

Temporal-Augmented Observation for Navigation of Unmanned Aerial Vehicles: a Recurrent Reinforcement Learning Architecture

Gabriele Gemignani, Adolfo Perrusquía, Antonios Tsourdos, Lorenzo Pollini

Abstract—In aerial robotics, data-driven Reinforcement Learning (RL) approaches have proven highly effective for obstacle avoidance and goal-directed navigation, especially when operating on high-dimensional sensor data that provide only partial, local information about the environment. Such limited observability, combined with irregularly shaped obstacles, poses significant challenges for reactive control policies that rely solely on instantaneous observations. To address these issues, this paper introduces a Twin Deep Deterministic Policy Gradient (TD3)-based algorithm that leverages explicit Temporal Augmentation of the Observation space (TAO-TD3). The proposed method preserves the simplicity of the original TD3 framework by augmenting the observation with a short history of past states and incorporating a lightweight recurrent network, without requiring changes to the TD3 training paradigm. Extensive simulations across diverse environmental topographies and irregular obstacle shapes demonstrate that the proposed approach nearly halves the collision rate and improves overall navigation success compared to feedforward RL-based architectures.

I. INTRODUCTION

The navigation of Unmanned Aerial Vehicles (UAVs) towards a known target through obstacle-dense environments is a key problem of interest across various sectors of the automation industry. Several challenges still need to be tackled before reliably and efficiently deploying navigation systems in real-world UAV applications; above all, we mention strict onboard payload, limited knowledge of the surrounding environment, deadlocks caused by local minima or irregularly-shaped obstacles, and scarcity or degradation of sensory data (e.g., GPS-denied indoor scenarios). Classical global planning algorithms such as A* and Dijkstra rely on discrete map representations and strong assumptions on environmental knowledge, which limit their standalone applicability in modern UAV navigation [1]. Recent works therefore combine them with numerical optimization techniques to enhance model constraints [2]. However, optimization-based methods, including Particle Swarm Optimization [3], still pose significant computational challenges for onboard deployment, especially in large-scale environments. Lightweight approaches like Artificial Potential Fields (APF) have shown effectiveness in handling dynamic obstacles and local minima [4], [5], but they strongly depend on

a priori map information. This assumption becomes fragile in scenarios with frequent reconfigurations, where obstacles are carried around (e.g., packages in a warehouse) and the navigation system requires manual re-tuning.

To overcome the reliance on explicit environment modeling, learning-based, model-free Reinforcement Learning (RL) has emerged as a promising alternative for UAV navigation in obstacle-dense environments [6]. RL maps raw sensor data directly into control commands, enabling adaptive navigation without requiring a predefined map. Such an approach has been successfully applied both to navigation on discrete representations [7]–[9] and to end-to-end control in continuous state and action spaces [10]–[14]. Extensions of these methods further address three-dimensional navigation [9], [10] and dynamic environments, including teammates and moving targets [15], [16].

Most works equip UAVs with perceptive, obstacle-aware observations to enable adaptive navigation in dynamic environments. The simplest perception model augments the observation vector with distances to nearby obstacles [14], [17], which clearly struggles with complex geometries. Another approach uses cameras to extract visual features of the environment [12], [13]; this typically relies on Convolutional Neural Networks (CNNs), which are nevertheless computationally demanding at run-time. Finally, raw distance measurements from sensors such as multi-beam rangefinders are simulated with varying degrees of accuracy [10], [15]; this approach is much more lightweight and, as shown in [16], a rangefinder-based RL navigation policy can be successfully deployed in real-world scenarios.

Neural architecture is critical for UAV navigation under partial observability and complex obstacle topologies. Several works introduced memory and temporal processing into the policy to help the agent integrate information over time and mitigate issues such as local minima and “infinite-loop” behaviors in cluttered environments. Recurrent models based on Long Short-Term Memory (LSTM) networks [12], [18], [19] and partially observable deep-Q-learning [20] have been explored. Closely related to our work, Zhang et al. [21] employ an LSTM-based architecture for real-time UAV path planning in a rangefinder-based continuous path-planning problem, and confirm the benefit of recurrent models in capturing temporal dependencies. More recently, attention- and transformer-based architectures, which can selectively focus on salient spatial and temporal features, have been investigated to handle long-range dependencies in partially observable RL navigation tasks [22]–[24]. Taken together, these studies show that temporal modeling is increasingly im-

Gabriele Gemignani, Adolfo Perrusquía, and Antonios Tsourdos are with the Faculty of Engineering and Applied Sciences, Cranfield University, MK43 0AL, Bedford, UK.

Gabriele Gemignani is a PhD student of the DAuSy National Programme for Polytechnic of Bari, Dept. of Electrical and Information Engineering, Via Re David 200, 70125 Bari, Italy, email: g.gemignani@phd.poliba.it

Gabriele Gemignani and Lorenzo Pollini are with the University of Pisa, Dept. of Information Engineering, 56126 Pisa, Italy.

portant for UAV navigation. However, unlike our approach, they do not explicitly represent the input as a temporally augmented observation, nor do they integrate such memory into a deterministic actor-critic framework such as TD3.

This manuscript addresses the problem of goal-oriented UAV navigation with rangefinder-based perception to provide obstacle-awareness to rapidly changing maps. Building upon this environment, our main contribution lies in the training framework: we propose a recurrent-style neural network featuring a Temporally-Augmented Observation and trained based on the TD3 algorithm (TAO-TD3) [25], already used in the RL community for continuous navigation [10]. We employ the mechanism of memory in our actor-critic architecture, as investigated similarly by other works [16], but differently introduce an augmented observation that feeds a sequence of past observations to the actor network, requiring minimum change to the algorithm. Moreover, unlike similar works that approximate obstacles with simple convex shapes (e.g., circles) [21], [24], our navigation scenario includes generic polygonal obstacles, increasing realism and further exacerbating local-minima phenomena that commonly arise in reactive navigation.

The rest of the article is organized such that: in Section II we formulate the UAV navigation problem; in Section III we describe our neural architecture proposal; in Section IV we present the numerical simulations and in Section V we conclude with findings of this study and future works.

II. UAV NAVIGATION PROBLEM FORMULATION

A. Physical Scenario

We consider a 2D continuous map of size L . While the work we present can be extended to non-squared maps without loss of generality, the planar model hypothesis is, to the authors' opinion, reasonable when employing UAVs, as commonly assumed for instance in coverage and inspection problems [26]. Let $\mathbf{p} = (x, y)$ and $\mathbf{p}^* = (x^*, y^*)$ be the positions on the map respectively of the UAV agent and of the target point. Also, denote as θ the UAV's yaw angle taken w.r.t. the inertial reference frame. We model the UAV planar motion as a unicycle-like discrete kinematics with control inputs the linear and angular velocity, v and ω :

$$\begin{aligned} x_{k+1} &= x_k + \Delta T v_k \cos \theta_k \\ y_{k+1} &= y_k + \Delta T v_k \sin \theta_k \\ \theta_{k+1} &= \theta_k + \Delta T \omega_k \end{aligned} \quad \text{with} \quad \begin{cases} 0 \leq v_k \leq \bar{v} \\ -\bar{\omega} \leq \omega_k \leq \bar{\omega} \end{cases} \quad (1)$$

Here, ΔT is the control sampling time, and the inputs are saturated. The objective is to generate control sequences that steer the UAV from a randomly initialized position $\mathbf{p}_0$ to the target in minimum time while avoiding obstacles along the path.

B. Obstacle Generation

Obstacles are generated at the beginning of each scenario and remain static during the UAV navigation. Two environment parameters define the obstacle density: ϱ , the fraction of the map area occupied by obstacles, and Z , the total number

of obstacles. The area of each obstacle is hence sampled randomly ensuring the total obstacle area is ϱL^2 .

Obstacles can be either circular or irregular polygons. The latter are obtained by radially perturbing a nominal circle: a set of vertices is placed uniformly around a center point, and each vertex's radial distance is randomly varied to produce irregular, star-shaped geometries. Obstacles are then positioned on the map by uniformly sampling their centers. Any obstacle whose footprint overlaps with the UAV's initial or target location is discarded and resampled.

C. Perception Model

Aside from global information on the target localization, the UAV receives feedback on its surroundings through a multi-beam range sensor. This allows to cope with obstacles of diverse locations and geometries.

1) *Global Perception*: The global observation vector at discrete time k , $\mathbf{o}_k^g$, is computed given the absolute positions of the agent and the target:

$$\mathbf{o}_k^g = \left[d_k/L, (\cos \psi_k + 1)/2, (\sin \psi_k + 1)/2 \right] \quad (2)$$

with the euclidean distance d and the heading angle ψ simply computed from planar geometry:

$$d_k = \|\mathbf{p}_k - \mathbf{p}^*\|_2, \quad \psi_k = \text{atan2}(y^* - y_k, x^* - x_k) - \theta_k$$

The algebraic manipulations carried out in (2) seek to normalize the quantities between 0 and 1.

2) *Local Perception*: Our 2D rangefinder sensor model resembles, for instance, the ones in [10], [24] (Fig. 1). The number of rangefinder beams N are symmetrically distributed and equally spaced in the angular range $[\theta_k - \varphi, \theta_k + \varphi]$, around the yaw angle. Each rangefinder beam

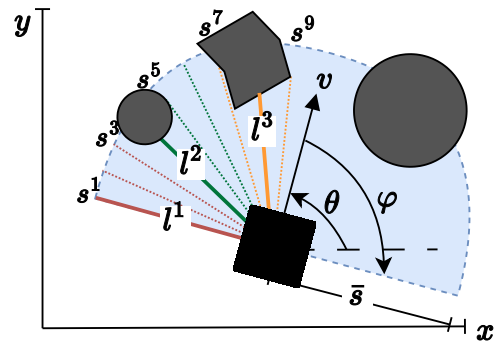


Fig. 1: **Local Range measurement Model.** Illustrative example with parameters N, B, ϕ different from the ones used in Sec.IV. Three sub-groups of rangefinder beams (highlighted in different colors) are shown within the full field of view 2φ (light-blue region). Solid beams represent the reduced observation obtained by taking the minimum range within each group, whereas the remaining beams (dotted) are discarded.

measures:

$$s^i = \begin{cases} \bar{s} & \text{if beam } i \text{ intercepts no obstacles} \\ \text{distance} + \mathcal{N}(0, \sigma^s) & \text{otherwise} \end{cases} \quad (3)$$

where $\bar{s}$ is the maximum range of the sensor and σ^s is the standard deviation of a zero-mean noise, inserted to enhance the realism of the sensor's model. Instead of using an N -dimensional observation vector containing all beam measurements, we employ a method to compress the feedback. This reduces the network size and facilitates training convergence, while preserving the most relevant information. Let B be a design parameter that represents the number of sub-groups equally splitting the whole rangefinder data. Then, for each group, we only keep the most informative beam, which is the minimum distance from an obstacle:

$$l^b = \min\{s^{(b-1)(N/B)+1}, \dots, s^{b(N/B)}\}, \quad b = 1, \dots, B \quad (4)$$

These B quantities form the local observation vector, which is again normalized between 0 and 1 by dividing for the maximum range distance:

$$\mathbf{o}_k^\ell = [l_k^1/\bar{s}, \dots, l_k^B/\bar{s}] \quad (5)$$

The navigation policy, in our case an RL-trained neural network, receives as input the observation vector $\mathbf{o}_k = [\mathbf{o}_k^g, \mathbf{o}_k^\ell]$ and outputs the control $\mathbf{u}_k = [v_k, \omega_k]$.

D. Reward Function

RL algorithms train a neural network optimizing the sum of discounted rewards obtained by the agent along its trajectory [6]. The reward function $r_k(\mathbf{o}_k, \mathbf{u}_k)$ must therefore reflect the objective of the problem; for our case study, we designed it as follows:

$$r_k = \begin{cases} \begin{cases} 100 & \text{goal reached} \\ -100 & \text{obs. or walls hit} \end{cases} \rightarrow \text{scenario ends} \\ v_k - \frac{|\omega_k|}{2} - \frac{\Gamma(\min_{1 \leq b \leq B}\{l_k^b\})}{2}, & \text{otherwise} \end{cases} \quad (6)$$

with the function Γ which introduces a proximity penalty for obstacles within a threshold of 1 meter, namely:

$$\Gamma(c) = \max(1 - c, 0) \quad (7)$$

The penalty increases the closer the UAV gets to an obstacle. The first two cases in (6) represent strong, sparse rewards corresponding to task success or failure, ensuring that reaching the goal or colliding with an obstacle dominates the overall reward. They also determine termination of the current scenario, a.k.a. episode. The third case encourages desirable behavior during normal operation: i) The linear velocity term v_k incentivizes faster motion, promoting shorter task duration. ii) The angular velocity penalty $|\omega_k|/2$ discourages sharp turns, encouraging smoother trajectories. iii) The proximity penalty Γ described in (7) encourages safe around-the-target motion.

III. TEMPORALLY-AUGMENTED TD3 ARCHITECTURE

A. Augmented Observation and Actor-critic Architecture

We define a Temporally-Augmented Observation (TAO) vector using a First-In-First-Out (FIFO) queue of fixed length H . At the start of each training episode, the queue is initialized with H copies of the initial observation. At each subsequent time step, the newest observation is appended to the back of the queue, and the oldest entry is removed. The TAO $\tilde{\mathbf{o}}_k$ is obtained by vertically stacking (notation $[\cdot; \cdot]$) the H past observation vectors:

$$\tilde{\mathbf{o}}_k = [\mathbf{o}_{k-H+1}; \dots; \mathbf{o}_k] \in \mathbb{R}^{H \times |\mathbf{o}|} \quad (8)$$

with $|\mathbf{o}| = 3 + B$ following (2) and (5). At this point, the TAO is fed to both actor and critic, whose architecture is schematically illustrated in Figure 2. The actor and critic networks share a recurrent feature extraction backbone. The TAO is first processed by a feedforward feature extractor applied independently at each row of $\tilde{\mathbf{o}}_k$, namely each state of the queue from $k - H + 1$ to k . The resulting temporal feature sequence is then passed to a single LSTM layer, which models temporal correlations across the observation history. Only the final hidden state of the recurrent module is retained, providing a compact representation of the recent trajectory. In the actor network, this temporal embedding is mapped to the control action through a feedforward head and a hyperbolic tangent activation to scale the outputs in the $[-1, 1]$ range. Since the policy is deterministic, namely the actor outputs directly the control variables, an external noise is added to account for exploration of new policies during training, as standard practice in RL algorithms [6]. Finally, before being applied to the navigation environment, the control is scaled to match the kinematics given by (1). On the critic side, instead, the LSTM's last hidden state is fed to two head networks responsible for estimating the Q-functions of the TD3 training algorithm.

B. TD3 Training Framework

At each episode step, transitions of the form $(\tilde{\mathbf{o}}_k, \mathbf{u}_k, r_k, \tilde{\mathbf{o}}_{k+1})$ are stored in a replay buffer and, at the end of the episode, a TD3 training is carried out using randomly sampled mini-batches. This represents the only difference from the classical TD3 training scheme: a list of observations $\tilde{\mathbf{o}}_k$ is stored and subsequently used as training batch rather than a single observation vector $\mathbf{o}_k$.

Compared to standard Deep Deterministic Policy Gradient, TD3 improves training stability by employing a double critic architecture [25]: let Q_{ϕ_j} denote the two critic networks ($j = 1, 2$) parametrized by weights ϕ_j . Each critic is trained by minimizing the squared Bellman error:

$$\mathcal{L}_j^{\text{critic}}(\phi_j) = \mathbb{E}[(Q_{\phi_j}(\tilde{\mathbf{o}}_k, \mathbf{u}_k) - y_k)^2], \quad (9)$$

$$y_k = r_k + \gamma \min_{j=1,2} Q_{\phi_j}'(\tilde{\mathbf{o}}_{k+1}, \pi_{\zeta'}(\tilde{\mathbf{o}}_{k+1}) + \epsilon) \quad (10)$$

with ϵ a clipped white noise and γ the discount factor. The actor π_ζ is unique, parametrized by weights ζ , and is

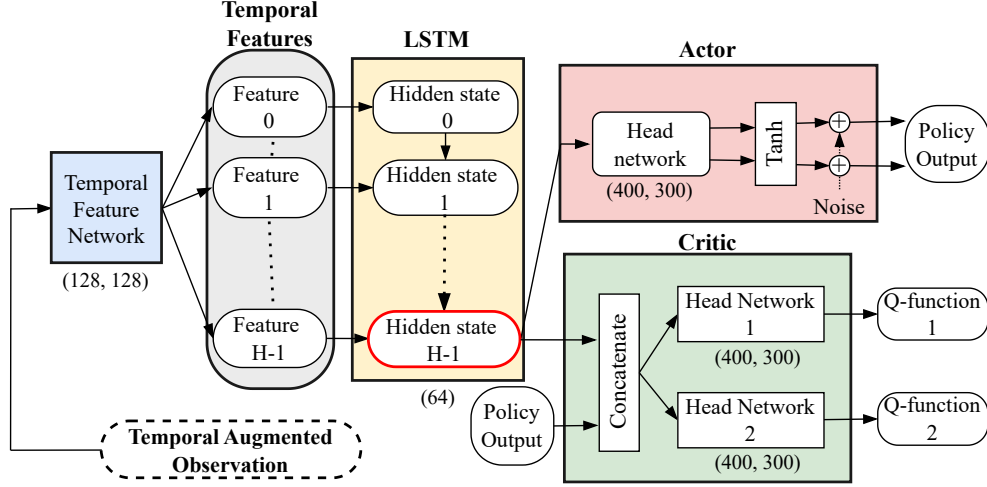


Fig. 2: TAO-TD3 actor and critic networks. The TAO input is applied independently to each row (H entries) and processed through a feedforward network to have H temporal features describing the agent’s past trajectory. These pass through an LSTM layer, whose last hidden state provides a compact representation of the trajectory history. This representation is subsequently fed into both the actor and the critic networks, with the critic additionally receiving the policy output. The sizes of the feedforward and recurrent modules reported in parentheses correspond to our implementation. LeakyReLU is used all over as non-linear activation.

optimized to maximize the first critic value:

$$\mathcal{L}^{\text{actor}}(\zeta) = \mathbb{E}[Q_{\phi_1}(\tilde{\mathbf{o}}_k, \pi_{\zeta}(\tilde{\mathbf{o}}_k))] \quad (11)$$

In TD3, the actor is updated less frequently than the critics to further improve training stability. At the end of the training loop, target networks (ϕ'_j, ζ') are updated through standard soft updates with rate τ , ensuring stable temporal propagation of value estimates:

$$\phi'_j \leftarrow \tau \phi_j + (1 - \tau) \phi'_j, \quad \zeta' \leftarrow \tau \zeta + (1 - \tau) \zeta'. \quad (12)$$

IV. NUMERICAL RESULTS

A. Simulations Setting

We validated our TAO-TD3 algorithm for 2D UAV navigation in an environment defined as follows. We took a map of size 10 m and, accordingly, a maximum linear and angular velocity in norm equal to 0.5 m/s and 1 s^{-1} . The sampling time of the algorithm is chosen as $\Delta T = 0.3 \text{ s}$. For the obstacles, we set the following parameters: $Z = 8$, $\varrho = 0.2$, since the resulting scenario is mildly dense of obstacles of different shapes. As regards the rangefinder, we opted for $N = 180$, $\bar{s} = 7 \text{ m}$, $\varphi = \pi/2$ and $\sigma^s = 0.08$. The number of groups in which the rangefinder beams are divided is $B = 20$. Interestingly, this reduces the observation space dimension from 183 to 23, namely the 20 range measurements in (4) plus the three quantities in (2). From an algorithmic perspective, after some trial-and-error, we found that the number of past observations to store which yields a boost in performance is around $H = 20$. The performance is compared with feedforward baselines, namely architectures like Fig.2 but without the LSTM module and with non-augmented, classical environment observation. In

particular, we chose TD3 [25] and Soft-Actor-Critic (SAC) [27], respectively a deterministic and stochastic policy in the field of continuous RL. For an honest comparison, we used the same set of common hyperparameters, plus some extra fine-tuned ad-method hyperparameters (Table I). Both training and simulation campaigns were run on NVIDIA A100 GPU with 80 Gb of RAM.

B. Training Performance

During training, we evaluated the network on environments with the same parameters but varying obstacle configurations and initial conditions. Every 50 training episodes, we ran 50 evaluation episodes and recorded key metrics. Fig. 3 shows the fraction of episodes ending in collision and the average episodic reward. The TAO-TD3 approach outperforms feedforward baseline architectures on both metrics, although learning progresses slightly slower due to the larger network size. Overall, the recurrent TAO architecture improves collision avoidance and better aligns with the objectives encoded

TABLE I: Training Hyperparameters

Actor/Critic Learning rate	1e-4
Batch size	64
Buffer size	5e4
Critic updates per episode	10
Actor updates per episode	5
Discount factor γ	0.99
Soft update τ	0.005
Temperature Learning rate (SAC)	1e-4
Policy noise ϵ (TD3/TAO-TD3)	$\text{clip}(\mathcal{N}(0, 0.15), -0.5, 0.5)$
Std Clipping (SAC)	1e-3

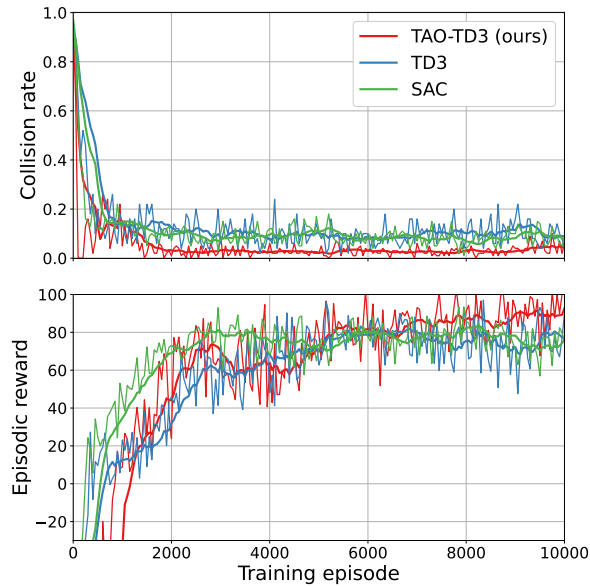


Fig. 3: Training trend across all algorithms. The more solid line is a 10-point moving average of the raw data, these represented as more transparent color.

in the reward. In contrast, no significant differences were observed between the deterministic TD3 and the stochastic SAC policies.

C. Policy Evaluation

All policies were evaluated in a Monte-Carlo fashion to obtain global performance metrics. For a fair comparison, each test used the same map, UAV initialization, and target positions across all algorithms. The mean results are reported in Table II. TAO-TD3 outperforms both TD3 and SAC in episodic reward, goal achievement, and collision rate, while achieving similar performance to TD3 in navigation length. The largest improvement is observed in the collision rate, which is nearly halved compared to the baselines. It should be noted that collision values may include cases where the UAV or target were initialized very close to obstacles; thus, these metrics serve as relative indicators under consistent conditions. A drawback of the proposed approach is its higher computational cost, being roughly 100 times more intensive than SAC.

Lastly, we illustrate representative simulations of all analyzed policies in the navigation scenario in Fig. 4:

TABLE II: Monte-Carlo mean metrics.

Metric	SAC	TD3	TAO-TD3
Episodic Reward	78.67	77.48	85.01
Goal Rate (%)	90.7	89.6	92
Collision Rate (%)	7.3	8.9	4.4
Steps to Goal (1 step = 0.3s)	43.6	38.3	38.6
Computational Time (ms)	0.33	0.20	36.65

*Results over 3000 random tests.

a) Fig. 4a shows that TAO-TD3 is the only policy capable of overcoming an irregularly-shaped, “edgy” obstacle. This occurs because TAO-TD3 leverages the history of states to infer the obstacle shape, whereas feedforward architectures collide once the obstacle leaves the sensor’s view (this phenomenon is known in the literature as *Wrap-Around*).

b) Fig. 4b depicts a typical local minima situation. The UAV initially points to the right; SAC and TD3 first approach the target along this path but then encounter a narrow passage between an obstacle and a wall. TD3 collides within the passage, while SAC eventually escapes and reaches the target via an alternative route. TAO-TD3 discovers the narrow passage more quickly and takes the shortest route to the target, effectively following the same path as SAC but earlier.

c) Fig. 4c illustrates another local minima scenario. Again, TAO-TD3 is the quickest to escape a dead-end caused by densely-grouped obstacles. Additionally, when the target is close to obstacles, SAC and TD3 produce jerky, segmented trajectories, whereas TAO-TD3 generates smooth control, demonstrating improved performance in tight environments.

V. CONCLUSIONS

In this work, we presented TAO-TD3, a temporally-augmented variant of TD3 for UAV navigation in unknown, obstacle-dense 2D environments. By incorporating a short history of past states and processing it through an LSTM, with minimal changes to the TD3 training framework, the policy effectively captures temporal correlations, improving robustness to partial observability, local minima, and irregularly-shaped obstacles. Simulation results show that TAO-TD3 achieves lower collision rates with irregular obstacles’ morphologies, higher and faster goal-reaching success trajectories compared to feedforward TD3 and SAC, indicating that the observed improvements are primarily due to the recurrent neural architecture rather than the underlying RL algorithm choice.

While these results are obtained in moderately cluttered maps, they suggest that memory-based policies can significantly enhance UAV performance in cluttered scenarios. Future work will explore multi-agent settings and more complex environments, including moving obstacles and dynamic targets, to further evaluate the potential of temporally-augmented policies in realistic applications.

ACKNOWLEDGMENTS

Project co-funded by the European Union – Next Generation Eu - under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 1 Investment 4.1 - Decree No. 118 (2nd March 2023) of Italian Ministry of University and Research - Concession Decree No. 2333 (22nd December 2023) of the Italian Ministry of University and Research, Project code D93C23000450005, within the Italian National Program PhD Programme in Autonomous Systems (DAuSy).

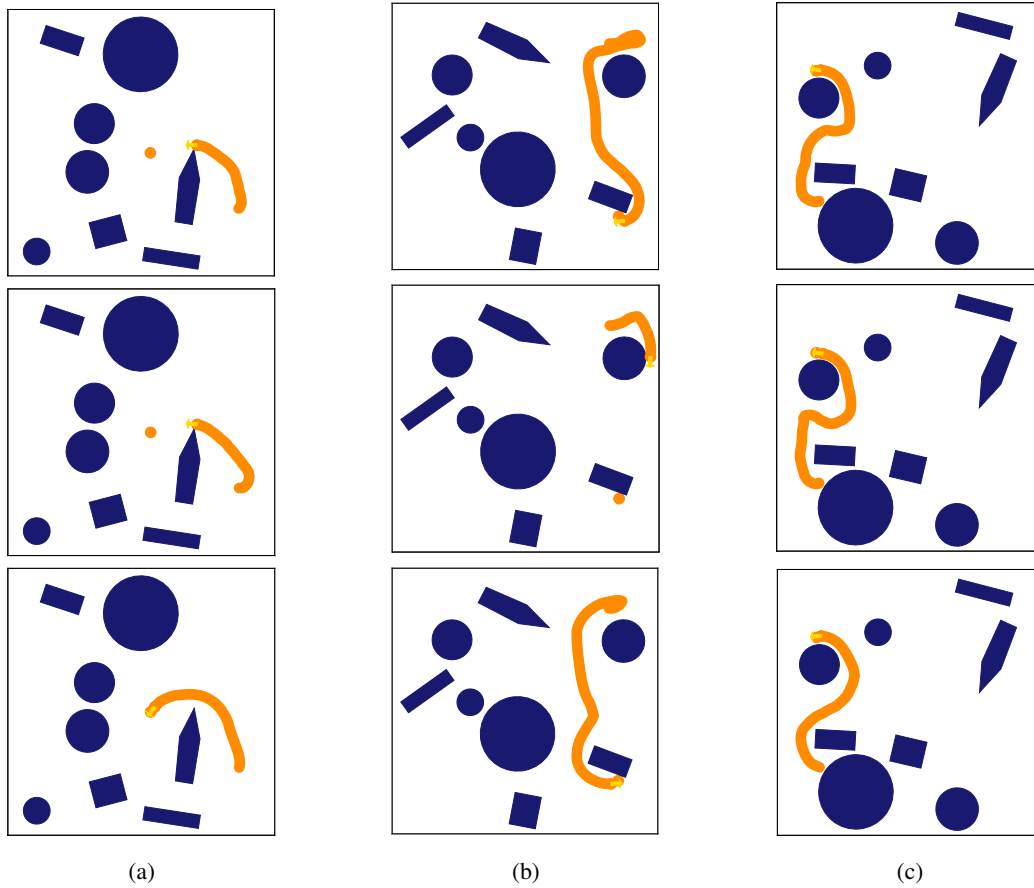


Fig. 4: Sample trajectories with different navigation algorithms. From the top to the bottom row: SAC, TD3, TAO-TD3.

REFERENCES

- [1] A. M. Ali, A. Perrusquía, W. Guo, and A. Tsourdos, "Flight plan optimisation of unmanned aerial vehicles with minimised radar observability using action shaping proximal policy optimisation," *Drones*, vol. 4, 2023.
- [2] N. Hohmann, S. Brulin, J. Adamy, and M. Olhofer, "Three-dimensional urban path planning for aerial vehicles regarding many objectives," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, 2023.
- [3] M. Haris and H. Nam, "Path planning optimization of smart vehicle with fast converging distance-dependent pso algorithm," *IEEE Open Journal of Intelligent Transportation Systems*, 2024.
- [4] H. M. Jayaweera and S. Hanoun, "A dynamic artificial potential field (d-apf) uav path planning technique for following ground moving targets," *IEEE Access*, vol. 8, 2020.
- [5] T. Huang, D. Huang, N. Qin, and Y. Li, "Path planning and control of a quadrotor uav based on an improved apf using parallel search," *International Journal of Aerospace Engineering*, 2021.
- [6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, 2015.
- [7] 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, 2021.
- [8] G. Gemignani, M. Bongiorno, and L. Pollini, "An energy-aware decision-making scheme for mobile robots on a graph map based on deep reinforcement learning," in *2024 18th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2024.
- [9] H. Han, J. Cheng, M. Lv, and Z. Xi, "Autonomous navigation of uavs in unknown 3d environments using deep reinforcement learning for path planning," *IEEE Transactions on Vehicular Technology*, 2025.
- [10] S. Zhang, Y. Li, and Q. Dong, "Autonomous navigation of uav in multi-obstacle environments based on a deep reinforcement learning approach," *Applied Soft Computing*, vol. 115, 2022.
- [11] K. Tamanakijprasart, A. Perrusquia, S. Mondal, and A. Tsourdos, "Autonomous path selection of unmanned aerial vehicle in dynamic environment using reinforcement learning," in *AIAA SCITECH 2025 Forum*, 2025.
- [12] J. Wu, Y. Ye, and J. Du, "Multi-objective reinforcement learning for autonomous drone navigation in urban areas with wind zones," *Automation in Construction*, vol. 158, 2024.
- [13] H. Wu, W. Wang, T. Wang, and S. Suzuki, "Model-free uav navigation in unknown complex environments using vision-based reinforcement learning," *Drones*, vol. 9, no. 8, 2025.
- [14] X. Ren, N. Geng, Y. Zhang, L. Xiao, and D. Gong, "Pg-td3: A potential field-guided deep reinforcement learning approach for uav path planning after disaster," *IEEE Transactions on Automation Science and Engineering*, vol. 22, 2025.
- [15] J. Wang, P. Zhang, and Y. Wang, "Autonomous target tracking of multi-uav: A two-stage deep reinforcement learning approach with expert experience," *Applied Soft Computing*, vol. 145, p. 110604, 2023.
- [16] Y. Xue and W. Chen, "Multi-agent deep reinforcement learning for uavs navigation in unknown complex environment," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, 2024.
- [17] M. Collignon, A. Perrusquía, A. Tsourdos, and W. Guo, "Search and rescue operations in wildfires using unmanned aerial vehicles: A multi-agent deep reinforcement learning approach," *Neurocomputing*, 2025.
- [18] G. Tong, N. Jiang, L. Biye, Z. Xi, W. Ya, and D. Wenbo, "Uav navigation in high dynamic environments: A deep reinforcement learning approach," *Chinese Journal of Aeronautics*, 2021.
- [19] X. Yuntao and C. Weisheng, "A uav navigation approach based on deep reinforcement learning in large cluttered 3d environments," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, 2022.
- [20] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in uav with limited envi-

- ronment knowledge,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, 2021.
- [21] 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.
 - [22] Z. Huang, Z. Yang, R. Krupani, B. Şenbaşlar, S. Batra, and G. S. Sukhatme, “Collision avoidance and navigation for a quadrotor swarm using end-to-end deep reinforcement learning,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.
 - [23] Z. Liu, Y. Cao, J. Chen, and J. Li, “A hierarchical reinforcement learning algorithm based on attention mechanism for uav autonomous navigation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, 2023.
 - [24] W. Jiang, T. Cai, G. Xu, and Y. Wang, “Autonomous obstacle avoidance and target tracking of uav: Transformer for observation sequence in reinforcement learning,” *Knowledge-Based Systems*, vol. 290, 2024.
 - [25] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International conference on machine learning*. PMLR, 2018.
 - [26] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr., “Survey on coverage path planning with unmanned aerial vehicles,” *Drones*, vol. 3, no. 1, 2019.
 - [27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*. Pmlr, 2018.



DSPACE

<https://dspace.org/>

Temporal-augmented observation for navigation of unmanned aerial vehicles: a recurrent reinforcement learning architecture

Gemignani, Gabriele; Perrusquía, Adolfo; Tsourdos, Antonios; Pollini, Lorenzo

2026-06-15

Institute of Electrical and Electronics Engineers (IEEE)

<https://doi.org/10.1109/icuas69441.2026.11598652>

Gemignani G, Perrusquía A, Tsourdos A, Pollini L. (2026) Temporal-augmented observation for navigation of unmanned aerial vehicles: a recurrent reinforcement learning architecture.

In: Proceeding of the 2026 International Conference on Unmanned Aircraft Systems (ICUAS), 15-18 Jun 2026, Corfu, Greece, pp. 496-502