put the "life" in my work-life balance, allowing me to push through the particularly difficult moments in this process. Finally, I need to thank my sister Aubrey, who has been my best friend and greatest supporter since day one.

This work supported in part by the National Science Foundation under grant no 1935216 and in part by the US Air Force Office of Scientific Research under a grants FA9550-16-1-0087, titled "Avian-Inspired Multifunctional Morphing Vehicles", FA9550-19-1-0213, titled "Brain Inspired Networks for Multifuncional Intelligent Systems in Aerial Vehicles", and FA9550-21-1-0325, titled "Towards Neural Control for Fly-by-Feel Morphing", all mon-itored by Dr. BL Lee.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

UAV Uncrewed Aerial Vehicle

AC Aerodynamic Center

CG Center of Gravity

MPC Model Predictive Control

LQG Linear Quadradic Gaussian

SMA Shape Memory Alloy

MFC Macro Fiber Composite

ML Machine Learning

RL Reinforcement Learning

MDP Markov Decision Process

ANN Artificial Neural Network

DRL Deep Reinforcement Learning

PID Proportional Integral Derivative

PPO Proximal Policy Optimization

TD Temporal Difference

A2C Advantage Actor-Critic

A2C(λ) Actor-Critic with Eligibility Traces

TRPO Trust Region Policy Optimization

Off-PAC Off Policy Actor-Critic

ER Experience Replay

TTBM Time To Baseline Maximum reward

DDPG Deep Deterministic Policy Gradient

MO	Mitigating Overshoot
PD	Proportional Derivative
LIN	Linear model
LSTM	Long Short-Term Memory
PWM	Pulse Width Modulation
MSD	Multi-Step Dense
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
GLA	Gust Load Alleviation
LIDAR	Light Detection and Ranging
PIV	Particle Image Velocimetry
IQR	Inter-Quartile Range
DL	Deep Learning

LIST OF SYMBOLS

α	Learning rate
α^{ER}	Learning rate for ER updates
α^{OP}	Learning rate for Off-PAC updates
Δt	Period of one timestep in seconds
ϵ	Clipping parameter
γ	Discount factor
$\hat{A}_t$	Advantage
$\hat{V}$	State *value*
Λ	Smoothing factor
$\lambda^{\mathbf{w}}$	Critic trace decay rate
λ^{θ}	Actor trace decay rate
$\mathbb{L}^{CLIP}$	PPO actor objective function
$\mathbb{L}_{critic}$	PPO critic objective function
π_θ	Policy used for action selection
π_θ	Trained policy
σ	Standard deviation
$\mathbf{w}$	Critic weights
$\mathbf{x}$	State features
$\mathbf{z}^{\mathbf{w}}$	Critic eligibility trace vector
$\mathbf{z}^{\theta}$	Actor eligibility trace vector
θ	Actor weights
ε	Epsilon-greedy exploration probability

a	Agent action
CLE	Comparative lift error
G	Acceleration from gravity
GRP	Gust reduction percentage
LE_B	Lift error with baseline
LE_C	Lift error with controller
R	Reward
r	Policy change ratio
s	Agent current state
s'	Agent following state
$SCLE$	Settled lift error
t	Timestep
T_g	Total number of timesteps during a gust
T_k	Acceleration from thrust at timestep k
v_k	velocity at timestep k

ABSTRACT

Adding intelligence to uncrewed aerial vehicle (UAV) design to improve performance in a variety of dynamic environments offers benefits to many civilian and military mission objectives. Urban areas with tall buildings and vast street systems create drastic changes in the flight environment that are challenging for autonomous flight vehicles to overcome. Forrest fires with large temperature gradients also create difficult wind conditions for a drone to accurately survey the area. Autonomously reacting to these environmental changes would improve the adaptability of a single UAV design, allowing performance in a broader range of conditions and effectively increasing the mission scope for these vehicles. Thus far, the fields dedicated to developing adaptive and intelligent systems for aircraft have remained split. One popular area of focus uses multifunctional smart materials to create unique shape changes in an aircraft structure, known as morphing. Another area of research that is rapidly gaining interest is the field of artificial intelligence and machine learning for controller development. In this work, I brought these two fields together to create an intelligent multifunctional morphing system for autonomous adaptive flight.

First, I developed a reinforcement learning (RL) training format for autonomous policy development in a physical hardware environment. The pseudo-episodic training scheme alternates traditional training episodes with exploration episodes designed to randomly reset the following training episode. This format creates space for additional policy updates using off-policy actor-critic and experience replay between traditional training episodes. I tested the pseudo-episodic training format on an airsled-airtrack experimental environment used as a one-dimensional analogy for an aircraft at equilibrium. The inclusion of these additional updates improved learning speed and consistency

Next, I used deep reinforcement learning (DRL) to create two controllers for a macro fiber composite (MFC) actuated camber morphing airfoil. I tested the two DRL controllers on the physical morphing airfoil over a series of step responses when using true state observations from an external sensor, and imperfect state observations from the two state inference models. I compared the performance of the two learned controllers to a traditional proportional-derivative (PD) controller. I found that the learned controllers were faster and more accurate than the PD method, and were able to account for the hysteretic behavior inherent to the piezoelectric MFC actuators when provided imperfect feedback.

Finally, I used DRL to train gust load alleviation (GLA) controllers for an MFC morphing wing that consisted of three morphing sections. The trained controller used pressure sensors for state observation and learned to reduce the change in lift experienced during upward and downward gusts. I found that increasing from one pressure tap to three pressure taps significantly improved overall controller performance, but increasing from three pressure taps to six pressure taps did not show significant improvement. Overall, the controllers were able to reduce the change in lift experienced by the wing during a gust by 71% to 87%.

In summary, the work presented in this dissertation shows a gradual increase in environmental complexity from chapter to chapter in order to develop intelligent controllers for adaptive morphing UAVs. I began with a simple one-dimensional analogy for trim, and ended with an autonomous gust alleviation system trained entirely on hardware using DRL. This work offers an exciting combination of multifuncitonal material morphing with learned autonomous control as a step towards developing truly intelligent morphing UAVs.

CHAPTER I

Introduction

1.1 Motivation and Scope

1.1.1 Intelligent Aircraft

Adaptability is an objective of great interest in the field of aerospace engineering [1]. A single aircraft with the ability to perform under a variety of circumstances is valuable for military, commercial, and civilian application. This focus has pushed the field to look toward implementing intelligent systems in aircraft design to allow uncrewed aerial vehicles (UAV) to autonomously react to changes in their environment. For the purposes of this dissertation, these intelligent systems are categorized into three primary focuses, including: morphing structures, multifunctional materials, and autonomous control (Fig. 1.1). At the intersections between these primary subjects are interesting questions that must be addressed to achieve truly intelligent aircraft design.

Currently one of the most popular intersections for study falls between multifunctional materials and morphing structures [2], [3]. This intersection seeks to answer the question, how can we efficiently perform these structural shape changes through incorporating smart material actuators. These systems show improvement in mechanical complexity, weight reduction, and aerodynamic efficiency over traditional

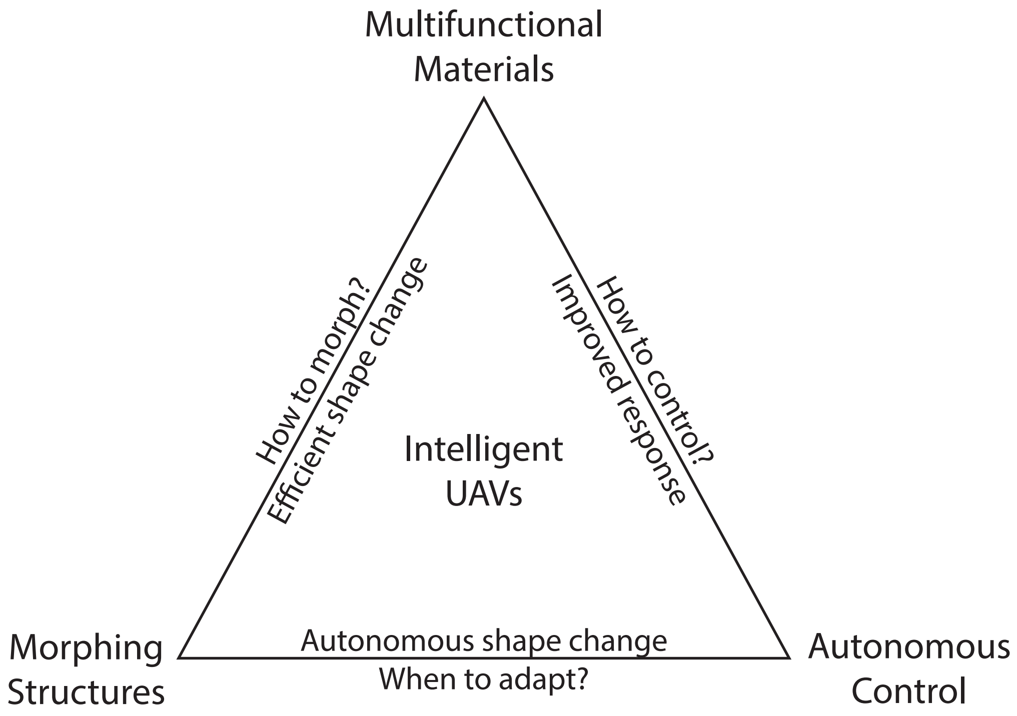

Figure 1.1: Representation of UAV intelligence components.

methods [4], [5].

Another intersection of interest is that between morphing systems and autonomous control. Researchers in this field look to determine when to produce structural shape changes to improve aircraft efficiency or performance [6], [7].

Finally, the least explored of these intersections looks at the relationship between multifunctional materials and autonomous learned control [8]. Although smart materials provide many benefits for aircraft design, they come with challenges for consistent control. For this reason, it is important to find control methods that effectively mitigate these challenges, so the benefits of these multifunctional materials can be more fully used to our advantage.

1.2 Gust Alleviation

1.2.1 Motivation

Maneuverability is a key characteristic for an adaptable aircraft, and in this thesis will be referred to as an aircraft's ability to change translational and rotational velocity in magnitude and direction [9]. This allows the aircraft to respond quickly to environmental changes. However, achieving this comes at the direct cost of stability [10]. Longitudinally speaking, stable aircraft are designed to have the aerodynamic center (AC) behind the center of gravity (CG), so that flight in a perturbed state would naturally push the aircraft back toward a level trim condition. Although this is beneficial for efficient flight at cruise, stable flight reduces an aircraft's ability to deviate rapidly from trim, requiring greater forces to overcome its natural tendency toward level steady flight. Unsteady aircraft require smaller forces and moments to adjust flight trajectory, improving maneuverability, but are unable to passively reject external perturbations. This is amplified as aircraft weight decreases, increasing the impact for small UAVs [11].

Birds overcome this obstacle by shifting between stable and unstable flight [12]. They are able to change their AC without significantly adjusting their CG, allowing for a rapid change between flight conditions. Many highly maneuverable aircraft also can adjust flight stability using wing sweep; however, these shape changes currently occur over a longer period of time [1]. Until aircraft are able to change stability configuration more rapidly, it is beneficial to combine an inherently unstable aircraft design with an active control system to reject perturbations, thus allowing for stability and maneuverability when needed [10].

In addition to added maneuverability, gust alleviation systems reduce structural loads experienced by aircraft during external disturbances [13]. Structural designs are constrained by stresses experienced during peak load situations, such as gusts. Therefore, reducing the loads experienced during gusts can relax design requirements, allowing for lighter and more efficient designs [14].

Combining these ideas, gust load alleviation (GLA) systems allow design engineers to focus more heavily on efficiency than structural and aerodynamic stability, allowing greater use and production of highly efficient aircraft that are typically inherently unstable, such as flying wing designs [15]. Shifting aircraft design focus in this direction would improve aircraft performance in both maneuverability and efficiency, creating aircraft that are more adaptable to changes in the environment and require less fuel. Greater adaptability means fewer variations of specialized aircraft designs, further reducing the carbon footprint associated with the manufacturing process.

1.2.2 Gust Alleviation Methods

With the goal of automatic gust alleviation, passive systems have been developed to reduce loading without the need for extra energy expenditure. Many of these methods include hinge and spring systems that allow a wing tip or flap to give when subject to excessive aerodynamic loads. These methods are useful; however, they are highly sensitive to spring stiffness and therefore are unable to adjust to a variety of flight conditions [16].

Active methods typically use predictive models and traditional control surfaces (ailerons, elevators, and spoilers) to achieve GLA [17], [18]. Model predictive control (MPC) methods use predictive models and optimization for action selection [19]. Dillsaver used a linear quadradic gaussian (LQG) controller to reduce gust effects by 47% [20].

Recently, these active methods have started using non-conventional techniques for sensing and actuation. State observations are crucial for effective control. Traditional aircraft sensors are often used with Kalman filters for state inference [19]. Recently, Light Detection and Ranging (LIDAR) forecasts wind disturbances 50 meters ahead of the aircraft [21], [22]. Pankonien recently incorporated hair-like sensors for gust perturbation sensing [23]. Outside of traditional control surfaces, alternative control methods have been used for more efficient gust alleviation. Li used fluidic actuators to change the airflow around a wing during a gust, and wang used spanwise trailing edge morphing [14], [24]. In this work, we focus on multifunctional smart material actuators for morphing based control.

1.3 Multifunctional Materials for Morphing

1.3.1 Smart Material Actuators

Incorporating intelligence into aircraft design is not a new topic of research. In the 1990s NASA began an Aircraft Morphing Program dedicated to developing "self-adaptive flight" [25], [26]. This program focused on improving aircraft efficiency by incorporating intelligent sensing, actuation, and control systems. Smart materials became a large focus in this program due to their unique behaviors promoting multi-functionality. Acting as both structure and actuator, these smart materials reduced mechanical complexity in morphing systems, limiting the number of components capable of failing. Through this program, NASA identified shape memory alloys (SMA), magnetostrictive materials, and piezoelectric fiber composites for containing these useful qualities [25].

Shape memory alloys experience strains of up to 8% when subject to a thermal load. Capable of relatively large force production, SMAs are limited in actuation frequency due to the time constraints necessary to cycle the material temperature [27]. Still, SMAs are used widely in aircraft morphing (Fig. 1.2a) [4], [28], [29]. Magnetostrictive material operates more rapidly, but produce limited strains of only 0.2%, reducing their functionality for aircraft morphing [5]. Macro fiber composites (MFC) combine piezoelectric rods within an epoxy layup to produce a material with rapid actuation and relatively high strains. For this reason, MFCs are a popular multifunctional actuator for morphing UAV design, including pitch and yaw control effectiveness and yaw stability for a novel avian inspired rudderless UAV with camber morphing MFC tail actuator (Fig. 1.2b) [30]–[33]. MFCs are my actuator of choice for the remainder of this dissertation. For this reason, I next provide a more in depth review of the uses and challenges of MFCs in morphing aircraft design.

1.3.2 MFC Morphing Aircraft

Macro fiber composites are low-profile piezoelectric actuators which have gained substantial attention within the morphing aircraft community [1]. Piezoelectric actuators operate by generating strain when voltage, and hence an electric field, is applied to the electrodes [35]. Piezoelectric actuators are also well known for their capabilities to produce high force output and a high-speed actuation response. Yet unlike traditional piezoelectric actuators which are composed of solid piezoelectric material, MFCs are manufactured using a series of thin piezoceramic rods in a composite laminate layup, allowing them to exhibit excellent flexibility while still maintaining

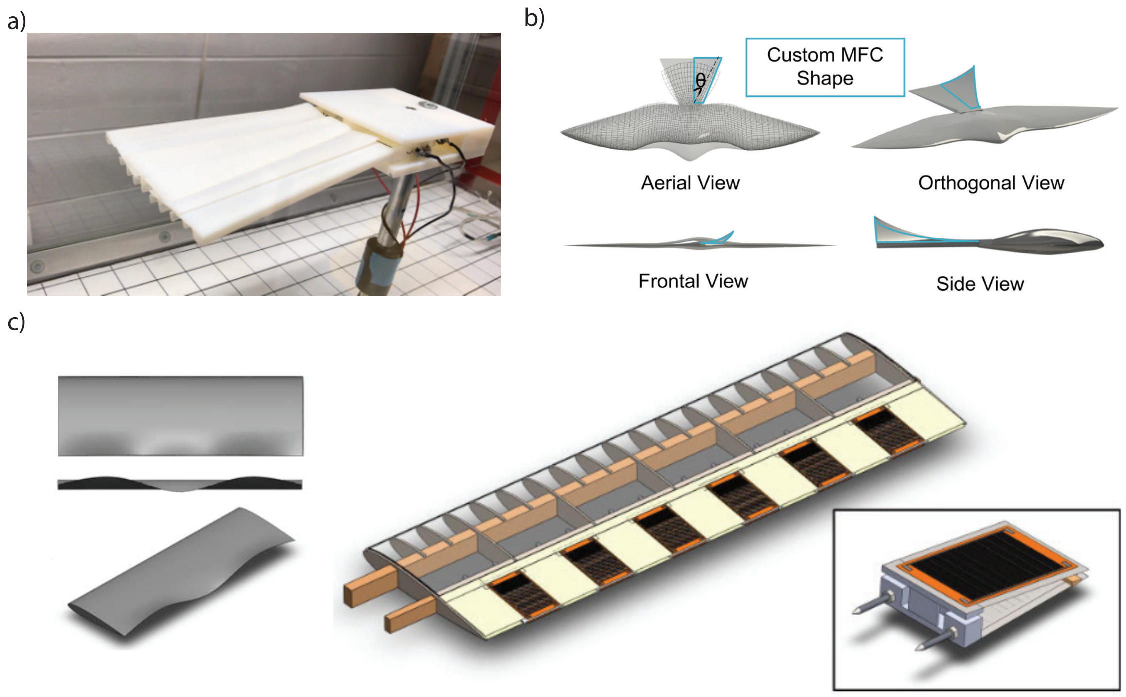

Figure 1.2: Smart materials are used for aerospace morphing designs. Antagonistic SMA springs are used to actuate an avian inspired planform morphing tail a) [28]. Two custom designed MFCs are used for yaw control in an avian inspired UAV b) [30]. Smooth spanwise camber morphing is achieved by implementing several MFCs c) [34].

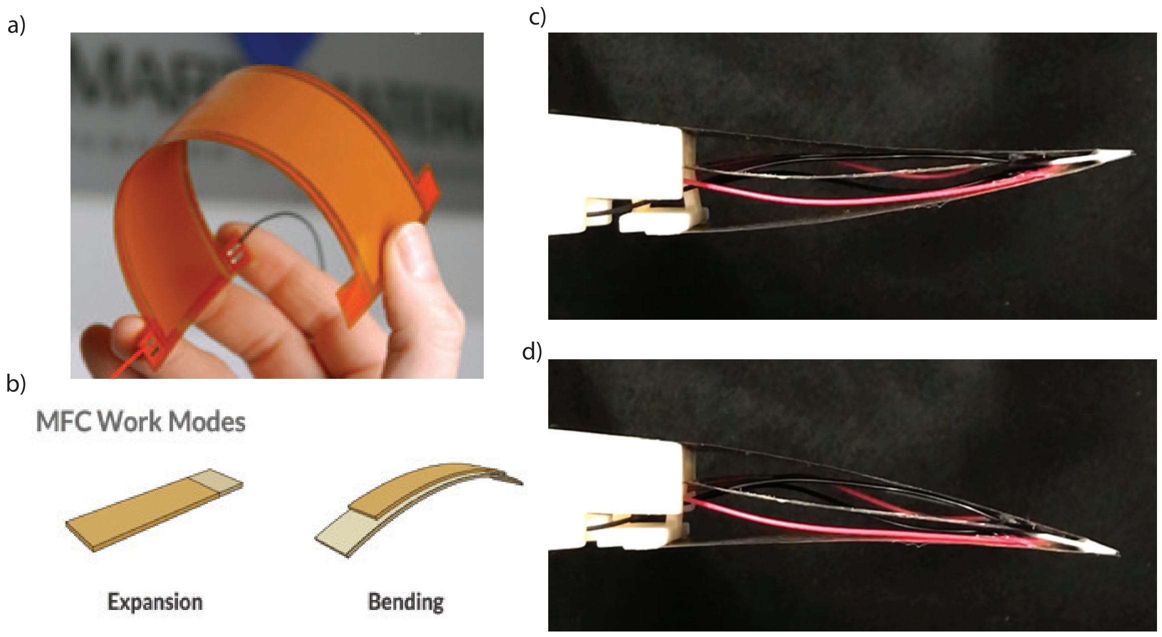

Figure 1.3: MFCs are used for camber morphing. MFCs are very flexible, a), and can be bonded to a thin intextensible substrate to create bending geometries b) [1], [38]. Combining two antagonistic MFC unimorphs, we can develop a system capable of deflecting the trailing edge of an airfoil upwards, (c), and downwards, d).

the performance benefits attributed to traditional piezoelectric actuators, including actuation speeds up to 700kHz [31], [36]. Furthermore, MFCs exhibit large out-of-plane curvatures when bonded to a thin inextensible substrate like steel shim which shifts the structure's neutral axis (Fig. 1.3). This behavior is especially attractive for camber morphing airfoil applications and has spurred a large subset of research in the field of morphing aircraft [1], [37].

Though the field of morphing aircraft is brimming with novel morphing mechanisms, camber morphing wings and airfoils have proven to be especially beneficial due to their ability to increase control authority and improve efficiency [39]–[42]. Macro fiber composite actuators have been widely used in camber morphing wings in part because they are capable of seamlessly generating cambered actuation, allowing them to serve both as the airfoil skin and actuator [1]. Furthermore, the lightweight nature of MFCs, in addition to their rapid actuation response, is advantageous in UAV applications, as reductions in aircraft weight lead to greater fuel efficiency, and rapid actuation allows for greater maneuverability. Initial research into MFC-driven camber morphing airfoils constructed the airfoil's trailing edge using an MFC bimorph as the upper and lower surface, and integrated a flexure box which permitted sliding shear motion during actuation at the junction between the rigid and morphing sections [37]. This camber morphing airfoil was later used in the Spanwise Morphing Trailing Edge concept, which integrated a series of 6 MFC-based camber morphing actuators along the span of a wing (Fig. 1.2c) [34]. With this wing configuration, the spanwise camber of the wing could also be locally optimized to adapt to and recover from adverse aerodynamic disturbances including stall [43]. Camber morphing MFCs have also been used in rudderless UAV applications. When integrated in a spanwise morphing configuration, simulations of a rudderless UAV with camber morphing MFC wings showed large improvements in efficiency in addition to impressive roll and pitch controllability [44].

Although MFCs have demonstrated incredible potential for camber morphing applications, they are not without drawbacks. While piezoelectric actuators generate high force output, the thin and flexible nature of MFCs make them vulnerable to displacement under large out-of-plane forces (Fig. 1.4). As the wings of aircraft are the primary lifting surface, large aerodynamic loads are prone to inducing aeroelastic deformation of MFC-actuated airfoils, reducing the total camber. However, this can be remedied using control algorithms which utilize feedback control to tune the actuator's input voltage such that the desired camber or trailing edge displacement is achieved [5]. Though this has proven successful, the inherent hysteresis of MFCs is a challenging hurdle for traditional control algorithms, many of which are linear by nature [45]. In contrast, deep reinforcement learning is well equipped to handle nonlinear control relationships and may be a promising alternative.

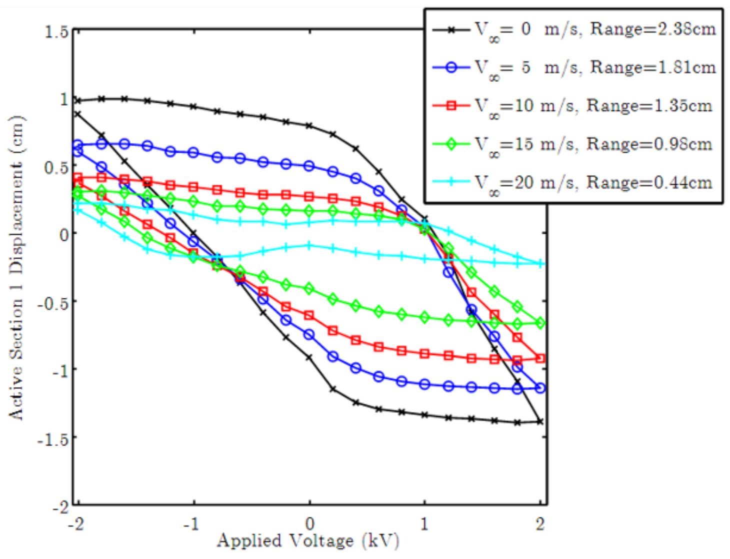

Figure 1.4: MFCs have reduced range of motion under out of plane loading. These hysteresis curves show the change in deflection through the actuation cycles of an MFC camber morphing airfoil and wind tunnel flow speeds ranging from 0 m/s to 20 m/s [5]

1.4 Reinforcement Learning

Reinforcement learning has gained significant attention in the field of artificial intelligence over the past several years. Agents and environments are the two fundamental pieces of any RL problem, where the agent is the subject of policy-determined actions, and the environment is the defined space within which the agent may act (Fig 1.5). The RL problem is traditionally further defined by a Markov decision process (MDP), consisting of every possible state-action combination, and their state outcomes, for the agent within the environment. Depending on the goal of the agent, certain state-action pairs may be preferable over others, as determined by rewards.

Although reward-based learning algorithms are not a new concept, recent advancements in hardware and increased data accessibility have made it possible to achieve accurate estimation of nonlinear systems using large quantities of data [47]–[49]. Artificial neural networks (ANN), deep ANNs specifically, are frequently the preferred method of function approximation in supervised learning tasks and have

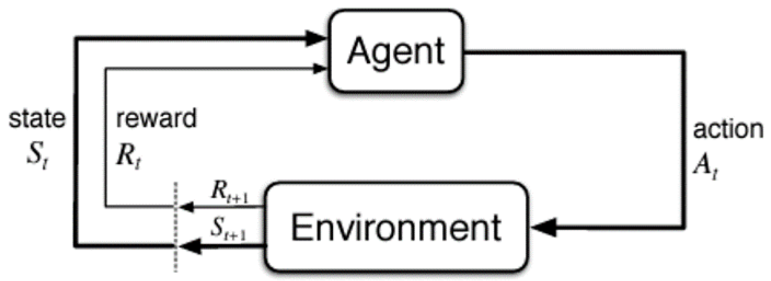

Figure 1.5: Reinforcement learning is a controller development scheme through trial-and-error. Each RL problem contains an agent and an environment. The agent selection actions based on its current state, and the environment provides the agent with a new state and reward to represent the cost or benefit of the previous state-action selection [46]

gained popularity in the reinforcement learning community. There is now a subset of RL dedicated to the use of multilayered ANNs called deep reinforcement learning (DRL). This is not without merit, for they have led to superhuman performance in Atari game environments, Dota 2, and Go, as well as many simulated physics environments offered by the OpenAI libraries and MuJoCo to name a few [48], [50]–[52]. While this is impressive, and certainly beneficial for the RL community, it is necessary to build from what we've learned using simulated environments, where data accumulation is plentiful and instrumental wear and tear plays no role, and apply that knowledge to real-world environments and robotics.

1.4.1 RL in aerial environments

UAV flight and maneuverability provide interesting control problems, particularly when operating in variable environments where adaptability offers a great benefit. We are not the first to aspire toward this objective. The ability to autonomously adapt to aerial environments is an exciting goal that many in the reinforcement learning community are pursuing. The literature has many examples where control is achieved in simulated flight environments. Bohn and his colleagues achieve attitude control competitive with traditional proportional integral derivative (PID) methods using proximal policy optimization (PPO) for a simulated fixed wing UAV [53]. Koch et al. achieved similar results, outperforming PID methods when using PPO to train an open-source simulation for quadrotor flight [54]. In addition to attitude control, navigating a simulated environment is another challenge frequently accomplished with RL methods [55], [56]. Each of these cases shows the power of RL in simulation, but impressive work has been achieved in the real world as well. In 2003, Ng et al. used reinforcement learning to achieve autonomous control for a helicopter [57]. More recently, a trained quadrotor was capable of recovering from complex initial conditions, including an upside-down position [58]. However, in both cases a model or simulation was first developed for training, and no training was performed in real-time on the physical equipment. In 2018, navigation of an unknown environment was achieved in by a Parrot AR quadrotor drone; however, PID was used to aid in control of the UAV and, due to the constraint of battery life, human intervention was allowed for learning to continue after a UAV failure [59].

1.4.2 RL in real-world environments

While there has been substantial success in the simulated DRL environments, the data-hungry nature of ANNs has led to solutions geared towards artificially increasing the amount of data available to these nonlinear approximation functions. Many examples in the literature use transfer learning techniques to train an agent in a simulated environment and then load the pre-trained policy onto the physical agent of a real-world environment [60]–[62]. Other examples use imitation learning, manually guiding the agent through the motions desired for acceptable performance [63], [64]. Both methods are popular ways to limit the time spent training on hardware, and reduce wear and tear on equipment; however, not all environments lend themselves easily to accurate simulation. In many cases, manual examples may not be desirable, or even possible, to produce. For this reason, I emphasize the use of model-free methods in this dissertation. The reinforcement learning methods used in this research fall under a category called temporal difference (TD) learning. TD learning allows for policy development through experiences gathered directly from agent-environment interaction. This presents an opportunity to develop a controller without mathematically approximating the dynamics of the environment. In two chapters I perform training directly on the physical hardware environment (ch. II, IV). In Chapter III I train a model using only data gathered directly from the physical hardware environment to develop a simulation for training.

1.5 Outline and Contributions of Dissertation

This dissertation is designed to gradually build toward autonomously developed controllers for complex morphing wing system to improve efficiency and adaptability in small UAVs. Each chapter represents separate projects that build on information gained from the previous. Adding complexity to the system at each step, I began with a very simple unsteady experimental analogy for trim and ended with a self-taught gust alleviation system for a multifunctional morphing wing.

Chapter II focuses on developing a reinforcement learning format capable of autonomously completing training on the physical hardware environment. As mentioned, the environment is a simple analogy for achieving and maintaining trim after beginning in a perturbed state. Much of the RL community is focused on improving algorithm designs to achieve state-of-the-art performance in simulated environments. Those that venture outside of simulation typically rely heavily on modeling to train quickly in physical hardware. This limits the potential benefit of using RL since one of its greatest attributes is the ability to produce controllers for environments of great complexity. My contribution is a training method that doesn't rely on prior modeling or human intervention and is cognizant of time and safety constraints for learning in hardware by creating space to improve learning speed and consistency. I achieved this by alternating on-policy and off-policy updates during training and exploration episodes. I presented this and related work as a poster at the Multifunctional Materials and Structures Gordon Research Conference in Ventura, California, on January 22, 2020, and at the International Conference on Adaptive Structures and Technologies Digital Workshop, which was held virtually on October 7, 2020. Additionally, a version if this chapter has been published [65].

Chapter III increases environmental complexity by applying DRL to an multifunctional MFC morphing airfoil. Previously, this morphing system used traditional control methods, including PD and PID, with internal flex sensor signals for feedback [5]. I used machine learning techniques to improve state inference over the traditional linear inference method, and PPO to develop two controllers for the morphing system. I found that ML techniques improved state inference and controller performance. Additionally, the learned controllers improved speed and accuracy in control. Finally, I found that including an additional penalty within the reward structure reduced controller overshoot, producing an overall better controller over a traditional PD method, especially when the less accurate linear inference method was used. I presented this and related work at the CIMTEC Forum on New Materials in Perugia, Italy, on June 27, 2022, and at the American Society of Mechanical Engineers (ASME) conference on Smart Materials, Adaptive Structures, and Intelligent Systems (SMASIS) in Dearborn, Michigan, on September 13, 2022 [66]. Additionally, a version of the work presented in this chapter has been published [67].

Chapter IV presents the culminating achievement of my dissertation work. With the primary goal of improving adaptability and efficiency in small UAV flight, I used DRL to develop a gust alleviation system for an MFC morphing wing capable of responding to environmental changes directly from on-board pressure signals. As mentioned previously, much of the work in gust rejection relies on modeling and prediction to detect gust appearance and infer the aerodynamic impact on lift production before determining an appropriate response through traditional control surface actuation. Here, we eliminated this prediction, reducing computational complexity and potential for modelling error, resulting in a rapid *fly-by-feel* reaction as the gust occurs, similar to the autonomic perturbation response in avian flight. I presented the work in this chapter as a poster at the Multifunctional Materials and Structures Gordon Research Conference in Ventura, California, on September 26, 2022.

Chapter V provides a summary of the work presented in this dissertation and presents my conclusions and contributions.

CHAPTER II

Autonomous Training using Pseudo-Episodic Format

2.1 Summary

For practical considerations, reinforcement learning has proven to be a difficult task outside of simulation when applied to a physical experiment. Here I derived an approach to model-free RL, achieved entirely online, through careful experimental design and algorithmic decision making. I designed a reinforcement learning scheme to implement traditionally episodic algorithms for an unstable 1-dimensional mechanical environment. The training scheme was completely autonomous, requiring no human to be present throughout the learning process. I showed that the pseudo-episodic technique allows for additional learning updates with off-policy actor-critic and experience replay methods. I showed that including these additional updates between periods of traditional training episodes improved speed and consistency of learning. Furthermore, I validated the procedure in experimental hardware. In the physical environment, several algorithm variants learned rapidly, each surpassing baseline maximum reward. The algorithms in this research were model free and used only information obtained by an onboard sensor during training.

2.2 Motivation and Background

Reinforcement learning is a subset of ML that, through trial and error, and the use of a reward system, is capable of autonomously developing controllers for agents in a variety of environments. The preferability of a state is defined by its *value*, and is determined by calculating the long-term reward obtained by the agent, from the environment, after residing in the specified state. In many algorithms the state values are used to determine the agent's policy, which is a map deciding which actions are to be taken given the current state. In the simplest cases, values can be recorded in a table; however, when environments become more complicated, requiring a continuous state space, they must be approximated. In recent years, ANNs have been the cutting-edge function approximation method of choice for many in the field of RL, and for good reason. Given the correct structure and enough data, ANNs are incredibly proficient at accurately approximating nonlinear functions. This type of RL is known as DRL. The success of DRL has brought a surge in popularity for the RL community, resulting in new algorithms and techniques to improve learning speed and final overall performance. This is particularly true for traditional baseline environments, including Atari games and MuJoCo physics simulators for higher dimensional problems [68]–[70].

This work focused primarily on policy gradient methods, specifically actor-critic based methods, to achieve learning. Actor-critic methods consist of two function approximators, an actor and a critic. The Actor is a parameterized policy function and uses a soft-max distribution to represent each action as a probability. Actor and critic weight updates, $\Delta\boldsymbol{\theta}$ and $\Delta\mathbf{w}$, are performed with gradient decent, using

$$\Delta\mathbf{w} = \alpha\delta\nabla\hat{V}(s, \mathbf{w}), \tag{2.1}$$

and

$$\Delta\boldsymbol{\theta} = \alpha\delta\nabla\pi_\theta(a|s,\boldsymbol{\theta}), \tag{2.2}$$

where α is the learning rate. $\hat{V}(s,\mathbf{w})$ is the *value* of the current state, s, given the critic weights, $\mathbf{w}$. For linear function approximation, the gradient of $\hat{V}(s,\mathbf{w})$ is equal to the state features, $\mathbf{x}$(s, $\mathbf{w}$), for which I used 3rd order Fourier cosine basis functions [71]. For updating the actor's weights, $\boldsymbol{\theta}$, I used the natural log gradient of the trained policy, π_θ, calculated as

$$\nabla\mathbf{ln}\pi_\theta(a|s,\boldsymbol{\theta}) = \mathbf{x}(s,a) - \sum\nolimits_b \pi_\theta(b|a,\boldsymbol{\theta})\mathbf{x}(s,b), \tag{2.3}$$

where a is the current action. To perform both of these updates I first calculated the temporal difference error, δ, found as

$$\delta = R + \gamma\hat{V}(s',\mathbf{w}) - \hat{V}(s,\mathbf{w}), \tag{2.4}$$

the difference between the current *value* and the combination of the expected *value* of the next state, s', and the newly obtained reward, R. In this work, the discount factor, γ, is equal to one.

The algorithms of choice are Advantage Actor-Critic (A2C), Actor-Critic with Eligibility Traces (A2C(λ)), and Proximal Policy Optimization (PPO) [46], [72]. Eligibility traces are a means to use information gathered from each time step and propagate it through parameters updated in previous time steps, according to the contribution of the previous states. The update equations for the actor and critic including eligibility traces are

$$\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{V}(s,\mathbf{w}), \tag{2.5}$$

$$\mathbf{z}^{\theta} \leftarrow \gamma\lambda^{\theta}\mathbf{z}^{\theta} + \nabla\mathbf{ln}\pi_\theta(A|S,\boldsymbol{\theta}), \tag{2.6}$$

$$\Delta\mathbf{w} = \alpha\delta\mathbf{z}^{\mathbf{w}}, \tag{2.7}$$

and

$$\Delta\boldsymbol{\theta} = \alpha\delta\mathbf{z}^{\theta}. \tag{2.8}$$

When calculating the eligibility trace vectors for the actor and critic, $\mathbf{z}^{\theta}$ and $\mathbf{z}^{\mathbf{w}}$ respectively, trace decay rates, λ^{θ} and $\lambda^{\mathbf{w}}$, are used to adjust an update's impact on previous states. PPO is one of the current leading on-policy actor-critic algorithms, showing impressive performance in several high dimensional MuJoCo physics environments. This performance improvement was achieved through the introduction of clipping. Clipping is a simple implementation for an idea akin to trust region policy optimization (TRPO), meant to limit the size of policy updates and mitigate overshooting [72], [73]. This is done by looking at the ratio, r, between an action's probability under the new and old policies, and limits that ratio to fall within $1 \pm \epsilon$, where ϵ is the clipping parameter. I used the suggested value $\epsilon = 0.2$. This relationship is described by the objective function,

$$\mathbb{L}^{CLIP}(\boldsymbol{\theta}) = \hat{\mathbb{E}}[\mathbf{min}(r(\boldsymbol{\theta})\delta, clip(r(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\delta]. \tag{2.9}$$

2.2.1 RL in the physical world

Learning in real-world environments, such as robotics and vehicles, emphasizes different challenges in the reinforcement learning problem. Dulac-Arnold et al. lists several of these challenges, including safety constraints and learning from limited opportunities for data collection [74]. Due to the innate cost of completing training in the physical environment, such as wear and tear on equipment from extended use and the time necessary to accumulate experience required for learning, some researchers choose to take their learning off-line, using batch updates or simulated environments [62], [75]–[77]. When accurate simulations aren't easily attainable, and learning must be performed online, sample efficiency becomes more crucial. Improving sample efficiency is a common research problem in RL. Many have found that the use of models during training can accelerate learning [78]–[80]. Others have delved into the field of meta RL, where the goal of an algorithm is to learn how to learn more quickly [81].

Reinforcement learning is a powerful tool to solve complex problems; however, these problems often take hundreds of thousands or millions of iterations to solve. Because of this, the push for autonomy in RL is highly preferable. Instead of human intervention, some RL problems in the literature are performed in environments with already well established means of control, and the goal of the RL algorithm is to discover a more optimal policy [82]. This can provide a safety net for the agent for cases where exploration approaches a known portion of the state space prone to damaging equipment and allows for simple episode reset, but limits RL to environments where control has already been achieved. Haarnoja et al. use Soft Actor-Critic to train a four-legged robot to locomote over a variety of walking surfaces [70]. This was achieved model-free and without simulation, but human intervention was necessary for resetting the environment when the robot fell over or wandered too far from the initial state. Zhu et al. pushed towards true autonomy in RL, abandoning the use of resetting mechanisms and relying solely on the robots' own sensors for state observation and reward signaling [83].

In this research I produced an option for model-free reinforcement learning that can be achieved autonomously and entirely online. In this work I considered training time and equipment safety through careful experimental design and algorithm decision making. Due to the limited availability of data and the computationally expensive nature of ANNs, I chose to forgo the use of these nonlinear function approximators, opting instead for linear function approximation. Although ANNs were not used in this work, I leveraged some techniques that found success in DRL. I chose to design the experiment to imitate the episodic style of many simulated reinforcement learning problems while maintaining strict autonomy of the agent. Additionally, I took advantage of the pseudo-episodic format to create additional opportunities for learning, where traditionally there are none, by using time and actions taken during the preparation of subsequent training episodes to perform policy updates.

The rest of this chapter is organized in the following manner. Section 2.3 presents the theory and methods I followed to achieve learning and preliminary simulation work used to determine algorithm viability and hyperparameter values. Section 2.4 describes the experimental setup used to demonstrate and validate the training scheme a real-world environment. In section 2.5 I discuss the results of the experiments. The conclusions and future work are found in section 2.6.

2.3 Theory and Simulation

In this section I cover the theory behind the decisions made and methods used in order to achieve learning in the air-sled environment. I first describe the approach to achieve sufficient exploration while considering the safety of the equipment. Subsequently I present the additional updates I chose to implement during the "exploration episodes" that are at the heart of the pseudo-episodic training method. Although this method is designed to perform in physical environments, I investigated its viability in simulation to reduce unnecessary wear and tear on experimental hardware. The simulated training was not used to accelerate real-world learning, but is described at the end of this section for completeness.

2.3.1 Exploration and Safety

Exploration is essential for achieving optimal control through reinforcement learning; however, it often comes at the expense of compromised safety when operating in the real world. Because of this, methods are often implemented to limit the exploration necessary to achieve sufficient learning, including transfer learning and imitation learning, as mentioned earlier. For the purposes of this research, with the goal of fully autonomous learning, I chose to use only online and model-free temporal difference learning methods; therefore, any exploration of the environment must be achieved by the physical agent itself.

In this work, I achieved exploration by means that are often successful in simulated environments, including randomized initial positions, negative rewards, and epsilon-greedy. The latter two are easily implemented in the physical environment as well as in simulation. The use of negative rewards facilitates exploration when the adjustable weights of the approximation function are initialized to zero, presetting the values for each state-action combination to zero. When rewards are negative, a value of zero is only achieved when the goal is met perfectly, which is impossible to reach for practical purposes. At this point, any action taken in any state would result in a reward less than what is expected, resulting in different future action selections when following the learned policy. This method is known as *optimistic initial values*, and is particularly useful early in learning and for stationary tasks specifically [46]. To achieve this technique, I assigned rewards as the negative squared distance between the current position and the target. Another common technique used to facilitate the exploration of the action space is epsilon-greedy [46]. At the time of action selection there is a set probability, ε, that a random action is selected. For this case,the ε value was 0.1, prompting for random action selection approximately 10% of the time. The other 90% of actions selected were considered "greedy" and followed the agent's trained policy, π_θ, for achieving highest expected reward. Performing RL in simulated environments lends itself easily to an episodic format that comes with built-in exploration. At the conclusion of each individual training episode, the environment may be reset with any initial conditions within the state space, allowing for the potential of a previously unseen, or rarely seen, state to be visited. By randomizing this initial position, the agent may explore the state space in a relatively uniform manner. Unfortunately, selecting an initial state for an agent that has yet learned to control itself can be difficult. Additionally, for the sake of autonomy, setting this initial state via human intervention was not an option. To overcome this, I chose to alternate training episodes with what I called “exploration episodes,” in which the agent selected actions randomly from a modified action space for half the number of steps in a training episode. This allowed each training episode to begin in a pseudo-randomized state; however, I found that, due to a slight bias in the action space, the agent would end the exploration episode predominantly in a portion of the state space below the target position. To enable more complete exploration of the state space, I split the exploration action space into two modified spaces, one containing the six lowest motor outputs and another containing the six highest. By alternating between these two action spaces for each exploration episode I ensured initial states occurring both above and below the target position.

While this method allowed for ample exploration of the state space, choosing random actions for extended series of consecutive timesteps is potentially hazardous to the equipment, running the risk of relatively high-speed collision between the air-sled and the end of the air-track. To account for this, I implemented several safety precautions, one of which was achieved during action selection. If the air-sled came within a specified distance of either end of the air-track, 5 centimeters, and did not have a velocity of at least 0.1 m/s toward the center of the air-track, then the maximum or minimum action was selected appropriately and automatically to accelerate the air-sled back toward the safe exploration zone.

2.3.2 Exploration Updates

Implementing this method of alternating learning episodes with exploration episodes as a means of randomized initial position had benefits for exploring the state space but was accomplished at the cost of time. The period of each exploration episode occurred without updating any weights for the actor or critic, yet the equipment still experienced the wear and tear of additional use and fuel or energy consumption. This time, although not wasted, could be spent more efficiently by including additional updates. The two methods studied in this work were using off-policy updates and applying an experience replay (ER) during the exploration episodes.

Off-Policy Actor-Critic (Off-PAC) is an algorithm introduced by Thomas Degris et al. for updating both the actor and critic from actions made using a decision process other than the trained policy [84]. For this case, these updates were performed using the randomized actions selected during exploration episodes. As with any off-policy algorithm, an importance sampling ratio was included to estimate the expectation of the trained policy when given a sample from the acting policy. Off-PAC traditionally uses an eligibility trace format to perform updates; however, for the purposes of this research, I chose to set the trace decay rates to zero so that the updates were equivalent to that of a one-step TD algorithm. I chose to do this because the goal was not to further complicate traditional learning algorithms with additional parameters in need of tuning, but to illustrate the advantage of exploiting time otherwise ignored during autonomous training in a physical experiment.

Experience replay is another tool commonly used in RL to increase the speed of learning and mitigate catastrophic forgetting in deep learning by keeping a memory of previous quadruples, (S, A, S', R), each containing a state and action with the following state and corresponding reward earned [85]–[87]. More recently, this technique has been applied to actor-critic methods and has become incredibly popular for DRL methods due to its ability to provide series of uncorrelated data for batch updates for a neural network [88], [89]. In this case I built a mini-batch that held up to 5 episodes' worth of the most recent quadruples, from which they were selected randomly in sets of 10 for each exploration episode timestep. The quadruples contained enough information to calculate a gradient and perform an update similar to that which I performed when using Off-PAC, but with an importance sampling ratio of one since actions were selected following the trained policy. This allowed for the agent to continue to learn from past experiences when updates would otherwise not occur. The implementations of these methods are further illustrated in Algorithm 1.

Algorithm 1: Exploration with Off-PAC or Experience Replay

```
Input: Off-PAC ← Boolean
Input: REPLAY ← Boolean
initialization;
$NE$ ← Number of episodes;
$NS$ ← Number of steps per episode;
$NR$ ← Number of updates from replay buffer;
for iteration=0,1,...,NE do
    if iteration is even then
        for step=0,1,...,NS do
            Follow policy $\pi_\theta$;
            Update $\boldsymbol{\theta}$ and $\mathbf{w}$;
            if REPLAY is True then
                Save (S,A,S',R) to replay buffer;
            end
        end
    end
    for step=0,1,...,NS/2 do
        Follow random action policy;
        if Off-PAC is True then
            Update $\boldsymbol{\theta}$ and $\mathbf{w}$ using Off-PAC;
        end
        else if REPLAY is True then
            for replays=1,2,...,NR do
                Sample (S,A,S',R) from replay buffer;
                Update $\boldsymbol{\theta}$;
            end
        end
    end
end
```

2.3.3 Simulations

Due to wear and tear experienced by equipment from regular use, it was preferable to limit time spent tuning and training in the physical environment. Because of this, I decided to perform preliminary tests in a simulated environment, where many repetitions of training could be used for tuning hyperparameters and testing algorithm designs, without the consequence of equipment damage. The environment that I simulated is depicted in Figure 2.1. For this air-sled/air-track environment, the target position for the air-sled was 0.6 meters from the end of the air-track. The state space was continuous and two dimensional, capturing all position values within the range of 0 and 1.2 meters and velocities falling between -1 and 1 meters per second. Set at a 10-degree incline, the sled had an analog DC propeller motor that produced a force output for position control. The incline rendered the open loop system unstable. The physics of the air-track/air-sled environment are captured by these iterative kinematic equations:

$$x_{k+1} = x_k + \frac{T_k - G}{2}\Delta t^2 + v_k \Delta t \tag{2.10}$$

and

$$v_{k+1} = v_k + (T_k - G)\Delta t. \tag{2.11}$$

Here G is the acceleration due to gravity, which in this case is 0.52 m/s2 to account for the air-track incline,T_k is the acceleration from thrust, Δt is the simulation timestep size of 0.05 seconds, x_k is the position at timestep k, and v_k is the velocity at time step k. An air-sled mass of 0.170 kg was used to determine T_k. These equations were used to determine the air-sled's state (position and velocity) over the 400 timesteps of each training and testing episode as well as the 200 timesteps of each exploration episode.

This research aimed to achieve learning in a real-world environment; thus, it

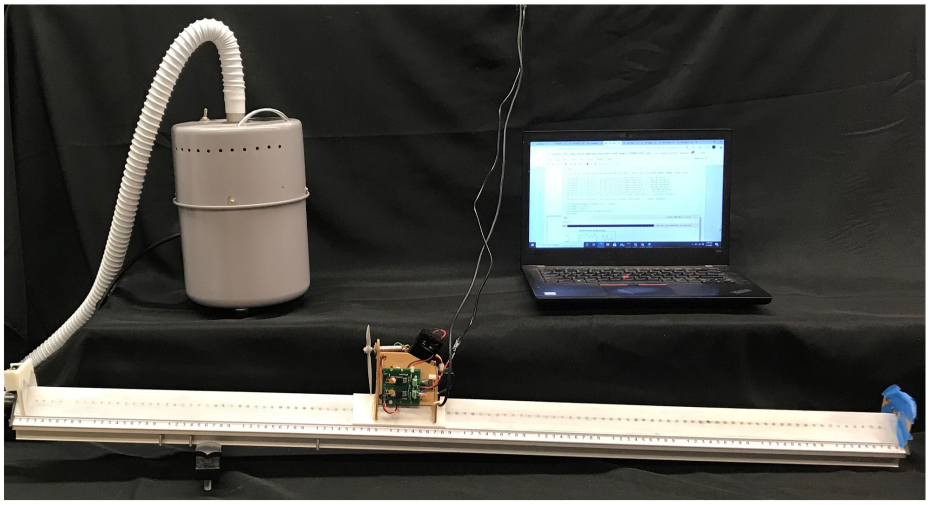

Figure 2.1: Air track and air-sled environment. Because the track was tilted at a 10-degree angle, the system was inherently unstable, requiring constant feedback control to maintain the air-sled's position.

was important to create as accurate a simulation as possible. Incorporating the noise produced by the propeller thrust output was crucial to accurately model the uncertainty experienced by the controller. Modeling the propeller thrust noise was achieved by first measuring samples of the output force produced by the propeller at a series of motor values. I chose to make force measurements for ten different motor values ranging from 25 to 250. These measurements were taken at 1000 Hz for 5 seconds each, resulting in 5000 force output readings for each determined motor value. These samples created a series of non-Gaussian distributions best represented by the triangular distribution function in python's "scipy" statistics library. Due to the 0.05 second timestep, a force output for a given motor value was determined by averaging fifty samples from the respective triangular distribution. Following the central limit theorem, this led to Gaussian distributions of force outputs for each motor value. These average acceleration values, calculated using the air-sled's mass, and their standard deviations are shown in Figure 2.2. As illustrated, 90% of the average acceleration from the propeller was achieved by a motor value of 100, after which the increase in propeller force diminished. Because of this, I chose to limit

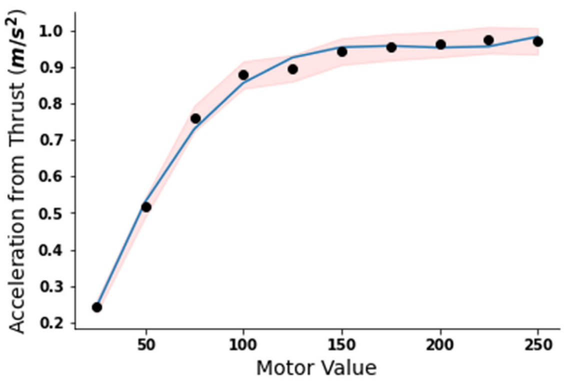

Figure 2.2: Average acceleration due to propeller output for each measured motor value. A fitted line represents the sampling mean during simulation. The shaded area is 1 standard deviation from the averaged measurement values.

the discrete action space to 11 evenly spaced motor values ranging from 0 to 100. The average motor values not covered by the original data set were estimated with the fitted curve also depicted in Figure 2.2. The average standard deviation of force output samples for motor values up to 100 was 0.005. I used this to determine the distributions from which the acceleration values were sampled in simulation.

To mimic the real-world environment as closely as possible, I used the unique exploration methods described in Section 2.3.2. This required forgoing the traditional random initialization of each episode to instead incorporate alternating exploration episodes, simulating the pseudo-episodic format of the physical experiment. Additionally, during exploration episodes, I implemented the safety zones meant for aided collision prevention. Throughout each training iteration, the total reward earned from every other training episode was stored for later comparison between algorithms. This was done to account for the alternation of action spaces between exploration episodes so that recorded rewards came from episodes with similar initial positions given the pseudo-episodic format. Although training occurred during these episodes in the same way as experienced in the other training episodes, I denoted these episodes as "testing episodes" for clarification. It is important to keep in mind that although

I performed a total of 101 episodes for each rendition of training, the graphs only consider the testing episodes, starting with the first, to set a baseline performance without training, and then occurring every fourth episode subsequently.

Figure 2.3 presents the learning curves for each base algorithm and their respective variants in the simulated environment. Because consistency is crucial when applying RL to real-world applications, several iterations of training occur. For the simulated case, I chose to average 10 iterations for each base algorithm and its variants, each iteration with random seeds ranging from 0 to 9. Since the final goal wasn't to achieve learning in simulation, but instead on the physical system, I only briefly describe the simulation results. Off-PAC appeared to accelerate learning for each of the baseline algorithms; however, the average reward earned converged on values slightly less than that achieved by the baseline algorithms. The inclusion of ER during exploration showed the fastest learning of all algorithm variations for A2C and PPO, and, in all cases, earned the highest average reward. These results were encouraging for implementing the learning algorithm completely online in the experimental hardware.

As with any implementation of RL, hyperparameter tuning was a necessity. This intermediate step allowed us to pinpoint hyperparameter values for each algorithm that found success in this reinforcement learning problem. Several of these hyperparameters remained unchanged when learning was moved to the experimental environment. Only the learning rates for each algorithm, α, α^{OP} and α^{ER} were subject to change, but the relationships between the exploration update learning rates, α^{OP} and α^{ER}, and the common learning rate, α, was maintained. These relationships and all other hyperparameters are given in Table 2.1.

Determining these hyperparameters was one of two main purposes of this simulation work. The second of which was as a justification to the merit of applying the new training format, including policy updates during exploration episodes, to this

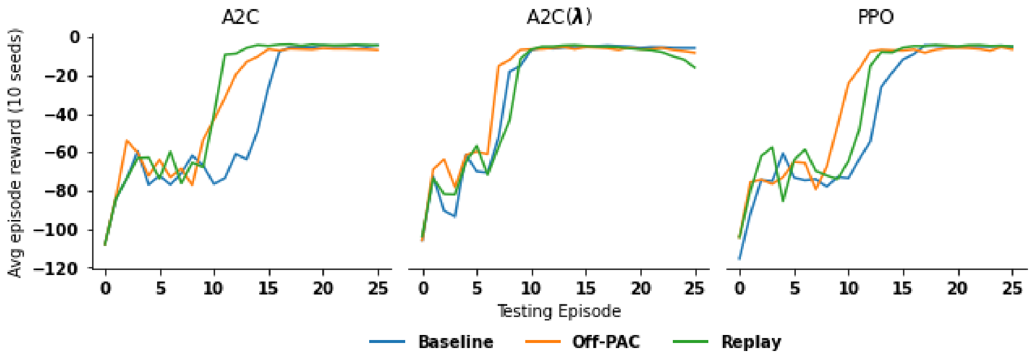

Figure 2.3: Average reward earned per testing episode in simulated training environment. I performed training for each baseline algorithm (A2C, A2C(λ), and PPO) and exploration episode update variants (Off-PAC and Experience Replay). It is important to note that a combined total of 101 episodes of traditional training and exploration were completed for each round of training. Testing episodes occurred every other training episode (every fourth episode when including exploration) to improve initial position consistency for testing and reward comparison.

RL problem before real-world implementation. So, although the experiment did not use any offline training, I did use simulation to determine a reasonable approach for hardware autonomy.

2.4 Experimental Demonstration and Validation

The objective of this research was to present an autonomous training scheme for developing learned controllers in a physical environment that would typically require human intervention or an additional controller for environment reset. In this section I introduce the experimental setup chosen to demonstrate such an environment, including the additional safety measures put in place to mitigate hardware damage. Additionally, I describe an adjustment made to the action selection strategy used in the traditional learning algorithm, and my reasoning for making this adjustment for hardware implementation.

2.4.1 Experimental Environment

Stability is the cost of maneuverability for aerobatic aircraft, requiring autonomous feedback control to maintain trim without explicit piloting. Using sensor input to accomplish equilibrium in an unstable environment provides an appropriate analog for such maneuverable flight vehicles. Therefore, to demonstrate the capability of the pseudo-episodic training scheme with exploration updates I chose to develop a controller for an air-sled such that it maintains a desired position on an inclined one-dimensional air-track, as seen in Figure 2.1. A pump is used to force air through a series of holes set along the length of the track to allow for a near frictionless surface for the sled to slide along, controlled by a single propeller, simulating one dimensional trimmed flight. The propeller output was connected through a motor control board to an Arduino Nano microcontroller. In addition to providing a motor value signal between 0 and 100 to the motor control board, the Arduino Nano received position information from an infrared distance sensor and communicated with external devices through USB. Distance gauged by the infrared sensor was used to calculate the velocity of the air-sled after each 0.05 second timestep. The USB connection allowed information to be relayed between the air-sled and a laptop. The laptop ran a custom Python-based RL script that used state data (position and velocity) as inputs and output the selected action for each timestep. Distance values were continually measured and stored as averaged sets of two in the Arduino Nano's serial buffer. At each timestep, when the learning algorithm asked for the air-sled position, the two most recent averaged values were read into the Python script to be averaged again and the serial buffer was emptied. This was done to limit the amount of data stored in the serial buffer and to smooth the noisy distance data obtained from the infrared sensor.

After moving to the hardware environment, two physical safety measures were put in place, in addition to adjusted exploration action selection, to mitigate the severity of any impact that may occur. The first physical safety measure occurred naturally through the placement of the power supply wiring and communication wires between the air-sled's Arduino Nano and the computer. By using the correct length and placement of this wiring, mounted above and centered over the air-track, the air-sled moved freely about most of the state space; however, when the air-sled approached either end of the air-track, an additional force was applied to air-sled toward the center of the air-track due to the weight of the wiring. This reduced the velocity of the air-sled and often prevented a collision all together. This safety measure did come at a cost. Any movement experienced by the wire had momentum and placed additional external force on the air-sled. This created a greater variance in the accelerations experienced by the air-sled to be overcome during training. In case this safety measure failed to prevent the air-sled from bumping into either end of the air-track, I included padding at either end of the air-track to mitigate any damage that may have occurred.

2.4.2 Deterministic Policy Gradient

Each of the algorithms chosen for this experiment were actor-critic methods which are typically stochastic in their decision-making processes. While stochasticity has many benefits for certain environments and allows for exploration to be more implicit in an algorithm's action selection, a stochastic controller continues to take suboptimal actions after training. Although this undesirable decision-making is infrequent, it can lead to less stability in the final control of the air-sled, particularly when training time is limited. One option to get around this issue is to use a stochastic policy during training and then only select the action of highest probability during testing; however, this too failed to guarantee sufficient control after a short period of training in the experiment. As mentioned, one point of concern for training in a hardware-based environment is the amount of time necessary to achieve sufficient control of the air-sled. While time is always a constraint in RL, it becomes more influential in the physical setting due to wear and tear on equipment, potential accidents, and fuel consumption.

I found that by switching to a deterministic action selection during training, not only was the air-sled capable of smoother control, but it learned to do so in a much shorter training period, completing training in under 30 minutes. Figure 2.4 highlights this improved learning speed. To account for the loss of intrinsic exploration, I chose to implement an epsilon greedy action selection, as described in Section 2.3.2. Although deterministic policy gradient methods have been used before, as in [90], I took an approach similar to a simple off-policy update, using an importance sampling ratio between the stochastic probability of an action's occurrence, defined by trained policy, and the probability defined by the action selection policy. When following the deterministic action selection policy this probability was equal to one; however, when the action was selected following the random policy, occurring 10% of the time due to epsilon-greedy, the action selection probability was the inverse number of available actions. This led to the full update equation,

$$\Delta\boldsymbol{\theta} = \alpha\delta\nabla\mathbf{ln}\pi_\theta(A|S,\boldsymbol{\theta}) * \frac{\pi_\theta(A|S,\boldsymbol{\theta})}{\pi_b(A|S)}. \tag{2.12}$$

I applied the autonomous learning concepts to hardware, taking that step toward self-adaptive flight. After tuning the common learning rate, α, I found that a value of 10e-4 allowed for learning to be fast enough to complete training in 101 episodes (25 minutes) for all but one case, but slow enough to be stable and allow for recognizable differences between algorithm variants. All other hyperparameter values are available in Table 2.1. Although the values of the exploration update learning rates were adjusted for real-world training, their relation to the common learning rate was maintained as described in Table 2.1. All other hyperparameters were consistent with

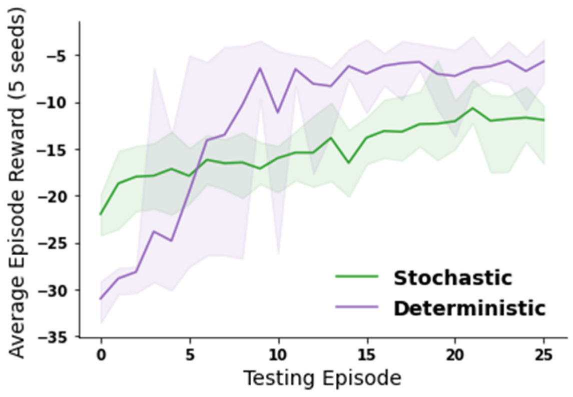

Figure 2.4: Training comparison between stochastic and deterministic PPO. Deterministic PPO earned higher average reward per testing episode in the second half of training on the physical experiment. The shaded area includes all reward values earned by each random seed in respective testing episodes.

those used in simulation.

2.5 Results

Episode reward is a common metric used to gauge an algorithm's ability to learn. Additionally, due to noise and randomness that occurs naturally in physical environments, consistency becomes increasingly important for measuring the viability of an algorithm. To account for consistency, each algorithm of consideration, and its two additional variants, were repeated 5 times. The rewards from the 5 renditions, each with a different random seed ranging from 0 to 4, were averaged to represent the reward for that algorithm or variant thereof. These values, along with the standard deviation of reward earned per episode, are shown for each algorithm and the additional variants in Figure 2.5. While standard deviation may not be specifically meaningful for these small distributions, it can still be a useful means to quantify spread, and therefore consistency, of the algorithm's operation. In addition to consistency, another metric I considered for an algorithm's performance is speed of learning. This is crucial for learning in a real-world environment. In addition to the increased

Table 2.1: Algorithm Hyperparameters

Hyperparameter	A2C	A2C(λ)	PPO
α	0.01	0.01	0.01
α^{OP}	3α	3α	3α
α^{ER}	0.5α	0.2α	0.1α
α^{zw}	0.01	0.01	0.01
λ	-	0.4	-
ϵ	-	-	0.2

[a] The variant learning rates, α^{OP} and α^{ER}, are listed as relationships to common learning rate, α

[b] α is adjusted to $10e-4$ when training on experimental equipment

wear and tear placed on the equipment during lengthy training sessions, this extends the period in which poor actions may be taken, putting the agent in potentially dangerous positions. A black dashed line is included to represent the highest reward earned for each of the base algorithms, allowing us to quantify the speed of learning as the number of episodes required for each algorithm variant to surpass the base algorithm's maximum reward.

Figure 2.5 shows that with the basic A2C algorithm, the separate additions of Off-PAC and ER during the exploration episodes dramatically accelerated learning in the first several episodes, approaching and surpassing the maximum after 11 and 7 testing episodes for the addition of Off-PAC and ER respectively. The baseline A2C did not achieve this until the 22nd testing episode. The standard deviation plots show that incorporating Off-PAC and ER during exploration episodes both have improved the consistency of the algorithm's learning. One thing to note when looking at the standard deviation plots is that, across all three base algorithms, the standard deviation is always highest when the algorithm is doing most of its learning and the reward is still increasing. Similar performance between iterations was achieved at the episode in which the standard deviation dropped below and maintained a value

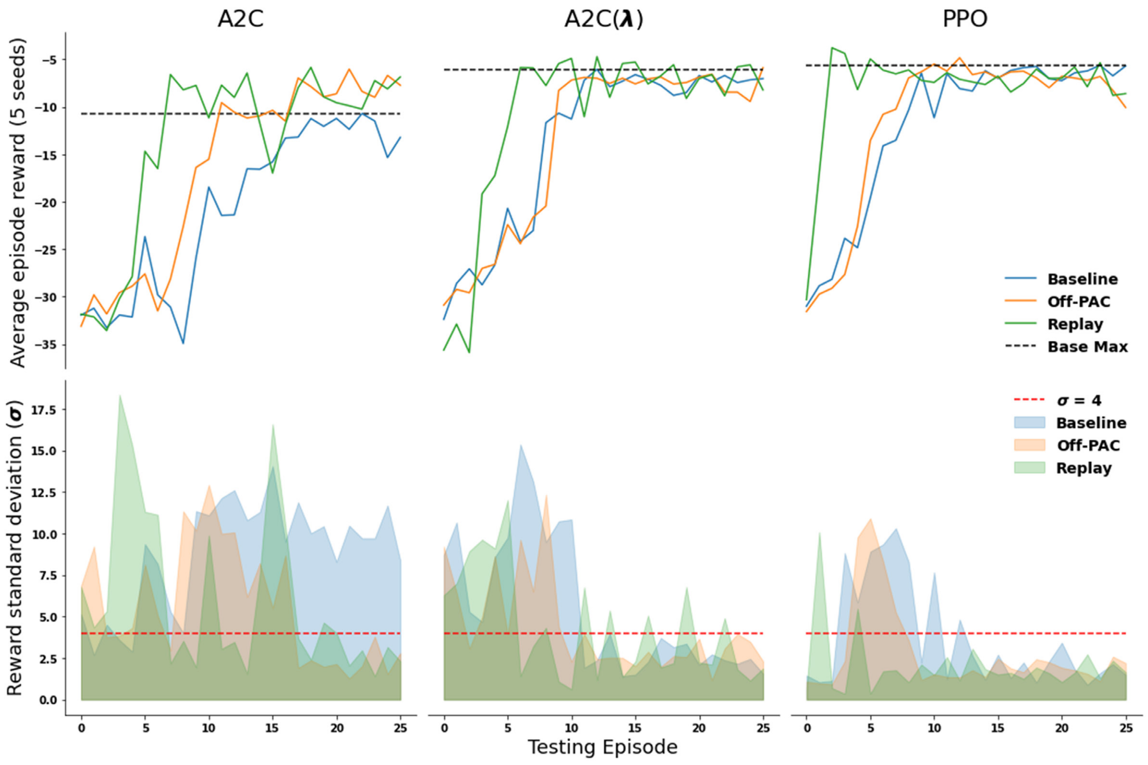

Figure 2.5: Performance comparison between baseline algorithms (A2C, A2C(λ), and PPO) and their exploration episode update variants (Off-PAC and Experience Replay) in real-world training. The top 3 plots show the average reward earned per testing episode. The dashed black line represents the maximum average reward achieved by the respective baseline algorithm. Consistency in learning performance is gauged by the bottom three plots illustrating the standard deviation of rewards earned per training episode for each algorithm and its variants. I consider a standard deviation held below 4 as an indication of consistent learning performance.

beneath a defined low point. This addressed to the consistency of an algorithm's learning. After 9 test episodes, A2C had relatively high standard deviation. This means that although one of the algorithm's iterations may have approached its highest reward, and learned a suitable policy, several iterations had yet to do so. This is true even after training has concluded; however, this changed with the addition of Off-PAC or ER. Adding Off-PAC to the algorithm allowed for consistency in learning after 17 test episodes. For the case of ER, consistency was almost achieved as early as test 7, but there were two high peaks in standard deviation that occur around tests 10 and 15, and then again a smaller spike at 19. Because of this, I defined the metric of "achieving consistency" ($\sigma < 4$) as the first test episode in which the standard deviation dropped below 4, represented by the red dotted line in the Fig. 2.5, and then remained below 4 until training was completed. Additionally, I chose to record the total number of test episodes with standard deviations below 4. With this metric, A2C with ER achieved consistency after 21 episodes and had a total number of 13 test episodes where learning was consistent. These comparisons, in addition to the number of test episodes needed to reach the baseline algorithm's maximum reward (TTBM) as well as the overall maximum reward of each variant, were made for each of the three base algorithms and presented in Table 2.2.

Through these metrics, several patterns developed. In each case, the addition of Off-PAC or ER improved the overall learning achieved by the baseline algorithm when comparing highest average rewards achieved during training, with the ER variant consistently earning the highest average rewards overall. With regards to learning speed, measured by TTBM, the implementation of ER surpassed both the baseline and Off-PAC variant for each of the three algorithms; however, the addition of ER did not frequently aid in learning consistency. This is well illustrated in both the A2C and A2C(λ) cases. Although implementing ER would typically achieve standard deviation values below 4 early in training, as the training continued the standard deviation

Table 2.2: Hardware Learning Performance Metrics

Algorithm	Max avg. reward	TTBM	$\sigma < 4$(total)
A2C	-10.7	22	(4)
A2C Off-PAC	-6.02	11	17 (12)
A2C ER	-5.85	7	21 (13)
A2C(λ)	-6.05	12	11 (15)
A2C(λ) Off-PAC	-5.88	25	10 (18)
A2C(λ) ER	-4.71	6	23 (14)
PPO	-5.62	23	13 (18)
PPO Off-PAC	-4.83	10	8 (22)
PPO ER	-3.78	2	5 (24)

[a] Metrics include the maximum average reward achieved, the number of test episodes needed to reach the baseline maximum rewared (TTBM), and the earliest test episode in which the reward standard deviation dropped below and maintained a value less than 4 through the remainder of training ($\sigma < 4$)

[b] The total number of test episodes achieving a standard deviation less than 4 during training is also listed under$\sigma < 4$ in parenthesis.

frequently spiked, briefly increasing to a value above 4. These spikes often only lasted for one episode, but at times the spike required 3 testing episodes before reducing back to a value representative of consistent learning. On the other hand, introducing Off-PAC updates during the exploration phase improved consistency in every case. Off-PAC was not the fastest algorithm variant when comparing TTBM. When added to A2C(λ), Off-PAC was slower than the baseline. However, adding Off-PAC updates during exploration achieved learning consistency after the fewest number of testing episodes for 2 of the 3 algorithms. Interestingly, this was not the case for PPO, where ER proved to give an advantage in speed and also showed capability in consistent learning. This could be due to PPO's clipping mechanism that is designed specifically to prevent updates from becoming too large and overshooting, potentially improving learning consistency. Although PPO's baseline algorithm does not appear to be as fast at achieving consistency as A2C(λ) according to $\sigma < 4$, this could be due to PPO's slower learning speed, not reaching its maximum average reward until episode

23. Therefore, when another algorithm was used to accelerate learning, such as ER, the result was a fast learning algorithm where the overall standard deviation of reward remained low.

In addition to reward earned throughout training, it was important that the policy learned by the algorithms could control the air-sled to a satisfactory degree after training was completed. Figure 2.6 presents the position values over a period of 20 seconds for the random seed of best performance for each of the base algorithms and their variants. In each plot I see the behavior for two initial positions, one near each end of the air-track. Due to the noise of the infrared sensor, I applied an averaging technique for smoothing the data to improve recognition of the air-track's position for each timestep. The black dotted lines represent locations of 10% error from the target position of 0.6 meters from the end of the air-track. In many cases of control, metrics such as rise time and settling time are used to gauge performance. However, the controllers were trained based on another metric, earned reward, which I used again to compare overall performance between learned policies for the two initial conditions. When considering traditional metrics, A2C(λ) with Off-PAC achieves the best control from an initial position of 0.2 meters and the baseline A2C algorithm trained the best policy for control from an initial position of 1 meter, needing only 4.2 and 6.25 seconds to settle between the 10% error margins respectively. However, when I compare the rewards earned over the duration of the control test, the overall best performance was achieved by the PPO algorithm with Off-PAC exploration updates, earning a combined reward of -11.35. The next best performances in order came from A2C(λ) with ER, A2C(λ) with Off-PAC, and A2C with ER, each earning a combined reward greater than -13. The top 4 performances came from algorithms with the addition of Off-PAC or ER. This suggests that the inclusion of some form of off-policy update during the exploration episodes can benefit training for control with a variety of actor-critic algorithms, assuming performance is measured by the

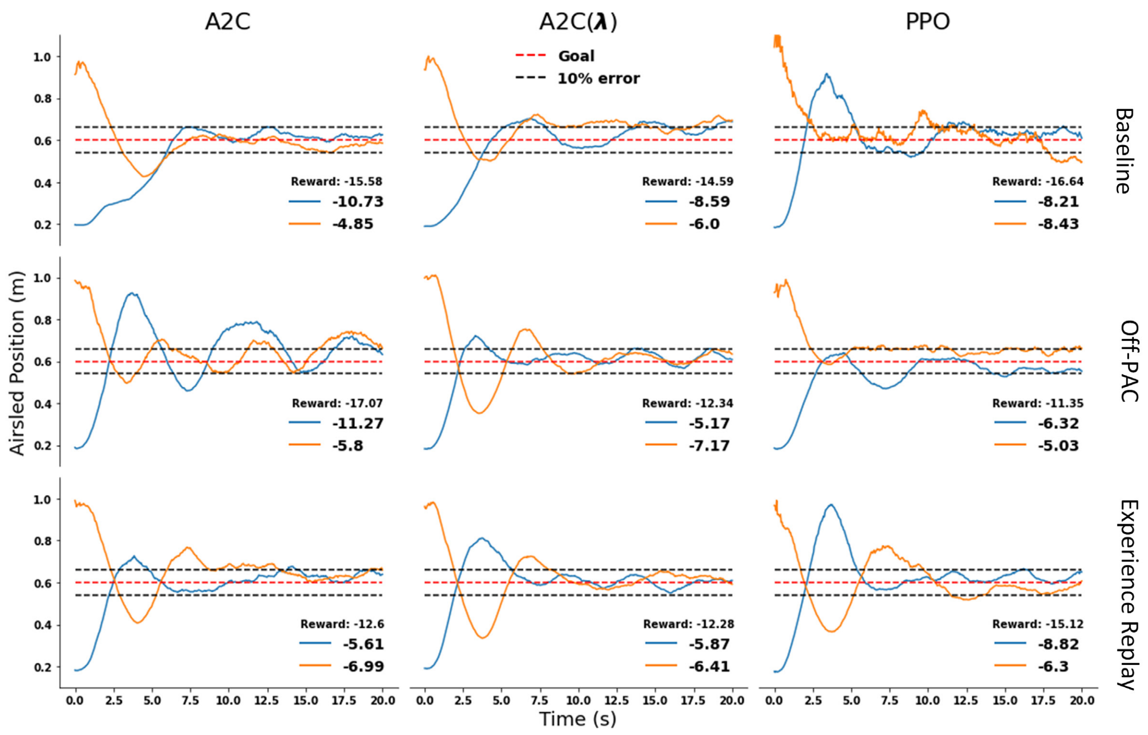

Figure 2.6: Learned controller response performances. I compared position over time of the air-sled on the air track for the best performing policy from each baseline algorithm (A2C, A2C(λ), and PPO) and their exploration episode update variants (Off-PAC and Experience Replay). Two initial positions were considered, each located ± 0.4 meters (± 1.31 ft) from the target. The target position of 0.6 meters (1.97 ft) is represented as a red dashed line and positions of 10% error from the target are black. Reward earned for each time series and the combined total reward are given in the bottom right corner of each plot.

reward system used to direct learning.

2.6 Conclusion and Future Work

The implementation of reinforcement learning in physical experiments has proven to be a difficult task. In this work I developed a pseudo-episodic approach for the autonomous training of an RL agent in a one-dimensional, unstable environment. Our method is model-free and used only information gathered by an on-board sensor. Although training was performed entirely online, its structure allowed for additional policy updates to occur between training episodes. Additionally, I validated this autonomous training method in experimental hardware. The addition of ER and Off-PAC updates during the exploration episodes showed training benefits such as improved speed and consistency in learning respectively. When paired with PPO, ER performed particularly well, overcoming its weakness in maintaining learning consistency. Further improvement was displayed in controller performance when using reward as the compared metric.

This work focused on using novel techniques to improve speed and consistency in learning, while maintaining safety and autonomy in a real-world environment; however, the resulting controllers lacked accuracy. Learning quickly and safely may be prioritized over accuracy in many environments, such as a UAV with precious cargo adapting to a new environment where it is more important to quickly learn safe flight than to achieve optimum performance. With that being said, there are several environments where final performance is the priority; therefore, future work should be dedicated to achieving optimal control, at the sacrifice of speed if necessary. Another potential avenue for this work is to implement neuromorphic chips to allow fast and continuous hardware based learning in this unstable system, taking another step toward in flight adaptation [91].

CHAPTER III

Intelligence for MFC airfoil

3.1 Summary

Smooth camber morphing aircraft offer increased control authority and improved aerodynamic efficiency. Smart material actuators have become a popular driving force for shape changes, capable of adhering to weight and size constraints and allowing for simplicity in mechanical design. As a step towards creating UAVs capable of autonomously responding to flow conditions, this work examines a multifunctional morphing airfoil's ability to follow commands in various flows. I integrated an airfoil with a morphing trailing edge consisting of an antagonistic pair of MFCs, serving as both skin and actuator, and internal piezoelectric flex sensors to form a closed loop composite system. Closed loop feedback control is necessary to accurately follow deflection commands due to the hysteretic behavior of MFCs. Here I used a deep reinforcement learning algorithm, PPO, to control the morphing airfoil. Two neural controllers were trained in a simulation developed through time series modeling on long short-term memory recurrent neural networks. The learned controllers were then tested on the composite wing using two state inference methods in still air and in a wind tunnel at various flow speeds. I compared the performance of the neural controllers to one using traditional position-derivative feedback control methods. The experimental results validate that the autonomous neural controllers were faster and more accurate than traditional methods. This research shows that deep learning methods can overcome common obstacles for achieving sufficient modeling and control when implementing smart composite actuators in an autonomous aerospace environment.

3.2 Motivation and Background

Uncrewed aerial vehicles are growing in popularity for both civilian and military applications, which makes improving their efficiency and adaptability for various aerial environments an attractive objective [1]. Many studies pursue this goal using morphing techniques that incorporate shape changes not typically seen in traditional aircraft [2], [3]. Due to weight and volume constraints consistent with smaller flight vehicles, smart materials, such as MFCs, have been used to achieve the desired shape changes [4], [5]. Macro fiber composites are flexible and lightweight, and when bonded to a thin inextensible skin, like steel shim, can create smooth out-of-plane curvatures. These qualities allow MFCs to behave as both the skin and actuator for a camber morphing airfoil, which offers benefits in efficiency and control authority [1], [37], [39]–[42].

Although MFCs provide aerodynamic and structural benefits in camber morphing applications, they bring some challenges as well. The thin and flexible nature of MFCs that allows the out of plane morphing behavior also makes them susceptible to displacement when subject to large aerodynamic loads. This reduces the total camber achieved by the multifunctional actuators. Using feedback control can mitigate this problem, but the hysteresis and creep associated with MFCs provide an additional challenge for traditional linear controllers. [5], [45]. Deep reinforcement learning has proven to be proficient at performing accurately in nonlinear control environments and may be a promising alternative.

As mentioned previously, RL is a means of autonomously achieving control through trial-and-error. Like in biological systems, a reward system is used to meet a specified goal. Each reinforcement learning problem consists of two fundamental parts, the agent, or object of concern whose actions are determined by a learned policy, and the environment in which the agent observes its state and performs actions. A state's value is measured as the long term expected reward to be received after residing in that specific state [46]. If the state space is large and best represented as continuous, function approximation is used to reduce memory requirements. ANNs are an effective method for function approximation because of their ability to accurately represent nonlinear functions when trained on large quantities of data. Recent work has combined multi-layered ANNs with RL to create the subfield DRL, which has found great success in many simulated and game based environments [48], [50]–[52], [92]. Success in perfectly controlled environments, such as games and simulations, continues to advance the field; however, there is a growing need to apply the knowledge gained through simulation to robotics and other physical hardware environments [60], [61], [65], [74].

Aerial vehicles have been the environment of choice for many reinforcement learning problems with the goal of creating autonomous UAVs that can adapt to their environment or a changing mission [54], [58]. RL supports the complexity of a morphing aircraft by producing a controller that learns to use morphing control surfaces and operates in several configurations as well as by determining the best configuration for a given flight situation [93]–[95]. Although these shape changes are achieved using traditional methods, such as servos and motors, some have implemented RL in smart material based morphing simulations [7], [96]. One case presented by Goecks et al. implemented deep deterministic policy gradient (DDPG) with an SMA actuated airfoil [8]. They found learning in the physical hardware environment to be difficult due to limited time constraints; however, through deep learning, they were able to accurately model the behavior of the SMA actuated airfoil in a wind tunnel and achieved control in a simulated morphing environment. Much of the work performed in DRL, and almost all of the current literature around RL in morphing UAV environments, is performed entirely in simulation.

Through a sim-to-real policy transfer I present the first successful application of RL for an MFC actuated morphing system in a physical hardware environment (Fig. 3.1). In this research, I found that DRL countered the hysteretic behaviors present in the MFC morphing airfoil to effectively control trailing edge tip deflection in the physical hardware. In the simulated environment, I trained two controllers for use on the physical morphing airfoil, including one with a simple position error based reward system (RL) and another with an adjusted reward system designed to mitigate overshoot (MO). Control effectiveness of these two controllers and a traditional proportional-derivative (PD) controller was compared on physical hardware in situations where “true” state information was supplied through an external Keyence profilometer, as well as for on-board sensing where state information was estimated through a piezoelectric flex sensor signal. I used two methods for state inference with the flex sensor signal including a linear model (LIN) and a long short-term memory (LSTM) neural network model. I first made comparisons in an unloaded environment, where system dynamics data was gathered initially to develop the simulation for training, and then additionally when the airfoil was subjected to aerodynamic loads at three different flow speeds.

I found that the learned controllers, specifically the MO controller, outperformed the traditional PD feedback method. This was especially true for control metrics considering speed and accuracy. From this, I verified that autonomously developed controllers are not only viable for MFC actuated camber morphing, but may be a superior option for erratic environments in which rapid adjustments must be made, as in the case of turbulent flow.

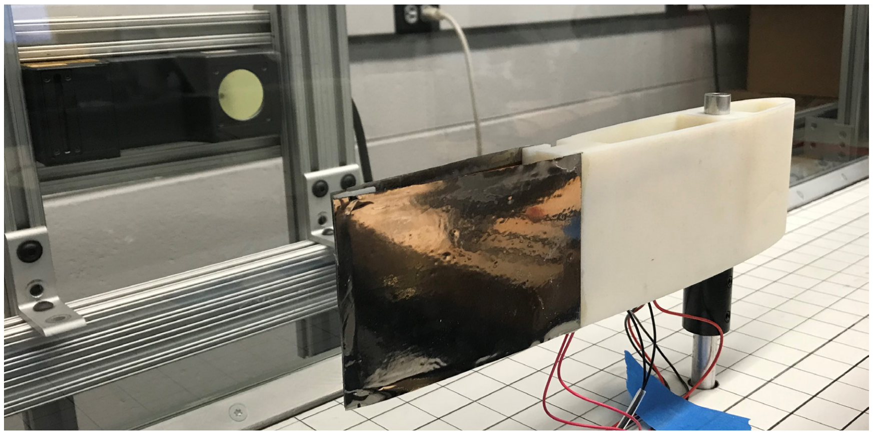

Figure 3.1: Image of the morphing airfoil system within the 1'x1' wind tunnel. Also visible in the upper left corner is the Keyence 2D profilometer used to measure the true deflection of the trailing edge.

3.3 Methods

3.3.1 Morphing Airfoil System Design

I assessed the performances of the learned controllers using the camber morphing airfoil design developed by Pankonien et al. [34].The baseline geometry for the morphing airfoil was a NACA0012 airfoil with a 310 mm chord (Fig. 3.2). I 3D printed the leading edge using an Objet Connex500 multi-material 3D printer, which can print both rigid and flexible materials. The multi-material printing capabilities were crucial for this airfoil design. The flexure box with integrated compliant material hinges interfaced the rigid leading edge with the morphing trailing edge. This allowed a shearing motion that amplified the maximum tip displacement of the morphing trailing edge [34]. I assembled the morphing trailing edge using 2 MFC unimorphs, one on either side of the airfoil. I constructed each unimorph by bonding a M8557-P1 MFC to a 0.025 mm thick sheet of stainless steel shim. The morphing section also included two Flex Sensors from Spectra Symbol, which function as unidirectional variable resistors. To increase the sensor sensitivity, I bonded each flex sensor to a 0.025 mm thick strip of stainless steel shim as well. The flex sensors measured the internal displacement

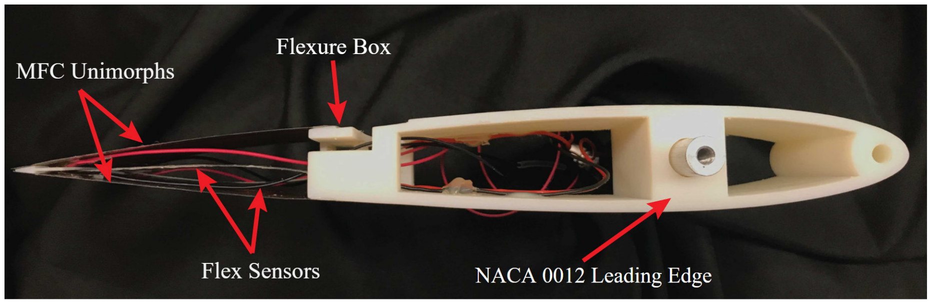

Figure 3.2: Image of the morphing airfoil system including two antagonistic MFC unimorphs, two piezoelectric flex sensors, the multi-material flexure box, and NACA 0012 leading edge.

of the morphing trailing edge, and were wired in a voltage divider configuration.

The control architecture for the morphing airfoil can be seen in Figure 3.3. A Keyence LJ-V7300 2D profilometer gathered true deflection information, and two piezoelectric flex sensors within the morphing section provided signals for deflection inference. This sensor information was fed through an Arduino Mega to a laptop where it was stored during data collection for model training or used for action selection via a Python script in Jupyter notebooks. For controller deployment, either true deflection information or a flex sensor signal in conjunction with one of the two inference models (LIN or LSTM) was used to provide state information to one of the three controllers (PD, RL, MO) to determine an output voltage signal [66]. From the Python script, a voltage signal was sent to the Arduino Mega and converted into a pulse width modulation (PWM) signal for the voltage amplifier. From there the corresponding voltages were supplied to the antagonistic MFC unimorphs that form the trailing end of the airfoil.

3.3.2 Data Collection

Reinforcement learning, although a powerful tool, is time consuming due to its trial and error format. To refrain from subjecting the system to unnecessary wear and tear, I created a simulation in which I could experiment with RL methods to

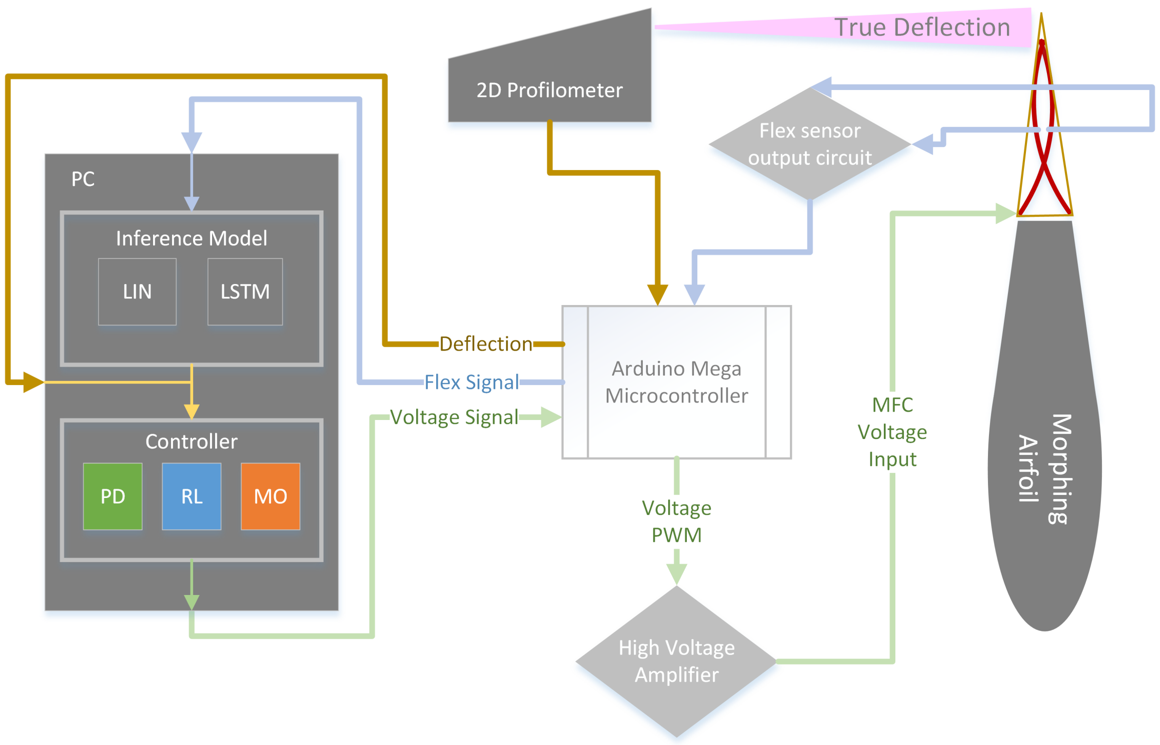

Figure 3.3: Data flow diagram for the morphing airfoil experiment. Deflection information was captured through the 2D profilometer and two piezoelectric flex sensors and provided to a laptop for data collection and controller decision making.

develop a sufficient controller. Since I performed all training in simulation and transferred that controller directly to the hardware system, any inaccuracy in this model would contribute to poor controller performance. In addition to the complex nonlinear behavior of the system, any imperfections that occurred in manufacturing the composite wing provide a potential for variation in the behavior of the MFC actuators. To accomplish accurate modeling specific to the morphing airfoil's dynamics, I collected true deflection and flex sensor information from randomized sweeps of the action and state space of the morphing trailing edge.

During data collection, I ensured sufficient coverage of the state-action space by applying a range of voltage changes from a randomized series of initial voltages. The randomization of the voltage selection was crucial to accurately capture the hysteretic behavior of the MFC system. The set of initial voltages included all even percentages of possible voltage outputs. For this case, a voltage signal of 0 represented the largest negative supplied voltage and a signal of 100 was the largest positive supplied voltage. Thus, 50 was a neutral supply of 0 volts. From the initial voltages, a random voltage change signal was selected from the range of even values between -30 and 30. This change in voltage signal was applied to the MFC actuators for 100 timesteps of 0.05 seconds, supplying the new voltage for five seconds in total, after which the initial voltage was again supplied for another five seconds. This was repeated until the set of voltage change signals was exhausted and then restarted for the next randomly selected initial voltage signal. This process was repeated ten times for ten different random seeds.

3.3.3 Modeling Dynamics

Following data collection I implemented three neural network structures to comparatively model the dynamics of the system, consisting of a multi step dense (MSD) network, a one-dimensional convolutional neural network (CNN), and a long short-term memory (LSTM) network [97]–[99]. The input for each of the models included state information over the ten previous timesteps. Each timestep state consisted of the current deflection value and the current and previous voltage signal. From this input, the models predicted the deflection for the next immediate timestep. Of the ten datasets collected using different random seeds, the first nine were combined and split into an 8:2 ratio for training and validation respectively. The tenth data set was not included in the training process and was used solely for testing. I found that the LSTM model achieved the lowest error in both validation and testing (Fig. 3.4a). This is visualized by the LSTM model's ability to accurately represent the system's dynamics for a 100 second example section of the collected testing data (Fig. 3.4b). The dynamics modeled here were based on the true deflection of the morphing trailing edge and did not include any information from the piezoelectric flex sensors. This must be considered when implementing controllers trained in a simulation informed by this model.

3.3.4 Reinforcement learning environment and controllers

Many RL algorithms have been developed to fit a variety of learning problems. Model free methods remove computational complexity attributed to learning a model of the environment. Instead they focus on learning a function to dictate which actions are preferred given the current state [46]. That function is known as the policy. Focusing on learning a policy, as opposed to an environment model, creates a reactive controller concerned only with the current state instead of selecting actions based on predicted outcomes. On-policy methods use the learned policy for all decision making and are frequently safer since they actively avoid states that result in low reward during training [46]. It is for these reasons that I chose to use the model free on-policy algorithm, PPO [72].

I developed the learned controllers using PPO in a simulation informed by the

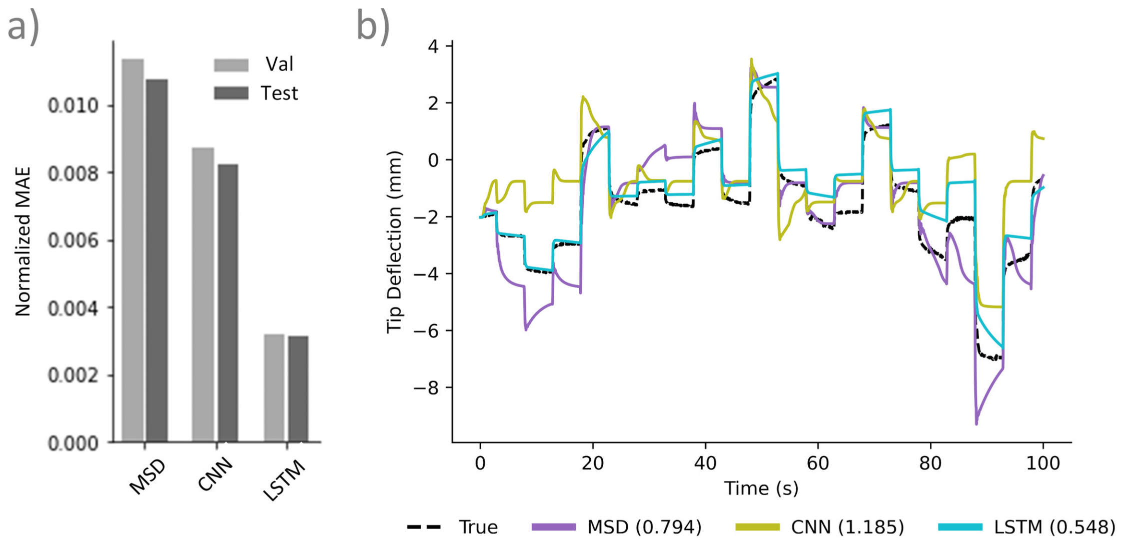

Figure 3.4: Comparison of learned dynamics models, including multi-step dense (MSD), convolutional neural network (CNN), and long short-term memory (LSTM). The normalized mean absolute error (MAE) earned during training validation (Val) and testing (Test) is presented in a), and b) illustrates the performance of the different models over a 100 second section of data collection when given the true state at time 0. Mean absolute error, in terms of millimeters, over the 100 seconds for each model is shown in parentheses.

LSTM dynamics model. PPO is among the top RL algorithms in many Atari, OpenAI, and MuJoCo environments and is frequently a baseline for comparison for new algorithm performance [70]. At its base PPO is an actor-critic method. This means that it approximates two functions, the critic, a value function that represents the preferability of being in a given state, and the actor, that learns the policy π. Given that there are two neural networks to update in PPO, there are also two loss functions to be combined when performing gradient ascent. The actor and critic loss functions are defined as follows,

$$\mathbb{L}^{CLIP}(\boldsymbol{\theta}) = \hat{\mathbb{E}}[\mathbf{min}(r(\boldsymbol{\theta})\hat{A}_t, clip(r(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t], \tag{3.1}$$

$$\mathbb{L}_{critic}(\boldsymbol{w}) = (V_w(s_t) - V_t^{target})^2, \tag{3.2}$$

where A_t is the advantage at time t described by

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + ... + (\gamma\Lambda)^{T-t+1}\delta_{T-1}, \tag{3.3}$$

with

$$\delta_t = R_t + \gamma V(s_{t+1}) - V(s_t), \tag{3.4}$$

and $r_t(\boldsymbol{\theta})$ is the ratio between the new and old policy for the current action, a_t and state s_t,

$$r_t(\boldsymbol{\theta}) = \pi_\theta(a_t|s_t)/\pi_\theta^{old}(a_t, s_t) \tag{3.5}$$

In the above equations, $\boldsymbol{\theta}$ is the vector of weights for the policy network, $\boldsymbol{w}$ is the vector of weights for the *value* network, and V is the *value* of a state. The parameter γ is a discount factor used to gradually degrade the impact of future state *values* on the current state *value*, hence emphasizing the impact of more imminent states. The smoothing factor, Λ, is used to reduce the variance in training and improve stability. Finally, the main difference between PPO and previous actor-critic methods comes from the actor loss function where the clipping factor, ϵ, is introduced to limit overall step size of the policy update to prevent an individual update from growing too large. The values of these parameters, γ, λ, and ϵ, were set to equal 0.99, 0.95, and 0.2 respectively, which are the values suggested by Schulman when first presenting PPO [72]. the implementation of PPO was drawn from an opensource example in Pytorch [100]

The network structures for both the actor and critic are presented in Figure 3.5. State observations presented to the controller consisted of the current normalized deflection observation, the normalized goal deflection, as well as the current and previous normalized voltage signals for each timestep. I used a 1D CNN for the initial layer of both the critic and actor networks in the PPO algorithms, each followed by three fully connected layers with rectified linear unit (ReLU) activation functions [98], [101]. This input layer was given the ten most recent state observations and goal deflections, providing additional temporal information to both the critic and actor for value approximation and policy generation. Instead of using a continuous action space to cover the large selection of voltage signals, I used a smaller discrete action space consisting of seven potential changes in voltage signal ranging from -6 to 6. Training consisted of 5000 training episodes, each lasting 200 timesteps and beginning with randomized initial conditions and goal deflection values. I trained two controllers in the simulated environment. For the first learned controller the action space included voltage signal changes [-6, -4, -2, 0, 2, 4, 6] and a simple reward scheme that distributed negative rewards equal to the squared error between the current deflection and the goal deflection. After initial testing of this controller in the physical hardware environment, I noticed room for improvement regarding the overshoot experienced. In the second learned controller I chose to include smaller potential voltage changes within the action space, [-6, -2, -1, 0, 1, 2, 6], with the goal of achieving finer control. Additionally, I augmented the original reward scheme

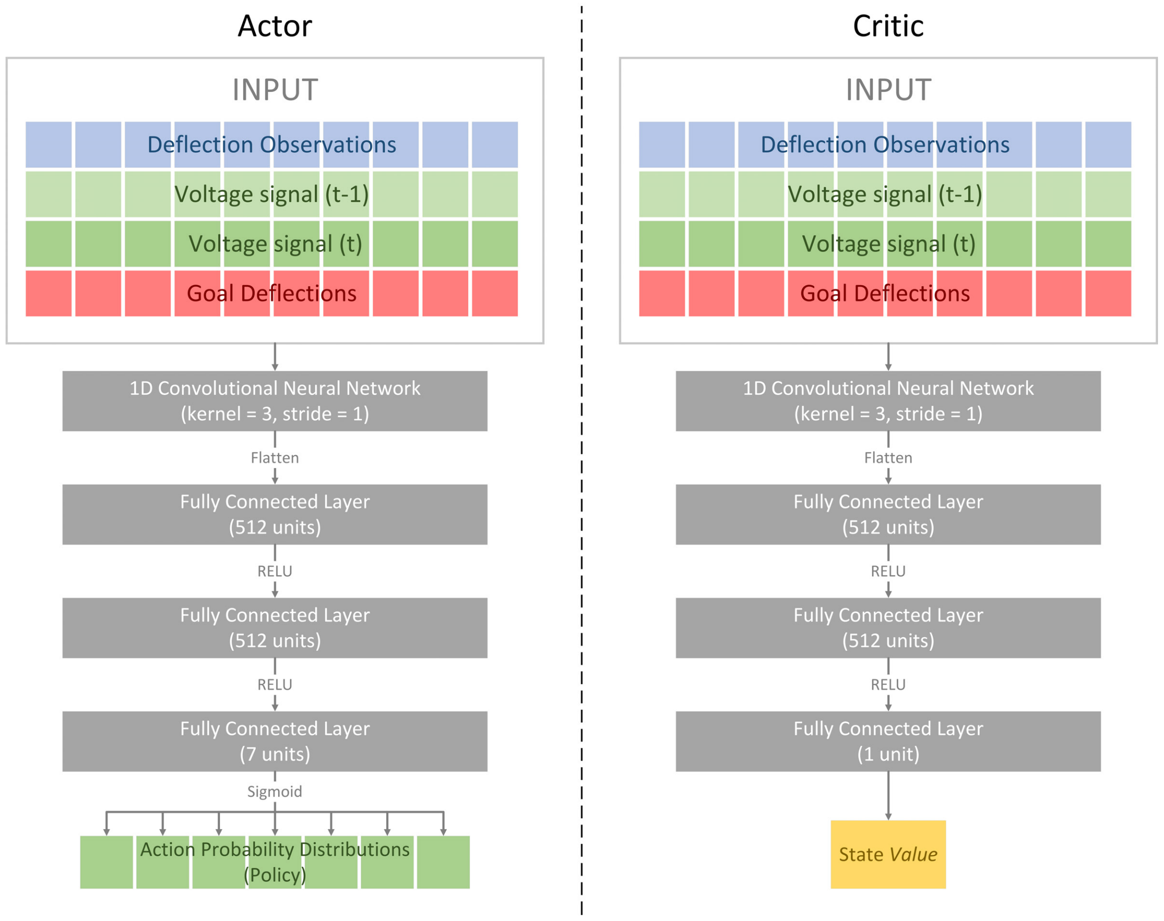

Figure 3.5: Actor and critic neural network structures. The actor and critic networks each have the same structure, including a 1D CNN layer and three fully connected layers with ReLU activation functions. The actor produces a probability distribution for each possible action, and the critic a single *value* representing the estimated long term expected reward of the current state.

for the second learned controller to include an extra penalty equal to ten times the original cost when the controller experienced an overshoot greater than 1% of the tested response step size. Both these controllers were compared to a traditional PD controller. To distinguish between the two learned controllers, I referred to the initial controller with the simple reward scheme as the RL controller, whereas the second one trained with the amended reward scheme was labeled the mitigated-overshoot (MO) controller.

3.3.5 Modeling flex sensor

The controllers developed in simulation were based on true deflection information gathered by the 2D profilometer and did not account for errors that occurred in state estimation. However, in a realistic implementation of these controllers, state information would be observed through onboard sensors, in this case piezoelectric flex sensors, that give imperfect state measurements. Therefore, to provide accurate state observation, I used the information gathered during data collection (Sec. 3.3.2) to model the relationship between the piezoelectric flex sensor signals and the true deflection of the MFC trailing edge (Appendix 5.2)[66]. For the purposes of this model I used two methods, a traditional linear method (LIN) and a time series neural network. The LIN inference model was structured as neural network with a single node and a linear activation function, leading the model to behave as a sum of weighted state values and an additional bias. Initially, I trained models using information only from the current timestep and included supplied voltage values. Although these appeared to be accurate in training, implementation showed that the voltage values were heavily weighted in the model, neglecting the sensor values and ignoring the hysteretic behavior. Therefore I removed the voltage values from the state inference model input and used a time series of the 10 most recent timesteps to infer the current deflection. Each time step consisted of the current sensor reading and the previous estimate of

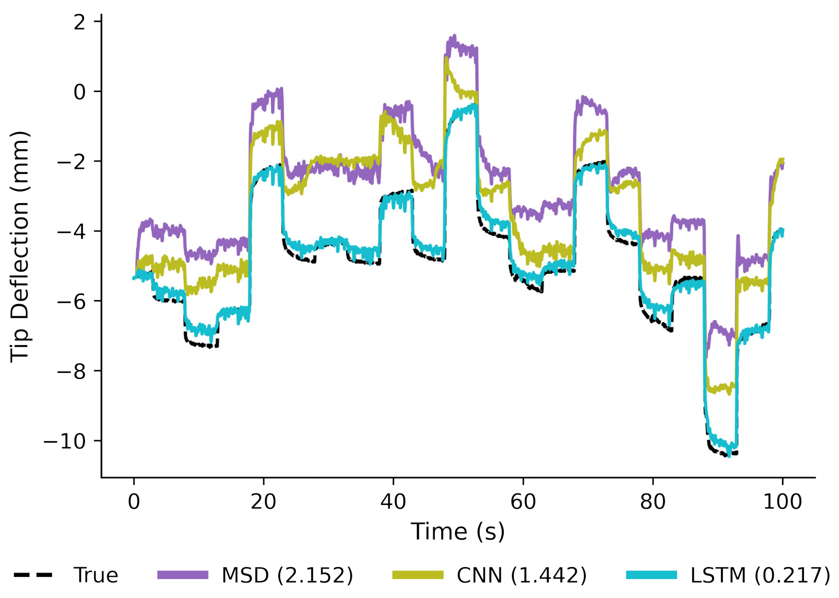

Figure 3.6: State inference comparisons. This plot compares the flex inference models using 3 different neural network structures, including multi-step dense (MSD), convolutional neural networks (CNN), and long short-term memory (LSTM), with the mean absolute error shown in parentheses. The LSTM inference model provided the most accurate state approximation.

the true deflection. Once again I found LSTM networks provided the most accurate prediction when compared to the other neural network structures (Fig. 3.6).

3.3.6 Experiment

Using the same data flow strategy as described in Figure 3.3, for each test I implemented a series of step responses spanning a portion of the state space ranging from normalized deflection values of -1 to 1. The composite airfoil began each test in a neutral position without deflection and performed eight step responses of magnitude 0.5 (3.31 mm), beginning with two positive steps, followed by four negative steps, and finishing with two additional positive steps to complete a cycle within the designated testing space. Similar to data collection, each step response was held for 100 timesteps before transitioning to the following step. Due to the black-box nature of these learned controllers, stability is still an open research problem in RL control[102]. For this reason, I use repetition to empirically show what response and overall performance can be expected from these controllers. Therefore, I repeated each test five times for each controller and state observation method, providing us 40 step responses for each controller-observation method combination. As a point of comparison, I used the Ziegler-Nichols open loop method to tune a PID controller; however, the controller experienced high overshoot and integral windup [103]. To mitigate this, I used two anti-windup methods including the addition of an anti-windup term based on controller saturation, as well as a reduction of the integrated error value [104]. I found that dropping the integral term, and therefore using a PD controller, produced an accurate controller that limited the overshoot when compared to the other PID methods. This was particularly true when paired with the flex sensor inference methods, providing a fair and interesting baseline for comparing to the learned controllers (Appendix A). Figure 3.7 presents the step response cycle for a PD controller using each of the inference models in an unloaded environment. Both the inferred position (blue), as determined by the flex sensors and the respective inference model, and the true position (green), as determined by the 2D profilometer, are illustrated. Although the inferred deflection often followed the target position closely, the true deflection was sometimes off by several millimeters for the less accurate inference models. Increasing in complexity, and accuracy, from the linear model to the LSTM model, I found a decrease in state estimation error that is particularly noticeable in the intermediate steps. For the remainder of this paper, although state information was provided to the controllers using each of the inference methods, performance metrics and comparisons were made based on the true deflection achieved by the MFC airfoil.

I compared controller performance using several metrics including three traditional controller step response metrics: rise time, settling time, and overshoot. The rise time and settling time were measured from the beginning of a step response until the true

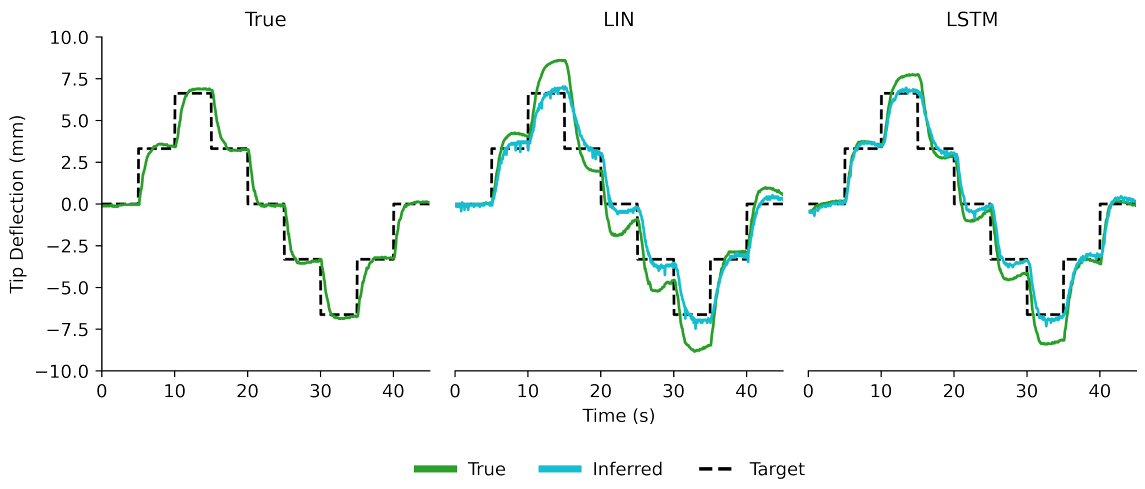

Figure 3.7: Controller performance changed with state observation accuracy. The far left plot shows the performance of the traditional PD controller when given true deflection information. The following two plots show the true deflections and inferred deflections experienced by the PD controller when provided the linear (LIN) and LSTM inference models.

deflection first crossed, or remained within, a 10% maximum test deflection boundary of the goal position. The overshoot was measured as a percentage with respect to the size of the step response. I included an additional metric common to RL: total earned reward. This was the value optimised by the RL algorithm. For consistency, I chose the simpler reward of squared error between the true position and the goal position as the metric for each of the controllers. As an error based metric, I used it as an indicator for the overall accuracy achieved in a test run.

MFCs are known to perform differently under mechanical loads, but all of the data used for training the controller and state estimation models were collected without consideration for aerodynamic loading [34]. Therefore, to seriously consider these controllers for autonomous UAV flight, I performed the same step response tests for a variety of flow speeds. For this purpose, I repeated the testing process in a 1'x1' (30 cm x 30 cm) wind tunnel for three flow speeds, 5 m/s, 7.5 m/s, and 10 m/s, to determine the controllers' ability to adapt to the environmental differences. These tests provided information on the learned controllers' performances in the presence of aerodynamic loading. It was not within the scope of this project to perform any aerodynamic analysis of the airfoil and therefore the angle of attack was maintained at $0°$.

Note that prior to conducting the wind tunnel tests a new flex sensor circuit was integrated into the composite wing, due to a malfunction in the original. The circuit was built in the same manner, however there was a noticeable shift in the sensor readings. To account for this, the mean and standard deviation of the sensor values were adjusted, allowing the normalized values to be similar to those experienced prior to the changed circuit. All testing performed within the wind tunnel was subject to this change and therefore comparisons between controllers with shared flow speeds were fair.

3.4 Results

3.4.1 Unloaded environment

For the initial experiments I compared the learned controllers to that of the tuned PD controller in the environment in which they were trained, without aerodynamic loading. The step response tests for each controller when provided true state information and using each of the 3 inference methods are visualized in Figure 3.8. I found that each of the controllers were able to track the target value with precision when accurate state observations were provided by the 2D profilometer. The learned methods reached the desired deflections more rapidly than the PD method, although the RL controller often overshot the target. The MO controller improved on this while still maintaining most of the speed seen by the initial RL controller.

Next I investigated the three metrics (rise time, overshoot, settling time) as well as the total earned reward (Fig. 3.9). This figure visualizes the eight step responses in

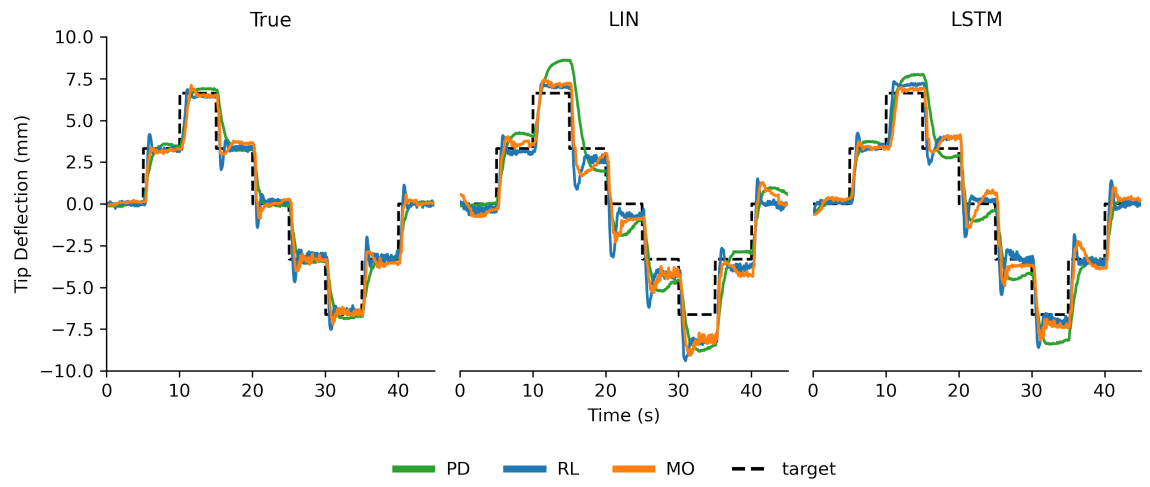

Figure 3.8: Controller tracking comparisons.This plot shows the step responses of each controller (PD, RL, and MO) in an unloaded environment when supplied true deflection information as well as state observations provided by the two piezoelectric flex sensor inference models (LIN and LSTM).

each of the five tests for the three controllers. From this, each controller's strengths and weaknesses are shown. Of the three controllers, PD (green lines) consistently earned the lowest reward. This was expected because the two learned controllers directly used this reward, or at least some form of it, to develop their policies. In addition to the low reward, the PD controller typically had a longer rise time, and a lower overshoot percentage. Although these characteristics allowed for smoother control, they may have led to the observed higher settling times (Fig. 3.9). In contrast, the RL controller (blue lines) achieved the highest reward and fastest rising times, but at the cost of much higher overshoot percentages. However, even with the higher overshoot, the RL controller had fast settling times that frequently outperformed those of the PD controller. Finally the MO controller (orange lines), found a middle ground between the two aforementioned, with medium rewards and rising times that were generally faster than the PD controller but typically not as fast as the RL controller. Where the MO controller greatly improved over the RL controller was in its overshoot percentage, experiencing much lower values than the PD controller.

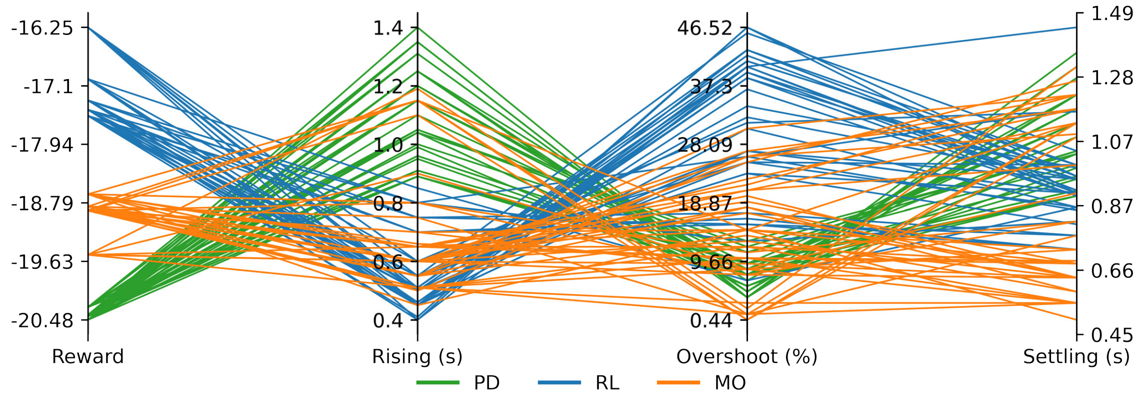

Figure 3.9: Performance metric comparisons. Comprehensive presentation of performance metrics achieved by each controller given true state observations. This plot includes every step response performed in testing and can therefore be used to determine expected trends for each controller's performance.

The MO's performance highlighted how the adjusted reward scheme led to a lower overshoot percentage. Additionally the MO controller achieved the fastest settling times of the three controllers.

Although the improved performance of the learned controllers, especially the MO controller, over the traditional PD method built confidence in these more complex controllers, it is not realistic to expect an airfoil controller to have perfect state observations during flight. Thus, I conducted the same step response tests using both of the state inference methods (LIN and LSTM). I found a substantial impact of the state observation accuracy for each of the controllers (Fig. 3.7). Interestingly, although the different controllers were provided the same modeling methods for state inference, in some cases the learned controllers were quicker to overcome the obstacles offered by inaccurate estimation, settling on deflections closer to the designated goal than that achieved by the traditional PD controller. This was particularly apparent in the early steps for the linear model, and the intermediate steps for the LSTM model. For a more direct comparison, Figure 3.9 considers each of the performance metrics and presents the average performance difference between a given learned controller and inference model combination and the PD controller when using the same estimation method. The error bars represent 95% confidence intervals to illustrate significance in difference. With the exception of overshoot, both the learned controllers outperformed the PD controller on average. This difference was significant in all cases when considering rise time, and for 5 separate controller/inference model combinations in reward and settling time. As mentioned before, overshoot was the metric where the learned controllers struggled the most; however, when paired with the linear model for state observation, the MO controller trended towards achieving a lower average overshoot than the PD controller.

3.4.2 Loaded environment

The step response tests for all three controllers when given perfect and imperfect state information in the tested environment of 10 m/s flow speed are visualized in Figure 3.11. These loaded responses with true state information indicate that the controllers responded quickly and accurately to the various changes in goal deflection, similar to that seen without loading. There were similar trends to those seen in the unloaded testing when comparing the performance metrics of the controllers in a 10 m/s airflow environment (Fig. 3.12). Interestingly, the learned controllers outperformed their unloaded condition, specifically the MO controller with regard to overshoot.

The step response plots in Figure 3.11 provide an example of each controller's step response given the available state observation models. Unlike when given true state observations, the control was less smooth. This may have been caused by vibrations in the sensors due to airflow. To further investigate the results, I examined the average absolute error between the true state and model observed state (Fig. 3.13). I found a substantial improvement in the linear model accuracy from the unloaded

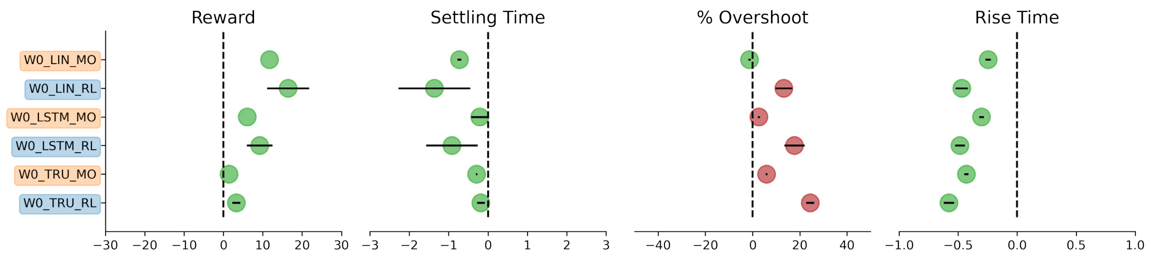

Figure 3.10: Direct comparison of average performance achieved by learned controllers in the unloaded environment for each metric to those achieved by the PD controller given the same state observation method. Comparisons that trend in the favor of the learned controller are in light green, and those in favor of the PD controller are in dark red. 95% confidence interval error bars are provided to clarify significance of perceived difference. Labels on the y axis are color coded, those in blue represent comparative RL controller performances and those in orange depict comparative performances achieved by the MO controller.

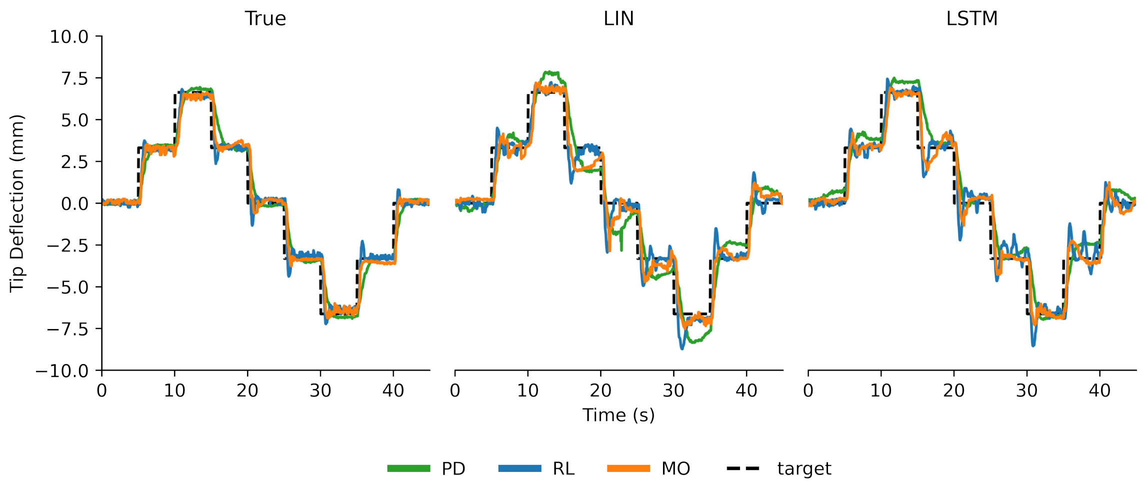

Figure 3.11: Loaded step responses. This plot shows the step responses of each controller (PD, RL, and MO) in an 10 m/s aerodynamically loaded environment when supplied true deflection information as well as state observations provided by the two piezoelectric flex sensor inference models (LIN and LSTM).

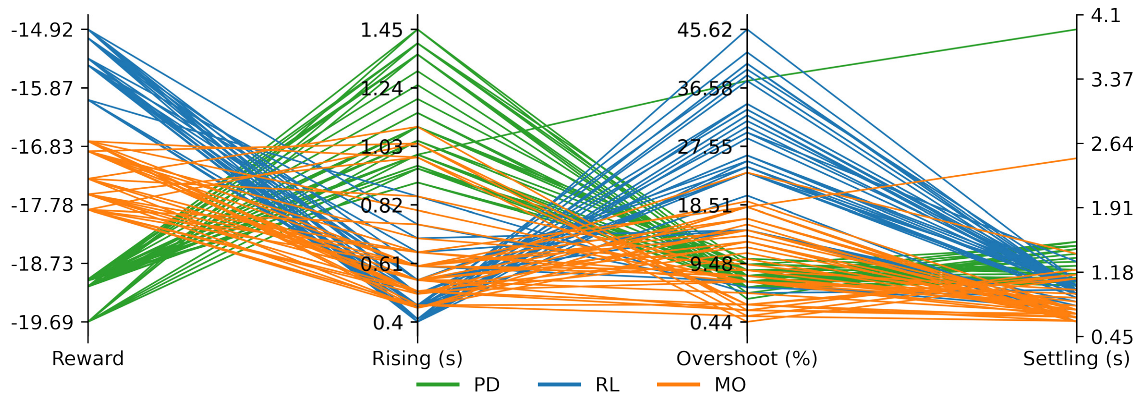

Figure 3.12: Loaded performance metrics. Comprehensive presentation of performance metrics achieved by each controller given true state observations in a 10 m/s airflow environment. This plot includes every step response performed in testing and can therefore be used to determine expected trends for each controller's performance.

environment in the wind tunnel tests, possibly due to the normalization adjustment mentioned previously. The LSTM model was not substantially affected, in some cases performing better and other cases performing worse. This suggests that the LSTM model generalized well to the adjusted environment.

As with the unloaded testing, the main objective was to compare the performance of the controllers developed through RL to that of the traditional PD controller. These direct comparisons are presented in Figure 3.14 for all three flow speeds. The learned controllers no longer completely dominated the settling time and reward metrics. For all but one of the tests in which the PD controller outperformed the compared learned controller in at least one of these two metrics, the most complex state estimation method (LSTM) was used. The one exception to this was for the average settling time when the controllers had access to true state observations. In this case the average difference between the PD controller and the RL controller was 0.014 seconds. This difference was less than the time-step size of 0.05 seconds, and therefore was not significant enough to consider a trend in either direction. Additionally, of the

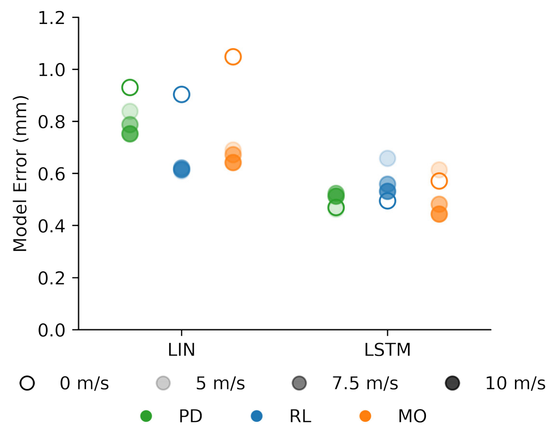

Figure 3.13: Average error for all controller/flex model combinations at all four flow speeds. Noticeable improvement in linear inference error is possibly due to the normalization adjustment made between unloaded and loaded environment testing.

tests where the PD controller outperformed the learned controller, this difference in performance was only great enough to be considered significant in two of the cases according to the 95% confidence intervals, but the trend was still worth mentioning.

On the other hand, when using the linear model for state estimation, both learned controllers significantly outperformed the PD controller at all flow speeds. The MO controller specifically, when combined with the linear inference model, was the only controller-inference model combination to outperform the PD controller, on average, for all four performance metrics, including overshoot, at all three flow speeds and at rest. This does not mean that this was the best controller-inference model combination overall, only that it performed better comparatively given the same flow speed and state estimation conditions. To determine the best performances overall, Figure 3.15 presents the average performance metric values for each controller, inference model, and flow speed.

Assuming perfect state observations, the previous comments were further validated. The learned controllers excelled in achieving fast and accurate control, according to rising time, settling time, and reward. The PD controller still achieved the

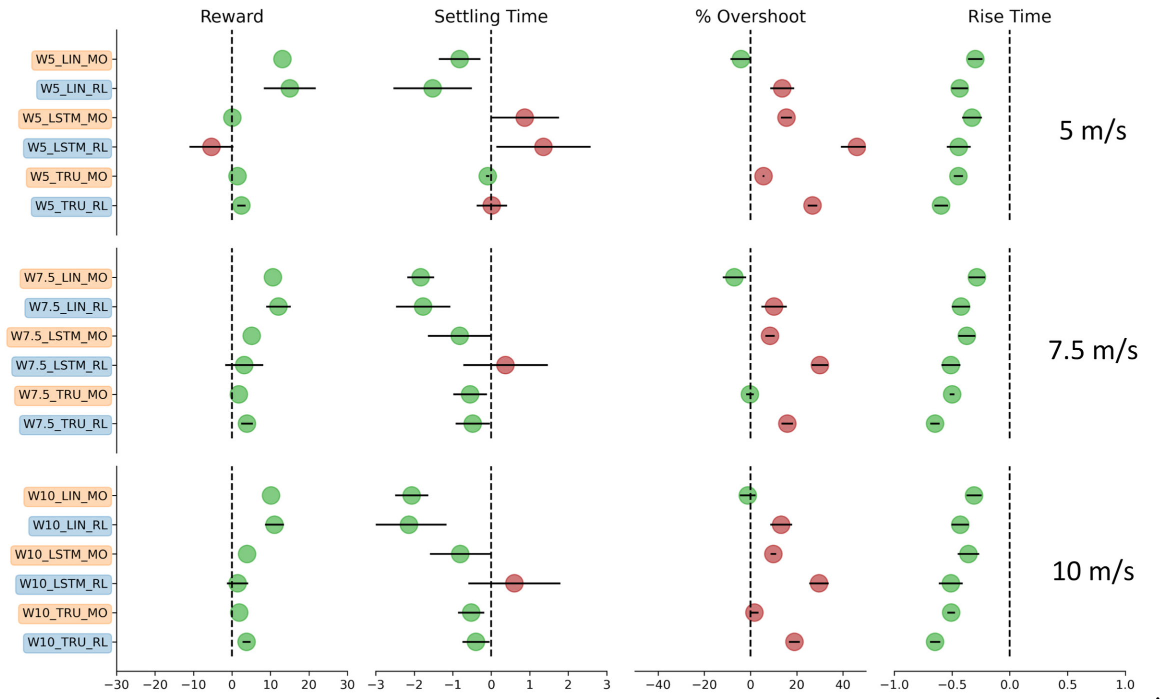

Figure 3.14: Average performances achieved by learned controllers in the aerodynamically loaded environments. Each metric is directly compared to that achieved by the PD controller given the same state observation method. Comparisons that trend in the favor of the learned controller are in light green, and those in favor of the PD controller are in dark red. 95% confidence interval error bars are provided to clarify significance of perceived difference. Labels on the y axis are color coded, those in blue represent comparative RL controller performances and those in orange depict comparative performances achieved by the MO controller.

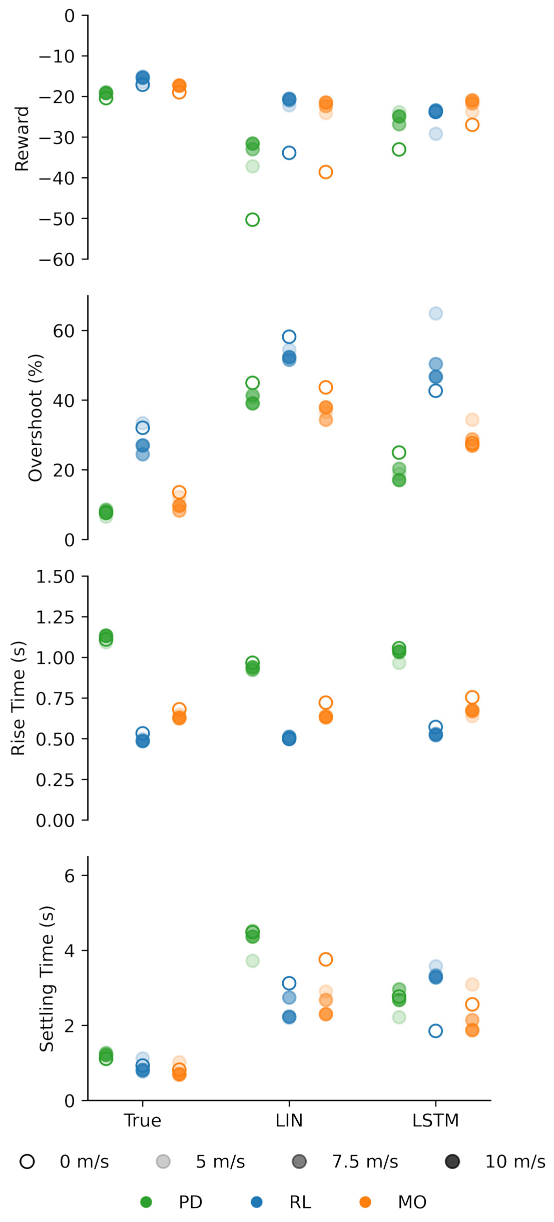

Figure 3.15: A summary of the average performances for each controller, under all flow speed conditions, and with both observation methods.

lowest overshoot of the controllers on average, but the MO controller produced only 2.13% greater overshoot on average. This was a great improvement over the additional 10% of average overshoot produced by the RL controller than the PD controller. I found a continuation of this trend when considering imperfect state observations. I found the learned controllers consistently outperformed the PD controller when using the linear flex sensor model, LIN. The PD controller only outperformed one learned controller, the RL controller, in the single metric of overshoot. I found that the more accurate LSTM model improved performances for all controllers in the unloaded environment and most cases in the Loaded environment. Interestingly, the RL controller did not see notable improvement in performance from the more accurate inference model in the loaded environment, but the MO controller did, showing decreased overshoot compared to using the LIN inference method and achieving the fastest settling times and highest reward of all 3 controllers. The improved accuracy produced noticeable improvement in most metrics for the PD controller as well. Given these findings, the MO controller appeared comparable, and often preferred, over the traditional PD controller, especially for instances where rapid control and overall accuracy was the primary focus.

3.5 Discussion

These results provide insight for future controller design in the pursuit of *fly-by-feel* solutions for morphing composite wings. I have validated the use of learned controllers in the physical multifunctional morphing airfoil environment. Furthermore, I found that through DRL techniques, I developed controllers superior to a traditionally tuned PD controller. This was particularly true when emphasizing speed and accuracy in control. For instances where overshoot is the only metric worth optimizing, choosing a traditional PD controller with the most accurate state inference method available (True or LSTM) would be the primary option. However, this is rarely the case. When considering all controller metrics used in this research, the learned controllers show superior overall performance. Additionally, the MO controller produced comparable speed and accuracy to the RL controller while greatly reducing the overshoot. This superior performance was most emphasized when state inference complexity was limited to a linear model. Because the learned controllers most clearly outperformed the PD controller when using the least accurate inference model, I suspect the learned controllers were internally accounting for the hysteretic behavior of the system. Since the learned controllers used 1D convolutions of the ten most recent time steps of state information, RL and MO learned to recognize the nonlinear pattern within the dynamics of the system to better inform action selection. The PD controller had no internal mechanism to recognize hysteresis and therefore relied heavily on the accuracy of the feedback signal.

The improvement achieved by the MO controller brought to light another question: can I further optimize the controller through additional reward engineering? There is a philosophy that achieving general intelligence in a trial and error format only requires a simple reward structure that captures the goal of the controller [105]. This was the philosophy I followed when designing the first controller (RL), and found it created a strong controller with emphasis on speed and accuracy. However, after seeing it struggle to mitigate overshoot, I added a rule to the reward scheme. This amendment created a controller with an impressive balance between speed, accuracy, and overshoot (MO). It may be argued that for this purpose, mitigating overshoot falls within the bounds of the goal that must be characterized by the reward function. Others may suggest that this is an example of the Reward Engineering Principle: "as reinforcement-learning-based AI systems become more general and autonomous, the design of reward mechanisms that elicit desired behaviours becomes both more important and more difficult" [106]. Regardless of philosophy, I tested one reward function augmentation, and in doing so developed a highly effective controller for the desired purpose. This suggests that there are a variety of controllers that can be learned, each with adjusted reward schemes designed to emphasize controller characteristics crucial for an individual control problem on the MFC morphing system. Additionally, this idea of greater customization in controller development can lead to larger and more complex problems with multifunctional MFC airfoils.

This project showed that MFC morphing UAVs present an environment where RL is not only a possible solution, but often a preferable one. However, the control problem in this work was limited to the basic functionality of the multifunctional morphing airfoil under loading at a single neutral angle of attack. On this basis, future projects can look toward using RL to pursue goals more complex than achieving a desired tip deflection. This may include stall rejection, efficiency optimization, or gust alleviation [43], [107], [108]. Additionally, in this work, all training was performed in simulation. If I aim to produce truly adaptive controllers for complex varying environments, another avenue for future research is to pursue these goals with real time learning on the physical hardware instead of offline in simulation. Finally, the learning algorithms and control commands were executed on an external computer, not contained within the airfoil. For autonomous morphing aircraft, future work will need to use adaptive learning techniques with embedded systems or neuromorphic chips [91], [109]. The next missing link to creating totally autonomous composite wing systems for morphing is understanding how to integrate computing chips into a composite material. Researchers must solve the mechanics of composite issues associated with embedding a computing chip into a layered composite to allow it to survive mechanical loads and thermal gradients caused by self-heating without degrading RL performance.

3.6 Conclusion

In this work, I presented the first successful application of RL to a physical MFC actuated system, outperforming traditional PD control methods. To achieve this, I compared three controllers for an MFC morphing airfoil: a traditional PD controller, a learned controller using a simple error based reward scheme (RL), and another learned controller incorporating an additional penalty to mitigate controller overshoot (MO). Through deep supervised learning, I accurately modelled the dynamics of the smart composite actuators, capturing their hysteretic behavior. These models were used to develop a simulation to train effective controllers for an MFC system through offline policy optimization. Due to the nonlinear hysteretic MFC behavior, this environment required closed loop feedback for accurate control. For this reason, two state inference methods, consisting of a linear and an LSTM network, were used in coordination with piezoelectric flex sensors for imperfect on-board state observation. I first tested these controllers with the different state observation methods in an unloaded environment and then at three flow speeds. I used a tuned PD controller as the comparative baseline.

I found both learned controllers to be comparable, and in many cases preferable, to the traditional PD controller. The MO controller in particular had impressive control effectiveness. This was especially true for instances where controller speed and accuracy were a priority. This result is promising for the field of autonomic morphing aircraft, where smart composites are used, and adapting to erratic environments is required.

CHAPTER IV

Learned Fly-By-Feel Gust Alleviation

4.1 Summary

Recently, there has been a growing expectation for UAVs to perform in a variety of new environments. Cities have been gaining interest for UAV operation; however, the tall buildings and street systems cause dynamic changes in the wind environment that pose a challenge for small UAV flight. Gusts of wind perturb small UAVs, detracting from effective mission completion and operational efficiency. Modern gust alleviation methods rely on traditional control surfaces and computationally expensive modeling to predict the effect of gusts before selecting a control action, leading to a delayed response. In this work I used DRL to create an autonomous gust load alleviation (GLA) controller for an MFC spanwise camber morphing wing, to reduce gust impact by 84%, directly from on-board pressure signals. Additionally, I found that performance differences between GLA controllers operating from three pressure tap signals and six pressure tap signals were not significant. Producing control decisions directly from pressure tap signals removes the need for computationally expensive dynamics models and state inference. This *fly-by-feel* style of control will allow small UAVs to react more rapidly to an environmental change, effectively reducing the impact of gusts. Achieving rapid gust alleviation improves UAV effectiveness in dynamic aerial environments to enable mission completion in previously inoperable locations.

4.2 Motivation and Background

In recent years, civilians and the military have added urban areas to the list of environments in which UAVs are expected to perform. Be it for surveillance, reconnaissance, or package delivery, UAVs must be capable of adapting to the dynamic environment offered by a large city [110]–[113]. The presence and position of tall buildings creates additional wind conditions that a UAV must account for during flight [114]. This is not limited to the strong winds experienced within passageways between buildings, but vertical wind velocities, such as updrafts and downdrafts, are strongly amplified in regions above buildings and street systems within a city [115]. A UAVs inability to adjust to these erratic environmental changes could result in a dropped package, inadequate surveillance, or a failed mission. For this reason, I look to alleviate the impact of these aerodynamic perturbations, or gusts, on UAV flight.

Gust response has been an interest in the field of aerospace since early aircraft design [21], [116]. Perturbations impact flight performance and tracking predefined trajectories [117]. This is especially true for small UAVs due to their lightweight nature. Traditionally, a pilot, or autopilot, responds to a perturbation with an antagonistic action. For instance, when a gust pitches the aircraft upward, the natural response is to deflect the elevators downward to apply a counteractive negative pitching moment. If maintaining a specific altitude is an important mission parameter, such as nap-of-the-earth flight, the aircraft would require additional negative pitch to achieve the correct effective angle of attack and lift through a sustained upward gust [118]. However, these corrections occur after the external force has accelerated the aircraft upward, and therefore, additional corrections are necessary to put the aircraft back on track, assuming the initial acceleration did not compromise the mission. Instead of responding to a perturbation after it occurs, it is beneficial to adjust the aerodynamic forces on the wings directly by changing the lift produced during a gust with GLA [13], [21]. In addition to mitigating deviations from the desired trajectory,

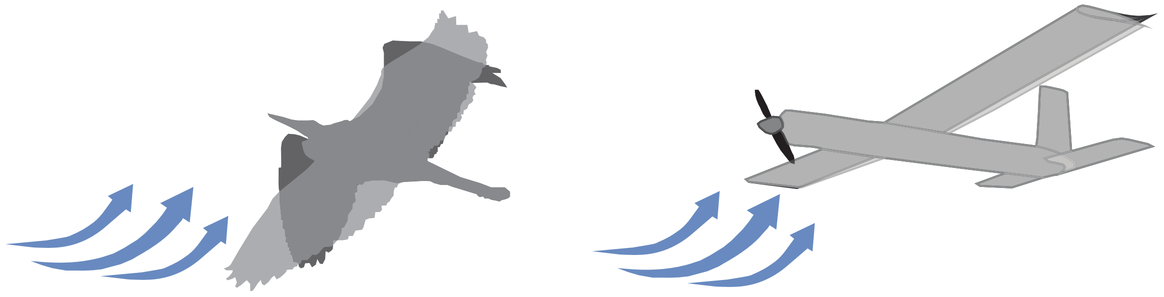
Figure 4.1: Birds change the shape of their wings to react to environmental changes. This inspired the use of camber morphing to create smooth shape changes that adjust lift production in response to gusts.

GLA reduces the loads experienced during a gust to alleviate critical stresses placed on the aircraft structure . This reduces the material, and therefore mass, improving overall aircraft efficiency.

For much of the history of GLA, traditional control surfaces, such as ailerons, elevators, and spoilers, have been used to mitigate the impact of environmental changes during flight [21]. Recently, unique methods have grown in popularity to improve actuation speed, efficiency, and mechanical complexity [14], [24]. One such method is structural morphing. UAV morphing is often inspired by a bird's ability to change shape to adapt their flight characteristics to a changing environment [119]. This ability has allowed birds of prey to adapt to hunting in an urban environment [120]. Similarly, morphing allows UAVs to perform smooth shape changes and maintain a continuous surface when actuated, improving adaptability, efficiency, and control effectiveness [1], [39]–[42], [119]. Birds are observed performing large shape changes to their wings when impacted by a gust [121]. Instead of performing a large shape change dominated by inertial effects for gust rejection, UAVs can use spanwise camber morphing techniques to reduce the aerodynamic loads placed on an aircraft wing during gusts (Fig 4.1) [24], [34], [122].

However, weight and volume constraints associated with small UAV design bring additional challenges for morphing implementation. Smart materials are frequently used to combat these challenges [1], [4]. MFCs have been used as multifunctional morphing mechanisms, acting as both actuator and skin to produce smooth camber morphing [30], [34], [36], [37]. This produces benefits in aerodynamic efficiency, speed, mechanical complexity, weight reduction, and overall improved control authority when compared to traditional rigid flap actuation methods [5], [43]. These characteristics make MFCs a desirable option as multifunctional actuators for GLA. However, MFC actuation also has attributes that make controlling the material challenging, including hysteresis, creep, and inconsistent performance under out of plane loading. Modeling and feedback control are often used to combat these challenges [5], [66], [123]. These challenges must be considered when using MFC camber morphing for GLA.

Many forms of control have been used for GLA, including feed-forward, PID, optimal control, and MPC, most of which require a model of the system [17]–[19], [124], [125]. Model-based controllers rely heavily on model accuracy to achieve sufficient control. Error in any of the models used before action selection propagates through the controller. This includes dynamics models as well as those used for state inference. Regardless of the control scheme, state observation is crucial for effective control. Typically the state of the system cannot be directly observed and therefore needs some form of state inference, such as a Kalman filter or a neural network, before the controller makes an action selection [19], [23]. Each model used for inference or prediction adds a layer of computational complexity to the control system and increases the time necessary to make a control action. Light Detection and Ranging (LIDAR) has grown in popularity for MPC and feedforward control to sense an incoming gust before perturbing the aircraft [21], [22], [126], [127]. This preview aides with potential delay in controller response, but still adds complexity that may not fall within a UAVs computational constraints [128]. Model-free DRL has shown enormous success in producing effective controllers capable of making action decisions directly from raw sensor inputs without using dynamics or state inference models [50]. Additionally, DRL has been shown to account for MFC hysteresis and produce effective deflection control in a camber morphing airfoil [67].

Reinforcement learning is a machine learning method used to develop controllers through trial-and-error [46]. Trial-and-error is often considered the “natural” method of learning in biological settings [46]. For instance, KleinHeerenbrink showed Harris’ hawks learn to minimize distance flown in stall when perching after repeatedly flying between two perches [129]. Through a structured reward system, RL autonomously develops a controller that maximizes the earned reward during training. Rewards are used to update the *value* and policy functions, where the *value* of a state-action is an estimation of long-term reward and the policy is what dictates action selection given the current state [46]. As control environments grow more complex, it is beneficial to use DRL, where deep neural networks are used as the base model structure to approximate the *value* and policy for each visited state and action. DRL has grown rapidly in popularity since achieving superhuman performance in a variety of complex environments [50], [130].

Using DRL to control adaptive UAVs is an ongoing research topic. Hu et. al used DRL to train a reconfigurable UAV to perform in different environments or mission scopes [93]. Motors and servos are traditionally used to perform these shape changes, but RL has also been applied to simulated and hardware smart material environments [7], [8], [96]. Recently, PPO was used to improve controller speed and accuracy for a MFC camber morphing environment [67]. PPO is an online actor-critic DRL algorithm that includes clipping to prevent training updates from overstepping optimal weights to achieve state of the art performance for training controllers in highly complex environments [72]. Most successful applications of RL are performed in simulation, but some environments are too complex to accurately model without a large computational cost. Methods used to accurately characterize gusts and turbulence are computationally expensive, and popular simplified engineering approximations are known for their high uncertainties [14], [131]. For this reason, there is a field dedicated to minimizing this uncertainty using data-driven approaches [132]. I avoid the computational costs and uncertainty associated with simplified approximation by using autonomous methods for training directly on the physical hardware environment [65]. In this work, I applied PPO directly to a physical morphing wing in a variable gust environment.

To create a gusting environment in which DRL could develop a GLA controller, I installed an MFC driven spanwise morphing wing in a wind tunnel downstream of a rigid airfoil acting as a gust generator (Fig 4.2). Deflecting the rigid airfoil changed the incoming freestream angle and velocity experienced by the morphing wing (Fig 4.3). [21], [23]. I used PPO to train controllers to make action decisions directly from on-board sensor signals in the physical hardware system. I used pressure taps in three different configurations for state observation [133]. The controllers achieved average performances ranging from 71% to 87% gust rejection for six gust conditions. Increasing the number of pressure taps improved overall performance and consistency, but the improvement between sensing configurations with three pressure taps and six pressure taps was not significant.

4.3 Methods and Materials

4.3.1 Morphing Wing Construction

I created the morphing wing with three 42 mm wide active sections separated by two 51 mm wide passive sections to form a 228 mm wide wing with a 320 mm chord. To construct the active sections, I followed the methods established in previous work, which combine a NACA0012 leading edge with a double antagonistic MFC unimorph trailing edge [34]. I used multi-material 3D printing to include a flexure box design at the interface between the rigid and morphing portion of the active wing section to

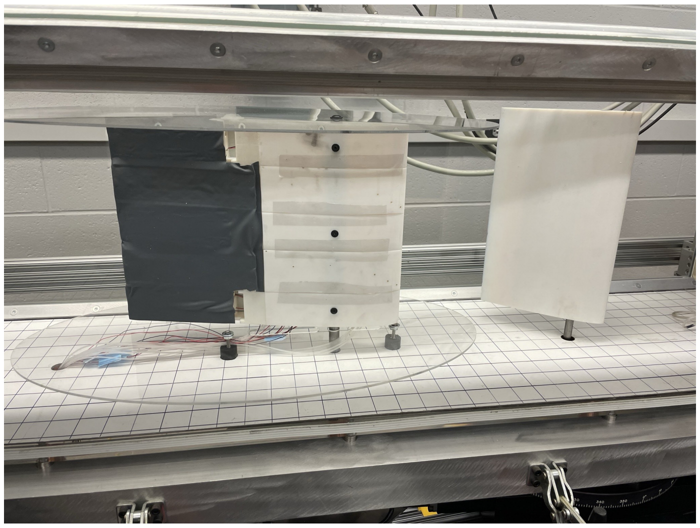

Figure 4.2: A rigid airfoil was mounted 30 cm in front of the MFC driven spanwise morphing wing. This airfoil was deflected to create a gusting environment within the University of Michigan 1' x 1' (30 cm x 30 cm) wind tunnel.

maximize deflection potential. Unlike in the previous work, I used narrower M8528-P1 MFCs to allow for three active sections to fit within the wind tunnel. Using epoxy, I bonded each MFC to a 0.025 mm stainless steel shim to produce a bending shape change when actuated. I used epoxy to attach the active trailing edge section to the flexure box interface at the rear of the rigid leading edge.

I constructed the passive sections following methods established by Pankonien et al. for a spanwise morphing wing [34]. The passive sections contain a rigid NACA0012 leading section, but don't have a rigidly structured trailing end. Instead, structure is provided by the spanwise skin extending across the full wing. Bonding a soft 3D-printed mixed cruciform honeycomb to the elastic silicon skin provided additional strength to the trailing edge of the passive section [28], [134]. This allowed the passive sections to change shape with the active sections while maintaining structural integrity under out of plane aerodynamic loading.

Within each passive section of the wing, I installed six 0.5 mm pressure taps for state observation. The pressure taps were located at positions of 0, 1.5, 5, 10, 40, and 50 percent of the chord length measured from the leading edge. I offset pressure taps at an angle of 30° for the front four pressure taps to mitigate the effect of upstream pressure taps on the flow [135]. Due to the large separation between the front four and rear two pressure taps, I installed the two rearmost pressure taps at a separate 30° angle, not including the front four taps to allow all taps to fit within the passive wing section . Each 1.5 mm pressure tap hole was included in the 3D printed NACA 0012 leading section of the airfoil. I used epoxy to fasten ethyl vinyl acetate (EVA) tubing into the pressure tap locations. After installation, I used a razor to cut the end of each pressure tap to be tangent with the surface of the morphing wing to avoid disrupting the flow over the wing.

4.3.2 Experimental Setup

The final morphing wing design was installed in the 30cm x 30cm wind tunnel at the University of Michigan (Fig 4.3). I mounted the morphing wing in the center of the wind tunnel at a 10° angle of attack operating at a 10 m/s flow, measured ahead of the gust generator. I included elliptical endplates on the wing to prevent wing tip vortices from forming, limiting this analysis to 2D airfoil effects. Mounted at the quarter-chord, I measured the morphing wing's lift using a six-axis ATI Delta load cell. Six compact differential low pressure transducers measured the pressures experienced by the six pressure taps in comparison to the static pressure located at the front of the testing section of the wind tunnel, as measured using a pitot-tube. Located 60 c m upstream of the morphing wing I mounted the 15 cm chord NACA 0012 rigid airfoil with a 25 cm span at its quarter-chord. I used a step-motor operated turntable to vary the rigid airfoil's angle of attack and create the desired gust deflection [21].

I used particle image velocimetry (PIV) to characterize the effects of the various gusts generated by the rigid airfoil (Fig 4.4). Oil based smoke particles were accelerated through the open-loop wind tunnel. An EverGreen double-pulse quantel laser

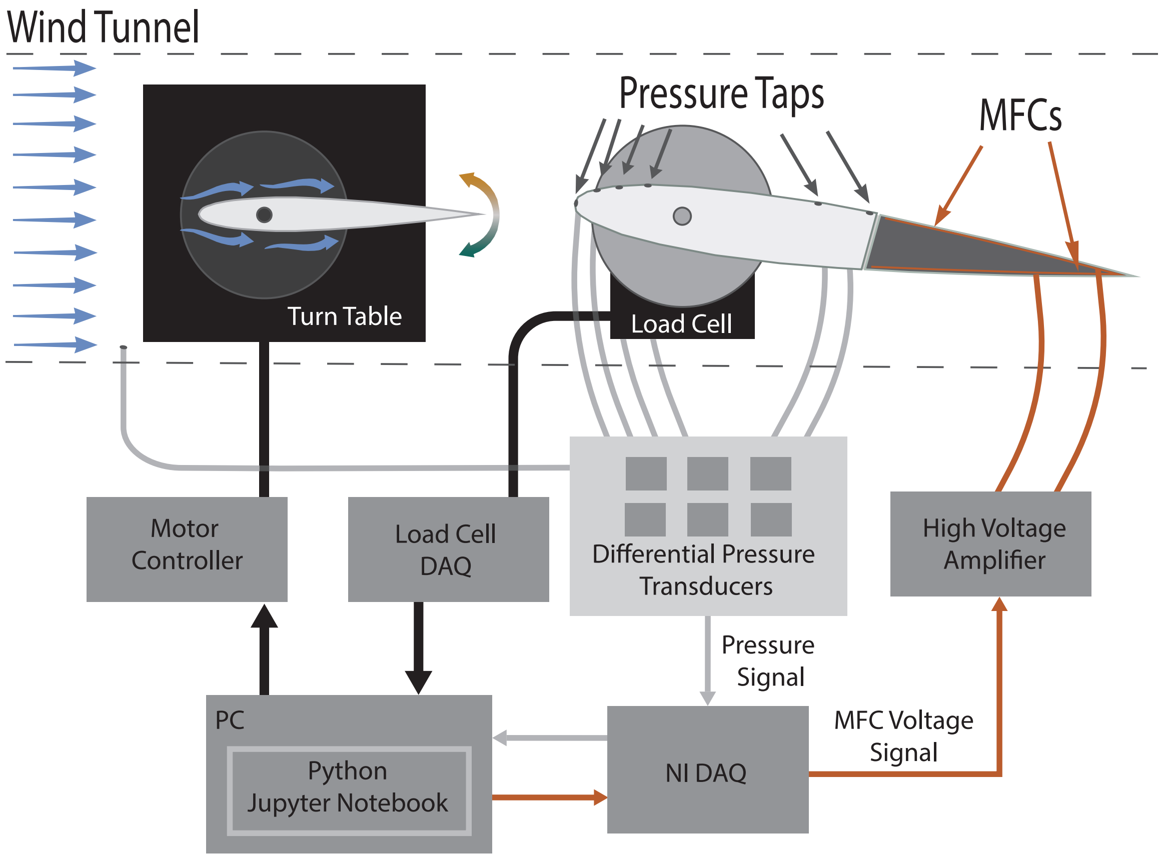

Figure 4.3: Data flow structure of the gusting wind tunnel experiment for controller training and testing. Training and testing were orchestrated using a Jupiter Notebook written in Python on a PC. The Python script informed the motor controller to rotate the turn table to deflect the rigid airfoil to a desired magnitude and direction. The change in airflow in the wake of the rigid airfoil was detected by the six pressure taps on the MFC morphing wing. The pressures were measured nd compared to a static pressure measured in front of the experimental setup using six differential pressure transducers. Signals from these pressure transducers were acquired by the NI-DAQ, and provided to the Python script. The Python script used this information for action selection. The selected action was provided to the NI-DAQ and transformed into an MFC Voltage signal which was then amplified to power the MFC camber morphing trailing edge of the wing. The lift produced by the change in camber was measured by the load cell, and provided to the Python script for reward calculation during controller training or performance measurement during controller testing.

mounted outside the wind tunnel illuminated a two-dimensional sheet of particles in the longitudinal dimensions. Above the wind tunnel, two Imager sCMOS cameras in a stereo configuration captured 50 sets of paired images with 15 microsecond intervals. From this, I captured the velocity profiles in the vertical and streamwise (x and z) directions of the wind frame of reference up stream of and around the morphing wing.

To create learned controllers capable of reacting to the changing environment, I used proximal policy optimization (PPO) to develop policies for the camber morphing wing. The DRL environment included a discrete action space comprised of 7 voltage changes [-6,-2,-1,0,1,2,6], and a continuous state space including normalized pressure signals [-2.5,2.5] and normalized MFC voltage signals [-2,2]. The actor and critic network structures are described in Figure 4.5, including a temporally aware 1D CNN input layer with the ten most recent state measurements for state observation. This layer included convolutions with kernel lengths of three and a stride length of one. The two subsequent hidden layers were structured linearly with 512 nodes each, and rectified linear unit activation functions [98], [101]. I used Adam optimization with a 0.0003 learning rate [136]. As validated by Magar, lift is sufficient for state estimation in a longitudinal pitch-plunge environment [124]. Therefore, I used change in lift as the optimization parameter, using load cell measurements to provide a reward for the learning algorithm. The goal of the learning algorithm was to develop a controller that minimized the change in lift experienced during a gust. Although lift measurements were used for the reward structure during training, the controllers did not use lift information for action selection. The learned policies only used pressure and MFC voltage signals for action selection. During testing, the load cell provided information to judge controller performance.

A Python script in Jupyter Notebooks orchestrated controller training and testing. Figure 4.4 presents the data flow structure for the experiment. Due to electromagnetic

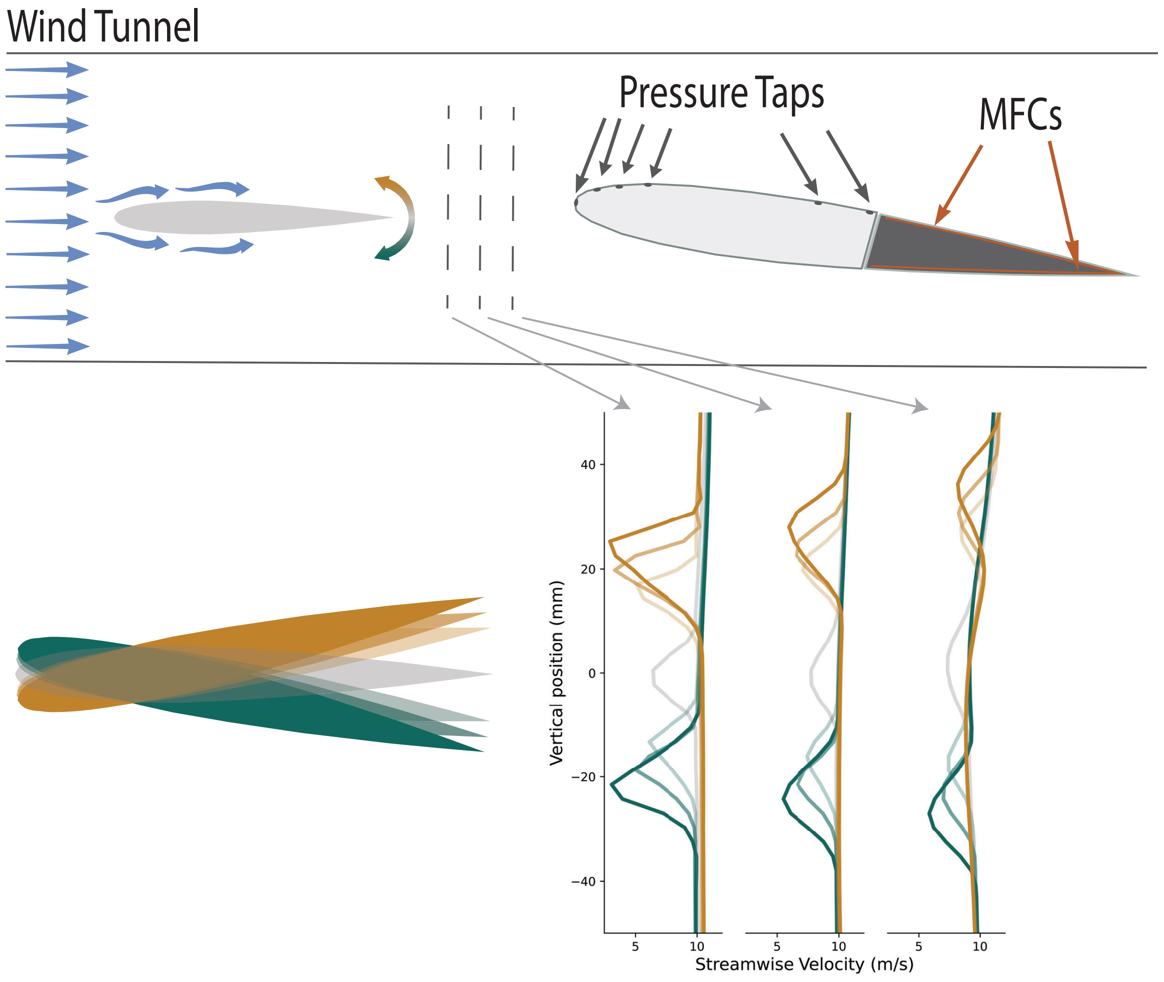

Figure 4.4: Illustration and characterization of the variable gusting wind tunnel environment. The rigid airfoil is mounted upstream of the MFC morphing wing. The MFC morphing wing includes six pressure taps to sense gusts. The rigid airfoil deflects upwards (yellow) and downwards (green) at varying degrees (depicted by opacity) to create a variety of velocity wakes (as measured using particle image velocimetry) to which the morphing wing must react to maintain lift.

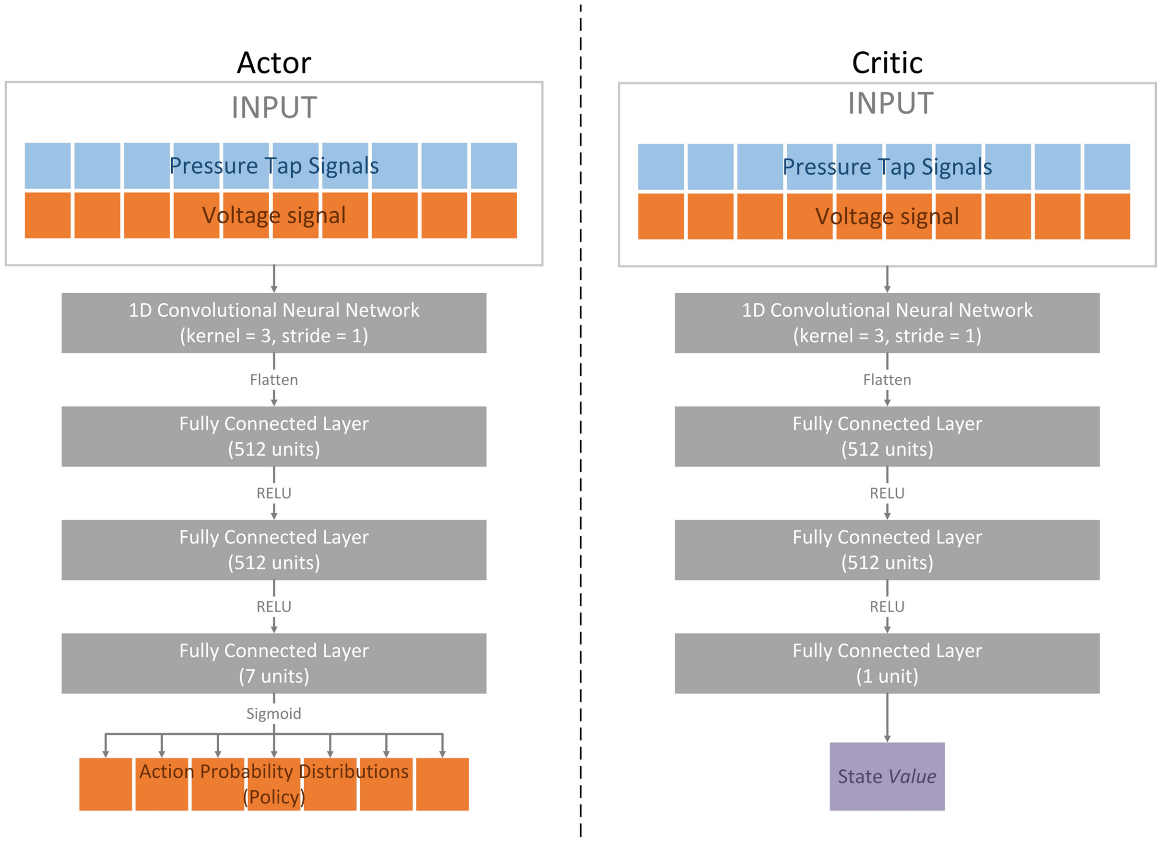
Actor
INPUT
Pressure Tap Signals
Voltage signal
1D Convolutional Neural Network (kernel = 3, stride = 1)
Flatten
Fully Connected Layer (512 units)
RELU
Fully Connected Layer (512 units)
RELU
Fully Connected Layer (7 units)
Sigmoid
Action Probability Distributions (Policy)
Critic
INPUT
Pressure Tap Signals
Voltage signal
1D Convolutional Neural Network (kernel = 3, stride = 1)
Flatten
Fully Connected Layer (512 units)
RELU
Fully Connected Layer (512 units)
RELU
Fully Connected Layer (1 unit)
State *Value*

Figure 4.5: The neural network structure for the actor and critic models in the PPO algorithm. Each network has the same base structure, including a 1D CNN layer followed by three fully connected layers with ReLU actuation functions. The input for each network includes the ten most recent voltage signals supplied to the MFCs, and the ten most recent pressure tap signals, containing either 1,3, or 6 measurements for each time step depending on the pressure tap configuration.

interference, the load-cell and pressure sensors were unable to provide accurate signals during step-motor operation. For this reason, I limited the experiments to a discrete square gust column environment, also known as a sharp-edge gust [117], [137], [138]. During training and testing, the script paused policy updates and data collection during gust wing rotation, then resumed training and testing when the rotating wing achieved the desired deflection. For this work, I defined a gust as a change in effective wind velocity, including speed and direction. This is analogous to a discrete updraft or downdraft, often used as a simplified model for gusts in the environment [21], [117], [137]–[139]. I chose to avoid the highly variable effects that occur in the wake of a stalling airfoil. Thus, I limited the gust generating wing to deflections between positive and negative 13° to prevent stall from occurring.

Training consisted of 1000 episodes, each episode consisting of 200 timesteps of 0.05 seconds each. Each episode began by rotating the gust generating airfoil, alternating between beginning the new episode at zero degrees and a random deflection within the range $\pm\,[7°, 13°]$. At zero gust deflection, the MFC actuators began without camber morphing in either direction. From this position the pressure taps provided a baseline signal for comparative pressure observations throughout the episode. After initialization, the policy was activated and the action selection and learning updates began. The initialized pressure and goal lift values were held for the following episode after the discrete square gust operation. The discrete gust was performed by deflecting the gust generating airfoil to a randomized position where it was held for the length of an episode, 200 timesteps, simulating an extended 10 second gust. The end of the gust operation signaled the end of the training episode, which returned the gust airfoil to zero degrees and the morphing wing MFCs to a neutral deflection position to begin a new episode and initialization. I used this procedure to train ten controllers for each of three different pressure tap configurations: using all six pressure taps, the front three pressure taps, and a single pressure tap on the leading edge of the morphing wing. I selected these pressure tap configurations based on the pressure distribution expected for the top surface of a symmetric airfoil and the sensitivity of the respective tap locations [135]. In all, this approach resulted in 30 controllers.

4.3.3 Testing

I performed testing in a similar manner for the six different gust conditions $(-12.5°, -10°, -7.5°, 7.5°, 10°, 12.5°)$ where positive values represented upward gust deflections and negative values represented downward gusts. Each testing episode began with an initialization period to reset the expected pressure tap signals during typical flow. After initialization, the tested controller began action selection. After 100 timesteps of neutral airflow, the gust generating airfoil deflected to a specified gust condition and held that position for 200 timesteps. Finally, the gust generating airfoil returned to a deflection of zero, concluding the discrete gust (Fig 4.6a). I tested each controller for each gust condition ten times. Additionally, I performed testing on the wing when unactuated to create a baseline for comparative controller performance.

I measured performance using a comparative lift error (CLE) experienced by the active camber morphing wing (Fig 4.6b). The lift error is a measurement of the change in lift that occurred during a discrete gust. The comparative lift error was the ratio between the lift error experienced by the wing when actuated by a given controller, LE_C, and the average of the lift error that occurred for an unactuated wing as a baseline, LE_B,

$$CLE(t) = \frac{|LE_C(t)|}{|\frac{1}{T_g}\sum_{t=0}^{T_g} LE_B(t)|}, \tag{4.1}$$

Where t is an individual timestep and T_g is the number of timesteps during the gust. Figure 4.6a illustrates the average lift error experienced by the morphing wing in actuated (solid line) and unactuated (dotted line) flight when subject to the largest magnitude gust conditions in the upward and downward directions.

Due to the black-box nature of neural networks, and the policies developed using such methods, I accounted for stability and robustness of control through repetition. I repeated gust alleviation performance tests ten (10) times for each combination of pressure tap configurations (3), trained controller (10), and gust condition (6). In all, this approach amounted to 1800 gust rejection tests in total. Figure 4.6 illustrates consistency in performance between tests, gust conditions, and training iterations when the other variables are held constant for the six-tap configuration. I calculated settled CLE (SCLE) for each gust response test by averaging the last 100 timesteps of lift error and comparing that to the average baseline lift error,

$$SCLE = \frac{|\frac{1}{100}\sum_{t=0}^{T_g} LE_C(t)|}{|\frac{1}{T_g}\sum_{t=0}^{T_g} LE_B(t)|}. \tag{4.2}$$

For a single training iteration and gust condition, the average standard deviation of $SCLE$ between individual tests was 0.039 (Fig 4.6c). When comparing $SCLE$ between average test performances for a single controller under each gust condition, performances typically varied with a standard deviation of 0.105 (Fig 4.6d). The average performances achieved by each training iteration for a single gust condition produced standard deviations of 0.082 (Fig 4.6e). Since the individual and average comparisons achieved standard deviations below 10%, I determined that the average performances achieved by an averaged controller provided a fair representation of the CLE for each gust condition (Fig 4.6b). However, the highest standard deviation was experienced when comparing performances between gust conditions (0.105) and was considered further in the discussion section (4.5 A comprehensive collection of the raw CLE test data, as shown in Figure 4.6c, for all gust conditions and pressure tap configurations is provided in Appendix B.

I compared the performance achieved by each of three pressure tap configurations, including all six pressure taps, the front three pressure taps, and only one pressure tap

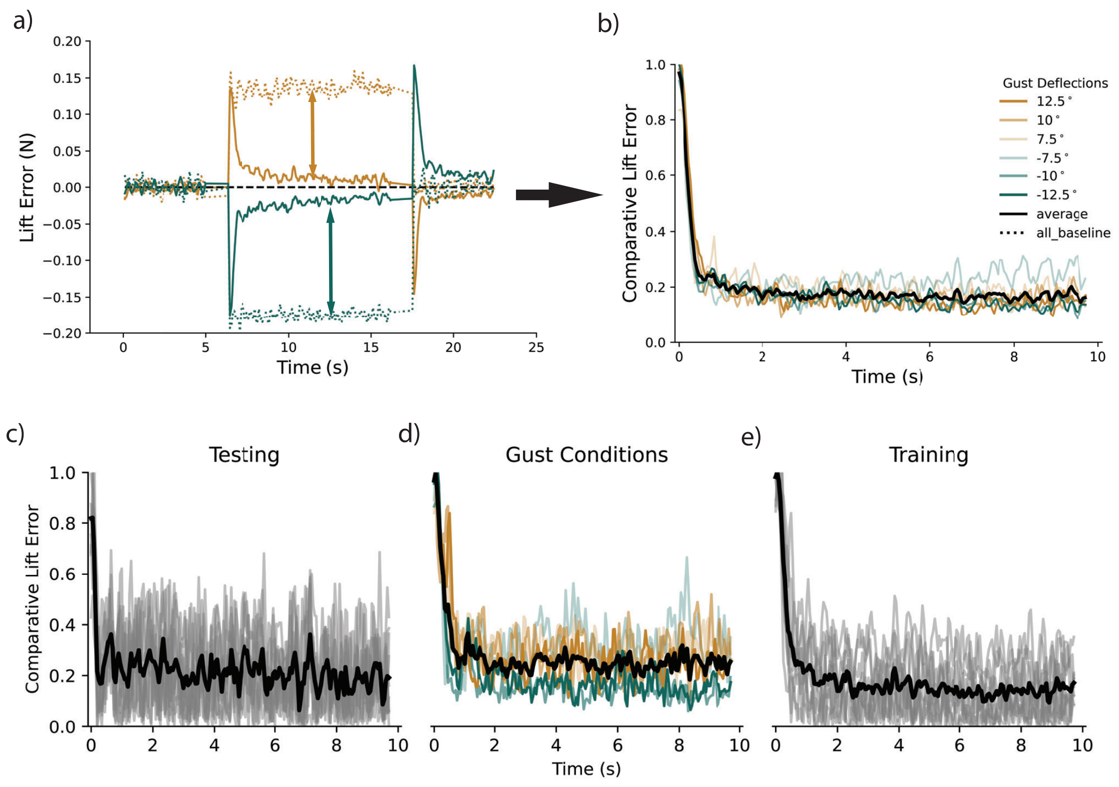

Figure 4.6: Trained controller comparative lift measured during gusts. I quantified the performance of the trained controllers by comparing the change in lift (Lift Error) experienced by the morphing wing during a gust between the actively controlled wing and inactive baseline, (a), as a Comparative Lift Error, (b). To account for controller robustness, I repeated ten (10) tests for each trained controller (10) in each gust condition (6). I compared all individual tests for one controller at a single gust condition to measure consistency between tests (c). I compared average performance for a single controller all for each gust deflection to quantify consistency between gust conditions (d). I compared average performance between individual trained controllers at a single gust condition to measure consistency in training (e).

located on the leading edge of the wing. Figure 4.7a illustrates one such comparison for a single gust condition where the black dashed lines represent the average $SCLE$ values. These values were used to determine to overall gust reduction percentage (GRP),

$$GRP = |1 - SCLE| \times 100\%, \tag{4.3}$$

for each test. For fairest comparison and representation of data, I calculated the GRP values for each individual test, providing distributions of n=100 GRP values for each combination of gust condition and pressure tap configuration. Due to the maximum bounded nature of this metric, many of distributions are skewed to varying degrees with an average skewness of -0.7 between distributions. For this reason, I used the mean as a conservative estimate of central tendency for the primary performance metrics. Additionally, I used a linear mixed effects model to determine the relationship between GRP and the number of pressure taps while considering the potential for random effects between individual tests for each trained controller. This allowed me to comment on the significance when comparing performances between different pressure tap configurations.

Finally, I measured the speed of the controllers using rise time, measured as the time needed for the learned controllers to adjust the CLE from 10% to 90% of the total achieved gust reduction (Fig. 4.8a). Rise times were also measured for each test. These distributions were highly skewed, with an average skewness of 1.7. For this reason, I used median and inter-quartile range (IQR) to represent the central tendency and spread for the controller speed metrics.

4.4 Results

GRP generally improved as the number of pressure taps used for state observation increased (Fig 4.7a). The simplest configuration, with only one pressure tap, achieved an average *GRP* of 76%. Including an additional two pressure taps significantly improved performance ($p = .009$), reducing the effect of gusts by 82% on average. Using six pressure taps, the learned controllers achieved a *GRP* of 84% on average, significantly outperforming the single-tap configuration ($p < .001$), but the improvement over the three-tap configuration was not significant ($p = .25$).

The trained controllers struggled to reject the $-7.5°$ gust deflection when compared to the other five gust conditions. This was particularly noticeable for the three and six pressure tap configurations, where this condition had a significantly reduced *GRP*. When not considering the $-7.5°$ gust condition, the range of *GRP* achieved by the three-tap and six-tap gust conditions were 82% to 85% and 84% to 87% respectively, where the bottom of each range was at least as high as their average overall performance. I explored the reason behind this reduced performance in detail in the discussion section (Sec. 4.5.2).

Performance consistency is important if this approach is to provide safe and reliable flight control for a future UAV. To quantify consistency, I calculated the standard deviation of the *GRP* distributions (Fig. 4.7b). Although standard deviation provided a good metric for general consistency, it is only a single value for each distribution and cannot provide information regarding significance when comparing the consistency metric. In addition to standard deviation, I measured the Euclidean distance from the mean for each *GRP* test value, calculated as an absolute difference between a single *GRP* value and the mean *GRP* of the distribution to which the single value belonged. From these new sets of data representing the spreads of the *GRP* distributions, I used another linear mixed effect model to determine the significance of consistency comparisons between pressure tap configurations. The single pressure

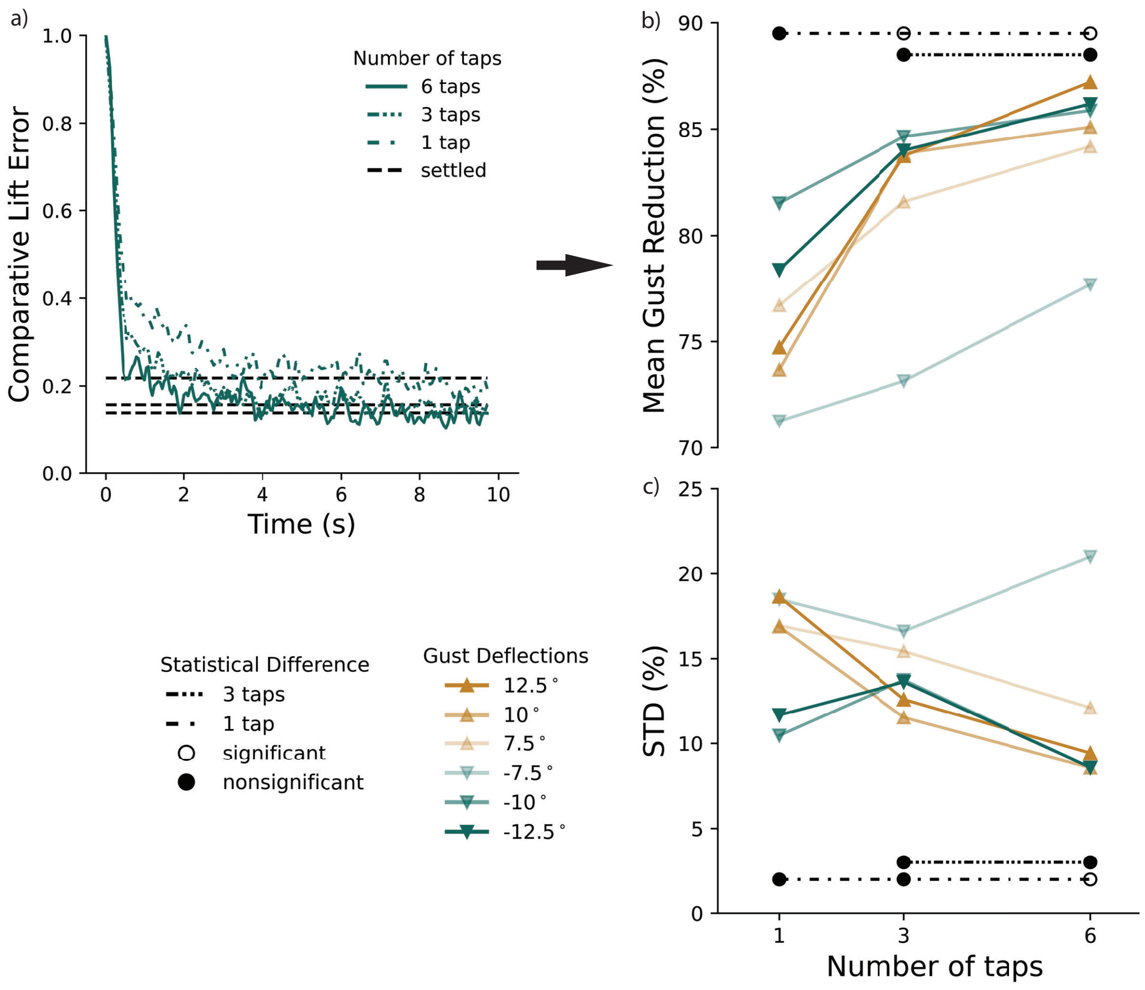

Figure 4.7: Gust reduction performance. I calculated the mean Gust Reduction Percentage (GRP) from the Settled Comparative Lift Error (SCLE) achieved in each gust response test (a). Increasing the number of pressure taps improved mean GRP, where SCLE was represented by the dashed black lines (b). Significance in improvement was represented by dashed and dotted lines where comparison to the one-tap configuration was represented by dashed lines alternating with a single dot, and comparisons to the three-tap configuration was represented by dashed lines alternating with three dots. Improvement was significant when increasing the number of pressure taps from one to three or six ($p = .009$ and $p < .001$ respectively) as represented by an open circle (b). Controller consistency was measured by standard deviation (std) where the only significant improvement was between using one pressure tap and six pressure taps ($p = .04$) (c).

tap configuration produced GRP distributions with an average standard deviation of 15%. The three-tap configuration improved consistency over the single tap with an average standard deviation (std) of 14% but was not statistically significant (GRP std = 14%; $p = .19$). The six-tap configuration improved consistency significantly over the one-tap configuration (GRP std = 12%; $p = .04$), but was not statistically different from the three-tap case ($p = .42$). Again, the $-7.5°$ gust deflection achieved the worst performance of all gust conditions, demonstrated by a higher standard deviation. This was apparent specifically for the three and six pressure tap configurations. When not considering the $-7.5°$ gust, the standard deviation ranges become 12% to 15% for the three-tap configuration and 8.6% to 12% for the six-tap configuration.

Similar to the gust rejection and standard deviation, I found that increasing the number of pressure taps improved the controller speed, quantified by the median rise time. The median rise times ranged from 0.30 seconds to 0.45 seconds, 0.33 seconds to 0.50 seconds, and 0.35 seconds to 0.62 seconds for the six-tap, three-tap, and one-tap configurations respectively (Fig. 4.8b). The three-tap and six-tap configurations both outperformed the one-tap configuration significantly ($p < .001$), whereas the difference in controller speed between the three-tap and six-tap configurations was not significant ($p = .09$). Additionally, increasing the number of pressure taps from one to three or six significantly improved consistency in controller speed ($p < .001$), and the difference between three pressure taps and six pressure taps was not significant ($p = .07$) (Fig. 4.8c).

4.5 Discussion

4.5.1 Strengths and Impact

The strength of this work is from the extensive testing performed to ensure trustworthy results. I developed ten (10) controllers for each pressure tap configuration

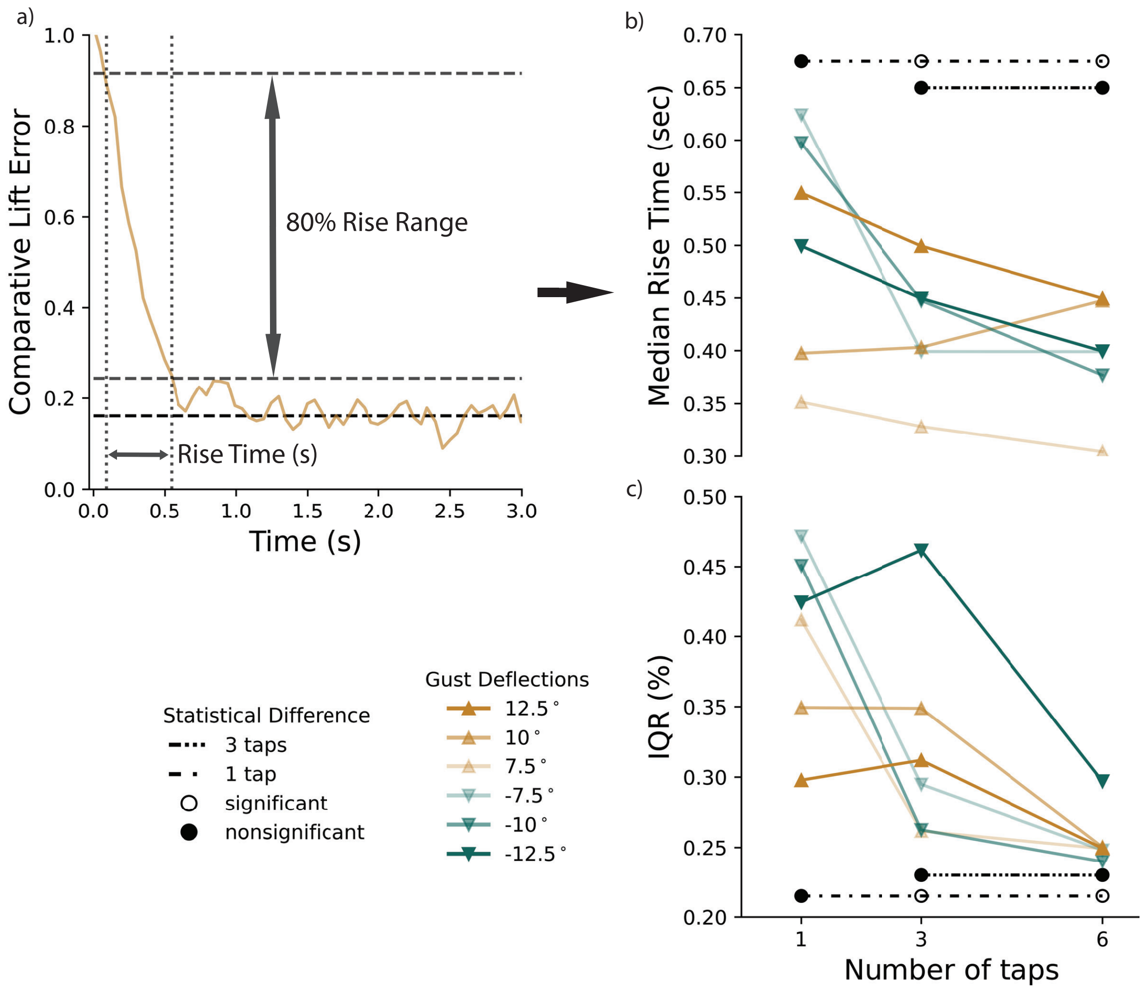

Figure 4.8: Trained controller speed. I calculated the rise time as the time needed to reduce the gust impact from 10% to 90% of the total reduction (a). This metric is used to represent the speed of the controller. Due to the highly skewed nature of these results, 1.7 average skewness, I presented the median rise time to illustrate central tendency for the speed metric. Increasing the number of pressure taps above one significantly improved rise time (p < .001) (b). Consistency of in speed was also significantly improved by increasing the number of pressure taps to three or six (p < .001) (c). However, in both cases, the difference between using three pressure taps, and six pressure taps was not significant (p = .09 and p = .07).

(3), and repeating testing ten (10) times at each gust condition (6) for controller. Therefore, I accumulated 1800 test results to verify the robustness of the findings and establish confidence in the conclusions.

I combined two novel approaches, multifunctional material camber morphing and DRL, to develop an effective gust alleviation system. The trained controllers used minimal sensor requirements and no dynamics or state inference modeling to achieve comparable, and often superior, performances to methods requiring significant modeling, prediction, and controller design. This is particularly impactful for smaller UAV design with the added weight, volume, and computational constraints. Additionally, although much of the work in the individual fields of gust alleviation and DRL is performed in simulation, I completed this work entirely in a physical hardware environment.

By using wing morphing for gust alleviation, the reaction reduces the change lift experienced by the wings during a gust, which mitigates associated perturbations to the aircraft's orientation and speed. The active control provided by the morphing wing allows UAVs to benefit from similar gust alleviation as birds that change their wing shape to potentially mitigate deviation from their flight path through a gust. Additionally, using MFCs to perform camber morphing improves aerodynamic efficiency and reduces the structural mass, an important variable for overall aircraft efficiency. GLA to mitigate critical loads experienced during gusts relaxes structural requirements, further reducing the overall weight of the aircraft [14].

Through DRL I developed policies that effectively controlled the MFC camber morphing wing to mitigate gust loads directly from pressure sensor signals without additional modeling or state inference. Traditional control methods, adaptive control, and MPC rely heavily on dynamics and state inference models. Achieving high accuracy comes with computational costs that grow with environment complexity. Modeling turbulence and MFC morphing mechanics requires model simplification, leading to suboptimal behavior due to approximation inaccuracies. A DRL controller reduces operational computational complexity by developing a policy to map pressure signals directly to actions, requiring no additional model, achieving *fly-by-feel*. Additionally, I found that the difference in performance between using three pressure taps and six pressure taps was not significant. Therefore, future designs do not need to incorporate the last three pressure taps. This reduces mechanical complexity and costs associated with the manufacturing process. Additionally, minimizing the number of input signals required for action selection further reduces the computational complexity of the neural controller. This reduced computational complexity makes this control method ideal for small UAVs. Operating continuously beneath the high-level navigation controllers, this will improve small UAV surveillance, reconnaissance, package delivery, and target tracking missions in erratic environments, such as cities.

4.5.2 Reduced GRP at $-7.5°$ Gust

On average, the trained controller performance degraded when reacting to the $-7.5°$ gust condition. Since GRP was measured as a comparative percentage of reduced gust, it depended on the average baseline lift error, LE_B (Eq. 2-3), which for the lower magnitude gusts was a smaller value. Therefore, the actual lift error, LE_C, was not necessarily any higher than that experienced at the other gust conditions (Fig 4.9a). The average lift error LE_C for the $-7.5°$ gust was well within the range of LE_C experienced at the other gust configurations. Although this is true, by this logic I should see similar GRP for the $7.5°$ gust deflection, but I found the performance was not as severely reduced as in the $-7.5°$ gust (Fig 4.7).

Although the LE_C for the $-7.5°$ gust is similar to that experienced at the other gust conditions, there is a greater change in LE_B between the smallest and largest magnitude gusts in the downward direction than there is for the upward direction. This is partly because the downward gusts generally showed a greater impact on

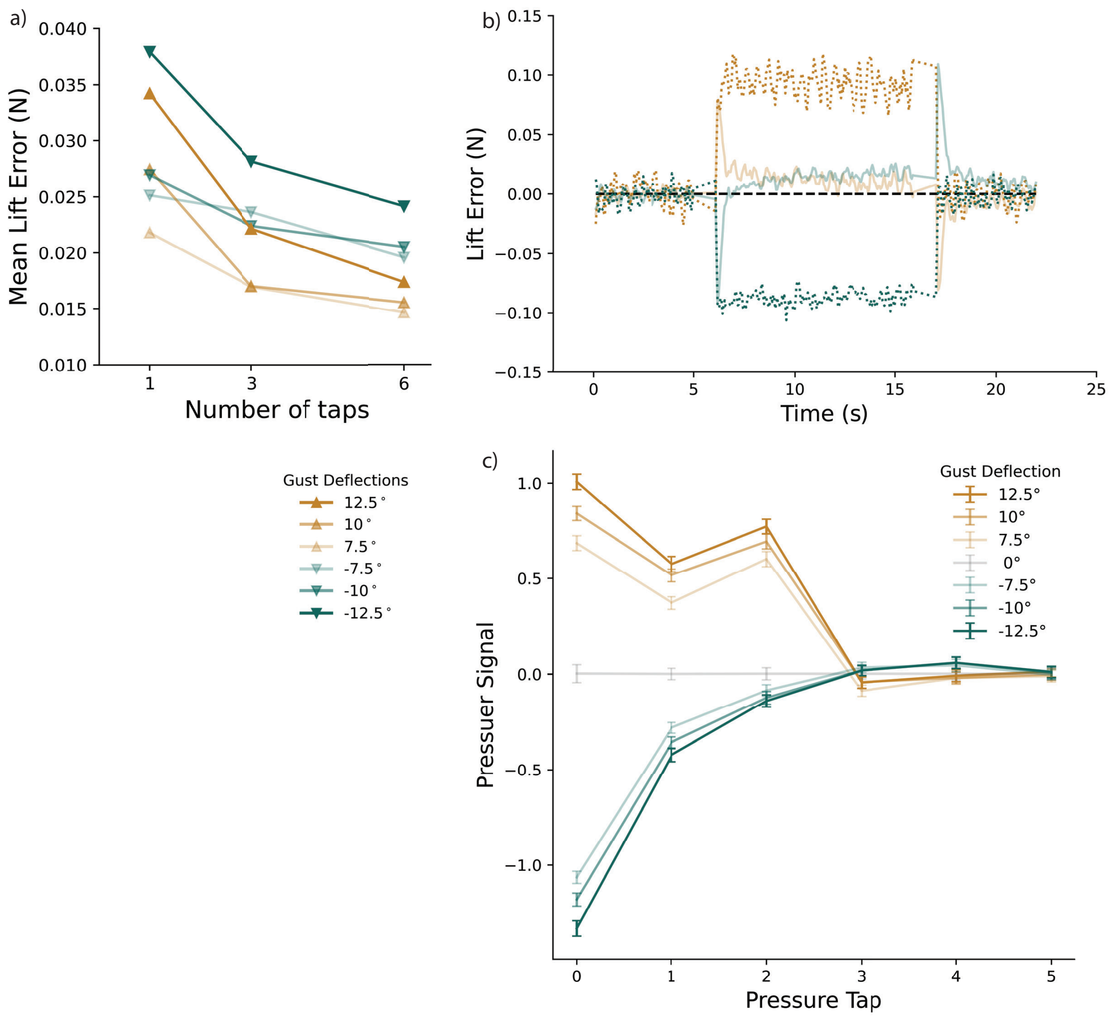

Figure 4.9: Considerations for low performance at $-7.5°$ gust condition. The trained controllers achieved mean lift error at the $-7.5°$ gust deflection similar that achieved at the other gust conditions (a). At the $-7.5°$ gust condition, the trained gust alleviation controllers using six pressure taps overshot zero lift error (b). Although the first three pressure taps produce sensitive pressure signals for the upward gust deflections, the third pressure tap is much less sensitive to downward gusts (16.7%) (c).

LE_B than the upward gusts. I looked at the change in lift error between the upward and downward gusts as percentages of the lift error at maximum gust magnitude in that direction. For the upward direction, the $10°$ and $7.5°$ gust conditions generated average baseline lift errors that were 78% and 69% of the average baseline lift for $12.5°$ respectively. Whereas for the downward direction, the $-10°$ and $-7.5°$ gust conditions generated average baseline lift errors that were 83% and 50% of the average baseline lift for $-12.5°$ respectively. This greater change in gust impact between deflection magnitudes could contribute to some of the controllers overshooting zero lift error for the $-7.5°$ gust condition (Fig. 4.9b). The overshoot experienced at the smallest downward gust suggests that some of the controllers expected the gust to produce a larger impact on the change in lift, as opposed to the $7.5°$ gust condition that does not overshoot zero.

I found that the controllers were limited in their ability to differentiate between downward gusts due to a reduced sensitivity in pressure readings for downward gusts (Fig 4.9c). I analyzed the sensitivity of each pressure tap for all tested gust conditions. For the tested gust conditions, the front three taps experienced greater change in pressure signal than the three rear pressure taps. The leading-edge pressure tap showed the greatest sensitivity for both positive and negative gust deflections. The second pressure tap showed less sensitivity for the downward gust than the upward gusts (27% reduction). The third tap, however, showed a steep reduction in sensitivity when experiencing a downward gust compared to an upward gust (83% reduction). This reduced sensitivity suggests some controllers that rely on the third pressure tap for state observation could have struggled to differentiate between the different magnitudes of downward gusts.

To determine why the system experienced reduced sensitivity for downward gusts, I performed particle image velocimetry (PIV) to measure the velocity changes across the top surface of the morphing wing at each tested gust condition compared to the baseline neutral gust condition (Fig 4.10). In the top plot I noticed that even for the 7.5° gust condition, there is a notable increase in velocity over the first three pressure taps. For the $-7.5°$ gust condition, I saw a reduced velocity at the leading edge of the wing. However, near the third pressure tap there was a thin line of white, representing a location where the change in velocity shifted from negative to positive. This resulted in a very minor change in velocity at that point. For the $-12.5°$ gust deflection, there was a larger reduction of velocity at the leading edge of the wing, illustrated by the darker blue. However, the velocity still transitioned from negative to positive near surface of the wing where the third pressure tap was located.

Between the greater change in lift between downward gust conditions and the location of the third pressure tap leading to a reduced sensitivity for downward gusts, the learned controllers struggled to reduce the $-7.5°$ gust condition as consistently as at the other gust conditions. However, it is important to note that the trained controllers still achieved average GRP values above 73% for the three-tap and six-tap configurations. Additionally, the controllers reduced the small downward gusts to average LE_C values like those achieved for the other gust conditions. Finally, I expect that this issue would be mitigated by including sensors on the bottom surface of the wing at positions mirroring the second and third pressure tap on the top surface.

4.5.3 Limitations

There are key assumptions made to facilitate this study that impose limitations on the extrapolation of the results. First, the stepper-motor that drove the gust generating airfoil produced electromagnetic signals that rendered the pressure sensors unreadable during gust-airfoil rotation. Therefore, I was unable to perform pressure or lift measurements during the growth and decay periods of each gust. For this reason, the presence of the gust is visible as a square wave in lift error as opposed to the traditional simplification of a 1-cosine model (Fig 4.6a). Sharp edged gust

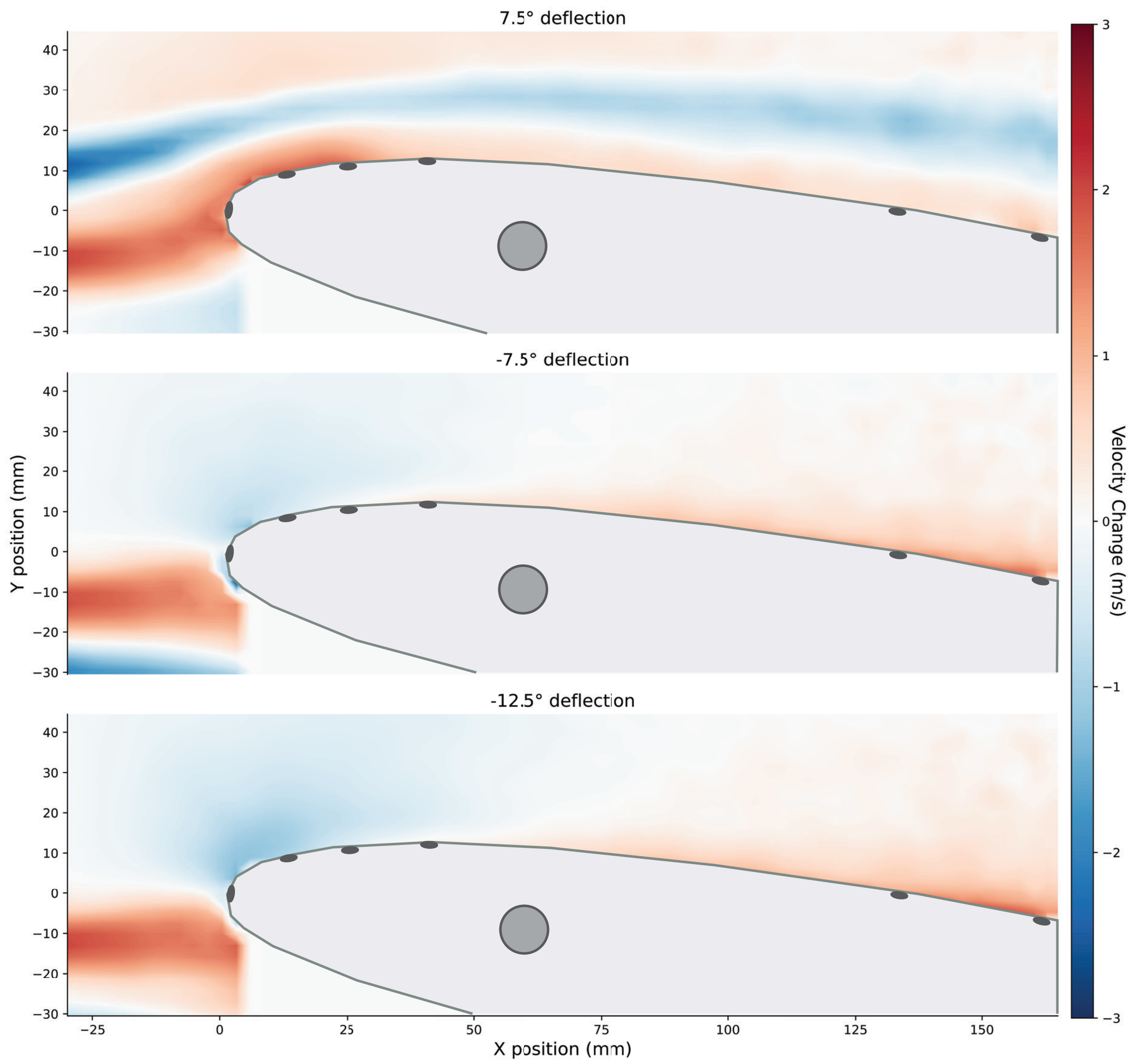

Figure 4.10: Particle image velocimetry (PIV) shows the environmental changes experienced by the wing during different gusts. The change in velocity is stronger over the front three pressure taps in the upward gust than in the downward gusts. This is most noticeable at the third pressure tap location.

models were first used in early gust alleviation research, and are still used today to represent gust columns like those experienced when encountering an updraft between two buildings [21], [117], [137], [138]. Further, this provided an additional challenge for the learned gust alleviation controllers: reacting to a gust without initial ramp up or warning. Instead, the controller began operation after the deflecting airfoil was at full magnitude, delaying the controller's response until the gust was already at its peak. Thus, the rise times presented in this work provided controller speed information when subjected to this limitation. Therefore, I focus the discussion on total reduction of gust influence on the wing, using the $SCLE$ and GRP measurements as opposed to rise time and other controller speed metrics. With that being said, the trained controllers achieved rise times of less than 0.5 seconds. Between the square nature of the gust environment and this rapid response, I expect future controllers, trained in a continuous environment, to be effective for continuous gust alleviation.

I performed all training and testing in this work at one flow speed, 10 m/s. Achieving control at other flow speeds will require additional training at a range of speeds encompassing the flight envelope for the small UAV operation. An additional state input measured by an upstream pitot tube should be included to differentiate interpretations of on-board pressure taps at different speeds.

Finally, this work is dedicated to mitigating gusts represented as a change in experienced airflow velocity, including magnitude and direction. I did not consider gust control for the highly variable flow that occurs behind a stalling wing. For this reason, the tested gusts were limited to magnitudes between deflection angles of $-12.5°$ and $12.5°$.

4.6 Conclusions and Future Work

In this work I used DRL to produce gust alleviation controllers for a multifunctional MFC spanwise camber morphing wing. These controllers achieved *fly-by-feel*, capable of making action decisions directly from pressure sensor signals. The GLA controllers reduced gust impact on lift by 71% to 87% for state observation configurations including six pressure taps, three pressure taps, and only one pressure tap. The three-tap and six-tap sensor configurations significantly outperformed the one-tap configuration in GRP and speed, but the difference between the three-tap and six-tap configuration was not significant. These results inform autonomous GLA design for future intelligent morphing aircraft.

Future work will consider more complex gusting environments. I will incorporate continuous gusting environments, including single 1-cosine gusts, continuous frequency-based gusts, and continuous randomized perturbations. From these gusting environments we can establish and mitigate aeroelastic and vibrational effects due to gusts. This will also allow the exploration of more chaotic and variable environments. The *fly-by-feel* nature of this work creates interesting opportunities for localized gust alleviation. For larger aircraft, different sections of the wings may experience varying gust levels. For this reason, it may be beneficial to incorporate multi-agent DRL to develop several controllers dedicated to alleviating gusts on individualized sections of the aircraft's lifting surfaces.

Additionally, thus far this work focused primarily on reducing the impact of a gust on the lift produced by the morphing wing. Although lift is considered sufficient for state measurement in gust alleviation, DRL allows the optimization of control variables that are more complex, and directly applicable, through reward design [124]. By incorporating dynamics models with the physical morphing system, we can develop controllers capable of directly minimizing change in pitch, plunge, or vibrations. This direction of research can include the training controllers for more high-level maneuvers, such as perching, as well. To maintain the *fly-by-feel* design, dynamics models should only be used for controller development, and unnecessary for controller operation since decisions would still be made directly from on-board sensor signals.

Finally, to push toward learning on-the-fly, it is beneficial to incorporate embedded systems and neuromorphic technologies on the morphing systems [109]. For this reason, it is important to characterize the material and performance constraints placed on embedded micro-computers and learning hardware when installed within an active morphing structure. From there we can begin to incorporate learning algorithms during flight, so that the intelligent UAV can learn to adapt to changing environments as it experiences them.

CHAPTER V

Conclusion

5.1 Key Results and Contributions

Frequently changing dynamic environments continue to provide a challenge for UAV design. Introducing intelligence to the system can allow UAVs to perform in various environments as well as react to perturbations. Although there is substantial work in the community focusing on morphing structures, multifuncitonal materials, and autonomous learned control, little work has been done to tie all three of these fields together. This dissertation aims to forge that connection between fields to create intelligent, autonomous, multifunctional morphing UAVs. Considering each problem from the angle of learned autonomous control provides an adaptive solution that may be applied to a variety of questions surrounding morphing UAV design. Each subsequent chapter of this dissertation provided a control problem of increased complexity, each solved through RL and applied on the physical system. This ensues confidence in the capability of RL to achieve desired control objectives so long as a repeatable environment can be produced, and a reward can be structured to represent the desired goal.

5.1.1 RL problem design allows autonomous controller development

In Chapter II I designed an RL training scheme that allowed training to be completed autonomously, without any human interaction, in a physical hardware environment.The environment was an airsled-airtrack experiment designed as a 1-dimensional analogy for an aircraft maintaining trim after experiencing an external perturbation. Additionally, I found that incorporating additional policy updates between traditional training episodes improved learning speed and consistency [65].

The pseudo-episodic training format I developed provided an opportunity to train a physical hardware environment in a manner similar to episodic methods traditionally seen in simulated environments. One of the greatest benefits of episodic training is the added exploration offered by beginning each new episode in a random state. Achieving a similar benefit in physical hardware often requires human intervention, external mechanical reset devices, or an additional controller operating specifically for resetting each episode. In each of these cases, the autonomy of the trained agent is compromised resulting in additional effort and careful observation by an external entity, limiting the benefits of using RL in the physical environment. The pseudo-episodic training format achieved episodic resets by following a randomised policy between traditional training episodes. Following the random policy improved exploration, similar to that seen in epsilon-greedy exploration techniques, while also providing time to incorporate additional policy updates.

I used methods such as Off-PAC and experience replay to perform additional policy updates during the exploration episodes used for autonomous episode resetting for three actor-critic RL algorithms: A2C, A2C(λ), and PPO. My results showed that each algorithm-exploration update combination was able to successfully return the airsled from a perturbed state to the trim position after only 25 minutes of training on the physical environment in real time. The inclusion of Off-PAC during exploration improved training consistency while including experience replay improved training speed and overall training performance. The best combination for training performance and consistency was using the PPO algorithm with experience replay updates during exploration.

Although training was fast and consistent, the resulting controllers lacked some accuracy. In some cases learning quickly is a priority to achieve safe operation in a changing environment; however, achieving optimum performance would likely require additional training, and future work for more complex environments should be willing to sacrifice training speed for superior performance. The work presented in this chapter is a version of a published research [65].

5.1.2 DRL accounts for hysteretic MFC behavior for accurate morphing control

In Chapter III I developed the first DRL controller for an MFC camber morphing airfoil, achieving faster and more accurate step response tracking than a traditional PD method [67].

For this work I used an LSTM network structure to develop a time-series forecasting model for the dynamics of the morphing airfoil tip deflection. This model simulated the morphing environment for training two controllers using PPO. The second trained controller incorporated an additional penalty for overshoot experienced during step response (MO). These controllers were tested on a series of step responses at a variety of airflow speeds and compared to a traditional PD controller. When operating with perfect feedback, the learned controllers achieved greater speed and accuracy, but the PD controller experienced less overshoot.

For more realistic implementation, I used internal flex sensor signals to provide imperfect feedback to the controllers. Two state inference models were trained using a linear model and an LSTM model to compare controller performances when operating with different levels of feedback accuracy. I found that the PD method relied more heavily on feedback accuracy than the learned controllers. When using the LSTM model, the PD controller still outperformed the learned controllers in overshoot. However, when using the linear inference model, the MO controller outperformed the PD controller in all performance metrics. This suggested that the trained controllers learned the behavior of the MFC tip deflection, and intrinsically accounted for the material hysteresis during controller operation, even with limited feedback accuracy.

The successful implementation of DRL for controller development in an MFC morphing environment ensues confidence in using learned control for morphing UAVs. Learning policies that intrinsically account for challenging control behaviors inherent in smart material actuators can reduce computational complexity during controller operation. These learned controllers map system states directly to actions, and therefore do not require a complex dynamical model for action selection. Reduced computational complexity combined with the speed and accuracy of the learned controllers make DRL an attractive method for developing controllers for rapidly reacting to environmental changes, such as gusts. A version of this chapter is published [67].

5.1.3 DRL achieves ***fly-by-feel*** morphing gust alleviation

In Chapter IV I trained 30 gust load alleviation controllers capable of reducing the impact of a variety of vertical gusts directly from pressure sensor signals using DRL.

In this work I constructed a spanwise morphing wing with three active MFC sections, similar to the morphing aileron presented in Chapter III. In the rigid leading edge of the morphing wing I installed a series of pressure taps for detecting changes in the aerial environment. These environments were generated by mounting a rigid airfoil upstream of the morphing wing. By deflecting the rigid wing, I created turbulent gusts similar to updrafts and downdrafts for the morphing wing to experience. After training each controller on a series of randomized gusts using PPO, the morphing wing learned to mitigate the change in lift experienced during such gusts.

I tested the gust alleviation response for six different gust conditions with three different pressure tap configurations. I found that by increasing the number of pressure tap signals from one to three significantly improved controller speed and total gust rejection, but increasing to six pressure taps did not offer significant improvement over using three. For all pressure tap configurations, the learned controllers achieved average gust rejection of 71% to 87%.

Using model-free policy development via DRL for morphing aircraft gust alleviation offers many benefits to the UAV community. Implementing controllers that operate directly from sensor signals, offering *fly-by-feel* capabilities, reduces the computational requirements compared to state inference and model based methods. This makes gust load alleviation systems more implementable on smaller UAVs that typically require computation constrains. Additionally, it increases the scope of possible mission environments to include locations with dynamic wind conditions, such as cities. Finally, this work showed that DRL can overcome challenges inherent to MFC systems so that their multifunctionality, resulting in weight and aerodynamic benefits, may be fully exploited for efficient flight and GLA.

5.2 Future Work

This dissertation presents my work using reinforcement learning to develop autonomous control systems for intelligent morphing UAVs. I began from a simple one dimensional analogy for trim and ended with a much more complex morphing wing trained to reject gusts. From here we can continue to increase complexity and implement the trained controllers in a more complete autopilot system for small UAVs.

Reinforcement learning creates controllers designed specifically to optimize goals defined by our reward function. This allows us to create controllers for highly complex environments and mission objectives with limited computational complexity during operation by mapping state observations directly to actions. Future work will take advantage of this to create controllers dedicated to more than low level lift management. Reward function engineering can take advantage of the spanwise morphing capabilities to reduce wing root moments in addition to root shear forces. Additionally these controllers will be trained in a continuous gusting environment where vibrations and frequency response can be mitigated for structural health. Furthermore, learning could use a continual format as opposed to the traditional episodic scheme.

In addition to continuing research on camber morphing control, future research should consider additional shape changes for gust alleviation. Drawing inspiration from avian fliers, we can use RL for unique control actions such as change in dihedral, sweep, or wing shapes that simulate variations in bird elbow and wrist angle. Additionally, we can look at alternate control surfaces such as the tail.

Thus far my work focused on a small wing section. Future work will incorporate RL on a full UAV. This increase in scale offers new challenges by increasing complexity from added actuators and sensors. This creates interesting opportunities for implementing hierarchical learning systems. We can implement our learned controllers under traditional high level autopilot systems to determine viability of controller cooperation. Additionally we could compare the performance between developing a single controller for the whole system as opposed to several cooperative but individually trained controllers.

In this work we emphasized training directly on physical hardware without modeling the dynamical system. However, future work can incorporate models to simulate UAV flight based on measurements of a physical system within a wind tunnel. Using this combination would allow the physical hardware sensors and actuator response to inform the simulation, and provide an opportunity to build a reward structure from higher level control objectives within the simulated environment. This would inform design decisions regarding sensor placement and action selection for achieving specific maneuvers or operating in dynamic environments. This increase in complexity brings the system closer to achieving complete physical flight.

Finally, the goal of this research is to develop controllers capable of rejecting perturbations in a dynamic UAV environment. For this reason future research should continue to push toward implementation on an unassisted small UAV flying in a gusting environment.

APPENDICES

APPENDIX A

Intelligent Sensing for MFC Airfoil

In Chapter III I consider feedback control for an MFC camber morphing airfoil using piezoelectric flex sensors. Because of their size, weight and flexibility characteristics, these sensors were capable of reporting deflection information without impeding the performance of a camber morphing airfoil. However, they are also subject to hysteresis and creep, producing imperfect state observations that must be modeled to provide accurate feedback for the controller. Although a simple linear inference method is typically used to model this relationship, it is ill-equipped to support the nonlinear and time dependent behavior of such sensors. In contrast, deep learning (DL) is becoming a popular tool for modeling such relationships. This appendix provides the methods used to achieve state inference from these models, and the impact of state inference accuracy on feedback control [66].

Data Collection

Developing an accurate inference model through ML requires sufficient data collection to represent the desired behavior. In this case, the system's variability from previous deflection due to hysteresis and creep was the primary behavior I wished to accurately represent; therefore, data collection had to be designed accordingly. Instead of performing simple voltage sweep through signals of 0 to 100, I performed 10 complete but randomized explorative sweeps in the following manner. Each explorative sweep began at a randomly selected voltage signal. Beginning at this initial voltage, a new voltage was generated by randomly selecting a voltage change value from a signal change space consisting of all even numbers between -30 and 30. This new voltage was held for 5 seconds, 100 timesteps, before returning to the initial voltage and held for an additional 5 seconds. This was repeated until each signal change value had been randomly sampled from the signal change space without replacement. This process was repeated for initial voltages including all even values throughout the voltage signal space for 10 random seeds. Throughout the 10 randomized explorative voltage sweeps, deflection information was gathered by the internal flex sensors as well as an external Keyence LJ-V7300 2D profilometer to provide the true deflection achieved by the system.

Modeling

Initial modeling used traditional supervised learning techniques to train a 3 layered feed forward neural network for inferring the true deflection achieved from applied voltage and flex sensor signal information. To incorporate some temporal information into initial inference modeling, the model inputs included the flex sensor signal and applied voltage at the current timestep, as well as the voltage applied at the previous timestep. Although this method appeared to accurately track the true deflection, after further inspection I found that the inference model heavily weighted the supplied voltages, ignoring the flex sensor signals and failing to account for hysteretic behavior. Since the goal was to develop a model to accurately infer the system deflection from flex sensor signal, regardless of applied voltage, I removed the voltage values from the model input, instead redesigning the training as a short time-series forecasting problem as opposed to single time-step inference. In doing so I used three different neural network structures, each with an input including the flex sensor signal and approximated true deflection for the ten most recent consecutive time-steps. The three tested structures consisted of an MSD, a one-dimensional CNN, and an LSTM recurrent neural network, each built and trained using Tensorflow (Fig A.1). The MSD and CNN networks included ReLU activation functions and dropout layers with dropout rates of 0.2 between hidden layers while the LSTM network relied on a single layer with only the sigmoid and hyperbolic tangent activation functions found within a typical LSTM.

After training each model using the Adam optimizer with a learning rate of 0.0001 and early stopping to prevent overfitting, I found the LSTM model achieved the greatest accuracy in training according to mean squared error in validation and testing (Fig A.2a). I trained two models using each network structure with two different methods for introducing the data to the model. In the first method, I simply combined all the training data from the eight training data sets, and then trained the model straight through. For the second method of training, I let the eight training sets remain separate (split) and then completed training the model on each set of data individually before continuing training on the next set. The two models were combined into an ensemble to improve robustness during inference. An additional performance comparison included implementing each model on a sequence of flex sensor and deflection data not used during training. Figure A.2 illustrates the performances of each neural network structures when given the same true initial position information and subsequent flex sensor signals over 100 seconds. It is visible that the LSTM inferred observations most closely follow the true deflection shown by the dashed black line.

In the literature, a simple linear model was used to map flex sensor signals to trailing edge deflection values [5]. Therefore, I developed a model in Tensorflow

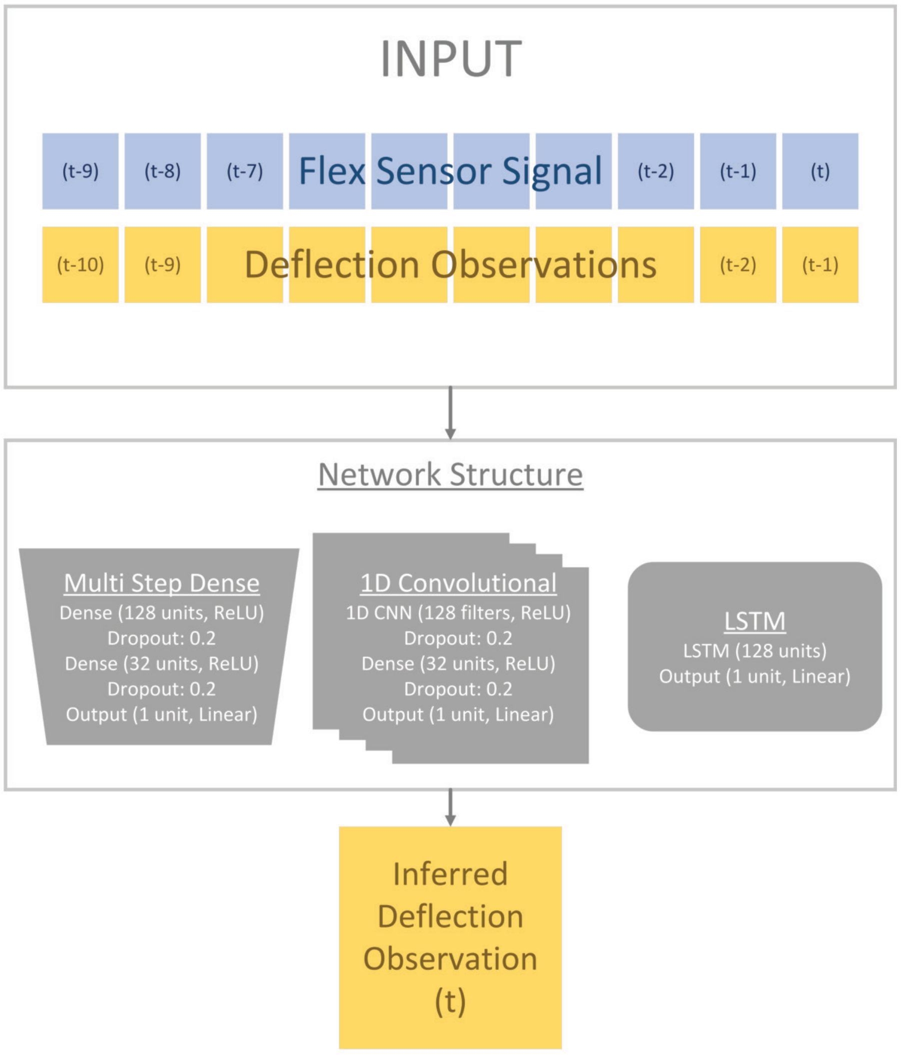

Figure A.1: Neural network structures used for state inference. The input data structure for state observation inference with a time-series forcasting format uses the ten most recent flex sensor signals as well as the ten previously inferred deflection observations to estimate the current deflection observation. I tested the performance of this style for three different network structures: Multi Step Dense (MSD), 1D-Convolutional (CNN), and Long Short-Term Memory (LSTM)

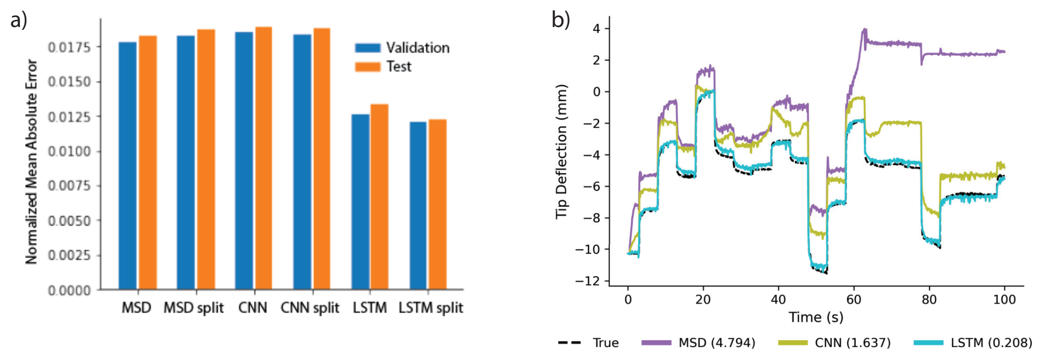

Figure A.2: State inference training and testing performance. LSTM achieved greatest accuracy for training on both complete and split data in validation and testing (a). When tested on a 100 second time-series of flex sensor signal data, LSTM inferred observations that followed the True Deflections with the greatest accuracy, with the mean absolute error in terms of millimeters shown in parenthesis (b)

using a single node without an additional nonlinear activation function for a linear comparison.

Results

When comparing the impact of observation accuracy on controller performance, it was useful to compare the inferred deflections with the true deflections for both inference models. When given true observations, the PD controller performed well, accurately tracking the desired deflection signal. When the controller made decisions from states observed by the flex sensor models, although the observed deflection (represented by the blue line) still tracked the target deflection signal well, there was a discrepancy between true and observed deflection states. This led the true deflection (represented by the green line) to lack accuracy in tracking. With that being said, the more complex observation model, LTSM, provided more accurate results, particularly for the intermediate steps, as indicated by the reduced gap between the true and observed deflection values.

Additionally, since the morphing system is meant for flight, it was important to observe the performance of the controller and sensor inference methods while under aerodynamic loading. This was particularly necessary since all sensor data used for training was accumulated in an unloaded environment, and therefore it is crucial to show the learned inference models can generalize to perform under changes in sensor behavior that may come from varied dynamics within the system due to the out of plane loading, or vibrations that may occur due to wind. To do so, I performed the same series of step response tests in a 1'x1' wind tunnel under three additional flow speeds of 5 m/s, 7.5 m/s, and 10 m/s (Fig A.3). In comparing these performances, I only used the true deflection achieved by the controller when operating under the three different state observation methods, true, linear, and LSTM. As expected, when provided true state observations (indigo line) the PD controller had the strongest performance, tracking the step signal accurately regardless of additional airflow. The linear state inference model (teal line) appeared to perform well; however, there was a noticeable lack of accuracy in the intermediate step responses occurring between 15 and 35 seconds in the test sequences. The LSTM inference method (light green line) appeared capable of improving the PD controller's ability to track the target deflection signal in these intermediate steps, even under aerodynamic loading conditions, performing more similarly to the PD controller with true observations than the linear method.

Finally, although looking at the step response curves can provide some insight into the performance of the PD controller when using the various observation methods, using specific metrics can provide a means for more direct and quantifiable comparison. I performed the series of step response tests 5 times for each observation method at each loading condition and determined the average performance in traditional modeling and control metrics (Fig A.4). The first metric I considered was the inference model error, used to determine the difference in millimeters between the inferred deflection and the true deflection achieved by the morphing system. This metric showed that in

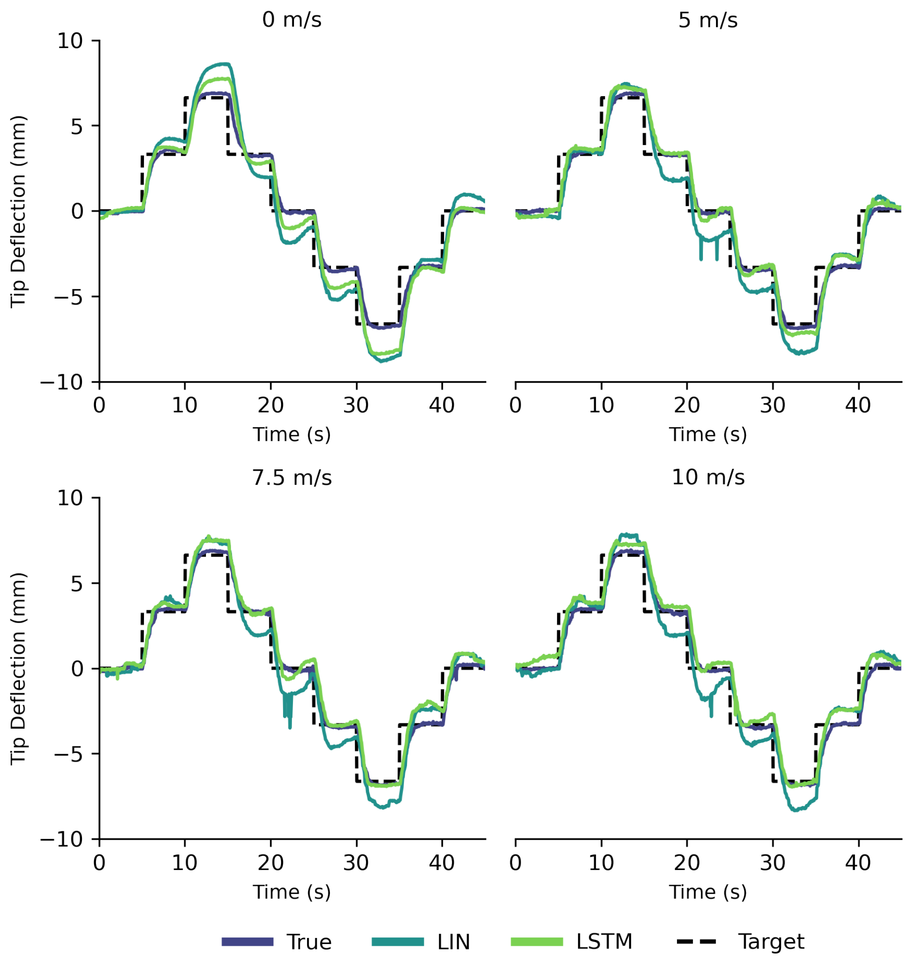

Figure A.3: Compared controller tracking performances using state inference methods. A series of eight step response tests performed at flow speeds 0 m/s, 5 m/s, 7.5 m/s, and 10 m/s show consistent improvement in tracking between tests where the PD controller was provided state observation information by the linear and LSTM inference models. This improvement is most noticeable at intermediate step response between times 15 and 30.

the unloaded environment the average error achieved by the LSTM model was nearly half of that produced by the linear method. Although that error increased slightly for the loaded environments, as indicated by filled points, the overall average error was still only 59% of that experienced by the linear models. The other metrics are specifically used to gauge the performance of the PD controller when operating with the variable levels of accuracy in state inference. This is the performance that the UAV ultimately will depend on. The first of these metrics was the settling time, used to determine how long it takes, in seconds, for the controller to bring the trailing edge to maintain a position within a distance of the goal deflection equal to 10% of the total step size. This metric shows that although the true deflection information provides the fastest control, on average the LSTM model provides control that is 1.6 seconds faster than the linear model. Another metric that is often balanced with controller speed is how far past the goal position the controller directs the deflecting trailing edge before settling on the correct position. This overshoot is measured as a percentage of the step size. Here the improved accuracy of the LSTM inference method reduced the average overshoot experienced by the controller when compared to the linear model by 10.6%. This was equivalent to cutting the overshoot to less than half of that experienced by the linear model. Finally, we considered rise time, another metric used to consider the quickness of the PD controller by measuring the time it takes to initially meet, and often surpass, the target position. Interestingly, we see an inverse relationship between what had been established in the previous metrics. In this case, the least accurate inference method achieved the quickest rise times on average. With that being said, all of the observation methods achieved rise times between 0.9 and 1.1 seconds, but this initial quickness may relate to the differences in overshoot and eventual final settling times.

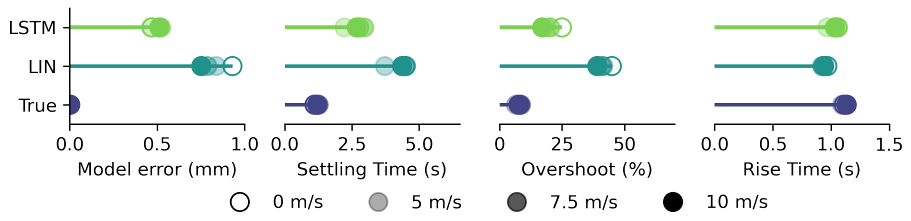

Figure A.4: Average controller metrics when using state inference methods. Metrics used to determine the impact of improved inference accuracy offered by the LSTM time-series model regarding traditional controller performance included model error, settling time, overshoot, and rise time. The values presented are the averages of 5 rounds of testing where each test included 8 step responses. The LSTM model achieved greater accuracy in estimating trailing edge deflection. This allowed the controller to achieve faster settling times and smaller overshooting improvements in accuracy did not show large difference in rise times.

Conclusion

In this work I showed the benefit of using time-series forecasting machine learning methods for state inference in smart material morphing aircraft systems. Long short-term memory networks greatly improved accuracy over traditional linear models for observing deflection states from piezoelectric flex sensor signals. The improved model accounted for hysteresis and creep behavior within the flex sensor system, providing accurate feedback for a traditionally tuned PD controller to achieve superior control authority in a series of step response tests.

APPENDIX B

PD vs PID control

When beginning my work designing a controller for the MFC morphing airfoil in chapter III, I developed a PID controller for a baseline comparison. I found that using a PD controller provided a better, and more interesting, baseline for this comparison due to the integral windup and overshoot experienced by the PID controller. After completing the work presented in Chapter III, I performed additional tuning, and used anti-windup techniques on original PID gains to compare and verify that the PD controller used was still a fair and interesting baseline for comparing to the learned controllers. This appendix presents the work used for this verification.

Method

To make sure that I provided a fair representation of the traditional control methods, I conducted additional experiments to compare the performance between the PD controller and more complex PID controller methods. To maintain consistency between controllers, I used the weights obtained using the Ziegler-Nichols open loop tuning method as a baseline and performed anti-windup and overshoot adjustments from there. All comparisons were made in still air. Additionally, the flex sensor signal

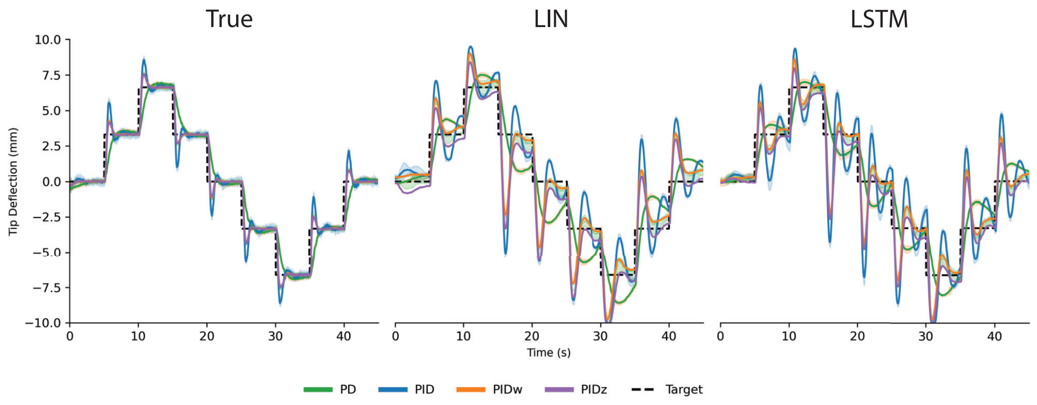

Figure B.1: A series of eight step response tests performed using true, LIN, and LSTM state inference to compare tracking performances between PD, PID, PIDw, and PIDz, controllers.

circuit was recreated since initial testing shown in Chapter III, so the normalization mean and standard deviations were adjusted to improve performance. Because of this, the performances cannot be directly compared to those experienced in Chapter III, but comparison between the methods shown here are fair. In addition to the PD method used, I showed the performance of the initial PID controller, a PID with a saturation based anti-windup component and a reduced integration function, reducing the integral component to incorporate only the two most recent timesteps (PIDw), and an additional PID controller that sets the integral to zero after switching between positive and negative error (PIDz) (Fig B.1).

Initial tests using the “True” state observations from the external laser showed that the PID methods were more accurate (measured by the reward) than the PD method used, but there is a significant increase in overshoot and the settling times were comparable (Fig B.2). When using the state inference methods, I notice a large change in performance. The PD method achieved the best accuracy and lowest overshoot. The PID methods experienced large oscillations about the goal position, gaining dramatically higher overshoot and settling times. There was an additional increase in inconsistency as shown by the error bars in these methods. When compared to the performances achieved by the learned controllers shown in Chapter III,

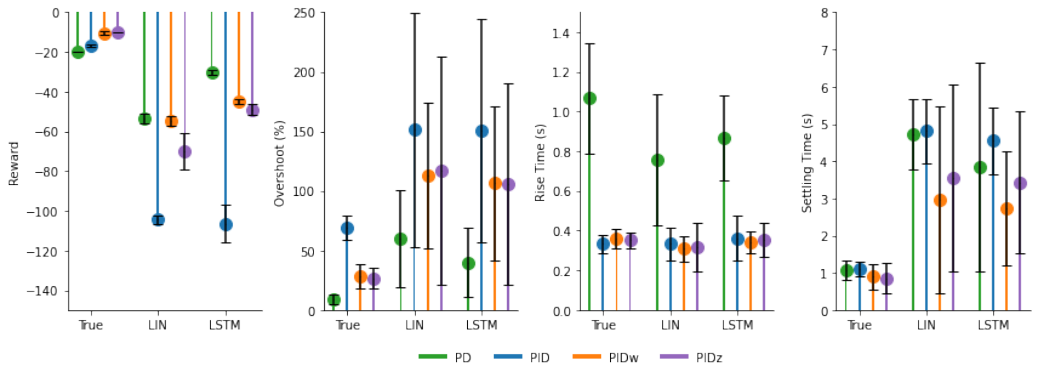

Figure B.2: Performance metrics used to compare the PD, PID, PIDw, and PIDz, controllers. These included: Reward, overshoot, rise time, and settling time. Metrics were mesured using true, LIN, and LSTM state inference methods.

the learned controllers achieve comparable and more often superior results to the PID based methods in speed, accuracy, and overshoot (particularly when using the state inference methods).

In addition to the PD controller's strong performance overall when paired with the LIN and LSTM state inference models, its ability to achieve low overshoot provided a definitive challenge for the learned controllers to compete with. The initial learned controller was fast and accurate but struggled to perform well in overshoot, which led to the development of a second learned RL controller designed to penalize overshoot. Therefore, I believe using the PD controller as a baseline was not only fair but provided a more interesting comparison.

APPENDIX C

Raw Tests for Gust Alleviation

In developing GLA controllers for a morphing wing, I performed extensive testing to provide a robust representation of the performance that can be expected from the DRL controllers. I trained ten controllers for each of the three pressure tap configurations, and tested the controllers ten times for each of the six gust conditions considered. Figure 4.6c in Chapter IV presents an example of the raw CLE measurements taken during the ten repeated tests for a single controller at a single gust condition. Here I present the same plot for all trained controllers under all test conditions for each of the three pressure tap configurations.

The average standard deviation between test repetitions for each controller and gust condition was 0.038 for the six pressure tap configuration.

The average standard deviation between test repetitions for each controller and gust condition was 0.043 for the three pressure tap configuration.

The average standard deviation between test repetitions for each controller and gust condition was 0.048 for the six pressure tap configuration.

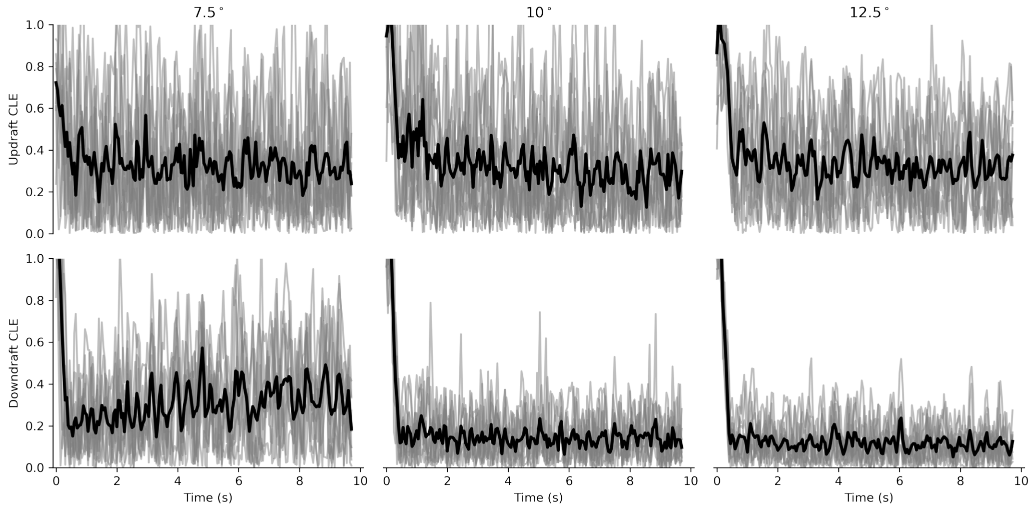

Figure C.1: CLE measurements of controller 1 for six pressure taps at all gust conditions.

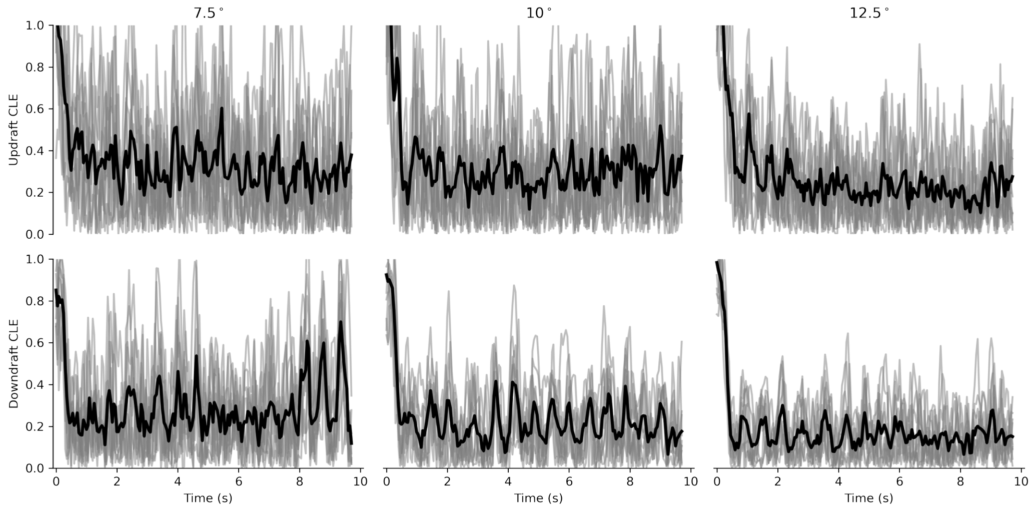

Figure C.2: CLE measurements of controller 2 for six pressure taps at all gust conditions.

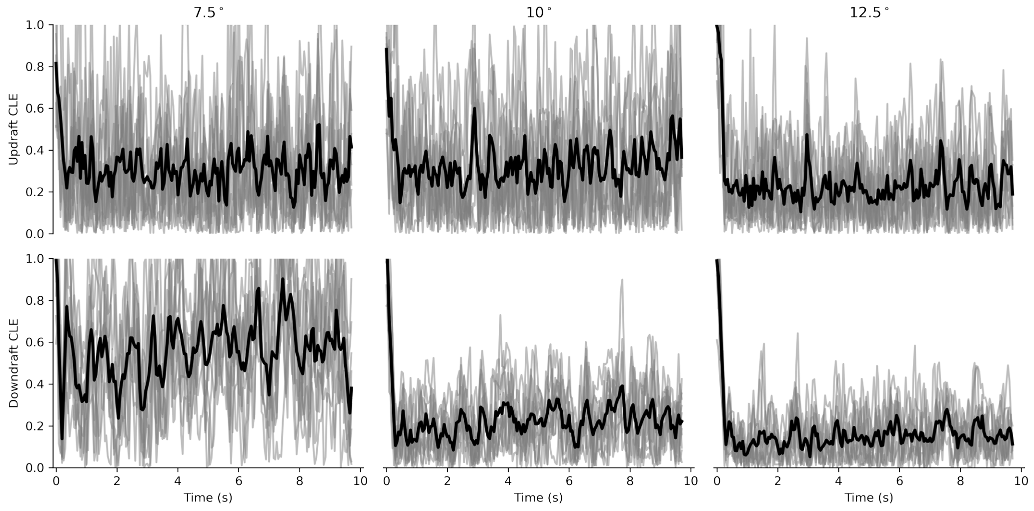

Figure C.3: CLE measurements of controller 3 for six pressure taps at all gust conditions.

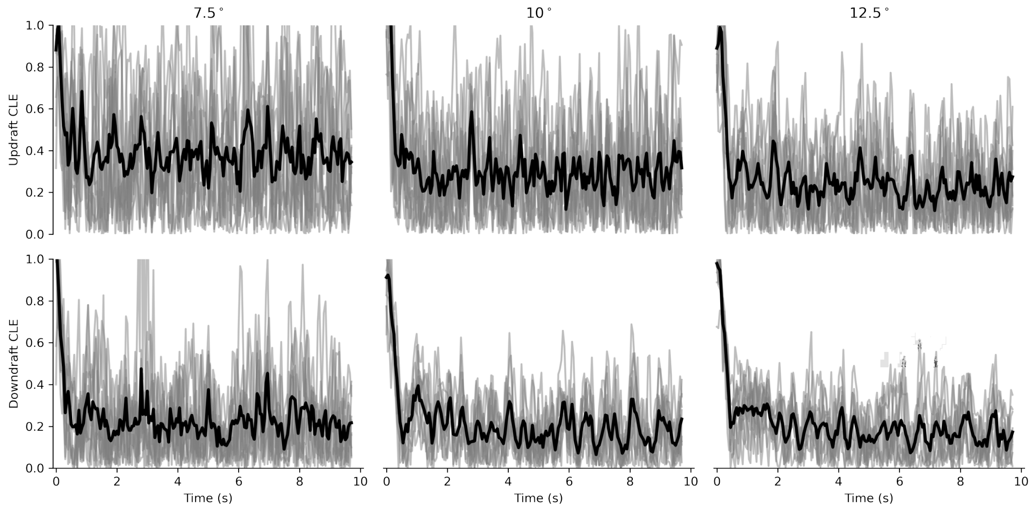

Figure C.4: CLE measurements of controller 4 for six pressure taps at all gust conditions.

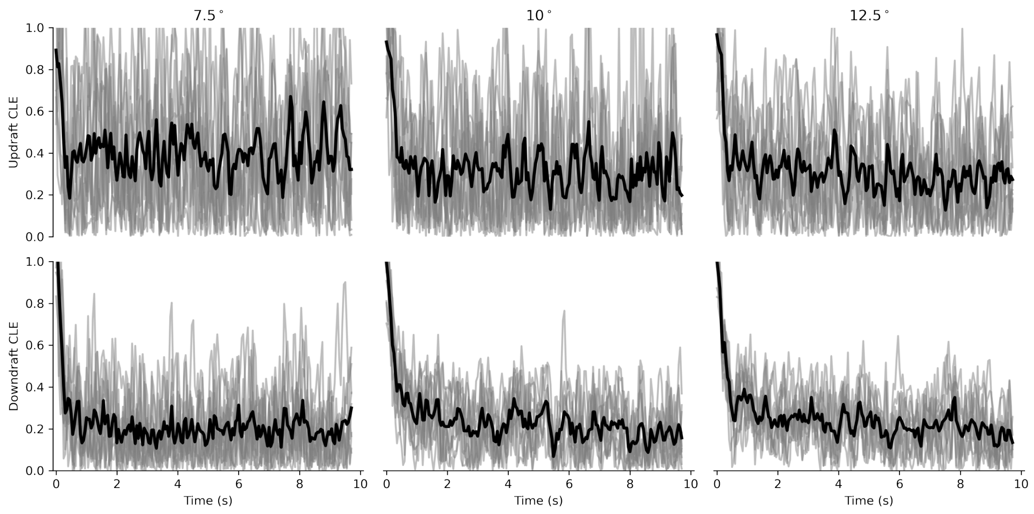

Figure C.5: CLE measurements of controller 5 for six pressure taps at all gust conditions.

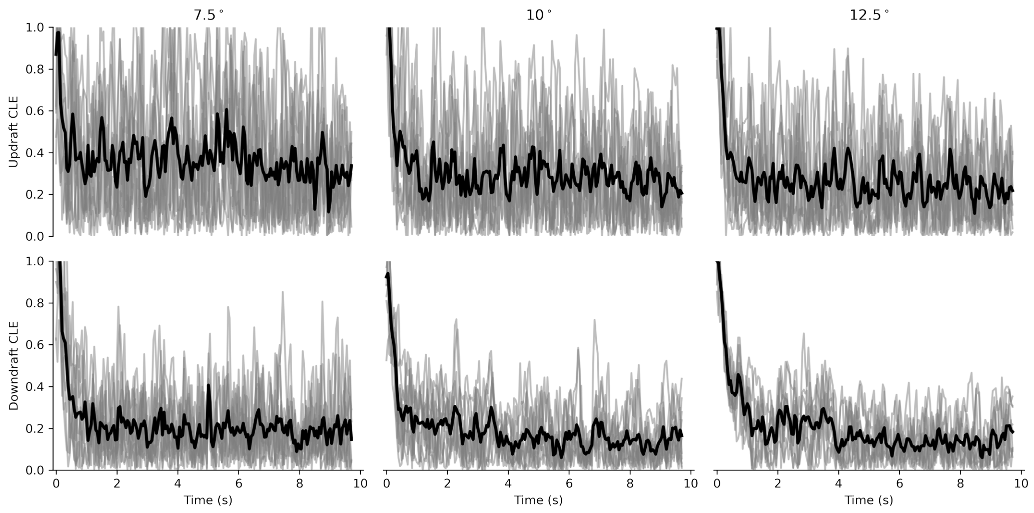

Figure C.6: CLE measurements of controller 6 for six pressure taps at all gust conditions.

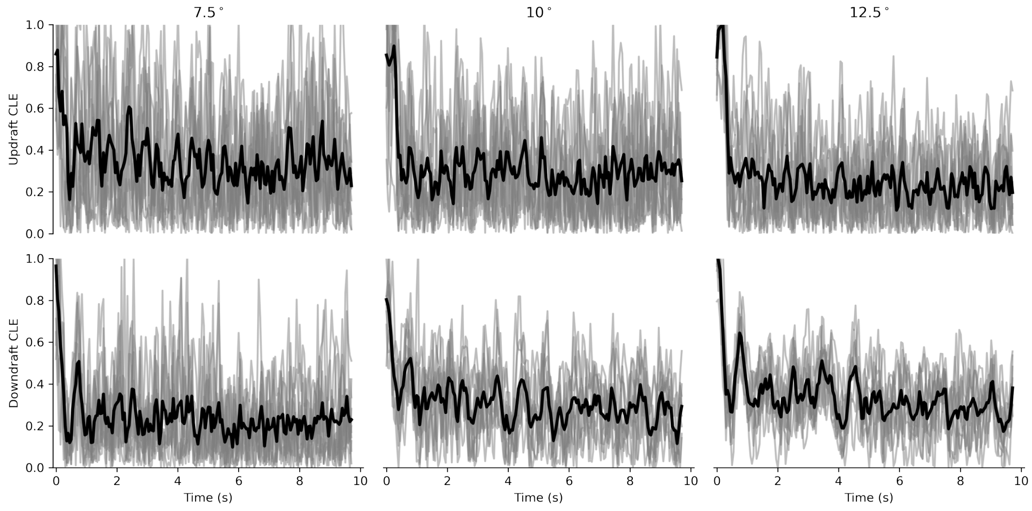

Figure C.7: CLE measurements of controller 7 for six pressure taps at all gust conditions.

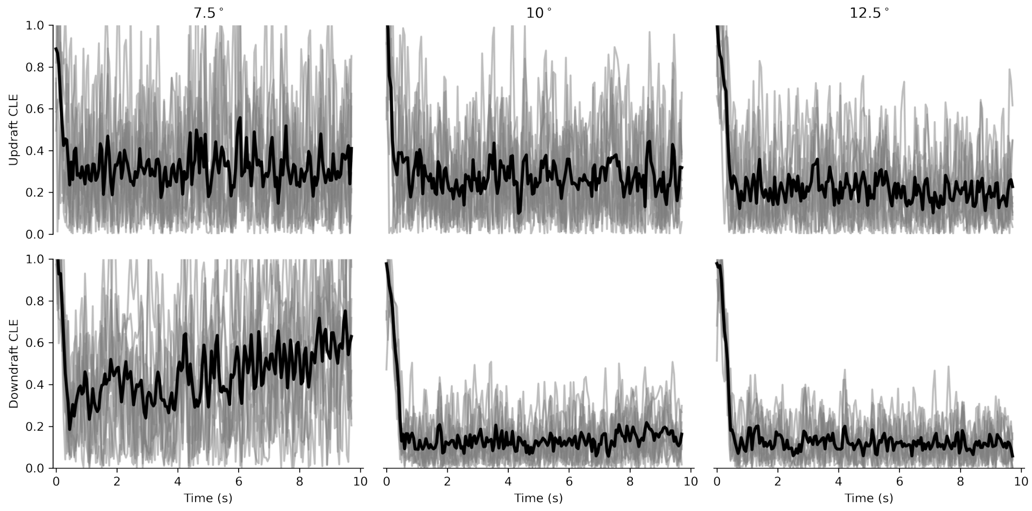

Figure C.8: CLE measurements of controller 8 for six pressure taps at all gust conditions.

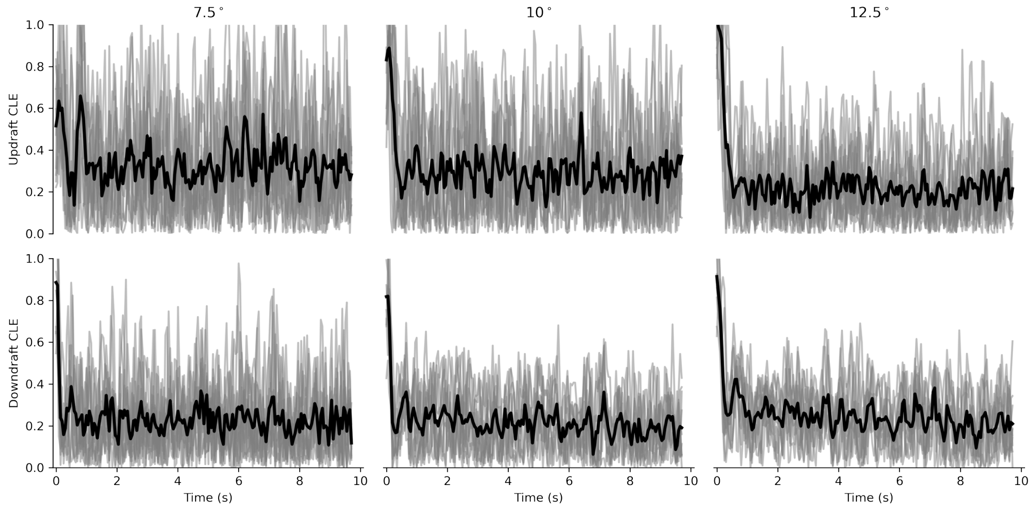

Figure C.9: CLE measurements of controller 9 for six pressure taps at all gust conditions.

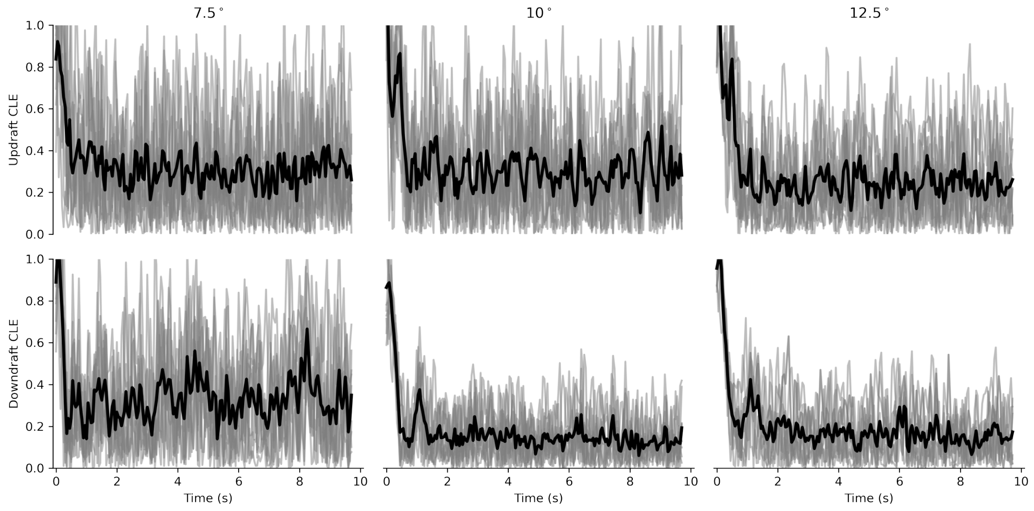

Figure C.10: CLE measurements of controller 10 for six pressure taps at all gust conditions.

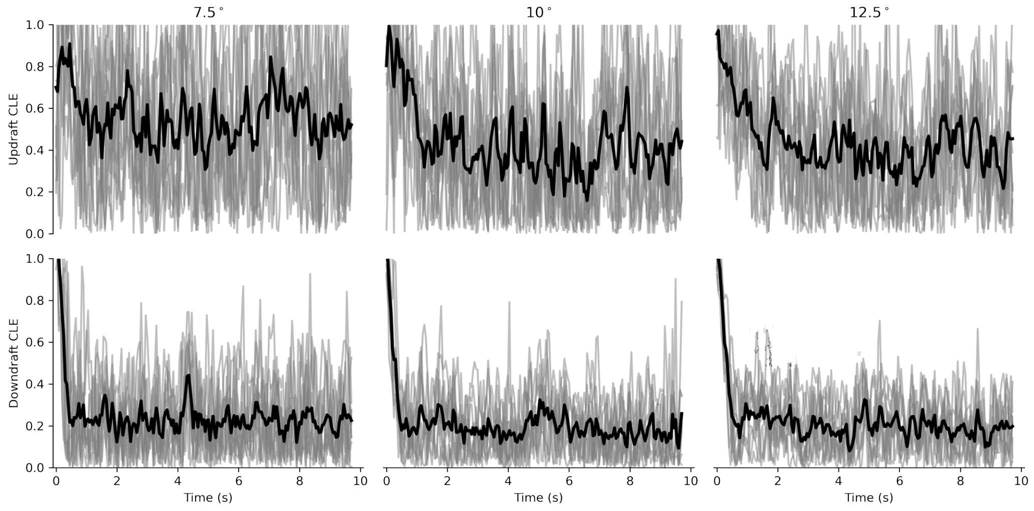

Figure C.11: CLE measurements of controller 1 for three pressure taps at all gust conditions.

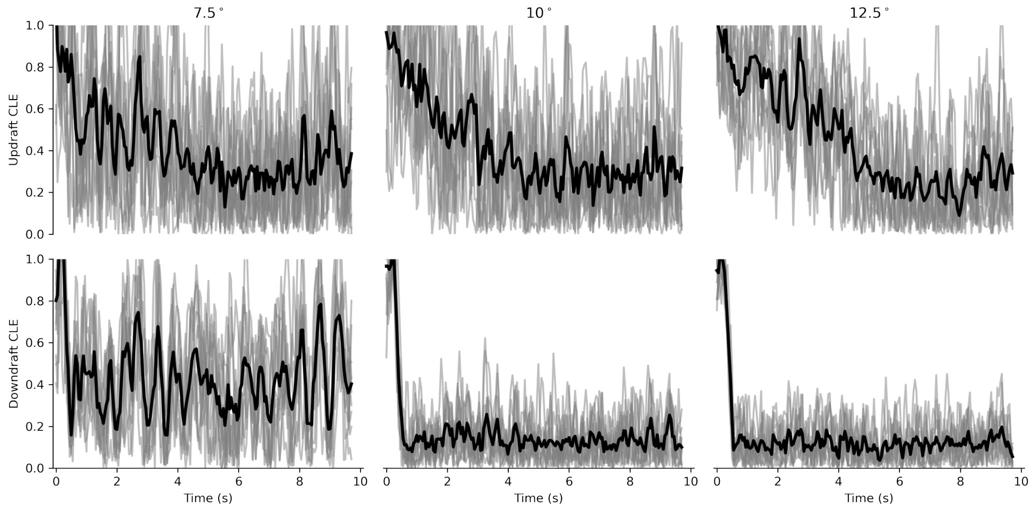

Figure C.12: CLE measurements of controller 2 for three pressure taps at all gust conditions.

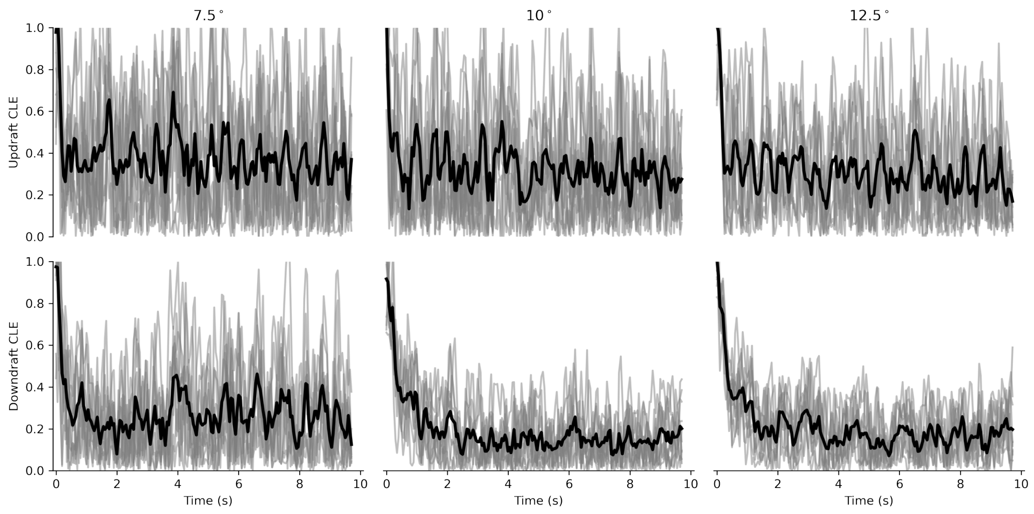

Figure C.13: CLE measurements of controller 3 for three pressure taps at all gust conditions.

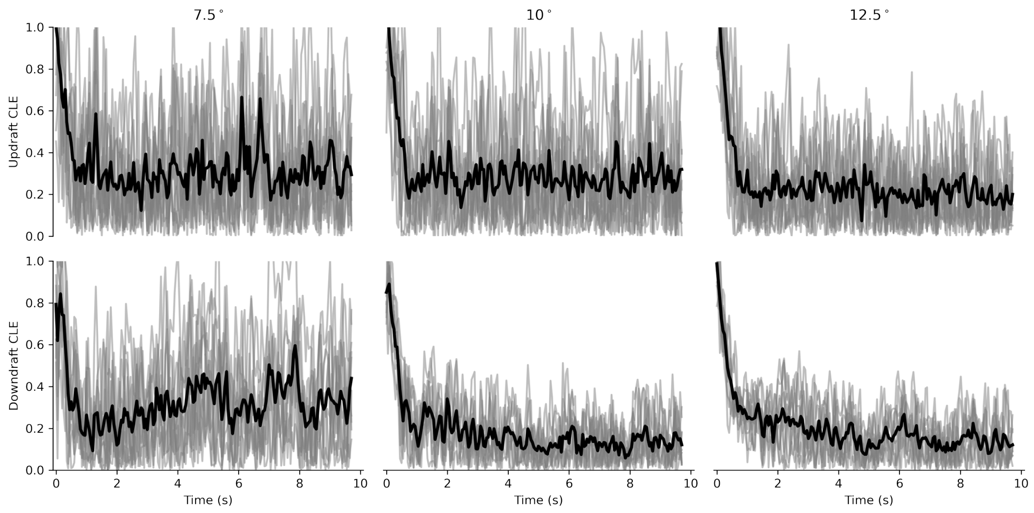

Figure C.14: CLE measurements of controller 4 for three pressure taps at all gust conditions.

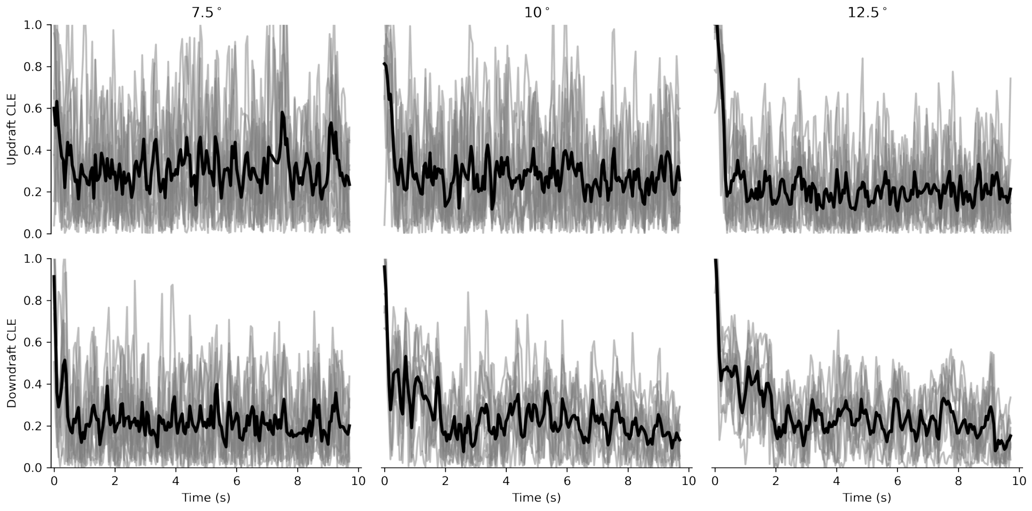

Figure C.15: CLE measurements of controller 5 for three pressure taps at all gust conditions.

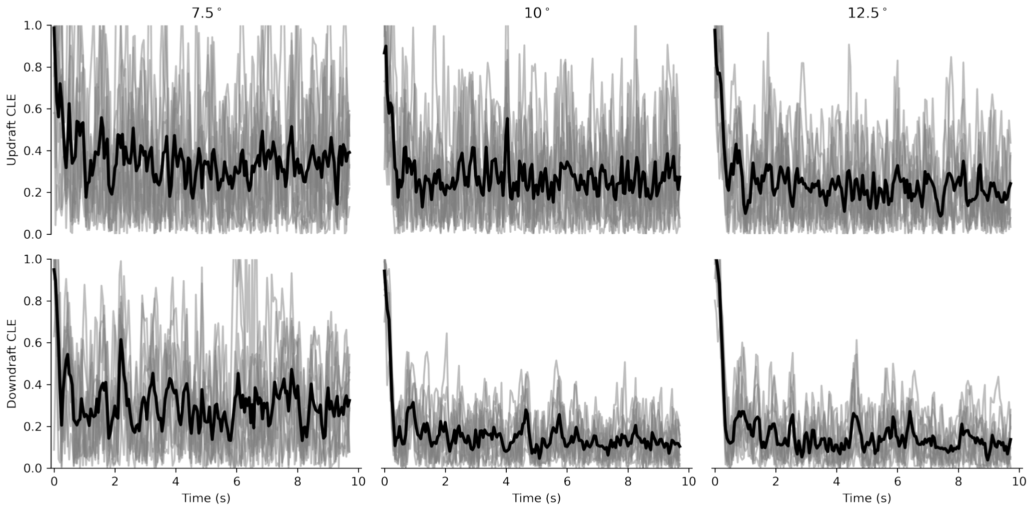

Figure C.16: CLE measurements of controller 6 for three pressure taps at all gust conditions.

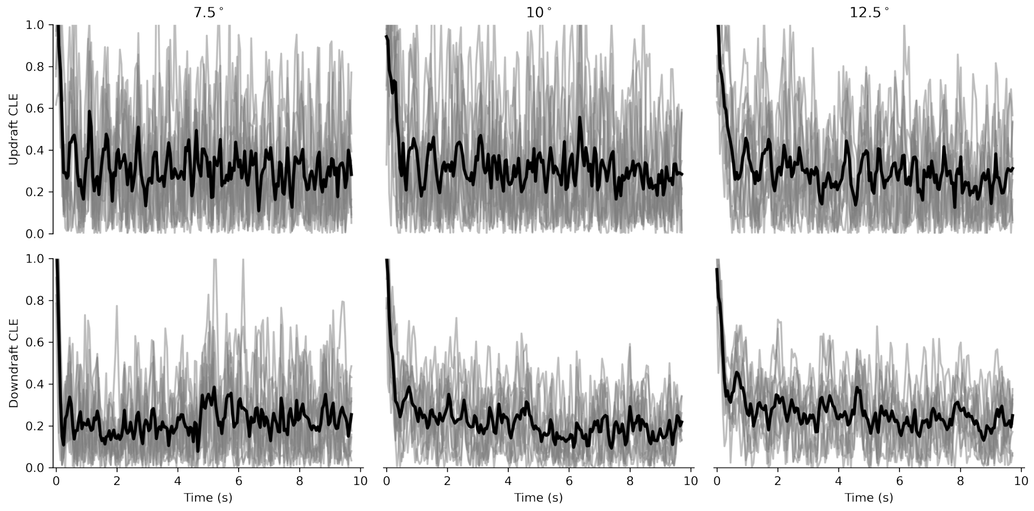

Figure C.17: CLE measurements of controller 7 for three pressure taps at all gust conditions.

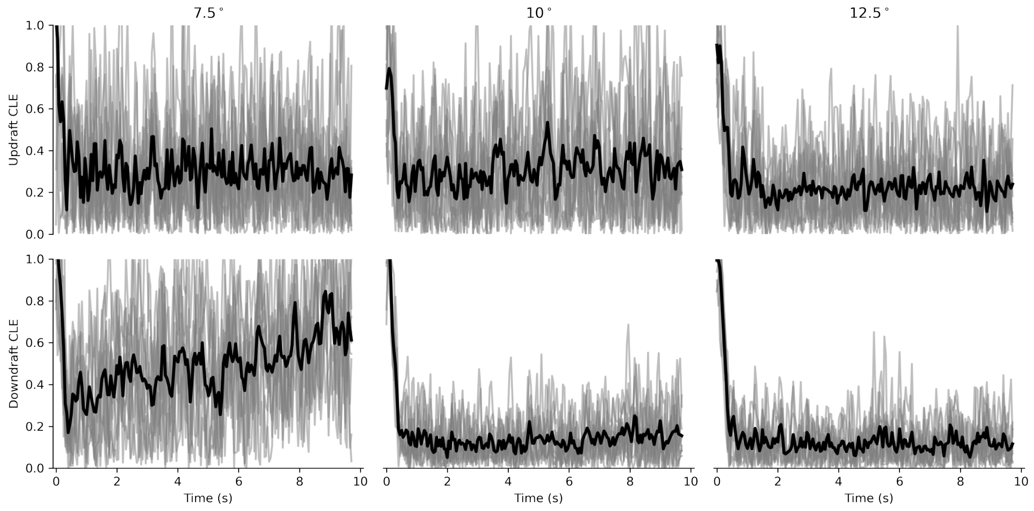

Figure C.18: CLE measurements of controller 8 for three pressure taps at all gust conditions.

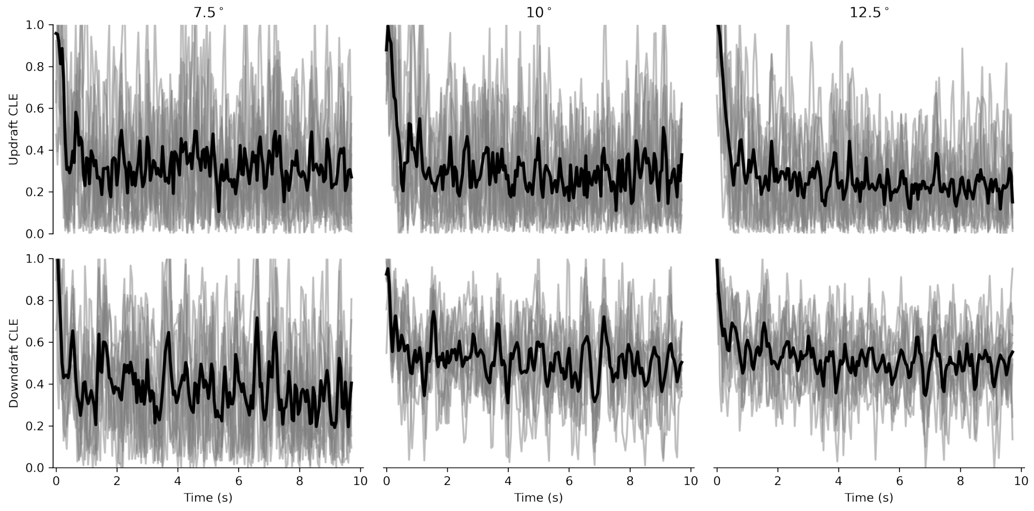

Figure C.19: CLE measurements of controller 9 for three pressure taps at all gust conditions.

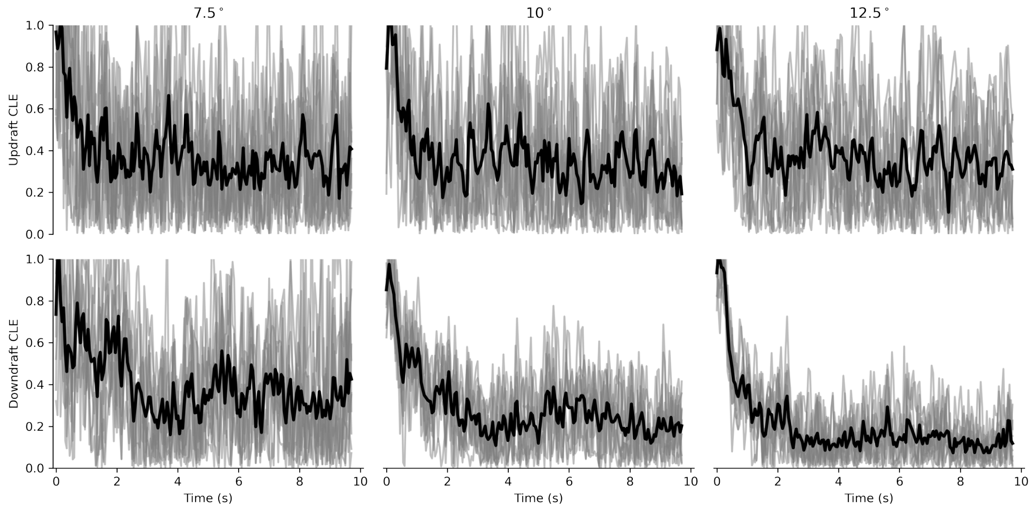

Figure C.20: CLE measurements of controller 10 for three pressure taps at all gust conditions.

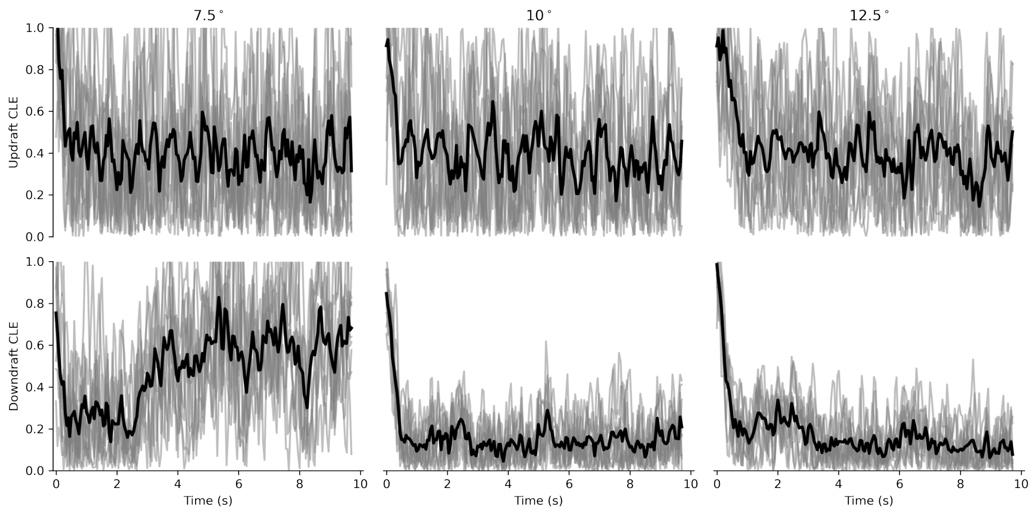

Figure C.21: CLE measurements of controller 1 for one pressure tap at all gust conditions.

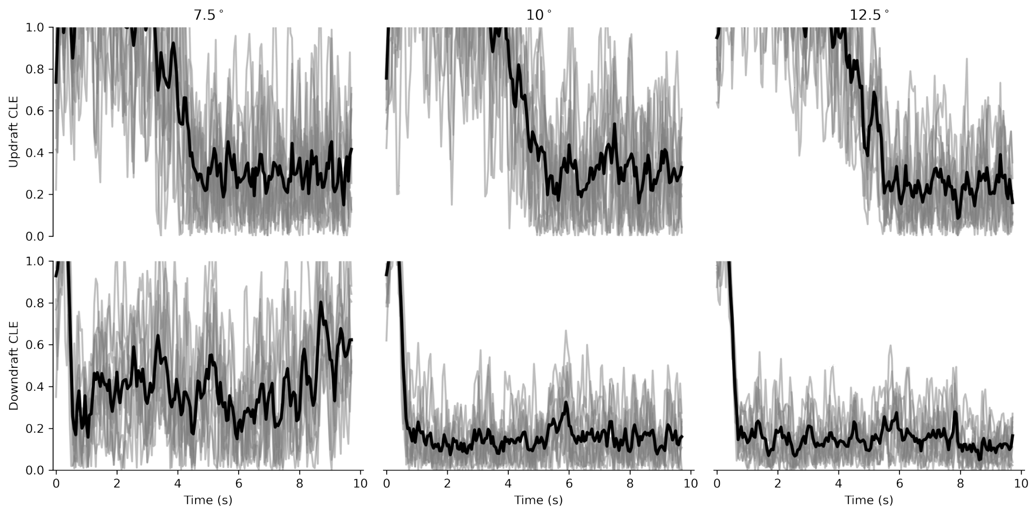

Figure C.22: CLE measurements of controller 2 for one pressure tap at all gust conditions.

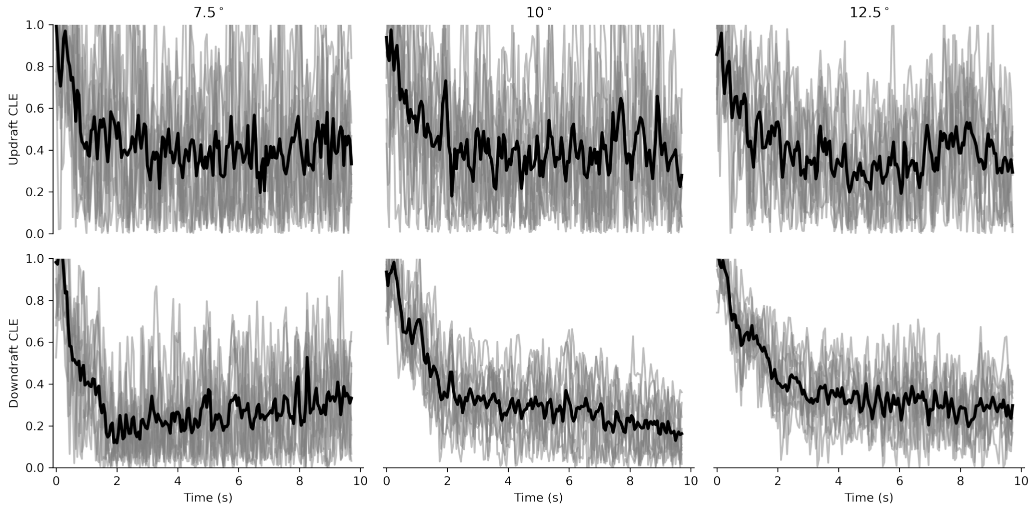

Figure C.23: CLE measurements of controller 3 for one pressure tap at all gust conditions.

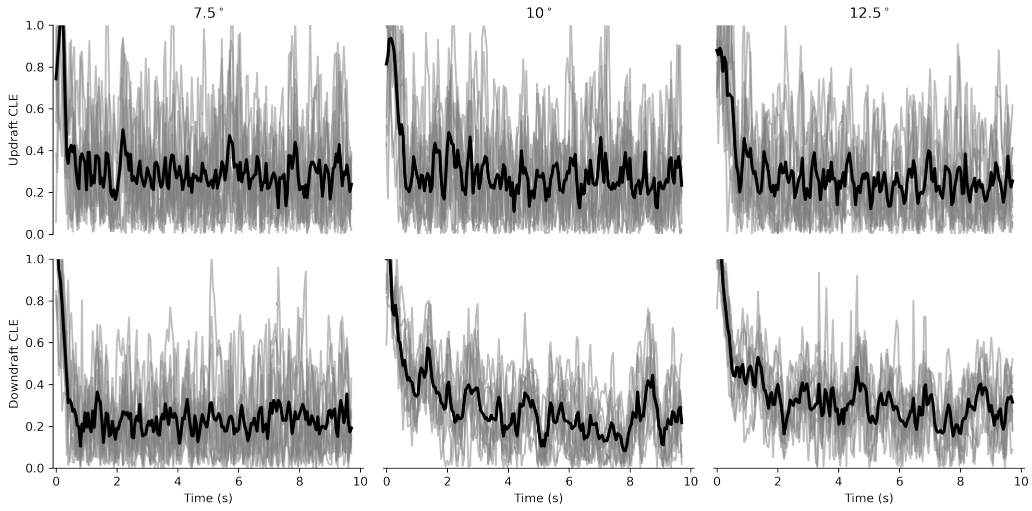

Figure C.24: CLE measurements of controller 4 for one pressure tap at all gust conditions.

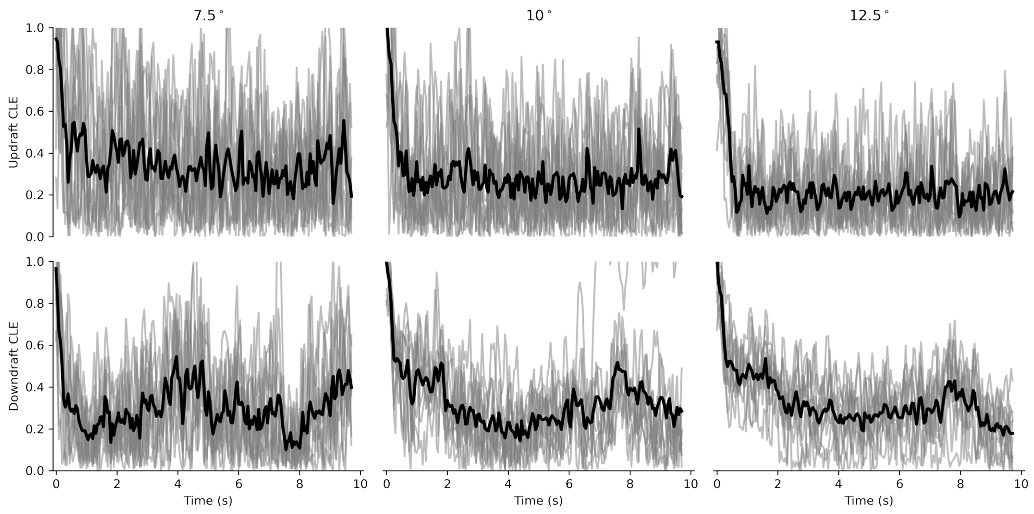

Figure C.25: CLE measurements of controller 5 for one pressure tap at all gust conditions.

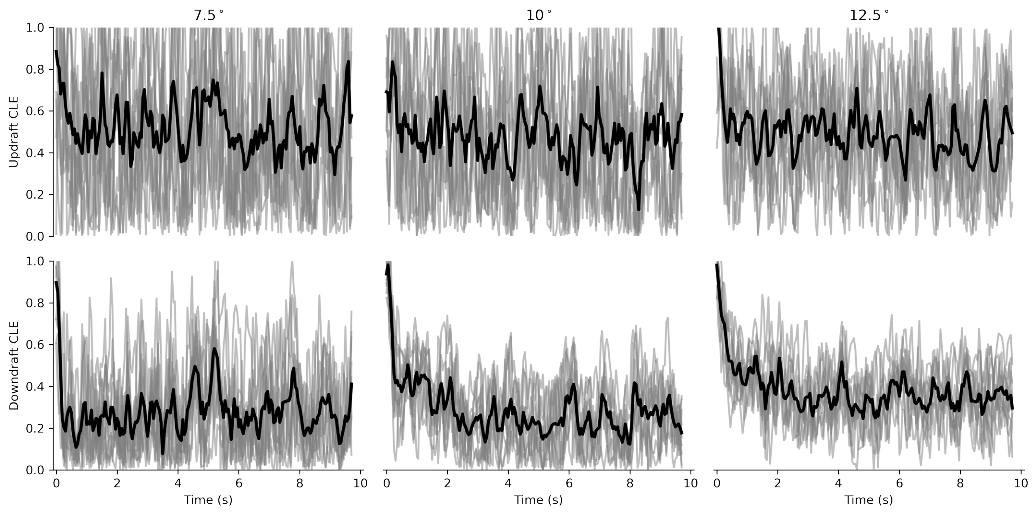

Figure C.26: CLE measurements of controller 6 for one pressure tap at all gust conditions.

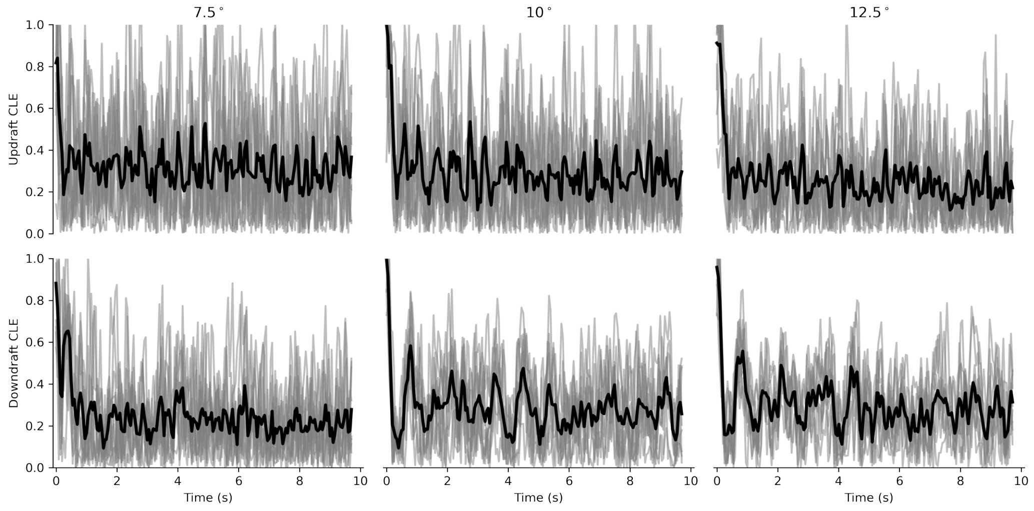

Figure C.27: CLE measurements of controller 7 for one pressure tap at all gust conditions.

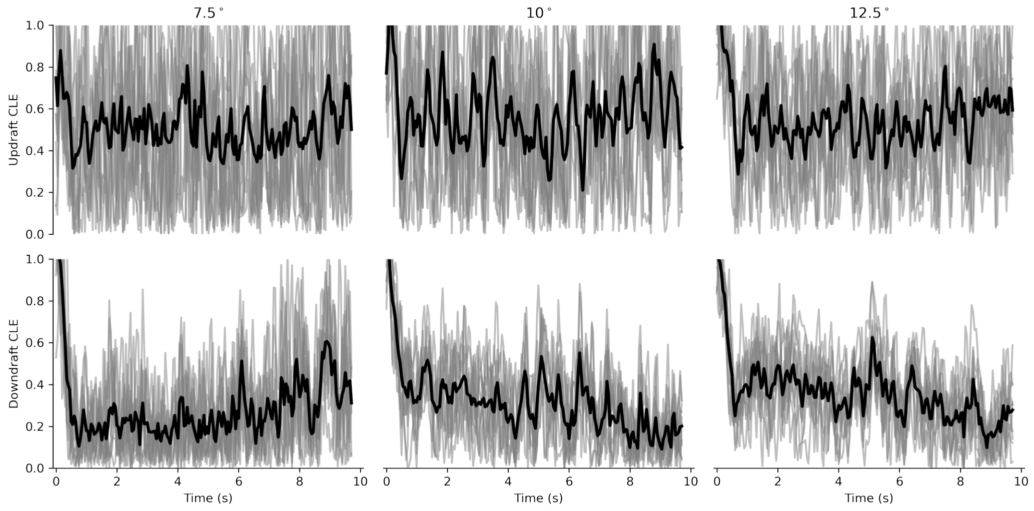

Figure C.28: CLE measurements of controller 8 for one pressure tap at all gust conditions.

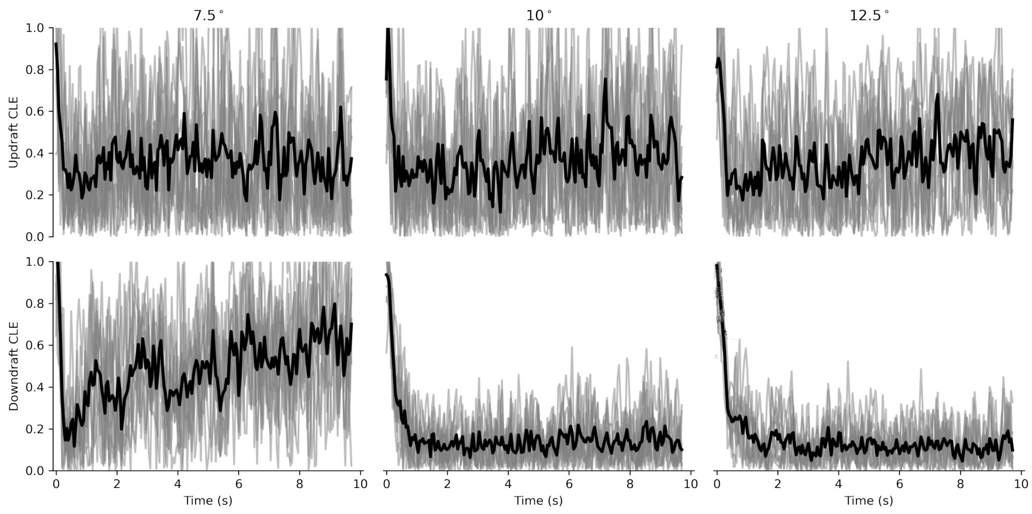

Figure C.29: CLE measurements of controller 9 for one pressure tap at all gust conditions.

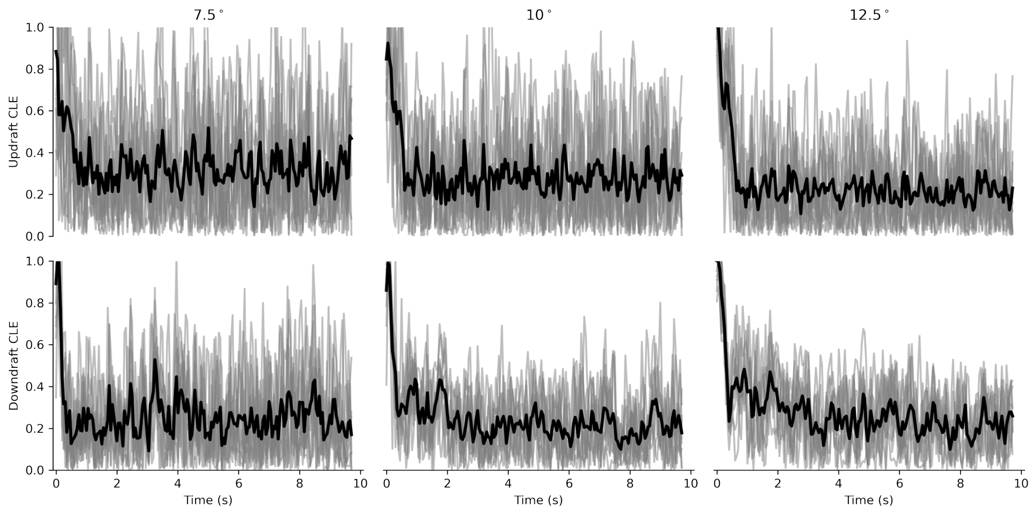

Figure C.30: CLE measurements of controller 10 for one pressure tap at all gust conditions.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Barbarino, O. Bilgen, R. M. Ajaj, M. I. Friswell, and D. J. Inman, "A review of morphing aircraft," *Journal of intelligent material systems and structures*, vol. 22, no. 9, pp. 823–877, 2011, Publisher: Sage Publications Sage UK: London, England, ISSN: 1045-389X.

[2] A. K. Jha and J. N. Kudva, "Morphing aircraft concepts, classifications, and challenges," vol. 5388, International Society for Optics and Photonics, 2004, pp. 213–224.

[3] J. C. Gomez and E. Garcia, "Morphing unmanned aerial vehicles," *Smart Materials and Structures*, vol. 20, no. 10, p. 103 001, 2011, Publisher: IOP Publishing, ISSN: 0964-1726.

[4] J. Sun, Q. Guan, Y. Liu, and J. Leng, "Morphing aircraft based on smart materials and structures: A state-of-the-art review," *Journal of Intelligent material systems and structures*, vol. 27, no. 17, pp. 2289–2312, 2016, Publisher: SAGE Publications Sage UK: London, England, ISSN: 1045-389X.

[5] A. M. Pankonien, "Smart Material Wing Morphing for Unmanned Aerial Vehicles.," 2015.

[6] J. Bowman, B. Sanders, and T. Weisshaar, "Evaluating the impact of morphing technologies on aircraft performance," 2002, p. 1631.

[7] "Improved Adaptive–Reinforcement Learning Control for Morphing Unmanned Air Vehicles," vol. 38, pp. 1014–1020, Aug. 2008, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), ISSN: 1941-0492. DOI: 10.1109/TSMCB.2008.922018.

[8] V. G. Goecks, P. B. Leal, T. White, J. Valasek, and D. J. Hartl, "Control of Morphing Wing Shapes with Deep Reinforcement Learning," en, in *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2018, ISBN: 978-1-62410-527-2. DOI: 10.2514/6.2018-2139.

[9] C. Harvey, "Avian Wing Joints Provide Longitudinal Flight Stability and Control," 2022.

[10] E. R. Molner, "Using Vertical Gust Alleviation to Improve the Target Tracking Capability of the Control Configured YF-16.," en, Tech. Rep., 1975, Section: Technical Reports. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA080520 (visited on 10/27/2022).

[11] W. Pisano and D. Lawrence, "Autonomous gust insensitive aircraft," 2008, p. 6510.

[12] C. Harvey, V. Baliga, J. Wong, D. Altshuler, and D. Inman, "Birds can transition between stable and unstable states via wing morphing," *Nature*, vol. 603, no. 7902, pp. 648–653, 2022, Publisher: Nature Publishing Group, ISSN: 1476-4687.

[13] B. Moulin and M. Karpel, "Gust loads alleviation using special control surfaces," *Journal of Aircraft*, vol. 44, no. 1, pp. 17–25, 2007, ISSN: 0021-8669.

[14] Y. Li, "Gust Load Alleviation by Fluidic Actuators on a Blended-Wing-Body Configuration," en, p. 222, 2020.

[15] S. Guo, Z. W. Jing, H. Li, W. T. Lei, and Y. Y. He, "Gust response and body freedom flutter of a flying-wing aircraft with a passive gust alleviation device," en, *Aerospace Science and Technology*, vol. 70, pp. 277–285, Nov. 2017, ISSN: 1270-9638. DOI: `10.1016/j.ast.2017.08.008`.

[16] S. Guo, J. E. De Los Monteros, and Y. Liu, "Gust Alleviation of a Large Aircraft with a Passive Twist Wingtip," en, *Aerospace*, vol. 2, no. 2, pp. 135–154, Jun. 2015, Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2226-4310. DOI: `10.3390/aerospace2020135`.

[17] J. Zeng, B. Moulin, R. de Callafon, and M. J. Brenner, "Adaptive Feedforward Control for Gust Load Alleviation," en, *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 3, pp. 862–872, May 2010, ISSN: 0731-5090, 1533-3884. DOI: `10.2514/1.46091`.

[18] Z. Wu, L. Chen, C. Yang, and C. Tang, "Gust response modeling and alleviation scheme design for an elastic aircraft," en, *Science China Technological Sciences*, vol. 53, no. 11, pp. 3110–3118, Nov. 2010, ISSN: 1862-281X. DOI: `10.1007/s11431-010-4141-y`.

[19] "Model Predictive Control for Aircraft Load Alleviation: Opportunities and Challenges," ISSN: 2378-5861, Jun. 2018, pp. 2417–2424. DOI: `10.23919/ACC.2018.8430956`.

[20] M. Dillsaver, C. Cesnik, and I. Kolmanovsky, "Gust Load Alleviation Control for Very Flexible Aircraft," in *AIAA Atmospheric Flight Mechanics Conference*, ser. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2011. DOI: `10.2514/6.2011-6368`.

[21] C. D. Regan and C. V. Jutte, "Survey of Applications of Active Control Technology for Gust Alleviation and New Challenges for Lighter-weight Aircraft," Tech. Rep. DFRC-E-DAA-TN4736, Apr. 2012, NTRS Author Affiliations: NASA Dryden Flight Research Center NTRS Document ID: 20120013450 NTRS Research Center: Armstrong Flight Research Center (AFRC). [Online]. Available: `https://ntrs.nasa.gov/citations/20120013450` (visited on 09/04/2022).

[22] Y. Hamada, K. Saitoh, and N. Kobiki, "Gust Alleviation Control using Prior Gust Information: Wind Tunnel Test Results," en, *IFAC-PapersOnLine*, 21st IFAC Symposium on Automatic Control in Aerospace ACA 2019, vol. 52, no. 12, pp. 128–133, Jan. 2019, ISSN: 2405-8963. DOI: `10.1016/j.ifacol.2019.11.125`.

[23] A. M. Pankonien, K. S. T. Magar, R. V. Beblo, and G. W. Reich, "Gust prediction via artificial hair sensor array and neural network," in *A Tribute Conference Honoring Daniel Inman*, vol. 10172, SPIE, Apr. 2017, pp. 55–64. DOI: `10.1117/12.2257243`.

[24] X. Wang, T. Mkhoyan, I. Mkhoyan, and R. De Breuker, "Seamless Active Morphing Wing Simultaneous Gust and Maneuver Load Alleviation," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 9, pp. 1649–1662, Sep. 2021, Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0731-5090. DOI: `10.2514/1.G005870`.

[25] R. W. Wlezien, G. C. Horner, A.-M. R. McGowan, *et al.*, "Aircraft morphing program," vol. 3326, SPIE, 1998, pp. 176–187.

[26] A.-M. R. McGowan, L. G. Horta, J. S. Harrison, and D. L. Raney, "Research activities within NASA's morphing program," NATIONAL AERONAUTICS and SPACE ADMINISTRATION HAMPTON VA LANGLEY RESEARCH CENTER, Tech. Rep., 2000.

[27] D. C. Lagoudas, *Shape memory alloys: modeling and engineering applications.* Springer, 2008, ISBN: 0-387-47685-7.

[28] K. P. T. Haughn, L. L. Gamble, and D. J. Inman, "Horizontal Planform Morphing Tail for an Avian Inspired UAV Using Shape Memory Alloys," en, American Society of Mechanical Engineers Digital Collection, Nov. 2018. DOI: `10.1115/SMASIS2018-7986`.

[29] J. Mabe, F. Calkins, and G. Butler, "Boeing's variable geometry chevron, morphing aerostructure for jet noise reduction," 2006, p. 2142.

[30] L. L. Gamble and D. J. Inman, "A tale of two tails: Developing an avian inspired morphing actuator for yaw control and stability," *Bioinspiration & biomimetics*, vol. 13, no. 2, p. 026 008, 2018, Publisher: IOP Publishing, ISSN: 1748-3190.

[31] W. K. Wilkie, R. G. Bryant, J. W. High, *et al.*, "Low-cost piezocomposite actuator for structural control applications," vol. 3991, SPIE, 2000, pp. 323–334.

[32] V. Wickramasinghe, Y. Chen, M. Martinez, F. Wong, and R. Kernaghan, "Design and verification of a smart wing for an extremely-agile micro-air-vehicle," 2009, p. 2132.

[33] O. J. Ohanian III, E. D. Karni, C. C. Olien, *et al.*, "Piezoelectric composite morphing control surfaces for unmanned aerial vehicles," vol. 7981, SPIE, 2011, pp. 1486–1498.

[34] A. Pankonien and D. J. Inman, “Experimental testing of spanwise morphing trailing edge concept,” vol. 8688, International Society for Optics and Photonics, 2013, p. 868 815.

[35] G. H. Haertling, “Ferroelectric ceramics: History and technology,” *Journal of the American Ceramic Society*, vol. 82, no. 4, pp. 797–818, 1999, Publisher: Wiley Online Library, ISSN: 0002-7820.

[36] J. W. High, *Method of fabricating NASA-standard macro-fiber composite piezoelectric actuators.* National Aeronautics and Space Administration, Langley Research Center, 2003.

[37] O. Bilgen, K. B. Kochersberger, D. J. Inman, and O. J. Ohanian III, “Novel, bidirectional, variable-camber airfoil via macro-fiber composite actuators,” *Journal of aircraft*, vol. 47, no. 1, pp. 303–314, 2010, ISSN: 0021-8669.

[38] S. M. Corp., *Mfc-macrofibercomposite*, `https://www.smart-material.com/index.html`, Accessed: 2022-11-29, 2021.

[39] B. Sanders, F. Eastep, and E. Forster, “Aerodynamic and aeroelastic characteristics of wings with conformal control surfaces for morphing aircraft,” *Journal of aircraft*, vol. 40, no. 1, pp. 94–99, 2003, ISSN: 0021-8669.

[40] B. Woods, O. Bilgen, and M. Friswell, “Wind tunnel testing of the fish bone active camber morphing concept,” *Journal of intelligent material systems and structures*, vol. 25, no. 7, pp. 772–785, 2014.

[41] W. W. Gilbert, “Mission adaptive wing system for tactical aircraft,” *Journal of Aircraft*, vol. 18, no. 7, pp. 597–602, 1981, ISSN: 0021-8669.

[42] J. Hetrick, R. Osborn, S. Kota, P. Flick, and D. Paul, “Flight testing of mission adaptive compliant wing,” in *48th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference*, 2007, p. 1709.

[43] L. L. Gamble, A. M. Pankonien, and D. J. Inman, “Stall recovery of a morphing wing via extended nonlinear lifting-line theory,” *AIAA Journal*, vol. 55, no. 9, pp. 2956–2963, 2017, Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0001-1452.

[44] G. Molinari, A. F. Arrieta, and P. Ermanni, “Planform, aero-structural, and flight control optimization for tailless morphing aircraft,” vol. 9431, International Society for Optics and Photonics, 2015, 94310Y.

[45] C. Zhang, J. Qiu, Y. Chen, and H. Ji, “Modeling hysteresis and creep behavior of macrofiber composite–based piezoelectric bimorph actuator,” *Journal of Intelligent Material Systems and Structures*, vol. 24, no. 3, pp. 369–377, 2013, Publisher: Sage Publications Sage UK: London, England, ISSN: 1045-389X.

[46] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018, ISBN: 0-262-35270-2.

[47] V. Mnih, A. P. Badia, M. Mirza, *et al.*, “Asynchronous methods for deep reinforcement learning,” PMLR, 2016, pp. 1928–1937.

[48] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015, Publisher: Nature Publishing Group, ISSN: 1476-4687.

[49] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, “Reinforcement learning through asynchronous advantage actor-critic on a gpu,” *arXiv preprint arXiv:1611.06256*, 2016.

[50] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, *Playing Atari with Deep Reinforcement Learning*, arXiv:1312.5602 [cs], Dec. 2013. DOI: `10.48550/arXiv.1312.5602`.

[51] C. Berner, G. Brockman, B. Chan, *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv preprint arXiv:1912.06680*, 2019.

[52] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018, Publisher: American Association for the Advancement of Science, ISSN: 0036-8075.

[53] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen, “Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization,” IEEE, 2019, pp. 523–533, ISBN: 1-72810-333-9.

[54] W. Koch, R. Mancuso, R. West, and A. Bestavros, “Reinforcement learning for UAV attitude control,” *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019, Publisher: ACM New York, NY, USA, ISSN: 2378-962X.

[55] K. Kersandt, “Deep reinforcement learning as control method for autonomous uavs,” 2018, Publisher: Universitat Politècnica de Catalunya.

[56] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, “Autonomous navigation of UAV by using real-time model-based reinforcement learning,” IEEE, 2016, pp. 1–6, ISBN: 1-5090-3549-4.

[57] A. Y. Ng, A. Coates, M. Diel, *et al.*, “Autonomous inverted helicopter flight via reinforcement learning,” in *Experimental robotics IX*, Springer, 2006, pp. 363–372.

[58] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, “Control of a quadrotor with reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017, Publisher: IEEE, ISSN: 2377-3766.

[59] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, “Autonomous uav navigation using reinforcement learning,” *arXiv preprint arXiv:1801.05086*, 2018.

[60] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” IEEE, 2017, pp. 31–36, ISBN: 1-5386-2682-9.

[61] O. M. Andrychowicz, B. Baker, M. Chociej, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020, Publisher: SAGE Publications Sage UK: London, England, ISSN: 0278-3649.

[62] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," IEEE, 2017, pp. 23–30, ISBN: 1-5386-2682-9.

[63] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010, Publisher: IEEE, ISSN: 1070-9932.

[64] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013, Publisher: Sage Publications Sage UK: London, England, ISSN: 0278-3649.

[65] K. P. T. Haughn and D. J. Inman, "Autonomous Learning in a Pseudo-Episodic Physical Environment," en, *Journal of Intelligent & Robotic Systems*, vol. 104, no. 2, p. 32, Feb. 2022, ISSN: 1573-0409. DOI: 10.1007/s10846-022-01577-5.

[66] K. P. Haughn, L. L. Gamble, and D. J. Inman, "Mfc morphing aileron control with intelligent sensing," in *Smart Materials, Adaptive Structures and Intelligent Systems*, American Society of Mechanical Engineers, vol. 86274, 2022, V001T03A013.

[67] K. P. Haughn, L. L. Gamble, and D. J. Inman, "Deep reinforcement learning achieves multifunctional morphing airfoil control," *Journal of Composite Materials*, 2022. DOI: 10.1177/00219983221137644.

[68] M. Hessel, J. Modayil, H. Van Hasselt, *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," 2018.

[69] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *Advances in neural information processing systems*, vol. 30, 2017.

[70] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," PMLR, 2018, pp. 1861–1870, ISBN: 2640-3498.

[71] G. Konidaris, S. Osentoski, and P. Thomas, "Value function approximation in reinforcement learning using the Fourier basis," 2011.

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

[73] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," PMLR, 2015, pp. 1889–1897.

[74] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, *et al.*, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021, Publisher: Springer, ISSN: 1573-0565.

[75] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," *arXiv preprint arXiv:1910.01708*, 2019.

[76] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," PMLR, 2019, pp. 2052–2062, ISBN: 2640-3498.

[77] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning*, Springer, 2012, pp. 45–73.

[78] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

[79] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," vol. 4, IEEE, 1997, pp. 3557–3564, ISBN: 0-7803-3612-7.

[80] W. Caarls and E. Schuitema, "Parallel online temporal difference learning for motor control," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 7, pp. 1457–1468, 2015, Publisher: IEEE, ISSN: 2162-237X.

[81] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," PMLR, 2017, pp. 1126–1135, ISBN: 2640-3498.

[82] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," vol. 4, IEEE, 2002, pp. 3404–3410, ISBN: 0-7803-7272-7.

[83] H. Zhu, J. Yu, A. Gupta, *et al.*, "The ingredients of real-world robotic reinforcement learning," *arXiv preprint arXiv:2004.12570*, 2020.

[84] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," *arXiv preprint arXiv:1205.4839*, 2012.

[85] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3, pp. 293–321, 1992, Publisher: Springer, ISSN: 1573-0565.

[86] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[87] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017, Publisher: National Acad Sciences, ISSN: 0027-8424.

[88] Z. Wang, V. Bapst, N. Heess, *et al.*, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[89] S. Zhang and R. S. Sutton, *A Deeper Look at Experience Replay*, arXiv:1712.01275 [cs], Apr. 2018. DOI: `10.48550/arXiv.1712.01275`.

[90] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," PMLR, 2014, pp. 387–395.

[91] Y. Chen, "Brain-Inspired Synaptic Resistor Circuits for Self-Programming Intelligent Systems," *Advanced Intelligent Systems*, vol. 3, no. 5, p. 2 000 219, 2021, Publisher: Wiley Online Library, ISSN: 2640-4567.

[92] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016, Publisher: Nature Publishing Group, ISSN: 1476-4687.

[93] D. Hu, Z. Pei, J. Shi, and Z. Tang, "Design, Modeling and Control of a Novel Morphing Quadrotor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8013–8020, Oct. 2021, Conference Name: IEEE Robotics and Automation Letters, ISSN: 2377-3766. DOI: `10.1109/LRA.2021.3098302`.

[94] D. Xu, Z. Hui, Y. Liu, and G. Chen, "Morphing control of a new bionic morphing UAV with deep reinforcement learning," *Aerospace science and technology*, vol. 92, pp. 232–243, 2019, Publisher: Elsevier, ISSN: 1270-9638.

[95] A. Lampton, A. Niksch, and J. Valasek, "Reinforcement learning of a morphing airfoil-policy and discrete learning analysis," *Journal of Aerospace Computing, Information, and Communication*, vol. 7, no. 8, pp. 241–260, 2010, ISSN: 1542-9423.

[96] J. Valasek, M. D. Tandale, and J. Rong, "A Reinforcement Learning - Adaptive Control Architecture for Morphing," en, *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 4, pp. 174–195, Apr. 2005, ISSN: 1542-9423. DOI: `10.2514/1.11388`.

[97] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997, Publisher: Elsevier, ISSN: 0169-7439.

[98] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107 398, Apr. 2021, ISSN: 0888-3270. DOI: `10.1016/j.ymssp.2020.107398`.

[99] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997, Publisher: MIT Press, ISSN: 0899-7667.

[100] P. Tabor, *Ppo in pytorch*, 2020. [Online]. Available: `https://github.com/philtabor/Youtube-Code-Repository/tree/master/ReinforcementLearning/PolicyGradient/PPO/torch`.

[101] B. Xu, N. Wang, T. Chen, and M. Li, *Empirical Evaluation of Rectified Activations in Convolutional Network*, arXiv:1505.00853 [cs, stat], Nov. 2015. DOI: `10.48550/arXiv.1505.00853`.

[102] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018, Publisher: Elsevier, ISSN: 1367-5788.

[103] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[104] C. Edwards and I. Postlethwaite, "Anti-windup and bumpless-transfer schemes," *Automatica*, vol. 34, no. 2, pp. 199–210, 1998, Publisher: Elsevier, ISSN: 0005-1098.

[105] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, vol. 299, p. 103 535, 2021, Publisher: Elsevier, ISSN: 0004-3702.

[106] D. Dewey, "Reinforcement learning and the reward engineering principle," 2014.

[107] R. G. Cook, R. Palacios, and P. Goulart, "Robust gust alleviation and stabilization of very flexible aircraft," *AIAA journal*, vol. 51, no. 2, pp. 330–340, 2013, Publisher: American Institute of Aeronautics and Astronautics, ISSN: 0001-1452.

[108] E. A. Hufstedler and P. Chatelain, "Loads Alleviation on an Airfoil via Reinforcement Learning," 2019, p. 0404.

[109] D. Nathan, A. Deo, K. Haughn, *et al.*, "Si-based self-programming neuromorphic integrated circuits for intelligent morphing wings," *Journal of Composite Materials*, p. 00 219 983 221 134 929, 2022, Publisher: SAGE Publications Sage UK: London, England, ISSN: 0021-9983.

[110] L. Geng, Y. F. Zhang, J. J. Wang, J. Y. H. Fuh, and S. H. Teo, "Mission planning of autonomous UAVs for urban surveillance with evolutionary algorithms," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, ISSN: 1948-3457, Jun. 2013, pp. 828–833. DOI: `10.1109/ICCA.2013.6564992`.

[111] M. A. Ma'sum, M. K. Arrofi, G. Jati, *et al.*, "Simulation of intelligent Unmanned Aerial Vehicle (UAV) For military surveillance," in *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Sep. 2013, pp. 161–166. DOI: `10.1109/ICACSIS.2013.6761569`.

[112] J. Saunders, S. Saeedi, and W. Li, *Autonomous Aerial Delivery Vehicles, a Survey of Techniques on how Aerial Package Delivery is Achieved*, arXiv:2110.02429 [cs, eess], Jul. 2022. DOI: `10.48550/arXiv.2110.02429`.

[113] A. Li, M. Hansen, and B. Zou, "Traffic management and resource allocation for UAV-based parcel delivery in low-altitude urban space," en, *Transportation Research Part C: Emerging Technologies*, vol. 143, p. 103 808, Oct. 2022, ISSN: 0968-090X. DOI: `10.1016/j.trc.2022.103808`.

[114] A. J. Dutt, "Wind flow in an urban environment," en, *Environmental Monitoring and Assessment*, vol. 19, no. 1, pp. 495–506, Oct. 1991, ISSN: 1573-2959. DOI: 10.1007/BF00401336.

[115] D. Hertwig, H. L. Gough, S. Grimmond, *et al.*, "Wake Characteristics of Tall Buildings in a Realistic Urban Canopy," en, *Boundary-Layer Meteorology*, vol. 172, no. 2, pp. 239–270, Aug. 2019, ISSN: 1573-1472. DOI: 10.1007/s10546-019-00450-7.

[116] J. C. Hunsaker and E. B. Wilson, *Report on Behavior of Aeroplanes in Gusts*, NTRS Author Affiliations: NTRS Report/Patent Number: NACA-TR-1 NTRS Document ID: 19930091026 NTRS Research Center: Legacy CDMS (CDMS), Jan. 1917. [Online]. Available: https://ntrs.nasa.gov/citations/19930091026 (visited on 10/04/2022).

[117] Z. Wu, Y. Cao, and M. Ismail, "Gust loads on aircraft," en, *The Aeronautical Journal*, vol. 123, no. 1266, pp. 1216–1274, Aug. 2019, Publisher: Cambridge University Press, ISSN: 0001-9240, 2059-6464. DOI: 10.1017/aer.2019.48.

[118] V. H. Cheng and B. Sridhar, "Considerations for Automated Nap-of-the-Earth Rotorcraft Flight," in *1988 American Control Conference*, Jun. 1988, pp. 967–976. DOI: 10.23919/ACC.1988.4789863.

[119] C. Harvey, L. L. Gamble, C. R. Bolander, D. F. Hunsaker, J. J. Joo, and D. J. Inman, "A review of avian-inspired morphing for UAV flight control," en, *Progress in Aerospace Sciences*, vol. 132, p. 100 825, Jul. 2022, ISSN: 0376-0421. DOI: 10.1016/j.paerosci.2022.100825.

[120] J. E. Pagel, C. M. Anderson, D. A. Bell, *et al.*, "Peregrine Falcons: The Neighbors Upstairs," en, in *Urban Raptors: Ecology and Conservation of Birds of Prey in Cities*, C. W. Boal and C. R. Dykstra, Eds., Washington, DC: Island Press/Center for Resource Economics, 2018, pp. 180–195, ISBN: 978-1-61091-841-1. DOI: 10.5822/978-1-61091-841-1_13.

[121] J. A. Cheney, J. P. J. Stevenson, N. E. Durston, *et al.*, "Bird wings act as a suspension system that rejects gusts," *Proceedings of the Royal Society B: Biological Sciences*, vol. 287, no. 1937, p. 20 201 748, Oct. 2020, Publisher: Royal Society. DOI: 10.1098/rspb.2020.1748.

[122] J. Pilakkadan, R. Ajaj, and M. Ammozgar, "LOAD ALLEVIATION USING THE SPANWISE MORPHING TRAILING EDGE CONCEPT," in *2022 International Forum on Aeroelastic and Structural Dynamics*, Jun. 2022.

[123] J. Xu, J. Lou, Y. Yang, T. Chen, H. Chen, and Y. Cui, "Hysteresis modeling and feedforward compensation of a flexible structure actuated by macro fiber composites using bias bipolar Prandtl-Ishlinskii model," en, *Journal of Intelligent Material Systems and Structures*, vol. 32, no. 18-19, pp. 2325–2337, Nov. 2021, Publisher: SAGE Publications Ltd STM, ISSN: 1045-389X. DOI: 10.1177/1045389X21995881.

[124] K. S. Thapa Magar, A. M. Pankonien, G. W. Reich, and R. Beblo, "Optimal Control Framework for Gust Load Alleviation using Real time Aerodynamic Force Prediction from Artificial Hair Sensor Array," en, in *2018 AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2018, ISBN: 978-1-62410-526-5. DOI: 10.2514/6.2018-0850.

[125] H. .-. Giesseler, M. Kopf, P. Varutti, T. Faulwasser, and R. Findeisen, "Model Predictive Control for Gust Load Alleviation," en, *IFAC Proceedings Volumes*, 4th IFAC Conference on Nonlinear Model Predictive Control, vol. 45, no. 17, pp. 27–32, Jan. 2012, ISSN: 1474-6670. DOI: 10.3182/20120823-5-NL-3013.00049.

[126] Y. Hamada, "Aircraft gust alleviation using discrete-time preview controller with prior gust information," Sep. 2013, pp. 1907–1912.

[127] J. Hansen, "Control Allocation of Flexible Aircraft for Load Alleviation," en_US, Thesis, 2021. DOI: 10.7302/1456.

[128] N. Mandel, M. Milford, and F. Gonzalez, "A Method for Evaluating and Selecting Suitable Hardware for Deployment of Embedded System on UAVs," en, *Sensors*, vol. 20, no. 16, p. 4420, Jan. 2020, Number: 16 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s20164420.

[129] M. KleinHeerenbrink, L. A. France, C. H. Brighton, and G. K. Taylor, "Optimization of avian perching manoeuvres," en, *Nature*, vol. 607, no. 7917, pp. 91–96, Jul. 2022, Number: 7917 Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/s41586-022-04861-4.

[130] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of Go without human knowledge," en, *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, Number: 7676 Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/nature24270.

[131] A. Beck and M. Kurz, "A perspective on machine learning methods in turbulence modeling," en, *GAMM-Mitteilungen*, vol. 44, no. 1, e202100002, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/gamm.202100002, ISSN: 1522-2608. DOI: 10.1002/gamm.202100002.

[132] K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence Modeling in the Age of Data," *Annual Review of Fluid Mechanics*, vol. 51, no. 1, pp. 357–377, 2019, _eprint: https://doi.org/10.1146/annurev-fluid-010518-040547. DOI: 10.1146/annurev-fluid-010518-040547.

[133] Z. Ren, W. Fu, and J. Yan, "Gust Perturbation Alleviation Control of Small Unmanned Aerial Vehicle Based on Pressure Sensor," en, *International Journal of Aerospace Engineering*, vol. 2018, e7259363, Jul. 2018, Publisher: Hindawi, ISSN: 1687-5966. DOI: 10.1155/2018/7259363.

[134] T. Zou and L. Zhou, "Mechanical property analysis and experimental demonstration of zero Poisson's ratio mixed cruciform honeycomb," en, *Materials Research Express*, vol. 4, no. 4, p. 045 702, Apr. 2017, Publisher: IOP Publishing, ISSN: 2053-1591. DOI: 10.1088/2053-1591/aa675c.

[135] M. S. Kuester, A. Borgoltz, and W. J. Devenport, "Pressure Tap Effects on the Lift Measurement of an Airfoil Section," en, in *32nd AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, Washington, D.C.: American Institute of Aeronautics and Astronautics, Jun. 2016, ISBN: 978-1-62410-438-1. DOI: 10.2514/6.2016-3654.

[136] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. DOI: 10.48550/arXiv.1412.6980.

[137] R. V. Rhode and E. E. Lundquist, *Preliminary Study of Applied Load Factors in Bumpy Air*, en. National Advisory Committee for Aeronautics, 1931, Google-Books-ID: MmTkmfSgbPUC.

[138] C. Badrya, A. R. Jones, and J. D. Baeder, "Unsteady Aerodynamic Response of a Flat Plate Encountering Large-Amplitude Sharp-Edged Gust," *AIAA Journal*, vol. 60, no. 3, pp. 1549–1564, 2022, Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.J060683, ISSN: 0001-1452. DOI: 10.2514/1.J060683.

[139] Y. Zhou, Z. Wu, and C. Yang, "Gust Alleviation and Wind Tunnel Test by Using Combined Feedforward Control and Feedback Control," en, *Aerospace*, vol. 9, no. 4, p. 225, Apr. 2022, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2226-4310. DOI: 10.3390/aerospace9040225.