**Indonesian Journal of Electrical Engineering and Informatics (IJEEI)**
Vol. 13, No. 3, September 2025, pp. 798~813
ISSN: 2089-3272, DOI: 10.52549/ijeei.v13i3.7067


# Enhancing GRU-Based DRL with Delta-LiDAR for Robust UAV Navigation in Partially Observable Dynamic Environments

**Maryam Allawi Haddad[1], Dhayaa Raissan Khudher [2]**
[1] Department of Computer Engineering, University of Basrah, Basrah ,Iraq
[2] Department of Computer Engineering, University of Basrah, Basrah, Iraq

**Article Info**

***Article history:***

Received Aug 4, 2025
Revised Sep 19, 2025
Accepted Sep 27, 2025

***Keywords:***

UAV autonomous navigation
PPO algorithm
GRU network
Delta-LiDAR
Feature encoder

**ABSTRACT**

Partial observability and sensor limitations are challenging for the navigation of autonomous Unmanned Aerial Vehicles (UAVs). Deep Reinforcement Learning (DRL) algorithms have emerged as potential tools in advancing this field. However, their effectiveness degrades in challenging environments, particularly in the presence of dynamic obstacles. Recent research trends emphasize the need for new DRL variants that guarantee robustness, real-time adaptability, and improved generalization under uncertainty. This paper proposes a lightweight DRL architecture that combines Proximal Policy Optimization (PPO) with a Gated Recurrent Unit (GRU), extended with a temporal LiDAR differencing feature called Delta-LiDAR. The difference between consecutive LiDAR scans is computed to provide the velocity and directional cues without the computational burden of Long Short-Term Memory (LSTM) networks. We evaluate three models, PPO-LSTM, PPO-GRU, and Delta-LiDAR augmented PPO-GRU in a 3D simulated UAV navigation environment characterized by noise, clutter, and dynamic obstacles. We considered several metrics, including success rate, collision frequency, trajectory smoothness, and computational efficiency, to determine the effectiveness of each architecture. The experimental results demonstrate that Delta-LiDAR improves GRU-based temporal reasoning. The deployment complexity is reduced compared with the LSTM-based architecture, which makes it ideal for real-time UAV operation in partially observable environments.

*Copyright © 2025 Institute of Advanced Engineering and Science.*
*All rights reserved.*

***Corresponding Author:***

Maryam Allawi Haddad,
M.Sc. Student, Department of Computer Engineering,
University of Basrah, Iraq,
Email: pgs.maryam.allawi@uobasrah.edu.iq.

## 1. INTRODUCTION

Unmanned aerial vehicles (UAVs) have become indispensable tools in numerous applications, such as surveillance [1], environmental monitoring, flood detection, and disaster relief. In these domains, autonomous flight in an unstructured and dynamic environment is critical [2-4]. However, robust and adaptable navigation in such scenarios remains challenging, particularly under conditions of partial observability [5], sensor noise, and environmental uncertainty [6]. Classic RL algorithms work well in stable and clear environments, but they often struggle and fail in situations with moving obstacles or fluctuating sensor input [7-9].

Early attempts to apply RL algorithms to UAV navigation were value-based methods, such as Deep Q-Networks (DQN) [10]. In this approach, the decision-making process is done by choosing the action with the highest value. While this method is effective in a discrete action space (e.g., simple grid-world navigation), it struggles in a continuous action space required in UAV applications [11]. Policy-based approaches can handle this by directly optimizing the policy using gradient ascent without requiring a value function. In high-

dimensional state, policy-based methods suffer from high variance and unstable training [12]. The actor-critic approaches (e.g., Asynchronous Advantage Actor Critic A3C [13], PPO [14]) have been introduced to stabilize learning by combining both learning a policy (the actor) and a value function (the critic). In traditional RL, the agents merely adhere to predetermined states and actions within a sequential decision-making, i.e., Markov decision process (MDP) [15, 16]. The approach was, however, confronted with serious constraints when used with autonomous systems [17, 18]. RL method faces limitations in high-dimensional observation space directly and fails to handle partial observability effectively [19, 20].

Recently, RL has been integrated with deep learning to mitigate some of these issues by enabling end-to-end learning, which is essential in tasks like autonomous navigation in unknown environments [21-24]. This synergy enhances the ability of an autonomous agent to handle high-dimensional sensory data (e.g., Red-Green-Blue (RGB) images, Light Detection and Ranging (LiDAR) scans, and Inertial Measurement Unit (IMU)) to effectively interpret complex environments [25-27]. The principal role of the deep neural networks is to replace the tabular representation of RL by approximating the policy function, the value function, or other models related to the environment [18].

In dynamic environments, the critical challenge that an agent faces is the ability to infer obstacle motion from sequential observation. Recurrent neural networks (RNNs), such as LSTM, demonstrated strong performances in capturing time-related patterns in sequence data [29, 30]. DRL models with memory augmentation, like PPO-LSTM, have been suggested to counter the problem caused by limited visibility as well as temporal change [31]. These models employ LSTM units for retaining history context to enable better decision-making in time-correlated scenarios. LSTM-based PPO models work better at handling sequences of data and learning from diverse obstacles. Zheng et al. [32] demonstrated that PPO-LSTM achieves strong performance in UAV navigation, but the high computational cost and large parameter count hinder real-time deployment. These constraints render LSTM models less appropriate for dynamic, resource-limited contexts that necessitate rapid and consistent decision-making.

The high computational complexity and large number of parameters make real-time implementation on UAV platforms challenging. The GRU is a lighter option with faster training and less computational cost [33]. The GRU has limited effectiveness in responding to fast, dynamic change due to shorter memory retention [34, 35]. Zhang et al. [34] conducted a comparative study between GRU and LSTM networks within a memory-based DRL framework. The result shows that GRU achieves faster convergence and lower computational overhead. This finding supports the growing preference for GRU in resource-constrained environments. Nonetheless, it requires an additional embedding network to preprocess the observations.

Despite significant advancements in the application of RL and DRL to UAV navigation, certain gaps still exist. Memory-augmented models such as PPO-LSTM considered temporal correlations, but high computational requirements and enormous parameter lists prohibit real-time deployment on resource-constrained UAV platforms. In comparison, lightweight models such as GRU conserve processing burden and converge faster but lack in depicting rapidly changing obstacle dynamics due to shorter memory retention. More importantly, recent methods primarily employ raw sensory observations without explicitly modeling motion cues across time steps. This absence of representation degrades the agent's ability to foresee obstacle motion in dynamic environments. Therefore, there is a clear gap in developing a method that (i) is computationally light for real-time UAV deployment, (ii) can handle temporal dynamics with the presence of partial observability, and (iii) employs motion-related information not embedded explicitly in raw observations.

To address these limitations, we propose a new technique for temporal augmentation called Delta-LiDAR, which captures the temporal difference in LiDAR scans between consecutive time steps. This augmentation provides explicit motion cues, such as obstacle approaching speed and direction, that are not represented explicitly in raw observations. In contrast to prior works, our Delta-LiDAR augmentation merely feeds motion-related temporal information directly into the GRU, simplifying the architecture while improving dynamic awareness.

This study investigates the effectiveness of PPO-GRU with Delta-LiDAR compared to standard PPO-LSTM models in dynamic simulated environments, assessing their autonomous flight capabilities. The experimental results demonstrate that our model has comparatively better performance with faster convergence, lower computational complexity, and improved navigation success rates. Gazebo 11 is used to simulate real-world UAV navigation environments. The world contains dynamic obstacles that move randomly throughout and constantly, posing a challenging navigation scenario.

The main contribution of this work is to propose a Delta-LiDAR that captures the motion dynamics by computing differences between consecutive LiDAR scans. Further, we design a lightweight PPO-GRU framework to improve the temporal reasoning ability without increasing model complexity. Moreover, we compare the proposed architecture against PPO-LSTM in complex environments with dynamic obstacles, partial observability, and sensor noise.

## 2. RESEARCH METHOD

This section presents our proposed GRU-enhanced framework for autonomous UAV navigation under partial observability. The architecture integrates Delta-LiDAR, a multilayer perceptron, GRUs, and a PPO-based policy optimization pipeline. The objectives of this approach are to capture both spatial and temporal information for robust decision-making under uncertainty. The method starts by modeling the task as a Partially Observable Markov Decision Process (POMDP), followed by a detailed explanation of both the observation and action spaces. Next, we explain the reward function design, the PPO-GRU architecture, the training procedure, the neural network configuration, and the computational considerations.

To improve temporal sensitivity, Delta-LiDAR is fused with raw sensor inputs. The GRU is selected due to its lightweight memory efficiency without sacrificing temporal modeling capacity. Further, we nominate PPO to ensure stable learning via a clipped surrogate objective and to stabilize convergence speed.

### 2.1. Problem formulation

The UAV navigation task in dynamic, partially observable environments is formulated as a POMDP and defined by the set:

$$(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \Omega, \gamma)$$

Where:

- $\mathcal{S}$: the state space of the environments,
- $\mathcal{A}$: set of possible actions (e.g., thrust, yaw rate),
- $\mathcal{O}$: set of observation space (from onboard sensors),
- $\mathcal{T}(s_{t+1}|s_t, a_t)$: transition dynamics,
- $\mathcal{R}(s_t, a_t)$: reward function,
- $\Omega(o_t|s_t)$: observation probability function,
- $\gamma \in (0, 1)$: discount factor.

At each time step *t,* the UAV receives sensory input signals from a 2D 360° LiDAR scan. $\text{LiDAR}_t$, the temporal difference $\Delta\text{LiDAR}_t = \text{LiDAR}_t - \text{LiDAR}_{t-1}$, velocity vector $\mathcal{V}_t = [v_x, v_y, v_z]$, and attitude $\theta_t = [roll, pitch, yaw]$. These input signals are first normalized and then concatenated into a unified observation vector. Equation 1 defines the resulting vector:

$$\mathcal{O}_t = [\text{LiDAR}_t, \Delta\text{LiDAR}_t, \mathcal{V}_t, \theta_t] \quad (1)$$

In the feature encoder stage, the $\mathcal{O}_t$ It is passed through a Multilayer perceptron (MLP) to project it into a latent feature space. The MLP consists of two dense layers with a nonlinear activation function (ReLU) applied in between them. The first layer transforms the normalized observation vector into an intermediate representation to enable the network to learn abstract features. This activated feature vector is denoted in Equation 2. The second layer compresses this feature into a compact latent representation as defined in Equation 3.

$$h_1 = W_1\mathcal{O}_t + b_1 \quad (2)$$

$$h_2 = W_2 ReLU(h_1) + b_2 \quad (3)$$

Where $W_1, W_2$ and $\text{b}_1, \text{b}_2$ These are learnable weight parameters. $\text{h}_2 \in \mathbb{R}^{\text{d}}$ It is the latent embedding of the observation. This compact feature vector $h_2$ captures the current sensory state and is forwarded to the recurrent network for temporal modeling (see subsection 2.5). Figure 1 illustrates the overall system pipeline.

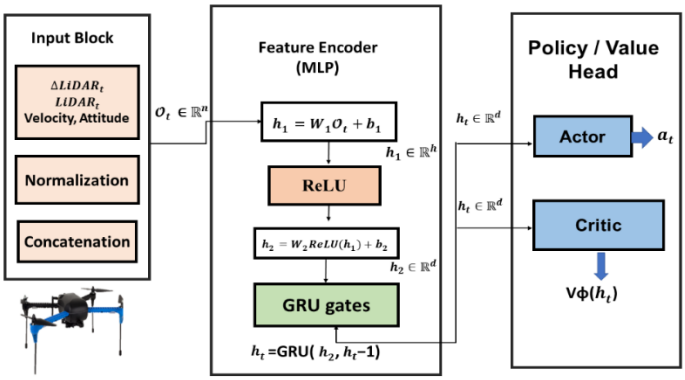


Figure 1. Architecture of the proposed PPO-GRU framework with Delta-LiDAR integration. The system processes raw sensor inputs, encodes features via MLP, and models temporal dependencies through GRU.

### 2.2. Observation and Action Space

The observation space consists of a 360° LiDAR scan with 360 beams, the drone's current velocity vector [$vx$, $vy$, $vz$], and the drone's current position and acceleration vector [p$x$, p$y$, p$z$, a$x$, a$y$, a$z$]. These inputs are combined and normalized before being fed to the PPO network. This sensor configuration enables robust perception and motion sensing without maintaining a large state representation. Depending on obstacles or sensor range, the environment may be observed either fully or partially. The action space is continuous and consists of five parameters: [$vx$, $vy$, $vz$, $\omega y$, $\omega z$], representing three linear velocity commands and two angular velocity commands. The velocity commands are directly converted to MAVROS velocity setpoints and sent to the PX4 flight controller in OFFBOARD mode. The connection between MAVROS and PX4-SITL) PX4 Software-In-The-Loop) an autopilot is shown in Figure 2.

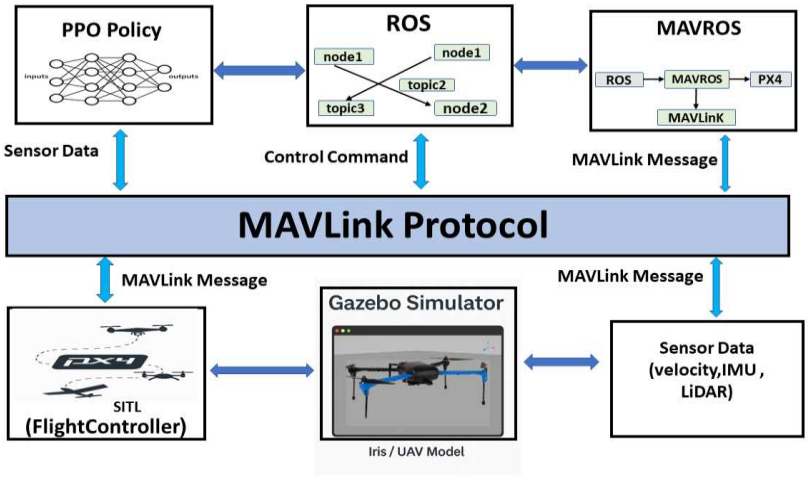


Figure 2. Communication pipeline between ROS, MAVROS, and PX4-SITL. Sensor data and control commands are swapped between PPO and PX4 flight controller by MAVROS in OFFBOARD mode.

### 2.3. Reward Function Design

The reward function is designed for enhancing the agent's goal-reaching through safe behaviors, collision avoidance, and choosing shorter paths at a suitable speed. It incorporates a combination of sparse and dense rewards; each targeting a specific navigation objective. To align these overarching objectives of the task, penalties apply in the event of collisions or unproductive motions. The reward function consists of four components.

The first type is the goal-reaching reward, which provides a substantial positive numerical value when the agent successfully attains the target. Equation 4 represents this type of reward.

$$R_{goal} = \begin{cases} +1500 & if\ ||p_t - p_{goal}|| \leq 0.5m \\ 0 & otherwise \end{cases} \quad (4)$$

Where $p_t - p_{goal}$ Is the Euclidean distance from the drone position to the goal at time t.

The second type is the collision penalty. $R_{collision}$ Defined in Equation 5, which imposes a significantly adverse reward on the agent when a collision with an obstacle occurs. This will motivate the agent to learn safe trajectories and avoid dangerous behaviors.

$$R_{\text{collision}} = \begin{cases} -200 & \text{If a collision is detected at time } t \\ 0 & otherwise \end{cases} \quad (5)$$

The third type is the stability penalty, which aims to reduce abrupt drone motion and promote smooth trajectory tracking. It incentivizes the agent based on roll and pitch errors at each step, as represented in Equation 6. This is important for keeping the flight safe, especially when flying through dense or constricted areas where abrupt angular turns would result in instability or crashes.

$$R_{\text{stability}} = -(|roll_t| + |pitch_t|) \quad (6)$$

To promote stability of control and discourage abrupt changes in motion, the agent is penalized for velocities changing quickly in recent timesteps. Instead of operating on instantaneous velocity differences, the

penalty is computed as the average difference in linear velocity within a short time horizon, as illustrated in Equation 7.

$$R_{\text{smooth}} = \frac{1}{K}\sum_{k=1}^{K}|v_{t-k+1} - v_{t-k}|^2 \quad (7)$$

Where $K$ is the smoothing window size and $v_t$ Is the velocity vector at time t? The total reward is shown in Equation 8.

$$R_{total} = R_{goal} + R_{\text{collision}} + R_{\text{stability}} + R_{smooth} \quad (8)$$

This reward shaping method combines sparse rewards (arrival at goals and collision) with dense feedback (stability and smoothness) to encourage the agent to achieve robust, safe, and efficient navigation. This reward design is effective when integrated with GRU-based policies that retain temporal context from past observations and actions.

### 2.4. Delta-LiDAR fusion mechanism

The 2D spatial snapshot of the environment at time $t$ alone cannot detect a change in dynamic obstacles. Therefore, we compute the differential representation over time steps as illustrated in Equation 9.

$$\Delta\text{LiDAR}_t = \text{LiDAR}_t - \text{LiDAR}_{t-1} \quad (9)$$

The agent uses the resulting vector associated with other sensory data to infer the object motion. The increasing value of $\text{Delta} - \text{LiDAR}_t$ Indicates that the UAV is approaching an obstacle or the obstacle itself is moving towards the UAV (case of a dynamic obstacle). As a result, the GRU model can easily distinguish between static and dynamic obstacles.

Figure 3 illustrates the mechanism for computing $\text{Delta} - \text{LiDAR}_t$. This diagram illustrates the computation and role of $\text{Delta} - \text{LiDAR}_t$ In enhancing temporal awareness for UAV navigation. The left-side section shows two consecutive 2D LiDAR scans: $\text{LiDAR}_{t-1}$ (previous) and $\text{LiDAR}_t$ (current). These signals are captured as the drone observes its environment over time. The gray bar chart represents the older scan, and the blue bar outlines the current scan.

The notation $\Delta L_t \approx \frac{\partial L}{\partial t}$ Treats this as a temporal derivative to indicate a signal for motion or approaching obstacles. In the right-side section, the drone processes both $\text{LiDAR}_{t-1}$ and $\text{Delta} - \text{LiDAR}_t$ To get a richer temporal understanding of its environment. This fusion allows the drone to detect moving obstacles, approaching threats, and dynamic structure changes. Before differencing, we register $\text{LiDAR}_{t-1}$ to the ego frame at t through onboard odometry (SE (2) translation and rotation).

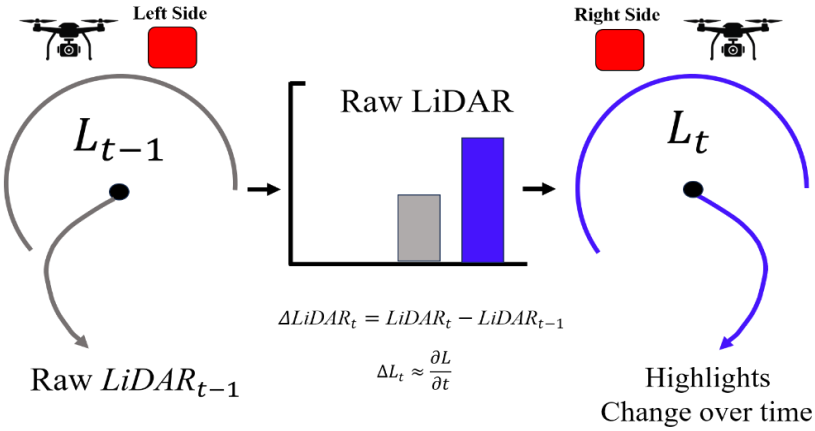


Figure 3. Mechanism for computing Delta-LiDAR by subtracting consecutive 2D LiDAR scans. The resulting temporal perception enables the GRU to distinguish between static and moving entities.

The LiDAR sensor used in the study has a range of 0.06 meters to 5 meters and allows accurate calculation of distance within this range. The sensor has a horizontal angular resolution of 1 degree with full 360-degree coverage, allowing it to cover the whole environment surrounding it. To get accurate readings, the sensor also has Gaussian noise with a mean of 0 and a standard deviation of 0.01.

The sensor also operates at a frequency of 20 Hz, so it refreshes its measurements 20 times per second. These requirements enable the LiDAR to effectively detect and measure obstacles within its range without a high level of error due to noise.

### 2.5. GRU-Based Temporal Feature Modeling

The GRU allows the agent to build a belief state from a sequential input to maintain a hidden state $h_t$. This hidden state captures spatial features and temporal context necessary for partial observability. The GRU can encode both recent and past information by updating its hidden state vector using the previous hidden state $h_{t-1}$ and two internal gates. The reset gate $r_t$ (indicate how much of the previous memory to forget). The update gate $u_t$ (determines how much of the new candidate state should be combined with the previous state), see Equations 10 and 11:

$$r_t = \sigma(W_r h_2 + U_r h_{t-1} + b_r) \quad (10)$$
$$u_t = \sigma(W_u h_2 + U_u h_{t-1} + b_u) \quad (11)$$

Where $\sigma(\cdot)$ is the sigmoid function, and $W_*, U_*, b_*$ Are learnable GRU parameters.

The new input and the reset-modulated previous state are combined in a temporary state candidate as represented in Equation 12:

$$\tilde{h}_t = tanh\ (W_h h_2 + U_h(r_t \odot h_{t-1}) + b_h) \quad (12)$$

Where $tanh$ is the hyperbolic tangent and $\odot$ is the element-wise multiplication. The final hidden state $h_t$ It is updated as a convex combination of the previous and candidate hidden states, as in Equation 13:

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t \quad (13)$$

This mechanism enables the GRU to remember relevant temporal features (e.g., obstacle movement) and reject irrelevant information such as noise. As a result, the hidden state $h_t$ Becomes a temporal-aware embedding that integrates both environmental dynamics and internal UAV states. Instead of retaining every single raw sensor data, the GRU keeps a smart summary that highlights key events. This summary is efficient (lower-dimensional than raw sensor data) and useful for decision-making under partial observability.

#### 2.5.1 Integration of GRU for Detection of Obstacles and Temporal Context

The Integration of GRU has been included in the model to learn temporal relationships in sequence data. An essential component of autonomous UAV navigation operates in dynamic environments where future actions are reliant upon past actions and states. The GRU allows the model to remember the previous states, i.e., velocity changes, orientation, and sensor readings, and aggregate this information to make more knowledgeable decisions over time. This is especially useful in partially observable environments, where the system does not always observe the whole context. The addition of GRU allows the model to learn temporal dependencies in the data well without having to perform much computation, in contrast to other types of recurrent networks like LSTM, making it possible for real-time applications with UAVs.

The system reacts to orientation changes well by adding orientation data (roll, pitch, yaw) onto the input observations directly to the model. Since the UAV is subjected to continuous orientation change while in flight, these are values that are needed to understand the motion and location of the UAV in 3D space. The GRU considers this data, as well as other sensing inputs (e.g., velocity and LiDAR), such that it can calculate how the UAV's navigation decisions and course are affected by changes in orientation. Integration of GRU for Detection of Obstacles and Temporal Context. This allows the model to realign the navigation strategy in real-time whenever orientation is modified, leading to smooth control during tilting or rotation. Therefore, the GRU gives the system the capacity for smooth handling of orientation changes in real-time.

The model can also allow for variations in external forces like wind drift and mobile barriers. The Delta-LiDAR (providing temporal differences of LiDAR data) allows the model to keep track of variations in the environment around the UAV, e.g., wind drift changes or mobile obstacles. The GRU updates this sequential data so that the model can learn to adjust to such variations and make appropriate adjustments to its decision-making for navigation. For instance, if the wind causes the UAV to drift, the model can detect such deviation from the LiDAR measurements and correct its direction. The GRU also tracks dynamic obstacles and adjusts decision-making in response. However, the responsiveness of the system to rapid environmental change is limited by data sampling rate and processing rate, and thus, very quickly moving obstacles or rapidly changing environmental conditions might be out of the model's scope if the data from sensors is not rapidly processed enough.

### 2.6. PPO Training Mechanism

We employ the PPO algorithm to optimize the memory-augmented policy under partial observability. The PPO algorithm restricts the policy update to stay within a clipped range, leading to improved training stability. At each training iteration, PPO maximizes the clipped objective function, defined in Equation 14:

$$\mathcal{L}^{\text{PPO}}(\theta) = E_t[\min(\rho_t(\theta)\widehat{A_t}, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t})] \quad (14)$$

Where $\widehat{A_t}$ Is the advantage estimate at time t, $\rho_t(\theta)$ The probability ratio between the new and old policy, and $\epsilon$ is the clipped threshold (set to 0.2). The advantage function is computed using Generalized Advantage Estimation (GAE), as in Equation 15:

$$\widehat{A_t} = \sum_{l=0}^{\text{T}} (\gamma\lambda)^l \delta_{t+l} \quad (15)$$

Where $\gamma \in (0, 1]$ is the discount factor, $\lambda \in [0,1]$ Are the controls the bias–variance trade-off? $\delta_t$ is the temporal difference error for value estimation and can be computed from the immediate reward and value estimate of the current and next hidden state, as $\delta_t = r_t + \gamma V(h_{t+1}) - V(h_t)$.

The final training objective consists of three components: policy loss, value loss, and entropy bonus, as shown in Equation 16:

$$\mathcal{L}_{total} = \mathcal{L}^{PPO} - c_1.\mathcal{L}_{\text{value}} + c_2.\mathcal{L}_{\text{entropy}} \quad (16)$$

With

- $\mathcal{L}_{value} = \frac{1}{2}(V(h_t) - V_{target})^2$
- $\mathcal{L}_{entropy} = -\mathbb{E}[\pi_\theta(a_t|h_t) log \pi_\theta(a_t|h_t)]$
- $c_1, c_2$ are scalar coefficients for balancing value and entropy terms.

### 2.7. Training Setup and Hyperparameters

In this subsection, we define the training configuration, environment parameters, and hyperparameters used in our PPO-based memory-augmented policy. The first module of the neural network is the feature encoder (MLP), which has two hidden layers. The first hidden layer is 256-dimensional, and the second one is 128-dimensional, both with ReLU activation. They are trained to learn and extract high-level features of the input data required for decision-making in reinforcement learning, as one may notice in Table 1. The GRU module is a single layer of 128 hidden units that is used to preserve temporal relationships and sequential data. The GAE (Generalized Advantage Estimation) parameters $\lambda = 0.95$ are utilized to estimate the value of each action taken, balancing bias and variation in policy updates for convergent learning. The Adam optimizer, with a specified learning rate of 3e-4, is employed to update the policy, utilizing a batch size of 64. Entropy regularization using an entropy coefficient of $c_2$=0.01 is also employed to prevent premature convergence, motivating exploration, and facilitating generalization of the acquired policy. Table 1 shows the training hyperparameters used for the PPO algorithm. For ensuring an optimal balance between policy stability, learning efficiency, and generalization under dynamic conditions, all parameters were empirically tuned by iterative testing.

To ensure a robust balance between policy stability, learning efficiency, and generalization under dynamic conditions, we empirically adjusted all parameters through iterative testing.

Table 1. The training hyperparameter of the PPO algorithm.

| Parameter | Value |
|---|---|
| Network Type | GRU-based Actor-Critic |
| MLP Hidden Layers | [256, 128] |
| GRU Hidden Size | 128 |
| Activation Function | ReLU |
| Discount factor $\gamma$ | 0.99 |
| GAE parameter $\lambda$ | 0.95 |
| Clipping threshold $\epsilon$ | 0.2 |
| Learning rate | 3e-4 |
| Optimizer | Adam |
| Entropy coefficient $c_2$ | 0.01 |
| Value loss coefficient $c_1$ | 0.5 |
| Batch size | 64 |
| Epochs per update | 4 |
| Max training steps | 1M – 3M |

The training was performed on a computer equipped with a GPU (NVIDIA RTX 3090), an Intel i7 CPU, and 32 GB RAM. The software environment included Ubuntu 20.04, Python 3.8, ROS Noetic, Gazebo 11, and PyTorch 1.13. Algorithm 1 summarizes the training procedures of our proposed PPO-GRU agent with Delta-LiDAR.

**Algorithm 1:** PPO-GRU with Delta-LiDAR training pipeline

**Require:** Policy network. $\pi_\theta$, value function $V_\phi$
**Require:** Discount factor $\gamma$, GAE parameter $\lambda$, clipped threshold $\epsilon$
**Require:** Coefficients $c_1$ (value loss), $c_2$ (entropy bonus)
1: **for** each t=1 to T **do**
2: $\delta_t \leftarrow r_t + \gamma V(h_{t+1}) - V(h_t)$
3: **end for**
4: **for** each t=1 to T **do**
5: $\widehat{A_t} \leftarrow \sum_{l=0}^{\mathrm{T}} (\gamma\lambda)^l \delta_{t+l}$
6: **end for**
7: **for** each t=1 to T **do**
8: $r_t(\theta) \leftarrow \frac{\pi_\theta(a_t|h_t)}{\pi_{\theta_{\text{old}}}(a_t|h_t)}$
9: $\mathcal{L}_{PPO,\,t} \leftarrow \min\left(r_t(\theta)\widehat{A_t},\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}\right)$
10: $\mathcal{L}_{value,\,t} \leftarrow \frac{1}{2}\left(V_\phi(o_t) - V_{target,t}\right)^2$
11: $\mathcal{L}_{entropy,\,t} \leftarrow -\mathcal{H}\left(\pi_\theta(\cdot|\, h_t)\right)$
12: **end for**
13: $\mathcal{L}_{total} \leftarrow \frac{1}{T}\sum_{t=1}^{T}\left[\mathcal{L}^{PPO} - c_1 \cdot \mathcal{L}_{value,\,t} + c_2 \cdot \mathcal{L}_{entropy,\,t}\right]$
14: Update parameters $\theta$ and $\phi$ using gradient descent on $\mathcal{L}_{total}$
15: **return** updated $\pi_\theta$ and $V_\phi$

## 3. RESULTS AND DISCUSSION

To evaluate the performance of the proposed memory augmented learning for UAV navigation, we conducted three key experiments. The first experiment established a baseline using the PPO-LSTM structure. The second experiment replaced the LSTM with a GRU to see whether it would learn temporal dependencies with fewer parameters and less training time. The third test utilized the developed PPO-GRU with the Delta-LiDAR model, which adds GRU-based memory by providing the difference between consecutive LiDAR scans, Delta-LiDAR, to the raw LiDAR input. This study evaluates whether combining GRU-based temporal memory with the detection of active environment change further improves the decision-making ability of the UAV in partially observable or dynamic environments. The same quantitative measures were used, including total reward, collisions, loss in value, policy entropy, trajectory length, and trajectory smoothness. Training time and model parameter count were also compared across the three settings to highlight the computationally efficient character of the GRU and the additional reward of Delta-LiDAR. The smoothness and the length of the path were especially noted to demonstrate how the Delta-LiDAR input enables the UAV to follow shorter and smoother paths, which translates to more stable and intelligent control policies.

The simulation environment is built using Gazebo 11, integrated with PX4-SITL Autopilot (for low-level flight control), and MAVROS (for ROS communication). Figure 4 illustrates the simulated world from both a top view and a front view. This subsection focuses on the simulation environment, UAV configuration, and system-level integration. Figure 5 illustrates the training environments, including a corridor-like world designed to simulate indoor or constrained drone navigation. The corridor is modeled as a rectangular space with dimensions of 30 m (length) × 6 m (width) × 6 m (height). The left and right walls bounded the width, and we left the ceiling open to avoid z-axis constraints. The environment consists of moving circular objects and dynamically walking human agents, which are distributed randomly. The dynamic obstacles were used to challenge the UAV's perception and collision avoidance in the case of partial observability. These obstacles maintain a minimum clearance of 0.4 m on both sides and are positioned at various locations along the y-axis to assess lateral movement.

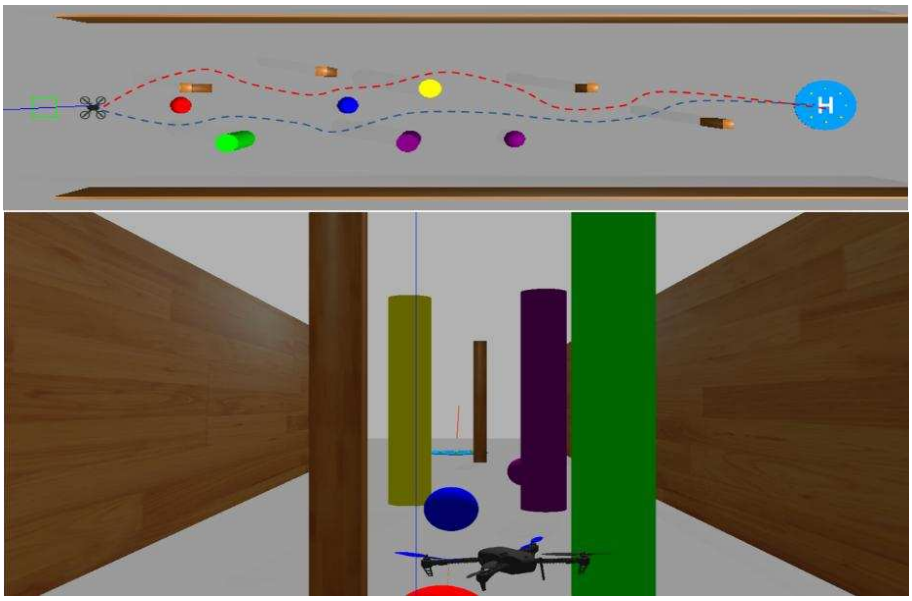

Figure 4. Top and front views of the designed UAV simulation environment in Gazebo. The figure depicts a corridor structure with various static and dynamic obstacles of different shapes and sizes.

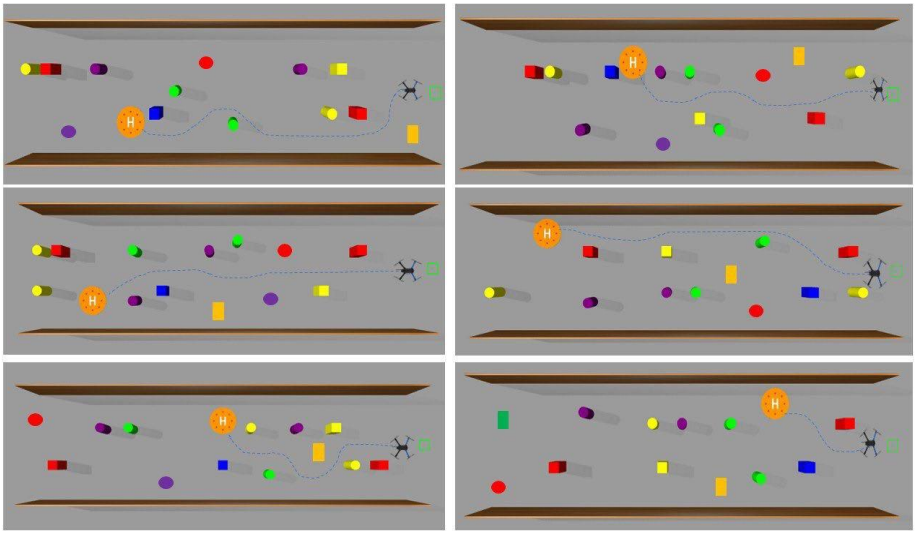

Figure 5. Training environments with six different obstacle layouts in a 3D corridor of (30 m × 6 m × 6 m). These configurations are used to assess the UAV's adaptability and collision avoidance capability.

### 3.1. Total Reward Per Episode

Policy learning performance is usually quantified in terms of cumulative reward across a single training episode. Learning curves of PPO-GRU, PPO-GRU with Delta-LiDAR, and baseline PPO-LSTM across 30,000 training episodes are given in Figure 6. The episode number is shown on the x-axis, and the total reward is represented on the y-axis. PPO-GRU with Delta-LiDAR also shows a smooth, consistent rise in rewards with convergence at around 129,000 at episode 25,000, outperforming the other two methods. The data shows that GRU-based temporal modeling and Delta-encoded LiDAR data make it easier to interpret and make decisions. The PPO-GRU baseline, on the other hand, goes up and down a lot during the initial training and ends with a lower reward level of roughly 122,000. Such behaviors illustrate the challenging problem of policy convergence in partial observability when more informative temporal cues are not available. The PPO-LSTM model also learns quickly to a reward threshold around 120,000 but possesses a flattened learning curve and thus shows early saturation. While LSTM is demonstrated to be initially powerful, it may be less effective at the steeper temporal sensitivity needed for long-run reward enhancement in dynamic worlds.

Overall, the results clearly show that combining Delta-LiDAR with GRU helps the agent understand time better and makes the process more stable, leading to better performance.

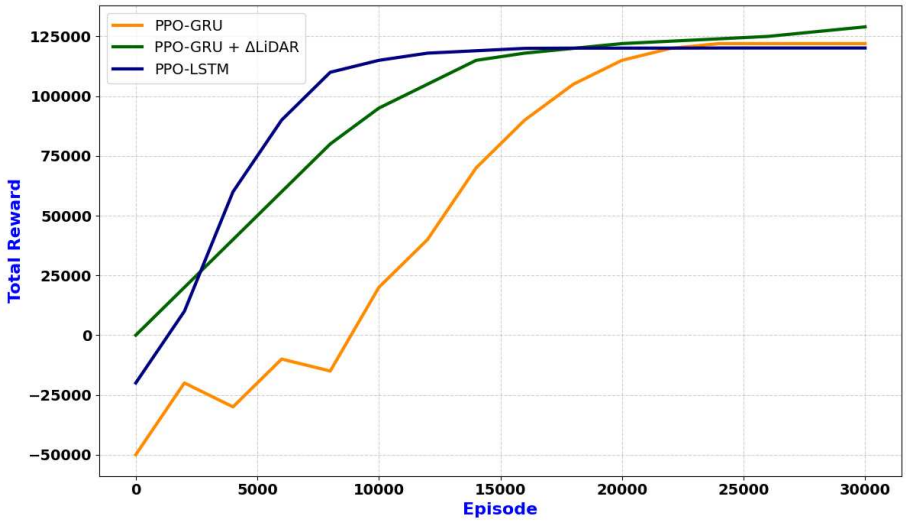


Figure 6. Cumulative training reward per episode for PPO-LSTM, PPO-GRU, and PPO-GRU with Delta-LiDAR. The Delta-LiDAR variant shows faster convergence and higher final reward.

### 3.2. Number of Collisions per Episode

Figure 7 illustrates the average collision rates per episode of the three architectures, the x-axis represents episode Number, and the y-axis represents the average number of collisions. The PPO-GRU with Delta-LiDAR exhibits a smooth and consistent reduction in collision rate and converges to approximately 2 collisions per episode. The baseline PPO-GRU has a high variance and cannot drop below 10 collisions per episode, reflecting poor stability and less adaptability. The PPO-LSTM converges sooner than GRU-based policies, but at approximately 7 collisions. These results support the hypothesis that Delta-LiDAR enhances temporal-spatial awareness that leads to safer and more stable navigation performance.

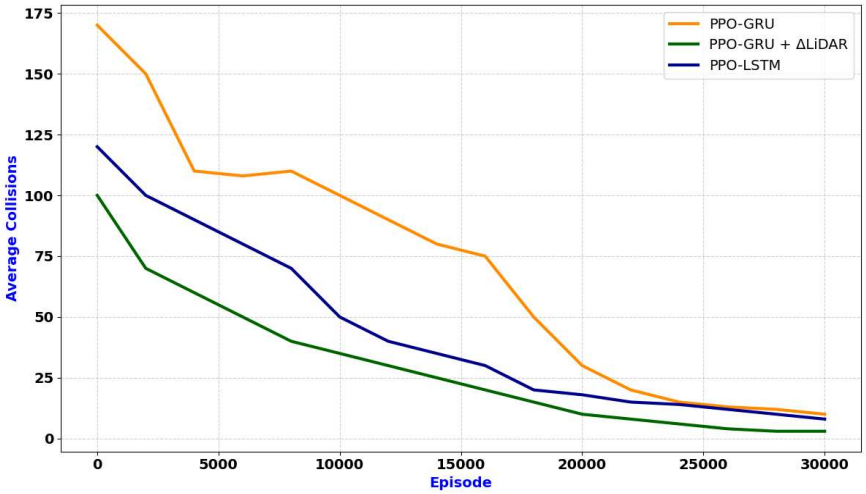


Figure 7. Average number of collisions per episode, PPO-GRU vs PPO-GRU with Delta-LiDAR. The proposed framework achieves a lower collision rate.

### 3.3. Trajectory Length over Time

Figure 8 illustrates that employing PPO-GRU with Delta-LiDAR data enhances trajectory efficiency, where the x-axis represents episode number and the y-axis represents average trajectory length (m). The proposed method exhibited the shortest average trajectory length when compared to standard PPO-GRU and PPO-LSTM. The utilization of time-aware delta LiDAR data significantly facilitates navigation in areas with limited visibility.

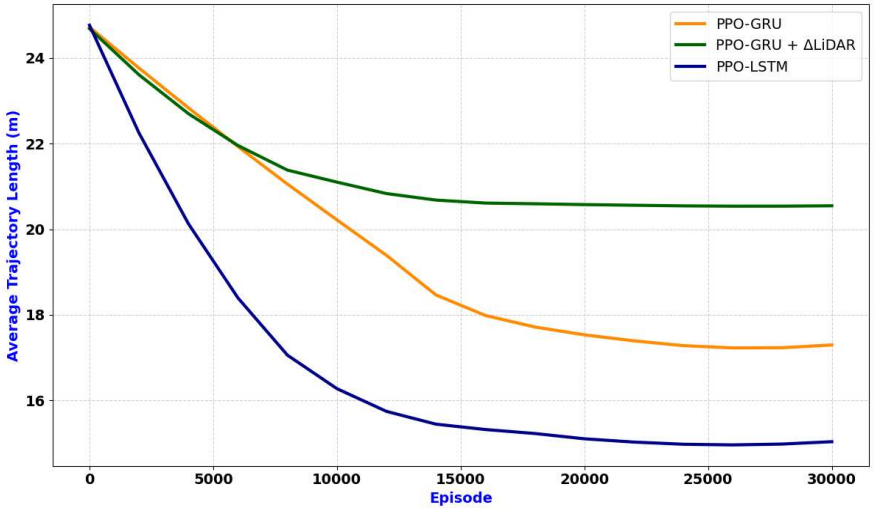


Figure 8. Average trajectory length (m) per episode for PPO-LSTM, PPO-GRU, and PPO-GRU with Delta-LiDAR. The Delta-LiDAR model follows shorter paths.

### 3.4. Policy Entropy During Training

This type of measure is being able to estimate the expected returns, and this is especially crucial for stable policy improvements. In Figure 9, the x-axis represents episode number, and the y-axis represents average policy entropy. One can see that the PPO-GRU critic updates are extremely unstable. The orange line is defined by massive spikes and massive drops into negative areas, indicating that the value estimation process is extremely unstable and difficult to optimize. Whereas some stability is achieved around the 200k timestep mark, it is so after an extremely long period of instability, subtracting from the overall consistency of learning. In contrast, the PPO-GRU with Delta-LiDAR (green line) reflects a smooth and always decreasing policy entropy, indicating a much more stable and better-organized learning process. A smooth trajectory indicates that the critic learns valuable information better, which leads to improved reward propagation and improved value function approximation. These results clearly demonstrate that incorporating more Delta-LiDAR enhances further training stability and higher critic reliability. Thus, the policy improves progressively with each step, leading to better overall training performance and deployment.

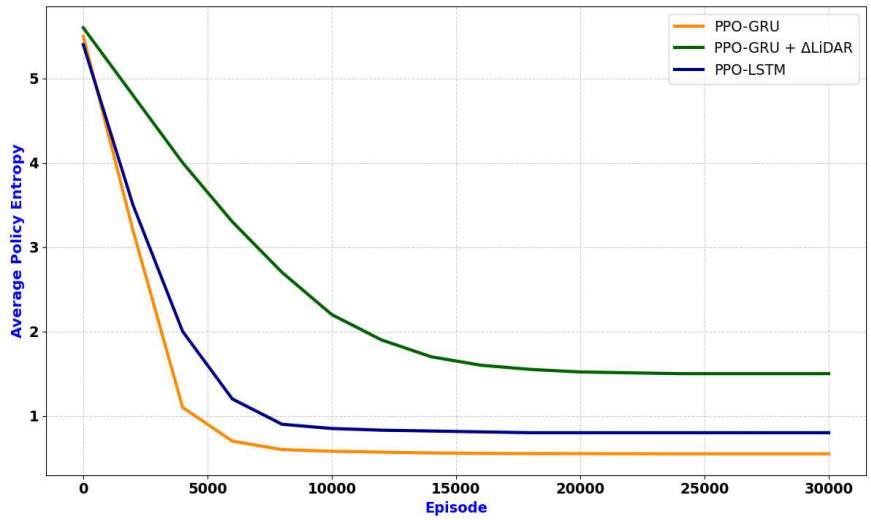


Figure 9. Average policy entropy during training. PPO-GRU with Delta-LiDAR exhibits smoother entropy decay.

### 3.5. Value Loss over Time

This metric measures how effectively the agent explores the action space to identify better actions. Entropy measures higher corresponding to more extensive exploration and stronger policy learning with reduced behaviors of premature convergence to sub-optimal solutions. Figure 10 compares the value loss of

the PPO-GRU, PPO-LSTM, and PPO-GRU- Delta-LiDAR models; the x-axis represents episode Number, and the y-axis represents average value loss. All begin high (~5.5), which is the initial exploration phase. However, PPO-GRU (orange) falls sharply and prematurely to ~0.25, which indicates rapid exploitation and potential convergence at sub-optimal policies. PPO-LSTM (blue) experiences a gradual fall, stabilizing at ~0.8. In contrast, PPO-GRU-Delta-LiDAR (green) undergoes a gradual fall, stabilizing at ~1.5. Such a smoother transition signals enhanced temporal and spatial perception, allowing for better exploration and more stable learning. Overall, the Delta-LiDAR-enhanced model weighs exploration and exploitation more effectively.

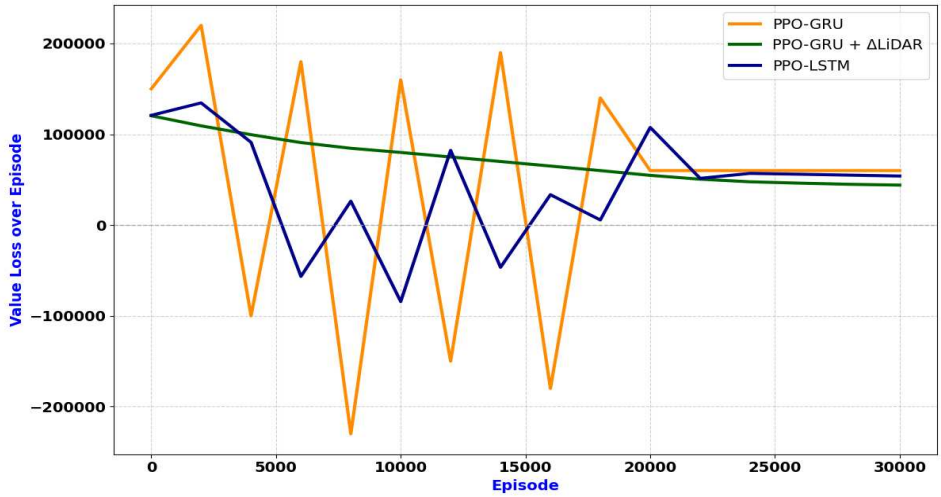


Figure 10. Average Value loss progression per episode for PPO-GRU vs PPO-GRU with Delta-LiDAR.

### 3.6. Trajectory Smoothness over Time

Figure 11 shows how smoothly the agents of the three models, PPO-GRU, PPO-LSTM, and PPO-GRU Delta- LiDAR, move in comparison to each other. The x-axis represents episode Number, and the y-axis represents average trajectory smoothness. Lower numbers mean that the movement is smooth and steady. The suggested PPO-GRU with Delta-LiDAR gives the best final value (~8.58), with a faster and more stable improvement than PPO-GRU (~15.78) and PPO-LSTM (~23.47). This data shows that adding Delta-LiDAR makes motion stability and control performance during training much better.

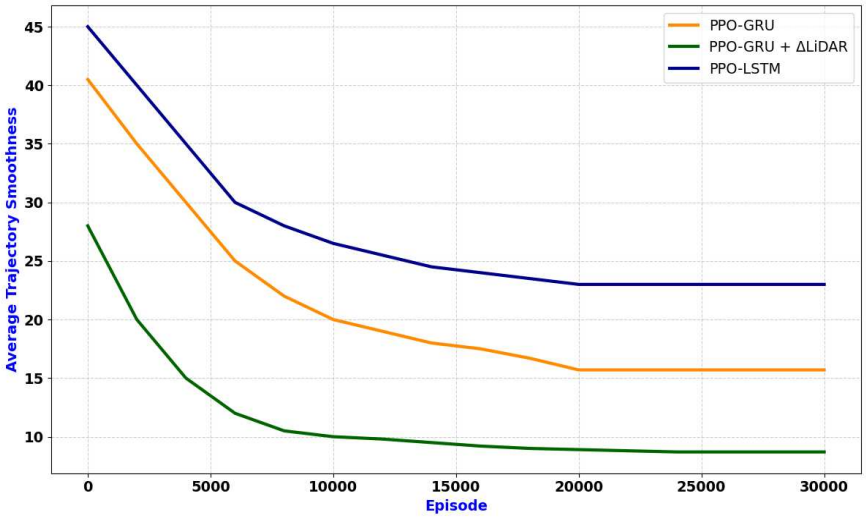


Figure 11. Average Trajectory Smoothness metric per episode. PPO-GRU with the Delta-LiDAR model yields a smooth and more stable flight path compared to the baseline model and PPO-GRU.

In the study, angular motion and curvature change along the trajectory. Smoothness is how gradual in-direction changes are mitigated, and is expressed mathematically as follows:

Curvature is a measure of how rapidly the direction of a motion is changing. The curve of a 2D path can be calculated in Equation (17):

$$\kappa(t) = \frac{|x'(t)y''(t) - y'(t)x''(t)|}{(x'(t)^2 + y'(t)^2)^{3/2}} \quad (17)$$

$\boldsymbol{x(t), y(t)}$These are the trajectory coordinates.
$\boldsymbol{x'(t), y'(t)}$: The velocity along the x and y directions.
$\boldsymbol{x''(t), y''(t)}$: The acceleration along the x and y directions.

Smoothness thresholds: The movement is smooth if the curvature κ(t) is less than some threshold. In our experiments, a smooth motion is defined by having a curvature less than 0.5, which ensures that the motion has gradual direction changes without sudden turns. Motions with a higher curvature of more than 0.5 are less smooth.

The upper bound for angular velocity for a smooth trajectory is computed at 10 degrees per second to allow the UAV ample time to follow a smooth path without abruptly changing its direction of heading. These metrics provide an exact characterization of the path's smoothness and are employed to assess the model's performance in motion stability during training.

### 3.7. Generalization to Noisy Tunnel Environment

The robustness of the proposed framework is evaluated by introducing an unseen environment with moving obstacles to test the PPO-GRU with the Delta-LiDAR agent. We trained agents in simulated corridor-like environments with dynamic obstacles and clean LiDAR input conditions that are optimal for simulation and controlled learning. To properly test the performance of the agent outside the unseen world, we transferred the learned models to a novel, highly challenging tunnel environment that has a very different structure and sensing conditions.

The tunnel environment in Figure 12 imposed additional strict spatial constraints and sensor corruption by adding Gaussian noise to each LiDAR range measurement. The PPO-GRU with the Delta-LiDAR agent was still able to fly safely through the tunnel despite these difficulties and showed strong robustness in both spatial and sensory perturbations. It was successful 92.7% of the time whereas the baseline PPO-GRU agent was only 82.5% successful when subjected to the same conditions.

These results validate the stability and generalization of the proposed method. The Delta-LiDAR subcomponent enhances temporal understanding in the recurrent policy, distinguishing sensor noise from critical environmental signals. Conversely, the PPO-GRU agent that lacked this temporal delta information struggled to differentiate between noise and meaningful cues, resulting in more frequent navigation errors and a higher number of missed targets.

This experiment shows the benefits of using memory-based structures GRU along with Delta-LiDAR's time information, particularly when transitioning from simulated hallway training to real-world tunnel deployments with higher uncertainty.

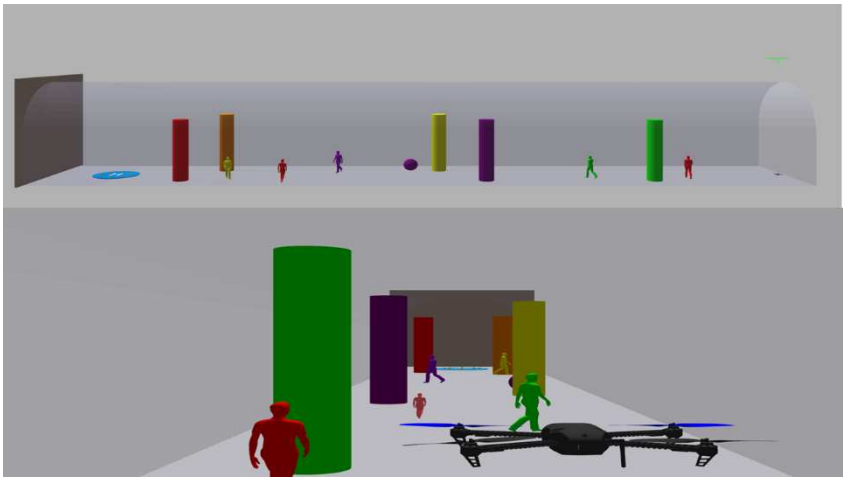

Figure 12. Visualization of an unseen noisy tunnel environment used for generalization testing. The scenario includes dynamic obstacles and LiDAR noise to evaluate the robustness of the trained policy.

Figure 13 illustrates the drone's navigation trajectory from start to endpoint, demonstrating a smooth and consistent flight path. As it is clear from the figure, there are no erratic deviations or sharp corrections,

which suggests robust integration of temporal LiDAR data with the memory-based policy, enabling anticipatory maneuvering rather than reactive path adjustments.

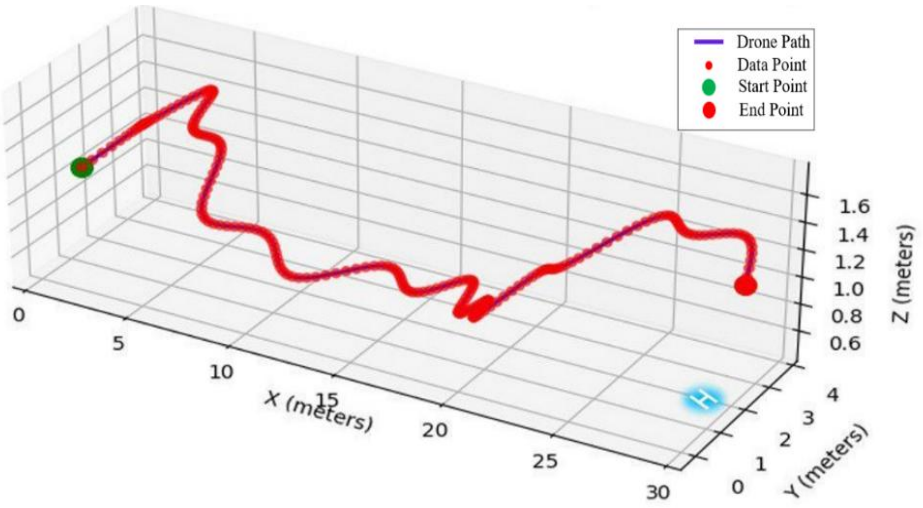


Figure 13. Sample flight trajectory of the PPO-GRU with the Delta-LiDAR agent in the tested environment. The smooth and direct path highlights stable control and anticipatory decision-making under temporal uncertainty.

The detailed comparison between PPO-LSTM, PPO-GRU, and PPO-GRU with Delta-LiDAR is summarized in Table 2. The complete experimental results, including all figures, tables, and demonstration videos, are available in the project's GitHub repository: (https://github.com/Maryamallawi96/PPO_GRU_Delta_LiDAR/tree/main/Media2).

Table 2. Comparative Analysis of PPO Variants' Performance (mean ± std over 50 test episodes).

| Metric | PPO-LSTM | PPO-GRU | PPO-GRU-Delta-LiDAR |
|---|---|---|---|
| Success rate % | 91.7 ±0.8 | 82.5±1.1 | 92.7±0.7 |
| Avg. distance goal (m) | 19.89±0.6 | 18.43±0.5 | 6.27±0.4 |
| Avg. Collisions | 7.03±0.4 | 10.0±0.8 | 2.0±0.3 |
| Trajectory Length (m) | 20.54±1.0 | 17.24±0.7 | 12.9±0.6 |
| Smoothness | 23.46±1.2 | 15.78±0.9 | 8.58±0.6 |
| Training time (minutes) | 126±2 | 63±1.5 | 92±1.8 |
| Model parameter | 710K(fixed) | 589K(fixed) | 580K(fixed) |

## 4. CONCLUSION

This study presented a novel approach to enhance UAV navigation in partially observable and dynamic environments. This method includes integrating the Delta-LiDAR feature with a GRU-based PPO framework. The temporal difference between consecutive LiDAR scans is computed to provide motion cues about the environment changes. This difference enhances the agent's temporal awareness of obstacles' motion without increasing the computational complexity.

The experimental results demonstrate that the proposed PPO-GRU with Delta-LiDAR outperformed both standard PPO-GRU and PPO-LSTM in terms of various performance metrics. It achieved the highest success rate of 92.7%, surpassing both PPO-LSTM (91.7%) and PPO-GRU (82.5%). The proposed method reduces the trajectory length to approximately 12.9 meters by episode 15,000, which is less than PPO-GRU (17.24 meters at episode 25,000) and PPO-LSTM (20.54 meters at episode 20,000). These results reflect that the proposed model achieves faster convergence, learning efficiency, and safer navigation, all while maintaining computational effectiveness.

The generalization capability of the model was further validated using an unseen, noisy tunnel environment. The Delta-LiDAR enhanced the agent, maintaining robust performance despite all spatial constraints. Overall, the integration of Delta-LiDAR with PPO-GRU provides a practical, lightweight, and high-performing solution for real-time UAV navigation in partially observable and dynamic environments.

In future work, we aim to extend this architecture by incorporating hierarchical or Transformer-based temporal models with Delta encoding for complex mission planning in dynamic and uncertain environments. The perception and planning capabilities may be enhanced by incorporating more sensory input (e.g., vision, radar, or event-based cameras).

While the proposed PPO-GRU with Delta-LiDAR architecture significantly improves UAV navigation for partial observability, there are issues that remain. This work is based on 2D LiDAR, which is beneficial for the detection of planar obstacles but does not fully capture vertical features. While ego-motion correction was employed, small residual discrepancies can still influence the Delta-LiDAR signal in the case of aggressive maneuvers. Additionally, sensor sampling speeds and onboard processing speeds inherently limit the system's responsiveness to extremely rapid changes in the environment. Finally, since the technique has been simulated and verified, eventual field deployment will involve further hardware integration as well as robustness testing.

## ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the professors and colleagues at the Department of Computer Engineering, University of Basrah, for their valuable support and guidance throughout the development of this study.

## REFERENCES

[1] Z. Fang and A. V. Savkin, "Strategies for Optimized UAV Surveillance in Various Tasks and Scenarios: A Review," Drones, vol. 8, no. 5, Art. no. 193, May 2024, doi: 10.3390/drones8050193.
[2] W. Alawad, N. Ben Halima, and L. Aziz, "An Unmanned Aerial Vehicle (UAV) System for Disaster and Crisis Management in Smart Cities," Electronics (Switzerland), vol. 12, no. 4, Feb. 2023, doi: 10.3390/electronics12041051.
[3] H. S. Munawar, F. Ullah, S. Qayyum, S. I. Khan, and M. Mojtahedi, "UAVs in disaster management: Application of integrated aerial imagery and convolutional neural network for flood detection," Sustainability (Switzerland), vol. 13, no. 14, Jul. 2021, doi: 10.3390/su13147547.
[4] Štimac, Igor and Mihetec, Tomislav, "Application of Drones in Urban Areas," Transportation Research Procedia, vol. 81, pp. 84-97, 2024, doi.org/10.1016/j.trpro.2024.11.010
[5] Ke Wang and Yining Chen, "Multi-UAV Navigation for Partially Observable Communication Coverage by Graph Reinforcement Learning," IEEE Transactions on Mobile Computing, vol. 22, no. 7, 2023.
[6] M. Wisniewski, P. Chatzithanos, W. Guo, and A. Tsourdos, "Benchmarking Deep Reinforcement Learning for Navigation in Denied Sensor Environments," arXiv preprint arXiv:2410.14616, vol. 1, 2024.
[7] V. J. Hodge, R. Hawkins, and R. Alexander, "Deep reinforcement learning for drone navigation using sensor data," Neural Comput Appl, vol. 33, no. 6, pp. 2015–2033, Mar. 2021, doi: 10.1007/s00521-020-05097-x.
[8] A. P. Kalidas, C. J. Joshua, A. Q. Md, S. Basheer, S. Mohan, and S. Sakri, "Deep Reinforcement Learning for Vision-Based Navigation of UAVs in Avoiding Stationary and Mobile Obstacles," Drones, vol. 7, no. 4, p. 245, 2023, doi: 10.3390/drones7040245.
[9] Y. Sheng, H. Liu, J. Li, and Q. Han, "UAV Autonomous Navigation Based on Deep Reinforcement Learning in Highly Dynamic and High-Density Environments," Drones, vol. 8, no. 9, Sep. 2024, doi: 10.3390/drones8090516.
[10] J. Terven, "Deep Reinforcement Learning: A Chronological Overview and Methods," AI (Switzerland), vol. 6, no. 3, p. 46, 2025, doi: 10.3390/ai6030046.
[11] J. Hu, X. Yang, W. Wang, P. Wei, L. Ying, and Y. Liu, "Obstacle Avoidance for UAS in Continuous Action Space Using Deep Reinforcement Learning," IEEE Access, vol. 10, n. 90623, 2022, doi: 10.1109/ACCESS.2022.3201962.
[12] T. Guo, N. Jiang, B. Li, X. Zhu, Y. Wang, and W. Du, "UAV navigation in high dynamic environments: A deep reinforcement learning approach," Chinese Journal of Aeronautics, vol. 34, no. 2, pp. 479–489, 2021, doi: 10.1016/j.cja.2020.05.011.
[13] Y. Zhou et al., "Deep Reinforcement Learning for Real-Time Airport Emergency Evacuation Using Asynchronous Advantage Actor–Critic (A3C) Algorithm," Mathematics, vol. 13, no. 14, p. 2269, 2025, doi: 10.3390/math13142269.
[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms." arXiv preprint arXiv:1707.06347, 2017.
[15] S. L. Brunton and J. N. Kutz, "Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control", 2nd ed. Cambridge, U.K.: Cambridge University Press, 2022, doi: 10.1017/9781108380690.
[16] J. Zhao et al., "Deep Reinforcement Learning-Based End-to-End Control for UAV Dynamic Target Tracking," Biomimetics, vol. 7, no. 4, 2022, doi: 10.3390/biomimetics7040197.
[17] Z. Jiang and G. Song, "A deep reinforcement learning strategy for UAV autonomous landing on a platform," arXiv:2209.02954, 2022.
[18] O. Araar, N. Aouf, and I. Vitanov, "Vision-Based Autonomous Landing of Multirotor UAV on Moving Platform," Journal of Intelligent and Robotic Systems: Theory and Applications, vol. 85, no. 2, pp. 369–384, 2017, doi: 10.1007/s10846-016-0399-z.
[19] Z. Hu, K. Wan, X. Gao, Y. Zhai, and Q. Wang, "Deep reinforcement learning approach with multiple experience pools for UAV's autonomous motion planning in complex unknown environments," Sensors, vol. 20, no. 7, Art. 1890, 2020, doi: 10.3390/s20071890.
[20] J. Zhao, J. Sun, Z. Cai, L. Wang, and Y. Wang, "End-to-end deep reinforcement learning for image-based UAV autonomous control," Applied Sciences (Switzerland), vol. 11, no. 18, pp. 1–21, 2021, doi: 10.3390/app11188419.

[21] F. Hêche, O. Barakat, T. Desmettre, T. Marx, and S. Robert-Nicoud, "Offline reinforcement learning in high-dimensional stochastic environments," Neural Computing and Applications, vol. 36, pp. 585–598, 2024, doi: 10.1007/s00521-023-09029-3.
[22] B. Rubí, B. Morcego, and R. Pérez, "Quadrotor Path Following and Reactive Obstacle Avoidance with Deep Reinforcement Learning," Journal of Intelligent and Robotic Systems: Theory and Applications, vol. 103, no. 4, 2021, doi: 10.1007/s10846-021-01491-2.
[23] H. Samma and S. El-Ferik, "Autonomous UAV Visual Navigation Using an Improved Deep Reinforcement Learning," IEEE Access, vol. 12, pp. 79967–79977, 2024, doi: 10.1109/ACCESS.2024.3409780.
[24] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned Aerial Vehicle Path Planning Algorithm Based on Deep Reinforcement Learning in Large-Scale and Dynamic Environments," IEEE Access, vol. 9, pp. 24884–24900, 2021, doi: 10.1109/ACCESS.2021.3057485.
[25] P. Miera, H. Szolc, and T. Kryjak,"LiDAR-based drone navigation with reinforcement learning,"*TechRxiv*, Jul. 2023, doi:10.36227/techrxiv 23784246.v1.
[26] F. S. Leira, H. H. Helgesen, T. A. Johansen, and T. I. Fossen, "Object detection, recognition, and tracking from UAVs using a thermal camera," Journal of Field Robotics, vol. 38, no. 2, pp. 242–267, 2021, doi:10.1002/rob 21985.
[27] S. Zhao, W. Wang, J. Li, S. Huang, and S. Liu, "Autonomous Navigation of the UAV through Deep Reinforcement Learning with Sensor Perception Enhancement," Mathematical Problems in Engineering, vol. 2023, 2023, doi: 10.1155/2023/3837615.
[28] A. T. Azar and A. Koubaa, "Drone deep reinforcement learning: A review," Electronics, vol. 10, no. 9, Art. 999, 2021, doi: 10.3390/electronics10090999.
[29] K. Ullah et al., "Short-Term Load Forecasting: A Comprehensive Review and Simulation Study with CNN-LSTM Hybrids Approach," IEEE Access, vol. 12, pp. 111858–111881, 2024, doi: 10.1109/ACCESS.2024.3440631.
[30] J. Zhang, Y. Guo, L. Zheng, Q. Yang, G. Shi, and Y. Wu, "Real-Time UAV Path Planning Based on LSTM Network," Journal of Systems Engineering and Electronics, vol. 35, no. 2, pp. 374–385, 2024, doi: 10.23919/JSEE.2023.000157.
[31] H. Liu, Y. Shen, S. Yu, Z. Gao, and T. Wu, "Deep reinforcement learning for mobile robot path planning," J. Theory Pract. Eng. Sci., vol. 4, no. 3, 2024.
[32] Z. Zheng and H. Duan, "UAV maneuver decision-making via deep reinforcement learning for short-range air combat," Intelligence and Robotics, vol. 3, no. 1, pp. 76–94, 2023, doi: 10.20517/ir 2023.04.
[33] B. Ghojogh and A. Ghodsi, "Recurrent neural networks and long short-term memory networks: Tutorial and survey," arXiv:2304.11461, 2023. [Online]. Available: https://arxiv.org/abs/2304.11461
[34] Y. Zhang, W. Li, H. Wang, and M. Chen, "Memory-based deep reinforcement learning for COLREGs-compliant obstacle avoidance in USV with limited environmental knowledge," Ocean Engineering, vol. 297, p. 121978, 2025. doi: 10.1016/j.oceaneng.2025.121978.
[35] Z. Niu et al., "Recurrent attention unit: A new gated recurrent unit for long-term memory of important parts in sequential data." Neurocomputing, vol. 517, pp. 1–9, 2023, doi: 10.1016/j.neucom.2022.10.050.

## BIOGRAPHY OF AUTHORS

**Maryam Allawi Haddad** obtained a bachelor's degree in computer engineering from the University of Basra, College of Engineering, in 2020. She is currently pursuing a master's degree in computer engineering at the same university. Her research focuses on artificial intelligence, deep learning, and object detection, with a particular emphasis on practical applications and improving learning algorithms.

**Dhayaa R. Khudher** is a researcher at the University of Basrah's Department of Computer Engineering, where he investigates machine learning-driven control systems for robotics and UAVs. He received his Ph.D. from Brunel University London. His work aims to advance the intelligence and operational efficiency of autonomous systems in dynamic environments.