SRTTU

Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

JECEI

Research paper

Vision-Based Autonomous UAV Navigation through GPS-Denied Narrow Passages Using Deep Reinforcement Learning

Mahdi Shahbazi Khojasteh, *Armin Salimi-Badr* *

Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran.

Article Info

Article History:
Received 18 February 2025
Reviewed 09 April 2025
Revised 16 May 2025
Accepted 06 June 2025

Keywords:
Deep reinforcement learning
Proximal policy optimization
Obstacle avoidance
Trajectory planning
Autonomous navigation
Unmanned aerial vehicle

*Corresponding Author's Email Address:
a_salimibadr@sbu.ac.ir

Abstract

Background and Objectives: Unmanned Aerial Vehicles (UAVs) face significant challenges in navigating narrow passages within GPS-denied environments due to sensor and computational limitations. While deep reinforcement learning (DRL) has improved navigation, many methods rely on costly sensors, such as depth cameras or LiDAR. This study addresses these issues using a vision-based DRL framework with a monocular camera for autonomous UAV navigation.
Methods: We propose a DRL-based navigation system utilizing Proximal Policy Optimization (PPO). The system processes a stack of grayscale monocular images to capture short-term temporal dependencies, approximating the partially observable environment. A custom reward function encourages trajectory optimization by assigning higher rewards for staying near the passage center while penalizing further distances. The navigation system is evaluated in a 3D simulation environment under a GPS-denied scenario.
Results: The proposed method achieves a high success rate, surpassing 97% in challenging narrow passages. The system demonstrates superior learning efficiency and robust generalization to new configurations compared to baseline methods. Notably, using stacked frames mitigates computational overhead while maintaining the effectiveness of the policy.
Conclusion: Our vision-based DRL approach enables autonomous UAV navigation in GPS-denied environments with reduced sensor requirements, offering a cost-effective and efficient solution. The findings highlight the potential of monocular cameras paired with DRL for real-world UAV applications such as search and rescue and infrastructure inspection. Future work will extend the framework to obstacle avoidance and general trajectory planning in dynamic environments.

How to cite this paper:
M. Shahbazi Khojasteh, A. Salimi-Badr, "Vision-based autonomous UAV navigation through gps-denied narrow passages using deep reinforcement learning," J. Electr. Comput. Eng. Innovations, 14(1): 55-72, 2026.
DOI: 10.22061/jecei.2025.11563.815
URL: https://jecei.sru.ac.ir/article_2349.html

Introduction

Unmanned Aerial Vehicles (UAVs) have emerged as transformative tools across a wide range of applications, impacting both civilian and military sectors [1]. This surge in UAV deployment is attributed to their inherent advantages, including extended endurance and stability in diverse operational conditions, enhanced maneuverability, cost-effectiveness, and the ability to access hazardous or difficult-to-reach locations [1], [2]. UAVs find utility in diverse fields, encompassing precision agriculture, search and rescue missions, surveillance, remote sensing, infrastructure inspection, package delivery, and more [2].

While UAVs have shown great promise, their performance is hindered by limitations such as restricted battery endurance and flight time [3]. To enhance UAV capabilities, careful selection of onboard sensors is essential. UAVs commonly rely on sensors such as accelerometers, magnetometers, gyroscopes, and GPS for navigation [4], [5]. Additionally, they utilize stereo cameras, LiDAR, ultrasonic sensors, and distance sensors to gather environmental data [6].

Despite their utility, many of these sensors present drawbacks, including high cost, significant weight, increased energy consumption, and operational complexity. In contrast, the camera is a critical component [5], [7], providing a versatile and efficient approach to enhancing UAV environmental perception [8]. Monocular cameras, in particular, are highly suited for applications where minimizing energy, size, and weight is crucial [9], [10].

UAVs are increasingly used in applications that demand a high degree of autonomy. Traditional methods for UAV navigation often struggle in complex environments, particularly in confined spaces like narrow passages. These approaches, which typically rely on pre-programmed flight paths or rule-based systems, exhibit limitations in adaptability and responsiveness to unforeseen obstacles or environmental changes [2], [11].

To handle these limitations, the integration of Artificial Intelligence (AI), particularly Machine Learning (ML) and Deep Learning (DL), offers promising avenues for enhancing UAV autonomy and navigation capabilities [6], [12], [13]. ML empowers UAVs to learn from data, enabling them to adapt to environments and make informed decisions without explicit programming [14]. DL uses artificial neural networks with multiple layers to extract intricate patterns and representations from data, allowing for more sophisticated and robust autonomous capabilities [12], [15].

A prominent method for indoor navigation [16] uses DL with Convolutional Neural Networks (CNNs) [17]. This approach treats navigation as a classification task, where a CNN classifies video feeds from a monocular camera into actions. After training on a diverse dataset of images, the CNN can guide UAV maneuvers autonomously without extra sensors or 3D maps. However, it needs a large, labeled dataset for reliable performance across scenarios.

To ensure a UAV can perform tasks safely and efficiently, it must be able to make decisions and adapt to changing conditions without human input. In this context, Reinforcement Learning (RL) provides a foundation for learning optimal decision-making policies. By combining both methods, Deep Reinforcement Learning (DRL) forms that offer significant potential for achieving higher levels of autonomy [2], [18].

DRL enables an agent to learn optimal actions through interactions with its environment, receiving rewards for desired behaviors and penalties for undesirable ones [19]. This trial-and-error learning process allows the agent to refine its decision-making policy progressively, improving navigation performance over time.

Numerous studies have adopted DRL for autonomous vehicles [20]-[22] and UAV navigation [23]-[27], showcasing its effectiveness in intelligent decision-making and adaptability to challenging scenarios. A key drawback of approaches like [9] and [28] is that they rely on direct access to depth image data through simulations for UAV navigation to estimate the distance to obstacles using visual cues. However, obtaining depth map data from the camera in real-world scenarios is often expensive. It necessitates specialized equipment, such as stereo cameras or depth sensors, which can significantly increase the overall cost.

UAV navigation in narrow passages typically relies on vision-based methods, including image processing for window detection, homography for pose estimation, and visual servoing for trajectory control [29]-[31]. Some approaches enhance monocular vision with optical flow sensors or use stereo vision for depth information [29], [32]. Simultaneous Localization and Mapping (SLAM) also allows simultaneous map building and localization through visual or other sensors.

Navigating in confined spaces presents several challenges for UAVs, arising from limitations in sensing, computational resources, environmental factors, and the complexity of path planning and control. One significant hindrance is the unreliability or unavailability of GPS signals indoors [32].

Enclosed spaces often require precise localization and mapping [33], making techniques like SLAM potentially useful yet challenging due to their high computational and power demands, which strain the limited resources of small UAVs [16], [29], [34]. Furthermore, monocular SLAM suffers from scale ambiguity [30]. In indoor environments, featureless surfaces can impair feature-based SLAM performance, while navigating narrow passages requires accurate obstacle detection [16], [32], [35]. [36] discusses utilizing a rangefinder sensor, noting that while it serves the purpose of navigation, a camera might be a better option and lead to improved performance.

This paper addresses the challenge of enabling autonomous UAV navigation through narrow passages in a GPS-denied environment using a vision-based DRL approach with a monocular camera. Additionally, it explores the performance of DRL in controlling the UAV's decision-making, focusing on key metrics such as success rate, computational costs, and collision avoidance. The principal contributions of the paper are as follows:

- We implement a DRL-based navigation system that utilizes only a monocular vision system for obstacle avoidance through perceptual spatial information extraction;
- We stack input images to provide the agent with a short history of observations, capturing short-term temporal dependencies to approximate the underlying Partially Observable Markov Decision Process (POMDP) while reducing computational overhead;
- We propose a reward function to simplify and speed up the agent's problem-solving process by focusing on navigating narrow passages and assigning higher rewards as the agent moves closer to the center of the passage;
- We evaluate our method in GPS-denied narrow passages using realistic 3D simulations to replicate real-world scenarios, including configurations with incomplete visual cues.

The structure of this paper is as follows: the background and relevant literature are reviewed in the next section. The Methodology section outlines the methods and materials used in this study. The results are presented and analyzed in the Results and Discussion section. Lastly, the Conclusion section provides final remarks and suggests potential directions for future research.

Related Work

A. Classical Methods

Classical approaches for UAV navigation and obstacle avoidance have been extensively studied, with prominent methods including graph-based algorithms, potential field methods, and rule-based systems. Graph-based algorithms, such as Dijkstra's and A* algorithms, discretize the environment into a graph and employ search algorithms to find the optimal path [12], [18]. These methods are known for their ability to find the shortest path, but can suffer from computational inefficiency, especially in large and complex environments [12], [37]. On the other hand, potential field methods [38] treat the UAV as a point mass moving in a potential field generated by obstacles and the target. These methods are computationally efficient but are prone to local minima, especially in the presence of concave obstacles [11]. Rule-based systems rely on a set of predefined rules to guide the UAV's movement. These systems are simple to implement but lack flexibility and adaptability to complex environments [37].

These methods are more suitable for path planning in a known global environment. An example is [39], where it uses A* to find the best route for food delivery. Although it developed a complete prototype system, it opted against conducting autonomous flights due to safety concerns. The A* algorithm heavily depends on the heuristic cost function and requires continuous computation and storage [40].

Despite the constraints, the algorithm has faced many improvements over the years. For instance, [41] proposed a multi-objective programming model for UAV navigation by incorporating error correction and path constraints. The algorithm proves effective in a grid map environment, but its usage diminishes in high-dimensional spaces [42]. Dynamic A* (D*) has been introduced to mitigate the limitations [42]. Nevertheless, creating the cost map for D* is a laborious and error-prone endeavor [40].

Continuing with the traditional methods, the Rapidly-exploring Random Tree (RRT) is another widely used sampling-based technique within the roadmap algorithms [37] that operates as a purely random search for path-planning with various motion constraints [42]. Study by [43] incorporates RRT with pruning to eliminate redundant nodes in cooperative multiple UAV navigation, ensuring a collision-free path between two endpoints by limiting sampling within a feasible region based on the maximum turning angle. However, the algorithm does not assure the optimality of the generated path and encounters inefficiency in cluttered or narrow spaces.

Another enhancement introduced to RRT is [44], where it utilizes a cost function as heuristic information that includes both path length and path threat strength costs to guide the expansion of new nodes. Nevertheless, [43] simulates in a 2D environment, thus requiring further validation in a 3D environmental setup, and [44], despite being done in a 3D environment, is only suitable for static environments with all details to be known a priori.

Bug algorithms are practical navigation strategies that break down the task into moving toward the goal and navigating around obstacles encountered. While these methods do not require prior knowledge of the environment, their resulting path is often suboptimal

[21], [45]-[47]. Additionally, assumptions such as finding the M-Line for obstacle avoidance are idealized and unrealistic in the real world [45].

B. Deep Learning Methods

Most classical techniques can find optimal paths, but they typically require complete knowledge of the environment, making them unsuitable for unknown environments [48]. In contrast, DL approaches are model-free and demonstrate strong generalization when facing new or changing scenarios. Although DL algorithms require large datasets and extensive training time, they offer fast inference once trained. DL's ability to learn complex data representations from real-world environments makes it well-suited for autonomous robotic applications, particularly for UAVs [15], [49].

CNNs excel at extracting spatial features from images, making them highly effective for tasks such as obstacle detection and environment mapping, where visual data is crucial. On the other hand, Recurrent Neural Networks (RNNs) handle sequential data [50], enabling them to process temporal patterns and dependencies, which is particularly useful for trajectory prediction and motion planning. For instance, [51] proposes a fast global path planner that uses RNN capability to generate safe, collision-free paths. Compared to classical global path-planning algorithms, this approach performs better in complex and challenging environments.

Hybrid models, such as CNN-RNN architectures, effectively combine spatial and temporal data analysis, making them well-suited for tasks like those explored in [52]-[54]. However, a notable downside of such hybrid models is their increased computational complexity and training time, which may pose challenges in resource-constrained environments.

Another set of techniques commonly employed for navigation, path planning, and obstacle avoidance is the DRL approach. DRL is becoming increasingly essential for achieving autonomous decision-making in complex scenarios [15], [55], 56], and has demonstrated exceptional performance, particularly in challenging environments. DRL techniques combine the representational power of DL with RL's capability to learn optimal policies through trial and error [57]. This enables UAVs to acquire complex behaviors and adapt to their environments [58].

RL refers to a manner of learning wherein an agent engages in a trial-and-error process and acquires knowledge and skills by leveraging the reward signals and feedback obtained from the environment [59]. There is no predefined dataset in this learning paradigm from which the model can learn. Instead, the agent generates the training data in real-time through interactions with the environment [19]. DRL improves upon this methodology by utilizing deep neural networks to estimate the policy, value functions, or both to determine the most advantageous behavioral choices.

In UAV navigation, DRL has proven remarkably effective in obstacle avoidance, trajectory planning, and real-time adaptation to dynamic circumstances. Notably, the study by [27] emphasizes DRL's ability to effectively manage high-dimensional state observations. Additionally, it introduces an innovative memory pool that enhances learning efficiency and accelerates training, further supporting DRL as a robust solution for UAVs operating in complex and unpredictable environments.

As another example, [60] introduces a memory-based DRL approach that enables UAVs to avoid obstacles such as pedestrians with limited environmental knowledge. Similarly, [61] creates a DRL framework for UAV navigation in indoor environments, demonstrating the potential of DRL for challenging indoor navigation scenarios. A study by [62] focuses on collision-free UAV navigation using a monocular camera and DRL, highlighting the use of vision-based sensors for autonomous navigation. Likewise, [63] studies autonomous UAV navigation in large-scale complex environments using a DRL approach, showcasing the applicability of DRL for navigation in challenging outdoor scenarios.

Methodology

RL tasks are framed as interactions within an agent-environment system, where the agent decides actions in response to the environment's current state and receives feedback for its choices [19]. This control loop process is depicted in Fig. 1. While RL problems are typically modeled using the Markov Decision Process (MDP) [64] under the assumption of fully observable environments, real-world scenarios rarely satisfy the Markov property [65]. Thus, critical factors remain hidden in such cases, creating uncertainty for effective decision-making.

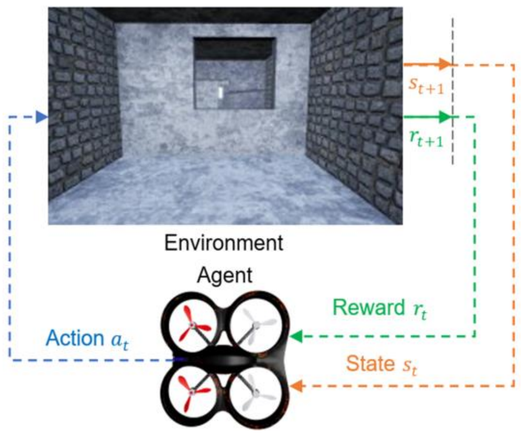


Fig. 1: High-level overview of RL control loop. The agent applies an action to the environment based on its state, receives rewards, and observes the next state.

To address this, we propose modeling using a POMDP [66], which generalizes the MDP framework to handle incomplete observations. The POMDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O})$ with each component representing states, actions, transition probabilities, rewards, and observations $o \in \Omega$, and the conditional probability distribution over Ω based on the state-action pairs $(o_t|s_{t+1}, a_t)$.

The agent seeks to maximize the cumulative reward over time [57]. The UAV refines its policy, π, to favor actions that optimize the expected total reward. This objective is expressed through the following expectation:

$$\mathbb{E}_{s_t=s, a_t=a, \tau\sim\pi}[R_t] = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{1}$$

where τ denotes a trajectory sampled from the policy π, R_t is the cumulative reward starting from time t, $\gamma \in [0,1)$ is the discount factor, and r_t represents the reward received at time step t.

A significant challenge of training DRL agents with policy gradient algorithms is their vulnerability to sudden performance degradation, where their effectiveness sharply declines [67]. Addressing this issue is complicated as the agent begins generating suboptimal trajectories, which serve as poor data for further policy updates, compounding the problem.

Proximal Policy Optimization (PPO) [68] is a family of optimization techniques that employs an actor-critic framework to address these challenges. PPO employs an on-policy learning approach, where the decision-making policy is updated using a small batch of experiences collected from interactions with the environment. After updating, it discards these experiences and collects a new batch using the revised policy.

PPO offers notable advantages, especially in robotic path planning and navigation. Its superior sample efficiency and stability make it a robust choice for continuous control tasks, where consistent performance is vital [69]. The algorithm can effectively optimize the UAV's path in scenarios like navigating narrow corridors, ensuring minimization of target time and practical obstacle avoidance, thus enhancing the reproducibility and clarity of experimental results [70].

Furthermore, PPO demonstrates strong stability and versatility, maintaining minimal deviations in performance across different environmental conditions. Its reliability makes it effective in both simple and complex environments, excelling in obstacle avoidance and optimal route planning [70].

Additionally, PPO demonstrates potent performance when encountering noise or disturbances. Even under challenging conditions with external perturbations, the algorithm maintains stable flight, achieving desired speeds and angles with low computational overhead [71].

PPO introduces a relative policy performance metric that quantifies the difference in performance between two policies. Applying a constraint on the step size during policy updates prevents performance collapse and ensures monotonic improvement. With $J(\pi)$ as the objective function, π representing the current policy, and π' denoting the updated policy after an iteration, the relative policy performance identity can be expressed as follows:

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau\sim\pi'}\left[\sum_{t=0}^{T} \gamma^t A^{\pi}(s_t, a_t)\right], \tag{2}$$

with $A^{\pi}(s,a)$ being the advantage function under π that measures whether a given action is better or worse than the policy's average action in a given state, defined as:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s), \tag{3}$$

where $Q^{\pi}(s,a)$ represents the action-value function, describing the expected return starting from state s_t, taking action a_t, and following π, defined as (4), and $V^{\pi}(s)$ is the state-value function, which provides the expected return starting from state s_t under policy π [72], expressed in (5).

$$Q^{\pi}(s,a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a\right]. \tag{4}$$

$$V^{\pi}(s) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s\right]. \tag{5}$$

A common approach in implementing actor-critic-based policy gradient methods involves using Generalized Advantage Estimation (GAE) [73]. GAE introduces a discounted, exponentially weighted sum of temporal difference (TD) errors to balance short-term and long-term return estimates. It addresses the bias-variance trade-off in advantage estimation by interpolating between high-bias, low-variance (1-step TD) estimators and lower-bias, higher-variance (n-step TD) estimators. This flexibility allows GAE to compute more stable and accurate advantage values, improving policy updates. Mathematically, GAE is expressed as:

$$A^{\pi}_{\mathrm{GAE}}(s_t, a_t) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \tag{6}$$

where $\lambda \in [0,1]$ controls the trade-off between bias and variance. Smaller values of λ place greater emphasis on the value function estimate, while larger values give more weight to the actual rewards. δ_t in (6) is the TD error, defined as:

$$\delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t). \tag{7}$$

The relative difference is defined in (2) acts as a measure of policy improvement. A positive difference indicates that the updated policy, π', is better than the previous policy. In each policy iteration, the goal is to select a new policy π' that maximizes this difference. Consequently, optimizing the objective $J(\pi')$ is equivalent to maximizing this performance gap, and both can be achieved through gradient ascent. Therefore, we can express it as follows:

$$\max_{\pi'} J(\pi') \Leftrightarrow \max_{\pi'} \big(J(\pi') - J(\pi)\big). \tag{8}$$

At its core, PPO employs a surrogate objective function that ensures steady policy improvement, that is $J(\pi') - J(\pi) \geq 0$. However, a limitation arises as the expectation in (2) requires trajectories from π', which is not available until after the update. To address this, we assume that two policies are close, so their state distributions are similar. This allows us to approximate the equation using trajectories from π with importance sampling weights. We can express the probability ratio between the new policy and the old policy, parameterized by θ, as follows:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, \tag{9}$$

where $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t in the state s_t under the new policy, and $\pi_{\theta_{old}}(a_t|s_t)$ is the probability under the old policy. Thus, we can get $r_t(\theta_{\text{old}}) = 1$ from it. This allows us to rewrite the objective function (2), with advantages calculated using the older policy, as follows:

$$J(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t^{\pi_{\theta_{old}}}(s_t, a_t)\right] = \mathbb{E}_t[r_t(\theta)A_t]. \tag{10}$$

To control and prevent large, unstable policy updates, a constrained clipped version of the surrogate function is employed, restricting the policy updates to stay within a trust region that is both easy to implement and computationally efficient, as follows:

$$J^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \tag{11}$$

where ϵ is a hyperparameter that defines the clipping neighborhood and can be decayed during training. The function takes the minimum of the unclipped and the clipped objective to ensure conservative updates. PPO updates the policy iteratively using gradient ascent, with α representing the learning rate, as follows:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J^{CLIP}(\theta). \tag{12}$$

PPO includes a separate objective to train the value function $V^\pi(s)$ using mean-squared error:

$$J_t^{VF}(\theta) = \mathbb{E}_t[(V_\theta(s_t) - R_t)^2]. \tag{13}$$

PPO clips the value function like the clipped surrogate objective by minimizing the following loss:

$$V_{\text{clip}}(s_t) = \text{clip}(V_\theta(s_t), V_{old}(s_t) - \epsilon_v, V_{old}(s_t) + \epsilon_v), \tag{14}$$

$$J_t^{VF}(\theta) = \mathbb{E}_t\left[\max\left((V_\theta(s_t) - R_t)^2, \left(V_{\text{clip}}(s_t) - R_t\right)^2\right)\right]. \tag{15}$$

A study by [74] found no empirical evidence supporting the effectiveness of value function loss clipping in improving performance. Furthermore, [75] argue that $V_{\text{clip}}(s_t)$ may even hinder performance. We empirically tested this in our case and observed that value clipping does not enhance our model's performance and, thus, is not used in our implementation.

Decaying the ϵ values for clipping helps stabilize training in PPO by gradually reducing the allowed deviation. Initially, larger clipping values allow more freedom for exploration and faster updates, while decaying ϵ encourages more conservative updates as training progresses. For ϵ clipping range, we can use the following linear decay:

$$\epsilon_t = \epsilon_{start} + (\epsilon_{end} - \epsilon_{start}) \times \left(1 - \frac{t}{T}\right), \tag{16}$$

where t is the current timestep out of the total timesteps T, ϵ_t is the current epsilon value, and ϵ_{start} and ϵ_{end} are the starting and ending values of the clipping range.

The total loss combines the policy objective, value function loss objective, and an entropy term to ensure sufficient exploration:

$$J_t^{TOTAL}(\theta) = \mathbb{E}_t[J_t^{CLIP}(\theta) - c_1 J_t^{VF}(\theta) + c_2 H(\pi_\theta)], \tag{17}$$

where c_1 and c_2 are coefficients to control the importance of each term, and $H(\pi_\theta)$ is the entropy of the policy for exploration. The flow diagram of the PPO is depicted in Fig. 2.

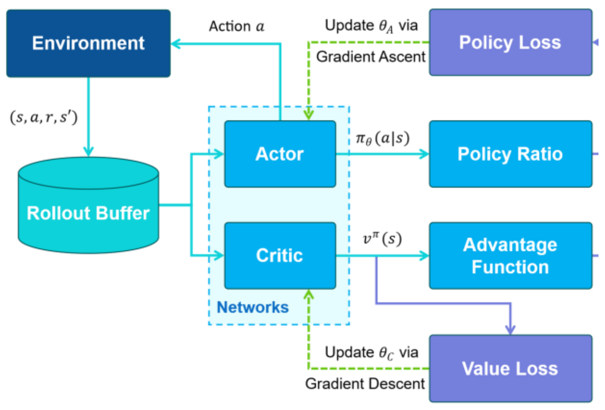


Fig. 2: The data flow diagram of the PPO framework.

The camera captures RGB images with a field of view of 90° and dimensions of $(3, 50, 50)$. To approximate the environment's underlying POMDP, we stack input images to provide the agent with a short history of observations. This effectively captures temporal dependencies and reduces ambiguity in state representation. We transform the RGB images to grayscale, normalize them to the range $[-1,1]$, and stack the subsequent captured frames along the width dimension. The stack is processed through CNN layers, flattened, and fed into a fully connected layer for further processing. The output features are then passed to the actor and critic networks to determine the actions and estimate the values. It is important to note that actions don't directly generate the output. Instead, the actor-network produces the mean and sigma of a normal distribution from which actions are sampled. The architecture of the proposed method is illustrated in Fig. 3.

This approach transforms the task closer to a first-order MDP by embedding critical contextual information from recent frames, allowing the agent to infer short-term dependencies on the current state more accurately. By doing so, we eliminate the need for computationally expensive recurrent mechanisms, such as RNNs, which explicitly model long-term dependencies but often introduce significant overhead.

This balance enhances efficiency while maintaining effective policy learning in partially observable environments. The implementation details are outlined in Algorithm 1.

A. Action Space

The action space is a high-level, two-dimensional continuous control space that directly manages the linear velocities along the principal axes in the UAV's local NED coordinate system. These velocities are defined in meters per second and are illustrated in Fig. 4. The movement along the x-axis is set to a fixed value and remains unchanged throughout the flight.

Consequently, the quadrotor maintains constant forward motion while the model controls the lateral translation along the y-axis and the altitude translation along the z-axis. Each action is applied to the drivetrain for 0.5 seconds.

B. Reward Function

The reward function consists of three terms: the first term rewards the agent consistently after passing the first passage based on the total number of passages. It is calculated as:

$$r_1 = \left\{a + i \cdot \frac{b-a}{N-1} \,|\, i = 0,1,\dots,N-1\right\}, \tag{18}$$

where a is the starting value, b is the ending value, N is the number of total passages, and i is the index of the current passage; the second term accounts for the distance from the center of the current passage. Each passage is an imaginary cube with all three principal axes of a length of 2 meters.

Algorithm 1: Proximal Policy Optimization

```
    Input: single-channel image with size $(C,H,W)$
    Output: continuous action $a_t$
1:  Set $c_2 \geq 0$, entropy regularization weight
2:  Set $c_1 \geq 0$, importance weight of $J_t^{VF}(\theta)$
3:  Set $\epsilon \geq 0$, the clipping variable
4:  Set $K$, the number of epochs
5:  Set $T$, the time horizon
6:  Set $M \leq T$, the minibatch size
7:  Set $\alpha \geq 0$, the learning rate
8:  Set $N$, the stack size
9:  Initialize the actor and critic parameters $\theta_A, \theta_C$ with
    Orthogonal initialization for weights and constant for
    biases.
10: Initialize the old actor network $\theta_{A_{old}}$
11: for $i = 1,2,\dots,MAX\_STEPS$ do
12:     Initialize an empty rollout buffer
13:     Initialize $s_0$ as zeros with size $(C,H,W \times N)$
14:     Observe $o_t$
15:     for $j = 1,2,\dots,T$ do
16:         $s_t \leftarrow$ Roll the stack along the width dimension and add
            $o_t$
17:         Apply action $a_t \sim \pi_{\theta_A}(a_t|s_t)$
18:         Receive $r_t$ and observe $o_{t+1}$
19:         Record $(o_t, a_t, r_t, o_{t+1})$ into rollout buffer
20:         $o_t \leftarrow o_{t+1}$
21:     end for
22:     Set $\theta_{A_{old}} = \theta_A$
23:     Calculate $V_{tar,1}^{\pi}, \dots, V_{tar,T}^{\pi}$ using the critic network $\theta_C$
24:     Compute advantages $A_1, \dots, A_T$ using (6) with $\theta_{A_{old}}$
25:     Let $batch$ with size $T$ consist of the collected trajectories,
        advantages, and target values
26:     for $epoch = 1,2,\dots,K$ do
27:         for minibatch $m$ in $batch$ do
28:             Calculate $r_m(\theta_A)$
29:             Calculate $J_m^{CLIP}(\theta_A)$ using (11), with the advantages
                $A_m$ and $r_m(\theta_A)$
30:             Calculate entropies $H_m$ using $\theta_A$
31:             Calculate policy loss: $J_{pol}(\theta_A) = J_m^{CLIP}(\theta_A) - c_2 H_m$
                based on (17)

32:             Calculate predicted values $\hat{V}^{\pi}(s_m)$ using $\theta_C$
33:             Calculate value loss $J_{val}(\theta_C)$ using (15)

34:             Update actor parameters $\theta_A$ according to (12) using
                calculated $J_{pol}(\theta_A)$
35:             Update critic parameters $\theta_C$ according to (12) using
                calculated $J_{val}(\theta_C)$
36:         end for
37:     end for
38:     Decay clipping $\epsilon$ variable
39: end for
```

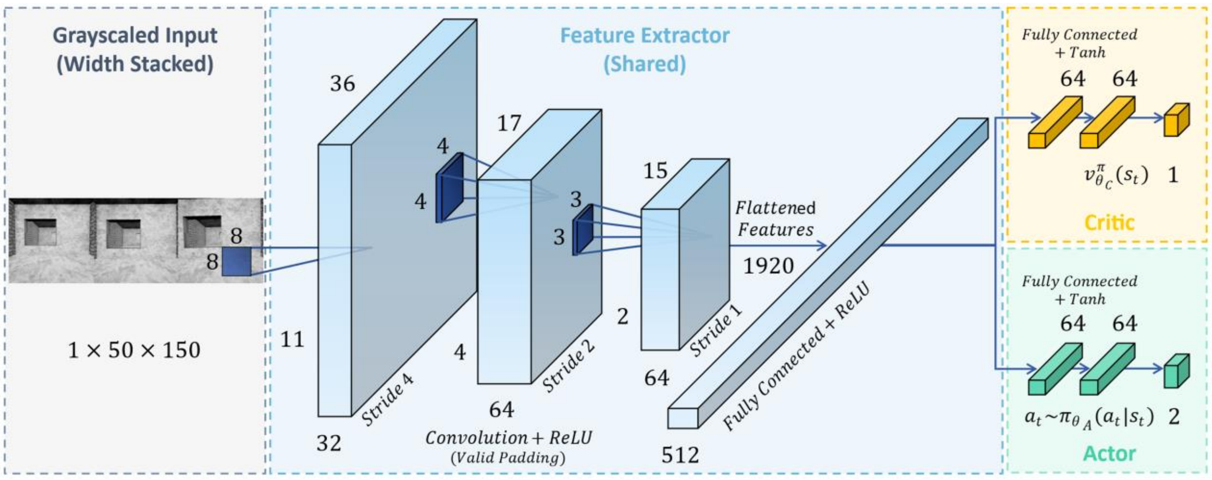

Grayscaled Input (Width Stacked)
1 × 50 × 150
Feature Extractor (Shared)
Stride 4
Stride 2
Stride 1
Convolution + ReLU (Valid Padding)
Flattened Features
1920
Fully Connected + ReLU
512
Fully Connected + Tanh
64 64
$v^{\pi}_{\theta_C}(s_t)$ 1
Critic
$a_t \sim \pi_{\theta_A}(a_t|s_t)$ 2
Actor

Fig. 3: Architecture of the proposed PPO model with consideration of temporal correlations. The number of stacked frames is 3. Number of filters, stride and output size are mentioned for each convolutional layer.

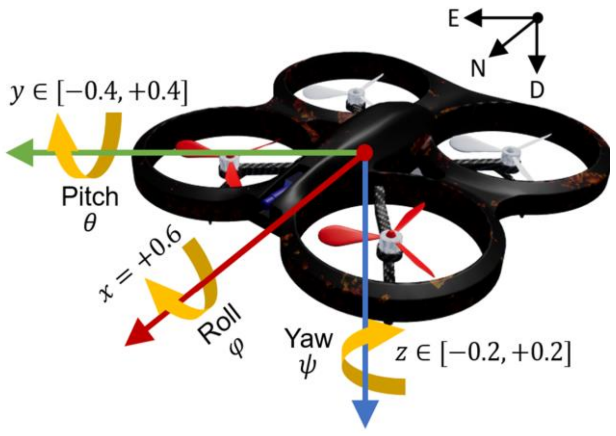


Fig. 4: Overview of the high-level continuous action space, controlling lateral (y-axis) and altitude (z-axis) velocities, with constant forward motion along the x-axis.

Based on the UAV's distance from the cube's center, it receives a negative reward, which becomes less penalized as it gets closer to the center.

Conversely, when the UAV is inside the imaginary cube, the negative reward transitions to a positive reward:

$$r_2 = \begin{cases} \sqrt{\dfrac{1}{\left|y_{\text{uav}} - \frac{y_{\text{cube}}}{2}\right|} + \dfrac{1}{\left|z_{\text{uav}} - \frac{z_{\text{cube}}}{2}\right|}}, & \text{if inside} \\ -\left(\sqrt{\|p_a - p_b\|}\right), & \text{if outside} \end{cases}, \quad (19)$$

where $p_a = (y_{\text{uav}}, z_{\text{uav}})$ and $p_b = (y_{\text{cube}}, z_{\text{cube}})$ represent the UAV's position and the imaginary cube position, respectively; the third reward term provides a larger constant reward if all passages are successfully passed. In the case of a collision, a penalty is imposed, and the trajectory terminates due to a critical failure. The final reward is computed as the sum of all the terms:

$$r_t = r_1 + r_2 + r_3. \quad (20)$$

The parameters of the reward function are determined empirically to achieve the best results.

The reward function implicitly plays a crucial role in addressing safety issues by directing the UAV to maintain its position in the center as it moves through narrow passages. It improves navigation and helps avoid obstacles while significantly decreasing the likelihood of crashes, especially in tight spaces.

Results and Discussion

All experiments are conducted on a standard laptop with an Intel Core i7-6700HQ processor running at 2.6 GHz, 16 GB of RAM, and a GeForce GTX 960M GPU with 4 GB of VRAM.

We use the AirSim simulation platform [76] to achieve realistic simulations. This platform is specifically designed to develop RL algorithms for autonomous agents in real-world scenarios. AirSim accurately simulates the environmental and aerodynamic properties of UAVs by utilizing the graphical capabilities of Unreal Engine to create a practical virtual environment. This paper focuses on using a multi-rotor UAV, known for its superior hover and maneuverability performance [48], making it an ideal choice for civilian applications outside the military domain.

We design our environment based on [77], structuring it into several sections. A wall with a narrow passage separates each section, requiring the quadrotor to navigate through to proceed. The final section transitions the quadrotor from an indoor to an outdoor environment, which then marks the task as completed. The training environment and its corresponding map are demonstrated in Fig. 5. The dimensions of the elements in the environment are shown in Fig. 6. To assess the effectiveness and generalization of the policy, we conduct experiments using the training setup and scenarios with varied passage arrangements.

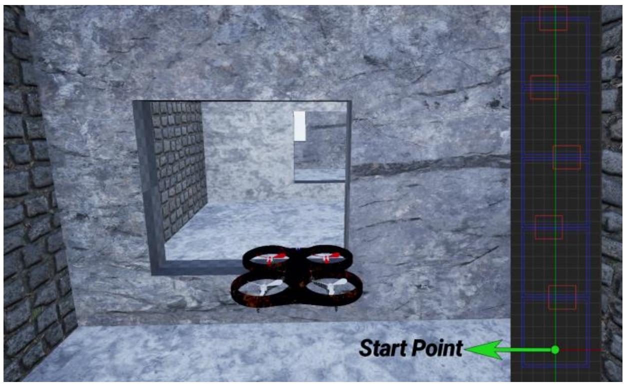


Fig. 5: The training environment with its map.

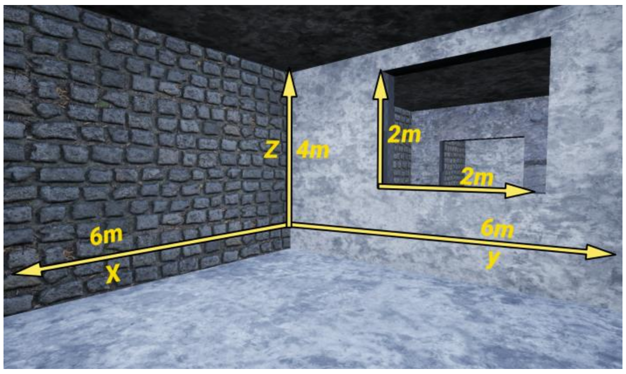


Fig. 6: Dimensions of each section and passage in the environment, all in meters.

A. Performance Comparison

For comparison, we evaluate our method against closely related approaches from [28] and [49]. Additionally, we evaluate our method against recent and state-of-the-art algorithms, including Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3), and Soft Actor-Critic (SAC). Furthermore, we include an analysis of the number of frames and evaluate its impact on performance. The training hyperparameters for each environment are detailed in Table 1.

The main criterion for assessing a navigation algorithm is the success rate, which is determined by the formula $S/K \times 100\%$, where S represents the count of successful trajectories and K indicate the number of times the algorithm has been executed [42].

Fig. 7 depicts the training performance results. As evident from the graphs in this figure, the plots (a), (b), and (c) share significant similarities with plots (a) and (b) demonstrating a strong correlation. As training progresses in the environment, the first observable trend in these graphs is that the agent learns effectively from the interactions over time, resulting in an increase in the average episode length across most approaches. This indicates that the agent learns an effective policy through iterative weight updates driven by gradient descent signals from network errors.

Table 1: Training hyperparameters

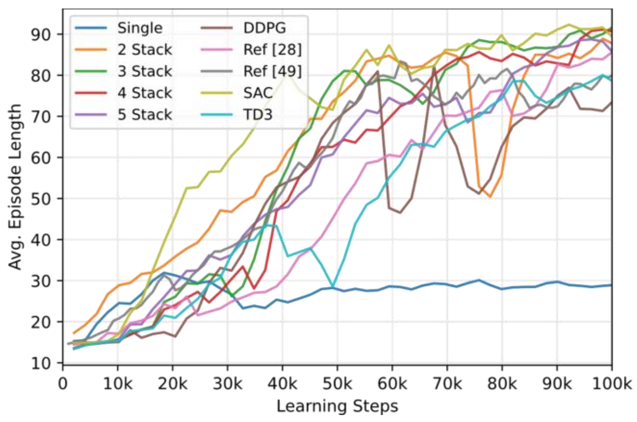
Hyperparameter	Value
Optimizer	Adam
Seed	2024
Frame Stack	3
Learning Rate	0.00007
Number Of Rollout Steps	2048
Batch Size	128
Update Epochs	30
Discount Factor	0.99
GAE Coefficient	0.7
Max Gradient Norm	0.5
Start Clipping Value	0.3
End Clipping Value	0.05
Entropy Coefficient	0
Value Function Coefficient	1
Total Learning Steps	100,000

Moreover, the rise in average episode length is accompanied by increased rewards, further highlighting effective learning. Among the approaches compared in this figure, it is evident that when using single grayscale images, there is no notable improvement in the average episode length or rewards compared to other methods. This suggests that the agent needs assistance to derive a meaningful understanding of the required policy from a single image and its extracted features, resulting in a failure of this approach.

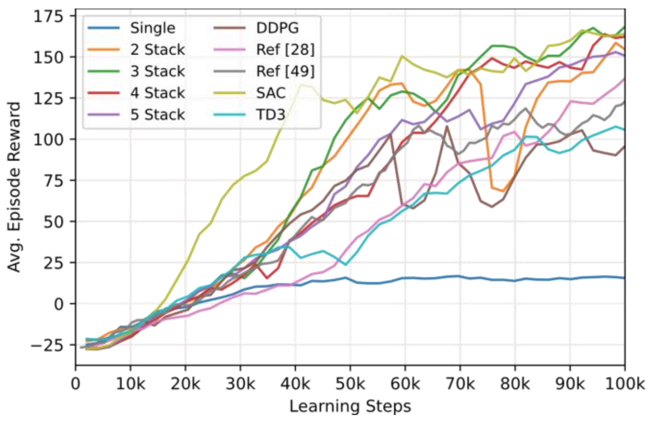
Given that actions are continuous, it becomes clear that considering correlations among multiple images is necessary to capture the influence of action changes on the input state space. This understanding can be established by introducing a short-term dependency through stacking input images.

For instance, when the stack size is two, the agent not only achieves a higher initial average episode length compared to other approaches but also demonstrates significantly better results in terms of average success rate, as shown in the plot (c), even compared to cases with larger stack sizes.

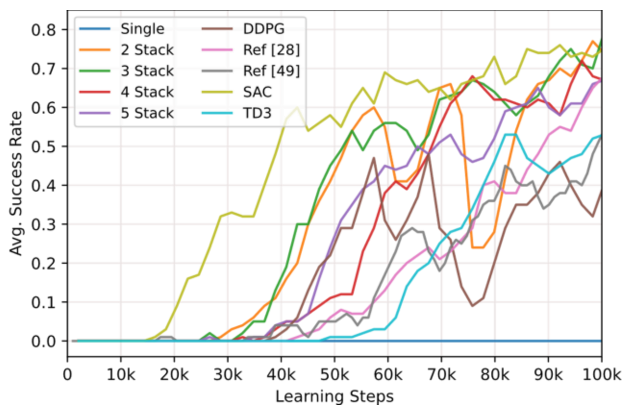
However, as seen across all plots and most noticeably in plot (d), this approach exhibits more significant fluctuations than others, indicating a need for more stability.



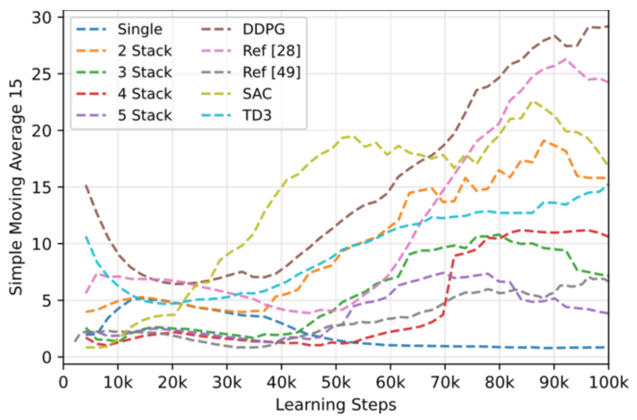
(a) Avg. Episode Length



(b) Avg. Episode Reward



(c) Avg. Success Rate



(d) Network Loss

Fig. 7: Training performance results.

Plot (d) shows that the network's error is highly volatile, which may lead to fundamental policy shifts during updates, sometimes steering the agent in counterproductive directions. These adverse effects are evident in plot (c) for this method.

When the stack size is increased to three, all graphs show much better stability, and it is apparent that this approach outperforms others significantly. This approach achieves the best performance across all metrics. Increasing the stack size to four or five results in slower learning rates in terms of success rate and rewards, alongside a higher computational overhead, without yielding substantial performance improvements. Approaches [28] and [49] demonstrate considerably weaker performance in solving the task and achieving success rates, further validating the reliability and effectiveness of the proposed method. The full detail of the experiments done is outlined in Table 2.

Unlike comparison methods, DDPG has the longest average episode length but generates a relatively low average reward and experiences a high collision rate. Its elevated loss indicates instability during training, primarily due to the difficulties associated with value-based learning in complex continuous-action environments. DDPG requires more training steps to achieve optimal policy refinement.

On the other hand, TD3 shows improved stability. However, its high loss and collision rate also imply that it needs more training steps.

This is due to the use of additional networks that rely on approximations. In contrast, SAC demonstrates a higher reward and success rate, ranking as the second-best method with a relatively low loss value. This success highlights how entropy-regularized policy learning effectively balances exploration and exploitation. Hence, SAC exhibits faster initial learning, as shown in Fig. 7, plots (b) and (c).

SAC's performance closely aligns with the two-frame stack setup.

However, the three-frame stack configuration achieves the highest average reward and success rate while maintaining the lowest collision rate, outperforming all other setups, including SAC. Our approach yields slow but gradual progress.

Table 2: Performance results of the final rollout during the training phase

Method	Avg. Episode Length	Avg. Episode Reward ↑	Success Rate ↑	Collision Rate ↓	Loss
Single	28.95	15.43	0.00	1.00	0.94
2 Stack	87.55	153.64	0.74	0.26	5.24
3 Stack	91.95	**169.56**	**0.79**	**0.21**	2.35
4 Stack	90.62	162.42	0.67	0.33	3.59
5 Stack	85.26	150.26	0.67	0.33	1.30
DDPG	96.91	73.91	0.40	0.60	25.37
TD3	78.33	105.22	0.53	0.47	21.74
SAC	88.92	164.28	0.75	0.25	6.69
Ref [28]	85.98	137.96	0.68	0.32	17.44
Ref [49]	80.29	124.00	0.54	0.46	3.69

B. Computational Cost Analysis

Computational cost analysis of image stack configurations provides insight into performance and resource consumption. Table 3 presents the inference time, frames per second (FPS), number of parameters, and VRAM allocation comparison for different stack sizes. Among these configurations, the three-frame stack achieves a balanced performance in terms of decision-making effectiveness while maintaining manageable computational overhead.

Table 3: Computational cost comparison for different frame stack during evaluation iterations

N-Frame	Inference Time (ms)	Frames Per Second	Number of Parameters	VRAM Allocation (MB)
One	8.17	17.79	277,603	3.20
Two	11.26	17.24	736,355	8.45
Three	20.55	16.58	1,129,571	12.95
Four	20.74	16.51	1,522,787	17.45
Five	21.07	16.37	1,916,003	21.95

The three-frame stack achieves an inference time of 20.55 ms, delivering 16.58 processed frames per second. Although the computational cost is slightly higher than that of the two-frame stacks, this increase is subtle as decision-making accuracy and stability are improved. Additionally, the three-frame stack requires 1,129,571 parameters and consumes 12.95 MB of VRAM, representing a reasonable trade-off between performance gains and resource usage. Compared to the four-frame and five-frame configurations, the three-frame stack maintains a lower computational footprint without sacrificing performance. Furthermore, the three-frame stack requires no complex hardware and achieves remarkably sounder performance with minimal additional computational cost compared to a single-frame input.

The network's number of parameters increases as more frames are stacked. While the three-frame stack has higher parameters than the single-frame setup, it remains well within the capabilities of UAV hardware. Compared to more frame stacks, which exceed 1.5 million parameters, the three-frame stack offers a more efficient balance between performance and computational demands. This configuration ensures that UAVs can process visual data effectively without overloading onboard processors, making it a practical choice for real-time navigation, especially in GPS-denied environments.

Fig. 8 further illustrates these findings. The comparative representation depicts the trade-offs between performance and computational cost across different stack sizes.

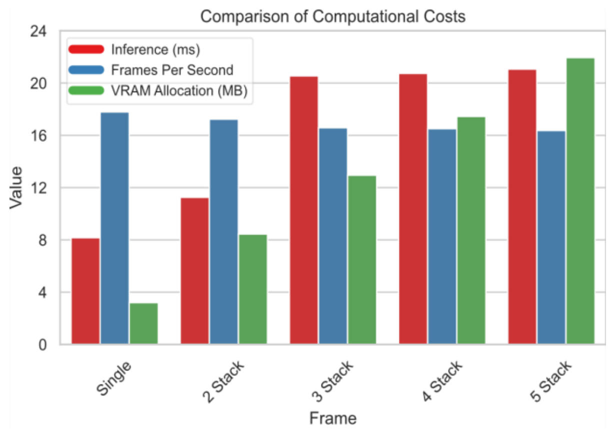


Fig. 8: Performance and computational cost across different frame stack configurations.

Based on the results, frame stacking enhances UAV navigation performance without adding additional overhead. Compared to single-frame input, stacking allows the UAV to utilize temporal visual cues, leading to more accurate decision-making while maintaining computational efficiency. This balance makes the frame well-suited for deployment on resource-constrained edge devices.

C. Generalization Verification

We conduct two tests to assess the model's generalization capability under different environmental variations. In the first test, we change the positions of the passageways, while in the second test, we modify both their position and shape from cubes to spheres. Fig. 9 illustrates the design and map of the second generalization test. Fig. 10 illustrates the results.

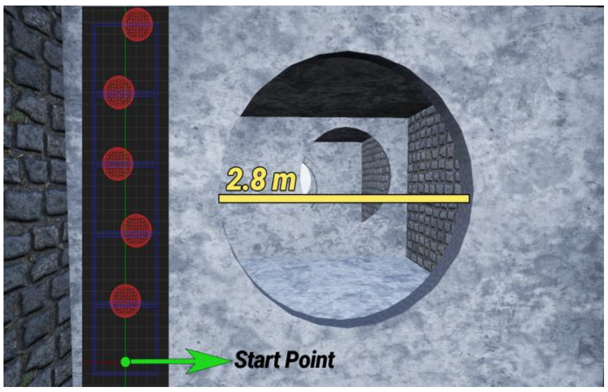
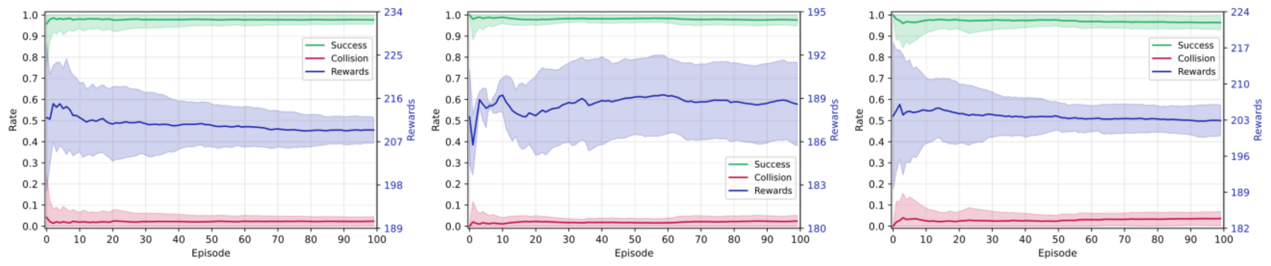


Fig. 10: Setup and map of the second generalization testing environment.



(a) Main Setup (b) Generalization 1 (c) Generalization 2

Fig. 9: Performance evaluation during the testing phases over 25 sets of 100 iterations, with shaded areas indicating the standard deviation across sets.

The proposed approach shows consistent stability in task completion and safety across various test scenarios. Success and collision rates remain within reasonable, acceptable ranges. There are slight variations in episode rewards, corresponding to expected behavioral adaptations to different environmental setups. The reward variations result from minor trade-offs in navigation efficiency, rather than from failures in core task execution. In particular, the agent receives more rewards when it is closer to the center of the passageway. As a result, confidence in the rewards is anticipated.

The comprehensive results of all tests are presented in Table 4.

Table 3: Performance results in the testing phases over 25 sets of 100 iterations, with their respective standard deviations

Test Category	Avg. Episode Length	Avg. Episode Reward	Success Rate	Collision Rate
Main Setup	97.773 ± 0.482	210.524 ± 5.228	0.978 ± 0.037	0.022 ± 0.037
Generalization 1	97.619 ± 0.633	188.652 ± 2.508	0.981 ± 0.026	0.019 ± 0.026
Generalization 2	97.852 ± 0.564	203.686 ± 4.255	0.971 ± 0.041	0.029 ± 0.041

Regardless of the setup, the model exhibits strong performance in its primary objectives, i.e., success and collision avoidance, demonstrating its ability to adapt to environmental changes. This consistency in critical metrics, accompanied by minor reward deviations, underlines the model's ability to generalize effectively without compromising its functional reliability. For a visual representation of the experiments, please refer to the supplementary movie clip[1].

D. Noise Impact Analysis

To evaluate the model's robustness to sensor noise, we introduce noise at different strengths to simulate varying noise levels. Specifically, distortions are applied at 5%, 10%, and 15% magnitudes to challenge the visual processing capabilities under diverse wavering degrees.

Here, sensor noise serves as a representative factor for broader adverse environmental conditions, such as low light, fog, and camera distortions, strengthening the real-world relevance of this analysis by approximating visual uncertainties that UAVs commonly encounter in practical scenarios.

Horizontal distortion blends the ratio of the noise pixel with the image pixel, which introduces fluctuations in the horizontal line of an image. We aim to assess how the model handles such disturbances.

Fig. 11 illustrates the distortion and fluctuations resulting from different noise levels. Additionally, we introduce 20% random noise blobs alongside the horizontal distortion to better simulate real-world uncertainties.

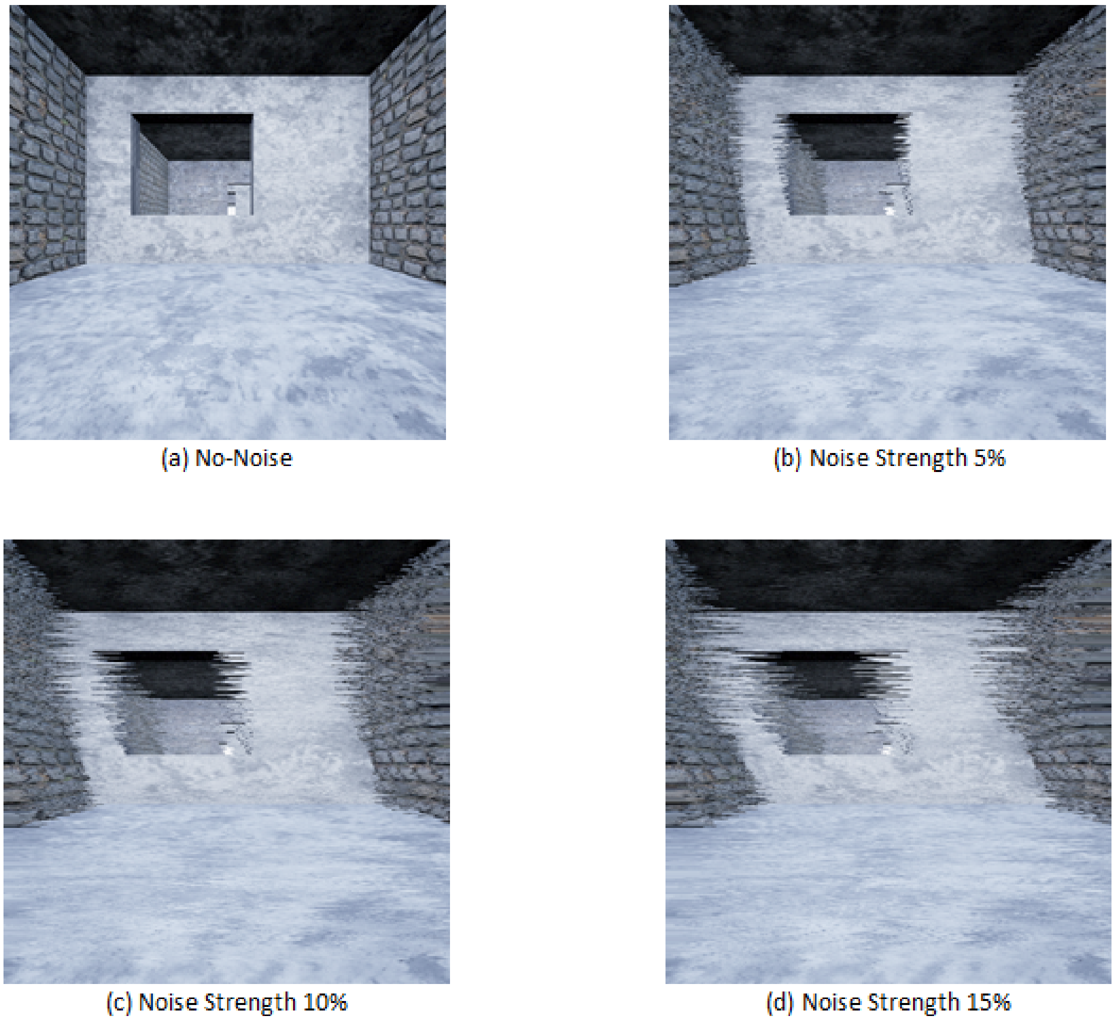

(a) No-Noise (b) Noise Strength 5%
(c) Noise Strength 10% (d) Noise Strength 15%

Fig. 11: Horizontal line distortion in a captured image. Distortion increases when the noise level gets higher.

[1] https://youtu.be/Raf12qr0bps

Fig. 12 shows Gaussian distribution plots of how different noise levels affect the actions. As noise increases, the distribution mean shifts, indicating slight changes in the UAV's behavior. At 5% noise, the distributions of lateral actions remain close to the no-noise baseline. However, the distribution mean shifts slightly more at 10% and 15% noise levels, resulting in more variability in action selection. Although the standard deviation remains constant in each plot, the distribution for lateral movements shifts toward more negative values as noise increases. In contrast, vertical action changes less and shows no consistent pattern in mean shift direction, pointing out that lateral control is more sensitive to noise than vertical movement. As a result, the agent struggles to locate the passage due to horizontal line distortion.

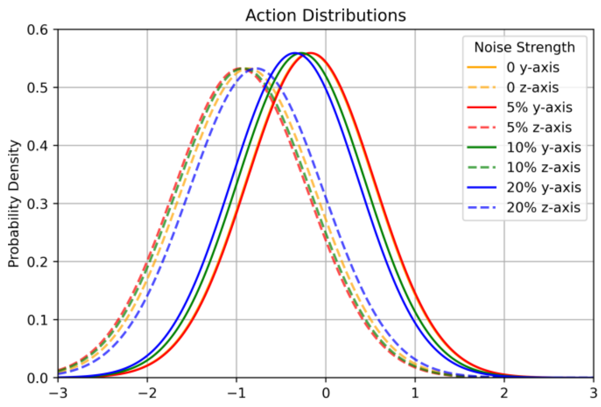


Fig. 12: Gaussian distribution of UAV actions for lateral (y-axis) and vertical (z-axis) movements at different noise levels.

Similarly, Fig. 13 shows the distribution of action data points for both actions under different noise levels. In the no-noise case, the action values are clustered around -1, indicating stable control. As noise increases, the distributions widen, especially along the y-axis, where the peaks become less defined and more spread out. In contrast, the z-axis peak changes moderately with a slight shift in the mean value. This implies that higher noise levels increase uncertainty, with more impact on lateral control.

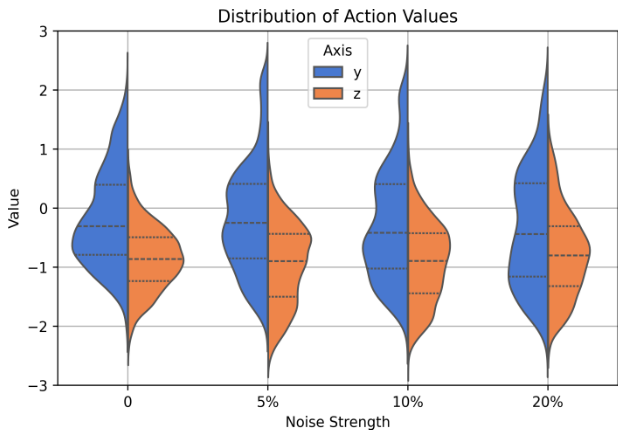
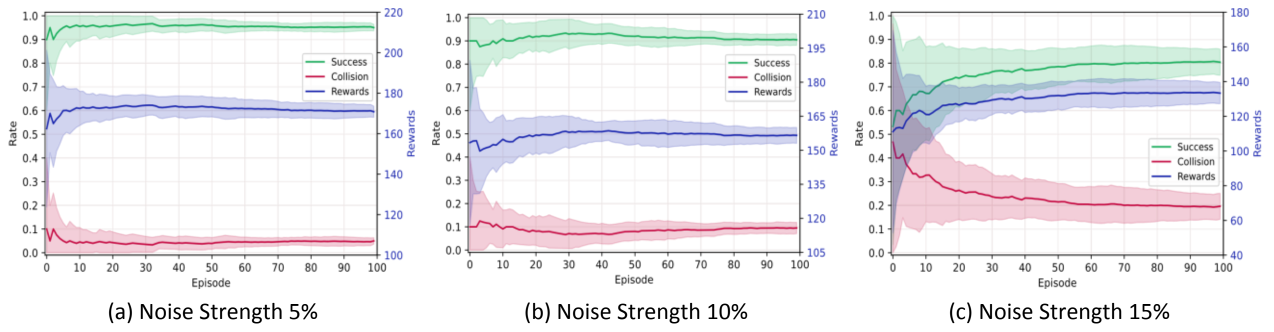


Fig. 14: Distribution of UAV action values for lateral (y-axis) and vertical (z-axis) movements under varying noise strengths.

Table 5 evaluates the navigation performance under sensor noise uncertainty, while Fig. 14 provides a visual representation, with shaded areas indicating the standard deviation across all execution sets.

Table 4: Performance assessment with sensor noise addition during the testing phase over 10 sets of 100 iterations

Noise Strength	Avg. Episode Length	Avg. Episode Reward	Success Rate	Collision Rate
5%	97.186 ± 1.035	172.127 ± 5.925	0.954 ± 0.036	0.046 ± 0.036
10%	96.238 ± 1.806	156.578 ± 6.736	0.913 ± 0.051	0.087 ± 0.051
15%	91.845 ± 3.955	129.747 ± 10.669	0.762 ± 0.108	0.238 ± 0.108



(a) Noise Strength 5% (b) Noise Strength 10% (c) Noise Strength 15%

Fig. 13: Performance evaluation with sensor noise addition during the testing phase over 10 sets of 100 iterations, with shaded areas indicating the standard deviation across sets.

Despite the added noise in captured images and increased uncertainty in action selection, the three-frame stack method does not rely solely on the current observation to predict the next movement. Since distorted pixels change with each frame, the agent utilizes temporal information to maintain reasonable success and failure rates with 5% noise.

However, as noise intensifies to 10%, the standard deviation area expands, and success trajectories decrease due to higher uncertainty, though the performance impact remains minimal. At 15% noise strength, as shown in Fig. 11, the distortion becomes severe, altering the perceived positions of passageways and making safe navigation challenging. In this scenario, the agent struggles to complete the task safely, resulting in a significant performance drop, with the success rate falling to approximately 0.76, a reduction of about 0.15 compared to the 10% noise setting. These findings indicate that the 15% noise level is too high for an agent not trained to handle such input noise. Thus, the three-frame stacked model can tolerate the noise up to 10% and maintain acceptable performance.

Conclusion

This paper presents a PPO-based approach for autonomous quadrotor navigation in GPS-denied environments using a monocular camera as the primary sensor. We effectively process temporal information and manage limited environmental data by utilizing stacks of grayscale images. The trained policy learns to maximize rewards by favoring actions that yield the highest returns. Using this policy, the agent robustly and precisely controls actions to navigate the aerial vehicle through challenging and narrow passages.

The quadrotor's reliable operation in GPS-denied settings is a tribute to its adaptability and robustness, ensuring its reliability for search and rescue, inspection, and surveillance applications. This paper highlights the transformative potential of DRL methods, such as PPO, particularly when paired with effective feature extraction and processing techniques. Our approach reduces equipment costs, provides fast inference, and facilitates simpler architectures. It is essential to note that our method is validated in simulation, which may not fully capture the aerodynamic effects experienced by UAVs in real-world environments, such as near gaps, walls, or wind gusts. These aerodynamic disturbances can impact the vehicle's dynamics.

In future work, we plan to design an image-based collision probability algorithm to improve performance and extend the method to general path planning in dynamic environments. Additionally, we will conduct real-world validation to ensure robustness to aerodynamic factors and environmental noise. As part of the deployment, explicit safety mechanisms, such as emergency stopping and fault-tolerant behaviors, are planned to increase operational reliability and address unforeseen failures during flight.

Author Contributions

M. Shahbazi Khojasteh conceptualized the study, developed the methodology, and conducted software development, validation, and analysis. A. Salimi-Badr supervised the project and provided manuscript review and feedback.

Funding

This paper has received no external funding.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancy, have been completely witnessed by the authors.

Abbreviations

UAV	Unmanned Aerial Vehicle
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
PPO	Proximal Policy Optimization
AI	Artificial Intelligence
ML	Machine Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
SLAM	Simultaneous Localization and Mapping
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process

References

[1] N. Elmeseiry, N. Alshaer, T. Ismail, "A detailed survey and future directions of Unmanned Aerial Vehicles (UAVs) with potential applications," Aerospace, 8(12): 363, 2021.

[2] M. A. Tahir, I. Mir, T. U. Islam, "A review of UAV platforms for autonomous applications: Comprehensive analysis and future directions," IEEE Access, 11: 52540-52554, 2023.

[3] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, M. H. Alsharif, "Towards the Unmanned Aerial Vehicles (UAVs): A comprehensive review," Drones, 6(6): 147, 2022.

[4] M. Javaid, A. Haleem, S. Rab, R. P. Singh, R. Suman, "Sensors for daily life: A review," Sens. Int., 2: 100121, 2021.

[5] D. J. Yeong, G. Velasco-Hernandez, J. Barry, J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," Sensors, 21(6): 2140, 2021.

[6] K. Telli et al., "A comprehensive review of recent research trends on Unmanned Aerial Vehicles (UAVs)," Systems, 11(8): 400, 2023.

[7] S. Campbell et al., "Sensor technology in autonomous vehicles : A review," in Proc. 2018 29th Irish Signals and Systems Conference (ISSC): 1-4, 2018.

[8] Y. Lu, Z. Xue, G. S. Xia, L. Zhang, "A survey on vision-based UAV navigation," Geo-spatial Inf. Sci., 21(1): 21-32, 2018.

[9] L. He, N. Aouf, B. Song, "Explainable deep reinforcement learning for UAV autonomous path planning," Aerosp. Sci. Technol., 118: 107052, 2021.

[10] M. Kim, J. Kim, M. Jung, H. Oh, "Towards monocular vision-based autonomous flight through deep reinforcement learning," Expert Syst. Appl., 198: 116742, 2022.

[11] J. Li, X. Xiong, Y. Yan, Y. Yang, "A survey of indoor UAV obstacle avoidance research," IEEE Access, 11: 51861-51891, 2023.

[12] S. Rezwan, W. Choi, "Artificial intelligence approaches for UAV navigation: Recent advances and future challenges," IEEE Access, 10: 26320-26339, 2022.

[13] B. Mahdipour, S. H. Zahiri, I. Behravan, "An intelligent two and three dimensional path planning, based on a metaheuristic method," J. Electr. Comput. Eng. Innovations, 13(1): 93-116, 2025.

[14] S. Y. Choi, D. Cha, "Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art," Adv. Rob., 33(6): 265-277, 2019.

[15] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," J. Sens., 2017(1): 3296874, 2017.

[16] R. P. Padhy, S. Verma, S. Ahmad, S. K. Choudhury, P. K. Sa, "Deep neural network for autonomous UAV navigation in indoor corridor environments," Procedia Comput. Sci., 133: 643-650, 2018.

[17] J. Gu et al., "Recent advances in convolutional neural networks," Pattern Recognit., 77: 354-377, 2018.

[18] Y. Chang, Y. Cheng, U. Manzoor, J. Murray, "A review of UAV autonomous navigation in GPS-denied environments," Rob. Auton. Syst., 170: 104533, 2023.

[19] R. S. Sutton, "Reinforcement learning: An introduction," A Bradford Book, MIT Press, Cambridge, MA, 2018.

[20] H. Taheri, S. Rasoul Hosseini, M. A. Nekoui, "Deep reinforcement learning with enhanced ppo for safe mobile robot navigation," arXiv e-prints, p. arXiv:2405.16266, 2024.

[21] A. S. Sadr, M. S. Khojasteh, H. Malek, A. Salimi-Badr, "An efficient planning method for autonomous navigation of a wheeled-robot based on deep reinforcement learning," in Proc. 2022 12th International Conference on Computer and Knowledge Engineering (ICCKE): 136-141, 2022.

[22] S. Mashhouri, M. Rahmati, Y. Borhani, E. Najafi, "Reinforcement learning based sequential controller for mobile robots with obstacle avoidance," in Proc. 2022 8th International Conference on Control, Instrumentation and Automation (ICCIA): 1-5, 2022.

[23] S. Sabzekar, M. Samadzad, A. Mehditabrizi, A. N. Tak, "A deep reinforcement learning approach for UAV path planning incorporating vehicle dynamics with acceleration control," Unmanned Syst., 12(03): 477-498, 2024.

[24] H. S. M. Mahalegi, A. Farhadi, G. Molnár, E. Nagy, "Enhancing UAV autonomous navigation in indoor environments using reinforcement learning and convolutional neural networks," in Proc. 2024 IEEE 22nd Jubilee International Symposium on Intelligent Systems and Informatics (SISY): 000091-000100, 2024.

[25] M. Ramezani, M. A. Amiri Atashgah, A. Rezaee, "A fault-tolerant multi-agent reinforcement learning framework for unmanned aerial vehicles–unmanned ground vehicle coverage path planning," Drones, 8(10): 537, 2024.

[26] P. R. Gervi, A. Harati, S. K. Ghiasi-Shirazi, "Vision-based obstacle avoidance in drone navigation using deep reinforcement learning," in Proc. 2021 11th International Conference on Computer Engineering and Knowledge (ICCKE): 363-368, 2021.

[27] M. S. Khojasteh, A. Salimi-Badr, "Autonomous quadrotor path planning through deep reinforcement learning with monocular depth estimation," IEEE Open J. Veh. Technol., 6: 34-51, 2025.

[28] A. P. Kalidas, C. J. Joshua, A. Q. Md, S. Basheer, S. Mohan, S. Sakri, "Deep reinforcement learning for vision-based navigation of UAVs in avoiding stationary and mobile obstacles," Drones, 7(4): 245, 2023.

[29] A. Pachauri, V. More, P. Gaidhani, N. Gupta, "Autonomous Ingress of a UAV through a window using Monocular Vision," arXiv preprint arXiv:1607.07006, 2016.

[30] T. Bera, A. Sinha, A. K. Sadhu, R. Dasgupta, "Vision based autonomous quadcopter navigation through narrow gaps using visual servoing and monocular SLAM," in Proc. 2019 Sixth Indian Control Conference (ICC): 25-30, 2019.

[31] J. Kumar, Himanshu, H. Kandath, P. Agrawal, "Vision based micro-uav navigation through narrow passages," in Proc. 2023 11th International Conference on Control, Mechatronics and Automation (ICCMA): 97-102, 2023.

[32] F. Valenti, D. Giaquinto, L. Musto, A. Zinelli, M. Bertozzi, A. Broggi, "Enabling computer vision-based autonomous navigation for unmanned aerial vehicles in cluttered gps-denied environments," in Proc. 2018 21st International Conference on Intelligent Transportation Systems (ITSC): 3886-3891, 2018.

[33] S. Veerawal, S. Bhushan, M. R. Mansharamani, B. Sharma, "Vision based autonomous drone navigation through enclosed spaces," in Proc. Computer Vision and Image Processing: 104-115, 2021.

[34] R. Xiao, H. Du, C. Xu, W. Wang, "An efficient real-time indoor autonomous navigation and path planning system for drones based on RGB-D sensor," in Proc. 2019 Chinese Intelligent Automation Conference: 32-44, 2020.

[35] A. G. Caldeira, J. V. R. Vasconcelos, M. Sarcinelli-Filho, A. S. Brandão, "UAV path-following strategy for crossing narrow passages," in Proc. 2020 International Conference on Unmanned Aircraft Systems (ICUAS): 26-31, 2020.

[36] Y. Xue, W. Chen, "A UAV navigation approach based on deep reinforcement learning in large cluttered 3D environments," IEEE Trans. Veh. Technol., 72(3): 3001-3014, 2023.

[37] S. Aggarwal, N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," Comput. Commun., 149: 270-299, 2020.

[38] D. Debnath, F. Vanegas, J. Sandino, A. F. Hawary, F. Gonzalez, "A review of UAV path-planning algorithms and obstacle avoidance methods for remote sensing applications," Remote Sens., 16(21): 4019, 2024.

[39] B. Y. Li, H. Lin, H. Samani, L. Sadler, T. Gregory, B. Jalaian, "On 3D autonomous delivery systems: Design and development," in Proc. 2017 International Conference on Advanced Robotics and Intelligent Systems (ARIS): 1-6, 2017.

[40] A. Puente-Castro, D. Rivero, A. Pazos, E. Fernandez-Blanco, "A review of artificial intelligence applied to path planning in UAV swarms," Neural Comput. Appl., 34(1): 153-170, 2022.

[41] J. Gao, Y. Zheng, K. Ni, Q. Mei, B. Hao, L. Zheng, "Fast path planning for firefighting UAV based on a-star algorithm," J. Phys. Conf. Ser., 2029(1): 012103, 2021.

[42] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, P. Wang, "Path planning techniques for mobile robots: Review and prospect," Expert Syst. Appl., 227: 120254, 2023.

[43] W. Zu, G. Fan, Y. Gao, Y. Ma, H. Zhang, H. Zeng, "Multi-UAVs cooperative path planning method based on improved rrt algorithm," in Proc. 2018 IEEE International Conference on Mechatronics and Automation (ICMA): 1563-1567, 2018.

[44] Y. Guo, X. Liu, X. Liu, Y. Yang, W. Zhang, "FC-RRT*: An improved path planning algorithm for UAV in 3D complex environment," ISPRS Int. J. of Geo-Inf., 11(2): 112, 2022.

[45] K. N. McGuire, G. C. H. E. de Croon, K. Tuyls, "A comparative study of bug algorithms for robot navigation," Rob. Auton. Syst., 121: 103261, 2019.

[46] M. Vazirpanah, S. H. Attarzadeh-Niaki, A. Salimi-Badr, "ROS-based co-simulation for formal cyber-physical robotic system design," in Proc. 2022 27th International Computer Conference, Computer Society of Iran (CSICC): 1-5, 2022.

[47] N. Mahdian, S. H. Attarzadeh-Niaki, A. Salimi-Badr, "A systematic embedded software design flow for robotic applications," in Proc. 2021 11th International Conference on Computer Engineering and Knowledge (ICCKE): 217-222, 2021.

[48] T. Elmokadem, A. V. Savkin, "Towards fully autonomous UAVs: A survey," Sensors, 21(18): 6223, 2021.

[49] S. Y. Shin, Y. W. Kang, Y. G. Kim, "Automatic drone navigation in realistic 3D landscapes using deep reinforcement learning," in Proc. 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT): 1072-1077, 2019.

[50] Y. Yu, X. Si, C. Hu, J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," Neural Comput., 31(7): 1235-1270, 2019.

[51] S. Chehelgami, E. Ashtari, M. A. Basiri, M. Tale Masouleh, A. Kalhor, "Safe deep learning-based global path planning using a fast collision-free path generator," Rob. Auton. Syst., 163: 104384, 2023.

[52] R. J. Alitappeh, N. Mahmoudi, M. R. Jafari, A. Foladi, "Autonomous robot navigation: Deep learning approaches for line following and obstacle avoidance," in Proc. 2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP): 1-6, 2024.

[53] L. O. Rojas-Perez, J. Martinez-Carranza, "DeepPilot: A CNN for autonomous drone racing," Sensors, 20(16): 4524, 2020.

[54] S. Daftry, S. Zeng, J. A. Bagnell, M. Hebert, "Introspective perception: Learning to predict failures in vision systems," in Proc. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 1743-1750, 2016.

[55] A. V. R. Katkuri, H. Madan, N. Khatri, A. S. H. Abdul-Qawy, K. S. Patnaik, "Autonomous UAV navigation using deep learning-based computer vision frameworks: A systematic literature review," Array, 23: 100361, 2024.

[56] Z. Xue, T. Gonsalves, "Vision based drone obstacle avoidance by deep reinforcement learning," AI, 2(3): 366-380, 2021.

[57] S. S. Mousavi, M. Schukat, E. Howley, "Deep reinforcement learning: An overview," in Proc. SAI Intelligent Systems Conference (IntelliSys): 426- 440, 2016.

[58] F. AlMahamid, K. Grolinger, "Autonomous unmanned aerial vehicle navigation using reinforcement learning: A systematic review," Eng. Appl. Artif. Intell., 115: 105321, 2022.

[59] A. K. Shakya, G. Pillai, S. Chakrabarty, "Reinforcement learning algorithms: A brief survey," Expert Syst. Appl., 231: 120495, 2023.

[60] A. Singla, S. Padakandla, S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," IEEE Trans. Intell. Transport Syst., 22(1): 107-118, 2021.

[61] O. Walker, F. Vanegas, F. Gonzalez, S. Koenig, "A deep reinforcement learning framework for UAV navigation in indoor environments," in Proc. 2019 IEEE Aerospace Conference: 1-14, 2019.

[62] Y. Chen, N. González-Prelcic, R. W. Heath, "Collision-Free UAV Navigation with a Monocular Camera Using Deep Reinforcement Learning," in Proc. 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP): 1-6, 2020.

[63] C. Wang, J. Wang, Y. Shen, X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," IEEE Trans. Veh. Technol., 68(3): 2124-2136, 2019.

[64] F. Garcia, E. Rachelson, "Markov decision processes," in Markov Decision Processes in Artificial Intelligence: 1-38, 2013.

[65] M. Hausknecht, P. Stone, "Deep recurrent q-learning for partially observable MDPs," in Proc. 2015 AAAI Fall Symposium: 9, 2015.

[66] H. Kurniawati, "Partially observable Markov decision processes and robotics," Annu. Rev. Control Rob. Auton. Syst., 5: 253-277, 2022.

[67] L. Graesser and W. L. Keng, Foundations of deep reinforcement learning: theory and practice in Python. Addison-Wesley Professional, 2019.

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

[69] R. Hoseinnezhad, "A comprehensive review of deep learning techniques in mobile robot path planning: Categorization and analysis," Appl. Sci., 15(4): 2179, 2025.

[70] T. Czarnecki, M. Stawowy, A. Kadłubowski, "Cost-effective autonomous drone navigation using reinforcement learning: simulation and real-world validation," Appl. Sci., 15(1): 179, 2025.

[71] W. Skarka, R. Ashfaq, "Hybrid machine learning and reinforcement learning framework for adaptive UAV obstacle avoidance," Aerosp., 11(11): 870, 2024.

[72] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, "Deep reinforcement learning: a brief survey," IEEE Signal Process. Mag., 34(6): 26-38, 2017.

[73] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," arXiv preprint arXiv:1506.02438, 2015.

[74] L. Engstrom et al., "Implementation matters in deep rl: A case study on ppo and trpo," in Proc. ICLR 2020 Conference Blind Submission, 2020.

[75] M. Andrychowicz et al., "What matters for on-policy deep actor-critic methods? a large-scale study," in Proc. ICLR 2021 Conference, 2021.

[76] S. Shah, D. Dey, C. Lovett, A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," Cham, 2018: Springer International Publishing, in Field and Service Robotics, pp. 621-635.

[77] B. Kabas, "Autonomous UAV navigation via deep reinforcement learning using PPO," in Proc. 2022 30th Signal Processing and Communications Applications Conference (SIU): 1-4, 2022.

Biographies

Mahdi Shahbazi Khojasteh was born in Tehran, Iran, in 1998. He received the B.Sc. degree in Computer Engineering from Jahan Nama Rayanee Higher Education Institute, Tehran, Iran, in 2021, and the M.Sc. degree in Computer Engineering from Shahid Beheshti University, Tehran, Iran, in 2024. He was a member and Research Assistant at the Robotics and Intelligent Autonomous Agents (RoIAA) Laboratory at Shahid Beheshti University, Tehran, Iran. His research interests include deep learning, reinforcement learning, and robotics, with a specialized focus on their applications in autonomous systems.

- Email: m.shahbazikhojasteh@mail.sbu.ac.ir
- ORCID: 0009-0007-6262-9460
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

Armin Salimi-Badr received the B.Sc., M.Sc., and Ph.D. degrees in Computer Engineering from Amirkabir University of Technology, Tehran, Iran, in 2010, 2012, and 2018, respectively, and the Ph.D. degree in Neuroscience from the University of Burgundy, Dijon, France, in 2019, where he was researching on presenting a computational model of brain motor control in the Laboratory 1093 CAPS (Cognition, Action, et Plasticité Sensorimotrice) of the Institut National de la Santé et de la Recherche Médicale (INSERM). He was a Postdoctoral Research Fellow with the Biocomputing Laboratory, Amirkabir University of Technology, from October 2019 to September 2020. Currently, he is an Assistant Professor with the Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, and the Head of the Artificial Intelligence and Robotics and Cognitive Computing Group, Faculty of Computer Science and Engineering, Shahid Beheshti University. He is also a Founder and the Chair of the Robotics and Intelligent Autonomous Agents (RoIAA) Laboratory, Shahid Beheshti University. His research interests include computational intelligence, computational neuroscience, and robotics. He is a Board Member of the Computer Society of IEEE Iran Section and the Chair of the Professional Activities Committee. Dr. Salimi-Badr is a recipient of the IEEE Young Investigator Award from the IEEE Iran Section in 2024.

- Email: a_salimibadr@sbu.ac.ir
- ORCID: 0000-0001-6613-7921
- Web of Science Researcher ID: R-2726-2019
- Scopus Author ID: 56312161900
- Homepage: https://facultymembers.sbu.ac.ir/salimi