The Aeronautical Journal (2021), 1–18
doi:10.1017/aer.2021.85

REGULAR PAPER

Reinforcement learning-based radar-evasive path planning: a comparative analysis

R.U. Hameed, A. Maqsood, A.J. Hashmi, M.T. Saeed, and R. Riaz

Research Centre for Modeling & Simulation, National University of Sciences & Technology, Islamabad, Pakistan
E-mail: adnan@rcms.nust.edu.pk

Received: 21 February 2021; **Revised:** 31 July 2021; **Accepted:** 6 September 2021

Keywords: Path planning; Reinforcement learning; Radar avoidance

Abstract
This paper discusses the utilisation of deep reinforcement learning algorithms to obtain optimal paths for an aircraft to avoid or minimise radar detection and tracking. A modular approach is adopted to formulate the problem, including the aircraft kinematics model, aircraft radar cross-section model and radar tracking model. A virtual environment is designed for single and multiple radar cases to obtain optimal paths. The optimal trajectories are generated through deep reinforcement learning in this study. Specifically, three algorithms, namely deep deterministic policy gradient, trust region policy optimisation and proximal policy optimisation, are used to find optimal paths for five test cases. The comparison is carried out based on six performance indicators. The investigation proves the importance of these reinforcement learning algorithms in optimal path planning. The results indicate that the proximal policy optimisation approach performed better for optimal paths in general.

Nomenclature

(x, y)	two-dimensional coordinates of an aerial vehicle
v	velocity
u	acceleration normal to flight path vector
U	maximum allowable lateral acceleration
z	constant height
g	acceleration due to the gravity
P_r	power reflected
N_{avg}	average noise power
$\frac{S}{N}$	signal-to-noise ratio (SNR)
P_{avg}	average power transmitted by radar
t_s	scan time for ω
A_e	antenna area
T_s	system noise temperature
R	distance of target from radar (radar range)
K	Boltzmann's constant
L	total system loss
$Q(s, a)$	action-value function
$Q(s_i, a_i)$	action-value function for iteration i
A	action space
S	state space
A_i	action take at iteration i
S_i	state vector at iteration i
a	all actions in action space

s	all states in state space
R_i	scalar reward received at iteration i
k	number of training periods

Greek symbol

α	learning rate
γ	discount factor
ϵ	clipping parameter
η	long-term reward
θ_k	policy at step k
ϑ	azimuth angle
λ	aspect angle
ξ	bank angle
σ	radar cross-section (RCS)
τ	target networks update rate
ϕ_k	value function at step k
φ	elevation angle
ψ	heading angle
ω	solid angle searched by radar

Abbreviations

RL	reinforcement learning
DRL	deep reinforcement learning
DDPG	deep deterministic policy gradient
GAE	generalised advantage estimation
TRPO	trust region policy optimisation
PPO	proximal policy optimisation
UAV	unmanned aerial vehicle
SGD	stochastic gradient descent
RCS	radar cross-section
SNR	signal-to-noise ratio

1.0 Introduction

Aircraft path planning is entering a new era with the emergence of artificial intelligence-based algorithms. This paper explores the usage of contemporary reinforcement learning-based algorithms in generating optimal paths either to avoid radar detection or to reduce exposure time. The existing techniques used in path planning generally include spline-based path planning, non-linear programming, various deterministic techniques and differential dynamic programming [1–3]. These techniques are efficient for distinct purposes to generate optimal paths. Similarly, trajectory generation for versatile manoeuvres such as hover-to-cruise [4], perching [5] and dynamic soaring [6] of Unmanned Aerial Vehicles (UAVs), typically done through established deterministic approaches, however, are susceptible to the initial guess. Recently, machine learning algorithms have been explored for different trajectory optimisation and path planning problems [7] to help generate trajectories that are insensitive to initial conditions. This shift is primarily attributed to their robustness in handling complex constraints and finding near-optimal solutions.

A classical approach for reducing radar exposure is to decrease the Radar Cross-Section (RCS) of the aircraft. Some of the methods to reduce aircraft RCS include the usage of specialised composite materials [8], paints that can absorb radar waves [9] and geometric alteration of the aircraft [10]. It is also reported that, by performing different manoeuvres, the radar exposure time can be reduced significantly [11]. Optimal path planning is considered a practical approach to minimise radar exposure. In a seminal study, Pachter and Hebert [12] used the calculus of variations technique to minimise radar exposure by minimising the energy reflected from a target. The method generated an optimal solution but was analytical, and its implementation on real aircraft and UAVs was deemed less practical. Moore [13] also carried out an interesting study of route planning to reduce the active RCC through control of bank and yaw angle. Similarly, Norsell [14] suggested optimising the trajectory by minimising the flight time between given positions and the exposure to radar. For this purpose, spline approximations were used to generate a smooth curve as a flight path, and the radar detection range was calculated from the aircraft position, heading and bank angles.

One of the most notable studies in optimal path planning of stealth aircraft in the presence of missiles was carried out by Kabamba et al. [15, 16]. This research used three distinct modules for problem formulation: aircraft, radar and missiles. The optimal control problem was formulated through aircraft lateral acceleration. Optimality conditions necessary for this problem were derived mathematically and considered to provide an efficient numerical solution to handle further complexities. Using this combination of aircraft, RCS, radar and missile models, optimal paths for aerial vehicles that could be used for practical problems by feeding in relevant/accurate data were generated. The model is used to generate paths for different numbers of radars under missile impact threats. The use of intelligent algorithms to solve similar problems may improve the performance and provide much better paths with minimum radar exposure, which is the niche examined herein.

Inclusion of machine learning techniques for path planning of aerospace vehicles has spurred a multidisciplinary initiative. Recently, autonomous agents have been trained for different purposes through deep learning techniques. Reinforcement learning, a type of machine learning, is a well-established framework to solve complex control problems and optimal path planning issues [17]. Recently, RL techniques have been applied to varied aerospace problems, such as fixed-wing UAV flocking problems [18], avoidance of flying obstacles [19] and surveillance by UAV swarm [20]. Merging the techniques of deep learning with reinforcement learning gave birth to deep reinforcement learning (DRL), which is now commonly used to solve control problems for continuous states and space [21, 22]. DRL can help to overcome the inherent uncertainties present in the model and decipher data from noisy sensors. DRL has shown promising results for UAVs performing different actions and tasks [23].

DRL has attracted significant attraction from the application-focused scientific community, including for aircraft dynamics, control and trajectory optimisation. Santos et al. [24] used DRL for attitude stabilisation and path tracking through learning automata, a stochastic-based reinforcement learning algorithm. Attitude and height parameters were tuned through this framework, and an optimal path to control the aerial vehicle was generated. This provided impressive stabilisation for UAVs in different environments, but discrete actions were used in this method for stabilisation and control.

Deterministic policy gradient algorithms provide a framework to deal with continuous actions and state space. These gradient-based algorithms can provide more effective estimation as compared with stochastic algorithms [25]. Interest from the application-focused scientific community in the discipline of RL was aroused when AlphaGo, an AI trained agent, defeated the world's best Go game players [26]. The algorithm was trained using a deep Q-network (DQN), a DRL algorithm developed by merging DL and RL techniques. The deep deterministic policy gradient (DDPG) algorithm by Lillicrap et al. [27] is probably a better choice for solving complex control problems such as radar avoidance path planning, where aircraft must avoid radar signals to minimise their radar exposure in continuous action and state spaces. To study the effectiveness of DDPG in complex environments, a comparison study was recently conducted between DDPG and the Recurrent Deterministic Policy Gradient (RDPG) technique [28]. Path planning with DDPG was fast as compared with RDPG. Note that the sensor noise in a natural environment is a big challenge. However, the effectiveness of the DDPG algorithm in path planning was established, thus providing a lead for the use of this algorithm.

Schulman et al. [29] proposed another attractive policy gradient-based algorithm in 2015 and named it Trust Region Policy Optimisation (TRPO). This algorithm showed robust performance on several different tasks. By adjusting hyperparameters, TRPO can provide improvements in a variety of tasks, regardless of approximations. TRPO was aimed to handle high-dimensional problems in a continuous

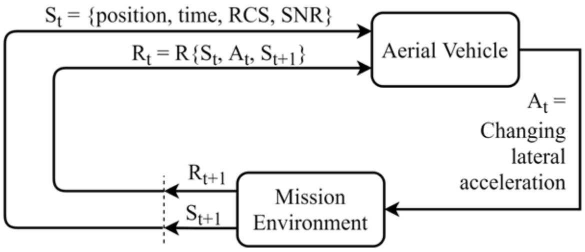

S_t = {position, time, RCS, SNR}
$R_t = R\{S_t, A_t, S_{t+1}\}$
Aerial Vehicle
A_t = Changing lateral acceleration
R_{t+1}
S_{t+1}
Mission Environment

Figure 1. *The RL framework for path planning.*

state. For aerial vehicles, Hwangbo et al. [30] used TRPO with gradient descent to control a quad-rotor. The results demonstrated the robustness of the approach even under harsh initialisation conditions.

Proximal Policy Optimisation (PPO), a much-refined version of TRPO, is known for its simplicity and ease of implementation [31]. PPO outperformed all the policy gradient methods and has shown surprising results for different problems. Heess et al. [32] and Lopes et al. [33] demonstrated effective PPO implementations for the stabilisation and control of a quad-rotor. For fixed-wing UAVs, Bohn et al. [34] used a PPO-based controller for efficient attitude control. The results, in comparison with the Proportional–Integral–Differential (PID) controller, were less sensitive to initial conditions and disturbances. Finally, Koch et al. [35] carried out a comparative study between RL algorithms and PID controllers for the problem of stability and control of UAVs. DDPG, PPO and TRPO were chosen from among the RL algorithms to train in a simulated environment. The results of these algorithms were compared with those obtained using the PID controller. The findings illustrated the enhanced performance of the RL algorithms in terms of time efficiency and accuracy and the desired stabilisation and control performance.

In this research, the common problem of radar evasion is selected to explore the efficacy of DRL algorithms in generating optimal paths. A modular approach is adopted, including three distinct segments: the aircraft kinematic model, the RCS model and the radar tracking model. An OpenAI Gym-based environment is created to train the DRL agents. Three DRL algorithms, viz. DDPG, TRPO and PPO, are used to compare their optimal solutions for path planning problems in five different scenarios. The results explore the efficacy and limitation of each algorithm for radar-evasive optimal path planning.

This section presents an introduction and review. Section 2 provides a brief review of deep reinforcement learning and the algorithms used in this research. In Section 3, the aircraft, RCS and radar models are discussed, along with the OpenAI gym model. Subsequently, the results and discussion section includes a comparison of the three algorithms for the five different cases. Conclusions are drawn in the final section.

2.0 Deep reinforcement learning

Reinforcement learning agents interact with the environment over time to maximise a reward [17]. An initial state is given to the agent. The agent performs an action A_t in the environment and receives the next state S_{t+1} and the scalar reward R_{t+1} for time t by following a specific policy. Based on the action taken, received reward and next state, the agent performs the next action in the environment and continues to update the policy to maximise its performance. A simple RL framework for the optimal path planning problem is shown in Fig. 1.

RL is implemented using either value- or policy-based algorithms. For value-based algorithms, RL depends upon the Q-function approximation. Q-learning and DQN are value-based algorithms. Policy parameterisation is performed for policy-based algorithms. For both cases, the use of these substantial approximators allows RL to handle complicated problems. Neural networks are widely used as approximators in applications of supervised learning. We decided to use three DRL algorithms named DDPG, TRPO and PPO to solve our problem. DDPG [27] is a policy-based algorithm that operates on the actor–critic framework. It is an extension of the DQN algorithm for continuous state and action pairs. TRPO [29] belongs to the on-policy family of RL algorithms and is used as a trust region constraint. TRPO makes use of KL divergence to ensure that the policy is not shifting much away from initial points. PPO [31] also belongs to the on-policy algorithms. It is an improved version of TRPO, ensuring that the new policy is close to the old policy while utilising only first-order optimisation techniques. PPO is much simpler to implement and shows monotonic results for various applications compared with other RL algorithms. Both PPO and TRPO use the advantage function, while DDPG makes use of the Q-value function.

Several reasons have made RL more popular than conventional optimal control techniques in the trajectory optimisation community. Both RL and traditional optimisation techniques can be used to achieve optimal control for a specific problem. The line between reinforcement learning approaches and traditional optimisation techniques can be blurry depending on the problem. The significant difference is that traditional optimisation approaches focus on finding values that maximise or minimise some objective function to achieve optimal control based on the problem. On the other hand, in reinforcement learning, an agent interacts with the environment and received some reward or penalty based on the action. The agent interacts with and explores the environment to achieve the maximum cumulative reward so that an optimal solution can be achieved based on the problem.

In traditional optimisation techniques, the system is said to be optimised if a local or global optimum is found, while the gradient field is theoretically zero. Meanwhile, in reinforcement learning, agents find the right actions to take in the environment with a time-based reward and penalty to achieve the maximum reward. As the problem becomes more complex, traditional optimisation techniques can become stuck in local optima and may not achieve optimal control. On the other hand, reinforcement learning is more successful for finding optimal control for complex control problems. Also, in traditional techniques, controllers learn offline. They are then fixed with the same solution, while in reinforcement learning, the agent can improve itself with time by interacting more and more with the environment.

The robustness of reinforcement learning in a complex dynamic environment, finding an optimal solution based on maximum reward and iterative learning, and improvement over time are the significant reasons to choose the optimal solution.

In this research, the optimal path planning problem is solved using DDPG [27], TRPO [29] and PPO [31]. The detailed mathematical formulation and performance of each algorithm are provided in the cited studies. The function approximation technique used in DDPG is not efficient as it sometimes fails to handle a much simple environment and may not provide effective and monotonic results. It also offers much lesser data efficiency and is not much flexible for different types of problems. TRPO has higher computation cost and is highly complex. Moreover, it is not amenable to noisy systems. Meanwhile, PPO is much simpler to implement, and its computation cost is significantly lower due to the first-order optimiser. It adds a soft restraint to the objective function, which can be easily handled without significantly affecting the policy. It may initially show some bad choices, but it tends to handle problems quite well over time by providing a good balance against optimisation speed. Theoretically, it is evident that PPO shows the best results, but it is also experimentally proven that PPO shows monotonic results while having a very simple implementation [35].

3.0 Modeling and problem formulation

The radar-evasion path planning problem is modelled using three sub-models for the aircraft kinematics, the RCS and the radar. The path planning is modelled in a two-dimensional planar space. An OpenAI Gym-based simulated environment is used to implement the DRL techniques.

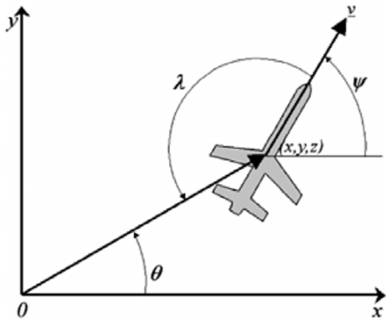


Figure 2. *Aircraft position, velocity, azimuth, heading and aspect angles [16].*

3.1 Aircraft model

A basic aircraft kinematic model in planar space is used. It is assumed that the aircraft will move with constant speed in the horizontal plane at a constant height z. The following equation presents the aircraft model in the horizontal plane [16]:

$$\dot{x} = v \cos \psi \tag{1}$$

$$\dot{y} = v \sin \psi \tag{2}$$

$$\dot{\psi} = u/v \tag{3}$$

where x and y are Cartesian coordinates, ψ is the heading angle as shown in Fig. 2, v is the constant velocity and u is the control input. The control input is basically the acceleration normal to the flight path vector [15]. note that $|u| < U$, where U represents the maximal permissible lateral acceleration.

3.2 RCS model

The RCS of the aircraft depends upon its size, shape, material used, direction and orientation. In this study, the RCS of an ellipsoid is considered. It is calculated from the combination of the aspect angle, elevation angle and bank angle [16]. The RCS is a function of these angles and calculated as

$$\vartheta = \arctan(y/x) \tag{4}$$

$$\lambda = \vartheta - \psi + \pi \tag{5}$$

$$\varphi = \arctan(z/\sqrt{x^2 + y^2}) \tag{6}$$

$$\xi = \arctan(u/g) \tag{7}$$

$$RCS = \sigma(\lambda, \varphi, \xi) \tag{8}$$

where λ is the aspect angle, φ is the elevation angle and ξ is the bank angle. ϑ is the azimuth angle, and z is the aircraft altitude (considered constant in this problem(. u is the input signal that is being used as a control signal to solve this problem, and g is the acceleration due to gravity.

In its simplified form, the RCS model of the aircraft has similar effects compared with the ellipsoid model. It does not represent any specific aircraft but captures three eminent attributes useful for aircraft RCS: the larger beam aspect RCS, the relatively smaller RCS from the front face and the higher RCS from the top and below perspective. By considering these characteristics, an ellipsoid model can actually represent the aircraft RCS if the principal semi-axes of the ellipsoid are handled carefully [36]. The model is represented as

$$\sigma(\lambda, \varphi, \xi) = \frac{\pi a^2 b^2 c^2}{(a^2 \sin^2 \lambda_a \cos^2 \xi_a + b^2 \sin^2 \lambda_a \sin^2 \xi_a + c^2 \cos^2 \lambda_a)^2} \tag{9}$$

where λ_a and ξ_a are considered to be

$$\lambda_a = \arccos\left[\cos(\varphi)\cos(\lambda)\right] \tag{10}$$

$$\xi_a = \xi - \left[\frac{\tan(\varphi)}{\sin(\lambda)}\right] \tag{11}$$

where a, b and c are the principal semi-axes. The values of these axes must be adjusted carefully so that the ellipsoid model can represent the aircraft RCS properly.

3.3 Radar model

The radar equation combines three important characteristics that include properties of the radar and target along with signal characteristics. The RCS is an important target property that can be combined with the radar characteristics, transmitted power and antenna area to calculate the reflected radar power. To track an aircraft using radar, the signal-to-noise ratio (SNR) must be calculated. To calculate the SNR, the signal power reflected by the target and received at the radar is calculated as [37, 38]

$$P_r = \frac{P_t \times G_t \times \sigma(\lambda, \varphi, \xi) \times A_e}{16\pi^2 R^4} \tag{12}$$

where P_t is the peak transmitter power and G_t is the transmit gain calculated as $4\pi A_e/(\lambda_w)^2$. λ_w is the wavelength, A_e is the antenna area and R is the radar range. The average noise power for radar can be calculated as

$$N_{\text{avg}} = K \times T_s \times B_n \tag{13}$$

where K is Boltzmann's constant, T_s is the system noise temperature and B_n is the noise bandwidth of the receiver. In this paper, the problem of a search radar is formulated, being used to determine the unknown location of a target. The signal-to-noise ratio (SNR) for the search radar is calculated as

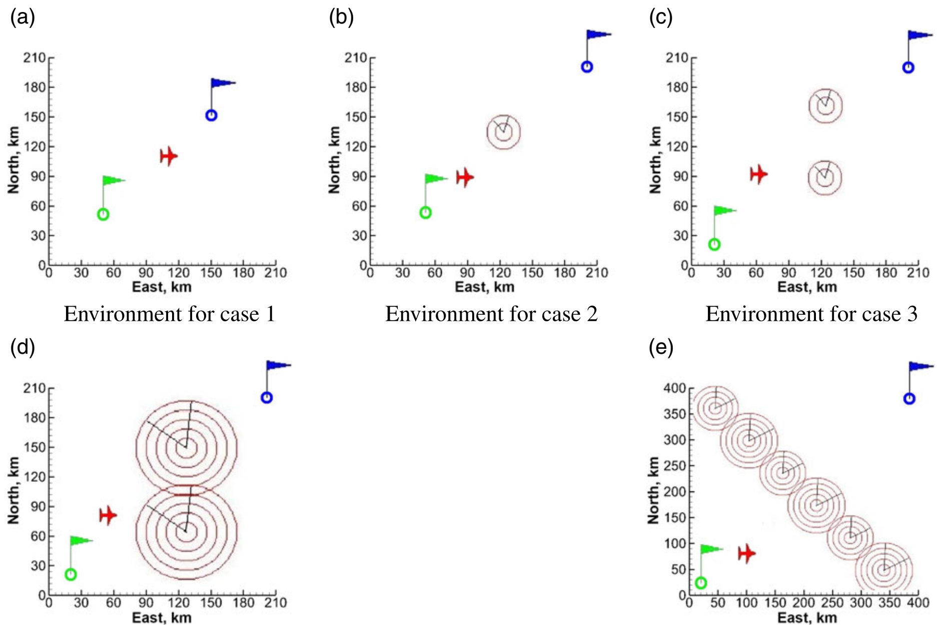
$$\frac{S}{N} = \frac{P_{\text{avg}} A_e t_s \sigma}{4\pi \omega R^4 K T_s L} \tag{14}$$

Here, the average power channelled by the radar is denoted as P_{avg}, ω is the solid angle searched by the radar and t_s is the scan time for ω. A_e represents the antenna area, T_s is the system noise temperature and R is the radar range. σ is the RCS calculated using the RCS model described in Section 3.2, K is Boltzmann's constant and L is the total system loss. After calculating the SNR for the target, the probability of detection is calculated based on the exposure time, RCS and range, thus providing a reward or penalty for the DRL agent during training.

3.4 The OpenAI Gym environment

OpenAI is designed to expand the credibility of the AI field, provide uniformity and establish better benchmarks by offering multifaceted environments with much easier setup methods. Similarly, the Gym toolkit is designed for the development and comparison of RL algorithms. The Gym contains versatile environments with both discrete and continuous states and action pairs, using different DRL algorithms. These are compatible with Keras, TensorFlow and PyTorch. In addition to already provided environments, customisation is also possible by following the general scheme of Gym environments.

In this research, the aircraft kinematics, RCS and radar models are implemented in the Gym simulation environment as continuous state and action space. The initial states and range of continuous actions



Environment for case 1 — Environment for case 2 — Environment for case 3 — Environment for case 4 — Environment for case 5

Figure 3. *OpenAI Gym environment representation.*

are described in the environment, along with the reward and step functions. Five different simulated environments are designed as optimal path planning cases.

In this simulated environment, a 400 × 400 pixel rectangle is taken. In Fig. 3, the green flag, blue flag, red circles and red aircraft represent the aerial vehicle starting position, target position, radars and aerial vehicle body, respectively. The initial and target positions are known a priori, whereas the radar positions are unknown to the DRL agent. The pixel to pixel distance is equated to 1,000m.

For case 1, as shown in Fig. 3(a), the starting point is (50, 50) while the target is at (150, 150). In this case, the aerial vehicle is supposed to reach the target in the absence of radar. The obvious optimal path will be the shortest distance. Figure 3(b) shows case 2, where the starting position is (50, 50) and the target position is (200, 200). A radar is placed at (120, 130) with a range of 20×10^3m. In cases 3 and 4, the aerial vehicle has to avoid two radars. The starting position in both cases is (20, 20), while the target position is (200, 200). In case 3,rRadars are located at (120, 90) and (120, 160), respectively, with range of 20×10^3m, as shown in Fig. 3(c). In case 4, radars are positioned at (120, 60) and (120, 150) with range of 50×10^3m, as shown in Fig. 3(d). In case 5, six radars are added in a diagonal to cover the maximum area and to offer a small window for the aerial vehicle to cross with minimum exposure, as shown in Fig. 3(e). For case 5, the starting point is (20, 20) while the target is located at (380, 380). The radars are located at (340, 40), (280, 100), (220, 160), (160, 220), (100, 280) and (40, 340). The first, third and fifth radar have a range of 50×10^3m, while the others have a range of 40×10^3m. The objective in all cases is to reach the target in the shortest possible time and with minimum radar exposure.

For all cases, the position of the radars is unknown to the agent (aerial vehicle). For a certain pixel location of the aerial vehicle, the RCS and SNR are calculated. A high SNR value indicates an increased probability of being tracked. As a result, the agent receives a negative penalty based on its action, due to which the agent will learn to avoid this position. The maximum reward for reaching the target R_{tar} is 100. There is a time-based penalty for entering the range of a radar. If the agent remains in the range of a radar for longer, the penalty will keep increasing as the probability of tracking will increase. There is

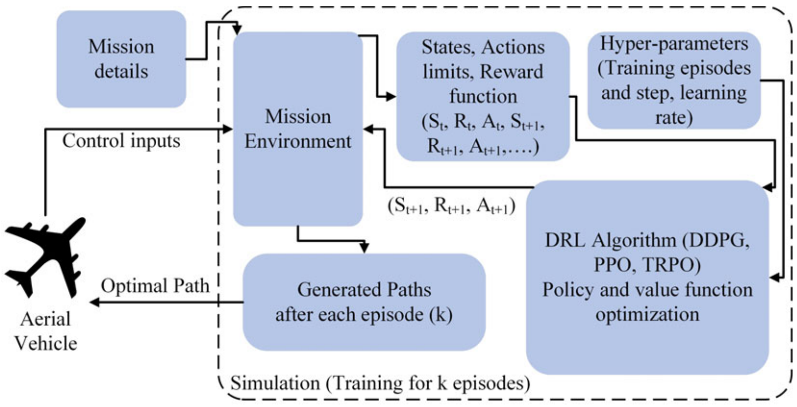

Mission details
Control inputs
Mission Environment
States, Actions limits, Reward function (S_t, R_t, A_t, S_{t+1}, R_{t+1}, A_{t+1},....)
Hyper-parameters (Training episodes and step, learning rate)
(S_{t+1}, R_{t+1}, A_{t+1})
DRL Algorithm (DDPG, PPO, TRPO) Policy and value function optimization
Optimal Path
Aerial Vehicle
Generated Paths after each episode (k)
Simulation (Training for k episodes)

Figure 4. *Optimal path planning architecture.*

also a negative penalty for each step if the agent cannot reach the target position, to minimise the time and distance. The total reward can be expressed as

$$R_i = R_{\text{tar}} + R_{\text{step}} + R_{\text{track}} \tag{15}$$

where R_i is the total reward for iteration i. R_{tar} is the scalar reward for reaching the target, R_{step} is a small reward for each step based on the action taken by the agent and R_{track} is the negative reward for radar tracking, which will keep increasing as the radar exposure time increases. These rewards can be expressed by the following equations:

$$R_{\text{tar}} = 100 \tag{16}$$

$$R_{\text{step}} = -(action_i^2 \times 0.5) \tag{17}$$

$$R_{\text{track}} = -(action_i^2 \times 0.5(Range) + (SNR) \times 0.07t) \tag{18}$$

Here the range is the distance from the radar to the aerial vehicle, SNR is the signal-to-noise ratio at a specific iteration of time and t is the time for which the aerial vehicle is in radar range. This equation has been designed through iterative refinements to serve our objectives of minimum radar observance and shortest distance travelled effectively.

The algorithms are now implemented for optimal path planning of aerial vehicles. Initially, aerial vehicle control inputs along with mission details are provided to generate the mission environment. This environment contains the states, actions and reward details, which are then transferred to the DRL algorithms and the hyperparameters used to train the DRL agents. The DRL algorithm provides information regarding the next state, action and reward to the mission environment and updates its policy and value function by comparing it with previous information. This procedure continues until a complete episode is terminated and a path is generated for that specific episode. Subsequently, the next episode starts, and DRL agents keep updating their policy to maximise the future reward. After completing the training, the optimal path generated by the DRL algorithms is provided to the aerial vehicle to complete the mission efficiently. The complete architecture for optimal path planning is shown in Fig. 4.

4.0 Results and discussion

This section compares the DDPG, TRPO and PPO algorithms in the OpenAI Gym environment for the problem under discussion. The environment includes the five different cases discussed in Section 3.4. For each case, a comparison of the performance of the algorithms in terms of the reward and steps taken is described. The optimal paths obtained using each algorithm after training are compared for each case separately.

The speed and altitude of the aircraft are kept constant at $v = 600$m/s and $z = 8,000$m, respectively, for all cases. Similarly, a maximum permissible lateral acceleration of $U = 78$m/s is added as a constraint to handle turns effectively. For the RCS model, the values of the principal semi-axes a, b and c are taken as 0.3172, 0.1784 and 1.003, respectively, so that the ellipsoid can represent the aircraft attributes efficiently for better calculation of the RCS. For the radar model, we use $P_{\text{avg}} = 0.5$MW, an antenna width of 4.9m and an antenna height of 2.7m to calculate the antenna area A_e. The solid angle searched by the radar is $\omega = 360$deg, and the scan time t_s for the solid angle searched is 30s. The system noise temperature $T_s = 500^{\circ}$ C and total system loss L are considered to be 10dB.

For DDPG, two hidden layers for both the actor and critic network are used, with 400 and 300 units in each layer. The actor learning rate is 10^{-3}, and the critic learning rate is 10^{-4}. The target update rate τ is 0.001 with a discount factor γ of 0.99. The experience replay buffer size is 10^6 with a mini-batch size of 64. For the actor network, the *relu* rectifier is used for all hidden layers except the output layer with a *tanh* layer. For the critic network, all the hidden layers use a *relu* rectifier. The Adam optimiser is used for gradient descent in both networks. The values of these hyperparameters are the same for all the cases.

For TRPO, the discount factor γ is 0.99, and the KL-divergence limit δ is 0.01 for higher stability. The learning rate for the value function is 0.001, the backtracking coefficient α is 0.8, the damping coefficient is 0.1 and the λ_{GAE} is 0.97. The number of backtracking iterations and conjugate gradient iterations is ten, to maintain the balance of performance and speed of convergence. These hyperparameters for TRPO are the same for all cases.

For PPO, the discount factor γ is 0.99 and the clipping ratio ϵ is 0.2. The learning rate for the value function is 0.001, the learning rate for policy is 0.0003, λ_{GAE} is 0.97 and the batch size is 64. The number of gradient descent steps to take on the policy loss and value function per epoch is 80, and the value of the target KL divergence is 0.01. The hyperparameters of PPO are kept the same for all cases.

The computational resource used to train algorithms is an Intel(R) Core(TM) i7-9750H CPU with a 2.60GHz processor. The system includes 12 logical processors with 12MB cache and 24GB RAM. The software was executed on a Kingston A2000 NVMe SSD and with an Nvidia GTX 1650 4GB graphics card.

4.1 Case 1:

For case 1, the goal is to reach the target while following the shortest path. All three algorithms are trained to obtain optimal paths as shown in Fig. 5. For this case, the DDPG agent initially was not able to reach the target for almost 100 episodes, as shown in Fig. 5(a). Throughout the whole training period, DDPG could not stabilise the episodic steps to achieve an optimal solution with the shortest path. This can be attributed to the issue of the lower sample efficiency of the DDPG algorithm. DDPG utilises the function approximation technique, which is not very efficient for giving monotonic results. The path obtained from DDPG training is not accurate as it shows chatter in the path and cannot manage to generate a smooth path without staggers, as demonstrated in Fig. 5(d). This can be addressed by applying some curve-fitting techniques to identify a more appropriate path to be followed.

Interestingly, TRPO converged rapidly in the initial 200 episodes, as shown in Fig. 5(b), but the path obtained from TRPO could be improved in terms of distance, as shown in Fig. 5(e). Due to the constraint used in TRPO, it did not try to find an optimal policy, which resulted in the construction of a less optimal path.

Finally, PPO gave optimum results for this case, achieving a balance between convergence speed and policy optimisation. It took more convergence time as compared with TRPO but was able to optimise the policy more efficiently, which is much more preferred. Figure 5(c) shows the PPO convergence during training. The path obtained from PPO is the shortest one in Fig. 5(f). Therefore, PPO was superior to

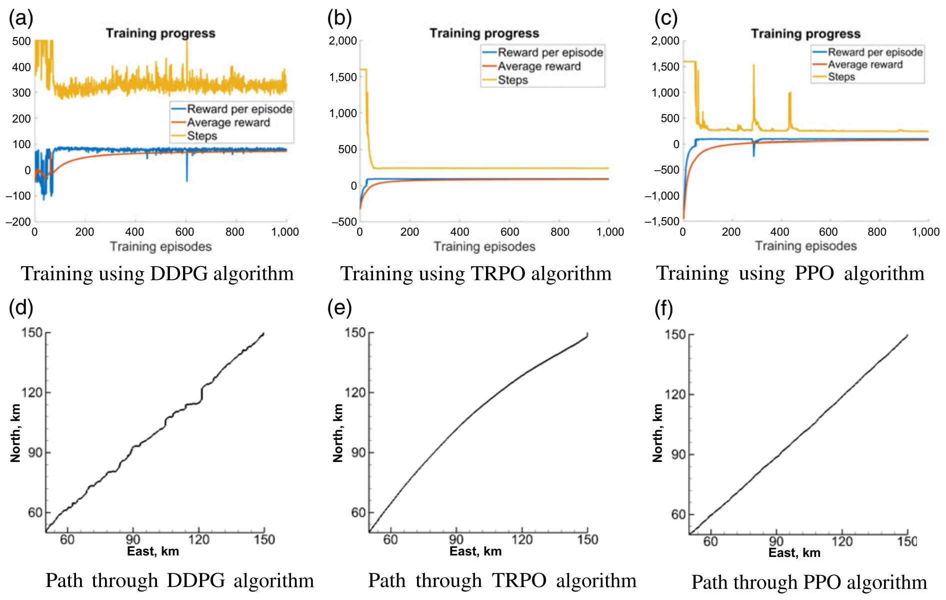


Figure 5. *Comparison of training progress and paths for DDPG, TRPO and PPO for case 1.*

DDPG and TRPO in terms of reaching the final target with the shortest path, although TRPO showed fastest convergence.

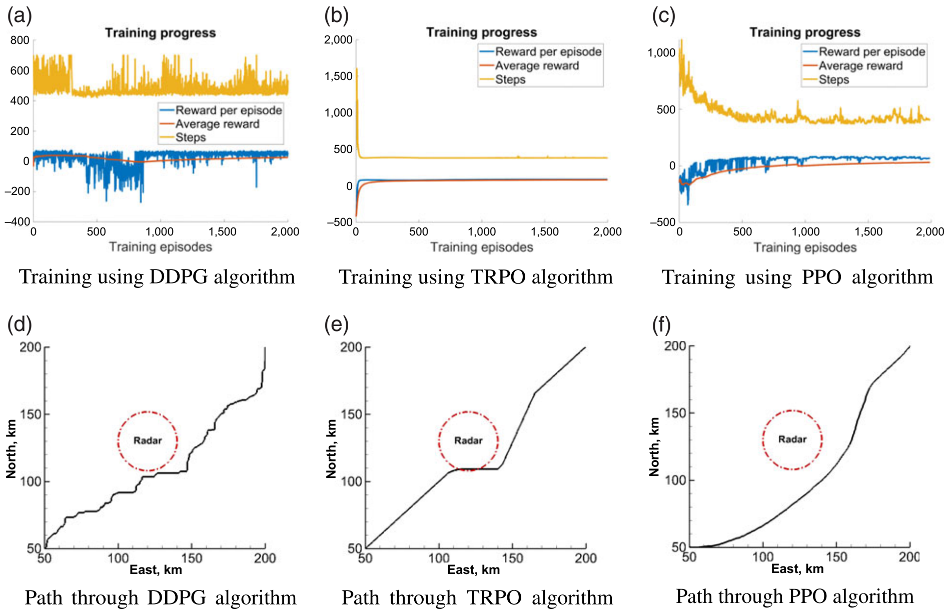
4.2 Case 2:

Case 2 includes one radar. The objective is for the aircraft to reach the target but avoiding radar tracking with the minimum distance. The radar is placed almost at the centre of the initial and target point, so the aerial vehicle must deviate from its path to reach the target safely. The RCS is calculated using the RCS model, then applied to estimate the SNR at each state. If the SNR is higher than the acceptable value, the DRL agent starts to receive a negative reward, such that it changes its path. The training progress and optimal paths of the three DRL algorithms are shown in Fig. 6.

The DDPG algorithm failed to stabilise the episodic steps in attaining the optimal path. Since it is an off-policy algorithm, the policy update by taking a gradient is very much appropriate for this case, as shown in Fig. 6(a). As the complexity of the problem is increased, chatter in the path is also much higher, as seen in Fig. 6(d). Some curve-fitting techniques could be used to achieve a better path.

TRPO performed best for this case. It managed to train much quickly and showed convergence in the initial 100 episodes. To obtain an optimal path, it managed to stabilise training steps efficiently by optimising the policy quite well, as shown in Fig. 6(b). During this training, the new policy obtained by objective function resulted in a much more efficient path with minimum distance. Figure 6(e) shows the optimal path obtained from TRPO, revealing that the agent entered the radar range but then managed to escape from it and reach the target without being tracked while following the shortest path. The agent thus learned to avoid the radar to reach the target with minimal radar exposure.

PPO suffered in optimising the policy for this case. The convergence speed was much slower, as shown in Fig. 6(c), and it could not stabilise the training steps well to generate an optimal path. It may need some hyperparameter tuning to make it more effective, or more training time is required to achieve proper convergence. The path obtained from PPO, shown in Fig. 6(f), is not the shortest or optimal path for this problem. The problem formulation may require some tweaking of hyperparameters.



Training using DDPG algorithm
Training using TRPO algorithm
Training using PPO algorithm
Path through DDPG algorithm
Path through TRPO algorithm
Path through PPO algorithm

Figure 6. *Comparison of training progress and paths for DDPG, TRPO and PPO for case 2.*

However, for comparison purposes, the hyperparameters are set the same for all the cases. Overall, TRPO outperformed DDPG and PPO for this case.

4.3 Case 3:

For case 3, two radars are placed at different positions in the path to increase the complexity of the optimal path planning problem. The goal is to reach the target with minimum exposure to both radars while following the shortest path. The SNR for both radars is estimated at each state to give the agent a reward or penalty based on the action taken in the environment. The aircraft must reach the target by deviating from the radars to achieve the maximum reward. The DDPG, TRPO and PPO algorithms are trained to generate optimal paths as shown in Fig. 7.

The performance of the DDPG algorithm for this case is not good, due to its poor convergence. It cannot stabilise the training steps and reward to generate an optimal path, as shown in Fig. 7(a). Due to the reduced data efficiency of DDPG and function approximation technique, the policy is not very optimal to achieve monotonic results. The path obtained from the training of this case with DDPG managed to reach the target, but the chatter in the path was much greater than in any previous case, as shown in Fig. 7(d). It is challenging to implement any curve-fitting technique on this path due to the amount of chatter.

TRPO tends to converge much earlier in this case too. It manages to stabilise the episodic steps to obtain monotonic results, as shown in Fig. 7(b). The path obtained from TRPO is quite good, but not optimal. It is not the shortest path to reach the target while avoiding radar exposure, as shown in Fig. 7(e). The TRPO agent managed to avoid the radar and reach the target without detection, but it converged on a local optimal policy and did not try to optimise it any further to obtain better results. Therefore, this higher convergence speed is of no use if the objective is not fulfilled in this way.

PPO training took more time to converge than TRPO training, but it provided optimal results. As seen from Fig. 7(c), PPO managed to stabilise the episodic steps by providing a balance between policy optimisation and convergence speed. This balance tends to generate a more optimal path, as shown in Fig. 7(f). The PPO agent managed to avoid radar to minimise the radar exposure and reach the target

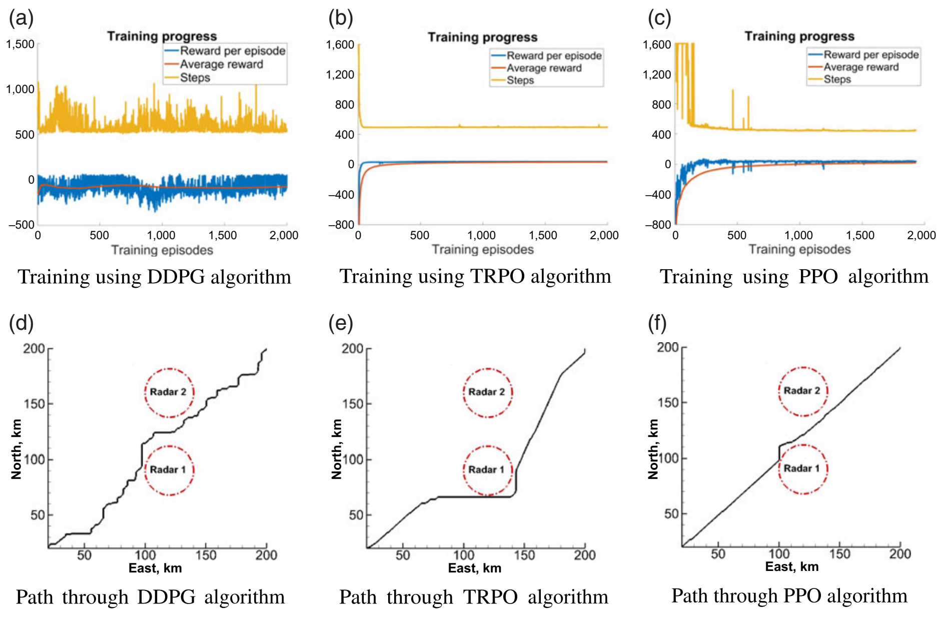


Figure 7. *Comparison of training progress and paths for DDPG, TRPO and PPO for case 3.*

by following the shortest path possible. The same hyperparameters values are used for this case, too, without any separate tuning. PPO is thus recommended for these types of cases.

4.4 Case 4:

The complexity of the optimal path planning problem was further increased by giving no open space for the aerial vehicle to move without radar detection. Here, the goal is to reach the target with as little radar exposure as possible. To achieve this, the aircraft can pass through edges or radars overlapping points by handling the scan time t_s of the radars efficiently; the aircraft speed, SNR and t_s play an important role in achieving the minimum radar exposure. The training progress of DDPG, TRPO and PPO along with the optimal path obtained from each algorithm are shown in Fig. 8.

DDPG completely under-performed for case 4. The DDPG agent did not even manage to reach the target for most of the cases, as seen from the training steps in Fig. 8(a). The maximum number of steps allowed for a single episode is 1,600, but even after 1,600 steps, it did not manage to reach the target. One of the paths obtained from DDPG training is shown in Fig. 8(d). It is not a straight path and shows significant chatter.

Both TRPO and PPO managed to converge by optimising the policy well. PPO took more time to achieve convergence as compared with TRPO. The training progress for these is shown in Fig. 8(b) and (c). PPO and TRPO follow different paths to reach the target to avoid radar tracking as much possible, as shown in Fig. 8(e) and (f).

4.5 Case 5:

Multiple radars are placed in a diagonal for maximum tracking in this case. Here, the objective is to reach the target with minimum radar tracking while having the shortest path. The SNR for each radar is calculated using the distance from the aerial vehicle and RCS to estimate the probability of tracking, based on which a positive reward or penalty is given to the DRL agents to improve the policy, leading

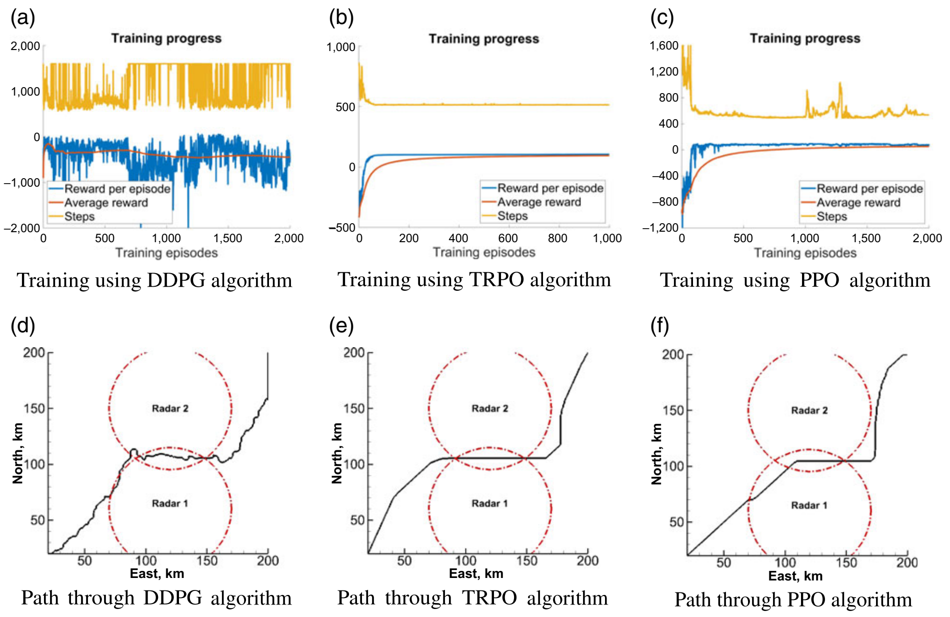


***Figure 8.** Comparison of training progress and paths for DDPG, TRPO and PPO for case 4.*

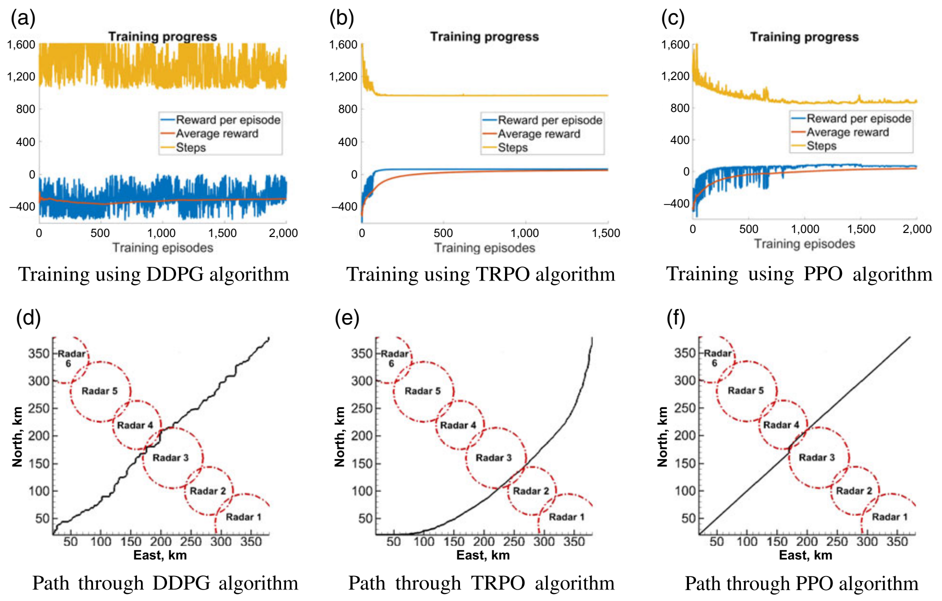


***Figure 9.** Comparison of training progress and paths for DDPG, TRPO and PPO for case 5.*

to a more optimal path with lower tracking probability. The training progress and path obtained by the DDPG, TRPO, and PPO agents is shown in Fig. 9.

The DDPG training results for these cases are not admissible, as in the previous case. The DDPG agent cannot reach the target even after 1,600 steps in most of the cases and failed to stabilise the training steps by optimising the policy, as shown in Fig. 9(a). The path with maximum reward obtained from DDPG is shown in Fig. 9(d). This can be the shortest path, but still the chatter is not acceptable for such a complex control problem.

Table 1. *Quantitative comparison of algorithms*

Parameter	Algorithm	Case 1	Case 2	Case 3	Case 4	Case 5
Average reward	DDPG	73.20106	25.58296	−73.4929	−442.543	−297.625
	TRPO	89.58213	82.86446	38.09815	95.77774	50.05741
	PPO	80.09247	31.28733	19.93104	52.78816	37.8582
Maximum episodic reward	DDPG	90.16232	73.30357	65.0645	58.41217	1.235457
	TRPO	94.59411	88.85028	44.61449	97.62202	64.78628
	PPO	94.63373	81.82337	69.84704	97.44884	91.70995
Minimum steps	DDPG	274	434	506	556	1,045
	TRPO	238	383	492	510	965
	PPO	238	369	435	489	851
Computing time (min)	DDPG	50	185	257	409	433
	TRPO	103	212	236	121	201
	PPO	6	19	27	31	39
Absolute distance (10^3m)	DDPG	100	150	180	180	360
	TRPO	100	150	180	180	360
	PPO	100	150	180	180	360
Episodes	DDPG	1,000	2,000	2,000	2,000	2,000
	TRPO	1,000	2,000	2,000	1,000	1,500
	PPO	1,000	2,000	2,000	2,000	2,000

TRPO and PPO managed to converge in this complex case too, showing some satisfactory results. TRPO converged much earlier as compare with PPO, as seen from Fig. 9(b) and (c). The policy optimisation of PPO was much better than TRPO, as is obvious from the path obtained from these algorithms. The paths obtained from TRPO and PPO are shown in Fig. 9(e) and (f), respectively. Both try to follow the path with minimum radar tracking probability. However, the path obtained from PPO training is much shorter and optimal as compared with TRPO, and the radar tracking probability is also lower for PPO too. More optimal paths and training results could be achieved by tuning the hyperparameters and training for slightly more time.

4.6 Comparison of algorithms for optimal paths

A quantitative comparison of DDPG, TRPO and PPO is carried out for the mentioned cases to select the best algorithm and optimal paths. For the quantitative comparison, some parameters are selected: average reward, maximum episodic reward, minimum steps taken to reach the target and computing time for a fixed number of episodes. Table 1 presents the quantitative values for these parameters. Here, TRPO shows the maximum average rewards for most of the cases. However, the path obtained from TRPO is not optimal compared with PPO, as shown in the previous section, which means that TRPO becomes stuck in a local optimum solution instead of finding the optimal global solution. PPO took more training episodes to achieve optimal solutions as compared with TRPO. On the other hand, PPO shows the maximum episodic reward for most cases because the paths obtained are more optimal in terms of radar avoidance and shortest distance. As the path obtained from PPO is shown to be the shortest possible, the minimum steps are taken through the PPO algorithm as illustrated by the quantitative values in the table. Computing time is also a major concern regarding RL algorithm efficiency. Here, DDPG, TRPO and PPO are calculated without rendering or writing data to disk. In terms of computing time, PPO shows excellent results. For case 5, the most complex case in this study, PPO took only 39

Table 2. *Selection of best algorithm based on performance parameters*

Parameter	Algorithm	Case 1	Case 2	Case 3	Case 4	Case 5	Best selection
Target reached	DDPG	√	√	√			√ √ √
	TRPO	√	√	√	√	√	√ √ √ √ √
	PPO	√	√	√	√	√	√ √ √ √ √
Shortest distance	DDPG						
	TRPO		√				√
	PPO	√		√	√	√	√ √ √ √
Maximum reward	DDPG						
	TRPO		√				√
	PPO	√		√	√	√	√ √ √ √
Minimum tracking	DDPG		√				√
	TRPO			√	√		√ √
	PPO		√	√		√	√ √ √
Fastest convergence	DDPG						
	TRPO	√	√	√	√	√	√ √ √ √ √
	PPO						
Optimal path	DDPG						
	TRPO		√				√
	PPO	√		√	√	√	√ √ √ √

min for 2,000, whereas DDPG took 433min and TRPO took 201min, revealing that PPO could be used for real-time training on some higher-specification system.

PPO shows optimal results based on optimal paths with minimum radar tracking and maximum episodic rewards. The shortest path is taken to reach the target with the minimum computing time for training. To select the best algorithm for optimal paths based on our cases, the performance parameters are compared in Table 2. The performance parameters selected for comparison include the target reached, shortest distance, maximum reward, minimum tracking, fastest convergence and resultant optimal path. For the earlier cases, DDPG can reach the target after some initial episodes. However, as the complexity increases, DDPG tends to reach the target just for some random episodes but cannot constantly follow that path or optimise it. TRPO and PPO performed well by reaching the target after some initial training episodes. PPO outperformed DDPG and TRPO in achieving the shortest distance, maximum reward and minimum track. At the same time, TRPO showed the fastest convergence in all cases. Based on the shortest distance, maximum reward and minimum tracking, PPO provides the desired optimal paths for most cases. The selection of the best algorithm for the mentioned cases is presented in Table 2 based on the selected parameters, indicating that PPO is the best choice in terms of most of the performance parameters.

Overall, DDPG under-performed in obtaining optimal paths, especially as the complexity increased. Both TRPO and PPO showed comparable results in each case. TRPO showed a tendency to converge to optimal local solutions and was thus unable to complete policy optimisation and achieve monotonic results. The results of PPO are deemed to be most impressive in most cases, but it may require hyperparameter tuning to achieve truly excellent results. Based on the findings of this study, PPO is the best algorithm considering the optimal paths with minimum radar tracking, shortest path obtained, maximum episodic reward and minimum computing time.

5.0 Conclusions

This paper used the OpenAI Gym-based environment to handle optimal path planning problems in the

In each case, the RL algorithms DDPG, TRPO and PPO are trained and optimal paths obtained. A comparison is then carried out to select the best algorithms for optimal path planning based on different performance parameters. The results confirm that RL can be used to obtain optimal paths. PPO outperformed DDPG and TRPO for generating optimal paths. DDPG under-performed in finding optimal solutions, while TRPO showed the fastest convergence. For our cases, the hyperparameter values were kept the same. Rigorous tweaking of hyperparameters may provide more optimal results. The results of this study could provide a baseline for further exploration of RL algorithms in more complex environments.

Supplementary material

To view supplementary material for this article, please visit https://dx.doi.org/10.1017/aer.2021.85.

References

[1] Raivio, T., Ehtamo, H. and Hämäläinen, R.P. Aircraft trajectory optimization using nonlinear programming, In Doležal and Fidler (Eds), System Modelling and Optimization, IFIP — The International Federation for Information Processing, Springer, Boston, MA, 1996. https://doi.org/10.1007/978-0-387-34897-1_52.

[2] Betts, J.T. Survey of numerical methods for trajectory optimization, *J. Guidance Control Dyn.*, 1998, **21**, (2), pp 193–207.

[3] Judd, K.B. and McLain, T.W. Spline based path planning for unmanned air vehicles, AIAA Guidance, Navigation, and Control Conference and Exhibit, 2001.

[4] Maqsood, A. and Go, T.H. Optimization of transition maneuvers through aerodynamic vectoring, *Aerosp. Sci. Technol.*, 2012, 23, (1), pp 363–371. 35th ERF: Progress in Rotorcraft Research.

[5] Feroskhan, M. and Go, T.H. Control strategy of sideslip perching maneuver under dynamic stall influence, *Aerosp. Sci. Technol.*, 2018, **72**, pp 150–163.

[6] Mir, I., Maqsood, A., Eisa, S.A., Taha, H. and Akhtar, S. Optimal morphing – augmented dynamic soaring maneuvers for unmanned air vehicle capable of span and sweep morphologies, *Aerosp. Sci. Technol.*, 2018, **79**, pp 17–36.

[7] Aggarwal, S. and Kumar, N. Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges, *Comput. Commun.*, 2020, **149**, pp 270–299.

[8] Pang, Y., Li, Y., Wang, J., Yan, M., Chen, H., Sun, L., Xu, Z. and Qu, S. Carbon fiber assisted glass fabric composite materials for broadband radar cross section reduction, *Compos. Sci. Technol.*, 2018, **158**, pp 19–25.

[9] Baek, S., Lee, W. and Joo, Y. A study on a radar absorbing structure for aircraft leading edge application, *Int. J. Aeronaut. Space Sci.*, 2017, **18**, pp 215–221.

[10] Zainud-Deen, S.H., Malhat, H.A.E.A. and Shabayek, N.A. Reconfigurable RCS reduction from curved structures using plasma based FSS, *Plasmonics*, 2020, **15**, pp 341–350. https://doi.org/10.1007/s11468-019-01048-y.

[11] Lingxiao, W. and Deyun, Z. Effective path planning method for low detectable aircraft, *J. Syst. Eng. Electr.*, 2009, **20**, (4), pp 784–789.

[12] Pachter, M. and Hebert, J. Optimal aircraft trajectories for radar exposure minimization, *Proceedings of the American Control Conference*, vol. 3, 2001, pp 2365–2369.

[13] Moore, F.W. Radar cross-section reduction via route planning and intelligent control, *IEEE Trans.* Control Syst. *Technol.*, 2002, 10, (5), pp 696–700.

[14] Norsell, M. Aircraft trajectories considering radar range constraints, *Aerosp. Sci. Technol.*, 2002, **6**, (1), pp 83–89.

[15] Kabamba, P.T., Meerkov, S.M. and Zeitz, F.H. *Optimal UCAV Path Planning Under Missile Threats*, vol. **16**. IFAC, 2005.

[16] Kabamba, P.T., Meerkov, S.M. and Zeitz, F.H. Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking, *J. Guid. Control Dyn.*, 2006, **29**, (2), pp 279–288.

[17] Sutton, R. and Barto, A. *Reinforcement Learning*, MIT Press, 2018.

[18] Yan, C., Xiang, X. and Wang, C. Fixed-wing UAVs flocking in continuous spaces: a deep reinforcement learning approach, *Robot. Auton. Syst.*, 2020, 131, p 103594.

[19] Ma, Z., Wang, C., Niu, Y., Wang, X. and Shen, L. A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles, *Robot. Auton. Syst.*, 2018, **100**, pp 108–118.

[20] Liu, Y., Liu, H., Tian, Y. and Sun, C. Reinforcement learning based two-level control framework of UAV swarm for cooperative persistent surveillance in an unknown urban area, *Aerosp. Sci. Technol.*, 2020, **98**, p 105671.

[21] Carlucho, I., De Paula, M., Wang, S., Petillot, Y. and Acosta, G.G. Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning, *Rob. Auton. Syst.*, 2018, **107**, pp 71–86.

[22] You, C., Lu, J., Filev, D. and Tsiotras, P. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning, *Rob. Auton. Syst.*, 2019, **114**, pp 1–18.

[23] Carrio, A., Sampedro, C., Rodriguez-Ramos, A. and Campoy, P. A review of deep learning methods and applications for unmanned aerial vehicles, *J. Sens.*, 2017, **2017**, pp 1–13.

[24] Dos Santos, Ś.R.B., Nascimento, C.L. and Givigi, S.N. Design of attitude and path tracking controllers for quad-rotor robots using reinforcement learning, IEEE Aerospace Conference Proceedings, 2012, pp 1–16.

[25] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. and Riedmiller, M. Deterministic policy gradient algorithms, 31st International Conference on Machine Learning, ICML 2014, vol. 1, 2014, pp 605–619.

[26] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L. and van den Driessche, G. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016, 529, (7587), pp 484–489.

[27] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. Continuous control with deep reinforcement learning, ICLR 2016, 2016, p 14.

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

[29] Schulman, J., Levine, S., Moritz, P., Jordan, M. and Abbeel, P. Trust region policy optimization, 32nd International Conference on Machine Learning, ICML 2015, vol. 3, 2015, pp 1889–1897.

[30] Hwangbo, J., Sa, I., Siegwart, R. and Hutter, M. Control of a quadrotor with reinforcement learning, *IEEE Rob. Autom. Lett.*, 2017, **2**, (4), pp 2096–2103.

[31] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. Proximal policy optimization algorithms, CoRR, abs/1707.06347, 2017.

[32] Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Ali Eslami, S.M., Riedmiller, M.A. and Silver, D. Emergence of locomotion behaviours in rich environments, CoRR, abs/1707.02286, 2017.

[33] Lopes, G.C., Ferreira, M., Da Silva Simoes, A. and Colombini, E.L. Intelligent control of a quadrotor with proximal policy optimization reinforcement learning, Proceedings - 15th Latin American Robotics Symposium, 6th Brazilian Robotics Symposium and 9th Workshop on Robotics in Education, LARS/SBR/WRE 2018, 2018, pp 509–514.

[34] Bohn, E., Coates, E.M., Moe, S, and Johansen, T.A. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization, 2019 International Conference on Unmanned Aircraft Systems, ICUAS 2019, 2019, pp 523–533.

[35] Koch, W., Mancuso, R., West, R. and Bestavros, A. Reinforcement learning for UAV attitude control, *ACM Trans.* Cyber Phys. *Syst.*, 2019, 3, (2), pp 1–13.

[36] Zhang, Z., Wu, J., Dai, J. and He, C. A novel real-time penetration path planning algorithm for stealth UAV in 3D complex dynamic environment, *IEEE Access*, 2020, **8**, pp 122757–122771.

[37] Zhao, Z., Niu, Y., Ma, Z. and Ji, X. A fast stealth trajectory planning algorithm for stealth uav to fly in multi-radar network, *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2016, pp 549–554.

[38] Richter, R. and Gomes, N.A.S. A-4 Skyhawk aircraft stealth capacity against L-band radar based on dynamic target detection, 2020 IEEE Radar Conference (RadarConf20), 2020, pp 1–5.