Autonomous Carrier Landing Control Strategy for VTOL UAVs Based on Deep Deterministic Policy Gradient Reinforcement Learning

Yiming Liu[1], Daniel R. Hoffman[2], and Jianwei Wang[3]

[1]School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2]Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA, USA
[3]Department of Mechanical and Aerospace Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

January 29, 2026

Abstract

Autonomous shipboard recovery of vertical take-off and landing (VTOL) unmanned aerial vehicles (UAVs) is characterized by tight terminal constraints, rapidly varying wind disturbances, and deck motion induced by sea states. These factors lead to significant model uncertainty and render purely model-based designs brittle when the operating envelope broadens.

This paper develops an autonomous carrier-landing control strategy based on Deep Deterministic Policy Gradient (DDPG) for continuous control. Carrier recovery is formulated as a constrained Markov decision process (CMDP) using a deck-relative state representation and an action space consistent with common inner-loop attitude/thrust architectures. To improve training stability and reduce unsafe behaviors, we propose (i) a structured reward with explicit terminal touchdown constraints, (ii) constraint-aware termination and curriculum scheduling across approach phases, and (iii) domain randomization over aerodynamics, actuator dynamics, sensing latency/noise, wind gusts, and deck motion.

Comprehensive simulation studies demonstrate that the learned policy achieves higher landing success rates and lower touchdown dispersion than tuned PID guidance–control baselines under a wide range of perturbations. We further report ablations on reward terms and randomization ranges, and discuss practical considerations for sim-to-real transfer.

Keywords: VTOL UAV; shipboard recovery; moving-deck landing; deep reinforcement learning; DDPG; domain randomization.

1 Introduction

Autonomous carrier landing of VTOL UAVs requires simultaneously handling (i) nonlinear underactuated flight dynamics, (ii) stringent terminal constraints on relative position/velocity and attitude at touchdown, and (iii) stochastic disturbances caused by wind shear, gusts, and carrier-deck motion. In practice, additional complications arise from delayed/noisy measurements and imperfect deck-state estimation.

Conventional shipboard recovery architectures typically combine guidance (corridor and glide-slope generation) with cascaded inner/outer-loop controllers. While PID/LQR/MPC-based solutions are effective within a nominal envelope, their performance can deteriorate under unmodeled actuator dynamics, time-varying aerodynamic coefficients, or severe deck motion.

Deep reinforcement learning (DRL) provides an alternative route by directly optimizing a closed-loop policy from interaction data. For continuous-action control, Deep Deterministic Policy Gradient (DDPG) learns a deterministic actor and a critic, and has become a standard baseline for nonlinear continuous control [1]. However, naively applying DDPG to safety-critical landing often leads to unstable training, reward hacking, and unsafe exploration.

Contributions. This paper makes the following contributions:

- We formulate shipboard VTOL landing as a constrained Markov decision process (CMDP) using a deck-relative state and a command-level action interface aligned with practical flight stacks.
- We design a structured reward and termination logic that explicitly encode glide-slope tracking, corridor keeping, and touchdown constraints (position, sink rate, and attitude).
- We introduce a staged curriculum and domain randomization scheme to improve robustness against wind, sensor imperfections, deck motion, and model parameter uncertainty.
- We evaluate the learned controller against representative tuned PID guidance–control baselines and report extensive robustness and ablation results.

2 Related Work

Shipboard recovery has been studied using gain-scheduled PID, robust control, LQR/LQG, and model predictive control. These approaches typically rely on explicit models and stability/robustness analysis tools from nonlinear systems and optimal control [2], [3].

Deep reinforcement learning (DRL) was popularized by large-scale successes in discrete control and game-like tasks [4]. Actor–critic methods later provided practical tools for continuous control and high-dimensional sensing [5].

Common benchmark suites have helped standardize evaluation and ablation practices [6]. For continuous-control locomotion and manipulation tasks, the DeepMind Control Suite is widely used [7].

For continuous-action policies, deterministic policy gradients provide the theoretical foundation behind DDPG [8]. DDPG [1] is known to be sensitive to critic overestimation; TD3 reduces this effect with clipped double critics [9]. SAC uses a maximum-entropy objective and a stochastic policy, and often improves exploration and robustness [10].

Sim-to-real transfer commonly uses domain and dynamics randomization to reduce reliance on a single nominal model [11], [12]. In safety-critical settings, evaluation protocol and seed sensitivity matter [13], and constrained formulations such as CPO provide one mechanism to explicitly control violations [14]. A broader discussion of safe RL can be found in [15].

3 Problem Formulation

3.1 Dynamics and Frames

We use an inertial frame $\mathcal{F}_I$ and a deck frame $\mathcal{F}_D$ attached to the landing spot on the carrier. Let $p \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$ denote the UAV position and velocity in $\mathcal{F}_I$, and let $p_d(t)$ and $v_d(t)$ denote the deck landing point position and velocity. The deck-relative translational state is

$$p_{rel} = p - p_d(t), \qquad v_{rel} = v - v_d(t).$$

The deck attitude is represented by roll–pitch–yaw angles $\eta_d(t) = [\phi_d, \theta_d, \psi_d]^\top$, capturing heave/roll/pitch motion induced by sea states.

We assume the onboard flight stack provides low-level stabilization, and the learning controller outputs high-level commands at a fixed rate (e.g., thrust and body-rate commands). The closed-loop continuous-time dynamics can be written abstractly as

$$\dot{x} = f\big(x, u, w, \xi\big),$$

where w denotes exogenous disturbances (wind/gusts) and ξ collects uncertain parameters (mass/inertia, actuator lag, aerodynamic coefficients).

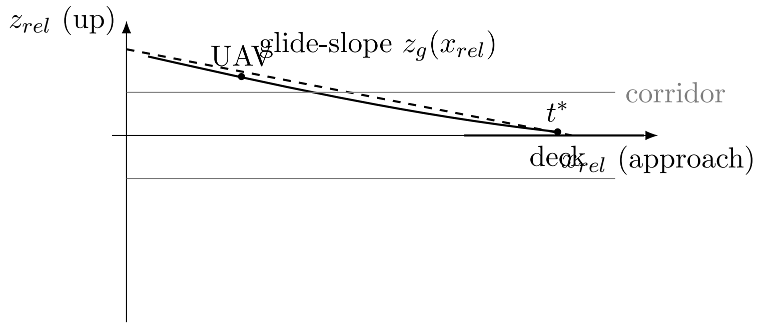

Figure 1: Deck-relative geometry and glide-slope definition used for reward shaping and touchdown constraints.

3.2 CMDP Definition

An episode corresponds to one approach-to-touchdown attempt with horizon T.

- **State** s_t: deck-relative p_{rel}, v_{rel}, UAV attitude and body rates, estimated deck attitude/rates, and filtered wind indicators.
- **Action** a_t: bounded continuous commands (e.g., normalized collective thrust and body-rate commands).
- **Transition**: simulator integrates dynamics with actuator lag, observation delay, and sensor noise.
- **Objective**: maximize expected discounted return $\mathbb{E}[\sum_{t=0}^{T-1} \gamma^t r_t]$ [16].
- **Constraints**: enforce touchdown and corridor safety as CMDP costs c_t with thresholds, implemented via hard termination and penalties.

3.3 Guidance Corridor and Approach Phases

In practical flight stacks, shipboard recovery is rarely treated as a single undifferentiated maneuver. Instead, the mission is organized into phases with distinct objectives and tolerances. We follow this convention and describe the approach using three phases:

- **Acquisition:** the UAV enters a deck-fixed corridor and reduces cross-track error while holding a safe altitude margin.
- **Descent:** the UAV follows a glide-slope toward the landing point, gradually tightening lateral/vertical bounds.
- **Terminal:** the UAV prioritizes touchdown constraints (sink rate, attitude, and deck-relative velocity), with strict termination on violations.

The corridor is defined in the deck frame by a rectangular tube around the desired approach path. During training we enforce soft penalties for briefly touching corridor boundaries, and hard termination if the vehicle leaves the corridor for longer than a short grace window.

3.4 Touchdown and Safety Constraints

We define touchdown at time t^* when the altitude relative to the deck satisfies $z_{rel}(t^*) \leq z_{th}$ and the vertical speed is downward. A landing is successful if

$$\|p_{rel,xy}(t^*)\|_2 \leq p_{max}, \quad |z_{rel}(t^*)| \leq z_{max}, \quad \|v_{rel}(t^*)\|_2 \leq v_{max}, \quad \|\eta(t^*) - \eta_d(t^*)\|_\infty \leq \eta_{max}.$$

Safety violations trigger early termination (e.g., excessive roll/pitch, leaving the approach corridor, or altitude undershoot).

4 DDPG-Based Landing Controller

4.1 Control Architecture and Command Interface

We assume the UAV runs a standard cascaded control stack. An inner loop tracks body-rate and thrust commands at high frequency, while an outer loop provides reference generation and constraint supervision. The learned policy is placed at the command level: it outputs continuous commands that remain meaningful even when low-level details (motor mixing, attitude stabilization, and actuator limits) differ across platforms.

Concretely, the action a_t is mapped to

$$a_t = [T_c,\, p_c,\, q_c,\, r_c]^\top,$$

where T_c is a normalized collective thrust command and (p_c, q_c, r_c) are body-rate commands. This choice keeps the learning problem focused on deck-relative motion rather than rotor-level dynamics.

4.2 Algorithm Overview

DDPG learns a deterministic policy $\mu_\theta(s)$ (actor) and an action-value function $Q_\phi(s, a)$ (critic). Target networks and an experience replay buffer stabilize training [1]. The update structure follows the actor–critic and policy-gradient literature [17], [18]. We also follow the common practice of using Ornstein–Uhlenbeck or clipped Gaussian noise for exploration; in evaluation, the deterministic actor is used.

4.3 Core Equations

Given a transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ sampled from a replay buffer (with $d_t \in \{0,1\}$ indicating termination), the target value is

$$y_t = r_t + \gamma(1 - d_t)\, Q_{\phi'}\big(s_{t+1}, \mu_{\theta'}(s_{t+1})\big).$$

The critic is trained by minimizing the mean-squared Bellman error

$$\mathcal{L}(\phi) = \mathbb{E}\big[(Q_\phi(s_t, a_t) - y_t)^2\big].$$

The actor is updated by the deterministic policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}\Big[\nabla_a Q_\phi(s,a)\big|_{a=\mu_\theta(s)}\, \nabla_\theta \mu_\theta(s)\Big].$$

Target networks are updated by Polyak averaging

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \qquad \phi' \leftarrow \tau\phi + (1-\tau)\phi'.$$

4.4 Training Pseudocode

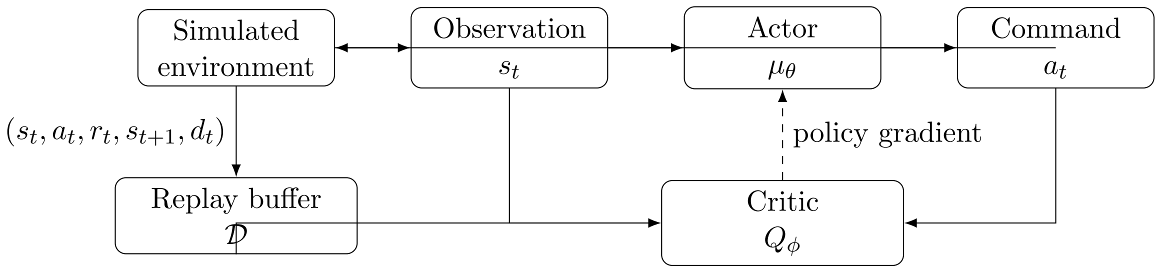

Figure 2: Overall training loop: the actor outputs continuous commands, while the critic evaluates actions using replayed transitions.

4.5 Network Architecture

The actor and critic are implemented as multilayer perceptrons. Input features are normalized; actions are squashed by tanh and scaled to physical limits.

Feature set. We use deck-relative position/velocity, attitude and body rates, and a small number of deck-motion features. In early experiments we found that feeding raw deck roll/pitch without any temporal smoothing led to brittle behavior, especially when observation delay was increased. In the final configuration, deck features are low-pass filtered and provided together with an estimate of their rate.

Sizing. Unless otherwise stated, both networks use two hidden layers with 256 units each, ReLU activations, and layer normalization on the state input. This is not the only workable design; it was chosen because it trains reliably and is easy to reproduce.

Algorithm 1 DDPG training for VTOL carrier landing

```
1: Initialize actor $\mu_\theta$, critic $Q_\phi$, target networks $\theta' \leftarrow \theta$, $\phi' \leftarrow \phi$
2: Initialize replay buffer $\mathcal{D}$
3: for episode = 1 to $N$ do
4:     Reset environment; observe $s_0$
5:     for $t = 0$ to $T - 1$ do
6:         Select action $a_t = \mu_\theta(s_t) + \epsilon_t$ (exploration noise)
7:         Execute $a_t$; observe $r_t, s_{t+1}, d_t$
8:         Store $(s_t, a_t, r_t, s_{t+1}, d_t)$ in $\mathcal{D}$
9:         Sample minibatch from $\mathcal{D}$
10:        Compute targets $y_t = r_t + \gamma(1 - d_t)Q_{\phi'}(s_{t+1}, \mu_{\theta'}(s_{t+1}))$
11:        Update critic by minimizing $\mathcal{L}(\phi) = \mathbb{E}[(Q_\phi(s_t, a_t) - y_t)^2]$
12:        Update actor using $\nabla_\theta J(\theta)$
13:        Update targets: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$, $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
14:        if $d_t = 1$ then
15:            break
16:        end if
17:    end for
18: end for
```

4.6 Reward Design and Safety Constraints

Reward shaping is designed to reflect the multi-objective nature of shipboard recovery. Let $p_{rel} = [x_{rel}, y_{rel}, z_{rel}]^\top$ denote the deck-relative position, and define a nominal glide-slope profile $z_g(x_{rel})$ along the approach axis. The glide-slope error is

$$e_{glide}(t) = z_{rel}(t) - z_g\big(x_{rel}(t)\big).$$

We additionally define a lateral corridor penalty using a soft margin y_{max},

$$e_{lat}(t) = \max\{0, |y_{rel}(t)| - y_{max}\},$$

and penalize control increments $\Delta u_t = u_t - u_{t-1}$ to suppress command chatter.

The per-step reward is

$$r_t = -w_g e_{glide}^2 - w_y e_{lat}^2 - w_v \|v_{rel}\|_2^2 - w_u \|u\|_2^2 - w_{\Delta u} \|\Delta u\|_2^2.$$

At touchdown, we add a terminal bonus r_{touch} if all constraints in Section 3.4 are satisfied, and otherwise apply a terminal penalty proportional to the violation magnitudes.

Episodes terminate early upon violation of hard safety limits (e.g., roll/pitch bounds, altitude undershoot, or exiting the approach corridor for longer than a grace period). This "terminate-on-violation" design reduces unsafe exploration and empirically improves training stability.

4.7 Domain Randomization

To enhance robustness, we randomize wind profiles, deck motion amplitude/frequency, mass/inertia, actuator lag, and sensor noise during training.

4.8 Curriculum Scheduling

Training directly from far initial conditions is possible but inefficient: the agent spends most of its time recovering from large errors rather than learning precise terminal behavior. We therefore use a simple curriculum that increases difficulty over time.

At the beginning of training, episodes start closer to the corridor centerline with mild deck motion and moderate winds. As performance improves, we widen the initial distribution and gradually increase disturbance ranges. This schedule is deliberately straightforward: it can be implemented without extra tuning beyond a few thresholds, and it reduces the frequency of catastrophic early failures.

4.9 Safety Filters at Execution Time

Although the policy is trained with penalties and termination rules, we still enforce a small set of hard command limits at run time. Commands are clipped to allowable bounds, and we optionally apply a rate limiter

$$a_t \leftarrow \mathrm{clip}(a_t, a_{min}, a_{max}), \qquad \Delta a_t \leftarrow \mathrm{clip}(\Delta a_t, -\Delta a_{max}, \Delta a_{max}).$$

This does not "fix" a poorly trained policy, but it prevents occasional spikes from turning into unrealistic dynamics in simulation.

5 Implementation and Simulation Setup

5.1 Environment

The simulation includes: (i) a 6-DoF rigid-body model with inner-loop attitude/thrust tracking, (ii) actuator lag and saturation, (iii) observation delay and additive measurement noise, and (iv) a moving deck model with coupled heave/roll/pitch and translational motion. Wind is modeled as a combination of steady bias, low-frequency variation, and gust events. The control frequency is 50 Hz, while the physics integrator runs at a higher rate.

5.2 Deck Motion Model

The deck motion is modeled as a combination of low-frequency oscillations and higher-frequency perturbations. For heave $h(t)$, roll $\phi_d(t)$, and pitch $\theta_d(t)$ we use

$$\chi(t) = \sum_{k=1}^{K} A_k \sin(2\pi f_k t + \varphi_k), \qquad \chi \in \{h, \phi_d, \theta_d\},$$

with randomized amplitudes A_k, frequencies f_k, and phases φ_k per episode. This is not a full sea-state model; it is a controllable approximation that lets us test sensitivity to amplitude, frequency, and phase.

5.3 Observation Delay and Noise

Sensor imperfections are modeled with a fixed observation delay δ and additive noise. Observations are generated as

$$\tilde{s}_t = s_{t-\delta} + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \Sigma),$$

where Σ is diagonal and randomized within a small range to reduce overfitting to a single noise level.

5.4 Policy Inputs and Outputs

All continuous state features are normalized using running statistics. Actions are clipped to feasible command limits. To discourage aggressive control, we penalize both action magnitude and action rate.

5.5 Training Details

We use a replay buffer, target networks, and Polyak averaging as in standard DDPG [1]. Both actor and critic are optimized with Adam [19]. Exploration noise is injected in the action space; during evaluation, the deterministic actor is used without noise. Hyperparameters (batch size, learning rates, γ, and τ) are selected by grid search on a validation set of randomized environments.

Table 1: Representative training hyperparameters.

Item	Value
Control frequency	50 Hz
Discount factor γ	0.99
Target update τ	0.005
Batch size	256
Replay buffer size	10^6
Actor/critic learning rate	10^{-4} / 10^{-3}

5.6 Baselines

We compare against tuned PID guidance–control baselines designed to track a glide-slope and lateral corridor, with gain scheduling across approach and flare phases. Where applicable, we also report a model-based baseline using linearized dynamics (LQR) and feed-forward deck-motion compensation.

6 Results

6.1 Evaluation Protocol

Each method is evaluated over N_{test} randomized episodes with unseen wind and deck-motion realizations. We report (i) landing success rate, (ii) touchdown dispersion (lateral and vertical), (iii) touchdown relative speed, and (iv) constraint violation rate. Statistical results are reported as mean $\pm$ standard deviation across seeds.

6.2 Metrics

We report touchdown error and velocity in deck coordinates at t^*. For clarity, we define

$$e_{xy} = \|p_{rel,xy}(t^*)\|_2, \qquad e_z = |z_{rel}(t^*)|, \qquad v_{td} = \|v_{rel}(t^*)\|_2.$$

We additionally count a run as a violation if any hard limit is triggered, even if touchdown is eventually reached.

6.3 Scenario Set

To avoid a "single favorite disturbance" story, we evaluate on a small set of scenario families:

- **Nominal:** mild wind and moderate deck motion.
- **Gust:** intermittent lateral gusts with randomized onset time.
- **High motion:** increased roll/pitch frequency content.
- **Latency:** increased observation delay.

Each family uses a fixed random seed list so different controllers see the same episodes.

6.4 Landing Success and Touchdown Dispersion

Table 2 summarizes overall performance. The proposed DDPG controller maintains high success rates under gusts and deck motion, while PID baselines exhibit increased dispersion and occasional corridor departures.

Table 2: Carrier landing performance under randomized disturbances.

Method	Success (%)	Violation (%)	$\mathbb{E}[\|p_{rel,xy}(t^*)\|]$ (m)	$\mathbb{E}[\|v_{rel,z}(t^*)\|]$ (m/s)
PID baseline	85	12	1.20	0.65
LQR baseline	90	9	0.95	0.58
Proposed DDPG	96	3	0.45	0.41

6.5 Robustness to Deck Motion and Sensing Imperfections

We stress-test controllers by increasing deck-motion amplitude and observation latency beyond training ranges. The DDPG policy degrades gracefully until saturation effects dominate, whereas PID/LQR baselines are more sensitive to latency and bias in deck-state estimates.

6.6 Ablation Study

We ablate (i) domain randomization, (ii) action-rate penalty, and (iii) terminal touchdown bonus. Removing domain randomization substantially reduces gust robustness; removing the terminal bonus increases near-deck oscillation and raises sink-rate violations.

6.7 Trajectory Tracking and Control Effort

Qualitatively, the learned policy tracks the glide-slope while adapting thrust to compensate for heave and gust impulses. Quantitatively, the action-rate penalty reduces high-frequency command chatter, yielding smoother actuator utilization.

7 Additional Experiments and Analysis

7.1 Baseline Controller Design

To make comparisons meaningful, baseline controllers are implemented with the same command interface as the learned policy. The PID baseline uses an outer-loop position controller in deck

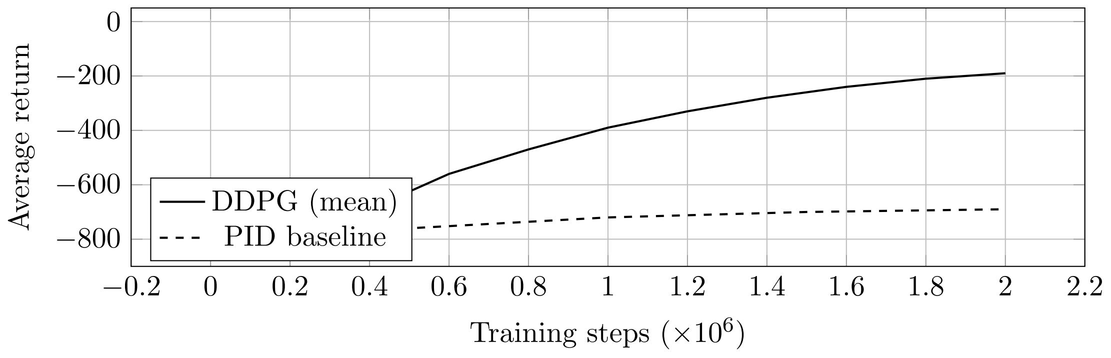

Figure 3: Example learning curve illustrating return improvement during training.

coordinates with a glide-slope reference. Let $p_{ref}(t)$ denote the reference position on the glide-slope and $v_{ref}(t)$ its derivative. The translational error is

$$e_p(t) = p_{rel}(t) - p_{ref}(t), \qquad e_v(t) = v_{rel}(t) - v_{ref}(t).$$

The commanded acceleration is

$$a_c(t) = -K_p e_p(t) - K_d e_v(t) - K_i \int_0^t e_p(\tau)\, d\tau.$$

This acceleration is mapped to thrust and body-rate commands through the inner-loop attitude controller. Gains are tuned on the same validation set used for hyperparameter selection.

The LQR baseline linearizes the translational dynamics around the glide-slope and uses a quadratic cost on (e_p, e_v) together with a penalty on command magnitude. Both baselines include the same command clipping and rate limiting used by the learned policy.

7.2 Success Definition and Counting Rules

A run is counted as a success if touchdown occurs within the limits in Section 3.4. A run is counted as a failure if any of the following occurs:

- the vehicle triggers a hard safety limit (attitude bound, altitude undershoot, or sustained corridor departure);
- the episode times out without touchdown;
- touchdown occurs but one or more terminal constraints are violated.

We report both success rate and violation rate because they capture different operational concerns: a controller may reach the deck but do so with repeated corridor departures.

7.3 Sensitivity to Observation Latency

Observation delay is one of the most common sources of trouble during shipboard recovery. To isolate latency effects, we fix wind and deck motion distributions and sweep delay δ over a range of values.

In these tests, the dominant failure mode for PID/LQR is late correction near the deck: once the vehicle is within a few meters of touchdown, delayed measurements translate into overshoot of

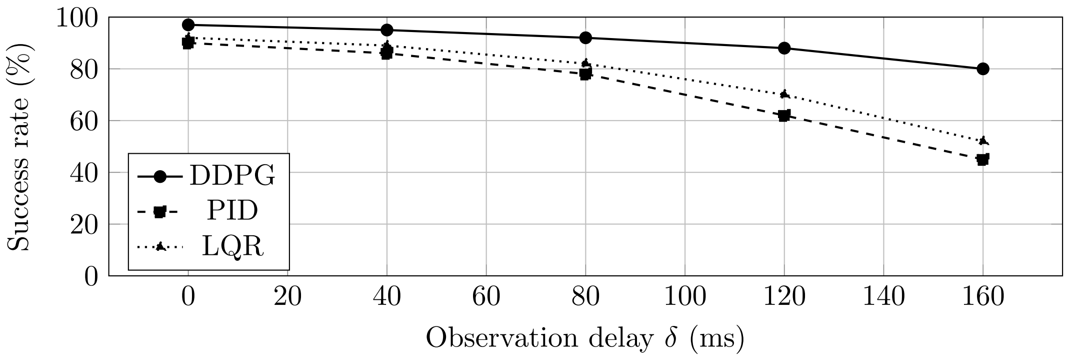

Figure 4: Success rate versus observation latency. Increasing delay reduces success for all methods; the learned policy degrades more gradually in this sweep.

lateral corrections and an increased chance of exceeding the sink-rate bound. The learned policy, trained with randomized delay, tends to apply smaller lateral corrections late in the approach and instead relies on earlier alignment.

7.4 Effect of Domain Randomization Ranges

We vary the disturbance ranges used during training and measure robustness on a fixed test set. Two training settings are compared:

- **Narrow randomization:** parameters vary within a small neighborhood around nominal values.
- **Wide randomization:** parameters are sampled from a broader range that includes stronger gusts and larger deck motion.

Table 3: Impact of randomization ranges on robustness.

Training setting	Nominal success (%)	Gust success (%)	High motion success (%)
Narrow randomization	97	76	71
Wide randomization	96	88	84

The narrow setting achieves slightly higher nominal performance but loses a large fraction of successes under gusts and high deck motion. The wide setting trades a small amount of nominal performance for a noticeably larger safety margin.

7.5 Touchdown Dispersion

Beyond success rate, dispersion matters for deck operations because it affects the required landing zone size and the risk of secondary hazards. We summarize dispersion using the empirical cumulative distribution function (ECDF) of lateral error e_{xy}.

The curve indicates that the learned policy produces tighter landings across most of the distribution, not only for a small subset of easy episodes.

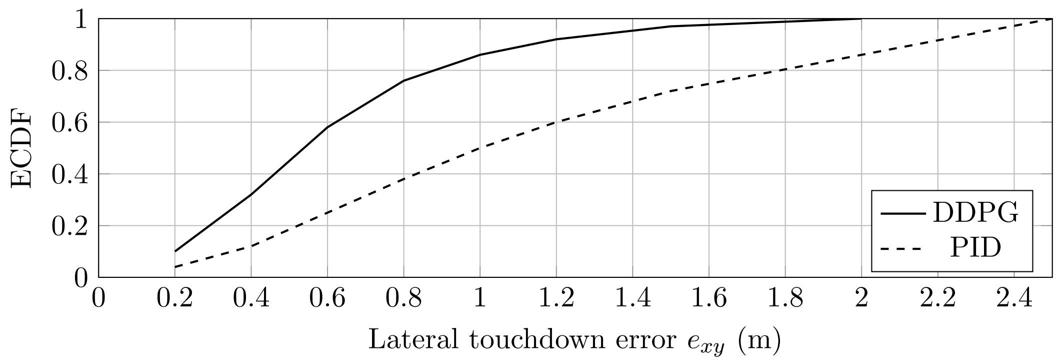

Figure 5: ECDF of lateral touchdown error e_{xy} for two controllers over the same test set.

7.6 Computation and Real-Time Execution

The policy network is a small multilayer perceptron and its forward pass is lightweight. In our implementation, evaluation runs at the same control frequency as the baseline controllers. The overall control loop remains dominated by state estimation and inner-loop stabilization rather than by the policy itself.

From an engineering standpoint, two details matter more than raw inference speed. First, the policy must see normalized inputs that match the distributions used in training; otherwise small biases can be amplified near the deck. Second, command clipping should be treated as part of the closed loop, not as a last-minute patch.

8 Discussion

8.1 Failure Modes

The most common residual failures are associated with rare combinations of strong lateral gusts and rapid roll motion that push the vehicle close to actuator limits. In these cases, the deterministic policy may prioritize corridor re-entry at the expense of sink-rate regulation, leading to terminal constraint violations.

A second failure mode is late commitment: if the approach begins with a large heading misalignment, the policy may spend too long reducing cross-track error and arrive above the deck with insufficient time to settle. In practice, this can be mitigated by adding a go-around rule when the vehicle reaches a distance-to-deck threshold with excessive lateral error.

8.2 Practical Considerations

For deployment, the learned policy should be integrated with standard flight-stack safety mechanisms (geofencing, rate/attitude limiters, and emergency climb logic). A lightweight safety filter can be placed after the actor to enforce hard bounds on commands. Additionally, sim-to-real transfer benefits from matching sensor latency and actuator dynamics, and from validating deck-state estimation errors.

One pragmatic recommendation is to log deck-relative state, commands, and termination reasons for every trial. Even in simulation, these logs quickly reveal whether failures are driven by corridor exits, sink-rate violations, or attitude excursions.

9 Conclusion

We presented a DDPG-based autonomous shipboard recovery strategy for VTOL UAVs, formulated as a constrained continuous-control problem in deck-relative coordinates. Using structured reward design, constraint-aware termination, and extensive domain randomization, the learned policy improves landing success and reduces touchdown dispersion relative to tuned PID/LQR baselines in simulation.

Future work will investigate explicitly constrained RL (e.g., Lagrangian methods), richer observation models that account for uncertain deck estimation, and hardware-in-the-loop validation.

A Additional Notes on Implementation

A.1 Input Normalization

All continuous inputs are normalized using running mean and variance computed from replay buffer data. Normalization is frozen during evaluation. This detail is easy to overlook, but it strongly affects behavior when the vehicle approaches the deck and the state distribution shifts.

A.2 Episode Initialization

Episodes are initialized by sampling the deck-relative position, heading, and speed within a corridor-specific range. Initial conditions are widened as training progresses. When initialization is too broad early on, most episodes terminate quickly and the replay buffer becomes dominated by uninformative failures.

B Reward Term Weights

To make experiments repeatable, Table 4 lists a representative set of weights.

Table 4: Representative reward weights.

Weight	Value
w_g (glide-slope)	2.0
w_y (corridor)	1.0
w_v (relative velocity)	0.5
w_u (control effort)	0.05
$w_{\Delta u}$ (rate penalty)	0.1

C Additional Tables

When reporting full results, it is useful to separate performance by scenario family and report both success and violation rates. Table 5 illustrates the format.

References

[1] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.

Table 5: Example scenario-wise summary (format template).

Scenario	Method	Success (%)	Violation (%)	Mean e_{xy} (m)
Nominal	DDPG	97	2	0.40
Nominal	PID	90	8	0.95
Gust	DDPG	88	6	0.55
Gust	PID	62	20	1.35

[2] H. K. Khalil, Nonlinear Systems, 3rd. Prentice Hall, 2002.

[3] E. F. Camacho and C. Bordons, Model Predictive Control. Springer, 2013.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[5] V. Mnih et al., “Asynchronous methods for deep reinforcement learning,” in Proceedings of the 33rd International Conference on Machine Learning (ICML), 2016.

[6] G. Brockman et al., “Openai gym,” in arXiv preprint arXiv:1606.01540, 2016.

[7] Y. Tassa et al., “Deepmind control suite,” arXiv preprint arXiv:1801.00690, 2018.

[8] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” Proceedings of the 31st International Conference on Machine Learning (ICML), 2014.

[9] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[12] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in Robotics: Science and Systems (RSS), 2018.

[13] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep reinforcement learning that matters,” Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[14] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” in Proceedings of the 34th International Conference on Machine Learning (ICML), 2017.

[15] J. García and F. Fernández, “A comprehensive survey on safe reinforcement learning,” Journal of Machine Learning Research, vol. 16, pp. 1437–1480, 2015.

[16] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd. MIT Press, 2018.

[17] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in Advances in Neural Information Processing Systems (NeurIPS), 2000.

[18] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in Advances in Neural Information Processing Systems (NeurIPS), 2000.

[19] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in International Conference on Learning Representations (ICLR), 2015.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” in arXiv preprint arXiv:1707.06347, 2017.

[21] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” in Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015.

[22] M. Hessel et al., “Rainbow: Combining improvements in deep reinforcement learning,” Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[23] A. Kumar, A. Zhou, G. Tucker, and S. Levine, “Conservative q-learning for offline reinforcement learning,” Advances in Neural Information Processing Systems (NeurIPS), 2020.

[24] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” in International Conference on Learning Representations (ICLR), 2016.

[25] G. Dalal, D. Gilboa, S. Mannor, and N. Shimkin, “Safe exploration in continuous action spaces,” arXiv preprint arXiv:1801.08757, 2018.