Reinforcement Learning for UAV Obstacle Avoidance

Dongxuan He
Department of Engineering
Purdue University
West Lafayette, IN, USA

I. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly used in inspection, delivery, and search-and-rescue missions. A key requirement for autonomy is navigating safely in partially known or changing environments. Classical pipelines combine global planners (e.g., A*/RRT) with tracking controllers (PID/MPC), but their performance degrades when maps are uncertain or obstacle layouts vary.

This project explores a model-free deep reinforcement learning (DRL) approach for *reactive* obstacle avoidance. Instead of computing explicit paths, a continuous-control policy is learned to map onboard observations to control actions. We formulate the task as a Markov Decision Process (MDP) and train in a 2-D simulation with randomized start–goal pairs and obstacle configurations. To contextualize performance, we additionally compare the learned policy against a simple classical baseline such as a potential-field controller.

To clarify the problem setting, the UAV is modeled as a 2-D double-integrator system with continuous acceleration commands. The agent receives only *local, partial observations*: range-sensor distances, goal-relative position, and velocity, resulting in a partially observable MDP (POMDP). Obstacle layouts are randomized each episode following a domain-randomization scheme, allowing the policy to generalize to unseen environments without retraining. A lightweight custom Gym-style environment is implemented for training and evaluation, and modern continuous-control DRL algorithms such as PPO and SAC are examined in this setting.

II. Background and Literature Review

A. Background

Autonomous navigation is essential for UAVs operating in dynamic or partially known environments. Beyond simply following a precomputed path, safe flight requires real-time obstacle avoidance when maps are inaccurate or the environment changes unexpectedly.

Traditional systems achieve this through a two-stage pipeline: global path planning methods such as A* or RRT determine a reference trajectory, while controllers like PID or MPC track the path. These methods are effective under well-modeled and static conditions but degrade when sensing is uncertain, dynamics differ from assumptions, or obstacles move—situations common in realistic missions.

To overcome these limitations, we explore a learning-based formulation that treats obstacle avoidance as sequential decision-making under uncertainty. By allowing a policy to be learned through environment interaction rather than manual design, reinforcement learning offers the potential for adaptive, map-independent navigation.

B. Literature Review

Traditional UAV navigation pipelines combine global path planning with model-based control. Comprehensive surveys such as Ghambari et al. [1] review graph-based (A*/Dijkstra), sampling-based (RRT/RRT*), and optimization-based techniques for UAV path planning. While effective in static or well-mapped scenes, these approaches degrade under partial observability or when obstacle layouts change between missions. Classical reactive strategies such as the artificial potential field (APF) method [2] offer real-time performance but suffer from local minima, a limitation extensively discussed in subsequent improvements to APF [3]. Model predictive control (MPC) frameworks have also been applied to quadrotor trajectory tracking (e.g., Mellinger [4]), but require accurate dynamics and incur high online computational cost.

Deep reinforcement learning (DRL) offers a model-free alternative by learning reactive control policies directly from interaction. Surveys such as Zhu et al. [5] and Le et al. [6] highlight the effectiveness of policy-gradient and actor-critic methods (e.g., PPO, DDPG, SAC) in mobile robot navigation under partial observability, where agents rely on local sensing rather than global maps. These works show that DRL-based controllers can avoid obstacles, generate smooth trajectories, and handle cluttered environments without explicit global planning.

A key challenge for DRL navigation is generalization to unseen environments. Domain randomization, introduced by Tobin et al. [7], randomizes environmental attributes during training to improve robustness and transferability. This strategy has been widely adopted in navigation and sim-to-real research, and directly motivates our use of randomized obstacle layouts and start–goal configurations to enhance policy generalization.

III. Problem Formulation

A. Environment

We use a lightweight custom Gym-style 2-D simulation environment implemented in Python to study UAV obstacle avoidance under controlled but representative conditions. The bounded workspace contains several static circular obstacles whose number and positions are randomized each episode, enabling diverse scenarios for training and evaluation. The environment provides a standard `reset`/`step` interface for continuous actions and includes shared geometric utilities such as collision checking and ray-based distance sensing. This design allows both DRL controllers (e.g., PPO/SAC) and classical baselines (e.g., potential-field or simple RRT* planners) to operate under identical dynamics, sensing, and obstacle layouts, ensuring a fair comparison.

B. Markov Decision Process Setup

We formulate UAV obstacle avoidance as a Markov Decision Process (MDP) in which the agent interacts with the environment to learn collision-free navigation behavior. Although the simulator maintains the full environment state, the policy receives only *local, partial observations* derived from onboard sensing, making the problem effectively a partially observable MDP (POMDP). The MDP is defined by the following components:

1) State Transition (Dynamics)
   The UAV is modeled as a 2-D double-integrator system, which captures essential inertial behavior while keeping dynamics simple for reinforcement learning:

   $$x_{t+1} = x_t + v_t \Delta t, \qquad v_{t+1} = \mathrm{clip}(v_t + a_t \Delta t,\, v_{\max}),$$

   where x_t and v_t denote position and velocity, a_t is the continuous acceleration command, Δt is the simulation timestep, and velocity is clipped to ensure feasible control. This explicitly addresses the dynamical model referenced by the instructor.
2) Observations
   Since global maps are unavailable, the agent must navigate using only local sensing:
   - $g_t = (x_g - x_t,\, y_g - y_t)$: goal-relative position.
   - $v_t = (v_x,\, v_y)$: current UAV velocity.
   - $d_t = [d_1, \ldots, d_N]$: normalized distances from N ray-based range sensors spanning a 180° field of view.

   This restricted observation space is the reason the problem is partially observable rather than fully observed.
3) Actions
   The action is a continuous acceleration command:

   $$a_t = (a_x,\, a_y), \qquad |a_i| \le a_{\max},$$

   representing planar thrust inputs compatible with PPO/SAC.
4) Reward
   The reward encourages efficient goal-reaching while penalizing unsafe or unstable behavior:
   - $r_t^{\text{prog}} = \alpha(\|g_t\| - \|g_{t+1}\|)$: progress toward the goal.
   - $r_t^{\text{goal}} = \kappa \mathbb{1}[\|g_{t+1}\| \le r_{\text{goal}}]$: goal reward.
   - $r_t^{\text{col}} = -\beta \mathbb{1}[\text{collision}]$: collision penalty.
   - $r_t^{\text{safe}} = -\beta_m \mathbb{1}[\min_i \|x_t - c_i\| \le r_i + m]$: safety-margin penalty.
   - $r_t^{\text{energy}} = -\lambda \|a_t\|^2$: energy penalty discouraging high thrust.
   - $r_t^{\text{smooth}} = -\mu \|a_t - a_{t-1}\|^2$: smoothness penalty discouraging abrupt control changes.
   - $r_t^{\text{time}} = -\eta$: per-step time penalty.

   The total reward is

   $$r_t = r_t^{\text{prog}} + r_t^{\text{goal}} + r_t^{\text{col}} + r_t^{\text{safe}} + r_t^{\text{energy}} + r_t^{\text{smooth}} + r_t^{\text{time}}.$$

   This separation of energy and smoothness directly addresses the instructor's comment that the two penalties represent different concepts.
5) **Objective**
   The goal of the agent is to learn a policy $\pi_\theta(a_t \mid s_t)$ that maximizes the expected cumulative discounted reward:

   $$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right], \tag{1}$$

   where $\gamma \in (0, 1)$ is the discount factor. Equivalently, the problem can be written as minimizing the cumulative discounted cost

   $$C = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t c_t\right], \tag{2}$$

   with $c_t = -r_t$. This formulation allows energy expenditure and smoothness penalties to be interpreted naturally as additive costs. The optimal policy π^* yields safe, smooth, and energy-efficient trajectories that reach the goal while avoiding obstacles.

IV. Experiments

A. Environment Setup

We evaluate all methods in a lightweight custom 2-D Gym-style environment. Each episode samples a new start–goal pair and a randomized set of static circular obstacles, following a domain-randomization scheme. This prevents overfitting to a fixed layout and ensures that the learned policy must generalize to unseen configurations without retraining.

The UAV follows a 2D double-integrator model with continuous acceleration commands. At each step the agent receives only local observations: goal-relative position, velocity, and N ray-based range measurements over a 180° field of view. No global map is provided, making the task partially observable.

All controllers including PPO and the artificial potential-field (APF) baseline operate in exactly the same environment and use the same collision checker and ray-based perception, ensuring a fair comparison under identical conditions.

B. Algorithms Evaluated

We compare our reinforcement-learning controller with a classical reactive baseline under identical sensing, dynamics, and obstacle layouts.

a) PPO (this paper).: We use Proximal Policy Optimization (PPO) to learn a continuous-control policy that outputs planar acceleration commands. The policy receives the goal-relative position, current velocity, and N ray-based distance measurements. PPO is chosen for its stability on continuous-action tasks and its ability to learn directly from partial observations without requiring a global map.

b) Artificial Potential Field (APF) baseline.: A classical artificial potential field controller is implemented for comparison. The goal provides an attractive potential, while each obstacle generates a repulsive potential within a finite influence radius. The controller follows the negative gradient of the combined potential field to produce an acceleration command. This method is simple and widely used in reactive navigation, but is known to suffer from local minima in cluttered environments.

Both controllers use the exact same environment, observations, collision checking, and dynamics, enabling a fair one-to-one comparison.

C. Training Procedure

The PPO agent is trained for 300k timesteps using the Stable-Baselines3 implementation with its default continuous-control hyperparameters (learning rate 3×10^{-4}, $\gamma = 0.99$, GAE $\lambda = 0.95$, minibatch size 64). Each training episode samples a new start–goal pair and a new randomized obstacle configuration, enabling the policy to generalize to changing environments without requiring retraining. This directly addresses the instructor's question regarding adaptation to moving or changing obstacles.

Training uses a simulation timestep of $\Delta t = 0.05\,\text{s}$ and a maximum episode length of 10 s. Every 20k timesteps, the current policy is evaluated on a fixed set of unseen layouts, and the final model is selected based on success rate. The APF baseline does not require training and is evaluated directly under the same conditions.

Reward Weights and Limits.: Unless otherwise noted, we use N=24 range rays over 180° FOV. Reward coefficients are: α=1.0 (progress), κ=10.0 (goal bonus), β=10.0 (collision), β_m=2.0 (safety margin), λ=0.005 (energy), μ=0.005 (smoothness), η=0.001 (time). Action/velocity limits are $|a_i| \le a_{\max}$=1.0 and $\|v\| \le v_{\max}$=1.0, with simulation step Δt=0.05 s. We set the goal tolerance to $r_{\text{goal}} = 0.2\,\text{m}$ and the safety margin to $m = 0.1\,\text{m}$ by default.

D. Evaluation Metrics

We evaluate each controller on a separate set of unseen randomized layouts. Performance is measured using the following metrics:

- Success rate: fraction of episodes in which the UAV reaches the goal without collision.
- Collision rate: fraction of episodes that terminate due to hitting an obstacle or leaving the workspace.
- Path length: total distance traveled before termination.
- Episode time: total flight time until success or failure.
- Energy per step: average squared acceleration magnitude, reflecting control effort.
- Smoothness (jerk) per step: average squared change in acceleration, measuring trajectory smoothness.

These metrics collectively measure safety, efficiency, and control quality, allowing a fair comparison between PPO and the potential-field baseline.

V. Results

This section presents the performance of the learned PPO policy during training and its final comparison against the classical artificial potential field (APF) baseline. We report learning curves over the 300k training steps as well as quantitative metrics evaluated on 200 unseen randomized environments.

A. Learning Curves

Fig. 1 shows the evolution of success rate as training progresses. The policy initially performs poorly due to random exploration, but after approximately 80k–100k steps the success rate increases rapidly, eventually stabilizing around 70–75%. A similar trend appears in Fig. 2, where the reward proxy (success minus collision rate) improves consistently, confirming that the agent learns both to reach goals and avoid unsafe maneuvers.

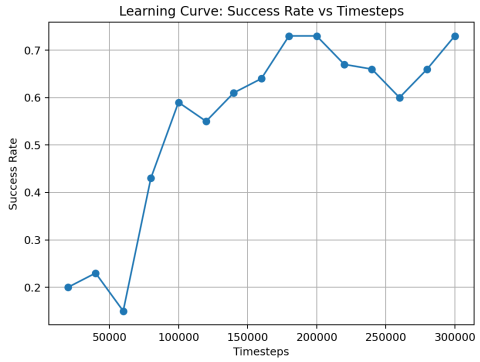

Fig. 1. Learning curve: success rate versus training timesteps.

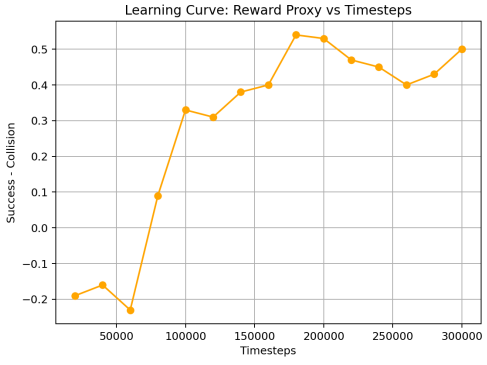

Fig. 2. Learning curve: reward proxy (success–collision) during training.

B. Final Policy Evaluation

After training, we evaluate the PPO policy and APF controller under identical randomized test scenarios. Fig. 3 and Fig. 4 visualize the success and collision rates. PPO achieves substantially higher reliability, reaching 76% success on unseen layouts, whereas APF fails to complete most episodes (0% success) and collides frequently.

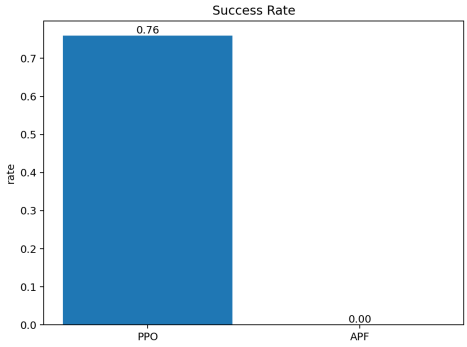

Fig. 3. Success rate comparison between PPO and APF.

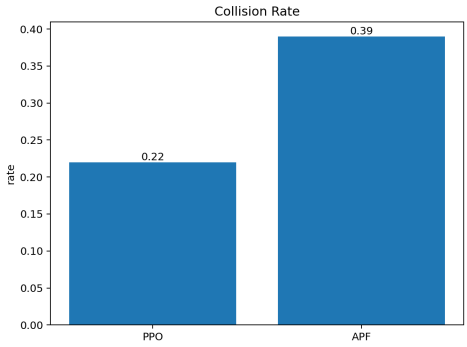

Fig. 4. Collision rate comparison between PPO and APF.

Beyond safety metrics, PPO also produces smoother and more energy-efficient trajectories (Fig. 5 and Fig. 6). This aligns with the reward structure that explicitly penalizes jerk and control effort.

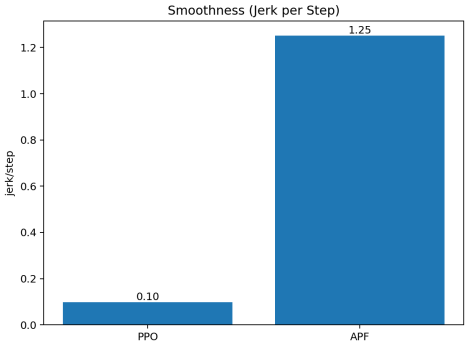

Fig. 5. Smoothness comparison (jerk per step).

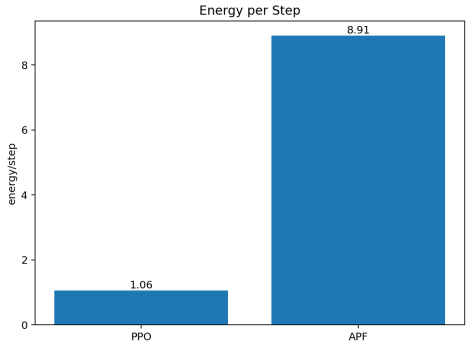

Fig. 6. Energy consumption per step.

C. Summary

Table I summarizes all metrics. PPO outperforms APF on every dimension except raw path length, where APF is slightly shorter but also far less safe and significantly less smooth.

TABLE I
PERFORMANCE COMPARISON ON TEST ENVIRONMENTS.

Metric	PPO	APF
Success Rate ↑	0.76	0.00
Collision Rate ↓	0.22	0.39
Avg Path Length (m) ↓	5.02	4.75
Energy per Step ↓	1.06	8.91
Smoothness (jerk/step) ↓	0.10	1.25

Overall, the PPO policy demonstrates strong generalization to randomized obstacle layouts and substantial improvements in safety, smoothness, and energy usage compared to a classical potential-field controller.

VI. Discussion

The PPO-based controller significantly outperformed the classical artificial potential field (APF) method across all metrics. It achieved a 76% success rate with fewer collisions and far smoother, more energy-efficient trajectories. This improvement stems from PPO's ability to integrate sequential observations and adapt to randomized layouts, effectively overcoming the local minima problem inherent to APF. The learning curves further indicate stable convergence after around 100k timesteps, confirming that domain randomization and carefully shaped rewards guide effective policy learning.

Nonetheless, the approach has limitations. Performance degrades in densely cluttered or narrow environments, where the partial 180° range-sensor observation limits spatial awareness. Reward tuning was also sensitive and required iterative balancing between goal progress, safety, and smoothness terms. Future work will focus on extending the framework to dynamic obstacles and real-world transfer, where robustness to noise and generalization beyond simulation remain open challenges.

VII. Conclusion

This project investigated a model-free deep reinforcement learning approach for UAV obstacle avoidance under partial

observability. Using domain randomization and a carefully shaped reward, the PPO policy learned to navigate cluttered environments with a 76% success rate, substantially outperforming a classical potential-field baseline in both safety and control smoothness. While the method shows strong generalization to unseen static layouts, challenges remain in handling highly cluttered scenes, dynamic obstacles, and real-world deployment. Future work will focus on extending the observation model, incorporating temporal memory, and evaluating the policy in more realistic settings.

REFERENCES

[1] S. Ghambari, A. Mousavi, and M. N. Ahmadi, "UAV path planning techniques: a survey," *RAIRO – Operations Research*, 2024.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[3] M. Guerra, M. Olfati-Saber, and J. Cortés, "Avoiding local minima in the potential field method using input-to-state stability," *Control Engineering Practice*, vol. 52, pp. 16–26, 2016.

[4] D. Mellinger, *Trajectory Generation and Control for Quadrotors*, PhD dissertation, University of Pennsylvania, 2012.

[5] K. Zhu, Z. Wang, and Z. Li, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[6] H. Le, T. Truong, and Q. Nguyen, "A comprehensive review of mobile robot navigation using deep reinforcement learning in crowded environments," *Journal of Intelligent & Robotic Systems*, 2024.

[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

AI USAGE CHECKLIST (YES/NO)

1) Used AI tools for code generation, debugging, or visualization — **YES**
2) Used AI tools for literature search, summarization, or concept clarification — **YES**
3) Used AI tools for language editing, grammar correction, or report polishing — **NO**
4) Used AI tools to formulate problems, generate algorithms, or develop proofs / derivations — **NO**
5) Used AI tools for data analysis, plotting, or simulation assistance — **NO**
6) Verified all AI-generated code, text, results, and references for correctness and validity — **YES**
7) Can fully explain all methods, algorithms, and results presented in this report — **YES**
8) The intellectual contribution, including understanding, reasoning, and application of course concepts, is my own — **YES**

QUALITATIVE SELF EVALUATION

Beyond the technical results, this project was a valuable learning experience that helped me understand what it actually means to conduct reinforcement learning research. I chose this topic because UAV navigation lies at the intersection of control, perception, and decision-making. This is the area I have always found intellectually appealing, yet had never fully explored through a complete end-to-end system. Working with PPO in a custom 2D UAV simulator let me dive into the practical challenges behind RL that go far beyond the textbook description, including reward shaping, environment design, training instability, and hyperparameter sensitivity.

Through repeated debugging and experimentation, I gained a clearer sense of how an RL agent acquires behavior gradually, how exploration interacts with safety constraints, and why even small modeling decisions can dramatically affect final performance. I also learned to interpret learning curves, verify baselines, and evaluate policies across success, collision, smoothness, and energy metrics — skills that feel much closer to real research than simple coding assignments. Overall, this project strengthened my intuition about RL systems and gave me confidence in reading, implementing, and critically evaluating modern reinforcement learning methods. It also taught me the importance of structured experimentation and careful analysis, lessons that I will carry into future coursework and research.