Special Issue: AFOSR

JOURNAL OF COMPOSITE MATERIALS

Journal of Composite Materials
2023, Vol. 57(4) 721–736

DOI: 10.1177/00219983221137644
journals.sagepub.com/home/jcm
SAGE

Deep reinforcement learning achieves multifunctional morphing airfoil control

Kevin P T Haughn[1], Lawren L Gamble[2] and Daniel J Inman[1]

Abstract

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 uncrewed aerial vehicles (UAVs) capable of autonomously responding to flow conditions, this work examines a multifunctional morphing airfoil's ability to follow commands in various flows. We integrated an airfoil with a morphing trailing edge consisting of an antagonistic pair of macro fiber composites (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 we used a deep reinforcement learning algorithm, Proximal Policy Optimization, 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. We compared the performance of our neural controllers to one using traditional position-derivative feedback control methods. Our 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.

Keywords

Morphing aircraft, multifunctional materials, reinforcement learning, smart composites, autonomous control

Introduction

Uncrewed aerial vehicles (UAVs) 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 macro fiber composites (MFCs), have been used to achieve the desired shape changes.[4,5] 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.[6] Piezoelectric actuators are also well known for their capabilities to produce high force-output and a high-speed actuation response. 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 the performance benefits attributed to traditional piezoelectric actuators.[7,8] 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. This behavior is attractive for camber morphing airfoil applications and has spurred a large subset of research in the field of morphing aircraft.[1,9]

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

[1]Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, USA
[2]Exponent, Menlo Park, CA, USA

Corresponding author:
Kevin P T Haughn, Department of Aerospace Engineering, University of Michigan, 1320 Beal Ave, Ann Arbor, MI 48108, USA.
Email: kevpatha@umich.edu

ability to increase control authority and improve efficiency.[10–13] MFC 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 and their rapid actuation response are advantageous in UAV applications. Reductions in aircraft weight lead to greater fuel efficiency and rapid actuation allows for greater maneuverability. MFC-driven camber morphing has been applied to several UAV control problems including localized optimization for adverse aerodynamic disturbance and stall recovery, as well as improved efficiency and control effectiveness in roll and pitch for a rudderless aircraft.[14–16] Finally, pitch and yaw control effectiveness and yaw stability were also demonstrated in an avian-inspired rudderless UAV with a camber morphing MFC tail actuator.[17]

Although MFCs have demonstrated potential for camber morphing applications, they have 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. 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 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.[18] In contrast, deep reinforcement learning (DRL) is well-equipped to manage nonlinear control relationships and may be a promising alternative.

Reinforcement learning (RL) is a means of autonomously achieving control in a manner analogous to that seen in biological systems. Like biological learning, trial and error is used in conjunction with a reward system 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.[19] If the state space is large and best represented as continuous, function approximation is used to reduce memory requirements. Artificial neural networks (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 remarkable success in many simulated and game based environments.[20–24] 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.[25–28]

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.[29,30] 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.[31–33] Although these shape changes are achieved using traditional methods, such as servos and motors, some have implemented RL in smart material based morphing simulations.[34,35] One case presented by Goecks et al. implemented deep deterministic policy gradient (DDPG) with a shape memory alloy (SMA) actuated airfoil.[36] 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 the current literature around RL in morphing UAV environments, is performed entirely in simulation.

Through a sim-to-real policy transfer we present the first successful application of RL for an MFC actuated morphing system in a physical hardware environment (Figure 1). In this research, we found that DRL countered the hysteretic behaviors present in our MFC morphing airfoil to effectively control trailing edge tip deflection in the physical hardware. In the simulated environment, we 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 were compared on physical hardware in situations where "true" state information was supplied through an external Keyence profilometer, and with on-board sensing where state information was estimated through a piezoelectric flex sensor signal. We 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. We 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.

We found that the learned controllers, specifically the MO controller, outperformed the traditional PD feedback method. This was especially true for control metrics

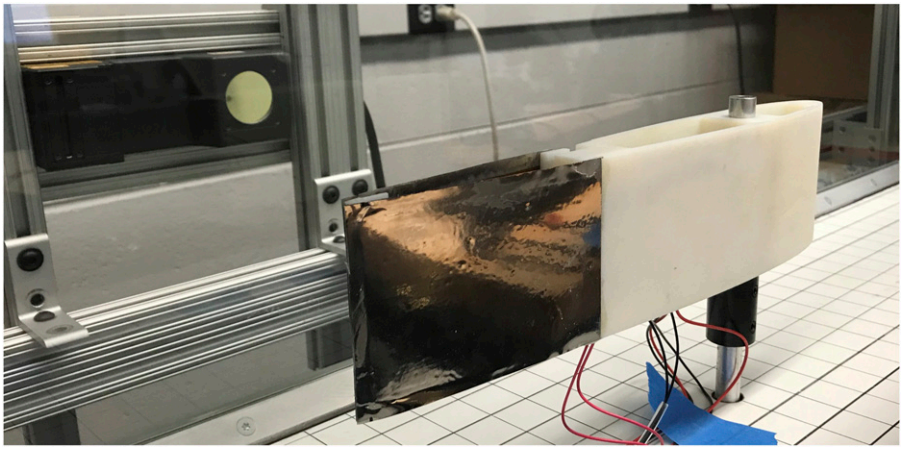

Figure 1. Image of the morphing airfoil system within the 1′×1′ (30 cm × 30 cm) wind tunnel. Also visible in the upper left corner is the Keyence 2D profilometer used to measure the true deflection of the trailing edge.

considering speed and accuracy. From this, we 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.

Methods

Morphing airfoil system design

We assessed the performances of the learned controllers using the camber morphing airfoil design developed by Pankonien et al.[14]Our baseline geometry for the morphing airfoil was a NACA0012 airfoil with a 310 mm chord (Figure 2). We 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.[14] We assembled the morphing trailing edge using two MFC unimorphs, one on either side of the airfoil. We 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, we bonded each flex sensor to a 0.025 mm thick strip of stainless-steel shim as well. The flex sensors measured the internal displacement of the morphing trailing edge and were wired in a voltage divider configuration.

The control architecture for our morphing airfoil is shown in Figure 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 directed 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 and LSTM) was used to provide state information to one of the three controllers (PD, RL, and MO) to determine an output voltage signal. 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 forming the trailing end of the airfoil.

Data collection

Reinforcement learning, although a powerful tool, is time consuming due to its trial-and-error format. To refrain from subjecting our system to unnecessary wear and tear, we created a simulation in which we could experiment with RL methods to develop a sufficient controller. Since we 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 our system, any imperfections that occurred in manufacturing the composite wing provide a potential for variation in the behavior of our MFC actuators. To accomplish accurate modeling specific to our morphing airfoil's dynamics, we collected true deflection and flex sensor information from randomized sweeps through the action and state space of the morphing trailing edge.

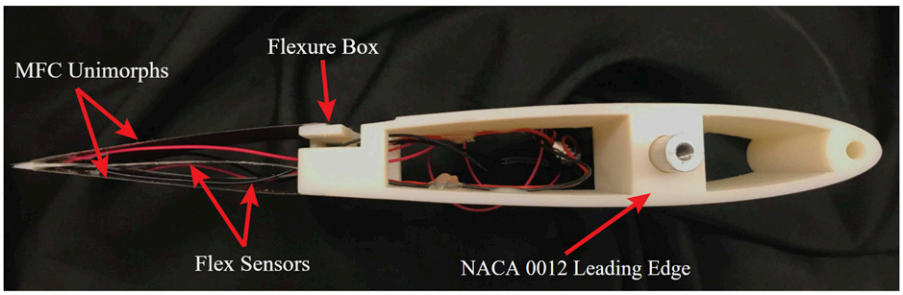

Figure 2. Image of the morphing airfoil system including two antagonistic MFC unimorphs, two piezoelectric flex sensors, the multi-material flexure box, and NACA 0012 leading edge.

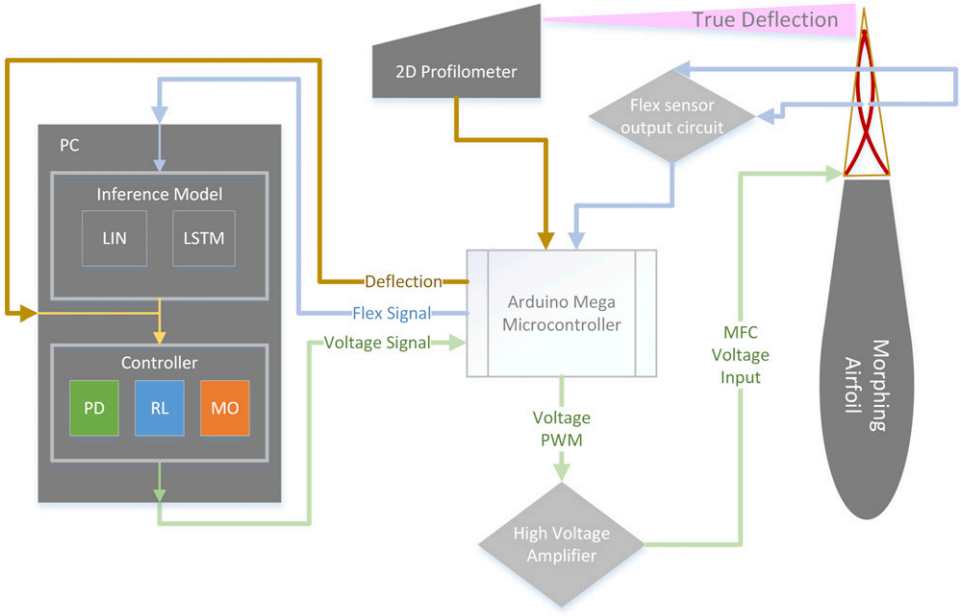

Figure 3. Data flow diagram for the morphing airfoil experiment. Deflection information was captured through the 2D profilometer and two piezoelectric flex sensors and provided to a laptop for data collection and controller decision making.

During data collection, we ensured sufficient coverage of the state-action space by applying a range of voltage changes from a randomized series of initial voltages. For our situation, the randomization of our 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 our case, a voltage signal of zero represented the largest negative supplied voltage and a signal of 100 was the largest positive supplied voltage. Thus, 50 was a neutral supply of zero 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 s, supplying the new voltage for 5 seconds in total, after which the initial voltage was again supplied for another 5 seconds. This was repeated until the set of voltage change signals had been exhausted and then restarted for the next randomly selected initial voltage signal. This process was repeated 10 times for 10 different random seeds.

Modeling dynamics

Following data collection, we implemented three neural network structures to comparatively model the dynamics of our system, including a multi-step dense network, a one-dimensional convolutional neural network (CNN), and a long short-term memory (LSTM) network.[37–39] The input for each of the models included state information over the 10 previous timesteps. Each timestep state observation consisted of the current deflection value and the current and previous voltage signal. From this the models predicted the

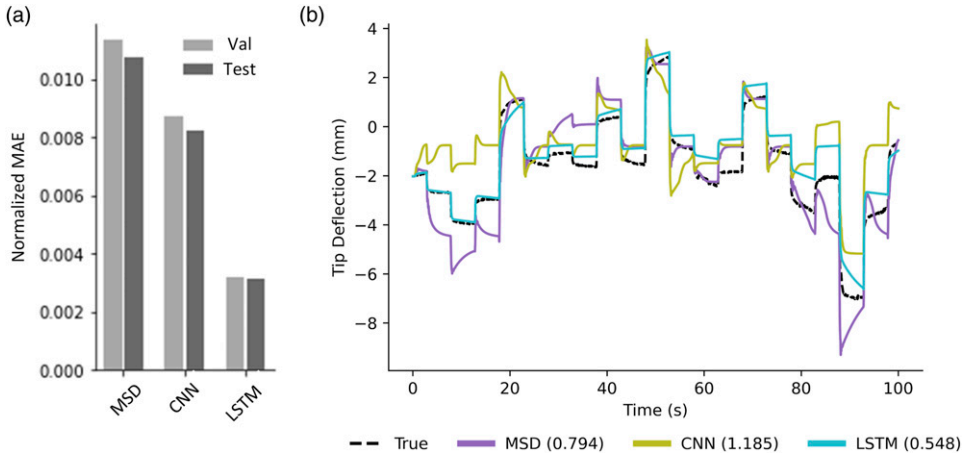

Figure 4. Comparison of learned dynamics models, including multi-step dense (MSD), convolutional neural network (CNN), and long short-term memory (LSTM). The normalized mean absolute error (MAE) earned during training validation (Val) and testing (test) is presented in a), and (b) illustrates the performance of the different models over a 100 s section of data collection when given the true state at time 0. Mean absolute error, in terms of millimeters, over the 100 s for each model is shown in parentheses.

deflection for the next immediate timestep. Of the 10 datasets collected using different random seeds, the first nine were combined and split into an 8:2 ratio for training and validation, respectively. The 10th data set was not included in the training process and was used for testing. We found that the LSTM model achieved the lowest error in both validation and testing (Figure 4(a)). This is visualized by the LSTM model's ability to accurately represent the system's dynamics for a 100 s example section of the collected testing data (Figure 4(b)). 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.

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.[19] 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.[19] It is for these reasons that we chose to use the model free on-policy algorithm, proximal policy optimization (PPO).[40]

We developed our learned controllers using PPO in a simulation informed by the 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.[41] 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, which learns the policy π. Given that there are two neural networks to update in PPO, there are also two loss functions combined for gradient-based optimization, for which we used Adam optimizer with a 3×10^{-4} learning rate.[42] The actor and critic loss functions are defined as follows

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

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

Where A_t is the advantage at time t described by

$$\widehat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{3}$$

With

$$\delta_t = R_t + \gamma V(s_{t+1}) - V(s_t), \tag{4}$$

where R_t represents the reward earned at time t, and $r(\boldsymbol{\theta})$ is the ratio between the new and old policy for the current action, a_t and state s_t

$$r_t(\boldsymbol{\theta}) = \pi_\theta(a_t|s_t)/\pi_\theta^{old}(a_t|s_t) \tag{5}$$

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 $\boldsymbol{\gamma}$ is a discount factor used to gradually degrade the impact of future state *values* on the current state *value*, hence

Figure 5. 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 action, and the critic outputs a single value representing the estimated long term expected reward of the current state.

emphasizing the impact of more immediate 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, λ, and ϵ, were set to equal 0.99, 0.95, and 0.2 respectively, as suggested by Schulman when first presenting PPO.[40] Our implementation of PPO was drawn from an opensource example in Pytorch.[43]

The network structures for both the actor and critic are presented in Figure 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. We used a 1D convolutional neural network (CNN) for the initial layer of both the critic and actor networks in our PPO algorithms, each followed by three fully connected layers with rectified linear unit (ReLU) activation functions.[38,44] This input layer was given the 10 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, we 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.

We 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, we noticed room for improvement regarding the overshoot experienced. In the second learned controller we 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, we augmented the original reward scheme for the second learned controller to include an extra penalty equal to 10 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, we 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.

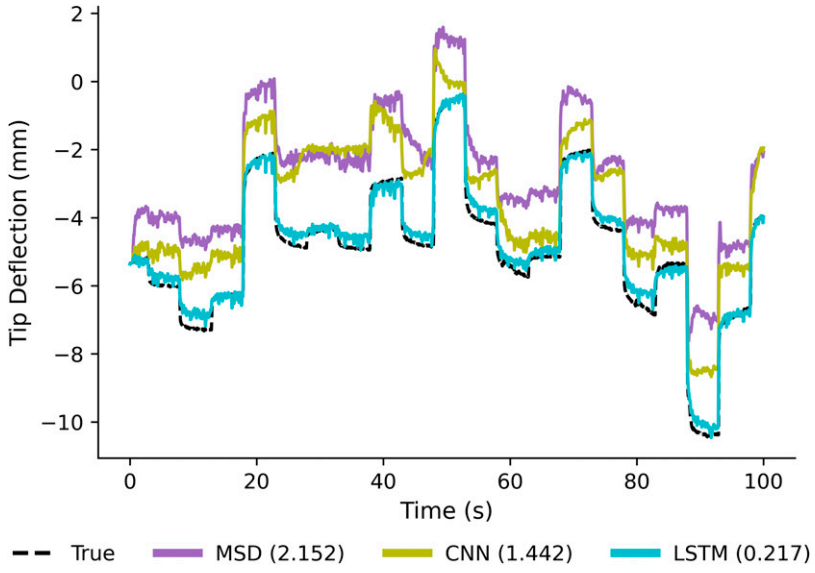

Figure 6. This plot compares the flex inference models using three different neural network structures, including multi-step dense (MSD), convolutional neural networks (CNN), and long short-term memory (LSTM), with the mean absolute error shown in parentheses. The LSTM inference model provided the most accurate state approximation.

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 our case piezoelectric flex sensors, that give imperfect state measurements. Therefore, to provide accurate state observation, we used the information gathered in the data collection section to model the relationship between the piezoelectric flex sensor signals and the true deflection of the MFC trailing edge. We used two methods for this model, 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, we 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, we removed the voltage values from the model input and used a time series of the 10 most recent timesteps to infer the current deflection. Each timestep included only the current sensor reading and the previous deflection estimate. Once again, we found LSTM networks provided the most accurate prediction when compared to the other neural network structures (Figure 6).

Experiment

Using the same data flow strategy as described in Figure 3, for each test we 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.[44] For this reason, we use repetition to empirically show what response and overall performance we can expect from these controllers. Therefore, we 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, we used the Ziegler-Nichols open loop method to tune a proportional-integral-derivative (PID) controller; however, the controller experienced high overshoot and integral windup.[46] To mitigate this, we 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.[47] We 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

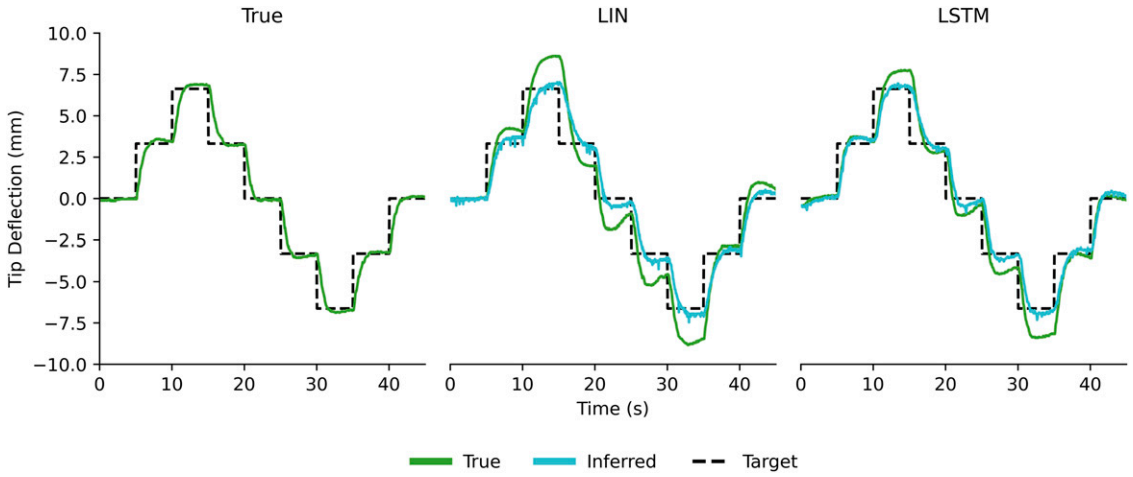

Figure 7. Controller performance changed with state observation accuracy. The left plot shows the performance of the traditional PD controller when given true deflection information. The following two plots show the true deflections and inferred deflections experienced by the PD controller when provided the linear (LIN) and LSTM inference models.

when paired with the flex sensor inference methods, providing a fair and interesting baseline for comparing to the learned controllers. Figure 7 presents the step response cycle for the 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 model. Increasing in complexity and accuracy from the linear model to the LSTM model, we 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.

We compared controller performances using several metrics, including three traditional controller step response metrics: rise time, settling time, and overshoot. We measured the rise time and settling time from the beginning of a step response until the true 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. We included an additional metric common to RL: total earned reward. This was the value optimized by the RL algorithm. For consistency, we chose the simpler reward of squared error between the true position and the goal position as our metric for each of the controllers. As an error-based metric, we used reward as an indicator for the overall accuracy achieved in a test run.

MFCs are known to perform differently under mechanical loads, but all the data used for training the controller and state estimation models were collected without consideration for aerodynamic loading.[14] Therefore, to seriously consider these controllers for autonomous UAV flight, we performed the same step response tests for a variety of flow speeds. For this purpose, we repeated the testing process in a 1′×1’ (30 cm × 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’ performance 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 like 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.

Results

Unloaded environment

For our initial experiments we compared our 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

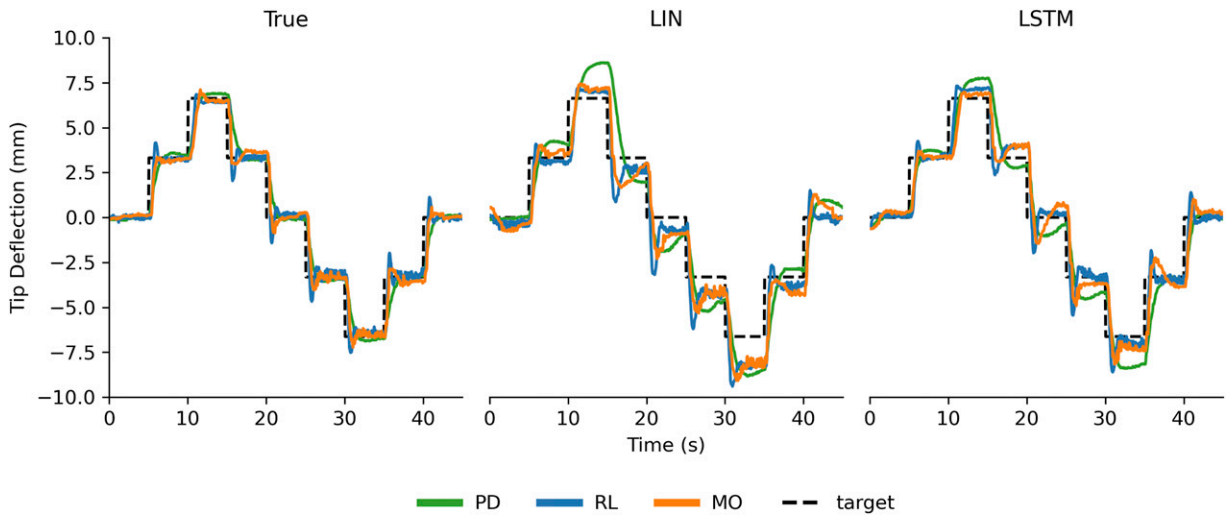

Figure 8. This plot shows the step responses of each controller (PD, RL, and MO) in an unloaded environment when supplied true deflection information as well as state observations provided by the two piezoelectric flex sensor inference models (LIN and LSTM).

provided true state information and using each of the three inference methods are visualized in Figure 8. We found that each of the controllers were able to track the target value with precision when provided accurate state observations 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, we investigated our three metrics (rise time, overshoot, settling time) as well as the total earned reward (Figure 9). This figure visualizes the eight step responses in each of the five tests for the three controllers. From this we can see each controller's strengths and weaknesses. Of the three controllers, PD (green lines) consistently earned the lowest reward. This was expected because the two learned controllers directly use this reward, or at least a 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 (Figure 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 previous controllers, achieving a medium reward and rising times that were faster than the PD controller but typically not as fast as the RL controller. The MO controller improved over the RL controller in its overshoot percentage, in some cases experiencing lower overshoot values than the PD controller. The MO controller's performance highlights 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 our confidence in these more complex controllers, it is not realistic to expect an airfoil controller to have perfect state observations during flight. Thus, we conducted the same step response tests for both state inference methods (LIN and LSTM). We found a substantial impact from the state observation accuracy for each of the controllers (Figure 8). Interestingly, although the different controllers used 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 10 considers each of the performance metrics and presents the average performance difference between a given learned controller-inference model combination and the PD controller when using the same state estimation method. The error bars represent 95% confidence intervals to illustrate the significance of the difference. Except for overshoot, both the learned controllers outperformed the PD controller on average. This difference was significant in all cases when considering rise time, and for five 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

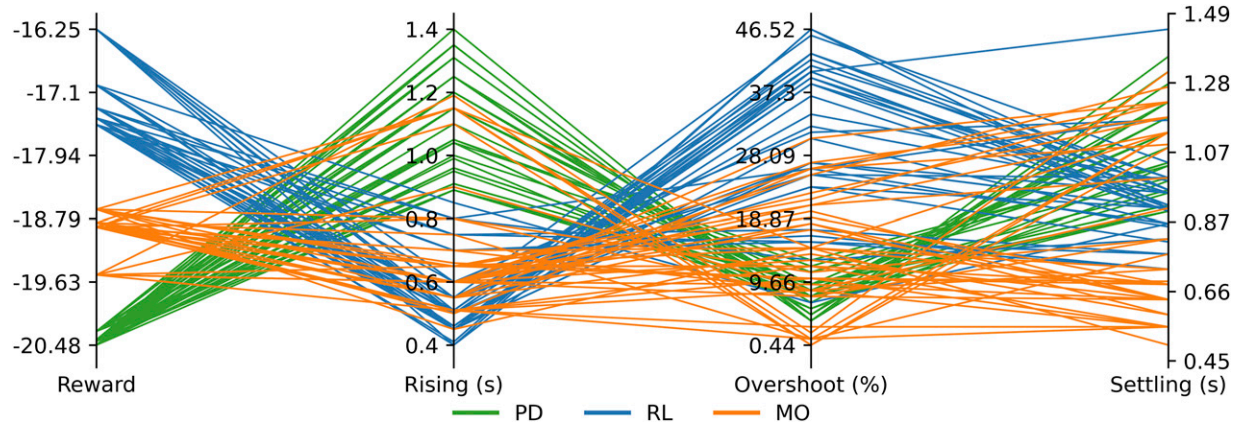

Figure 9. Comprehensive presentation of performance metrics achieved by each controller given true state observations. This plot includes every step response performed in testing and can therefore be used to determine expected trends for each controller's performance.

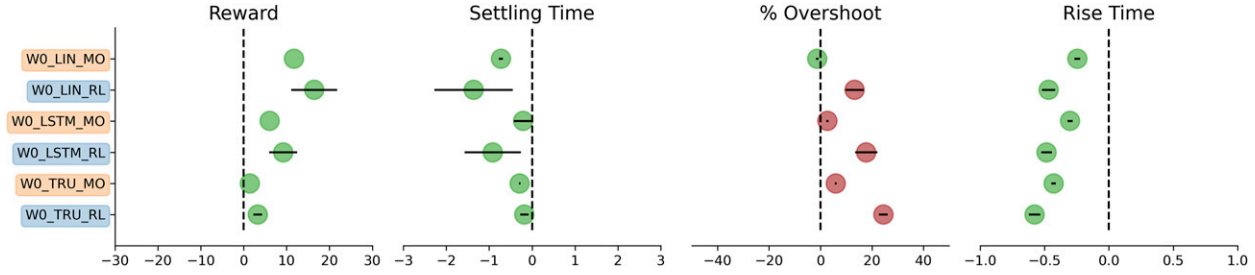

Figure 10. Direct comparison of average performance achieved by learned controllers in the unloaded environment for each metric to those achieved by the PD controller given the same state observation method. Comparisons that trend in the favor of the learned controller are in light green, and those in 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.

towards achieving a lower average overshoot than the PD controller.

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 11. These loaded responses with true state information indicate that the controllers respond quickly and accurately to the various changes in goal deflection, like that seen without loading. We see similar trends to those seen in the unloaded testing when comparing the performance metrics of the controllers in a 10 m/s airflow environment (Figure 12). Interestingly, the learned controllers outperformed their unloaded condition, specifically the MO controller with regards to overshoot.

The plots in Figure 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 our results, we examined the average absolute error between the true state and model observed state (Figure 13). We found a substantial improvement in the linear model accuracy from the unloaded 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, our main objective was to compare the performance of our controllers developed through RL to that of the traditional PD controller. These direct comparisons are presented in Figure 14 for all three flow speeds. We first noticed that the learned controllers no longer completely dominated the settling time and reward metrics. The most complex state estimation method (LSTM) was used for all but one of the tests in which the PD controller outperformed the compared learned controller in

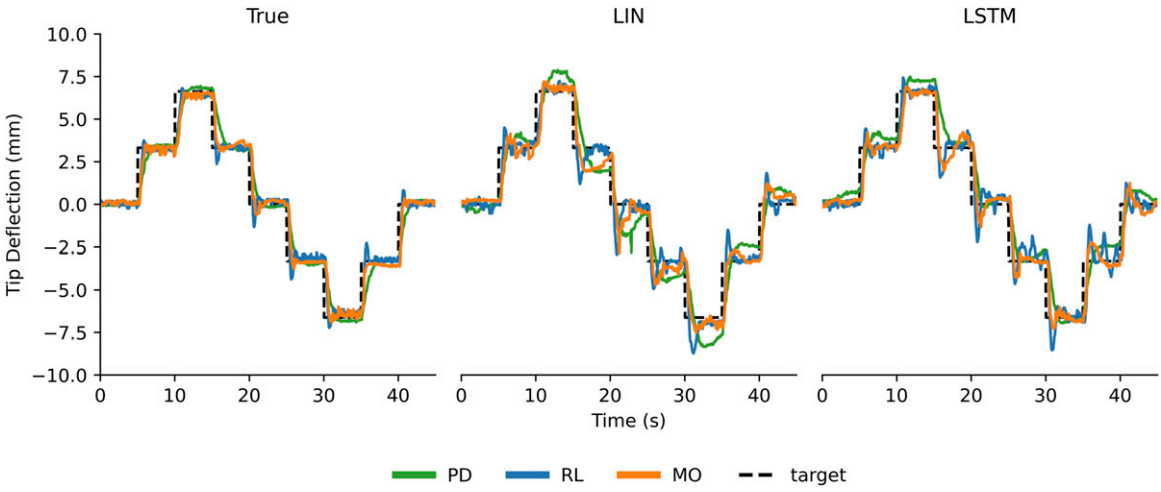

Figure 11. This plot shows the step responses of each controller (PD, RL, and MO) in a 10 m/s aerodynamically loaded environment when supplied true deflection information as well as state observations provided by the two piezoelectric flex sensor inference models (LIN and LSTM).

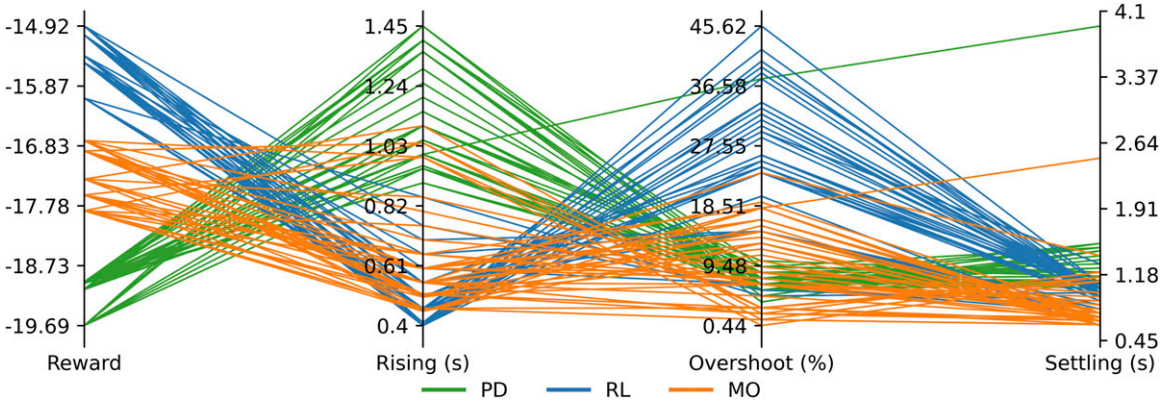

Figure 12. Comprehensive presentation of performance metrics achieved by each controller given true state observations in a 10 m/s airflow environment. This plot includes every step response performed in testing and can therefore be used to determine expected trends for each controller's performance.

at least one of these two metrics. 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 s. This difference is less than the timestep size of 0.05 s, and therefore was not significant enough to consider a trend in either direction. Additionally, of the 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 is 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. On average, when combined with the linear inference model, the MO controller outperformed the PD controller in 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 we can look at Figure 15, which presents the average performance metric values for each controller, inference model, and flow speed.

Assuming perfect state observations, our previous comments are further validated. The learned controllers excel in achieving fast and accurate control, according to rising time, settling time, and reward. The PD controller still achieved the lowest overshoot of the controllers on average,

but the MO controller produced only 2.13% greater overshoot on average. This is a great improvement over the additional 10% of average overshoot produced by the RL controller than the PD controller. We found a continuation of this trend when regarding imperfect state observations.

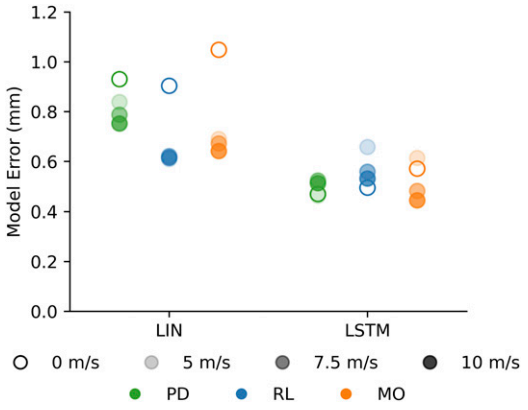

Figure 13. Average error for all controller-observation method combinations at all four flow speeds. The noticeable improvement in linear inference error is due to the normalization adjustment made between unloaded and loaded environment testing.

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. We 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 three 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.

Discussion

These results have given us insight for future controller design in the pursuit of fly-by-feel solutions for morphing composite wings. We have validated the use of DRL

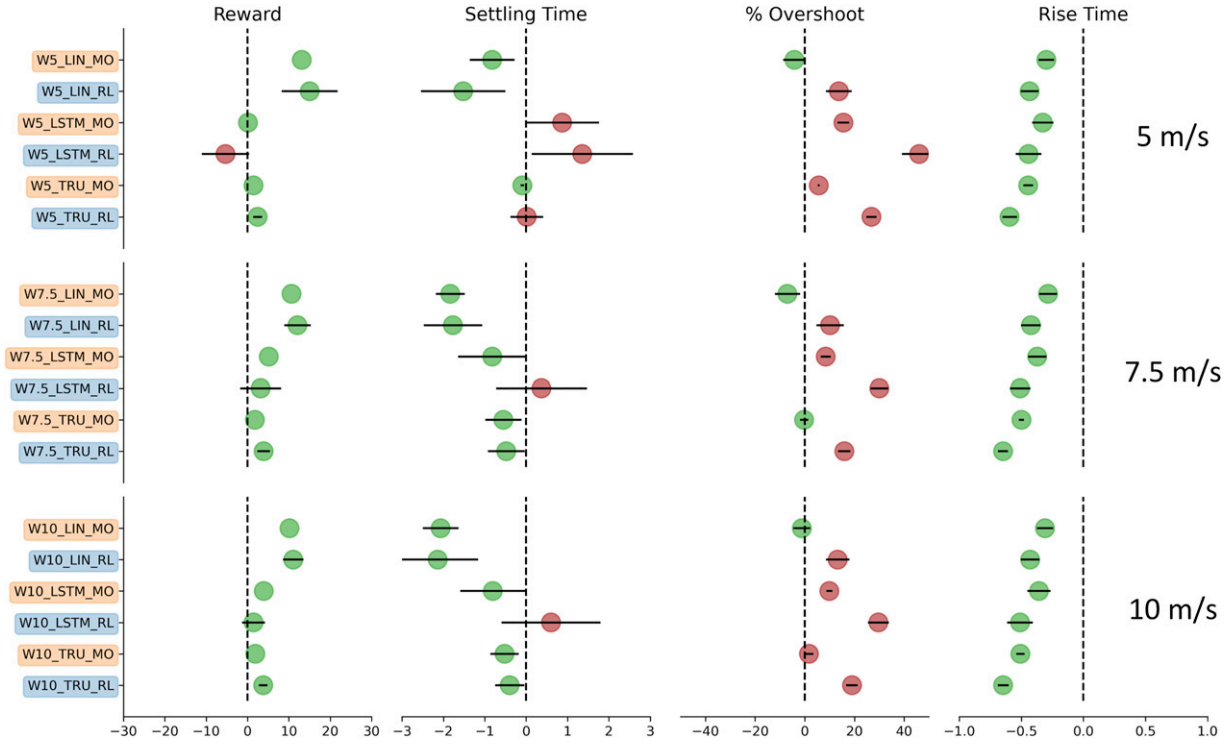

Figure 14. Average performances achieved by learned controllers in the aerodynamically loaded environments for each metric are directly compared to those achieved by the PD controller given the same state observation method. Comparisons that trend in the favor of the learned controller are in light green, 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.

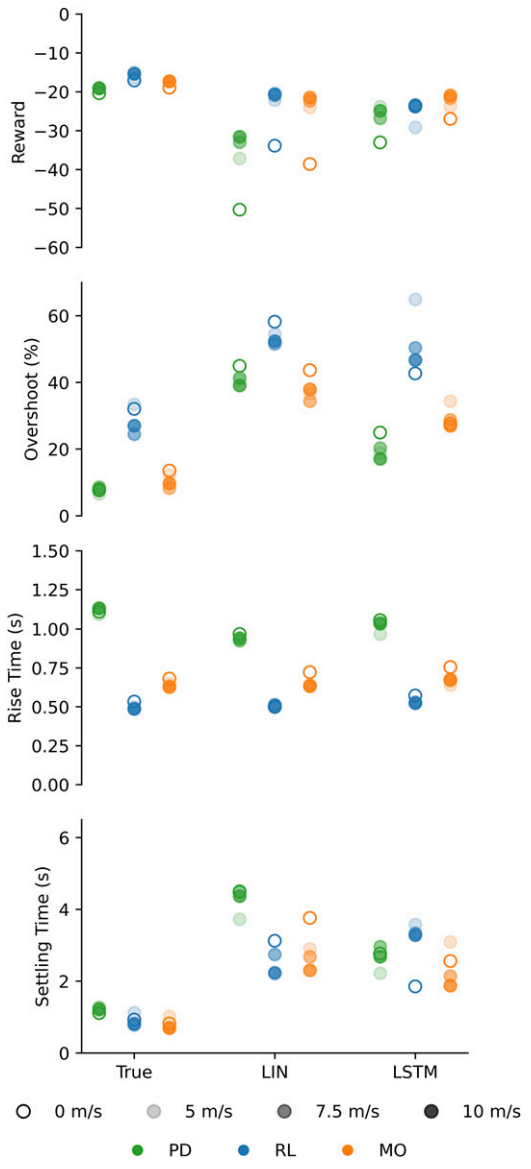

Figure 15. A summary of the average performances for each controller, under all flow speed conditions, and with all three observation methods.

controllers in our physical multifunctional morphing airfoil environment. Furthermore, we found the learned controllers were 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 reducing the overshoot. This improved 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, we suspect the learned controllers were internally accounting for the hysteretic behavior of the system. Since the learned controllers used 1D convolutions of the 10 most recent timesteps of state information, RL and MO may have 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 we further optimize our 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.[48] This was the philosophy we followed when designing our first controller (RL), and found it created a strong controller with emphasis on speed and accuracy. However, after seeing it struggle to mitigate overshoot, we added a rule to our reward scheme. This amendment created a controller with an impressive balance between speed, accuracy, and overshoot (MO). It may be argued that, for our purposes, mitigating overshoot falls within the bounds of the goal that must be characterized by our 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 behaviors becomes both more important and more difficult".[49] Regardless of philosophy, we evaluated one reward function augmentation, and in doing so developed a highly effective controller for our 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 our 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.[15,50,51] Additionally, in this work we validated our learned controllers in hardware, but all training was performed in simulation. If we 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.[28] 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.[52] The next missing link to create fully autonomous composite wing systems for morphing is understanding how to integrate computing chips into a composite material. We 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.

Conclusion

In this work, we presented the first successful application of RL to a physical MFC actuated system, outperforming traditional PD control methods. To achieve this, we 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, we 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 model and an LSTM network, were used in coordination with piezoelectric flex sensors for imperfect on-board state observation. Additionally, an external 2D profilometer was used for true state measurements. We first evaluated these controllers with the three state observation methods in an unloaded environment and then at three flow speeds. We used a tuned PD controller as the comparative baseline.

We 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.

Acknowledgements

The authors would like to thank Connor Stadler, an undergraduate student in the aerospace engineering department at the University of Michigan for his contribution of PD controller tuning.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is 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 grant number FA9550-16-1-0087, titled “Avian-Inspired Multifunctional Morphing Vehicles” monitored by Dr. BL Lee.

ORCID iD

Kevin P T Haughn https://orcid.org/0000-0002-8219-8983

References

1. Barbarino S, Bilgen O, Ajaj RM, et al. A review of morphing aircraft. *J Intelligent Material Systems Structures* 2011; 22(9): 823–877.
2. Jha AK and Kudva JN. Morphing aircraft concepts, classifications, and challenges. In *Smart structures and materials 2004: industrial and commercial applications of smart structures technologies*, Vol 5388. San Diego, CA, United States: International Society for Optics and Photonics, pp. 213–224, 2004.
3. Gomez JC and Garcia E. Morphing unmanned aerial vehicles. *Smart Mater Structures* 2011; 20(10): 103001.
4. Sun J, Guan Q, Liu Y, et al. Morphing aircraft based on smart materials and structures: a state-of-the-art review. *J Intell Material Systems Structures* 2016; 27(17): 2289–2312.
5. Pankonien AM. *Smart Material wing Morphing for unmanned aerial vehicles*. PhD Thesis, Ann Arbor, MI, United States: Michigan Publishing, 2015.
6. Haertling GH. Ferroelectric ceramics: history and technology. *J Am Ceram Soc* 1999; 82(4): 797–818.
7. High JW. *Method of fabricating NASA-standard macro-fiber composite piezoelectric actuators*. Hanover, MD, United States: National Aeronautics and Space Administration, Langley Research Center, 2003.
8. Wilkie WK, Bryant RG, High JW, et al. Low-cost piezocomposite actuator for structural control applications. In *Smart Structures Materials 2000: Industrial Commercial Applications Smart Structures Technologies*, Vol 3991. International Society for Optics and Photonics, pp. 323–334.
9. Bilgen O, Kochersberger KB, Inman DJ, et al. Novel, bidirectional, variable-camber airfoil via macro-fiber composite actuators. *J Aircraft* 2010; 47(1): 303–314.
10. Sanders B, Eastep F and Forster E. Aerodynamic and aeroelastic characteristics of wings with conformal control surfaces for morphing aircraft. *J Aircraft* 2003; 40(1): 94–99.
11. Woods BK, Bilgen O and Friswell MI. Wind tunnel testing of the fish bone active camber morphing concept. *J Intell Mater Syst Structures* 2014; 25(7): 772– 785.
12. Gilbert WW. Mission adaptive wing system for tactical aircraft. *J Aircraft* 1981; 18(7): 597–602.

13. Hetrick J, Osborn R, Kota S, et al. Flight testing of mission adaptive compliant wing. In 48th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference. p. 1709.
14. Pankonien A and Inman DJ. Experimental testing of spanwise morphing trailing edge concept. In *Active and passive smart structures and integrated systems*, Vol 8688. San Diego, CA, United StatesSPIE, 2013, p. 868815.
15. Gamble LL, Pankonien AM and Inman DJ. Stall recovery of a morphing wing via extended nonlinear lifting-line theory. *AIAA J* 2017; 55(9): 2956–2963.
16. Molinari G, Arrieta AF and Ermanni P. Planform, aerostructural and flight control optimization for tailless morphing aircraft. *J Intell Mater Syst Structures* 2018; 29(20): 3847–3872.
17. Gamble LL and Inman DJ. A tale of two tails: developing an avian inspired morphing actuator for yaw control and stability. *Bioinspiration & Biomimetics* 2018; 13(2): 026008.
18. Zhang C, Qiu J, Chen Y, et al. Modeling hysteresis and creep behavior of macrofiber composite–based piezoelectric bimorph actuator. *J Intell Mater Syst Structures* 2013; 24(3): 369–377.
19. Sutton RS and Barto AG. *Reinforcement learning: an introduction*. Cambridge, MA, United States: MIT press, 2018.
20. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:13125602 2013.
21. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015; 518(7540): 529–533.
22. Silver D, Huang A, Maddison CJ, et al. Mastering the game of go with deep neural networks and tree search. *Nature* 2016; 529(7587): 484–489.
23. Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 2018; 362(6419): 1140–1144.
24. Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:191206680 2019.
25. Dulac-Arnold G, Mankowitz D and Hester T. Challenges of real-world reinforcement learning. arXiv preprint arXiv: 190412901 2019.
26. Andrychowicz OM, Baker B, Chociej M, et al. Learning dexterous in-hand manipulation. *The Int J Robotics Res* 2020; 39(1): 3–20.
27. Tai L, Paolo G and Liu M. Virtual-to-real deep reinforcement learning: continuous control of mobile robots for mapless navigation. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 31–36.
28. Haughn K and Inman D. Autonomous learning in a pseudo-episodic physicalenvironment. *J Intell Robotic Syst Manuscript Accepted Publication* 2022; 104(2): 1–14.
29. Hwangbo J, Sa I, Siegwart R, et al. Control of a quadrotor with reinforcement learning. *IEEE Robotics Automation Lett* 2017; 2(4): 2096–2103.
30. Koch W, Mancuso R, West R, et al. Reinforcement learning for uav attitude control. *ACM Trans Cyber-Physical Syst* 2019; 3(2): 1–21.
31. Hu D, Pei Z, Shi J, et al. Design, modeling and control of a novel morphing quadrotor. *IEEE Robotics Automation Lett* 2021; 6(4): 8013–8020.
32. Xu D, Hui Z, Liu Y, et al. Morphing control of a new bionic morphing uav with deep reinforcement learning. *Aerospace Sci Technology* 2019; 92: 232–243.
33. Lampton A, Niksch A and Valasek J. Reinforcement learning of a morphing airfoil-policy and discrete learning analysis. *J Aerospace Comput Inf Commun* 2010; 7(8): 241–260.
34. Valasek J, Tandale MD and Rong J. A reinforcement learning-adaptive control architecture for morphing. *J Aerospace Comput Inf Commun* 2005; 2(4): 174–195.
35. Valasek J, Doebbler J, Tandale MD, et al. Improved adaptive– reinforcement learning control for morphing unmanned air vehicles. *IEEE Trans Syst Man, Cybernetics, Part B (Cybernetics)* 2008; 38(4): 1014–1020.
36. Goecks VG, Leal PB, White T, et al. Control of morphing wing shapes with deep reinforcement learning. In 2018 AIAA Information Systems-AIAA Infotech@ Aerospace. American Institute of Aeronautics and Astronautics, 2018. p. 2139.
37. Svozil D, Kvasnicka V and Pospichal J. Introduction to multilayer feed-forward neural networks. *Chemometrics Intelligent Laboratory Systems* 1997; 39(1): 43–62.
38. Kiranyaz S, Avci O, Abdeljaber O, et al. 1d convolutional neural networks and applications: a survey. *Mech Systems Signal Processing* 2021; 151: 107398.
39. Hochreiter S and Schmidhuber J. Long short-term memory. *Neural Computation* 1997; 9(8): 1735–1780.
40. Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv preprint arXiv:170706347 2017.
41. Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning. PMLR, pp. 1861–1870.
42. Kingma DP and Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.
43. Tabor P. ppo in pytorch, 2020. https://github.com/philtabor/Youtube-Code-Repository/tree/master/ReinforcementLearning/PolicyGradient/PPO/torch
44. Xu B, Wang N, Chen T, et al. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv: 150500853 2015.

45. Buşoniu L, de Bruin T, Tolić D, et al. Reinforcement learning for control: performance, stability, and deep approximators. *Annu Rev Control* 2018; 46: 8–28.
46. Ziegler JG, Nichols NB, et al. Optimum settings for automatic controllers. *Trans ASME* 1942; 64(11).
47. Edwards C and Postlethwaite I. Anti-windup and bumpless- transfer schemes In *emphAutomatica* 1998; 2: 199–210
48. Silver D, Singh S, Precup D, et al. Reward is enough. *Artif Intelligence* 2021; 103535.
49. Dewey D. Reinforcement learning and the reward engineering principle. In 2014 AAAI Spring Symposium Series. p. n/a.
50. Cook RG, Palacios R and Goulart P. Robust gust alleviation and stabilization of very flexible aircraft. *AIAA Journal* 2013; 51(2): 330–340.
51. Hufstedler EA and Chatelain P. Loads alleviation on an airfoil via reinforcement learning. In AIAA Scitech 2019 Forum. p. 0404.
52. Nathan D, Deo A, Haughn K, et al.. Si-based self-programming neuromorphic integrated circuits for intelligent morphing wings. *Journal of Composite Materials* 2022; 00219983221134929.