Applied Soft Computing 147 (2023) 110773

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance

Amala Sonny, Sreenivasa Reddy Yeduri, Linga Reddy Cenkeramaddi *

Department of ICT, University of Agder, Grimstad, 4879, Aust-Agder, Norway

ARTICLE INFO

Article history:
Received 14 March 2023
Received in revised form 21 July 2023
Accepted 21 August 2023
Available online 31 August 2023

Keywords:
Dynamic obstacle avoidance
Path planning
Q-learning
Shortest distance prioritization
Unmanned aerial vehicles (UAVs)

ABSTRACT

Recently, unmanned aerial vehicles (UAVs) have shown promising results for autonomous sensing. UAVs have been deployed for multiple applications that include surveillance, mapping, tracking, and search operations. Finding an efficient path between a source and a goal is a critical issue that has been the focus of recent exploration. Many path-planning algorithms are utilized to find an efficient path for a UAV to navigate from a source to a goal with obstacle avoidance. Despite the extensive literature and numerous research proposals for path planning, dynamic obstacle avoidance has not been addressed with machine learning. When the obstacles are dynamic, i.e., they can change their position over time, and the constraints of the path planning algorithm become more challenging. This in turn adds a layer of complexity to the path planning algorithm. To address this challenge, a Q-learning algorithm is proposed in this work to facilitate efficient path planning for UAVs with both static and dynamic obstacle avoidance. We introduced the Shortest Distance Prioritization policy in the learning process which marginally reduces the distance that the UAV has to travel to reach the goal. Further, the proposed Q-learning algorithm adopts a grid-graph-based method to solve the path-planning problem. It learns to maximize the reward based on the agent's behavior in the environment. Through results, the performance comparison between the proposed approach and state-of-the-art path planning approaches such as A-star, Dijkstra, and Sarsa algorithms are evaluated in terms of learning time and path length. We show through results that the proposed approach results in improved performance when compared to state-of-the-art approaches. Further, the effect of an increased number of obstacles are evaluated on the performance of the proposed approach.

1. Introduction

Path planning for Unmanned Aerial Vehicles (UAVs) is a critical component in the field of autonomous navigation. It involves finding the optimal path for a UAV in order to reach a given goal constrained by various parameters such as obstacles, energy consumption, and time limits [1,2]. Path planning aims to ensure that the UAV can safely and efficiently reach its destination while avoiding collisions and other hazards [3]. The existing methods for UAV path planning includes grid-based methods [4,5], potential field methods [6,7], sampling-based methods [8,9], model predictive control (MPC) based methods [10,11], and hybrid methods [12]. In grid-based methods, the algorithm divides the environment into a grid of cells and utilizes various search algorithms such as A-star [13] or Dijkstra [14,15] to find an optimal path for the UAV. The potential field method utilizes a combination of both repulsive and attractive forces that can be employed to steer a UAV to its target while avoiding any obstacles along the way. The repulsive force guides the UAV away from obstacles, while the attractive force attracts it towards the target. Sampling-based methods randomly sample the points in the environment and use probabilistic frameworks such as Rapidly-Exploring Random Trees (RRT) [16] or Probabilistic Roadmaps (PRM) [17] to find the optimal path for the UAV. Model prediction control is a control strategy that uses mathematical models of the UAV dynamics and the environment to predict its future behavior. The method then optimizes a cost function to find the best control inputs to guide the UAV to its goal. A trajectory tracking control method is proposed in [18] for a four-wheel omnidirectional Covid-19 aromatherapy robot. This method effectively addresses potential errors in robot movement, such as incorrect speed settings, deviations in movement direction, and changes in robot orientation. It utilizes a PID control approach to regulate motor speed, robot movement direction, and robot orientation direction. Integration of the PID control with the odometry system provides feedback for precise and stable trajectory tracking. A disturbance rejection control was proposed in [19] for medical telepresence robot combining a PID Control and Kalman Filter. The combination of the GY-521 MPU 6050

* Corresponding author.
E-mail address: linga.cenkeramaddi@uia.no (L.R. Cenkeramaddi).

https://doi.org/10.1016/j.asoc.2023.110773

sensor reading and the Kalman Filter aimed to achieve accurate readings. Simulation and real robot tests were conducted under different conditions, demonstrating the robot's ability to maintain balance for approximately one hour and re-balance itself when subjected to external disturbances. The Kalman filter proved effective in reducing noise and improving system performance. A particle swarm optimization (PSO)-based path planning approach is proposed in [20]. This framework utilizes the modified PSO algorithm for UAV path planning to support the rate requirements of the user. The proposed framework formulated the problem of joint path planning and energy consumption with the aim of improving the instantaneous sum rate of the user. To solve this formulation, the line of sight (LoS) probability is used to determine an optimal destination location where the UAV can establish a LoS link with the user and provide the required downlink rate. This step ensures effective communication between the UAV and the user. Subsequently, the modified PSO algorithm is employed to find the most energy-efficient path from the source to the destination. By leveraging the capabilities of the PSO algorithm, we can optimize the path-planning process and minimize energy consumption. Even though the proposed framework in [20] focused on addressing the rate requirements of the user and optimizing energy consumption, it did not consider dynamic obstacles in path planning. Finally, hybrid methods combine two or more of the above methods to find the most efficient path for the UAV. For example, a grid-based method could be used to find a rough path, which is then refined using a sampling-based method.

The choice of a method depends on the specific requirements of the UAV and the environment it is operating in. Grid-based methods may be well suited for environments with simple geometry and few obstacles, while sampling-based methods may be better suited for environments with complex geometry and a large number of obstacles. One of the challenges in path planning for UAVs is the requirement to respond promptly to real-time changes in the environment [21]. This can be achieved through the use of online algorithms that continually update the path based on new information about the environment. Another challenge is to take into account the dynamics of the UAV, such as its speed and acceleration limits, in order to ensure a smooth and safe flight. However, path planning for UAVs is a crucial component of autonomous navigation that involves finding the most efficient path for the UAV to reach its destination while taking into account various constraints. Path planning must also be able to handle real-time changes in the environment and take into account the dynamics of the UAV.

Path planning is a critical aspect of autonomous navigation and Q-learning is one of the advanced techniques mostly used for path planning in recent times. Q-learning focuses on training agents to make optimal decisions in a given environment using the reward obtained from an action by the agent. In UAV path planning based on Q-learning, the agent is the UAV and the environment is the scene where it needs to fly. The main focus of the agent is to find a reliable and efficient path, with maximum reward, that leads to the goal point safely by avoiding collision with the obstacles. It utilizes a Q-table to keep track of the expected reward for each combination of state and action, allowing it to make decisions based on the highest expected reward. The Q-table is updated as the agent interacts with the environment and estimates the best actions to take in each state for maximum rewards. The initial Q-table gets updated when the agent investigates the environment and learns the optimal policy. The state can be the UAV's position and orientation and the actions can be a set of possible movements such as moving forward, turning left, or turning right. The reward is a function of the UAV's position and the distance to the target. For example, the reward can be higher when the UAV is closer to the target and lower when it encounters obstacles. One of the challenges in using Q-learning for path planning is the large state–action space, which can make the algorithm slow and impractical. Another challenge is to plan the most efficient path while learning the environment. To deal with this challenge, [22] has proposed a path-planning approach for a group of robots that combines Q-learning with PSO. This approach enabled the agents to optimally decide the path by studying the unknown environment thoroughly. The proposed work in [22] did not consider the dynamic obstacles in the environment. In addition to all the challenges in UAV path planning, the priorities of the UAV can be different such as safety, energy consumption, and distance to the target to keep a reliable communication link. To address this situation, instead of applying dynamic programming or geometric-based methods, an offline Q-learning-based approach has been proposed in [23] for path planning. In this approach, the policy is concentrated on path length, safety, and energy consumption to decide the reward. A new deterministic Q-learning approach for a mobile robot has been proposed in [24]. This approach uses prior knowledge about the distance to the next state and the goal from the current state. This efficient use of information made the entire path planning less time complex as compared to other approaches. An improved Q-learning approach has been proposed in [25] for UAV path planning in an unknown antagonistic scenario. In [25], a Q-function initialization method along with a new action selection strategy is introduced in order to enhance the performance. A-star and Q-learning-based hybrid path planning has been proposed in [26] that analyzed UAV path planning in terms of local dynamic hierarchical planning and global static planning. An adaptive and random exploration (ARE) approach has been proposed in [27] to address the tasks of UAV path planning. Similar to [27], a Q-leaning-based path planning approach in an uncertain dynamic environment is proposed in [28,29].

The above-discussed approaches avoid the consideration of the presence of dynamic obstacles in the considered territory of the UAV. Even though the literature has many research proposals for path planning, dynamic obstacle avoidance has not been explored with machine learning. When the obstacles are dynamic, i.e., they can change their position over time, the constraints of the path planning algorithm become more challenging. It adds a layer of complexity to the path planning algorithm. Motivated by this, in this work, a Q-learning algorithm is proposed for UAV path planning with both static and dynamic obstacles avoidance. The prime contributions of this work are as follows:

- A Q-learning approach with a shortest distance prioritization policy is proposed for UAV path planning with static and dynamic obstacle avoidance.
- The proposed Q-learning algorithm adopts a grid-graph-based method to solve the path-planning problem. It learns to maximize the reward based on the agent's behavior in the environment.
- The performance of the proposed approach is analyzed with an increasing number of dynamic obstacles in the environment.
- Finally, we compare the proposed path planning approach with state-of-the-art approaches such as the A-star algorithm, Dijkstra algorithm, and Sarsa algorithm.
- Through numerical results, we show the superior performance of the proposed approach when compared to state-of-the-art approaches.

The later sections are organized as follows. Section 2 details the system model. The proposed Q-learning algorithm is described in Section 3 and Section 4 evaluates its performance. Finally, concluding remarks along with possible future scope are presented in Section 5.

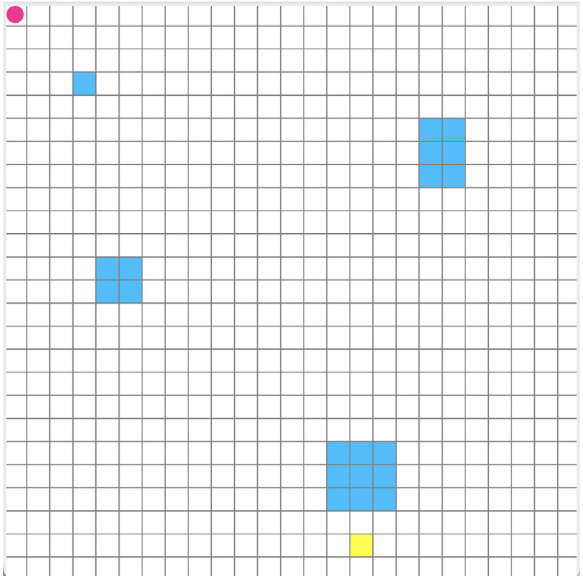


Fig. 1. Environmental modeling of path planning.

2. System model

A piece of in-depth information about the environment is the foundation for a robust path-planning approach. A detailed description of various kinds of obstacles that are present in the environment will make the obstacle avoidance process efficient and coherent. Further, the key to a successful approach to obstacle avoidance is realizing the kinds of obstacles and treating them differently. At lower altitudes, the environment presents several obstacles like buildings, trees, and hills in the UAV flight terrain that poses significant risk. Accurately modeling these obstacles for path planning purposes proves to be challenging. To streamline the modeling process and reduce computational complexity, we represent them with geometric shapes whose dimensions can be varied to represent real-time objects such as trees, hills, and buildings. The basic environmental structure considered in this study is given in Fig. 1. We consider a grid with 25×25 cells, each cell is fixed to 20 pixels. The red circle indicates the UAV and the yellow square denotes the target location (goal) for the UAV. The location of the UAV represents the starting position. All four blue rectangles are static obstacles present in the environment with various sizes and shapes. The UAV needs to discover an optimum path from the source to the target location in an efficient way to reduce the cost and time, in addition to avoiding obstacles to reduce the risk of collision. Next, we discuss the effect of an environment with dynamic obstacles for path planning.

2.1. Environment with dynamic obstacles

Path planning algorithms are designed to find an efficient path for a robot or a UAV to navigate from the source to the target while avoiding obstacles. However, when the obstacles are dynamic, i.e., these can change their positions from time to time, the constraints of the path planning algorithm become more challenging. The algorithm needs to process the information about obstacles continuously throughout the process and make real-time decisions to avoid collisions. This requires a fast and high-processing algorithm. Additionally, the algorithm needs to detect and track these dynamic obstacles accurately. As the number of dynamic obstacles increases, the computational complexity also increases. This makes path planning a challenging and time-consuming process, especially in complex environments. Also, predicting the future positions of the dynamic obstacle and keeping a safe margin to ensure the safety of the UAV makes the entire process more complex and it can adversely affect the accuracy of the path-planning algorithm. Due to all these factors, the presence of dynamic obstacles adds a layer of complexity to the path-planning algorithm and makes it more challenging. In this work, we consider that UAV and dynamic obstacles move with the same velocity.

3. Proposed Q-learning-based approach

The approach presented in this study suggests a Q-learning algorithm for UAV path planning amidst both static and dynamic obstacles. This approach adopted a policy that allows the agent to select actions corresponding to the maximum reward and the shortest path to the destination. The flowchart of the training framework for UAV path planning and collision avoidance is given in Fig. 2. The proposed approach comprises two main components. The first one is the shortest path selection where the agent selects the next action based on a Shortest Distance Prioritization policy from its current state. The next component is collision avoidance where the agent tries to adopt a trajectory by avoiding all the obstacles in its way. This work considers up to four dynamic obstacles in addition to static obstacles, whose location changes randomly over time. The agent learns the environment by following the policy of shortest-path selection and collision avoidance to reach the destination in a cost-effective manner without collisions. A detailed description of the proposed algorithm is given in the next section.

3.1. Q-learning algorithm

Q-learning is a time-difference-based reinforcement learning algorithm for solving tasks with the state–action values represented by a Q-table. It is the primary model-free reinforcement learning approach that does not require complete environmental awareness as it adopts a grid-graph-based method to solve the problem. Q-learning learns to maximize the reward based on the agent's behavior in the environment. The framework of the Q-learning algorithm is illustrated in Fig. 3. The structure of Q-learning consists of five key components namely an agent, an environment, a state, an action, and a reward. It explores the environment while implementing the concepts of reward and penalty. The agent chooses its next action based on the reward and the corresponding policy, and then, performs the action. The environment contains several states and corresponding rewards from which the agent can choose its next state. The action is the movement from one state to another. After each action taken by the agent, the environment transit to the next state and provides a corresponding reward. The next action selected by the agent depends on the strategy and the rewards correspond to different states. This process continues until the agent reaches its predefined destination. Through this learning process, the agent learns to proceed from every state by adopting suitable actions that provide the maximum reward. One of the primary advantages of utilizing Q-learning is its ability to employ the temporal difference (TD) method for offline learning and to solve for the optimal strategy of a Markov process using the Bellman equation. Hence, Q-learning remains a widely used and well-studied reinforcement learning algorithm. It is a simple and effective way to learn optimal policies in complex environments, and its simplicity and generality make it a popular choice for many applications.

The foundation of reinforcement learning lies in solving sequential decision-making problems as defined by the Markov decision process (MDP) [30]. In the learning process of the agent,

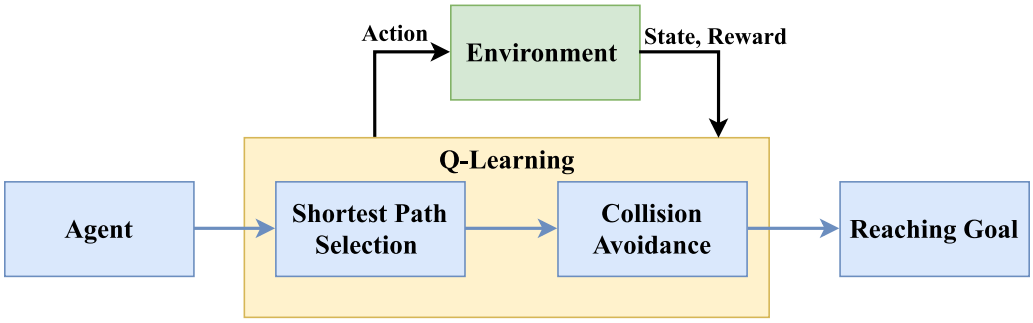


Fig. 2. Training framework for UAV path planning and collision avoidance.

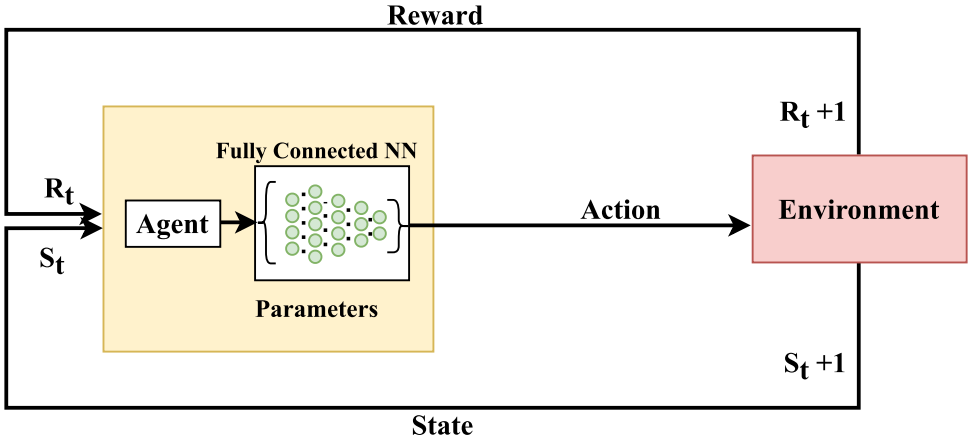


Fig. 3. Structure of Q-learning algorithm.

the value function is utilized to construct the Bellman equation, which is then resolved through Q-learning [31]. In a probabilistic setting, the transition state and reward following an action are both random. For selecting future actions at a particular state in such environments, certain rules called policies have to be followed. This type of approach is known as the reinforcement learning (RL) algorithm. Typically, RL algorithms are formulated using MDP. In RL, there are parameters such as state, action, reward, discount factor, policy, and state transition probability matrix that are used to evaluate the environment. The state refers to the environmental observations of the agent, and an action refers to the collection of possible actions that the agent can undertake in a specific state [32,33]. The reward is assigned to the agent to learn from the state transition probability numerically. This probability represents the movement of the agent between states. The discount factor is applied to accommodate the decrease in the value of rewards over time, and the policy is a set of rules the agent must follow when in a given state to determine the next action [34]. It refers to the process of mapping from a state and action to the likelihood of taking that particular action when the agent is in that state. It can be expressed as

$$\Pr(\lambda|\gamma) = P[\Lambda_t = \lambda|\Gamma_t = \gamma]. \tag{1}$$

Here, *Pr* represents the probability of the agent selecting a particular action λ when in a specific state γ. An agent decides on which action to perform next based on the reward associated with the action.

Value functions are used as the criteria to determine which policy will result in better reward in the next action. These functions represent the cumulative sum of all rewards expected to be obtained through a specific policy. The state-value function and the action-value function are the two distinct categories of value functions that exist in Q-learning. The state-value function can be computed as

$$v_p(\gamma) = E_p[R_{t+1} + \delta v_p(\Gamma_{t+1})|\Gamma_t = \gamma], \tag{2}$$

where p is the policy chosen by the agent, R_{t+1} refers to the expected reward to be received in the next time step, and δ represents the discount factor. (2) denotes the state-value function, which is an estimate of the total sum of rewards associated with the subsequent state. It is the overall reward expected for selecting a certain state γ. On the other hand, the action-value function characterizes the anticipated return that the agent will obtain by taking a particular action λ while in state γ. The action-value function is commonly known as the Q-function. It assesses the desirability of selecting a certain action in state s while adhering to policy p. The Q-learning algorithm targets to reach the optimal action-value function, represented as Q* [35]. The agent learns a series of trajectories based on the optimal Bellman equation, which is defined as

$$Q^*(\gamma, \lambda) = E_p[R_{t+1} + \delta \max_{\lambda \epsilon \Lambda} Q^*(\Gamma_{t+1}, \lambda')|\Gamma_t = \gamma, \Lambda_t = \lambda], \tag{3}$$

where λ' is the next action and $Q^*(\gamma, \lambda)$ can be approximated as $\tilde{Q}(\gamma, \lambda)$ at time t. The expected state at the next time instant is denoted as Γ_{t+1}. The agent executes an action Λ_t from the current state Γ_t to transit to the subsequent state Γ_{t+1} and receives the corresponding reward R_t from the environment. This process is often referred to as the agent-environment interaction or the agent-environment loop [24]. The Q-function and the value function are related as per the following equation

$$v_p(\gamma) = \sum_{\lambda \epsilon \Lambda} p(\lambda|\gamma) Q^*(\gamma, \lambda). \tag{4}$$

The value function is obtained by adding the value of the corresponding Q-function and policy for each action. Usually, the

Table 1
Experimental parameters [36].

Parameter	Value
b	5000
β	0.3
γ	[0.1,0.9]
ϵ	0.9

value function and Q-function are represented as Bellman equations. It describes the correlation between value functions of current and following states. The Q-learning algorithm consists of two primary components. The first is the collection of training data, which is guided by a predefined policy. The policy can be adjusted to suit the objectives of the problem at hand. The agent interacts with the environment and generates an estimated trajectory based on the policy, which can then be partitioned into the four components: Γ_t, Λ_t, R_t, and Γ_{t+1}. This kind of policy is referred to as a behavior policy which is used to control agents in an environment. The most common type of behavior policy is ϵ-greedy policy which can be defined as

$$\begin{cases} \underset{\lambda}{argmax}\ \tilde{Q}(\Gamma_t, \lambda), & 1-\epsilon \\ Uniform, & \epsilon \end{cases} \tag{5}$$

The ϵ-greedy algorithm is utilized to choose an action λ in state Γ_t. The agent carries out the action and receives a reward R_t which leads to a transition to state γ'. The Q-learning algorithm employs this greedy strategy to select actions at each state with the aim of maximizing the reward. In this particular study, the value of ϵ was set to 0.9 [36]. The second component of Q-learning is experience replay, which is a memory replay technique employed in reinforcement learning to store the experiences of the agent at each time step. To begin, a randomly selected quad $(\Gamma_t, \Lambda_t, R_t, \Gamma_{t+1})$ is drawn from the experience replay memory.

The tuple $(\Gamma_t, \Lambda_t, R_t, \Gamma_{t+1}, \tilde{Q}_{now}, \tilde{Q}_{new})$ is used to record the state, action, reward, and resulting state of the agent-environment interaction. The estimated Q-values $\tilde{Q}_{now}$ and $\tilde{Q}_{new}$ are also included in this tuple, and the value of $\tilde{Q}_{now}$ at position γ_j, λ_j is saved as $\hat{q}j$. The maximum value of the state Γ_{t+1} in $\tilde{Q}_{now}$ is then computed.

$$\hat{q}_{j+1} = \underset{\lambda}{max}\ \tilde{Q}_{now}(\gamma_{j+1}, \lambda). \tag{6}$$

Then, the TD of the target is calculated as

$$\hat{y}_j = r_j + \gamma \hat{q}_{j+1}, \tag{7}$$

and the TD error is calculated as

$$\delta_j = \hat{q}_j - \hat{y}_j. \tag{8}$$

Based on this, the new estimation of the action-value function, $\tilde{Q}_{new}$, for the current state is determined as

$$\tilde{Q}_{new}(\gamma_j, \lambda_j) \leftarrow (1-\alpha)\tilde{Q}_{now}(\gamma_j, \lambda_j) + \alpha\delta_j, \tag{9}$$

where, α and γ are the hyperparameters denoting the learning rate and discount factor, respectively. The value of α lies between 0 and 1 and is used in the update equation to determine the extent to which the newly acquired information overrides the existing Q-values. When γ is 0, it indicates that the agent prioritizes recent rewards, whereas $\gamma = 1$ signifies a long-term commitment to higher rewards. If the discount factor exceeds 1, the action value Q may diverge. In such cases, gradually increasing the discount factor from a lower value to the final value can improve learning efficiency. For instance, as the number of iterations increases, γ increases from 0.1 to 0.9. The experience replay considers a total of 5000 samples (b=5000), and β is set to 0.3. The list of parameters is tabulated in Table 1.

The Q-values for each state are continuously updated using the Bellman equation as expressed in (9). Prior to initiating the learning process, all possible rewards are initialized in the Q-table. The agent takes actions based on a pre-defined policy and transitions to the next state, repeating this process until the Q-values converge to a fixed threshold that is specific to a particular objective [37].

Algorithm 1: Proposed Q-learning algorithm for UAV path planning problem in the presence of static obstacles.

Input: Source location, destination location, and solution space
Output: Optimal path for UAV from source to destination
Initialize $Q(\gamma, \lambda) \leftarrow 0$ (Γ states, Λ actions);
for *each episodes* **do**
  set $\gamma_t \leftarrow \lambda$ random state from state set Γ ;
  while $(\gamma_t \neq target)$ **do**
    for *each* $\lambda_t^i \in \Lambda$ *where* $\Lambda = [up,down,left,right]$ **do**
      Choose λ_t^i from γ_t by using suitable policy (ϵ-greedy etc.)
    end
    Perform action λ_t^i and receive penalty or reward
    Update $Q(\gamma_t, \lambda_t)$
  end
end

Algorithm 2: Proposed Q-learning algorithm for UAV path planning problem in the presence of both static and dynamic obstacles.

Input: Source location, destination location, and solution space
Output: Optimal path for UAV from source to destination
Initialize $Q(\gamma, \lambda) \leftarrow 0$ (Γ states, Λ actions), J Dynamic Obstacles;
for *each episodes* **do**
  set $\gamma_t \leftarrow \lambda$ random state from state set Γ ;
  Determine the location loc_d^j of each dynamic obstacle $j \in J$;
  while $(\gamma_t \neq target)$ **do**
    for *each* $\lambda_t^i \in \Lambda$ *where* $\Lambda = [up,down,left,right]$ **do**
      Choose λ_t^i from γ_t by using suitable policy (ϵ-greedy etc.)
      Determine the location loc_d^j of each dynamic obstacle $j \in J$;
    end
    if $(loc_{\lambda_t^i} == loc_d)$ **then**
      break
    end
    Perform action λ_t^i and receive penalty or reward
    Update $Q(\gamma_t, \lambda_t)$
  end
end

4. Performance evaluation

This section outlines the experimental findings that evaluate the efficacy of the proposed method relative to other approaches.

Algorithm 3: Proposed Q-learning algorithm for UAV path planning problem in the presence of static obstacles.

Input: Source location, destination location, and solution space
Output: Optimal path for UAV from source to destination
Initialize $Q(\gamma, \lambda) \leftarrow 0$ (Γ states, Λ actions);
for *each episodes* **do**
  set $\gamma_t \leftarrow \lambda$ random state from state set Γ ;
  while $(\gamma_t \neq target)$ **do**
    for *each* $\lambda_t^i \in \Lambda$ *where* $i \in$ *[up,down,left,right]* **do**
      Determine location $loc_{\lambda_t^i}$ of agent by doing action λ_t^i
      Calculate distance $dist_t^i \in dist_t$ from $loc_{\lambda_t^i}$ to Target location.
      Choose $loc_{\lambda_t^i}$ corresponds to smallest $dist_t^i$ from $dist_t$
      Choose λ_t^i corresponds to $loc_{\lambda_t^i}$ which makes the agent move closer to Target location
    end
    Perform action λ_t and receive penalty or reward
    Update $Q(\gamma_t, \lambda_t)$
  end
end

Algorithm 4: Proposed Q-learning algorithm for UAV path planning problem in the presence of both static and dynamic obstacles.

Input: Source location, destination location, and solution space
Output: Optimal path for UAV from source to destination
Initialize $Q(\gamma, \lambda) \leftarrow 0$ (Γ states, Λ actions), J Dynamic Obstacles;
for *each episodes* **do**
  Set $\gamma_t \leftarrow \lambda$ random state from state set Γ ;
  Determine the location loc_d^j of each dynamic obstacle $j \in J$;
  while $(\gamma_t \neq target)$ **do**
    for *each* $\lambda_t^i \in \Lambda$ *where* $i \in$ *[up,down,left,right]* **do**
      Determine location $loc_{\lambda_t^i}$ of agent by doing action λ_t^i
      Calculate distance $dist_t^i \in dist_t$ from $loc_{\lambda_t^i}$ to Target location.
      Choose $loc_{\lambda_t^i}$ corresponds to smallest $dist_t^i$ from $dist_t$
      Choose λ_t^i corresponds to $loc_{\lambda_t^i}$ which makes the agent move closer to Target location
    end
    if $(loc_{\lambda_t^i} == loc_d)$ **then**
      break
    end
    Perform action λ_t and receive penalty or reward
    Update $Q(\gamma_t, \lambda_t)$
  end
end

4.1. Experimental conditions

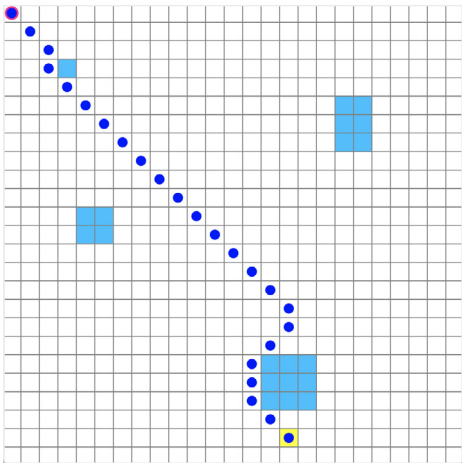
The goal of path planning is to ensure the safety and efficiency of the system while avoiding collision with both static and dynamic obstacles. Path planning algorithms take into account various factors such as the size, shape, and speed of the obstacles, as well as the system's constraints and objectives, to generate a safe and efficient path. Hence, in this study, we analyze the performance of the algorithm in the presence of static and dynamic obstacles. For this, we consider a scenario with four static obstacles of various sizes as shown in Fig. 1. For analyzing the performance of the original Q-learning and the proposed algorithm in the presence of dynamic obstacles, and to assess the algorithm's effectiveness in a crowded environment, we examined two scenarios. The first scenario involved two dynamic obstacles, while the second scenario had four dynamic obstacles. These scenarios are chosen specifically to evaluate the algorithm's performance in the presence of multiple moving obstacles. We randomly placed these dynamic obstacles in the environment without limiting their movement to any part of the environment. The performance of the algorithms in all the scenarios is described in the following section.

4.2. Q-learning with shortest distance prioritization

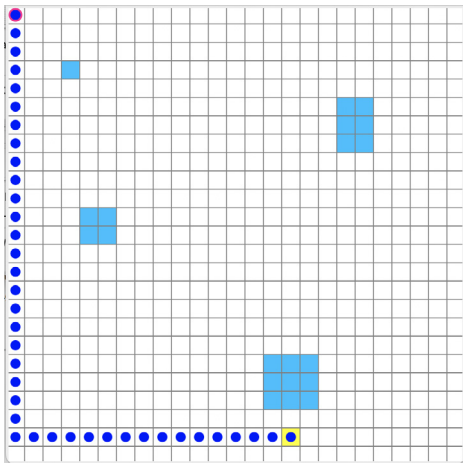
The major objective of every UAV-related application is to meet the mission target with high efficiency in terms of cost, fuel, and time without collision [38]. Attaining maximum fuel and cost efficiency is the main constraint in these applications. To minimize fuel consumption, the distance traveled by the UAV to reach the target needs to be reduced. An efficient path-planning approach can achieve this goal by reducing the unnecessary distance traveled by the UAV. Here, we propose a shortest-distance prioritization policy for Q-learning which marginally reduces the distance traveled by the UAV as compared to the original Q-learning with ϵ-greedy policy. The proposed shortest-distance prioritization policy is described in detail in Algo. 4. At each state, when the agent needs to choose an action that provides the maximum reward, the algorithm checks the distance criteria too. At stage s_t, the agent has to choose an action a_t from the action space [Up, Down, Left, Right]. Then, the algorithm determines the location, the agent has to reach, by taking each of these actions and calculates the corresponding distance to the target location from each of these locations. The action that makes the agent closer to the target point will get more weight than the other actions. If the agent reaches the target position, the reward value is set as 1. If the agent reaches an obstacle (static or dynamic) the reward set to -1. In all the other cases, the reward is set as 0. This way, the agent learns about the location of the target point in the environment, which leads to a reduction in learning time and also the distance that the UAV has to travel.

4.3. Performance analysis of the original Q-learning-based path planning approach for UAVs in an environment with only static obstacles

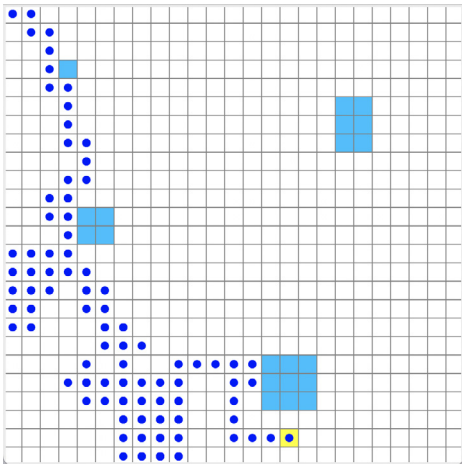
The effectiveness of the original Q-learning algorithm depends on the type of obstacle present in the environment. However, Q-Learning is a powerful algorithm that can successfully handle a variety of obstacles. The steps followed by the agent to reach the destination are described in Algo. 1. The process starts by initializing the source location and the target location along with the solution space, S states, and A actions. For each episode, the agent selects a state randomly from S and an action from A by using a suitable policy. Based on the penalty or reward obtained by the action, the Q-table gets updated until the agent reaches the destination. Q-Learning can effectively learn to avoid static obstacles by updating its Q-values based on the rewards received for successful and unsuccessful actions. The optimum path estimated with original Q-learning in our environment is given in Fig. 4(d). The steps and cost values with respect to the number of episodes are given in Fig. 6.



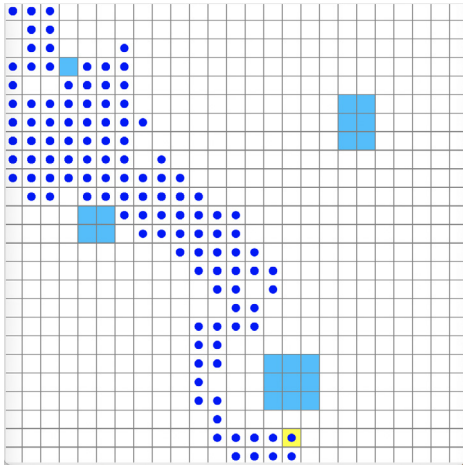
(a) A-star algorithm in the presence of static obstacles.



(b) Dijkstra algorithm in the presence of static obstacles.



(c) Sarsa algorithm in the presence of static obstacles.



(d) Original Q-learning algorithm in the presence of static obstacles

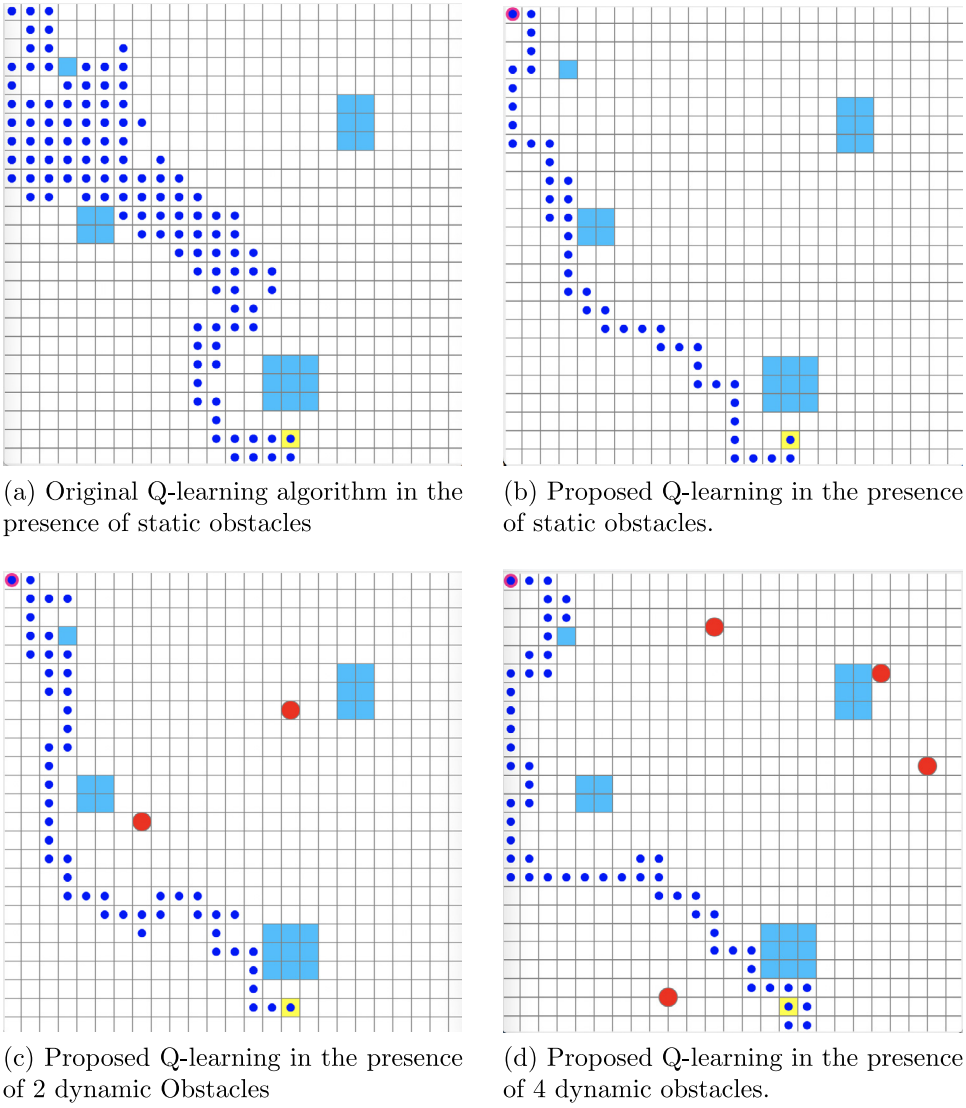
Fig. 4. Performance of various UAV path planning algorithms in the presence of static obstacles.

4.4. Performance analysis of the original Q-learning-based path planning approach for UAVs in an environment containing both static and dynamic obstacles

A significant advantage of Q-Learning is its ability to learn from prior experiences and adapt to evolving environments. This allows the algorithm to effectively handle dynamic obstacles, which change its position and velocity over time. By updating its Q-values based on the rewards obtained, the algorithm is capable of adapting to alterations in the environment. Therefore, we evaluate the effectiveness of the original Q-learning algorithm in navigating through both static and dynamic obstacles in the specified environment. The steps followed to reach the destination by the agent are described in Algo. 2. In each episode, the agent selects a state from the set of available states (S) and an action from the set of possible actions (A) using an appropriate policy. In addition, the location of the dynamic obstacles is continuously monitored. When the agent encountered a dynamic obstacle while learning, it breaks the loop. The path estimated by the algorithm is given in Fig. 5(a) and the corresponding steps and cost with respect to the episode are given in Fig. 7.

4.5. Performance analysis of the Q-learning-based path planning approach for UAVs in the presence of only static obstacles

In order to shape the reward signal, we need to modify the reward function to provide a more informative signal to the agent. For example, the reward can be increased when the UAV encounters a new obstacle or when it reaches a milestone on the way to the target. By designing the reward signal, the agent can accelerate its learning of the optimal policy. The proposed Shortest Distance Prioritization policy has brought a significant reduction in the distance traveled by the UAV to reach its destination. The steps followed to reach the destination by the agent following the Q-learning are described in Algo. 3. In addition to the usual steps of initialization and random selection of state and action, the policy calculates the distance to the target from all the possible next locations of the agent obtained by the action a_t. The policy gives more weight to the action that places the agent closer to the target. This in turn reduces the distance traveled by the agent to reach the target at each episode. The performance of the proposed Q-learning algorithm is assessed in an environment containing solely static obstacles, depicted in Fig. 5(b). The corresponding steps and cost with respect to the episode are given in Fig. 8.



(a) Original Q-learning algorithm in the presence of static obstacles

(b) Proposed Q-learning in the presence of static obstacles.

(c) Proposed Q-learning in the presence of 2 dynamic Obstacles

(d) Proposed Q-learning in the presence of 4 dynamic obstacles.

Fig. 5. Performance of the proposed approach in the presence of static and dynamic obstacles.

4.6. Performance analysis of the proposed Q-learning-based path planning approach for UAVs in an environment with both static and dynamic obstacles

Table 2 shows that in the presence of dynamic obstacles, the original Q-learning algorithm requires approximately ten times more time to estimate the optimal path compared to the scenario without dynamic obstacles. Hence, we evaluate the performance of the proposed Q-learning algorithm in navigation through environments that contain both static and dynamic obstacles. The results show that the proposed Q-learning algorithm is able to estimate the optimal path consistently, regardless of the number of dynamic obstacles present in the environment (i.e., two or four), in addition to the static obstacles as shown in Table 2. The steps followed to reach the destination by the agent with the proposed Q-learning are described in Algo. 4. The path traveled by the agent with the proposed Q-learning approach in this dynamic environment is depicted in Figs. 5(c) and 5(d). The corresponding step and cost with respect to the episodes are given in Figs. 9 and 10, respectively.

4.7. Performance measures

To evaluate the efficacy of our proposed approach, we conduct a comparative analysis with various state-of-the-art techniques based on the time required to execute and the length of the resulting path. The A-star algorithm is the first technique considered in our evaluation. This algorithm has been extensively researched in the literature and is widely used as a well-established path-planning technique. A-star algorithm is a graph traversal and heuristic search algorithm [39]. It runs fast in path searching and gives reliable real-time performance. The path estimated by the A-star algorithm in the same environment is given in Fig. 4(a). A similar comparison has been done with the Dijkstra algorithm as well. Dijkstra's algorithm is a classic and well-known algorithm used to find the shortest path between a source node and all other nodes in a graph [40]. It utilizes the weight of the edges of the graph to estimate the path that minimizes the total distance from the source node to all other nodes in the graph. Fig. 4(b) illustrates the path estimated by the Dijkstra algorithm for the given environment. The next algorithm we compared with is the State–action–reward–state–action (SARSA) algorithm [41]. It is a modified version of Q-learning, used to learn Markov decision

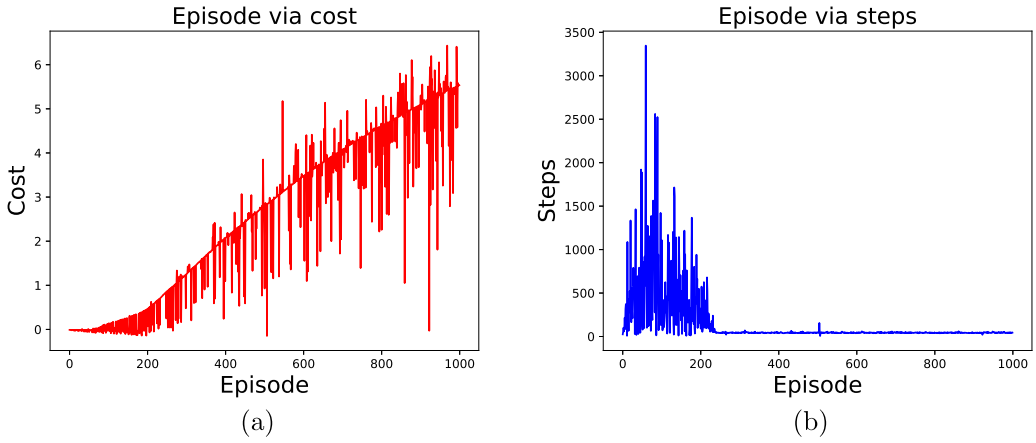


Fig. 6. Original Q-learning performance with no dynamic obstacles.

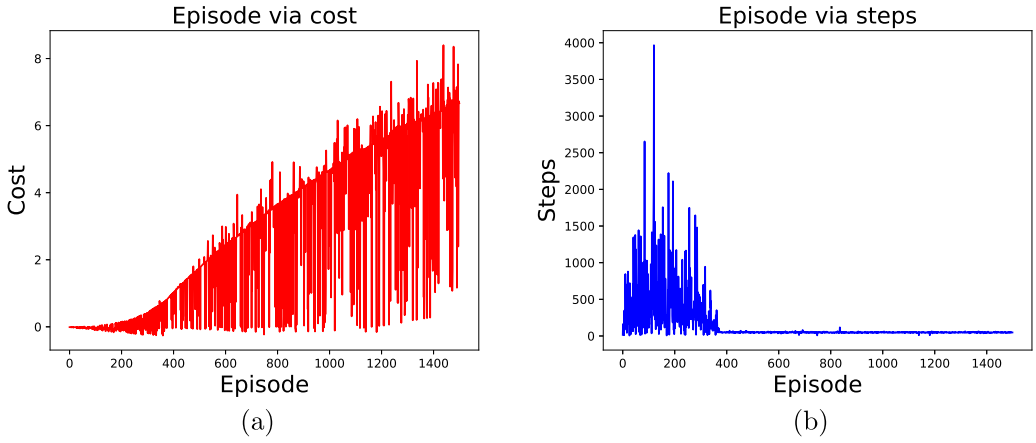


Fig. 7. Original Q-learning performance with dynamic obstacles.

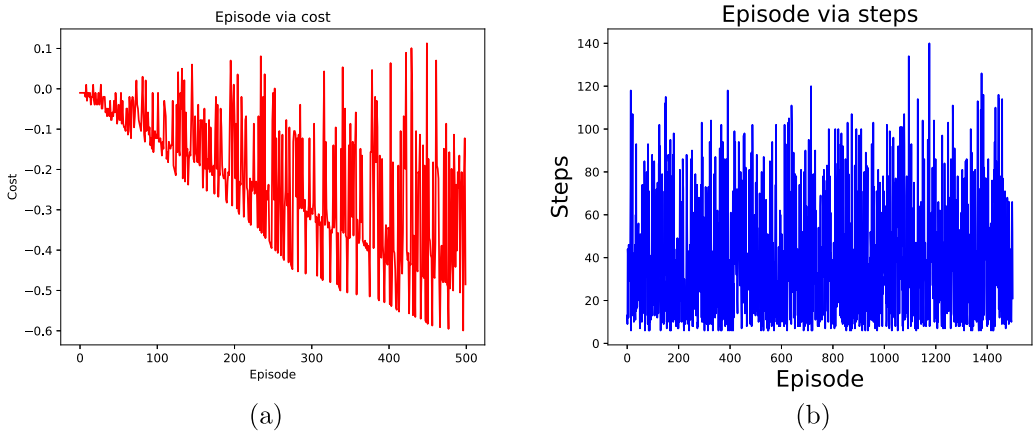


Fig. 8. Proposed Q-learning performance with No dynamic obstacles.

Table 2
Performance comparison of the proposed algorithms in terms of time and distance.

Parameter	Training time	Shortest distance	Longest distance
A-star [39]	0.01199 s	24	–
Dijkstra [40]	0.01559 s	37	–
SARSA [41]	1116.75 s	240	2087
Org. Q-learning [36]	347.2888 s	354	1025
Org. Q-learning with dynamic and static obstacles.	3128.7966 s	38	1794
Proposed Q-learning with only static obstacles.	326.4840 s	50	104
Proposed Q-learning with static and 2 dynamic obstacles.	336.5344 s	60	131
Proposed Q-learning with static and 4 dynamic obstacles.	357.3923 s	70	112

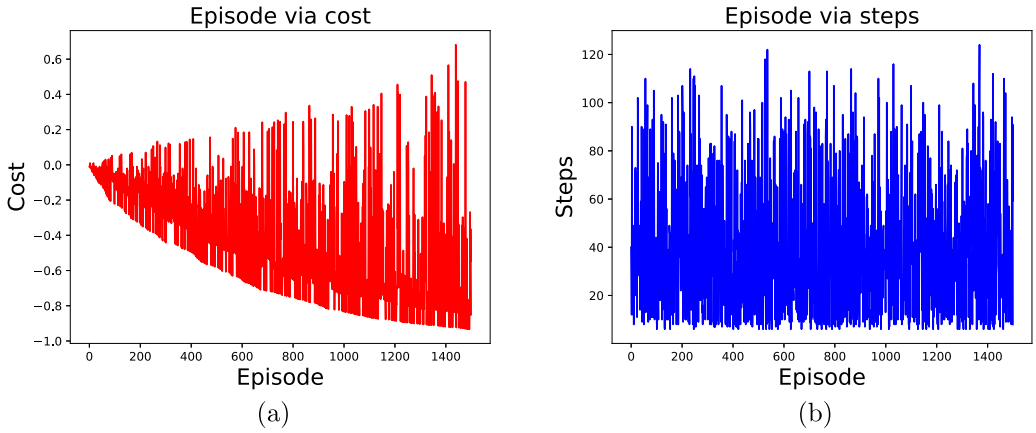


Fig. 9. Proposed Q-learning performance with 2 dynamic obstacles.

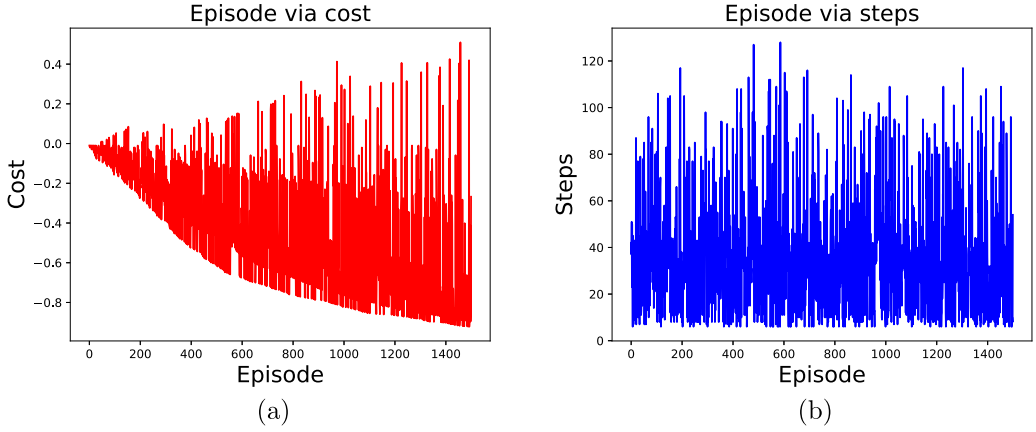


Fig. 10. Proposed Q-learning performance with 4 dynamic obstacles.

process policies in RL. The key distinction between Q-learning and SARSA is that Q-learning aims to maximize the expected reward in the future, whereas SARSA focuses on the reward that would result from the current action taken. In other words, Q-learning prioritizes future rewards, while SARSA prioritizes immediate rewards. SARSA follows the On-policy learning approach where the agent tries to learn from the current actions performed by it. Hence, it is independent of the previous learning, unlike the Q-learning algorithm. The path learned by the agent with the SARSA algorithm is given in Fig. 4(c). The next comparison is with the original Q-learning algorithm where the path estimated is given in Fig. 4(d). As some of these algorithms are not applicable to path planning with dynamic obstacles, all these algorithms are compared to our proposed approach under the assumption of only static obstacles present in the environment. The comparison of path length and training time is given in Table 2. As per Table 2, A-star and Dijkstra algorithms are faster with the shortest distance. However, as they are not applicable in environments with dynamic obstacles, they are not suitable for most real-time UAV applications. Further, the SARSA algorithm is slower in learning and contains more steps in the estimated path even though there are no dynamic obstacles present in the environment. The original Q-learning algorithm performs comparatively faster but with a longer path distance. Furthermore, the original Q-learning algorithm was observed to have the longest execution time among all the compared algorithms when navigating through an environment with dynamic obstacles. The proposed method shows good performance with respect to both the path length and the speed of execution in the presence of both static and dynamic obstacles.

5. Conclusion and future work

A Q-learning algorithm has been proposed to efficiently plan the path of UAVs in environments containing both static and dynamic obstacles. The proposed approach introduced the Shortest Distance Prioritization policy in the learning process which marginally reduces the distance that the UAV has to travel to reach the target point. The performance of the proposed approach was evaluated in terms of learning time and path length and compared with various state-of-the-art path planning approaches. It has been shown through the results that the proposed approach minimizes the total distance traveled by the UAV when compared to other approaches. The future plan is the integration of Q-Learning with other AI techniques to further improve the accuracy and efficiency of path planning. This will allow UAVs to better handle complex environments and make more informed decisions. The development of more advanced policies with Q-learning that can handle multiple objectives and constraints simultaneously, such as minimizing flight time, avoiding obstacles, and maximizing energy efficiency is also a possible future concern.

CRediT authorship contribution statement

Amala Sonny: Conceptualization, Methodology, Software, Writing – review & editing. **Sreenivasa Reddy Yeduri:** Writing

– review & editing, Supervision. **Linga Reddy Cenkeramaddi:** Conceptualization, Supervision, Reviewing and editing, Funding acquisition.

Declaration of competing interest

We, the authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article

Acknowledgments

This work was supported by the Indo-Norwegian Collaboration in Autonomous Cyber-Physical Systems (INCAPS), Norway project: 287918 of the INTPART program and the Low-Altitude UAV Communication and Tracking (LUCAT), Norway project: 280835 of the IKTPLUSS program from the Research Council of Norway.

References

[1] P. Chen, J. Pei, W. Lu, M. Li, A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance, Neurocomputing 497 (2022) 64–75.
[2] L. Yang, J. Qi, J. Xiao, X. Yong, A literature review of UAV 3D path planning, in: Proceeding of the 11th World Congress on Intelligent Control and Automation, 2014, pp. 2376–2381, http://dx.doi.org/10.1109/WCICA.2014.7053093.
[3] F. Borrelli, D. Subramanian, A. Raghunathan, L. Biegler, MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles, in: 2006 American Control Conference, 2006, p. 6, http://dx.doi.org/10.1109/ACC.2006.1657644.
[4] K. Fransen, J. Van Eekelen, A. Pogromsky, M.A. Boon, I.J. Adan, A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems, Comput. Oper. Res. 123 (2020) 105046.
[5] M. Kanehara, S. Kagami, J.J. Kuffner, S. Thompson, H. Mizoguhi, Path shortening and smoothing of grid-based path planning with consideration of obstacles, in: 2007 IEEE International Conference on Systems, Man and Cybernetics, 2007, pp. 991–996, http://dx.doi.org/10.1109/ICSMC.2007.4414077.
[6] J. Barraquand, B. Langlois, J.-C. Latombe, Numerical potential field techniques for robot path planning, in: Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments, 1991, pp. 1012–1017 vol.2, http://dx.doi.org/10.1109/ICAR.1991.240539.
[7] F. Bounini, D. Gingras, H. Pollart, D. Gruyer, Modified artificial potential field method for online path planning applications, in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 180–185, http://dx.doi.org/10.1109/IVS.2017.7995717.
[8] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, J. Nieto, An efficient sampling-based method for online informative path planning in unknown environments, IEEE Robot. Autom. Lett. 5 (2) (2020) 1500–1507.
[9] L. Jaillet, J. Cortés, T. Siméon, Sampling-based path planning on configuration-space costmaps, IEEE Trans. Robot. 26 (4) (2010) 635–646, http://dx.doi.org/10.1109/TRO.2010.2049527.
[10] J. Ji, A. Khajepour, W.W. Melek, Y. Huang, Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints, IEEE Trans. Veh. Technol. 66 (2) (2017) 952–964, http://dx.doi.org/10.1109/TVT.2016.2555853.
[11] C. Liu, S. Lee, S. Varnhagen, H.E. Tseng, Path planning for autonomous vehicles using model predictive control, in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 174–179, http://dx.doi.org/10.1109/IVS.2017.7995716.
[12] J. Li, G. Deng, C. Luo, Q. Lin, Q. Yan, Z. Ming, A hybrid path planning method in unmanned air/ground vehicle (UAV/UGV) cooperative systems, IEEE Trans. Veh. Technol. 65 (12) (2016) 9585–9596, http://dx.doi.org/10.1109/TVT.2016.2623666.
[13] F.H. Tseng, T.T. Liang, C.H. Lee, L.D. Chou, H.C. Chao, A star search algorithm for civil UAV path planning with 3g communication, in: 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2014, pp. 942–945, http://dx.doi.org/10.1109/IIH-MSP.2014.236.
[14] Z. He, L. Zhao, The comparison of four UAV path planning algorithms based on geometry search algorithm, in: 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Vol. 2, 2017, pp. 33–36, http://dx.doi.org/10.1109/IHMSC.2017.123.
[15] Y. Deng, Y. Chen, Y. Zhang, S. Mahadevan, Fuzzy dijkstra algorithm for shortest path problem under uncertain environment, Appl. Soft Comput. 12 (3) (2012) 1231–1237.
[16] B. Li, B. Chen, An adaptive rapidly-exploring random tree, IEEE/CAA J. Autom. Sin. 9 (2) (2022) 283–294, http://dx.doi.org/10.1109/JAS.2021.1004252.
[17] L. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (4) (1996) 566–580, http://dx.doi.org/10.1109/70.508439.
[18] A. Ma'arif, N.M. Raharja, G. Supangkat, F. Arofiati, R. Sekhar, D.U. Rijalusalam, et al., Pid-based with odometry for trajectory tracking control on four-wheel omnidirectional covid-19 aromatherapy robot, Emerg. Sci. J. 5 (2021) 157–181.
[19] I. Suwarno, A. Ma'arif, N.M. Raharja, T.K. Hariadi, M.A. Shomad, Using a combination of PID control and Kalman filter to design of IoT-based telepresence self-balancing robots during COVID-19 pandemic, Emerg. Sci. J. 4 (2020) 241–261.
[20] A. Sonny, S.R. Yeduri, L.R. Cenkeramaddi, Autonomous UAV path planning using modified PSO for UAV-assisted wireless networks, IEEE Access (2023) 1, http://dx.doi.org/10.1109/ACCESS.2023.3293203.
[21] S. Aggarwal, N. Kumar, Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges, Comput. Commun. 149 (2020) 270–299.
[22] S.I.A. Meerza, M. Islam, M.M. Uzzal, Q-learning based particle swarm optimization algorithm for optimal path planning of swarm of mobile robots, in: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019, pp. 1–5, http://dx.doi.org/10.1109/ICASERT.2019.8934450.
[23] K.B. de Carvalho, I.R.L. de Oliveira, D.K.D. Villa, A.G. Caldeira, M. Sarcinelli-Filho, A.S. Brandão, Q-learning based path planning method for UAVs using priority shifting, in: 2022 International Conference on Unmanned Aircraft Systems (ICUAS), 2022, pp. 421–426, http://dx.doi.org/10.1109/ICUAS54217.2022.9836175.
[24] A. Konar, I. Goswami Chakraborty, S.J. Singh, L.C. Jain, A.K. Nagar, A deterministic improved Q-learning for path planning of a mobile robot, IEEE Trans. Syst. Man Cybern.: Syst. 43 (5) (2013) 1141–1153, http://dx.doi.org/10.1109/TSMCA.2012.2227719.
[25] C. Yan, X. Xiang, A path planning algorithm for UAV based on improved Q-learning, in: 2018 2nd International Conference on Robotics and Automation Sciences (ICRAS), 2018, pp. 1–5, http://dx.doi.org/10.1109/ICRAS.2018.8443226.
[26] D. Li, W. Yin, W.E. Wong, M. Jian, M. Chau, Quality-oriented hybrid path planning based on A* and Q-learning for unmanned aerial vehicle, IEEE Access 10 (2022) 7664–7674, http://dx.doi.org/10.1109/ACCESS.2021.3139534.
[27] Z. Yijing, Z. Zheng, Z. Xiaoyi, L. Yang, Q learning algorithm based UAV path learning and obstacle avoidence approach, in: 2017 36th Chinese Control Conference (CCC), 2017, pp. 3397–3402, http://dx.doi.org/10.23919/ChiCC.2017.8027884.
[28] J.-h. Cui, R.-x. Wei, Z.-c. Liu, K. Zhou, UAV motion strategies in uncertain dynamic environments: A path planning method based on Q-learning strategy, Appl. Sci. 8 (11) (2018) 2169.
[29] Y. Gao, Y. Li, Z. Guo, A Q-learning based UAV path planning method with awareness of risk avoidance, in: 2021 China Automation Congress (CAC), 2021, pp. 669–673, http://dx.doi.org/10.1109/CAC53003.2021.9728342.
[30] M.L. Puterman, Markov decision processes, Handb. Oper. Res. Manag. Sci. 2 (1990) 331–434.
[31] R.S. Sutton, Generalization in reinforcement learning: Successful examples using sparse coarse coding, in: Advances in Neural Information Processing Systems, Vol. 8, 1995.
[32] A.R. Cassandra, Exact and Approximate Algorithms for Partially Observable Markov Decision Processes, Brown University, 1998.
[33] M.L. Littman, Value-function reinforcement learning in Markov games, Cogn. Syst. Res. 2 (1) (2001) 55–66.
[34] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, Vol. 12, 1999.
[35] C. Ye, J. Borenstein, A method for mobile robot navigation on rough terrain, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, Vol. 4, 2004, pp. 3863–3869, http://dx.doi.org/10.1109/ROBOT.2004.1308870, Vol.4.
[36] C. Wang, X. Yang, H. Li, Improved Q-learning applied to dynamic obstacle avoidance and path planning, IEEE Access 10 (2022) 92879–92888, http://dx.doi.org/10.1109/ACCESS.2022.3203072.
[37] M. Langlois, R.H. Sloan, Reinforcement learning via approximation of the Q-function, J. Exp. Theor. Artif. Intell. 22 (3) (2010) 219–235.

[38] E.I. Grøtli, T.A. Johansen, Path planning for UAVs under communication constraints using SPLAT! and MILP, J. Intell. Robot. Syst. 65 (1–4) (2012) 265–282.

[39] G. Tang, C. Tang, C. Claramunt, X. Hu, P. Zhou, Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment, IEEE Access 9 (2021) 59196–59210, http://dx.doi.org/10.1109/ACCESS.2021.3070054.

[40] M. Luo, X. Hou, J. Yang, Surface optimal path planning using an extended dijkstra algorithm, IEEE Access 8 (2020) 147827–147838, http://dx.doi.org/10.1109/ACCESS.2020.3015976.

[41] H. Boming, L. Wei, M. Fuzeng, F. Huahao, Research for UAV path planning method based on guided sarsa algorithm, in: 2022 IEEE 2nd International Conference on Software Engineering and Artificial Intelligence (SEAI), 2022, pp. 220–224, http://dx.doi.org/10.1109/SEAI55746.2022.9832224.