IJRRS
International Journal of Reliability, Risk and Safety: Theory and Application

Online ISSN: 2676-3346

OPEN ACCESS

Vol. 6/ Issue 2/ 2023/ pp. 93-105
DOI: 10.22034/IJRRS.2023.6.2.11
Received: 03 November 2023, Revised: 24 December 2023, Accepted: 25 December 2023
Available online at: https://www.ijrrs.com

Original Research Article

Target Interception in Uncertain Environment Using ART2-Based Reinforcement Learning

Mostafa Abbasi Kia[1]*, Shahram Khoshnavaz[2] and Razieh Hashemi Alem[3]

1-Department of Computer Science, Faculty of Basic Sciences, University of Lorestan, Khorramabad, Iran
2- Department of Computer Engineering, Faculty of Technology and Engineering, University of Lorestan, Khorramabad, Iran
3- Department of Computer Engineering, Faculty of Technology and Engineering, University of Arak, Arak, Iran
*** abbasikia.m@lu.ac.ir**

Abstract

Tracking moving targets using mobile robots is a crucial aspect of robotics. This paper presents a novel approach for tracking a moving target in an uncertain environment with various obstacles, even when the target's trajectory and speed are continuously changing and unknown. The proposed method utilizes reinforcement learning, a widely used technique for motion planning problems. However, applying reinforcement learning in uncertain dynamic environments poses a challenge due to the continuous state space. To address this issue, our algorithm employs an ART2 neural network for classifying the state space. Additionally, to enhance the speed of reaching the target, a point is predicted based on the target's speed, direction, and the robot's speed. The robot then selects its next move to approach this predicted point while avoiding contact with both static and dynamic obstacles. Simulation results demonstrate the efficiency of the algorithm, as the robot successfully reaches the target without colliding with any obstacles.

Keywords: Target Interception; Uncertain environment; Reinforcement Learning; ART2 Neural Network.

Nomenclature and Units

NN	Neural Network
ART	Adaptive Resonance Theory
RL	Reinforcement Learning
SS	Safe State
FS	Failure State
NSS	Non-Safe State
NNSS	Near-Non-Safe State
RRT	Rapidly-Exploring Random Trees

1. Introduction

Today, most of the exact, repetitive, and difficult or impossible tasks for humans are done by robots. One of the key features of automated robots is their ability to move and perform specific tasks. Motion planning is usually used to change the world situation by calculating a sequence of acceptable movements for the robot. For example, in path planning, a path without obstacles is calculated so that the robot can move from its initial position to its final position. This is the simplest issue in motion planning, but it requires high computational complexity [1].

In some cases, the goal of motion planning is not to change the world but rather to maintain certain conditions related to the world or to achieve a specific understanding of the environment. There are various issues in autonomous navigation., including environment exploration, environment investigation, navigation among movable obstacles (NAMO), mapping, localization, motion sequencing, energy consumption, and time limits. Tracking moving targets is also an important issue in motion planning, which arises in various applications such as robot soccer, automatic conduct, and caring systems [3][5].

In most real-world applications, the direction and speed of moving targets are not known and constantly changing. Additionally, the environment in which the robot operates may be unfamiliar and contain both static and dynamic obstacles. Under these circumstances, effectively tracking and reaching a target in the shortest possible time and path is a challenging problem that has yet to be properly solved. This paper presents a new method for robot motion planning in an uncertain

How to cite this article:
M. Abbasi Kia, Sh. Khoshnavaz and R. Hashemi Alem, "Target Interception in Uncertain Environment Using ART2-Based Reinforcement Learning," *International Journal of Reliability, Risk and Safety: Theory and Application*, vol. 6, no. 2, pp. 93-105, 2023.

dynamic environment with unknown static and dynamic obstacles. The goal is to reach a moving target whose speed and direction are unknown and constantly changing. The proposed method has been implemented and validated through various scenarios [7].

The paper is organized as follows: Section 2 provides a literature review on target tracking. In Section 3, we explain the problem and its characteristics. The main components of the proposed algorithm are described in Section 4. Section 5 presents the developed algorithm for target tracking in a dynamic and uncertain environment, and the simulation results are presented in Section 6.

2. Literature review

Target tracking issues have been studied in several studies. Fuzzy logic is one of the most widely used methods in target tracking. In [20], the authors presented an adaptive control algorithm in which they combine a fuzzy logic controller with a predictive method of position according to gray theory to track a moving target by a mobile robot. In [4], a target tracking control model was designed based on a real-time fuzzy method to distinguish related tracking behavior. Fuzzy control methods have various benefits but disadvantages. For example, it is necessary to establish appropriate fuzzy control rules concerning the fuzzy control relations of the system. When the environment is dynamic, and the target is moving, pre-built fuzzy rules will be inefficient [2].

Artificial neural networks (ANNs) are another approach to solving robot motion planning problems. In [17], researchers used neural networks to enable a robot to perceive the environment and perform feature extraction, which enabled them to determine the fitness of the environment for state action functions. Through hierarchical reinforcement learning, mobile robot needs are met by mapping the current state of these actions. The proposed algorithm performs well in all aspects and exhibits outstanding performance.

The potential field method was used to plan the speed of a moving robot while tracking a moving target [21]. Potential field methods for motion planning of moving robots in a static environment have considerable shortcomings that have not yet been resolved. These shortcomings are even more problematic when targets and obstacles are moving. For example, a robot could easily get stuck in a local optimum or deadlock, causing fluctuation.

Visual methods are another group of more applicable methods for solving target-tracking problems. By considering visual feedback, the authors of [6] have shown how to capture a moving target using a nonholonomic robot. One requirement of these algorithms is that the moving target should constantly be within sight of cameras or the sensing domain of other sensors. Otherwise, these methods will be defeated. Therefore, an approach is necessary to make a decision when the target cannot be seen or felt by the robot.

In [2], the ant colony algorithm is presented for tracking a moving target whose speed and direction are unknown and constantly changing. In this paper, the environment is known in advance, and all obstacles are considered fixed. In addition, in [30], it is assumed that the speed and direction of the target are known. However, the speed of dynamic obstacles and the location of static obstacles are unknown and calculated by the robot in an online manner. In this paper, a new concept, named Directive Cycle, is introduced and used to guide the robot's movements.

One of the challenges in using Q-learning for path planning is the large state-action space, which can slow down the algorithm and make it impractical. Another challenge is to plan the most efficient path while learning about the environment. To address this challenge, [22] 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 thoroughly studying the uncertain environment. The proposed work in [22] did not consider dynamic obstacles in the environment. In addition to all the challenges in unmanned aerial vehicle (UAV) path planning, the priorities of the UAV can be different, such as safety, energy consumption, and distance to the target to maintain 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 for path planning. In this approach, the policy concentrates on path length, safety, and energy consumption to determine the reward.

A new deterministic Q-learning approach for a mobile robot has been proposed [24]. This approach uses prior knowledge of the distance to the next state and the goal from the current state. This efficient use of information made the entire path-planning process less time-consuming than other approaches. An improved Q-learning approach has been proposed in [25] for UAV path planning in an unknown antagonistic scenario. In [26], a Q-function initialization method, along with a new action selection strategy, is introduced to enhance performance. A-star and Q-learning-based hybrid path planning has been proposed in [26], which analyzes 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].

There are still unresolved problems in this field. For instance, in certain algorithms, the robot follows the actual path of the moving target, which means that it cannot reach the target as quickly as possible. Some

algorithms are not suitable for situations where the direction of the target is unknown and rapidly changing or when the environment contains complex obstacles. In addition, some algorithms cannot make immediate decisions regarding robot movements [2]. Therefore, further research is required to resolve these issues.

3. Problem statement

The research field of motion planning involves various issues. These issues are expressed in different situations and classified based on criteria such as obstacle types, robot shape, movement limitations, environment dimensions, number of robots, environment changes over time, information source, and reliability of information [1]. Therefore, it is important to describe the topic and the specific conditions being considered accurately.

Robot motion planning problems can be divided into two categories based on the robot's knowledge of its surrounding environment: online and offline. In offline mode, the robot has complete environmental information and plans its moves accordingly. Essentially, the robot is aware of the environment in this situation. On the other hand, in online motion planning, the robot has very limited or no information about the environment and obstacles. In this type of problem, various sensors are used to gather information about the environment, and the robot's path is designed based on this increasing information as it moves. Updating information and redesigning the path is continuously performed until the robot reaches its target [8].

In this paper, a new algorithm is presented for solving the target tracking problem in a two-dimensional uncertain environment. The algorithm is designed to handle a target whose speed and direction are constantly changing and unknown. Additionally, the algorithm assumes that the locations of both static and dynamic obstacles, as well as the speed and direction of dynamic obstacles, are unknown and need to be computed by the robot in real time. The robot knows the initial location of the target but lacks information about its speed and direction.

To ensure successful tracking of the target, the algorithm assumes that the robot's speed is always greater than that of the target and obstacles. The robot is capable of moving in eight different directions. The main focus of the presented algorithm is the ability to make immediate decisions about the robot's next move, considering the high dynamics of the environment and the target.

4. Proposed algorithm's components

The algorithm proposed in this paper consists of three main components: reinforcement learning, ART2 neural network, and predicting target tracking points. ART2 is used to detect the robot's status concerning the environment, based on its distance from obstacles and its relative position regarding the target, to determine the correct movement of the robot. During the learning process, the robot gradually learns to take the best possible moves in each state, using reinforcement learning. By combining ART2 and reinforcement learning, a new neural network called QLART2 has been introduced.

To accelerate the achievement of the target, the predicted point to achieve the target is used instead of the current position of the target to determine the robot's status in the environment. In this section, we introduce the algorithm's components.

4.1 Reinforcement learning

Because of the unpredictable nature of dynamic and uncertain environments, most of the proposed solutions for navigating robots in such environments do not work in every situation. As a result, artificial intelligence approaches have acquired attention from researchers in recent years, aiming to improve robots' ability to navigate based on the knowledge acquired through experiments in the work environment.

The reinforcement learning algorithm is an artificial intelligence method in which a learning agent can acquire correct behavior by interacting with the environment [31]. In this method, the agent and the environment interact with each other in a sequence of discrete time steps, denoted as t=0, 1, …. At each time step t, the agent receives the state of the environment, represented as $\boldsymbol{s_t} \in \boldsymbol{S}$, where S is the set of all possible states.

Then, it chooses an action, $a_t \in A(s_t)$, based on the current state, where $A(s_t)$ is the set of all actions that the robot can choose in the state s_t. In the next step, the agent receives a numerical reward, r_t, and finds itself in a new state, s_{t+1}. The goal of the learning agent is to maximize the sum of received rewards in the long term [9].

In [10], the Q-learning method of the reinforcement algorithm is used to avoid collision of the robot with obstacles. In the reward function used in this study, the value of the target tracking state is 1, the value of the collision with obstacles state is -1, and the value of the other states is 0. In this study, the distances between the robot and the target and obstacles were used as input.

Reinforcement learning has been combined with other techniques like fuzzy logic and neural networks in certain algorithms. An example in [11] presents a hybrid approach that utilizes fuzzy logic and reinforcement learning. It is composed of two fundamental components: obstacle avoidance and target attainment. The behaviors were created separately but merged to select the appropriate behavior during implementation. Both of these behaviors are fuzzy engines that map the

environment state to a fuzzy output, which represents the desired operation. In this method, fuzzy rules are constructed using reinforcement learning.

A combination of the probabilistic roadmap approach and the Q-learning algorithm was used for a dynamic environment [12]. If an obstacle blocks the designed path, the Q-learning algorithm determines the best possible action. Based on this work, it appears that reinforcement learning is a good approach to solving motion planning problems in unknown or uncertain environments.

4.1.1 Q-learning

Q-learning is one of the most common methods of reinforcement learning. In this method, Q is an action-value function that approximates the optimal action-value function Q* [32]. The Q values are updated as follows:

$$Q(s,a) = Q(s,a) + \alpha\left[r + \gamma \max_{a'}(s',a') - Q(s,a)\right] \quad (1)$$

Where $Q(s,a)$ is an action-value function of taking action in state s. $Q(s,'a')$ is an action-value function in the resulting state(s') after acting, α is the step-size parameter between 0 and 1, r is an immediate reward and γ is the descending rate [9]. According to (1), $Q(s,a)$ is calculated according to the immediate reward, r, and delayed reward. In this formula, delayed reward, $\gamma \max_{a'}(s',a') - Q(s,a)$, is related to subtracting the maximum achievable value in the next state and the value of the current state.

If, in each state, a sufficient number of actions are executed in that case, the Q values will converge to the optimal value Q* with probability 1. Q-learning can also be extended to update states that have occurred in more than one step [19]. Once sufficiently trained, learning agents can take appropriate actions in any state.

4.2 ART2 Neural Network

Neural networks are one of the most efficient methods of pattern recognition and have been used in various fields because of their high efficiency in learning training data. Neural networks, which attempt to create a mapping between input and output patterns, are suitable tools for classification and clustering because of their ability in parallel processing and nonlinear characteristics [16]. One of the key issues in neural networks, including RBF and MLP, is the fixed number of categories or patterns in most of them.

Adaptive resonance theory (ART) neural networks can classify arbitrary sequences of input patterns in a stable and self-organized manner. If a new input pattern is used, ART detects it as a new pattern and adds it to the existing patterns without any difficulty in identifying previous patterns. ART is an unsupervised neural network. This type of NN can automatically increase its memory when sample patterns increase. They are able to acquire new knowledge without losing the previous patterns. These NNs can automatically increase their memory when patterns of samples increase. They perform steady classification through unsupervised competitive learning and self-organization mechanisms and can realize online learning with various data types and an unknown number of classes.

The learning agent can classify various states of the environment during the learning phase using ART2 NN in the Q-learning algorithm. In addition, the learning agent can learn appropriate behavior in every state. In this study, we used the architecture shown in Figure 1 for ART2.

Suppose input vector X is an analog N-dimension vector, then $X = \{x_0, x_1, \dots, x_{n-1}\}$.There are N processing unit in layer $F1$, each of which has six neurons, $z_i, q_i, u_i, v_i, p_i, s_i$. Each of these neurons has two filters which are used to improve the features and to eliminate noises of the input signal. The input signal, which is processed in the $F1$ layer, will be transferred to the $F2$ layer via bottom-up weights that are called long-term memory. In the F2 layer, the winning neuron,j^*, is selected through competition with other neurons. Then, the top-down weight coefficient of the neuron j^*, i.e., w_{j^*i}, gives signals to p_i node of $F1$ layer.

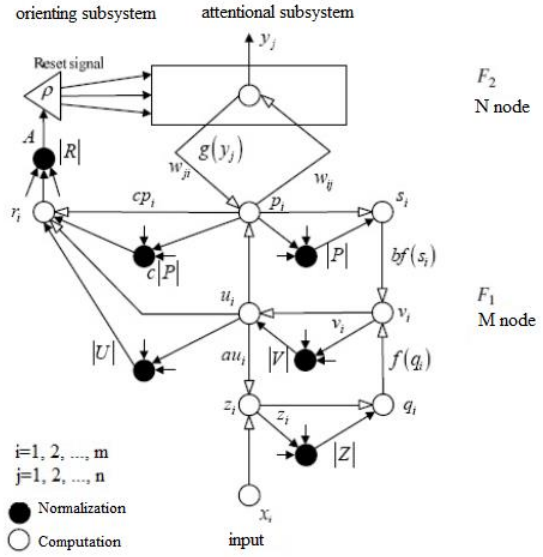


Figure 1. ART2 NN Architecture [14]

At this point, the node p_i is equal to a linear combination of the normalized noise-less input, u_i, and the center of the winner node's pattern, w_{j^*i}. The similarity between the input vector and the center of the winner node's pattern is evaluated by calculating the norm of node r. If the similarity is less than a predefined threshold ρ, then the orienting subsystem is activated and sends a reset signal to layer $F2$. As a result, the attention subsystem ignores the winner node. It repeats the

competition between other nodes, each representing a pattern of the inputs, to find a pattern that has an adequate match with the input. If the input vector is not sufficiently similar to any of the $F2$ layer's nodes, it is added to the $F2$ layer as a new pattern and chosen as the winner node. Here, the winner's weights are updated.

4.3 ART2 NN based reinforcement learning

In motion planning problems, the robot operates in a continuous state space, whereas reinforcement learning algorithms are typically used to solve Markov behavior problems with discrete and limited information. Applying RL algorithms to a continuous state space may lead to dimensionality issues. To address this problem, we incorporate the ART2 neural network alongside Q-learning to incrementally cluster the continuous state space of an uncertain environment through interaction with the environment online.

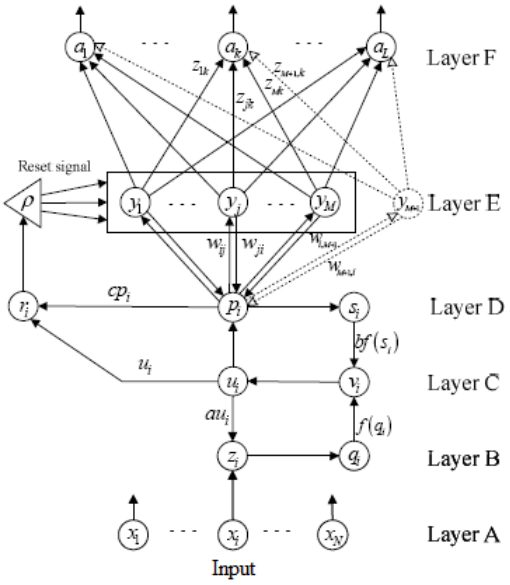


Figure 2. QLART2 NN architecture [14]

Figure 2 illustrates the structure of the Q-learning algorithm based on ART2 NN, also known as QLART2. This structure consists of six layers. Layer A serves as the input layer and represents the state space to be detected by the learning agent. Layers B, C, and D are responsible for noise reduction and feature enhancement processing of the input. Layer E serves as a pattern recognition layer. Layer F is the output layer, where each neuron represents one of the decisions that the learning agent can make.

Each neuron in layer E, which corresponds to a pattern in the state space, is connected to all neurons in layer F. The weights of these connections are denoted as z_{jk}, represent the estimated value for the action-value function $Q(y_j, a_k)$, where each weight corresponds to a specific pair of state space-behavior space patterns [15]. The general steps of the QLART2 algorithm are as follows:

- Applying the vector of the environment state to the input layer enhances features and decreases the noises of the input vector through layers B, C, and D.
- Applying modified input vector to layer E for doing competitive learning and selecting appropriate space state patterns.
- Competition in layer E to find winner behavior based on bottom-up weights, which can be adjusted based on Q-learning.
- The evaluation of the results of selected behavior in the environment by the learning agent and the identification of the state in which the learning agent, after performing the selected behavior, has been entered. Then, the new state space is detected by the A to F layers.
- The Modification of weights among winner neurons of F and E layers, based on immediate reward, which resulted from selected behavior, and delayed reward, which was generated by following optimal policy.

Figure 3 shows the general steps of the algorithm.

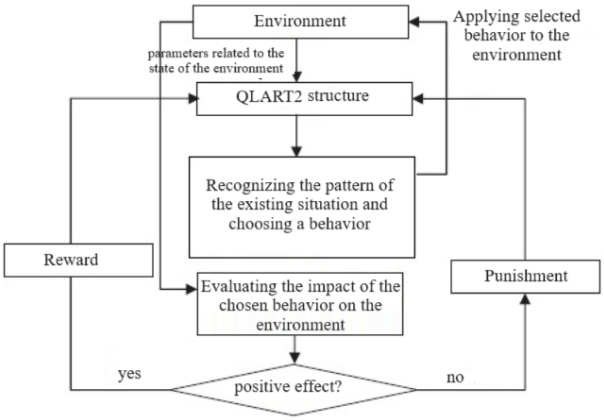


Figure 3. General steps of the QLART2 algorithm

Now, we describe the various steps of the algorithm in more detail.

1. Initializing parameters: The neurons in each of the six layers A, B, C, D, E, and F are initialized to zero. The coefficients of layers B, C, and E, denoted as a, b, and e, respectively, are initialized with arbitrary values. The input vector is N-dimensional. The filter factor and the number of neurons in layer E are set to $\theta = \frac{1}{\sqrt{N}}$ and $M = 1$, respectively. The number of neurons in layer F is equal to the number of

possible behaviors. If i and j represent neurons in layers D and E, respectively, the weight vectors between layers D and E are initialized as follows:

$$w_{ij} = 0,\ w_{ij} = \frac{1}{(1-d)\sqrt{N}} \quad (2)$$

2. The learning process is carried out by following steps 3-15 for each learning scenario. Increasing the number of learning scenarios brings the robot closer to its optimal behavior. However, the number of scenarios should not exceed a certain limit to avoid overfitting and reducing the robot's optimal efficiency. Each scenario concludes when a stop condition is met. The stop condition can be based on a specific time limit, a certain number of inputs applied to the network, or the completion of a specific task, such as placing the robot in the final state X_{end}.
3. The robot is placed in the initial state space, X_0, and the timer becomes zero.
4. At time t, the robot enters the state X_t, where $X_t = x_{0_t}, x_{1_t}, \dots, x_{N_t}$ represents the input to layer A. If X_t corresponds to the termination state X_{end}, it indicates that the robot has completed the current learning scenario and returns to Step 3. Otherwise, the process continues to Step 5.
5. Each neuron in the B layer is updated as follows:

$$z_i = x_i + au_i, q_i = \frac{z_i}{(e+||Z||)} \quad (3)$$

6. Neurons of layer C are updated as follows:

$$v_i = f(q_i) + bf(s_i), u_i = \frac{v_i}{(e+||V||)}$$
$$f(x) = \begin{cases} 0, & 0 \le x \le \theta \\ x, & x \ge \theta \end{cases} \quad (4)$$

7. The neurons in layer C are updated as follows:

$$p_i = u_i, u_i = \frac{p_i}{(e+||P||)} \quad (5)$$

8. Values of M neurons in layer E are determined as follows:

$$y_j = \sum_{i=1}^{N} p_i\, w_{ij} \quad (6)$$

9. The winner neuron in layer E is determined through a competitive mechanism, where $y_{j^*} = max\, y_j | j = 1, 2, \dots, M$. Afterwards, the feedback signal, w_{ji}, is sent to p_i neurons of the D layer:

$$g(y_j) = \begin{cases} d, & j = j^* \\ 0, & j \ne j^* \end{cases}$$
$$p_j = u_i + \sum_{j=1}^{M} g(y_j)\, w_{ji} \quad (7)$$

10. At this point, the modifying threshold is tested. Values of r_i neurons of layer D are computed as follows, based on u_i node and modified p_i node:

$$r_i = \frac{u_{i+}cp_i}{(e+||U||+c||P||)} \quad (8)$$

where,

$$||R|| = \sqrt{\sum_{i=1}^{N} r_i^2}$$

If $||R|| \le \rho$, then the current winner neuron is reset, and this process is repeated for the remaining nodes of layer E. The algorithm continues from step 11. Otherwise, the algorithm continues with step 12.

11. If all the neurons of layer E are reset, it means that the input vector is not similar enough to any of the previously identified state space patterns. Therefore, a new neuron, y_{M+1}, is created in layer E as the winner node.
12. The input-output weight vectors of the E layer's winner node are updated as below:

$$w_{j^*i} = du_i + [1 - d(1-d)]w_{j^*i}, \forall i$$
$$w_{ij*} = du_i + [1 - d(1-d)]w_{ij^*}, \forall i \quad (9)$$

13. a_{k^*} is selected as output action among the layer F nodes based on a defined mechanism or a probability function.
14. After performing action a_{k^*} in the environment, the robot moves to the next state, $X(t+1)$. Then, an immediate return, which is the reward or punishment of a selected action in state $X(t)$, is computed according to the Q-learning evaluation function.
15. Steps 5-12 are repeated to identify the winning state space pattern at the moment $t+1$, $y_{j^*}(t+1)$. Then, the maximum value for $Q(y_{j^*}(t+1), a)$, which corresponds to the maximum value of $z_{j^*k}(t+1)$ is estimated. Based on this, all weights of F are updated as follows:

$$z_{jk}(t) = z_{jk}(t-1) + \alpha[r(t) + \cdots$$
$$\gamma \max_{a_k} \left(z_{j^*k}(t+1)\right) - z_{jk}(t)], \forall j, k \quad (10)$$

In this step, in fact, the robot enters into the learning process of the next input vector, e.g., $X(t+1)$, and its corresponding state space pattern has been detected. So, the learning process is continued from step 13 [2].

4.4 Predicting the target achieving point

In this study, to increase the speed of reaching the target, a predicted point was used as a sub-target for moving regulation. In this method, when the robot sees the target, according to the speed and direction of the target and its speed, it predicts the point for reaching it. This predicted point is the secondary purpose of robot transfer. In this method, the dynamic and desired path of the target is estimated using many short linear paths.

4.4.1 An algorithm to predict the achieving point of the goal

Suppose t_0 is the time required to identify the target by the robot. During target tracking, when the target moves to a new location, the robot detects it. Suppose the times of these moves are $t_0, t_1, \ldots$, and the location of the target in time t_i is $G_i = G_i(x_{G_i}(t_i), y_{G_i}(t_i))$. In each t_i, the robot can compute the target speed using the following formula:

$$V_{G_i} = \frac{d(G_{i-1}, G_i)}{d(t_i, t_{i-1})} \tag{11}$$

Where $d(x, y)$ is the distance between x and y. Both consecutive cells traversed by the target create a linear path. These linear paths estimate the curve route of the target and are calculated as follows:

$$y = \frac{(y_{G_i} - y_{G_{i-1}})}{(x_{G_i} - x_{G_{i-1}})} \times (x - x_{G_{i-1}}) + y_{G_{i-1}} \ldots = k_i x + b_i \tag{12}$$

Because the estimated procedures are performed dynamically, the robot can track the target efficiency even if the direction and speed of the moving target change continuously. Suppose there is no obstacle in the path of the robot. Since the robot moves faster than the target, there is a linear path through which the robot and the target reach a common point simultaneously. Suppose the predicted tracking point is $C(x_c(t_c), y_c(t_c))$ (Figure 4). Since t_c is the same for the robot and the target, we can write:

$$\frac{\sqrt{(x_c - x_{G_i})^2 + (y_c - y_{G_i})^2}}{V_G} = \ldots \frac{\sqrt{(x_c - x_R)^2 + (y_c - y_R)^2}}{V_R} \tag{15}$$

In (15), x_c and y_c are unknown and can be calculated by combining (14) and (13). x_c is calculated as follows:

$$Ax_c^2 + Bx_c + C = 0 \tag{16}$$

where,

$A = \frac{1+k^2}{V_G^2} - \frac{1+k^2}{V_R^2}$

$B = \frac{2k(b - y_{G_i}) + 2x_{G_i}}{V_G^2} - \frac{2k(b - y_R) + 2x_R}{V_R^2}$

$C = x_{G_i}^2 - x_R^2 + (b - y_{G_i})^2 - (b - y_R)^2$

Since $B - 4 \times A \times C = 0$, so (x_c, y_c) is unique.

After predicting the achieving point and considering it as a secondary goal of the robot, the robot attempts to move toward it with respect to defined mechanisms until it detects the next change in the direction or speed of the target [12]. Because the target path is not a straight line, this method can be effective in achieving the target.

5. Description of the proposed algorithm

In general, we want to find the best possible function for each relative position of the robot in the environment. For this purpose, we assume that the robot is equipped with eight sensors that can calculate their distance to the nearest obstacle and have infinite visibility [23]. Sensors have been placed in eight directions at regular intervals.

At each step, the robot sends its distance from the 8 nearest obstacles in 8 directions, the angles between the line connecting it to the target, or forecasted point, and the vector (0,1) as input to the ART2 NN. After identifying the state of the robot, the best action should be determined with respect to reinforcement learning.

In the learning phase, the weights of the QLART2 NN change to be close to their optimum values. Once the robot has been trained enough with different scenarios, it will have appropriate behavior for avoiding contact with obstacles and achieving the goal in any environment.

In this algorithm, it is assumed that the robot knows the initial location of the target before its movement. When the target comes into sight of the robot, the robot predicts a point to reach the target, according to the speed and direction of the target, and tries to reach that point without contacting obstacles.

Robot learning is done by placing it in different static and dynamic environments; in each of them, the target and obstacle navigation and the number of static and dynamic obstacles are different. When the robot is far enough from obstacles can go straight to the forecasted point, and there is no need to save this state in the neural network. Otherwise, a nearly optimal action is determined for the robot so that the robot is as far away from obstacles without deviating from its direct path to the target as possible. Modifying the QLART2 NN's weights is done based on the effect of the selected action on the environment state.

Given that it is necessary to consider a reward for moving from a safe state to a non-safe state and vice versa, we have defined an intermediate state, called near non-safe state, which is stored in NN. To define the reward function, we assign a reward to each transition from one state to another. If S is the last transition state the robot has entered currently, SS will be Safe State, NSS will be Non-Safe State, NNSS will be Near-Non-Safe State, and FS will be Failure State; we can write the reward function as follows:

$$r = \begin{cases} 2, & S \subset NNSS \to SS \\ 0, & S \subset NNSS \to NNSS \\ -1, & S \subset NNSS \to NSS \\ 0, & S \subset NSS \to NSS \\ 1, & S \subset NSS \to NNSS \\ -2, & S \subset NSS \to FS \end{cases} \tag{17}$$

In this way, the robot approaches the optimal treatment in each step. Each learning scenario terminates when the target is reached, or the given steps have been passed. It should be noted that the robot will never collide with an obstacle because competition in the neural network occurs between acts that do not lead to collision with obstacles. However, when weights are updated to increase convergence speed in the learning phase, actions that lead to collisions receive a penalty.

6. Implementation and results

After training the robot with sufficient scenarios, the weights obtained between layers E and F of the QLART2 neural network, which indicate the optimal action in each state, can be used to direct the robot toward the goal. The algorithm proposed in this study has been effectively implemented and examined in different static and dynamic environments.

The parameters of the neural network were considered as follows: $a = 10,\ b = 10,\ c = 0.1,\ d = 0.9,\ \theta = 0,\ \rho = 1,\ \alpha = 0.2,\ \gamma = 0.8,\ \lambda = 0.05$. The initial weights from layer E to layer F and the weights of layer F to layer D are initialized to zero. Because the number of inputs is 9, the initial weights from layer D to layer F have been initialized by $\frac{1}{(1-0.9)\sqrt{9}}$.

The simulation of the algorithm is divided into two phases: learning and testing. In the learning phase, the robot is placed in various scenarios where obstacles and targets exhibit different behaviors. In the testing phase, the robot is placed in new environments that include both static and dynamic obstacles. It is able to avoid obstacles and reach the target effectively.

The robot's speed remains constant at 1 m/s in different scenarios, and its location at any given moment can be calculated using the following formula:

$$[p_{x_i}, p_{y_i}]^T = [p_{x_{(i-1)}}, p_{y_{(i-1)}}]^T + v[cos\theta, sin\theta]\Delta T \quad (18)$$

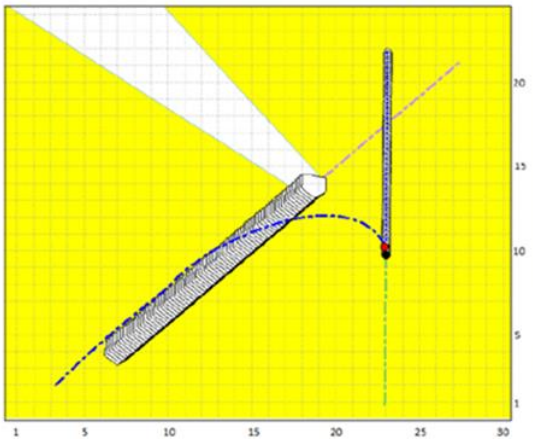
In which $[p_{x_i}, p_{y_i}]^T$ represents the location of the robot at time i, v is the speed of the robot, and θ is the direction of it, which can be one of the values 0, 45, 90, 135, 180, 225, and 270. ΔT is the elapsed time for the robot to move to a new location.

In this section, first, we will show the effect of predicting the achieving point to the target on the speed of reaching the target. Then, we will study one of the learning scenarios. Finally, we will compare a sample of found path in a scenario with the optimal path in that scenario. The color area in the figures is the visibility region of the robot.

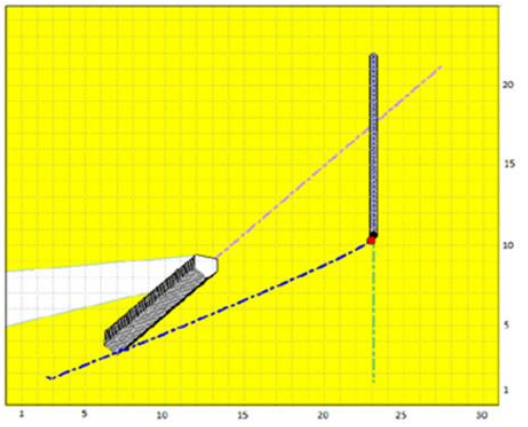
6.1 The effect of predicting the target interception point

Predicting the target interception point can result in increasing the speed of reaching the target by the robot. **Error! Reference source not found.** shows this case. In this figure, the target starts its movement from $[23, 22]^T$ and moves vertically with the speed of $v_{tar} = [0, -0.65]^T$. The obstacle at first is located in the point $[6, 4]^T$ and moves with the speed of $v_{obs} = [0.35, 0.35]^T$. The first position of the robot is $[2, 2]^T$. As can be seen in **Error! Reference source not found.**(a), if the robot moves directly to the target, it will reach the target in 86 steps. While pursuing the predicted point for reaching the target, 73 steps are necessary, as shown in Figure 4(b).

Other scenarios were conducted to check the increase in the speed of reaching the target when pursuing the predicted intercepting point by the robot. However, we do not consider them because they are similar to the example above.



(a)



(b)

Figure 4. The impact of the following predicted point on reaching the target by the robot. **(a)** The robot moves directly towards the target and reaches it in 86 steps; **(b)** The robot moves towards the predicted point to archive the target and can reach it in 73 steps.

6.2 Training phase

As previously mentioned, if the robot is sufficiently trained during the learning phase, it can navigate through obstacles in the testing environment and reach the target. In this section, we will describe a sample training scenario. Then, we will examine the impact of increasing the number of learning scenarios on the quality of the obtained answer.

It is important to note that in each scenario, the stored weights from previous scenarios have been used as the initial weights for the QLArt2 NN.

In a sample training scenario that we selected to describe, the target started moving from point $[9, 2]^T$ with a velocity $v_{tar} = [0.65, 0.65]^T$. The obstacle moved in a horizontal motion starting at point $[4, 17]^T$ and with a velocity $v_{obs} = [0.65, 0]^T$. In this scenario, the robot started its motion at point $[9, 31]^T$, as shown in **Error! Reference source not found.**. At the beginning of the scenario, the target is within the robot's visibility range. So, the robot predicts the intercepting point and moves directly towards it, assuming there are no obstacles in its path. At point $[12, 20]^T$, the robot gets close to the obstacle, and there is a possibility of collision in the next two moves. Hence, the QLART2 NN is used to determine the next action for the robot, ensuring that it avoids colliding with the obstacles and stays as close as possible to its original path toward the predicted point to reach the target.

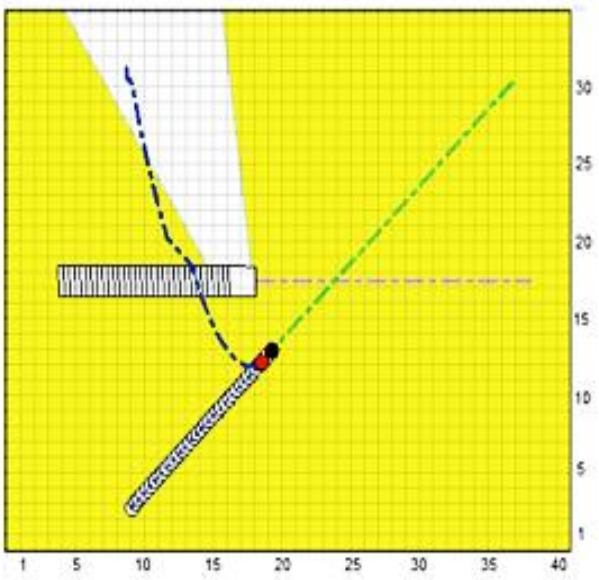
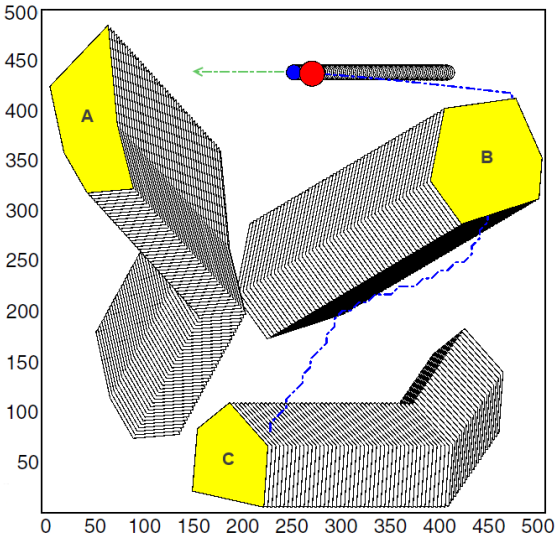


Figure 5. The effect of the robot's movement towards the predicted point to reach the goal.

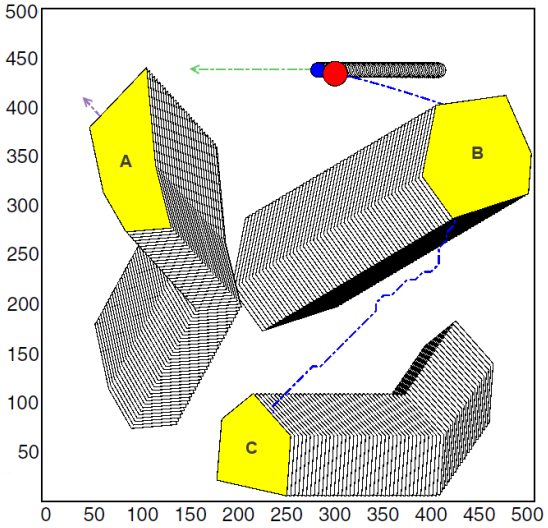
The effect of increasing the number of scenarios in the learning phase is observable in Figure 6. In Figure 6(a), the network has passed 10 scenarios for training, and it can be seen that the obtained path is not very suitable and, in this case, the robot takes 76 steps to reach the goal. In figures (b) and (c), the number of training scenarios is 40 and 60, respectively, and the robot takes 61 and 55 steps to reach the goal. In Figure (d), the network has 100 training scenarios, and the obtained path is very close to the optimal path. In this case, the robot takes 46 steps to reach the goal. Increasing the number of learning scenarios reduces the influence of the starting point on choosing the next move, allowing the robot to move toward the goal in a direction with a higher probability of seeing the goal. Table 1 summarizes the results of running the algorithm in 7 different environments using the above example networks, showing the overall average number of steps required for the robot to reach the goal in each environment. In these environments, the robot's speed is slightly higher compared to the example above, resulting in fewer steps needed to reach the goal. Examining the data in this table also confirms the accuracy of the stated information.

Table 1. Results of algorithm execution in 7 different environments

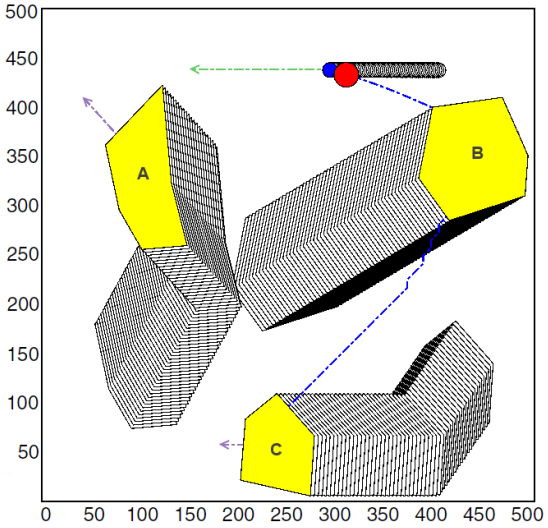
	20 Training scenarios	**40 Training scenarios**	**60 Training scenarios**	**100 Training scenarios**
$E1$	33	32	28	24
$E2$	55	51	45	39
$E3$	18	17	14	14
$E4$	68	55	54	54
$E5$	38	31	31	29
$E6$	39	32	30	30
$E7$	57	57	51	47
mean	44	39.28	36.14	33.85



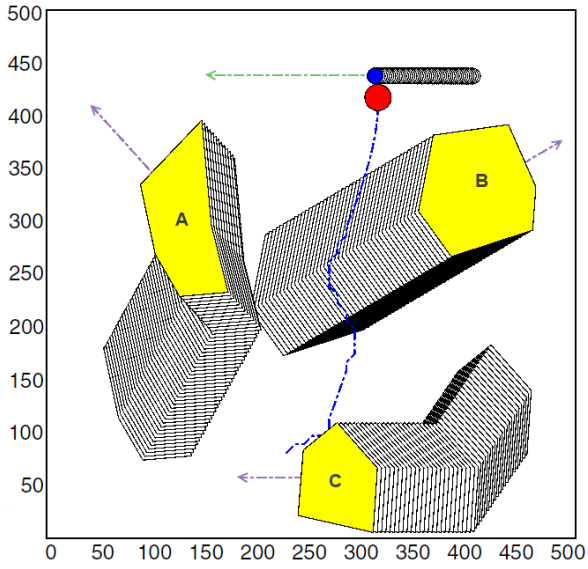
(a)



(b)



(c)



(d)

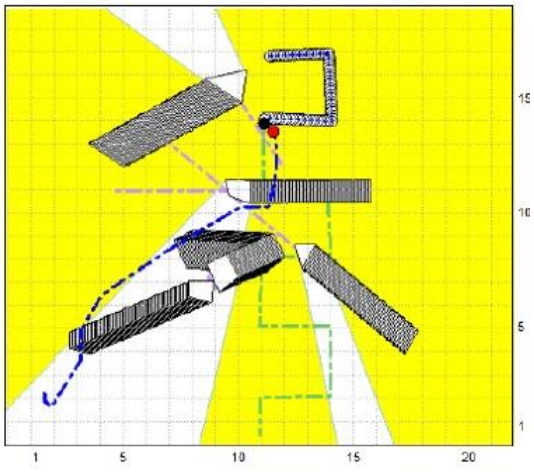
Figure 6. The effect of increasing the number of scenarios on improving robot behavior in target interception. **(a)** 20 training scenarios, **(b)** 40 training scenarios, **(c)** 60 training scenarios, and **(d)** 100 training scenarios.

6.3 Testing phase

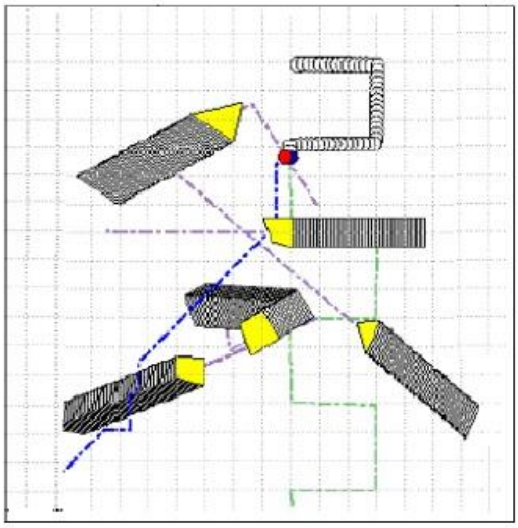
After the learning phase, we can begin assessing the robot's behavior in complex situations and scenarios. During testing scenarios, the robot makes decisions based on stored information in the QLArt2 NN. In addition, the robot can update its information during the testing phase. An example of a testing scenario is shown in Figure 7. In this scenario, there are 5 obstacles positioned in different locations and moving along various paths. Initially, the target is placed at the point $[11, 71]^T$ and follows a pseudo-sinusoidal path at two-thirds of the robot's speed. The robot is initially positioned at the point $[2, 2]^T$ at the beginning of the scenario. As depicted in Figure 7(a), the robot successfully reaches the target by following a path that avoids collisions. By comparing the robot's path in this scenario with the optimal path (Fig 7(b)), we observe that the robot's path to reach the target is very close to the optimal path.

Each of the methods mentioned in the literature review attempted to track moving targets without considering the presence of dynamic obstacles in the UAV territory. When obstacles are dynamic, i.e., they can change their position over time, the constraints of the path planning algorithm become more challenging. Therefore, to evaluate the quality of the obtained answers, we will compare the results of the proposed algorithm with a generalized version of the algorithm proposed in [18], which is widely used in motion planning. The algorithm, as mentioned earlier, is asynchronous and is therefore not suitable for uncertain

environments. The RRT algorithm performs path planning with prior knowledge of the shape, speed, and location of obstacles; therefore, it has good performance, and the obtained solutions are almost close to the optimal solution. Compared with state-of-the-art approaches such as the A-star, Dijkstra, and Sarsa algorithms, this algorithm results in improved performance. It is, therefore, a good choice for evaluating the performance of our algorithm. Since the algorithm operates on the basis of random samples, the solution and path obtained for it are not unique in all executions. Therefore, we run it 50 times for each environment and consider the best-obtained solution as the solution for comparison. Figure 6 shows the execution of the two algorithms in an environment with 5 obstacles.



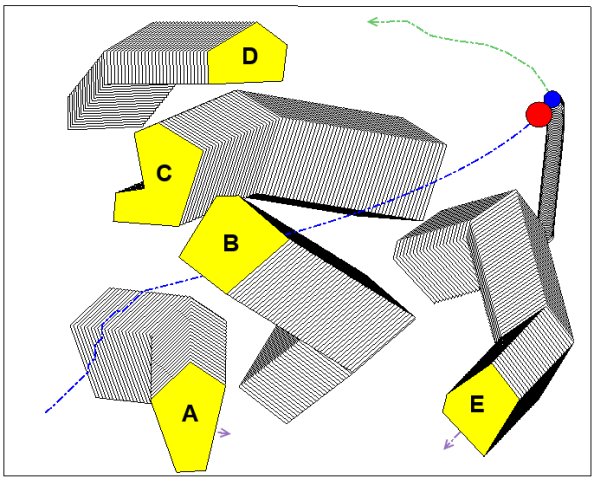
(a)



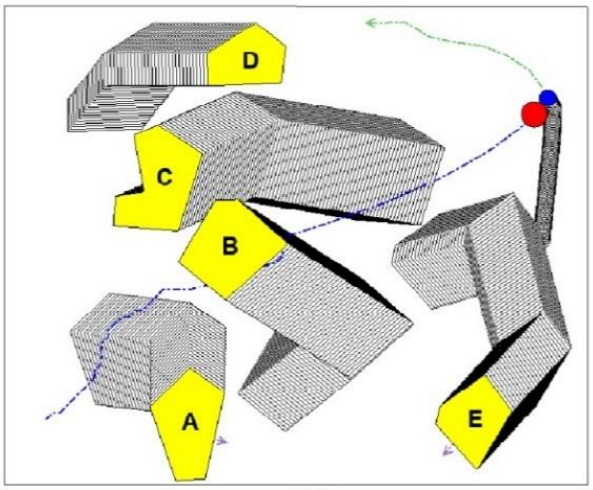
(b)

Figure 7. (a) Robot's path and initial location, goal, obstacles, and Robot's movement through obstacles to reaching the goal in a Test Scenario. **(b)** optimal path for this test scenario

In this example, the RRT algorithm produces a path with 84 steps, as shown in Figure 8(b), while our proposed algorithm (Figure 8(a)) yields a path with 82 steps, indicating a small difference. The following table shows the results obtained for two algorithms in 7 different environments. The analysis of the data obtained in this table shows that in simpler environments with fewer obstacles and complexity, our proposed algorithm produces a better answer than the RRT algorithm. However, as the number of obstacles and the complexity of the environment increase, the quality of our result and this difference between the two algorithms decreases. In such cases, the algorithm requires more training phases for the network.



(a)



(b)

Figure 8. Comparison of the proposed algorithm with the algorithm RRT. **(a)** Proposed Algorithm with prediction the point of achieving the goal after passing 100 training scenarios. **(b)** Algorithm RRT

Table 2. Execution of RRT and QLArt2 in 7 different environments

	RRT	QLArt2	Obstacle count
$E1$	33	27	3
$E2$	55	53	5
$E3$	18	15	2
$E4$	68	65	7
$E5$	38	31	3
$E6$	39	32	3
$E7$	57	58	8
mean	44	39.85	

7. Conclusion

In this paper, a new approach is presented for tracking a target in an uncertain environment. The speed and direction of the target and dynamic obstacles are constantly changing and unknown. The approach uses reinforcement learning based on the ART2 neural network to track the target without prior knowledge about the environment. There are no assumptions made about the movements of the target or obstacles. A new definition for the state space is introduced, and a forecasting point is used as a sub-goal to accelerate the robot's progress toward the target. One notable feature of this algorithm is its ability to make real-time decisions in target-tracking tasks after sufficient training. To evaluate the robot's ability to reach the target, various simulation experiments were conducted in different environments with different movements of the target and obstacles, as well as varying numbers of static and dynamic obstacles. These simulations demonstrate that the paths chosen by the robot using the presented algorithm are close to optimized paths, and the algorithm is efficient in dynamic, uncertain environments. In future work, the algorithm can be enhanced by adding a prediction of obstacle movements. Additionally, evaluations should be conducted to determine the optimal point for training the robot, as excessive training may lead to inefficient results.

8. References

[1] C. Zhou, B. Huang and P. Fränti, "A Review of Motion Planning Algorithms for Intelligent Robots." *Journal of Intelligent Manufacturing*, vol. 33, no. 2, pp. 387–424, Nov. 2021, doi: https://doi.org/10.1007/s10845-021-01867-z.

[2] Q. Zhu, J. Hu and L. Henschen, "A New Moving Target Interception Algorithm for Mobile Robots Based on Sub-Goal Forecasting and an Improved Scout Ant Algorithm." *Applied Soft Computing*, vol. 13, no. 1, pp. 539–549, Jan. 2013, doi: https://doi.org/10.1016/j.asoc.2012.08.013.

[3] P. Chen, J. Pei, W. Lu and M Li, "A Deep Reinforcement Learning Based Method for Real-Time Path Planning and Dynamic Obstacle Avoidance." *Neurocomputing*, 497: pp. 64-75, May 2022, doi: https://doi.org/10.1016/j.neucom.2022.05.006

[4] T-H.S. Li, S-J. Chang and W. Tong, "Fuzzy target tracking control of autonomous mobile robots by using infrared sensors," in *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 491-501, Aug. 2004, doi: http://doi.org/10.1109/TFUZZ.2004.832526.

[5] L. Yang, J. Qi, J. Xiao and X. Yong, "A literature review of UAV 3D path planning," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pp. 2376–2381, June 2014, doi: https://doi.org/10.1109/wcica.2014.7053093.

[6] L. Freda and G. Oriolo, "Vision-Based Interception of a Moving Target with a Nonholonomic Mobile Robot." *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 419–432, June 2007, doi: https://doi.org/10.1016/j.robot.2007.02.001.

[7] S. Lin, A. Liu, J. Wang and X. Kong, "A Review of Path-Planning Approaches for Multiple Mobile Robots." *Machines*, vol. 10, no. 9, p. 773, Sept. 2022, doi: https://doi.org/10.3390/machines10090773.

[8] E. Masehian and A. Naseri, "Mobile robot online motion planning using generalized voronoi graphs," *Journal of Industrial Engineering*, vol. 4, no. 5, pp. 1–15, Jan. 2010. [Online], Available: https://www.sid.ir/en/vewssid/j_pdf/1029920100501.pdf

[9] J. Fan, Y. Song and MR Fei "ART2 Neural Network Interacting with Environment." *Neurocomputing*, Elsevier BV, vol. 72, no. 1, pp. 170–176, Dec. 2008, doi: https://doi.org/10.1016/j.neucom.2008.02.026.

[10] WD. Smart and LP. Kaelbling, "Effective reinforcement learning for mobile robots," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, Washington, DC, USA, vol.4, pp. 3404-3410, 2002, doi: https://doi.org/10.1109/ROBOT.2002.1014237.

[11] H.R. Boem and H.S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Transaction on System, Man, and Cybernetics*, vol. 25, pp. 464-477, 1995, doi: https://doi.org/10.1109/21.364859

[12] J.J. Park, J.H. Kim and J.B. Song, "Path Planning for a robot manipulator based on probabilistic roadmap and reinforcement learning." *International Journal of Control, Automation, and Systems*, vol. 5, pp. 674–680, 2007.

[13] G.A. Carpenter, "ART 2-A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition." *Neural Networks*, vol. 4, no. 4, pp. 493–504, Jan. 1991, doi: https://doi.org/10.1016/0893-6080(91)90045-7.

[14] M. Yao, J. Li, Q. Gu, L. Tang and X. Qu, "Study on Q-learning algorithm based on ART2," *2010 8th World Congress on Intelligent Control and Automation*, Jinan, China, pp. 3161-3166, 2010, doi: https://doi.org/10.1109/WCICA.2010.5553787.

[15] J. Fan, Y. Song and MR Fei, "ART2 Neural Network Interacting with Environment." *Neurocomputing*, Elsevier BV, vol. 72, no. 1, pp: 170–176, 2008, doi: https://doi.org/10.1016/j.neucom.2008.02.026.

[16] E.N. Kazemi, N. Shabakhty, K. Abbasi and M.S. Sanayee, "Structural Reliability: An Assessment Using a New and Efficient Two-Phase Method Based on Artificial Neural Network and a Harmony Search Algorithm," *Civil Engineering Infrastructures Journal*, vol. 49, pp. 1–20, 2016, doi: https://doi.org10.7508/ceij.2016.01.001.

[17] J. Yu, Y. Su and Y. Liao. "The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning." *Frontiers in Neurorobotics*, vol. 14, Oct. 2020, doi: https://doi.org/10.3389/fnbot.2020.00063.

[18] S. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning." Technical Report, TR 98-11, 1998.

[19] H. H. González-Banos, D. Hsu, and J.-C. Latombe, "Motion planning: Recent developments. Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications," 2006.

[20] RC Luo, TM Chen, KL Su, "Target tracking using a hierarchical grey-fuzzy motion decision-making method," in *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 31, no. 3, pp. 179-186, May 2001, doi: https://doi.org/10.1109/3468.925657.

[21] L. Huang, "Velocity Planning for a Mobile Robot to Track a Moving Target — a Potential Field Approach." *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 55–63, Jan. 2009, doi: https://doi.org/10.1016/j.robot.2008.02.005.

[22] S.I.A. Meerza, M. Islam and 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)*, pp. 15, 2019, doi: https://doi.org/10.1109/ICASERT.2019.8934450.

[23] KB de Carvalho, IRL de Oliveira, DKD Villa, AG Caldeira, M Sarcinelli-Filho and AS Brandão, "Q-learning based Path Planning Method for UAVs using Priority Shifting," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 421–426, 2022, doi: https://doi.org/10.1109/ICUAS54217.2022.9836175.

[24] A. Konar, IG. Chakraborty, SJ. Singh, LC. Jain and AK. Nagar, "A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1141-1153, Sept. 2013, doi: http://dx.doi.org/10.1109/TSMCA.2012.2227719

[25] C. Yan and 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, doi: http://dx.doi.org/10.1109/ICRAS.2018.8443226.

[26] D. Li, W. Yin, W. E. Wong, M. Jian and M. Chau, "Quality-Oriented Hybrid Path Planning Based on A* and Q-Learning for Unmanned Aerial Vehicle," in *IEEE Access*, vol. 10, pp. 7664-7674, 2022, doi: 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), pp. 3397–3402, 2017, doi: http://dx.doi.org/10.23919/ChiCC.2017.8027884.

[28] J. Cui, R. Wei, Z. Liu and K Zhou, "UAV Motion Strategies in Uncertain Dynamic Environments: A Path Planning Method Based on Q-Learning Strategy." *Applied Sciences*, vol. 8, no. 11, Nov. 2018, p. 2169, doi: https://doi.org/10.3390/app8112169.

[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), pp. 669–673, 2021, doi: http://dx.doi.org/10.1109/CAC53003.2021.9728342.

[30] E. Masehian and Yalda Katebi. "Sensor-Based Motion Planning of Wheeled Mobile Robots in Unknown Dynamic Environments." *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 893–914, May 2013, Doi: https://doi.org/10.1007/s10846-013-9837-3.

[31] Z. Dehghani Ghobadi, F. Haghighi, A. Safari, "An Overview of Reinforcement Learning and Deep Reinforcement Learning for Condition-Based Maintenance." International Journal of Reliability, Risk and Safety: Theory and Application, vol. 4, no. 2, Dec. 2021, pp. 81–89, doi: https://doi.org/10.30699/ijrrs.4.2.9.

[32] S. Eidi, A. Safari and F. Haghighi, "Optimal Preventive Maintenance Policy for Non-Identical Components: Traditional Renewal Theory vs Modern Reinforcement Learning.", International Journal of Reliability, Risk and Safety: Theory and Application, vol. 6, no. 1, pp. 77–85, July 2023, doi: https://doi.org/10.22034/IJRRS.6.1.9.