AIAA AVIATION Forum
August 2-6, 2021, VIRTUAL EVENT
AIAA AVIATION 2021 FORUM

10.2514/6.2021-3218

A Deep Reinforcement Learning Control Strategy for Vision-based Ship Landing of Vertical Flight Aircraft

Bochan Lee*
Aerospace Engineering, Texas A&M University, College Station, TX 77843

Vishnu Saj†
Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843

Dileep Kalathil‡
Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843

Moble Benedict§
Aerospace Engineering, Texas A&M University, College Station, TX 77843

The paper discusses a deep reinforcement learning (RL) control strategy for fully autonomous vision-based approach and landing of vertical take-off and landing (VTOL) capable unmanned aerial vehicles (UAVs) on ships in the presence of disturbances such as wind gusts. The automation closely follows the Navy helicopter ship landing procedure and therefore, it detects a horizon bar that is installed on most Navy ships as a visual aid for pilots by applying uniquely developed computer vision techniques. The vision system utilizes the detected corners of the horizon bar and its known dimensions to estimate the relative position and heading angle of the aircraft. A deep RL-based controller was coupled with the vision system to ensure a safe and robust approach and landing at the proximity of the ship where the airflow is highly turbulent. The vision and RL-based control system was implemented on a quadrotor UAV and flight tests were conducted where the UAV approached and landed on a sub-scale ship platform undergoing 6 degrees of freedom deck motions in the presence of wind gusts. Simulations and flight tests confirmed the superior disturbance rejection capability of the RL controller when subjected to sudden 5 m/s wind gusts in different directions. Specifically, it was observed during flight tests that the deep RL controller demonstrated a 50% reduction in lateral drift from the flight path and 3 times faster disturbance rejection in comparison to a nonlinear proportional-integral-derivative controller.

I. Nomenclature

A	=	Action Space
a_t	=	Action at time t
b	=	Close-range controller constant
D	=	Replay Buffer
d	=	Deviation from target, meter
$de(t_k)$	=	Error difference between time t_k and t_{k1}
dt_k	=	Time difference between time t_k and t_{k1}
$e(t_k)$	=	Error at time tk
K_D	=	Derivative gain
K_I	=	Integral gain
K_P	=	Proportional gain
$L(\phi, D)$	=	Mean squared Bellman error function

*Graduate Research Assistant, Aerospace Engineering, Texas A&M University, College Station, TX 77843, AIAA member
†Graduate Researcher, Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843
‡Assistant Professor, Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843
§Associate Professor, Aerospace Engineering, Texas A&M University, College Station, TX 77843, AIAA senior member

m	=	Long-range controller constant
p	=	Indicator to check whether a terminal state or not
$Q(s,a)$	=	Q-value function
R_t	=	Cumulative Rewards
S	=	State Space
s_t	=	State at time t
$u(t_k)$	=	Control law
$V(s)$	=	Value function
x	=	Forward relative distance, meter
y	=	Sideward relative distance, meter
v_x	=	Forward relative speed, m/s
v_y	=	Sideward relative speed, m/s
γ	=	Discount Factor
μ	=	Mean value
Π	=	Policy Space
π	=	Policy
π^*	=	Optimal policy
ϕ	=	Q-value network parameter

II. Introduction

In recent years, there have been many studies focusing on the autonomous vision-based control of UAVs for ship landing, which can be used in GPS-denied environments. Using the visual information provided by vision sensors, a control system manipulates the UAV to autonomously approach and land on the moving ship deck. Various control algorithms have been proposed in the literature to address this problem such as proportional-derivative (PD) control [1, 2], proportional-integral-derivative (PID) control [3, 4], gain-scheduled PID control [5], linear quadratic regulator (LQR) [6–8], adaptive control [9–11], and model predictive control (MPC) [12]. While these approaches are partially successful in addressing the problem, they are often restricted to very specific settings due to two crucial weaknesses. Firstly, the performance of the PD/PID type controllers solely depends on the design of its gain parameters, which are typically difficult to fine-tune, especially in the simulator setting. PD/PID controllers have also limited transient response capabilities. Moreover, PD/PID controllers are not robust against uncertainties and disturbances such as wind gusts and parameter mismatches between the simulator model and the real-world system model. Secondly, MPC and LQR approaches often require a very sophisticated analytical model of the real-world UAV. Obtaining such a model can be very challenging in practice. Moreover, for computational tractability, the design of the optimal control policies using these approaches is often limited to simplified settings such as linear policy and quadratic costs. This may often result in unsatisfactory performance in the real-world setting.

In this work, we propose a deep reinforcement learning (RL)-based control algorithm for autonomous vision-based ship landing of UAVs. Using the representation power of neural networks, RL provides an attractive model-free approach for developing nonlinear control algorithms for high dimensional systems in an automated and computationally tractable way. We propose and develop a clever modification of the standard off-the-shelf RL approach to adapt it to the vision-based autonomous UAV landing problem and to make it robust against adversarial disturbances in the environments such as wind gusts.

In a prior work [13], we have addressed the autonomous vision-based ship landing problem using a heuristically designed nonlinear control strategy. The developed approach was inspired by the practical ship landing procedure that navy helicopter pilots follow. This procedure uses a gyro-stabilized (indicating the true horizon independent of ship motions) 'horizon reference bar' as a visual cue shown in Fig. 1. The pilot stabilizes the helicopter attitude using this visual cue and then commands vertical landing agnostic of the deck motion. Lee et. al developed a novel computer vision system to detect the visual cue and to estimate the distance to this cue from the helicopter [13]. Multiple processes such as image filtering, contour detection, corner points detection, and screening are conducted in this order. Also, the relative position and heading angle of the UAV with respect to the horizon bar are estimated using PnP (Perspective-n-Point) method and the accuracy is validated to sub-centimeter and sub-degree levels via the Vicon motion capture system [5]. Using the estimated positioning and heading, the nonlinear controller demonstrated both long-range and close-range tracking of a moving ship platform.

Fig. 1 Horizon reference bar

Multiple previous works have developed RL algorithms to address the autonomous landing of UAVs. RL-based high-precision, time-critical flight attitude control that could operate in unpredictable and harsh environments is discussed in [14]. It also discusses different policy gradient RL algorithms such as deep deterministic policy gradient (DDPG), proximal policy optimization (PPO), and trust region policy optimization (TRPO). An RL approach for UAV landing task on a moving platform using a variant of the DDPG algorithm is discussed in [15]. An actor-critic RL framework used to fly a UAV by following designated waypoints is presented in [16]. A variant of the DDPG algorithm is used to recover a UAV attitude quickly from an out-of-trim flight state [17]. However, these works do not address the problem of designing RL controllers that are robust against adversarial disturbances such as wind gusts. We adapt the state-of-the-art RL algorithm called twin delayed DDPG (TD3) and use the idea of domain randomization [18, 19] to make it robust. The disturbance rejection capability of the RL controller is compared to the previously developed nonlinear control system with the Kalman filter [13] through extensive simulations and flight tests. During flight testing, to verify the performance of the RL controller, wind gusts from different directions are suddenly imposed while the UAV attempts to approach and land on a sub-scale ship platform undergoing 6 degrees of freedom (DOF) deck motions. The UAV used is Parrot ANAFI quad-rotor that streams live video to a base station computer, which processes the image frames, generates control inputs, and then transmits the control commands back to the UAV through WIFI.

The following are the key contributions of this paper.

- Developed an RL-based control strategy that is robust against adversarial disturbances such as wind gusts.
- Demonstrated the superior disturbance rejection capability of the RL-based controller when strong wind gusts are suddenly imposed in different directions.
- Demonstrated the precise approach and landing on a sub-scale ship platform undergoing challenging 6-DOF deck motions using a novel approach that does not track deck movements.

The outline of the paper is as follows: Section 3 summarizes the computer vision system. In section 4, the development of a deep reinforcement learning control strategy is detailed. In section 5, the nonlinear control strategy is summarized. Extensive simulations and flight testing data are presented in sections 6 and 7, respectively. The conclusions inferred from this study are described in section 8.

III. Computer Vision System

The vision system provides the required visual information to the controller module in a given situation. The vision system is developed based on the understanding of how a helicopter pilot perceives and acts during the final approach and landing phases. Once reaching the close proximity of the ship, the horizon reference bar becomes visible, and from that point, the pilot stabilizes the helicopter by referring to the gyro-stabilized horizon bar, which remains horizontal independent of the ship motions, for a safe landing. A detailed discussion of the vision system is provided in our previous study [13]. The subsequent sections summarize the detection and estimation process. The vision system sequentially conducts the following procedures: image filtering, contour and corner detection, detected points screening, and the estimation of relative position and orientation of the UAV. The vision system has demonstrated robust detection and estimation capabilities in scenarios involving large UAV movements and different light conditions.

A. Detection of Corner Points

The image filtering steps are shown in Fig. 2. A Hue-Saturation-Value (HSV) filter is applied to sort out each green rectangle. Then, morphological opening and closing techniques are applied to remove small white patches and black voids, respectively. Finally, a watershed [20] image segmentation technique is utilized to obtain clear boundaries of the rectangles.

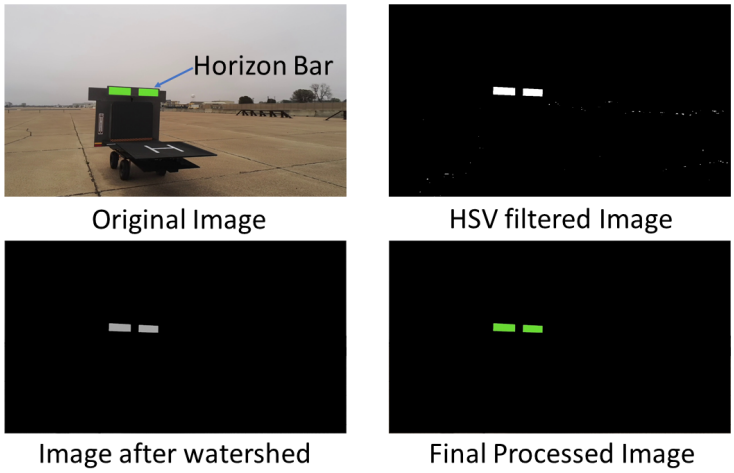


Fig. 2 Image filtering process

First, it identifies the contours of the detected region and bounds them in rectangles. The size and shape of the detected areas are very close to the green rectangles and the corners of those bounding rectangles can be used as rough estimates of the actual corners. Then, Förstner corner detection method is adopted to detect the corner points of the rectangles precisely based on the rough corners obtained by contour detection [21]. Lastly, a screening procedure is established to assure that there are no false corners present. It confirms the lengths and slopes between detected corners are within 10% and 5% of each other, respectively.

B. Estimation of Relative Position and Orientation

The estimation is based on a single camera calibration method using a planar object [22, 23]. The geometric relationship of the 2D image plane and 3D real-world coordinates is shown in Fig. 3.

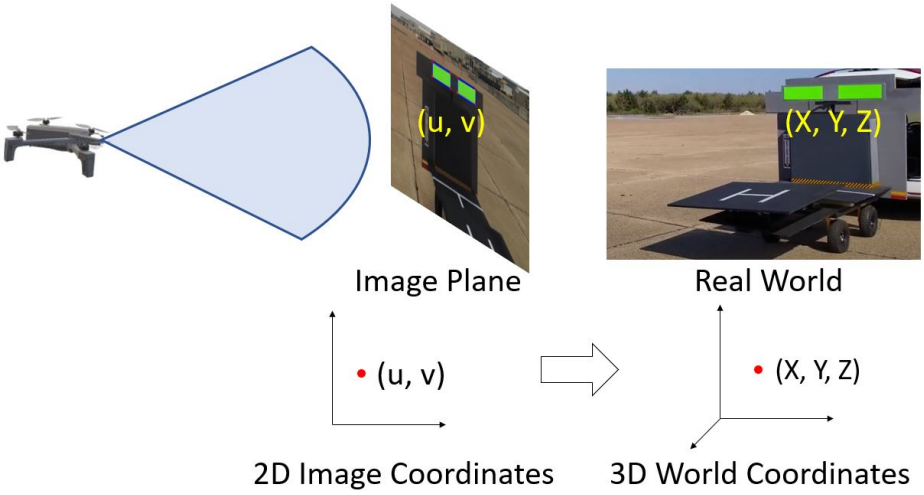


Fig. 3 Geometric relationship of 2D image coordinates and 3D real-world coordinates

The Perspective-n-Point (PnP) algorithm is implemented to determine the relative position and orientation. An iterative process is used in the PnP algorithm since it is robust for objects which consist of a planar surface. The iterative method is based on Levenberg-Marquardt optimization [24, 25]. In this method, the function minimizes re-projection error, which is the sum of squared distances between the observed image points and projected object points. The estimation method for the relative position and orientation is previously validated using a 3D motion capture system (Vicon), which demonstrated sub-centimeter and sub-degree accuracy as detailed in [5].

IV. Reinforcement Learning Control Strategy

Designing a conventional feedback control system requires a detailed model of the underlying system and computationally expensive procedures for finding a control policy that can achieve the desired objectives. Adaptive control approaches [26–28] can learn a control policy without the explicit system models. However, they require approximations of the uncertainties that can occur, and this approximation adds additional parameters that make the algorithm more complicated. The more advanced way of finding the optimal control strategy is by learning approaches. Online learning methods [29, 30] can learn the flight dynamics in real-time. However, their control capabilities are limited to the situations they experienced during the training/learning session, thus any inexperienced events can degrade the performance. There are also offline learning methods such as supervised learning [31, 32] and reinforcement learning [14, 17, 33]. The supervised learning method requires a large number of states and action data to learn the proper control strategy for different scenarios. It is difficult to obtain such a large amount of quality data and the capability of the method cannot surpass the training set's control quality since it learns from the provided data. We propose an RL-based control strategy for VTOL UAVs.

A. Reinforcement Learning Overview

Reinforcement Learning is a very intuitive state-of-the-art machine learning-based approach for finding the optimal control policy for an unknown dynamical system. In this approach, an agent interacts with an environment to learn the best action for any given state. After executing each action, the agent receives a reward and the system evolves to the next state. The primary objective of the agent is to learn the optimal policy that can maximize the cumulative rewards.

Mathematically, a policy $\pi \in \Pi$ is a function that maps state $s \in S$ to an action $a \in A$, where Π is the policy space, S is the state space, and A is the action space. The transition probability model $P(s_{t+1} \mid s_t, a_t)$ determines the dynamics of the system. At each timestep, policy π determines an action to be taken and the agent receives a reward $r(s_t, a_t)$ from the environment. The policy can be stochastic $\pi(a, s)$ (probability of taking action a in state s) as well as deterministic $\pi(s)$. The agent tries to maximize the cumulative discounted reward $R_t(s) = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$ of state s from time t to T. The T can be finite value (finite horizon episodes) or infinite value (infinite horizon episodes). The parameter γ is know as discount factor that determines the importance of future rewards. The two other important terms in RL formulation is the value function V^{π} and action-value function or Q-function Q^{π}. Value function is the expectation of discounted cumulative reward for a state s, defined as $V^{\pi}(s) = \mathbf{E}_{\pi}[R_t(s) \mid s]$. The optimal policy π^* can be expressed as one which maximizes the value function, $\pi^*(s) = \arg\max_{\pi} V^{\pi}(s)$.

B. Algorithm Selection

Robotic problems in general are associated with continuous state space and continuous action space. With the advancement in deep neural networks over the years, combining RL with deep learning shows impressive learning results. A deep version of Q-learning(DQN) algorithms has been successful in playing video games [34, 35]. Two key features of DQN are the experience replay and a target network. The experience replay stores prior experience (s, a, r, s', a') in a buffer and randomly select one for updating Q-value network. Target network helps in target remaining unchanged during several gradient updates. However, DQN algorithms were not successful in continuous action-continuous states space. Policy gradient algorithms are more useful in continuous action-continuous states space problems. Deep deterministic policy gradient (DDPG) algorithm [36] can operate over continuous action and state spaces and it uses experience replay and slow-learning neural networks of the double Q learning. Particularly for the control of a UAV, DDPG has demonstrated its successful implementations [17]. In this study, an improved version of DDPG, twin delayed DDPG (TD3) [37, 38], is selected as the RL algorithm.

TD3 learns two Q-functions Q_{ϕ_1} and Q_{ϕ_2} by mean square Bellman error minimization. It adds clipped noise to actions that make the policy difficult to extract the Q-functions. The target action is obtained as described in Eq. (1).

$$a'(s') = \text{clip}(\pi_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{\text{low}}, a_{\text{high}}), \epsilon \sim N(0, \sigma) \tag{1}$$

$\pi_{\theta_{\text{targ}}}(s')$ is the target policy and $a_{\text{low}} < a < a_{\text{high}}$ is the valid action range. This is known as target policy smoothing which essentially serves as regularizer for the algorithm. TD3 updates the policy less frequently than the Q-functions. Both Q-functions use the same target $y(r, s', p)$ (p indicates whether state s' is terminal state or not) which is the calculated using the smaller Q-value among the two Q-functions as given in Eq. (2).

$$y(r, s'p) = r + \gamma(1 - p) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s')) \tag{2}$$

The two Q-functions are then optimized by using the following two loss functions (mean squared Bellman error function) given in Eqs. (3) and (4).

$$L(\phi_1, D) = \mathbf{E}[(Q_{\phi_1}(s, a) - y(r, s', p))^2 \quad | \quad (s, a, r, s', p) \sim D] \tag{3}$$

$$L(\phi_2, D) = \mathbf{E}[(Q_{\phi_2}(s, a) - y(r, s', p))^2 \quad | \quad (s, a, r, s', p) \sim D] \tag{4}$$

ϕ_1 and ϕ_2 are network parameters and D is the replay buffer.

C. Training Procedure

The objective of the RL control strategy is for the UAV to stably hover and fly as desired against sudden wind gusts. To obtain an RL control policy that is effective in such a challenging condition, training is conducted in a realistic Gazebo simulation environment along with the precise UAV model as shown in Fig. 4.

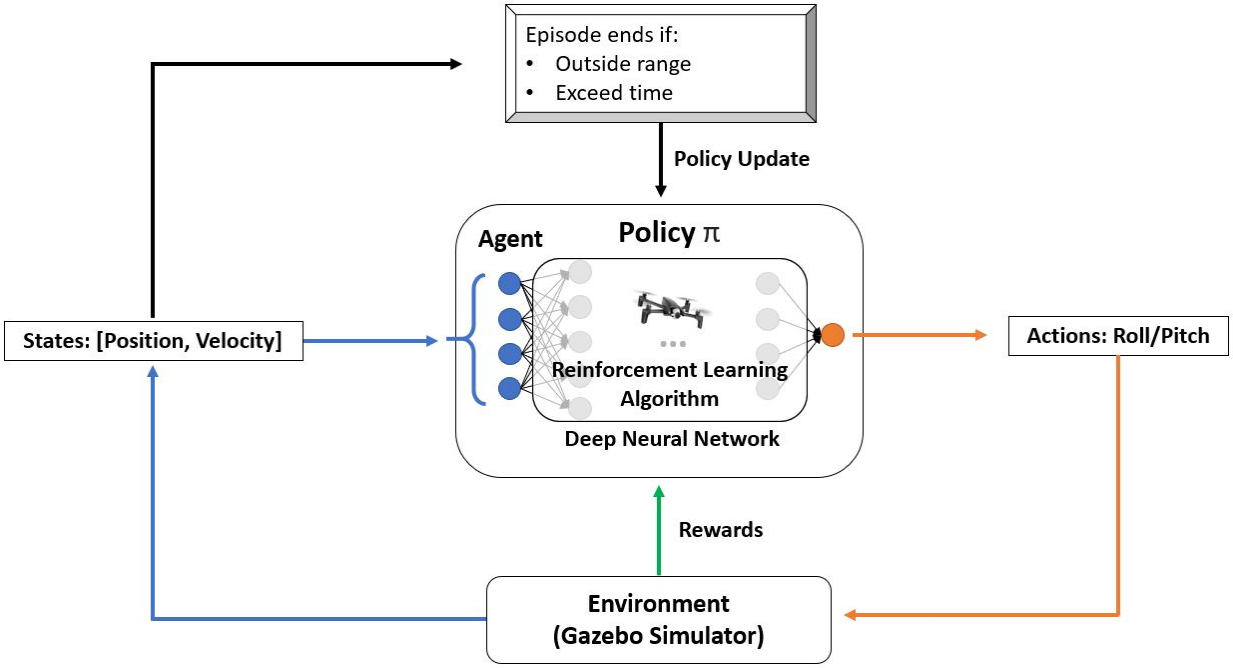


Fig. 4 Schematic showing the training process for reinforcement learning

During the training session, an agent takes some set of actions in an environment to achieve a particular task. The agent is a UAV that decides actions based on the rewards by using neural networks. The quality of those actions is quantified by user-defined rewards that are assigned in the environment and the better sets of actions to achieve the task will acquire higher cumulative rewards. A particular period that the agent takes for a set of actions by the RL control policy is called an episode. As the episode continues, the control policy keeps getting updated according to an RL algorithm and the cumulative rewards will be converged to a certain value.

To develop a successful RL control strategy, there are key requirements which are: (1) a precise model of the agent, (2) a realistic simulation engine, (3) an appropriate selection of states, (4) careful reward engineering, and (5) a powerful RL algorithm.

First, a commercial off-the-shelf (COTS) quadrotor UAV, Parrot Anafi, is used for this study. Along with the physical UAV, the precise UAV model(agent) that can be used in the Gazebo simulation is provided. Second, Gazebo is a realistic simulation engine that is widely used for robotics applications. It has closely simulated the real flight dynamics. The agent can communicate with the simulation engine using a framework called Olympe. Olympe allows the control of UAV and access to its sensors through python scripts.

1. Action Space

The UAV has four different control actions: Roll, Pitch, Yaw, and Heave. It is observed in the simulation that roll and pitch actions were heavily influenced by the wind as compared to the yaw and heave actions. Hence the roll and pitch actions are chosen to be part of the action space. It is worth noting that for the given UAV, the roll and pitch actions are independent of each other. Roll action only makes the UAV move right and left, while pitch action only makes it move forward and backward. Taking advantage of this uncoupled behavior, it is decided to split the 2-D action space into two 1-D action spaces. Two independent RL models were trained for roll and pitch controllers, respectively. This not only helps in reducing the dimension of the action space but also reduces the dimension of the state space used for the independent models. The splitting of the two controllers helps in reducing the complexity of the policy network and hence resulted in faster training and better convergence.

2. State Space

As mentioned earlier, the mission objective of the UAV is to achieve a target point and hover at that position in the presence of wind. So the position of the UAV is an obvious choice as a state. Since the action space is 1-D, the position state is also 1-D that is the position along the respective action axis for each controller. The roll controller objective is to achieve a certain target on the roll axis and maintain that position without considering the pitch motion. Similarly, for the pitch controller, the objective is to maintain a position on the pitch axis independent of the roll motion. Also, velocities are included as states, which can take an action quickly responding to velocity changes before the drift becomes large. Finally, the history of position and velocity are also selected as states for both roll and pitch controllers, which allows determining actions based on current states and trends.

To verify the effect of selected states, three different sets of states as shown in Table 1 are modeled for pitch control and roll control individually and applied for learning.

Table 1 Models with different sets of states

Model No.	Model 1	Model 2	Model 3
States for pitch control	$[x]_t$	$[x, v_x, v_y]_t$	$[(x, v_x, v_y)]_{t-i}$ where i = 0 to 5
States for roll control	$[y]_t$	$[y, v_x, v_y]_t$	$[(y, v_x, v_y)]_{t-i}$ where i = 0 to 5

x and y are the deviations along the x-axis and the y-axis, respectively. v_x and v_y are the velocities along the x-axis and the y-axis, respectively. t is the current time, thus model 3 is also taking five previous deviations and velocities as states.

During the training, the simulation engine imposes four different kinds of wind conditions: zero wind, constant magnitude wind (-10 m/s to 10 m/s), sudden wind magnitude change (-10 m/s to 10 m/s magnitude change), and a sinusoidal wind (amplitude of 5 m/s and time periods of 10 secs, 20 secs, 30 secs, 40 secs and 50 secs). These four different conditions were applied among different episodes. The idea here is that this makes the agent learn proper control actions for various wind scenarios. This is known as domain randomization and it has been an active area of research in RL [39]. It is observed that headwind affects the forward drift and crosswind affects the sideward drift. Hence crosswinds are applied for roll controller training and headwinds are applied for pitch controller training.

3. Reward Function

Reward functions are very important in the RL framework. A proper design of the reward function can yield a better convergence rate and good performance in the testing phase. A normalized reward function is opted here, which penalizes every action based on the deviation from the target point. The reward function is carefully designed to represent the control objective in numbers as shown in Eq. (5).

$$\text{Reward} = \begin{cases} -4 \times |d| & \text{if } |d| < 0.25 \\ -1 & \text{if } 0.25 \leq |d| \leq 2 \\ -(T_{\max} - T_{\text{inside}}) & \text{if } |d| > 2 \end{cases} \tag{5}$$

d denotes the deviation along the x-axis for the pitch control case and the deviation along the y-axis for the roll control case. T_{max} is the maximum episode time and T_{inside} is the time that the UAV stays within 2 meters from the hovering point. Thus, it can accumulate higher rewards (less negative) as it stays closer to the point. In other words, it will get more penalties as it deviates from the point. One episode ends if the deviation is greater than 2 meters and a new episode is started. Thus, the reward written by the time factor, $-(T_{max} - T_{inside})$, is acquired only once in the unsuccessful case. The idea here is that rather than exploring unnecessary states, it is better to give the future cumulative rewards in one step. This helps in maintaining the consistency of episodic rewards. In each episode, the UAV is spawned at a random position near the origin. Apart from the other common RL control strategies, the episode does not end even though the UAV achieves the target position. Instead, it is designed to continue maintaining the position until the end of the episode time, which can allow sufficient time to learn the proper control actions.

Successful training is heavily dependent on the selection of an RL algorithm. One of the modern policy gradient algorithms, TD3 is used for the study. The applied hyper-parameters that control the learning process are given in Table 2.

Table 2 Selected hyper-parameters for training

gamma	0.99	policy delay	2
learning rate	0.0001	target policy noise	0.2
buffer size	100000	target noise clip	0.5

4. Convergence

The training convergences of pitch and roll control cases are shown in Figs. 5 and 6, respectively.

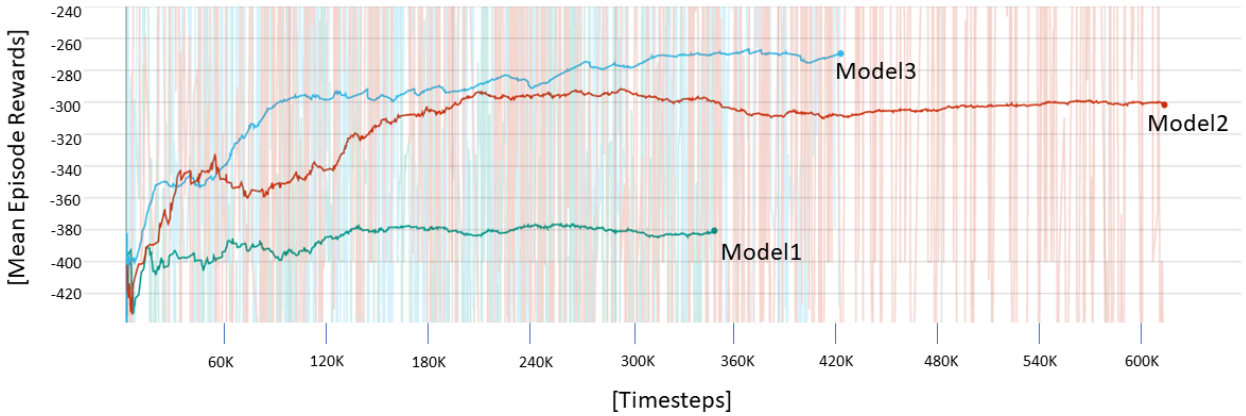


Fig. 5 Training convergence of different models for the pitch control case

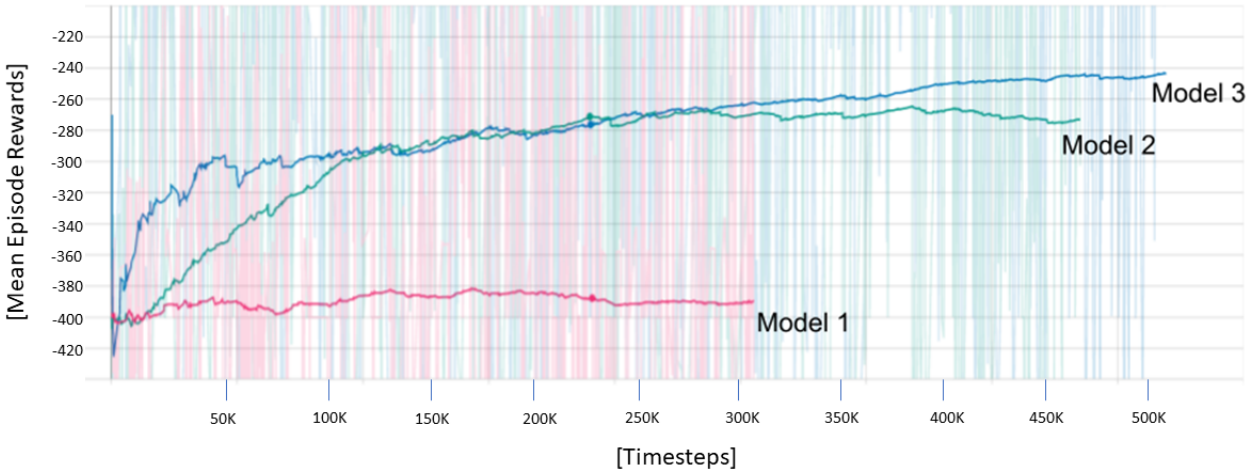


Fig. 6 Training convergence of different models for the roll control case

In both the pitch and roll control cases, model 3 converges to the higher accumulated reward value and learns faster than the other models on average. It demonstrates that taking previous positions and velocities as states is helpful, which is designed based on the intuition that neural networks can decode the wind gust information from those states. The obtained control policy is used in the simulations and flight tests.

The system used for training is LENOVO Legion Y740-15IRH, which is composed of Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2592 Mhz, 6 Cores, and 12 Logical Processors. It features an integrated NVIDIA GeForce GTX 1660 Ti 6GB Graphics and 8GB of LPDDR4 memory with a 128-bit interface. The Ubuntu 18.04 OS with Nvidia driver version 440 and CUDA version 10.2 are used. The same system is used for flight simulations and testing.

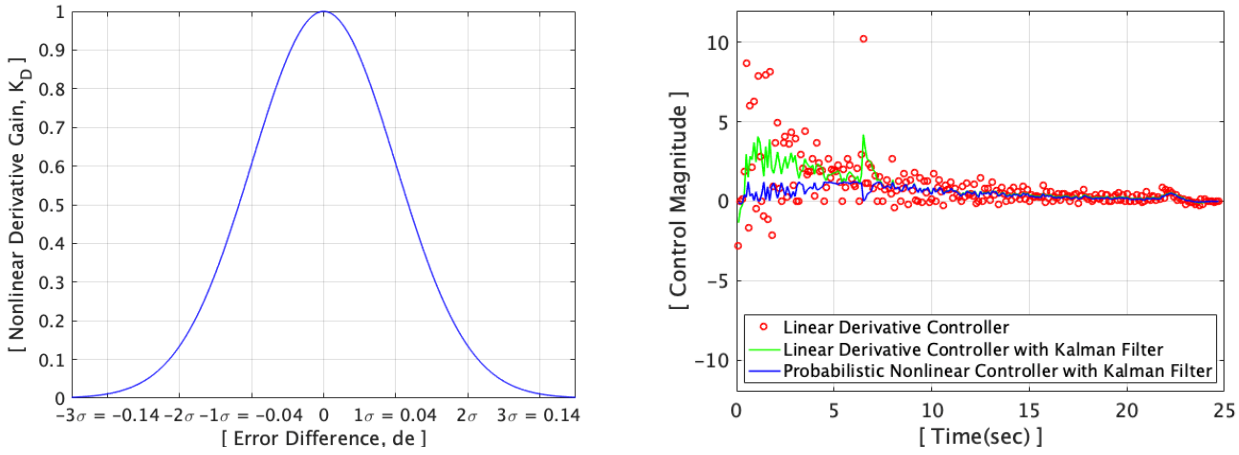
V. Nonlinear Control Strategy

A nonlinear PID control strategy was previously developed for ship landing that demonstrated good tracking capability [13]. In this study, the PID control was compared with the deep RL control strategy. Some key design aspects of the nonlinear control strategy are summarized here.

To take advantage of the derivative controller with minimum noise, the Kalman filter and nonlinear derivative controller are designed. A single state Kalman filter is applied with the unity feedback, which means that it does not require the prediction from the model. In addition, to reject the intermittent unrealistic estimation effectively, a nonlinear derivative controller with linear proportional and integral controllers is designed by utilizing the normal distribution concept as described in Eq. (6).

$$
\begin{aligned}
de(t_k) =& e(t_k) - e(t_{k-1}) \\
u(t_k) =& K_P e(t_k) + K_I \sum_{k=1}^{k} e(t_k) dt_k + K_D\{de(t_k)\}\frac{de(t_k)}{dt_k} K_D\{de(t_k)\} = \quad b e^{-0.5\frac{(de(t_k)-\mu)^2}{\sigma^2}}
\end{aligned}
\tag{6}
$$

$de(t_k)$ is an error difference between time t_k and t_{k-1}, and $u(t_k)$ is a control law that has linear proportional and integral terms, as well as the nonlinear derivative term. K_P and K_I are constant proportional and integral gains, and $K_D\{de(t_k)\}$ is the nonlinear derivative gain that is a function of error difference, $de(t_k)$. The probabilistic nonlinear derivative controller is constructed by taking the exponential term from the probability density function and multiplying it with the constant b that determines the control magnitude. Based on the observation of aircraft movement, σ is determined as 0.04, which means the distance that the aircraft can move during 0.03 seconds (one time step) has a 68.2 percent chance of being within 4 cm and a 95.4 percent chance of being within 8 cm. When b is 1 and μ is 0, the probabilistic nonlinear derivative controller computes derivative gains as shown in Fig. 7a. In this way, the controller does not respond to the large noise in the error data, which can trigger undesired and unstable maneuvers. Also, the magnitude of gain can vary according to the selection of constant b. The effects of the Kalman filter and the probabilistic nonlinear controller are shown in Fig. 7b.



(a) Probabilistic nonlinear derivative controller

(b) Effect on control input

Fig. 7 Probabilistic nonlinear derivative controller and its effect on control input

The red circles denote the control magnitude of the linear derivative controller without the Kalman filter and the noise is clearly evident. The green line is the result after applying the Kalman filter to the linear derivative controller. The noise is reduced, however, it still yields large control inputs in response to the incorrect estimation values. The blue line shows the control inputs generated by the probabilistic nonlinear controller with the Kalman filter and these inputs are relatively small and insensitive to the large unrealistic error differences. At the 6.5 second mark, the error difference *de* has a large value caused by incorrect estimation. In this case, the linear derivative controller with the Kalman filter reduced the magnitude to some extent but it is still affected by that particular spurious error value. However, the probabilistic nonlinear controller with the Kalman filter is able to screen out the wrong value, thereby minimizing the effect of incorrect error estimation. The finally selected gains through extensive flight tests and simulations are specified in Table 3.

Table 3 Selected gains for close-range controllers

Flight Mode	Controller	K_P	K_I	K_D		
				b	μ	σ
Corner Points Tracking	pitch	7.5	0.05	4.5	0.02	0.04
	roll	7.5	0.01	8.5	0	0.04
	heave	15	0.01	7	0	0.02
	yaw	5.5	0.05	1.75	0	5

VI. Simulation Results

In order to verify the disturbance rejection capability in challenging situations, sudden wind gusts are imposed while the UAV is hovering. 5 m/s diagonal wind (45°) and cross-wind (90°) based on UAV heading are simulated using a step function. The performances of the previously developed nonlinear PID-based feedback control system with the Kalman filter and the deep RL control strategy are compared based on the sideward and forward drifts. The pitch and roll control inputs are also examined. The simulated UAV and Gazebo simulation program is shown in Fig. 8.

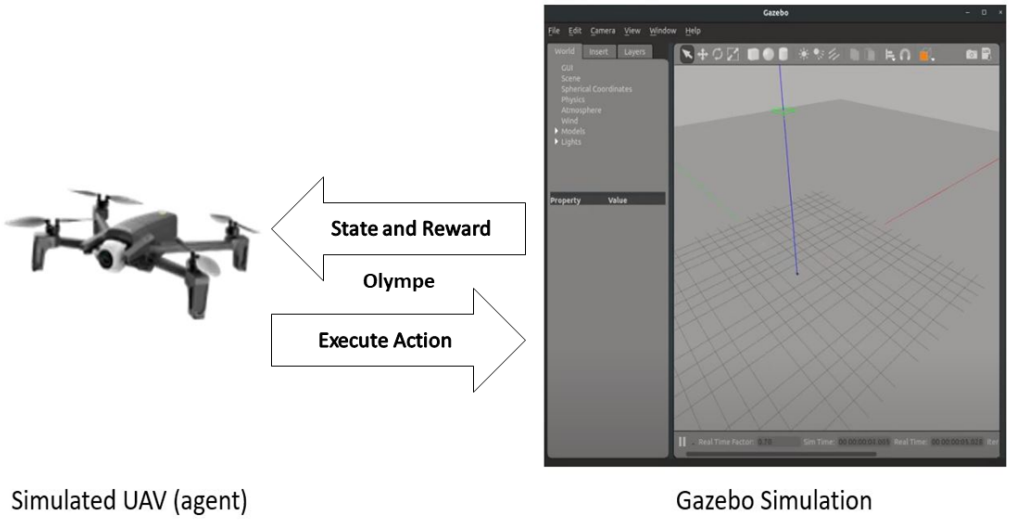
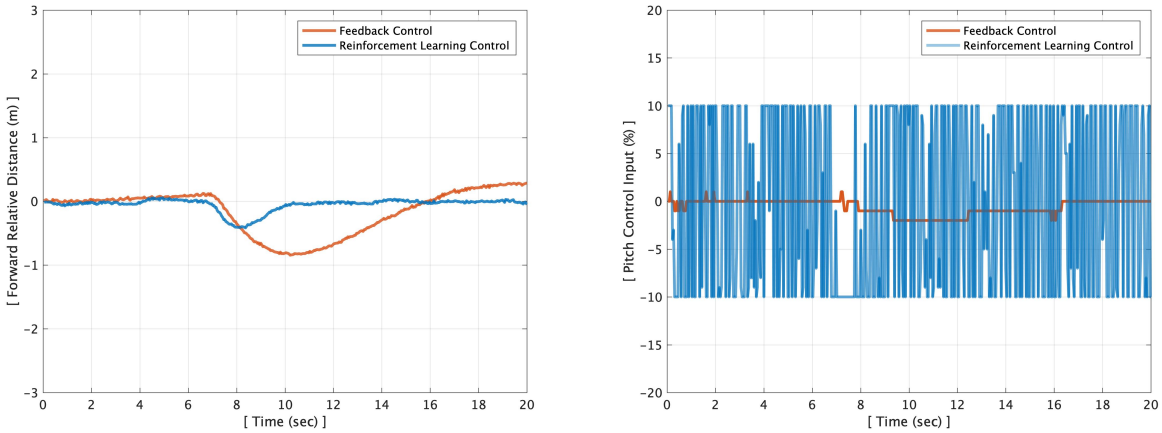


Fig. 8 Simulated UAV (agent) and Gazebo simulation environment

A. Case-1: Hovering at 5 m/s of diagonal wind gust (45°)

The sudden diagonal wind is imposed approximately at the 7 second mark, which affects the forward and sideward drifts equally. The forward drift from the hovering point and pitch control inputs in the case of the nonlinear PID controller and RL controller are shown in Fig. 9. The RL control regulates the impact of wind gust within 2 seconds with a maximum forward drift of 0.4 meters, which is 2 times smaller drift and 5.5 times faster settling time than the nonlinear PID control system. The RL control strategy yields high-frequency pitch commands via the trained control policy, which rejects the disturbance more effectively than the nonlinear PID control system.

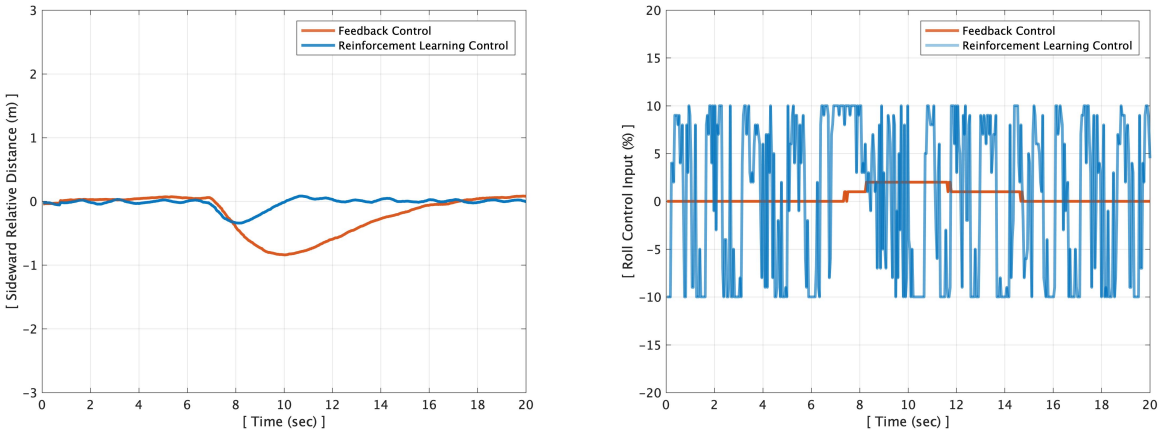


(a) Forward drift

(b) Pitch control inputs

Fig. 9 Forward drift and pitch control inputs for case-1

The sideward drift from the hovering point and roll control inputs in the case of the nonlinear PID control and RL control are shown in Fig. 10. The roll response looks similar to pitch, which is expected for hovering flight because of the symmetry.



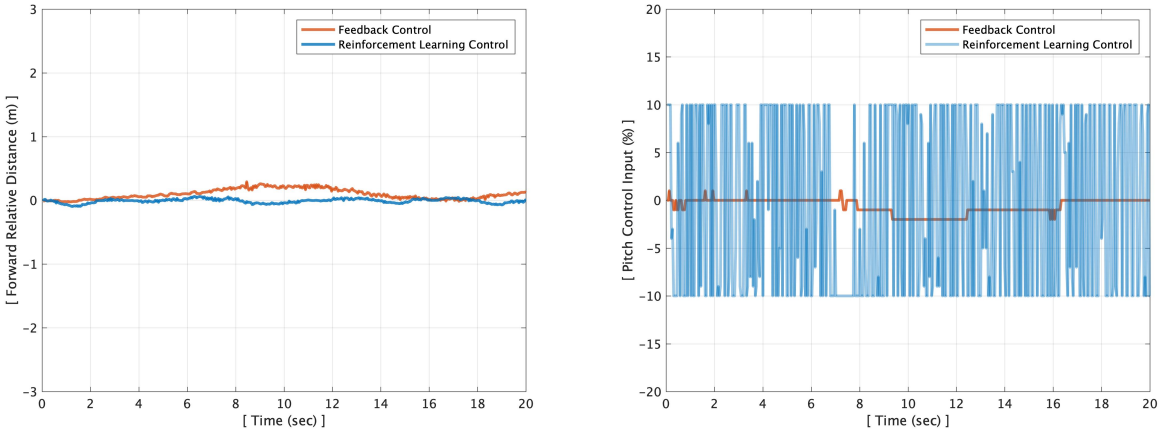
(a) Sideward drift (b) Roll control inputs

Fig. 10 Sideward drift and roll control inputs for case-1

The RL control regulates the effect of wind gust within 3 seconds with a maximum sideward drift of 0.3 meters, which is 3 times smaller drift and 3 times faster settling time than the nonlinear PID control system. Similar to the pitch commands, the RL control system generates high-frequency roll commands that effectively reject the disturbances better than the nonlinear PID control system.

B. Case-2: Hovering at 5 m/s of crosswind gust (90°)

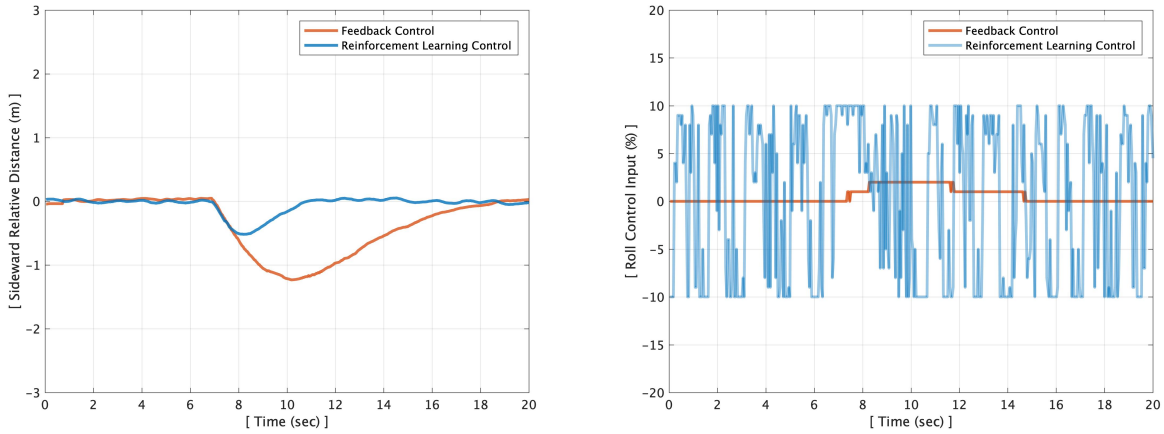
In this case, a sudden crosswind is imposed approximately at the 7-second mark. This will have a significant impact on the sideward drift. The forward drift from the hovering point and pitch control inputs in the case of the nonlinear PID control and RL control are shown in Fig. 11. Since the wind gust direction is orthogonal to the UAV heading, its effect on forward distance is small.



(a) Forward drift (b) Pitch control inputs

Fig. 11 Forward drift and pitch control inputs for case-2

The sideward drift from the hovering point and roll control inputs in the case of the nonlinear PID control and RL control are shown in Fig. 12.



(a) Sideward drift

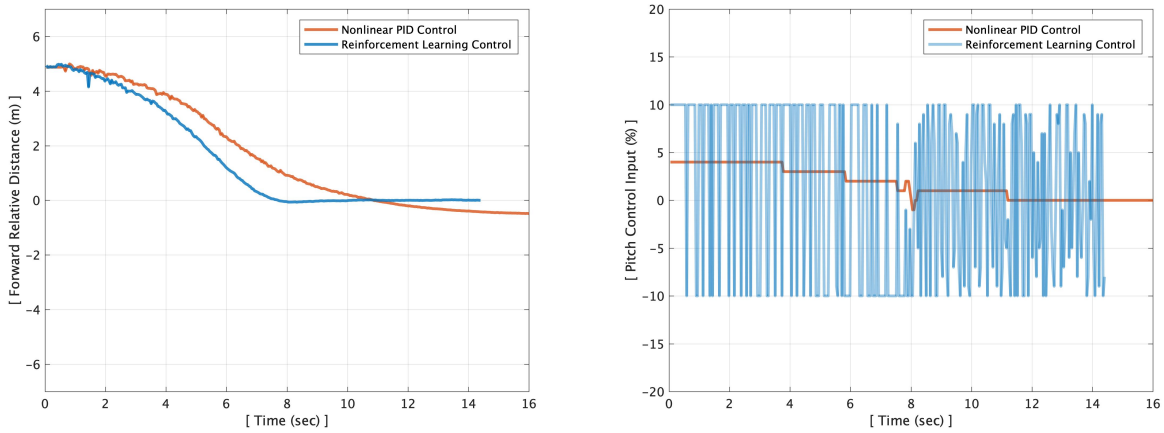
(b) Roll control inputs

Fig. 12 Sideward drift and roll control inputs for case-2

Compared to the control inputs commanded by the nonlinear PID control inputs, the RL control quickly responds to the crosswind gust with higher control magnitude, which demonstrates improved disturbance rejection capability. The RL control regulates the wind gust within 3 seconds with the maximum sideward drift of 0.4 meters, which is 2 times smaller drift and 3 times faster settling time than the nonlinear PID control system.

C. Case-3: Tracking at 5 m/s of diagonal wind gust (45°)

A 5 m/s of sudden diagonal wind is imposed approximately at the 5-second mark while the UAV approaches the ship platform. The forward relative distance between the UAV and landing deck center and the pitch control inputs in the case of the nonlinear PID control and RL control are shown in Fig. 13.

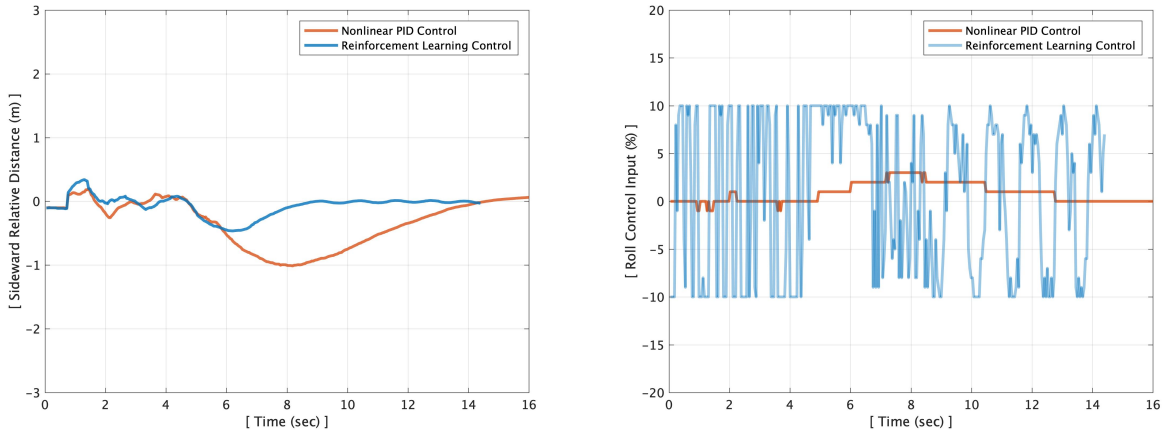


(a) Forward drift

(b) Pitch control inputs

Fig. 13 Forward drift and pitch control inputs for case-3

Since the initial forward relative distance of 4.5 meters is dominant, the drift by the wind gust is not clearly seen. The RL control inputs yielded by the control policy using deep neural networks are not fully describable, however, it is obvious that this control strategy demonstrates faster and accurate tracking than the nonlinear PID control system. The sideward drift during approach in the presence of diagonal wind gust and the roll control inputs in the case of the nonlinear PID control and RL control are shown in Fig. 14.



(a) Sideward drift

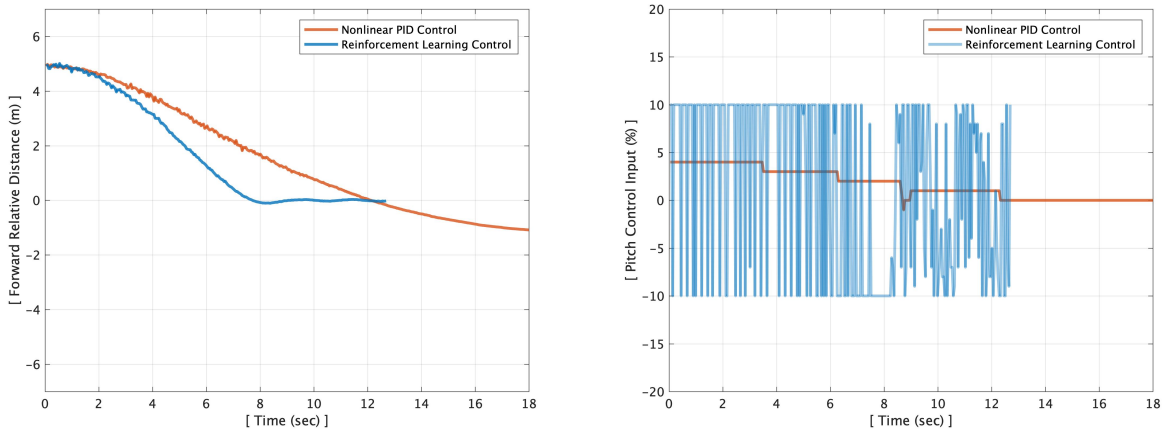
(b) Roll control inputs

Fig. 14 Sideward drift and roll control inputs for case-3

The RL control is able to adjust and stabilize the UAV in the wind gust within 3 seconds with the maximum sideward drift of 0.45 meters, which is 2 times smaller drift and 3 times faster settling time than the nonlinear PID control system.

D. Case-4: Tracking at 5 m/s of crosswind gust (90°)

At approximately the 5 second mark, a sudden crosswind is imposed which significantly affects the sideward drift and has minimum impact on the forward drift. The forward relative distance between the UAV and landing deck center and pitch control inputs for nonlinear PID control and RL control are shown in Fig. 15.

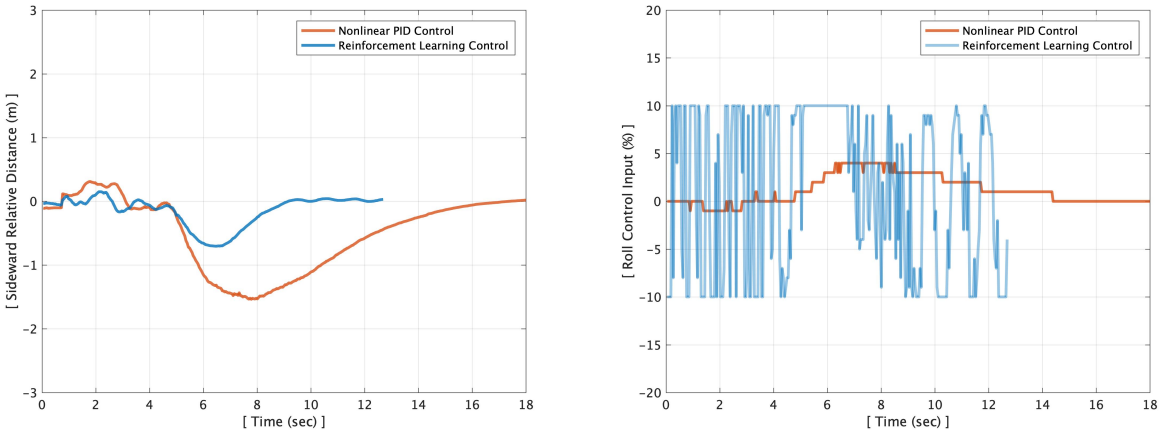


(a) Forward drift

(b) Pitch control inputs

Fig. 15 Forward drift and pitch control inputs for case-4

Since the wind gust direction is orthogonal to the UAV heading, its effect on forward distance is small. The RL control demonstrates 1.5 times faster tracking capability without overshoot than the nonlinear PID control system. The sideward drift during approach and roll control inputs in the case of the nonlinear PID control and RL control are shown in Fig. 16.



(a) Sideward drift

(b) Roll control inputs

Fig. 16 Sideward drift and roll control inputs for case-4

The RL control regulates the wind gust within 3 seconds with the maximum sideward drift of 0.7 meters, which is 2 times smaller drift and 3 times faster settling time than the nonlinear PID control system.

Since the RL control strategy computes control inputs (actions) from states, which include the current and 5 previous positions and velocities using deep neural networks, it is impossible to explicitly describe the relationship between the states and actions. As seen from the simulations, the resultant control inputs are significantly different from the control inputs from the nonlinear PID, which is an explicit control algorithm. However, this completely new way of controlling the UAV using the RL controller clearly demonstrates superior disturbance rejection capability than the nonlinear PID control system.

VII. Flight Testing

Flight tests were conducted to demonstrate the disturbance rejection capability when the 5 m/s of the cross and diagonal wind gust were imposed by a drum fan, and they were recorded as shown in video [40]. The UAV has an embedded (onboard) inner-loop autopilot that controls the rotational speed of each propeller to achieve the commanded inputs generated by the outer-loop vision-based control system (offboard).

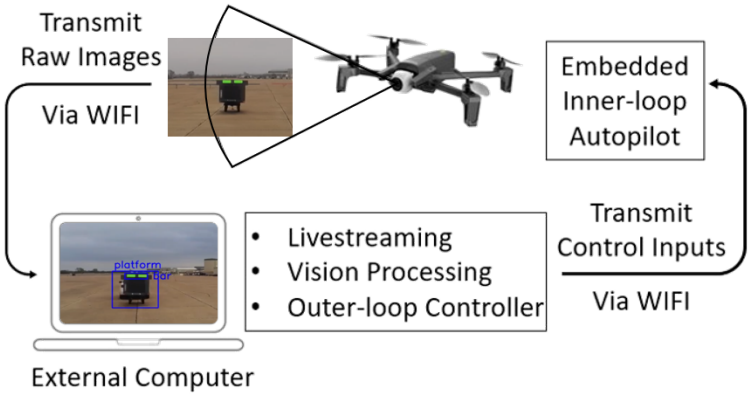


Fig. 17 Schematic showing the autonomous flight control system

The UAV is controlled by a Python script that runs on an external base station computer. The external computer communicates with the UAV through the WIFI connection. The UAV transmits raw images captured by the onboard camera to the computer in real-time. Then, the computer processes the images to provide perceived visual information. The image resolution used is 1280 x 720p and this affects the effective range for detection and estimation. Once the perceived visual information is fed into the feedback controller, it yields the corresponding command inputs which are roll, pitch, throttle (heave), and yaw. The commands are sent back to the UAV and then the embedded inner-loop autopilot controls the rotating speed of each propeller to achieve the commanded inputs.

A. Experimental Setup

The prescribed motion is the Navy helicopter ship landing limits defined by NATOPS. The ship motion limits for landing are set as ± 8° of roll and ± 3° of pitch. Even though the limits are defined by the maximum roll and pitch magnitudes, the frequency of the motion is also a crucial factor. According to the report for a similar size ship (length: 152.4 m, width: 15.2 m) motions conducted by the Sandia National Laboratory [41], the roll period is 10.1 seconds and the pitch period is 6.5 seconds as shown in Table 4.

Table 4 Typical ship characteristics

Ship Type	Length (m)	Width (m)	Roll Period (secs)	Pitch Period (secs)
Destroyer	152.4	15.2	10.1	6.5
Aircraft Carrier	304.8	38.1	15.8	8.8

A sub-scale ship platform is constructed including the horizon bar and motion deck as shown in Fig. 18. The width, height, and length of the ship platform are 5 ft, 5 ft, and 10 ft, respectively. The horizon bar always indicates a perfect horizon, and the motion deck has its own 6 DOF motions in addition to the forward translational motion, which is similar to what would be experienced on a real ship. This setup has been used for the nonlinear control system and RL-based control system.

Fig. 18 Constructed ship platform with horizon bar and motion deck

In addition, as shown in Fig. 19, a drum fan is installed for generating the wind gust to demonstrate the disturbance rejection capability of the developed RL control strategy. The fan could generate a wind gust with a maximum speed of 5 m/s in any direction relative to the UAV heading and flight path.

(a) Diagonal wind testing setup (b) Crosswind testing setup

Fig. 19 Disturbance rejection experimental setup

B. Disturbance rejection for diagonal wind gust case

Flight tests are conducted to demonstrate the disturbance rejection capability of the RL control strategy when the wind gust is suddenly imposed. During flight testing, the fan generates a 5 m/s wind gust at an angle of 45 degrees with respect to UAV heading while the UAV tries to autonomously approach and land on the ship platform. The performances of the developed nonlinear PID-based feedback control system with the Kalman filter and the deep RL control strategy are compared based on the sideward drift. The control inputs and the sideward drift using the RL control strategy are also examined. The sideward drift while using the nonlinear PID control and the RL control for the diagonal wind gust case is shown in Fig. 20.

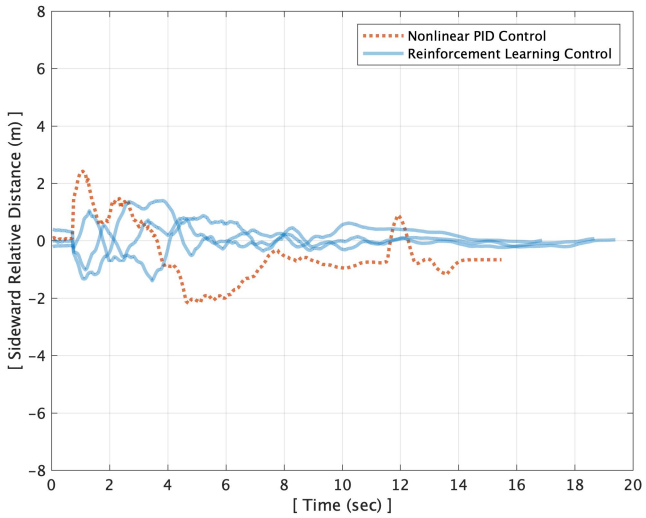
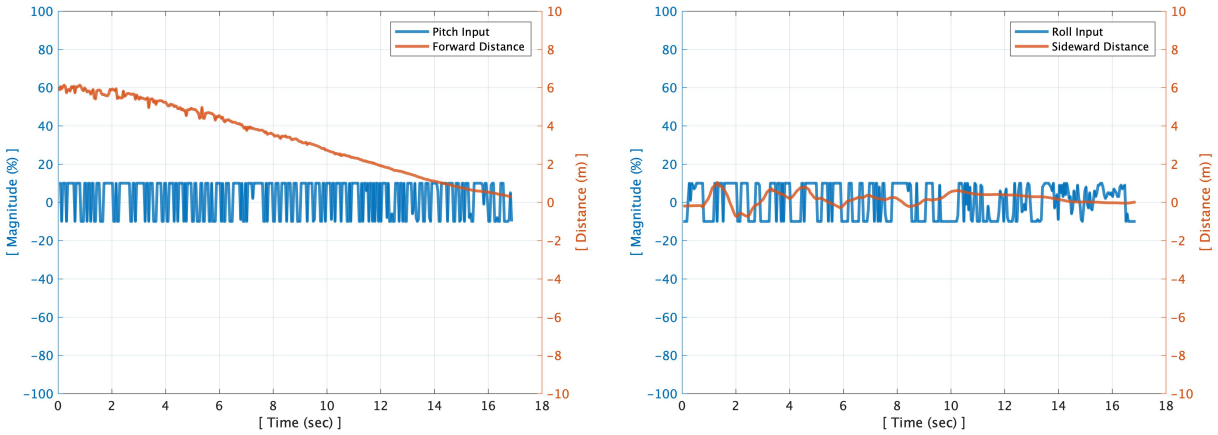


Fig. 20 Sideward drift for the 5 m/s diagonal wind gust case

The flight tests using the nonlinear PID control experience more than 2 meters of sideward drift and fail to land on the landing deck. However, the RL control strategy demonstrates less than 1.5 meters of sideward drift and successfully lands on the deck repeatedly.

The relative distances and control inputs of successful RL control strategy are shown in Fig. 21. RL controller demonstrates robust tracking by rejecting the diagonal wind gust effectively from about 6 meters distance from the landing deck. Also, it keeps the sideward drift below 1 meter during approach and landing by actively commanding the roll control inputs of high magnitude and frequency. The trained control policy yields control inputs by deep neural networks that consider the current and 5 previous positions and velocities. Therefore, for an RL controller, it is not possible to explicitly correlate the relationship between the states and actions. However, the flight tests clearly

demonstrated that the RL control effectively rejected the disturbance while approaching and landing.



(a) Forward relative distances and pitch control inputs

(b) Sideward relative distances and roll control inputs

Fig. 21 Disturbance rejection of reinforcement learning control for the 5 m/s diagonal wind gust case

C. Disturbance rejection for crosswind gust case

Flight tests are also conducted to demonstrate the disturbance rejection capability of the developed RL control strategy in a more challenging condition. During these flight tests, the fan is installed perpendicular to the approach course so that it experiences a sudden 5 m/s crosswind gust while approaching and landing on the ship platform. The performance of the deep RL-based control strategy is compared with the nonlinear PID and Kalman filter-based feedback control system in terms of the sideward drift and control inputs. The sideward drift of the nonlinear PID control and RL control in the sudden crosswind case is shown in Fig. 22.

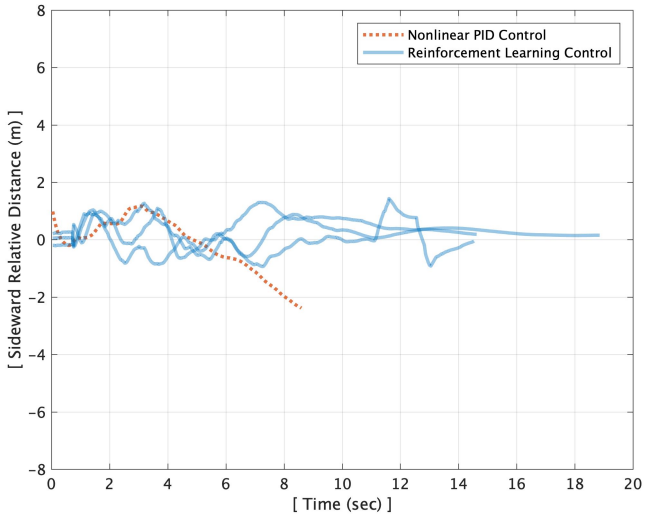
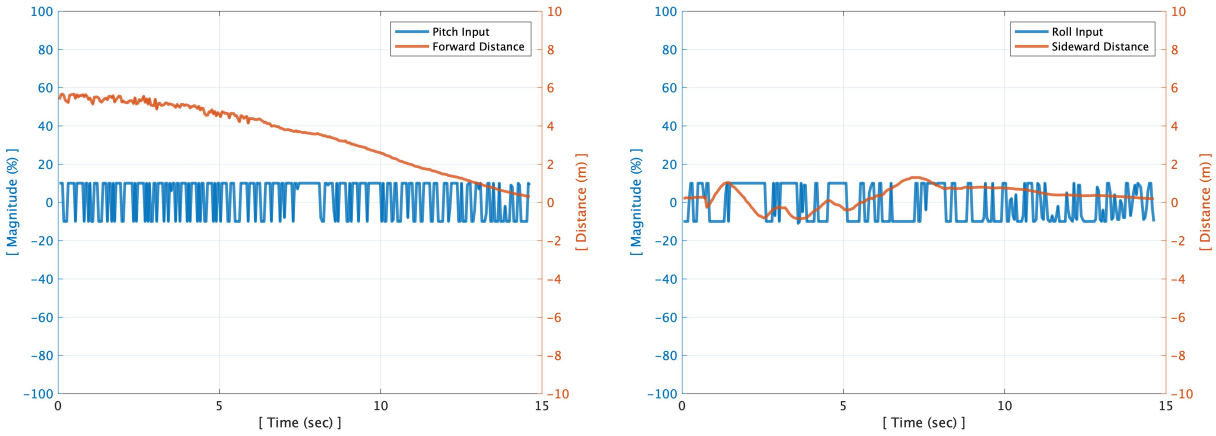


Fig. 22 Sideward drift for the 5 m/s crosswind gust case

This sudden crosswind affects the sideward drift in both cases. The nonlinear PID control fails to reject the disturbance quickly enough, and therefore, the UAV loses the visual cue at approximately 8.5 seconds. On the other hand, the RL control keeps the sideward drift below 1.5 meters and successfully lands on the deck repeatedly.

The relative distances and control inputs of the RL control strategy are shown in Fig. 23. The RL controller demonstrates robust tracking by rejecting the crosswind gust effectively at 6 meters distance from the landing deck. It has a limited effect on forward drift but a distinguishable effect on sideward drift due to the direction of the wind. The pitch control inputs are mainly commanded to move forward in such a wind condition and the roll control inputs are commanded to minimized the sideward drift caused by the wind gust. It kept the sideward drift below 1.3 meters during the approach and landing phases.



(a) Forward relative distances and pitch control inputs

(b) Sideward relative distances and roll control inputs

Fig. 23 Disturbance rejection of reinforcement learning control the 5 m/s crosswind gust case

D. Landing Accuracy

The final landing points are also collected from the multiple flight tests in the presence of wind gust and using the RL control strategy. While approaching and landing, 5 m/s diagonal wind (45°) and cross-wind (90°) are suddenly imposed by the fan (Fig. 19). The landing points are distributed within the set landing threshold as shown in Fig. 24.

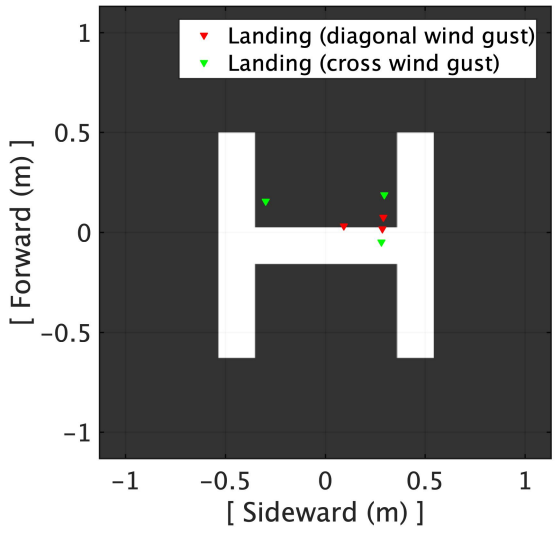


Fig. 24 Landing points on the deck for the different wind gust flight test cases

VIII. Conclusion

The goal of this study is to improve the tracking and disturbance rejection capability of a VTOL UAV while attempting to land on a ship deck by closely following the Navy helicopter ship approach and landing procedure. The automation has been achieved by using a single onboard camera without using GPS. The developed autonomous ship landing system consists of a computer vision system that estimates the relative distance and orientation based on the detected horizon bar corner points and a deep reinforcement learning control strategy for performing robust approach and landing maneuvers. The control policy that uses deep neural networks to generate control inputs was obtained by training in the Gazebo simulation program using the TD3 algorithm, the selected set of states, and the designed reward function.

Extensive simulations and multiple flight tests were systematically conducted to verify the tracking and disturbance rejection capability as well as the landing accuracy. The successfully trained control policy was directly implemented on a quadrotor UAV and flight tests were conducted. A 5 m/s of sudden wind gust was imposed from diagonal and orthogonal directions during the approach and landing on a sub-scale ship platform. The moving ship platform simulated the ± 8° of roll and ± 3° of pitch with the 10.1 seconds of roll period and 6.5 seconds of pitch period, which are derived from the helicopter ship landing limits defined by NATOPS and Sandia National Laboratory report. The developed RL control demonstrated superior disturbance rejection capability compared to a nonlinear PID control system.

Some of the key takeaways from this study are enumerated below.

1) The proposed VTOL UAV vertical landing approach, which is agnostic to the ship deck motions, was verified as a safe way of landing. This was demonstrated through multiple flight tests using a sub-scale landing platform mimicking challenging ship motions derived from the NATOPS helicopter ship landing operational limits and Sandia National Laboratory report (roll: ±8, pitch: ±3, roll period: 6.5 secs, pitch period: 10.1 secs).
2) The roll and pitch control policies are individually trained by the TD3 RL algorithm. Using this approach, the control action taken by the deep neural networks was set as only one action (pitch or roll), which significantly reduced the training time and converged to higher a cumulative reward at a faster rate. This was only possible because roll and pitch control are uncoupled for a quad-rotor UAV.
3) The optimal set of states were identified by comparisons of different models. Out of the three models analyzed, the model which used the history of position and velocity as states has the best mean episodic reward convergence.
4) The normalized negative reward function that yields a value between -1 and 0 was finally derived after multiple attempts with different reward functions. Also, it was designed to receive a high negative reward only if the UAV drifts out of the boundary that is 2 meters away from the setpoint. It was learned that even a high reward value for successful action and a wide range of rewards could overestimate the actions taken and thus the training could fail. The successful training was heavily dependent on the reward function design.
5) To learn the proper actions by the TD3 RL algorithm, various kinds of wind conditions were imposed as the training episodes continued. This was done as a part of domain randomization. The final trained control policy obtained robust flight capability in challenging conditions.
6) In comparison with the nonlinear PID control system, the deep RL control demonstrated superior disturbance rejection capability, which reduced lateral drift by 50% at a 3 times faster rate. This is a crucial improvement that can ensure a safe approach and landing at the proximity of the ship where the airflow is highly turbulent and unpredictable.
7) The precise landing capability of the RL control strategy in 5 m/s of sudden wind gusts was verified through the flight tests. The UAV consistently landed within the set landing boundary of 35 x 35 cm from the deck center which was defined considering the UAV and deck size, and safety margin.

In summary, the results demonstrate the capability of the deep RL control strategy for autonomous ship approach and landing for VTOL UAVs in the presence of disturbances such as wind gusts. This is a significant accomplishment since there are no known efforts in the literature, which demonstrated the disturbance rejection capability in the realistic ship landing situation while following the real helicopter ship landing procedure. The study showed that this RL-based methodology, if further developed, has the potential to significantly improve aircraft survivability in highly challenging and unpredictable ship landing situations.

Acknowledgments

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1946890, along with significant contributions from C-UAS industry members.

References

[1] Daly, J. M., Ma, Y., and Waslander, S. L., "Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays," *Autonomous Robots*, Vol. 38, No. 2, 2015, pp. 179–191.

[2] Holmes, W. K., and Langelaan, J. W., "Autonomous ship-board landing using monocular vision," *Proc. 72nd Am. Helicopter Soc Forum*, Vol. 2, 2016, p. 36.

[3] Araar, O., Aouf, N., and Vitanov, I., "Vision based autonomous landing of multirotor UAV on moving platform," *Journal of Intelligent & Robotic Systems*, Vol. 85, No. 2, 2017, pp. 369–384.

[4] Truong, Q. H., Rakotomamonjy, T., Taghizad, A., and Biannic, J.-M., "Vision-based control for helicopter ship landing with handling qualities constraints," *IFAC-PapersOnLine*, Vol. 49, No. 17, 2016, pp. 118–123.

[5] Lee, B., Saj, V., Benedict, M., and Kalathil, D., "A Vision-Based Control Method for Autonomous Landing of Vertical Flight Aircraft On a Moving Platform Without Using GPS," *arXiv preprint arXiv:2008.05699*, 2020.

[6] Lee, B., "Helicopter Autonomous Ship Landing System," Master's thesis, Texas A&M University, 2018.

[7] Lee, B., and Benedict, M., "Development and Validation of a Comprehensive Helicopter Flight Dynamics Code," *AIAA Scitech 2020 Forum*, 2020, p. 1644.

[8] Ghamry, K. A., Dong, Y., Kamel, M. A., and Zhang, Y., "Real-time autonomous take-off, tracking and landing of UAV on a moving UGV platform," *2016 24th Mediterranean conference on control and automation (MED)*, IEEE, 2016, pp. 1236–1241.

[9] Hu, B., Lu, L., and Mishra, S., "Fast, safe and precise landing of a quadrotor on an oscillating platform," *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 3836–3841.

[10] Kim, J., Jung, Y., Lee, D., and Shim, D. H., "Landing control on a mobile platform for multi-copters using an omnidirectional image sensor," *Journal of Intelligent & Robotic Systems*, Vol. 84, No. 1, 2016, pp. 529–541.

[11] Xia, K., Lee, S., and Son, H., "Adaptive control for multi-rotor UAVs autonomous ship landing with mission planning," *Aerospace Science and Technology*, Vol. 96, 2020, p. 105549.

[12] Vlantis, P., Marantos, P., Bechlioulis, C. P., and Kyriakopoulos, K. J., "Quadrotor landing on an inclined platform of a moving ground vehicle," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 2202–2207.

[13] Lee, B., Saj, V., and Benedict, M., "Machine Learning Vision and Nonlinear Control Approach for Autonomous Ship Landing of Vertical Flight Aircraft," *Proceedings of the 77th Annual Forum*, The Vertical Flight Society, Virtual, 2021.

[14] Koch, W., Mancuso, R., West, R., and Bestavros, A., "Reinforcement learning for UAV attitude control," *ACM Transactions on Cyber-Physical Systems*, Vol. 3, No. 2, 2019, pp. 1–21.

[15] Rodriguez-Ramos, A., Sampedro, C., Bavle, H., De La Puente, P., and Campoy, P., "A deep reinforcement learning strategy for UAV autonomous landing on a moving platform," *Journal of Intelligent & Robotic Systems*, Vol. 93, No. 1-2, 2019, pp. 351–366.

[16] Li, Y., Li, H., Li, Z., Fang, H., Sanyal, A. K., Wang, Y., and Qiu, Q., "Fast and accurate trajectory tracking for unmanned aerial vehicles based on deep reinforcement learning," *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, IEEE, 2019, pp. 1–9.

[17] Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M., "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, Vol. 2, No. 4, 2017, pp. 2096–2103.

[18] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., "Domain randomization for transferring deep neural networks from simulation to the real world," *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2017, pp. 23–30.

[19] Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P., "Sim-to-real transfer of robotic control with dynamics randomization," *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 3803–3810.

[20] Couprie, M., and Bertrand, G., "Topological gray-scale watershed transformation," *Vision Geometry VI*, Vol. 3168, International Society for Optics and Photonics, 1997, pp. 136–146.

[21] Förstner, W., and Gülch, E., "A fast operator for detection and precise location of distinct points, corners and centres of circular features," *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, Interlaken, 1987, pp. 281–305.

[22] Zhang, Z., "Flexible camera calibration by viewing a plane from unknown orientations," *Proceedings of the seventh ieee international conference on computer vision*, Vol. 1, Ieee, 1999, pp. 666–673.

[23] Araki, N., Sato, T., Konishi, Y., and Ishigaki, H., "Vehicle's orientation measurement method by single-camera image using known-shaped planar object," *Int. J. Innov. Comput. Inf. Control*, Vol. 7, No. B, 2011, pp. 4477–4486.

[24] Levenberg, K., "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, Vol. 2, No. 2, 1944, pp. 164–168.

[25] Marquardt, D. W., "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, Vol. 11, No. 2, 1963, pp. 431–441.

[26] Kannan, S. K., Chowdhary, G. V., and Johnson, E. N., "Adaptive control of unmanned aerial vehicles: theory and flight tests," *Handbook of Unmanned Aerial Vehicles*, Springer Netherlands, 2015, pp. 613–673.

[27] Dydek, Z. T., "Adaptive control of unmanned aerial systems," Ph.D. thesis, Massachusetts institute of Technology, 2010.

[28] Wang, B., Ali, Z. A., and Wang, D., "Controller for UAV to Oppose Different Kinds of Wind in the Environment," *Journal of Control Science and Engineering*, Vol. 2020, 2020.

[29] Dierks, T., and Jagannathan, S., "Output feedback control of a quadrotor UAV using neural networks," *IEEE transactions on neural networks*, Vol. 21, No. 1, 2009, pp. 50–66.

[30] Nicol, C., Macnab, C., and Ramirez-Serrano, A., "Robust neural network control of a quadrotor helicopter," *2008 Canadian conference on electrical and computer engineering*, IEEE, 2008, pp. 001233–001238.

[31] Shepherd III, J. F., and Tumer, K., "Robust neuro-control for a micro quadrotor," *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010, pp. 1131–1138.

[32] Williams-Hayes, P., "Flight test implementation of a second generation intelligent flight control system," *Infotech@ Aerospace*, 2005, p. 6995.

[33] Becker-Ehmck, P., Karl, M., Peters, J., and van der Smagt, P., "Learning to fly via deep model-based reinforcement learning," *arXiv preprint arXiv:2003.08876*, 2020.

[34] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[35] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[36] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M., "Deterministic policy gradient algorithms," *International conference on machine learning*, PMLR, 2014, pp. 387–395.

[37] Fujimoto, S., Hoof, H., and Meger, D., "Addressing function approximation error in actor-critic methods," *International Conference on Machine Learning*, PMLR, 2018, pp. 1587–1596.

[38] Up, O. S., "Twin Delayed DDPG," https://spinningup.openai.com/en/latest/algorithms/td3.html, 2021 (accessed June 21, 2021).

[39] Weng, L., "Domain Randomization for Sim2Real Transfer," *lilianweng.github.io/lil-log*, 2019. URL http://lilianweng.github.io/lil-log/2019/05/04/domain-randomization.html.

[40] Lee, B., "A Deep Reinforcement Learning Control Strategy for Disturbance Rejection," https://youtu.be/mcnH4MiJZIg, 2021 (accessed June 22, 2021).

[41] Doerry, A. W., "Ship dynamics for maritime ISAR imaging." Tech. rep., Sandia National Laboratories, 2008.