# Quantifying the Sim-To-Real Gap in UAV Disturbance Rejection

**Austin Coursey**
Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

**Marcos Quinones-Grueiro**
Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

**Gautam Biswas**
Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

## Abstract

Due to the safety risks and training sample inefficiency, it is often preferred to develop controllers in simulation. However, minor differences between the simulation and the real world can cause a significant sim-to-real gap. This gap can reduce the effectiveness of the developed controller. In this paper, we examine a case study of transferring an octorotor reinforcement learning controller from simulation to the real world. First, we quantify the effectiveness of the real-world transfer by examining safety metrics. We find that although there is a noticeable (around 100%) increase in deviation in real flights, this deviation may not be considered unsafe, as it will be within $> 2$m safety corridors. Then, we estimate the densities of the measurement distributions and compare the Jensen-Shannon divergences of simulated and real measurements. From this, we show that the vehicle's orientation is significantly different between simulated and real flights. We attribute this to a different flight mode in real flights where the vehicle turns to face the next waypoint. We also find that the reinforcement learning controller actions appear to correctly counteract disturbance forces. Then, we analyze the errors of a measurement autoencoder and state transition model neural network applied to real data. We find that these models further reinforce the difference between the simulated and real attitude control, showing the errors directly on the flight paths. Finally, we discuss important lessons learned in the sim-to-real transfer of our controller.

**2012 ACM Subject Classification** Computing methodologies → Control methods; Computing methodologies → Model development and analysis

**Keywords and phrases** sim-to-real, disturbance rejection, unmanned aerial vehicles

**Digital Object Identifier** 10.4230/OASIcs.DX.2024.16

**Funding** This work was supported by NASA award #80NSSC21M0087-21-S06.

**Acknowledgements** We want to thank Luis Alvarez at MIT Lincoln Laboratory for his help in running UAV flights.

## 1 Introduction

Reinforcement learning (RL) techniques have shown a wide range of success across a variety of tasks such as games [22], robotic hand manipulation [2], and racing simulations [28]. In the realm of diagnosis, this class of algorithms has been used to perform fault diagnosis of rotating machinery [9, 18] and to reduce data imbalance in fault diagnosis [10]. RL has also been used to increase the resilience of dynamical systems. For example, a recent study covers the application of RL in enhancing the reliability of power and energy systems [11]. RL methods are classified into dynamic response optimization, restoration and recovery, energy management, cybersecurity, and resilient planning. In unmanned aerial vehicles (UAVs), RL has been used to avoid collisions [16, 4] and increase safety in windy [30, 6] and fault conditions [23].

Novel RL methods hold great potential for real-world applications. However, due to the sample inefficiency of RL algorithms [31], they often cannot be trained in the real world. Additionally, RL-based agents can make dangerous decisions during the learning process that can risk the safety of the system [19]. Therefore, it is often desired to train these agents in simulation.

Inaccuracies in the simulation model used to train RL methods define the existing *sim-to-real gap* [26, 7]. This gap can impact the performance of controllers when transferred to the real world, leading to reduced performance [19]. While techniques such as domain randomization [26] have been developed to reduce this gap during simulation, methods for quantifying the gap and isolating causes for worse real-world performance are understudied. In this paper, we examine a case study of the transfer of an octorotor UAV disturbance rejection controller from simulation to real flights. We detail three methods for examining the sim-to-real gap. First, we quantify the overall safety of the real system and compare it to simulation using a set of safety metrics. Then, we estimate the density distributions of the measurements and determine which measurements failed to match between the simulation and the real world using their Jensen-Shannon divergence. Finally, we learn an Autoencoder neural network and system transition neural network on simulated data to determine which points along the UAV flight had a larger sim-to-real gap.

In this paper, we make the following contributions.

1. We show a recent case study of the transfer of an octorotor UAV disturbance rejection controller from simulation to the real world. This case study gives evidence for the potential of the combination of RL agents as supervisory controllers and widely used classic controllers to maintain UAV safety in the real world.
2. We study the potential of three methods for examining and isolating the sim-to-real gap. These are (1) system safety metrics, (2) measurement distribution analysis, and (3) neural network transfer error inspection.
3. We discuss important lessons we learned in transferring from simulation to the real world. These lessons can help future researchers ensure a more successful transfer of UAV disturbance rejection controllers.

The rest of the paper is structured as follows. In Section 2, we summarize related research on this topic. In Section 3, we give brief preliminary definitions. In Section 4, we detail the controller development process and its transfer to the real world. In Section 5, we present our methods for quantifying and identifying the sim-to-real gap. In Section 6, we examine the sim-to-real gap on our octorotor case study. In Section 7, we discuss important lessons we learned through this case study. Finally, in Section 8, we conclude the paper.

## 2 Related Work

Transferring a UAV controller from simulation to the real world is a common task in control literature. For example, in [14], a sigma-pi neural network is used to compensate for unknown dynamics in a PID control architecture. This algorithm is applied to a real quadrotor. In [12], they compare feedback control methods to a nonlinear model predictive control method for quadrotors in simulation. Then, they applied the algorithms to a real drone. They compared the performance of these algorithms in the real world. In [29], they design a controller for quadrotor disturbance rejection. They use this controller with a real quadrotor in an outdoor flight test. In [15], a deep reinforcement learning controller that directly controls the motor speeds of a UAV is developed. They apply their algorithm to a real hexacopter UAV. Although there is a wide range of literature that applies control algorithms developed in a simulator to the real world, most of these papers do not focus on methods for determining the success of this transfer.

Instead, much of the recent literature focuses on techniques for improving the sim-to-real transfer. For example, the sim-to-real gap has been studied in UAV computer vision tasks. In [8], they compare the drone detection performance in simulated and real-world settings. They attempt to match the environmental conditions in simulation to real videos. They use vision-specific metrics like the mean average precision to compare drone detection performance in simulation and the real world. In [24], they attempt to mitigate the sim-to-real gap by integrating computer vision into their autonomous landing algorithm. To reduce the sim-to-real gap, they replicate the appearance of real-world landing pads in their simulator. In [17], they present a framework for developing aerial robotics systems while avoiding the sim-to-real gap. Their framework involved developing the system in simulation, integrating the algorithm with the hardware, performing validation in augmented reality, and finally using the system in an industrial application. They apply this framework for UAV wind turbine inspection. In [27], they use domain randomization techniques to improve the sim-to-real transfer of a fixed-wing UAV control algorithm in a wind tunnel. They showed that domain randomization improves real-world performance. Techniques like these offer promising ideas for improving the transferability of algorithms developed for UAVs in simulation. However, the evaluation of the quality of the sim-to-real transfer is limited to qualitative flight profiles or state error computations.

Outside of UAVs, perhaps the most active area of sim-to-real research is on techniques for reducing the gap at simulation time. Domain randomization [26] is one such popular approach. In domain randomization, some parameters of the simulation are randomized during the model learning process. By learning to generalize across these parameters, the model should learn to generalize to the real world better. This, along with transfer learning, is widely successful in robotics tasks [19]. To address the fact that the randomized parameters need to be known beforehand, [13] proposes an approach that automates parameter tuning. They learn a causal graph to determine which environmental parameters lead to a trajectory difference. In [25], they improve the sim-to-real transfer of their robot by using system identification techniques to improve the accuracy of their simulation. Then, they improve the robustness of the controller with domain randomization, random perturbations, and reducing the observation space. Again, these techniques do not focus on quantifying the sim-to-real gap. Some recent papers do [5], but the evaluation is limited to inspecting high-level errors instead of focusing on understanding why the gap exists.

In this paper, we present a case study of the transfer of an octocopter UAV reinforcement learning disturbance rejection controller from simulation to the real world. Our analysis goes beyond state error metrics, comparing the distributions of metrics and leveraging the learning process of neural networks to isolate unexpected behaviors.

## 3 Preliminaries

In this paper, we attempt to quantify the *sim-to-real gap*. The sim-to-real gap, or reality gap, can be simply defined as a discrepancy between simulated and real-world environments [26, 7]. By reducing the gap, controllers or models developed in simulation can be better applied in real systems.

The controller we developed in simulation is a reinforcement learning UAV disturbance rejection controller. Reinforcement learning aims to find a solution to a sequential decision-making problem expressed as a Markov Decision Process (MDP). An MDP consists of $\mathcal{S}$, a set of environmental states, $\mathcal{A}$, the actions the agent can take in the environment, $p$, a transition function $p(s'|s,a)$ that denotes the probability of reaching a next state, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

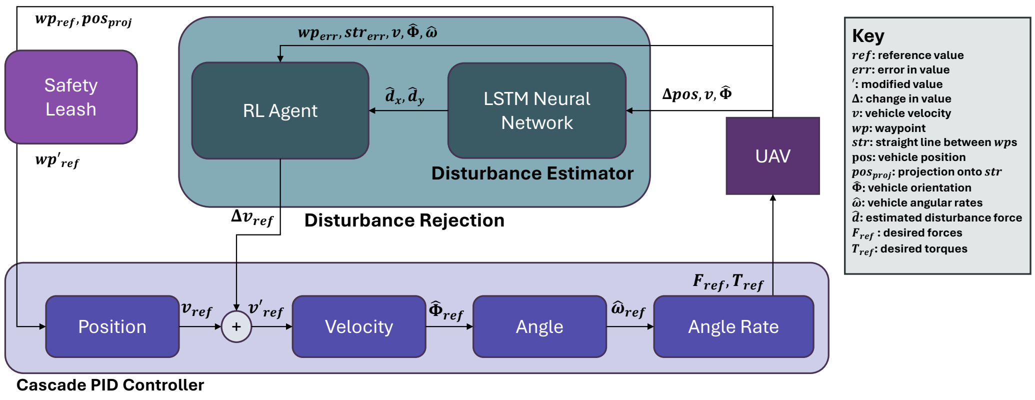


**Figure 1** Full UAV disturbance rejection controller. Figure and controller modified from [6] to include the estimated disturbance forces as an RL state.

a reward function for taking an action in a state. Therefore, the MDP $= \{\mathcal{S}, \mathcal{A}, p, r\}$. The solution to this problem is characterized by a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected discounted future reward.

## 4 Controller Development and Transfer

This paper discusses transferring a hybrid reinforcement learning and control-based UAV controller to the real world. The goal of this controller is to perform disturbance rejection. The controller should increase the safety and robustness of the UAV under disturbances. The controller was primarily developed with strong wind in mind. This section details the controller development process in simulation and its application on a real UAV.

### 4.1 Simulation

Before using any controllers on a real UAV, we developed a disturbance rejection controller in simulation. We modeled and simulated a Tarot T-18 octocopter UAV. This UAV model consisted of 8 18-inch propellers, was about 1.25 meters in diameter, and had a 10.66 kg mass. To simulate this UAV, we used an open-source Python multirotor UAV simulator [1]. To simulate flights, we provided a predefined set of reference waypoints the UAV needed to reach by getting within a 2-meter radius. In simulation, the UAV was allowed to fly in a 2-meter cylindrical corridor between waypoints, the safety corridor. In the real world, we may consider less strict constraints, such as with a 5-meter safety corridor. Outside this corridor, there could be obstacles that compromise the safety of the UAV.

A common UAV control scheme is a cascade proportional-integral-derivative controller. This control scheme stacks PID controllers to go from a reference position to reference forces and torques to be applied to the UAV. Despite its widespread use, it is a linear control scheme with limited robustness to strong wind conditions [6]. See Figure 1 for an illustration of this control scheme. To move beyond the rigidness of this controller while still maintaining its strengths as a stable and trusted control algorithm, we integrated it with a deep reinforcement learning agent.

To develop the agent, we considered a set of features that would be reliably observable at 2 Hz or faster on a real UAV. These made up the state space. The state space consisted of 17 features, the distance to the next waypoint in meters $wp_{err} \in \mathbb{R}^3$, the vehicle inertial velocity

$v \in \mathbb{R}^3$, the angular orientation $\hat{\Phi} = [\phi, \theta, \psi] \in [-\pi, \pi]^3$, the angular rates $\hat{\omega} \in \mathbb{R}^3$, the distance to the straight line between waypoints $str_{err} \in \mathbb{R}^3$, and the estimated disturbance forces in the $x$ and $y$ directions $Fx \in \mathbb{R}, Fy \in \mathbb{R}$. To estimate the disturbance forces, we trained a long short-term memory (LSTM) neural network model on data from simulated flights. This LSTM used data from the past second (sampled at 0.25 seconds for real flights and 0.1 seconds for real flights). It used the full-second time window of the change in position, the vehicle velocity, and the orientation as input to predict the disturbance X and Y forces.

The agent was allowed to modify the X and Y reference velocities in the cascade PID controller. These modifications are added to the reference velocity given by the position controller (see Figure 1). The actions were restricted to modify the reference velocity by at most 1 meter a second. Therefore, the action space was $\Delta v_{ref} \in [-1, 1]^2$. Additionally, the agent was constrained to only provide actions every 0.5 seconds and receive state observations every 0.25 seconds to mimic the delay between a computer on a ground station and a UAV. The actions were applied for the full half a second until the next decision.

To train the agent, we used the proximal policy optimization (PPO) algorithm [20]. We trained the agent to maximize the following reward function.

$$\text{reward}(\text{state}_t) = -\frac{\text{tte}(pos, wp_{\text{prev}}, wp_{\text{ref}})}{T_{\text{safe}}} + \text{bonus}_t \tag{1}$$

$$\text{tte}(pos, wp_{\text{prev}}, wp_{\text{ref}}) = \frac{\|(pos - wp_{\text{prev}}) \times (wp_{\text{ref}} - wp_{\text{prev}})\|}{\|(wp_{\text{ref}} - wp_{\text{prev}})\|} \tag{2}$$

In this equation, $pos$ is the current position, $wp_{\text{prev}}$ is the previous waypoint, $wp_{\text{ref}}$ is the desired waypoint, $T_{\text{safe}}$ is a normalization constant (the radius of the safety corridor), and the trajectory tracking error (tte) is the deviation from the straight line between waypoints. Additionally, there is a bonus for completing a mission or a penalty for failing. We trained the agent over a series of simulated flights in a square trajectory. The agent experienced random constant wind vectors with turbulence each flight. The controller was commanded to have a fixed yaw of 0 radians. For full details of this controller along with a rule-based waypoint modification scheme we designed, please refer to our previous work where we developed and evaluated a previous version of this controller strictly in simulation [6]. The only differences in the controller used for this paper are that it estimates the disturbance forces instead of the wind speed and can only make small X and Y velocity adjustments.

With the problem of sim-to-real transfer in mind, we took a few steps to increase the transfer effectiveness during this phase. First, we used domain randomization [26] during the training process. In domain randomization, some properties of the simulation are varied across trials. In our case, we varied the wind direction, the wind magnitude, and the flight direction in each episode. Intuitively, if an agent can learn a general policy across variations in the simulation, it may generalize better to the real world. Second, we added noise to the observations to mimic measurement noise. Third, we varied the orientation of the rotors on the UAV to empirically determine whether the UAV could generalize to model variations in simulation. Finally, we obtained real octocopter flight data from NASA Langley and qualitatively validated the realism of our measurements.

## 4.2 Real-World

We integrated our controller with the autopilot control of a DJI S100 octocopter UAV through a framework that uses the Mavlink protocol for data exchange. The airframe of the UAV has a diameter of about 1.25m and weighs around 10 kg, similar to the T-18 used for simulation. The octocopter has 8 18-inch propellers. A picture of the drone can be seen in Figure 2.

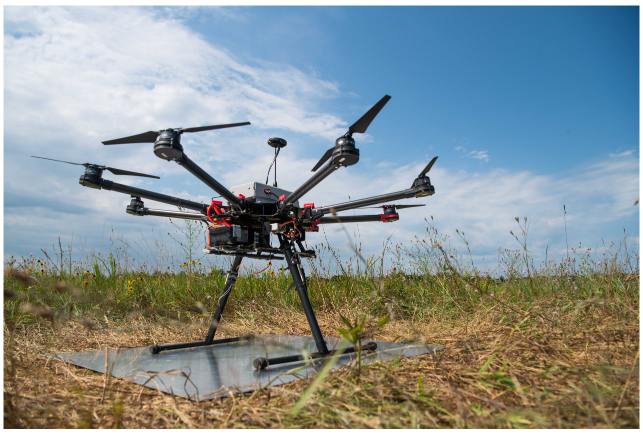
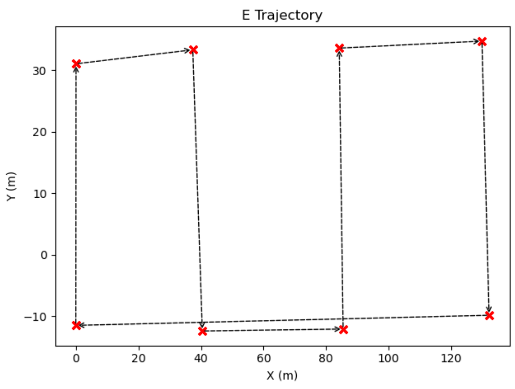
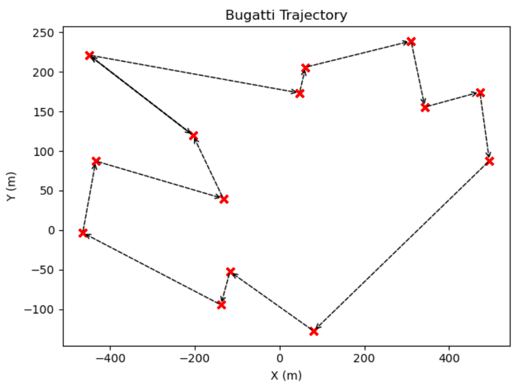


**Figure 2** Image of the octocopter we used for the real flight experiments.



**(a)** E shape trajectory.



**(b)** 'Bugatti' trajectory.

**Figure 3** Desired trajectories for the real flights. The desired altitude was constant.

The UAV was controlled through the ArduPilot software. To connect our controller to this software, a ground station computer used our position controller to compute reference velocities every 0.25 seconds, our data sampling rate. Then, those reference velocities were sent to ArduPilot to perform low-level control of the UAV. Every 0.5 seconds, our RL controller, also on a ground station computer, modified the reference velocities to counteract any estimated disturbances. Notably, this control scheme is slightly different than our simulation scheme. In simulation, we use a cascade PID scheme, described in the previous section. On the real UAV, we used the autopilot software. Although we modeled our cascade UAV after the controllers used in this software, we did not directly use it when developing our controller. This likely led to some differences in behavior in the real flights. Additionally, this software commanded the UAV to face its next waypoint when flying toward it instead of facing completely straight (it used yaw control). In simulation, we did not use this operating mode. The impacts of this change in operating mode are studied extensively in Section 6.

We flew 5 full flights using our controller at Fort Devens, Massachusetts. The trajectories the UAV flew can be seen in Figure 3. We flew a shorter E trajectory that flew in all cardinal directions and a longer "bugatti" trajectory with more complex turns.

## 5 Methods for Quantifying the Sim-to-Real Gap

With our real UAV flights wrapped up, we wanted to determine how well the controller we developed in simulation transferred to the real world. To do so we replicated the real flights in simulation, compared the overall safety of flights, analyzed the distributions of features, and leveraged the learning process of neural network models to uncover differences in real and simulated flight data.

To directly compare the real flights to simulated ones, we replicated the real flights in simulation. For two of the flight days, we made the UAV hold its altitude with no X or Y position commands. We took the average vehicle velocity during this phase as an estimate of the wind vector for that flight. This left us with 5 full flights (4 E shape and 1 bugatti shape), with known wind conditions. Then, we ran those flights in simulation, adding Dryden turbulence to the wind to increase the realism.

### 5.1 Safety Metrics

A simple way to quantify the sim-to-real gap in UAVs is to compare important flight-level metrics. In this case, since it is a disturbance rejection controller that aims to improve octorotor UAV safety, we considered the following metrics. In the computation of these metrics, we used the trajectory tracking error (Equation 2) and notated its input as a measurement vector, where $M$ is the set of all measurement vectors in that flight, to simplify notation.

- **Mean Deviation (Mean TTE)**. The average distance from the straight line between waypoints across the flight. If the mean deviation is smaller than the radius of the safety corridor, that means the UAV was safe on average.

$$\frac{1}{|M|}\sum_{m_t \in M} \mathrm{tte}(m_t) \tag{3}$$

- **Maximum Deviation (Max TTE)**. The maximum distance from the straight line between waypoints across the flight. If the maximum deviation is smaller than the radius of the safety corridor, then that means the UAV never violated safety in that flight.

$$\max(\{\mathrm{tte}(m_t) \mid m_t \in M\}) \tag{4}$$

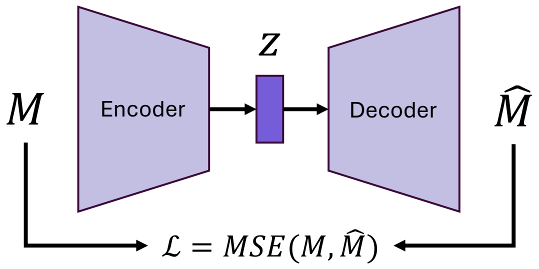
- **Percent of Time Outside Safety Corridor (% TOC)**. The percentage of the flight spent outside a predefined safety corridor. In the following equation, $r$ is the radius of the safety corridor.

$$\frac{|\{m_t \in M \mid \mathrm{tte}(m_t) > r\}|}{|M|} \tag{5}$$

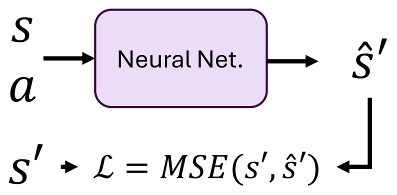
### 5.2 Measurement Distributions

Next, we quantified the sim-to-real gap for individual measurements. However, since the simulated and real measurements are not aligned in time or space we cannot directly use the difference in measurements as a fair comparison. Instead, we compared the distributions of measurements.

To model the distribution of measurements, we performed kernel density estimation using the statsmodels Python package [21]. Then, with a continuous distribution for each measurement, we compared the distributions using the Jensen-Shannon divergence (JSD),



**(a)** Measurement autoencoder.



**(b)** State transition model neural network.

**Figure 4** Neural network architectures used to quantify the sim-to-real gap. The measurement autoencoder will fail to reconstruct any differences in real-world measurements. The state transition model will predict an incorrect next state if there is a large sim-to-real gap.

shown in the equations below where $\mathcal{X}$ is a common sample space and $P$ and $Q$ are the estimated density distributions.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P(x)}{Q(x)} \right) \tag{6}$$

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2} D_{\mathrm{KL}}(P \parallel \frac{1}{2}(P+Q)) + \frac{1}{2} D_{\mathrm{KL}}(Q \parallel \frac{1}{2}(P+Q)) \tag{7}$$

A JSD close to 0 implies the two distributions are almost identical, and a JSD closer to $\ln(2) \approx 0.693$ implies the two distributions are completely different.

## 5.3 Leveraging Neural Networks

While measurement distribution analysis can give an overall sense of the sim-to-real gap, it does not allow for easy inspection of the specific scenarios where the sim-to-real transfer failed or was successful. To do so, we trained two neural network models, a measurement autoencoder and a state transition model.

### 5.3.1 Measurement Autoencoder

The first neural network we trained to quantify the sim-to-real gap was a measurement Autoencoder. We used a vanilla neural network autoencoder instead of a variational autoencoder or another variant because we do not care about the distribution of the latent space; we only care that it serves as a bottleneck. This Autoencoder, shown in Figure 4, compresses the measurements $M$ down to a latent vector $z$ using an encoder neural network. Then, this latent vector is projected back to the measurement space using a decoder neural network. It was trained to minimize the mean squared error of the measurement reconstructions, shown below where $n$ is the dimension of the measurement vector $M$.

$$MSE(M, \hat{M}) = \frac{1}{n} \sum_{i=1}^{n} (M_i - \hat{M}_i)^2 \tag{8}$$

We trained this Autoencoder on two simulated flights (one Bugatti and one E), leaving a final E flight for validation. The measurements we considered were the error to the next waypoint, the vehicle's velocities, the vehicle's angular orientation, and the vehicle's angular

**Table 1** Optimal hyperparameters. The same parameters were given to the encoder and decoder of the Autoencoder. Notice that the transition model trained for longer with a higher learning rate and a larger network, implying this may be a more difficult task.

| Model | Epochs | Learning Rate | Batch Size | Layers | Neurons | Latent Dim. |
|---|---|---|---|---|---|---|
| Autoencoder | 24 | 0.00060 | 16 | 2 | 256 | 6 |
| Transition | 44 | 0.00098 | 16 | 3 | 256 | - |

rate. These made up a 12-dimensional input space. We normalized this data and optimized the hyperparameters on the validation data using the tree-structured Parzen estimator algorithm [3]. The optimal hyperparameters are shown in Table 1.

Although the task of learning to reconstruct the input seems trivial, the compression of data to a latent vector loses information, keeping the Autoencoder from being an identity function. Intuitively, the reconstruction error should be high when applying this model to real data if there is a large sim-to-real gap. Additionally, we can inspect which measurements are causing this error and map these errors to flight paths.

### 5.3.2 State Transition Model

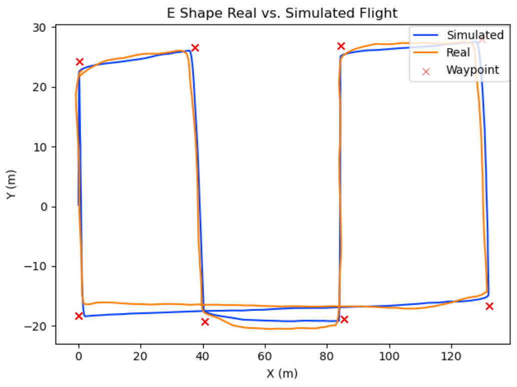
Next, we trained a neural network to mimic a system transition model. This neural network, shown in Figure 4, predicts the next state from the current state and action. It was trained to minimize the mean squared error of the predicted next state. The states were the same measurements as for the state Autoencoder above. The actions were the actions of the reinforcement learning agent, the change in reference X and Y velocity. Therefore, this neural network models the transition from the perspective of the reinforcement learning agent.

We trained this agent with the same training process as the Autoencoder network, optimizing the hyperparameters on a simulated validation flight and training on the other simulated flights. The optimal hyperparameters are shown in Table 1. By training on the simulated flights, it learns the simulation transition function. If there is a large sim-to-real gap, this network should incorrectly predict the next state on real-world data.

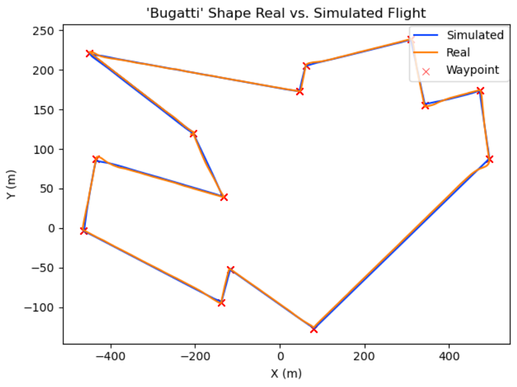
## 6 Results

First, we can analyze the sim-to-real gap in terms of the overall system safety. Figure 5 shows a qualitative comparison of the real and simulated flight paths. On the shorter E trajectory, we can see minor differences in the behavior in the real world. Notably, the simulated flight seems to be straighter. The real UAV slightly overshot the waypoint near (80, -20) and undershot the waypoint near (0, -20). Additionally, the flights seemed less stable overall in the real world. This difference appears more prominent when flying in the X direction, but this is likely due to the scale of the axes. In the Bugatti trajectory, the differences between the real and simulation seem even smaller. The only clear difference is minor waypoint overshooting, like at (-400, 100), in the real flight.

We can quantify the difference between these simulated and real flights by looking at safety metrics. Table 2 shows these metrics. From this table, we can see that there is a clear difference between the performance of the real and the performance of the simulated flights. In both the E shape and bugatti shape flight paths, the average deviation (Mean TTE) was about half in the simulation. The same pattern holds for the maximum deviation and total deviation. This indicates that there is a sim-to-real gap. However, this gap may not



**(a)** E flight path. Wind of (1, 0.5) m/s.



**(b)** 'Bugatti' flight path. Wind of (-3.5, -2.5) m/s.

**Figure 5** Real and simulated flight paths. Note the difference in axis scales between the E and Bugatti trajectories.

**Table 2** Safety metrics for real and simulated flights. In the E shape category, the mean of the 4 real E flights and the mean of the 2 simulated E flights are shown. The time outside the corridor metrics (TOC) are notated with the radius of the safe corridor.

| Type | Mean TTE (m) | Max TTE (m) | 2m TOC (%) | 5m TOC (%) |
|---|---|---|---|---|
| | | E Shape | | |
| Real | 1.180 | 4.617 | 15.9 | 0 |
| Simulated | 0.491 | 1.996 | 0 | 0 |
| | | Bugatti Shape | | |
| Real | 1.286 | 4.987 | 20.7 | 0 |
| Simulated | 0.544 | 2.042 | 0.2 | 0 |

have been large enough to impact the safety of the flights. None of the flights left a 5-meter corridor around the straight line between waypoints. Additionally, the real flights were within a 2-meter corridor on average, leaving at most 20% of the time during the Bugatti flight. Depending on the specific safety requirements, the quality of the sim-to-real transfer may be sufficient for these trajectories.

## 6.1 Distributional Analysis

From the safety metric analysis, we determined that there was a noticeable difference in performance in real flights. To isolate the cause of this difference, we can start by analyzing the measurement distributions. Figure 6 shows the kernel density estimations of the measurement distributions. It also shows the Jensen-Shannon divergence (JSD) between the distributions. For this trajectory, many of the measurements have similar distributions. For example, the X and Y positions and velocities have small JSD. Additionally, the difference in the deviation distributions reinforces the superior performance of the controller in simulation. There is a large difference in the yaw distributions. In the real flights, we were unable to fix the yaw at 0 radians as we did in the simulation. This is a dramatic change in the UAV behavior. (The UAV turned after reaching each waypoint instead of always facing forward.) By noticing the large JSD, we can isolate this change in behavior. Aside from the yaw, the disturbance force

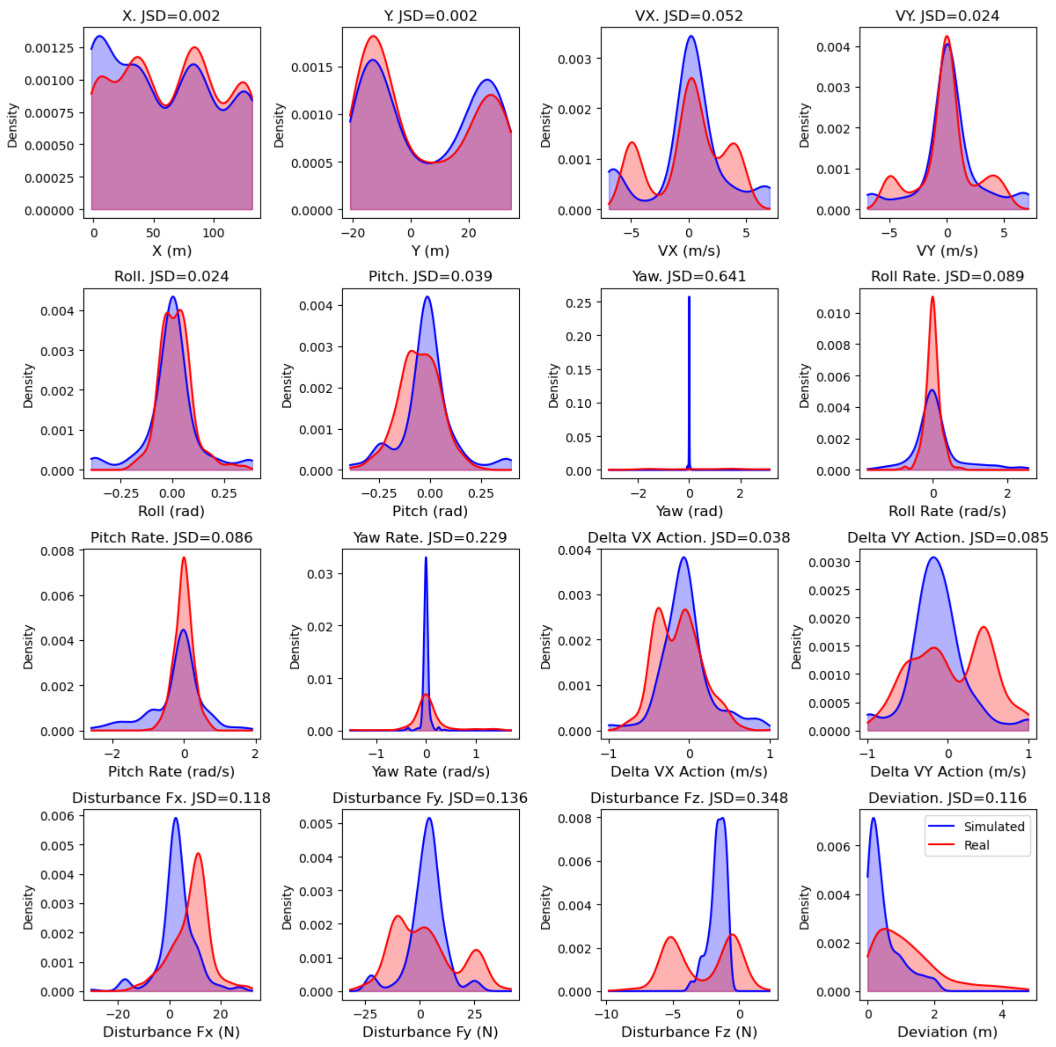


**Figure 6** Kernel density estimations and Jensen-Shannon divergences between real and simulated flight distributions for the E-shape flights. The disturbance forces are estimated by the disturbance estimation neural network in the controller.

estimations given by the controller had large divergences. These estimations are in the UAV body frame. Since the heading of the UAV changed in the real flights, the directions of the disturbances in the body frame perspective changed. Therefore, the change in yaw likely caused this behavior difference too.

Figure 7 shows similar results for the Bugatti shape flight. In this flight, we see larger deviations in the roll and pitch than the E shape, likely due to the more complex turns and longer stretches with differing yaw angles. The deviation from the straight line between waypoints in the simulated flights was multimodal for this trajectory. The second mode being centered around a deviation of 1 meter implies that this trajectory was more difficult.

To determine the impact of the change in heading on the disturbance estimator, we converted the estimated disturbances from the vehicle body frame to the inertial frame. Figure 8 shows these results for the E shape trajectory. The estimated disturbance forces for the simulated flights closely matched the prescribed wind vector (1, 0.5) while the disturbance

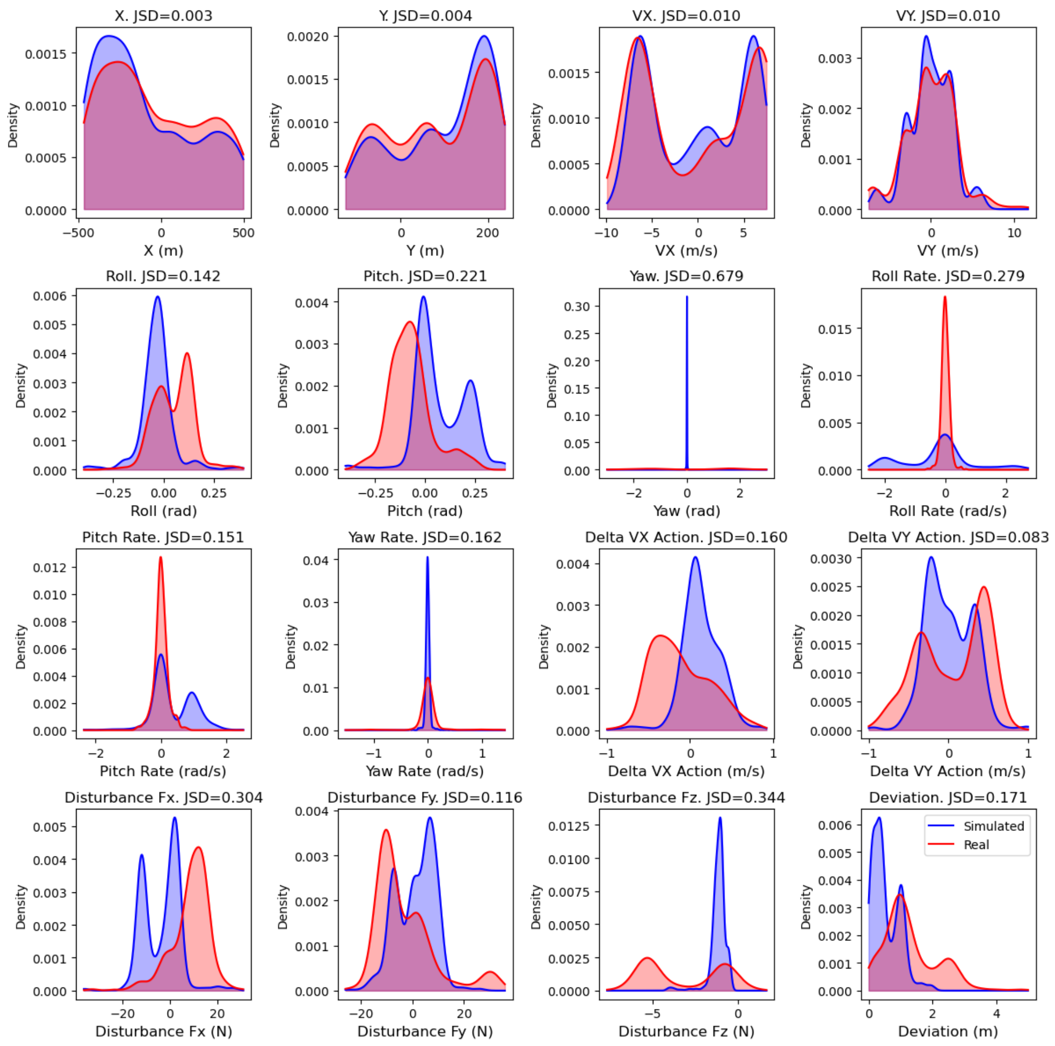
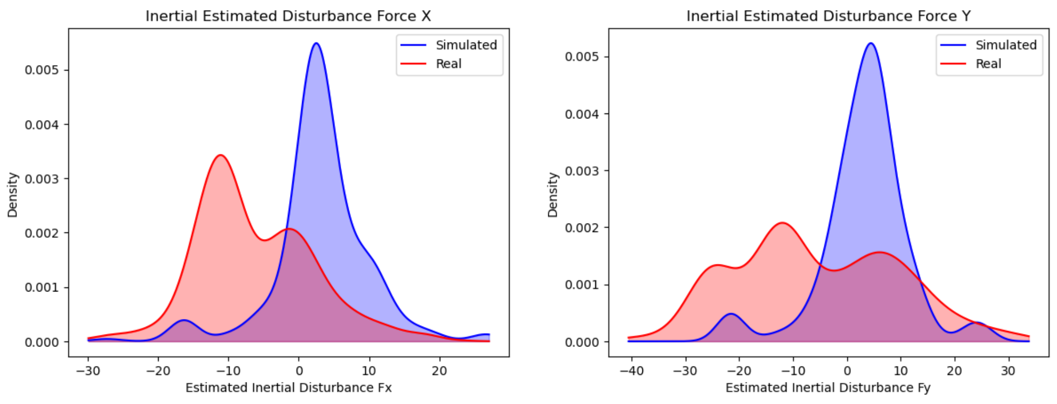


**Figure 7** Kernel density estimations and Jensen-Shannon divergences (JSD) between real and simulated flight distributions for the Bugatti-shape flights. The disturbance forces are estimated by the disturbance estimation neural network in the controller.

forces for the real flights did not. As previously mentioned, we estimated the wind vector after the flight, so the disturbance forces shown in Figure 8 may accurately represent the true disturbance forces during the flight. However, this difference may instead be caused by the disturbance estimator failing to generalize to real flights with a different operating behavior (heading towards the waypoint) than in simulation.

In both Figure 6 and Figure 7, we see evidence that the controller is operating as intended. The actions given during the real flights by the agent (Delta VX Action and Delta VY Action in the figures) were skewed in the opposite direction as the estimated disturbance forces. This shows the agent attempted to properly reject disturbances in the real world. Despite the sim-to-real gap in the angle control, the controller appeared to make proper decisions.



**(a)** Inertial estimated disturbance force X. **(b)** Inertial estimated disturbance force Y.

**Figure 8** Estimated disturbance forces output by the disturbance estimator converted to the inertial frame to account for differences in yaw control between simulated and real flights. Example shown for E shape trajectory. The simulated disturbances forces closely follow the estimated wind vector. The real estimated disturbance forces are much nosier and do not always match the wind.

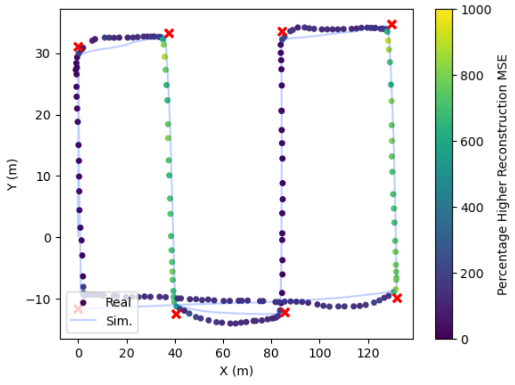
## 6.2 Quantifying the Gap along Flight Paths

Although these density estimations highlight important sim-to-real differences on a distribution level, they do not map these issues to the flight path and potentially miss subtle behavior differences. To do so, we can inspect the Autoencoder reconstruction error along the flights. This is shown in Figure 9. In this figure, the highest reconstruction errors are shown when the UAV is flying south. This is when the yaw is most different. When flying north, the reconstruction errors are very low. Additionally, the simulated flight reconstruction error is not higher when flying south. This indicates that the sim-to-real gap is largely caused by the difference in flight mode.

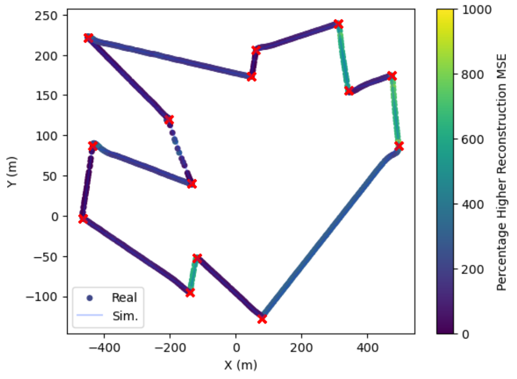
Figure 10 shows the autoencoder reconstruction errors without considering yaw or the yaw rate as measurements. Without these features, the reconstruction errors were much lower. This indicates that the other states more closely match between the simulation and the real world. The patterns in the reconstruction errors of these features are less obvious from the flight paths, indicating errors may be coming from more subtle differences or from secondary features that were impacted by the change in heading in real flights.

Finally, Figure 11 shows the errors of the system transition neural network along the flight paths. This quantifies the sim-to-real gap from the perspective of the reinforcement learning agent. If the error is higher, the system transition model may be different, and we may not be able to expect the controller to make consistent decisions. In this figure, we see that the largest errors are again when flying south. This may be a cause for the difference in safety metrics between simulated and real flights. However, the UAV does not deviate significantly, even while flying south. This means that even though there was a difference in the system transition, the agent may have been partially robust to it. Since the agent was trained with wind gusts, it sometimes experienced different yaw angles than 0 radians. Its generalizability to this scenario may be attributed to this.

These results indicate a clear next step. The controller needs to be expanded to support a yaw-changing operating mode or the autopilot on the real drone needs to be modified to force the yaw commands at 0 radians. Through our methods for sim-to-real quantification and visualization, we were able to isolate this issue and demonstrate its importance. We also

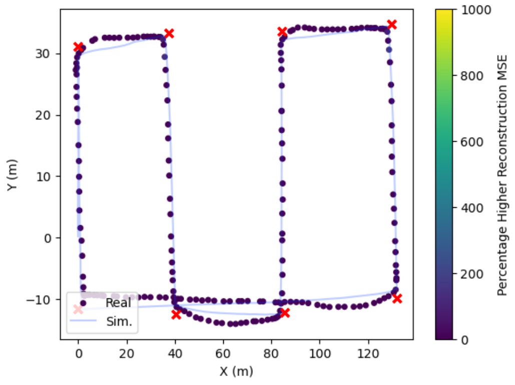


**(a)** E flight path.

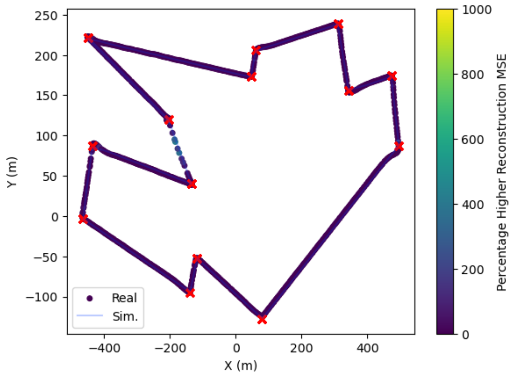


**(b)** 'Bugatti' flight path.

**Figure 9** Real flight paths colored by the percentage higher the real autoencoder reconstruction error was. A higher error by the autoencoder indicates a sim-to-real gap. Notice the main source of error comes when flying south, where the UAV faces the opposite direction as in the simulation.



**(a)** E flight path.



**(b)** 'Bugatti' flight path.

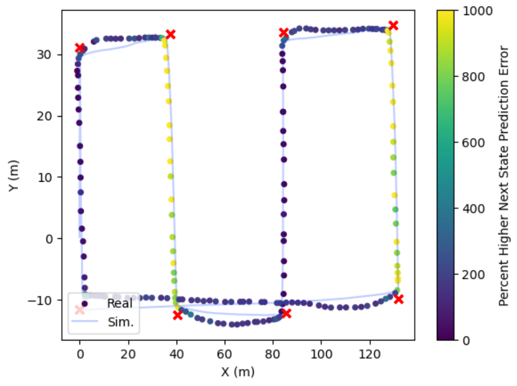
**Figure 10** Real flight paths colored by the autoencoder reconstruction error without considering yaw or the yaw rate. Notice that the errors are much smaller and less frequent. This implies fixing this controller difference may greatly improve the sim-to-real transfer.

found that this may not have significantly degraded the performance of the controller, as it still makes correct decisions and maintains an average deviation within a 2-meter safety corridor.

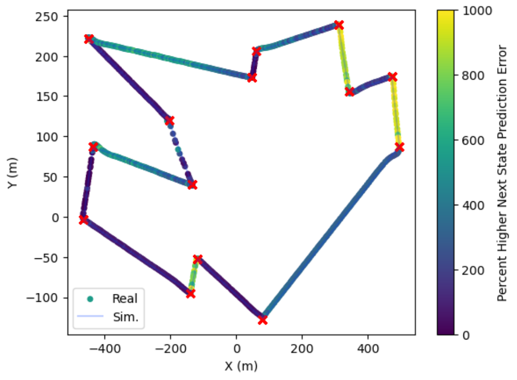
## 7 Discussion and Lessons Learned

Through this case study, we identified key points that may help future researchers as they transfer UAV controllers from simulations to the real world.

- **We can quantitatively isolate and identify issues in the sim-to-real gap.** In this paper, we demonstrated four methods to quantify the sim-to-real gap. Three of those, the kernel density estimations, measurement autoencoder, and state transition neural network, were able to isolate specific issues in the transfer to the real world. By calculating the divergence between measurement distributions, we could identify problematic features. To identify differences in behavior in the real world, we mapped the errors of the neural network models onto the flight paths. This can lead to faster and more accurate controller improvements.



**(a)** E flight path.



**(b)** 'Bugatti' flight path.

**Figure 11** Real flight paths colored by the state transition neural network error. A higher error indicates that the model predicted a different next state than was observed in the real world.

- **It is important to make reasonable assumptions when developing a controller in simulation.** When developing our controller, we assumed we could fix the desired UAV yaw to 0 radians on a real drone. However, this assumption did not hold. By avoiding restrictive assumptions like this and building flexibility into the controller in simulation, the sim-to-real gap can be lessened.
- **Despite the sim-to-real gap, controller performance may be sufficient.** We quantitatively identified issues in the transfer of our controller from simulation to the real world. However, the practical impacts of this sim-to-real gap may be small. The UAV was able to flight within a safe zone on average and the qualitative flight paths are reasonable. Further decreasing the sim-to-real gap may not be worth the time and cost of controller iteration.
- **We can develop controllers with the sim-to-real gap in mind.** We attribute the success of our controller in the real world to the techniques we used when developing the controller, namely randomizing parameters of the simulation during controller training and evaluation. By performing domain randomization, the agent may generalize to parameters of the real world. We also performed a comprehensive analysis of our controller's performance in simulation, trying to ensure that we limited the number of unexpected scenarios. In the real world, we commanded the UAV to hold its altitude at the end of flights so that we could replicate the wind conditions in simulation.

## 8 Conclusion

In this paper, we analyzed a case study on the transfer effectiveness of a UAV disturbance rejection controller from simulation to the real world. We developed a reinforcement learning controller that modifies the reference velocity of a cascade PID control scheme. This agent was trained and evaluated completely in simulation. Then, we flew real octocopter flights using this controller. Next, we replicated those flights in simulation. Using this data, we qualitatively inspected the difference in flight profiles, observing that the simulated UAV flew straighter and overshot waypoints less. Then, we compared the safety of simulated and real flights, finding that real flights deviated from a 2-meter safety corridor around 16% of the time while the simulated flights never left the corridor. Despite this difference, the safety of the real flights is adequate for many applications, such as ones with larger corridors. To isolate which features have the largest sim-to-real gap, we performed kernel density estimation on the measurements. We calculated the Jensen-Shannon divergence of those densities, finding that the yaw had the largest divergence due to the real UAV turning when reaching a waypoint. We also used these densities to show that the controller actions counteracted the estimated disturbances, implying the controller produced correct actions in the real world. Then, we used an autoencoder neural network model to learn to reconstruct simulated measurements. We mapped its reconstruction errors on real data onto flight paths to show that the largest sim-to-real gap was when flying south. Additionally, we trained another neural network to learn the state transition model to show that the transition when flying south was also different, potentially reducing controller performance. Finally, we discussed important lessons learned through this case study.

Future work should apply these sim-to-real quantification metrics on other systems to determine their generalizability across diagnosis domains. It should also test this controller in more extreme wind conditions in the real world to quantify the sim-to-real gap in settings where dire reinforcement learning intervention is needed. We hope that research in this area will improve the efficiency of transfer from simulation to the real world and the overall safety of real systems.

## References

1 Ibrahim Ahmed, Marcos Quinones-Grueiro, and Gautam Biswas. A high-fidelity simulation test-bed for fault-tolerant octo-rotor control using reinforcement learning. In *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*. IEEE, 2022.

2 OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. doi:10.1177/0278364919887447.

3 James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

4 Omar Bouhamed, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. Autonomous uav navigation: A ddpg-based deep reinforcement learning approach. In *2020 IEEE International Symposium on circuits and systems (ISCAS)*, pages 1–5. IEEE, 2020. doi:10.1109/ISCAS45731.2020.9181245.

5 Jack Collins, David Howard, and Jurgen Leitner. Quantifying the reality gap in robotic manipulation tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6706–6712. IEEE, 2019.

6 Austin Coursey, Allan Zhang, Marcos Quinones-Grueiro, and Gautam Biswas. Hybrid control framework of uavs under varying wind and payload conditions. In *2024 American Control Conference (ACC)*. IFAC, 2024.

7 Iván García Daza, Rubén Izquierdo, Luis Miguel Martínez, Ola Benderius, and David Fernández Llorca. Sim-to-real transfer and reality gap modeling in model predictive control for autonomous driving. *Applied Intelligence*, 53(10):12719–12735, 2023. doi:10.1007/S10489-022-04148-1.

8 Tamara Regina Dieter, Andreas Weinmann, Stefan Jäger, and Eva Brucherseifer. Quantifying the simulation–reality gap for deep learning-based drone detection. *Electronics*, 12(10):2197, 2023.

9 Yu Ding, Liang Ma, Jian Ma, Mingliang Suo, Laifa Tao, Yujie Cheng, and Chen Lu. Intelligent fault diagnosis for rotating machinery using deep q-network based health state classification: A deep reinforcement learning approach. *Advanced Engineering Informatics*, 42:100977, 2019. doi:10.1016/J.AEI.2019.100977.

**10** Saite Fan, Xinmin Zhang, and Zhihuan Song. Imbalanced sample selection with deep reinforcement learning for fault diagnosis. *IEEE Transactions on Industrial Informatics*, 18(4):2518–2527, 2021. `doi:10.1109/TII.2021.3100284`.

**11** Mukesh Gautam. Deep reinforcement learning for resilient power and energy systems: Progress, prospects, and future avenues. *Electricity*, 4(4):336–380, 2023.

**12** Bryan S Guevara, Luis F Recalde, José Varela-Aldás, Victor H Andaluz, Daniel C Gandolfo, and Juan M Toibero. A comparative study between nmpc and baseline feedback controllers for uav trajectory tracking. *Drones*, 7(2):144, 2023.

**13** Peide Huang, Xilun Zhang, Ziang Cao, Shiqi Liu, Mengdi Xu, Wenhao Ding, Jonathan Francis, Bingqing Chen, and Ding Zhao. What went wrong? closing the sim-to-real gap via differentiable causal discovery. In *Conference on Robot Learning*, pages 734–760. PMLR, 2023. URL: `https://proceedings.mlr.press/v229/huang23c.html`.

**14** Fan Jiang, Farhad Pourpanah, and Qi Hao. Design, implementation, and evaluation of a neural-network-based quadcopter uav system. *IEEE Transactions on Industrial Electronics*, 67(3):2076–2085, 2019. `doi:10.1109/TIE.2019.2905808`.

**15** Bodi Ma, Zhenbao Liu, Qingqing Dang, Wen Zhao, Jingyan Wang, Yao Cheng, and Zhirong Yuan. Deep reinforcement learning of uav tracking control under wind disturbances environments. *IEEE Transactions on Instrumentation and Measurement*, 72:1–13, 2023. `doi:10.1109/TIM.2023.3265741`.

**16** Sihem Ouahouah, Miloud Bagaa, Jonathan Prados-Garzon, and Tarik Taleb. Deep-reinforcement-learning-based collision avoidance in uav environment. *IEEE Internet of Things Journal*, 9(6):4015–4030, 2021. `doi:10.1109/JIOT.2021.3118949`.

**17** Rafael Perez-Segui, Pedro Arias-Perez, Javier Melero-Deza, Miguel Fernandez-Cortizas, David Perez-Saura, and Pascual Campoy. Bridging the gap between simulation and real autonomous uav flights in industrial applications. *Aerospace*, 10(9):814, 2023.

**18** Gensheng Qian and Jingquan Liu. Development of deep reinforcement learning-based fault diagnosis method for rotating machinery in nuclear power plants. *Progress in Nuclear Energy*, 152:104401, 2022.

**19** Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021. `doi:10.1109/ACCESS.2021.3126658`.

**20** John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. `arXiv:1707.06347`.

**21** Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

**22** David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

**23** Yves Sohège, Marcos Quiñones-Grueiro, and Gregory Provan. A novel hybrid approach for fault-tolerant control of uavs based on robust reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10719–10725. IEEE, 2021.

**24** Charalambos Soteriou, Christos Kyrkou, and Panayiotis S Kolios. Closing the sim-to-real gap: Enhancing autonomous precision landing of uavs with detection-informed deep reinforcement learning. In *International Conference on Deep Learning Theory and Applications*, pages 176–190. Springer, 2024.

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

**26** Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world.

In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017. doi:10.1109/IROS.2017.8202133.

27 Daichi Wada, Sergio Araujo-Estrada, and Shane Windsor. Sim-to-real transfer for fixed-wing uncrewed aerial vehicle: pitch control by high-fidelity modelling and domain randomization. *IEEE Robotics and Automation Letters*, 7(4):11735–11742, 2022. doi:10.1109/LRA.2022.3205442.

28 Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022. doi:10.1038/S41586-021-04357-7.

29 Lin-Xing Xu, Hong-Jun Ma, Dong Guo, An-Huan Xie, and Da-Lei Song. Backstepping sliding-mode and cascade active disturbance rejection control for a quadrotor uav. *IEEE/ASME Transactions on Mechatronics*, 25(6):2743–2753, 2020.

30 Xiaolin Xu MS and Jeffrey Sun. A new trajectory in uav safety: Leveraging reinforcement learning for distance maintenance under wind variations. *Journal of Aviation/Aerospace Education & Research*, 33(4):6, 2024.

31 Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020. doi:10.1109/SSCI47803.2020.9308468.