Robotics and Computer-Integrated Manufacturing 27 (2011) 135–149

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Reinforcement based mobile robot navigation in dynamic environment

Mohammad Abdel Kareem Jaradat[a,*], Mohammad Al-Rousan[b], Lara Quadan[b]

[a] Mechanical Engineering Department, Faculty of Engineering, Jordan University of Science & Technology, Irbid 22110, Jordan
[b] Computer Engineering Departments, Faculty of Computer & Information Technology, Jordan University of Science & Technology, Irbid 22110, Jordan

ARTICLE INFO

Article history:
Received 9 April 2009
Received in revised form
22 June 2010
Accepted 24 June 2010

Keywords:
Robot navigation
Q-learning
Unsupervised learning
Potential filed force
Path planning
Mobile target

ABSTRACT

In this paper, a new approach is developed for solving the problem of mobile robot path planning in an unknown dynamic environment based on Q-learning. Q-learning algorithms have been used widely for solving real world problems, especially in robotics since it has been proved to give reliable and efficient solutions due to its simple and well developed theory. However, most of the researchers who tried to use Q-learning for solving the mobile robot navigation problem dealt with static environments; they avoided using it for dynamic environments because it is a more complex problem that has infinite number of states. This great number of states makes the training for the intelligent agent very difficult. In this paper, the Q-learning algorithm was applied for solving the mobile robot navigation in dynamic environment problem by limiting the number of states based on a new definition for the states space. This has the effect of reducing the size of the Q-table and hence, increasing the speed of the navigation algorithm. The conducted experimental simulation scenarios indicate the strength of the new proposed approach for mobile robot navigation in dynamic environment. The results show that the new approach has a high Hit rate and that the robot succeeded to reach its target in a collision free path in most cases which is the most desirable feature in any navigation algorithm.

1. Introduction

In the recent years, research and industrial interests are focused on developing smart machines such as robots that are able to work under certain conditions for a very long time and without any human intervention. This includes doing specific tasks in hazardous and hostile environments. Mobile robots are smart machines that can do such tedious tasks. These robots are used in the areas where the robot navigates and carries a certain task at the same time such as, service robots, surveillance and explorations [1].

Mobile robot navigation in an unknown environment has two main problems: localization and path planning [5,6]. Localization is the process of determining the position and orientation of the robot with respect to its surrounding. The robot needs to recognize the objects around it. It needs to recognize each object as a target or as an obstacle. Many techniques deal with this localization problem using laser range finders, sonar range finders, ultrasonic sensors, infrared sensors, vision sensors and GPS that have been developed on-board or off-board. When a larger view of the environment is necessary, a network of cameras has been used.

The other problem is the path planning in which the robot needs to find a collision free path from its starting point to its end point. In order to be able to find that path, the robot needs to run a suitable path planning algorithm, to compute the path between any two points [7].

Many researchers studied the problem of robot path planning with obstacle avoidance and many solutions were proposed to deal with the problem [8,9]. Since the robot motion in dynamic field has a certain amount of randomness due to the nature of the real world, these solutions did not give accurate results under all conditions. In the recent years there was a drift toward artificial intelligent approaches to improve the robot autonomous ability based on accumulated experiences. In general, artificial intelligent methods can be computationally less expensive and easier than classical methods.

This research focuses on mobile robot path planning moving in a dynamic environment, where a new approach is proposed to solve this problem based on Q-learning algorithm. Q-learning algorithm has many features that make it suitable for solving the mobile robot navigation problem in dynamic environment. First, the Q-learning agent is a reinforcement learning [29] agent that has no previous knowledge about its working environment. It learns about the environment through interacting with it. This type of learning agents is called unsupervised learning agent. Since it was assumed that the mobile robot has no previous knowledge about its working environment a Q-learning agent is a

* Corresponding author.
E-mail address: majaradat@just.edu.jo (M.A. Jaradat).

0736-5845/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2010.06.019

good alternative for solving the mobile robot navigation problem in dynamic environment. Secondly, Q-learning agent is an on-line learning agent. It learns the best action to take at each state by trial and error. It chooses actions randomly and calculates the value for taking an action at a specific state. Through evaluating every state and action pair it can build a policy for working in the environment. In the mobile robot navigation problem in order to find a collision free path, the robot needs to find the best action to take at each state. It needs to learn this knowledge on line, while navigating its environment. Because Q-learning is very simple, it is a very appealing alternative [10].

2. Literature review

In the last decade many classical solutions tried to address the robot path planning problem. The most commonly used solution is the potential field method [22] and its variants [12,14]. This method has been studied extensively by scholars. It was introduced in its most common form by Borenstein and Koren [11]. The basic idea behind this method is to fill the robot environment with a potential field in which the robot is attracted to the target position and is repulsive away from obstacles. At any position, the robot calculates the position that has the global minimum repulsive force and moves toward this position, and repeats this method as much as it is required until it reaches its target. Many variations appeared to this method like the ones introduced by Ge and Cui [12,13] to solve the problems of non-reachable targets when the target is very close to the obstacle [12] and to propose a solution for robot path planning in dynamic field when both the target and the obstacles are moving [13]. The potential field method is a good method for robot navigation, but it suffers from the local minima problem. When the net of the attractive and repulsive forces on the robot is zero, the robot stops moving. Unless a change occurs in the environment of the robot the robot will never be able to reach the target.

Since the robot navigation in dynamic environment is considered a very complex problem because the environment is difficult to be modeled the classical solutions worked under certain conditions but failed in the presence of sudden events or changes in the environment. Therefore, there was a drift toward artificial intelligent solutions to solve the robot path planning problems. Many artificial intelligent techniques were used by scholars to solve the problem, like fuzzy logic, neural networks and genetic algorithms. In 1999 a new method was introduced by Pratihar et al. [9] to solve the problem using a genetic-fuzzy approach. In this solution the robot field was assumed to be dynamic but the target was assumed to be static. In this approach the robot acts according to the location of the moving obstacle in the immediate past. This may lead to inadequate actions under some conditions. In 2001 Mucientes et al. [4] came up with a new fuzzy control system for avoiding the moving object by the robots. In this system the robot applied fuzzy temporal rules to estimate the movement trend of the obstacles. In 2005 Meng Joo and Chang Deng [15], proposed a new hybrid learning approach and a neuro-fuzzy controller was developed based on supervised learning in a simulation environment. All the previous methods deal with environments where the obstacles are moving, but the target is static.

Some other methods tried to deal with the robot navigation problem in dynamic environment, but they were adapted to the robot soccer environments. The limit cycle method is one of those methods [16], which is a reactive method that relies on the sense-plan-act cycle. This method assumes that the target trajectories are known in advance, so the environment is not completely unknown to the robot. This assumption makes it suitable for robot soccer. Another approach for developing a self-learning agent using the adaptive Q-learning algorithm was proposed in [17] and it was adapted to robot soccer only. These methods cannot be applied to arbitrary environment because they use the rules of the soccer game in their implementation for the navigation algorithm.

3. Mobile robot path planning using Reinforcement Learning in literature

In the proposed approach the robot path planning is solved using Q-learning. This method was first introduced by Watkins [18] for learning from delayed rewards and punishments. In literature there were many attempts to solve the mobile robot path planning problem using Reinforcement Learning algorithms. These methods learn the optimal policy for navigation to select the action that produces maximum cumulative reward.

Smart and Kaelbling [23,24] used the Q-learning for mobile robot navigation and obstacle avoidance. The used reward function assigns a value of 1 to the goal state, a value of -1 for the collision state and a value of 0 to all other states. The perceptions to the system from the environment are the distance between the robot and the target and the distances between the robot and all other obstacles in the environment. The conducted experiments trained the robot for several times then evaluated the learned information by exposing the robot to an environment that contains one static target and one static obstacle. The achieved hit rate was considered small and the navigation time to the target was considered long using their method. However, the conducted experiments showed the promise in using Reinforcement Learning for controlling the robot to accomplish different tasks.

Aranibar and Alsina [19], used Reinforcement Learning to solve the mobile robot path planning in three dimensional environment. They used Q-learning, and neural Q-learning and they compared it to the dynamic programming using the greedy search algorithm. The reward function that has been used is the same reward function used by Smart and Kaelbling [23] but with the addition of the definition of invalid states which are states outside the working environment and giving them reward value of 0. They found that the dynamic programming is better to be used for small-state spaces. Q-learning is better for medium-size spaces and neural Q-learning works best for large-size spaces, but they worked on static three dimensional environments only.

Some algorithms tried to combine the unsupervised Reinforcement Learning with other learning techniques like Fuzzy Logic and Neural Networks. Boem and Cho [25] proposed a hybrid approach using Fuzzy Logic and Reinforcement Learning. The proposed method is an Environment Exploration Method (EEM). The navigation algorithm consists of two basic modules: avoidance behavior and goal-seeking behavior. Each behavior is designed independently in the design state and they were combined into behavior selection in the running state. Both the avoidance behavior and the goal seeking behavior are fuzzy engines that map the input state of the environment supplied by the sensor readings to fuzzy output which is the desirable action. The fuzzy rules bases are built using reinforcement learning which requires simple evaluation data rather than thousands of training data.

Yung and Ye [26] presents another hybrid learning approach for mobile robot navigation using Fuzzy Logic and Reinforcement Learning. The navigator consists of three basic modules: obstacle avoidance, move-to-goal and a fuzzy behavior supervisor. The obstacle avoidance module learns the avoidance fuzzy rules using reinforcement learning. It perceives sensory inputs that are fuzzified then the rules are constructed using reinforcement learning, after that decisions are made and defuzzification generates the output action. The same steps are used by the move-to-goal module. The third module perceives the output of the other two modules and outputs the appropriate motion action. In this paper, the rule learning and the real navigation are separated into two stages. In the first stage the rule learning is done using a small sample environment. In the real navigation stage a self-tuning module is used to tune the universe of discourse for the sensor inputs. Hence after training the robot in a sample environment, the learned rules can be generalized and used for any other environment. This method produced higher performance when compared to the Fuzzy EEM and it does not suffer from the local minima problem. However, both methods are applicable only for static environments and cannot be applied for dynamic ones.

Another hybrid approach is presented by Er and Zhou [27]. In this method an enhanced dynamic self-generated fuzzy Q-learning (EDSGFQL) approach for automatically generated fuzzy inference systems (FISs) is used. The structure identification and the parameters estimations for the FISs are achieved using unsupervised reinforcement learning. The Q-learning is used for clustering the input space for generating the FISs. Also Reinforcement Learning is used for adjusting and deleting fuzzy rules dynamically using reinforcement signals. EDSGFQL was used to control a Khepera mobile robot. The simulation studies for wall following and obstacle avoidance tasks by a mobile robot proved the efficiency of the new proposed approach. But like previous methods, the proposed approach is applicable for static environments only.

A hybrid learning approach for obstacle avoidance that uses a fuzzy inference system constructed based on the generalized dynamic fuzzy neural networks learning algorithm was implemented by Er and Deng [28]. After applying a supervised learning for the neuro-fuzzy controller, a reinforcement method based on the fuzzy actor-critic learning algorithm is employed so that the system can re-adapt to a new environment without human intervention. The reinforcement learning is used to tune the parameters of the antecedent parts of the fuzzy rules. The proposed approach was tested using computer simulation in static environment and it was capable of solving the obstacle avoidance problem.

Yang et al. [20] used continuous Q-learning algorithm for solving the path planning problem for autonomous mobile robots. They used a multilayer feed-forward neural network to approximate the Q-learning value function. For the robot to able to navigate through an environment in an obstacle collision free path, it needs to go through the environment for several times until it can reach the target. This method was applied for static environment.

Another hybrid approach using neural network and Q-learning was presented by Macek et al. [29]. In their approach discrete Q-learning was used instead of continuous Q-learning. Kohonen neural network was used to convert the continuous state space into a discrete state space. The learning is based on clustering of input data in order to group similar inputs and separate dissimilar ones. Also discrete action space was used for this method. This method has a better convergence rate than the contentious Q-learning method and it performs faster. However, this method is only applicable for static environments only.

A hybrid approach for mobile robot navigation using Probabilistic Road Map Method (PRM) combined with the Q-Learning Method was implemented by Park et al. [30]. The PRM is a popular path planning method that connects the start location of the robot and the location of the goal through roadmap constructed by drawing random nodes in the free configuration space. This method has robust performance in static environments but its performance is degraded for dynamic environment. The Q-learning algorithm is used for dynamic environment when an obstacle blocks the preplanned path to determine the best action to be taken at each state. This method uses image processing and to detect the dynamicity of the system by comparing two images of the system. Image processing is also used to produce the C-space mapping.

Lin [31] proposed a hierarchical Q-learning algorithm (HQ-L), to solve the problem of mobile robot navigation problem. This algorithm consists of a group of Q-learning agents; each agent learns a sub-problem of a main problem. For each specific state an agent suggests an action. A winner then is selected and its action is executed. A decision making agent learns the *Q*-values for state–agent pairs and the action of the agent of highest *Q*-value is selected. Humpheryes [34] compared the standard Q-learning agent to an HQ-L agent in a simulation of a house robot application. It was found that the multi-agent HQ-L system requires less memory than the standard Q-learning.

Another use for Q-learning by mobile robots in multi-agent systems is in the robot soccer game [17,32]. But in these systems the Q-learning is not directly used for obstacle avoidance, it is used for strategy selection and role assignment.

From the previous review it can be seen that there is a tendency among researchers to solve the mobile robot navigation problem using reinforcement learning since it is more suitable for unknown systems that has a great amount of uncertainty. Q-learning is showing promising results for solving the problem of mobile robot navigation. It is the most suitable method because it was implemented to deal with unknown non-deterministic Markovian systems.

This research focuses on mobile robot path planning in unknown environment. The mobile robot is usually occupied with different kind of sensors and actuators such as: sonar, infrared, laser or cameras, to navigate freely in any unknown environment, starting from its initial location and looking to reach its moving goal through a collision free path. Real world unknown environment is not always static to the mobile robot as addressed by different researches before [1–4,33]. The robot surrounding environment consists of different obstacles, some of them are static and the others are dynamic, such as another robot moving in the same indoor environment or even humans. In this paper the Q-learning algorithm was applied to a dynamic environment by implementing a new definition for the environment states.

4. Assumptions

The initial robot location and the goal are predefined to the robot, where the robot will try to reach the goal with free collision path in spite of the presence of obstacles in the robot's surrounding environment. There are no predefined assumptions on the velocity of the robot when it reaches its target, which means that it is a hard landing robot.

In this paper, it is assumed that the robot is equipped with all necessary sensors to supply the robot with all necessary sensory data required by the robot to have a complete information and vision of its navigation environment at any instant of time during the navigation time. Hence, the environment is assumed to be fully observable for the learning agent (the robot). Based on this assumption, we have the following modified [13] assumptions on the navigation environment:

Assumption 1. The position p, and the velocity v, of the robot are known at each time instant.

Assumption 2. The position p_{tar} and the velocity v_{tar} of the target are known at each time instant.

Assumption 3. The position of the obstacles p_{obsi} and the velocity v_{obsi}, $i=1, 2, \ldots, n$, where n is the total number of obstacles are known at each instant of time.

Assumption 4. At each time instant, the velocity of the robot is greater than or equals the velocity of the target $|v| \geq |v_{tar}|$.

Assumption 5. At each time instant, the velocity of the robot is greater than or equals the velocity of each of the obstacles $|v| \geq |v_{obsi}|$, $i=1, 2, \ldots, n$, where n is the total number of obstacles.

Assumption 6. At each time instant, the shapes of the target and the obstacles are known for the robot.

5. Methodology

In order to apply the Q-learning algorithm four major parts should be addressed: the working environment, the reward function, the value function and the adapted policy. In the following subsections, each part is explained in details.

5.1. The environment model

The environment of the robot consists of its target and the obstacles. Both the target and the obstacles may be static or dynamic. The robot mission is to find a collision free path to navigate through this environment by applying the Q-learning algorithm. The first step for applying the Q-learning algorithm in such environment is to define the states and actions spaces.

For a robot navigating a static environment, the set of states can be completely defined by the robot location or coordinates. This is not the case for a dynamic environment, where the coordinates of the robot do not tell much about the state of its environment since the environment is changing very rapidly. Hence, another information about the state of the environment should be used other than the location of the robot or the obstacles.

When a human tries to walk safely in a dynamic environment to reach a specific goal, he/she will not care about his/her specific location or the specific distance to the target or to the obstacles. He/she will care for the relative or approximate distances and directions between him and the target, and between him and the closest obstacle; since he/she will try to deal with the obstacles one at a time.

In the new approach the objective was to make the learning agent mimic the human reasoning. Hence, at each time instant the robot was considered to be the center of the universe and the environment around it was divided into four orthogonal regions: R1, R2, R3 and R4 as shown in Fig. 1. The state of the environment at each instant of time is determined by the region containing the target and the region containing the closest obstacle to the robot. For example, if at time t_n, the target is in R2 and the closest obstacle is in R4, then the state at this time instant is s_n=(R2,R4), but this definition of the state is not enough for the obstacle avoidance process. The obstacle and the target may be at the same region, and this may produce confusion for the robot. The angle between the robot to goal line and the robot to obstacle line should be added to solve this confusion. The four regions R1, R2, R3 and R4 with the angle θ are shown in Fig. 1.

To compute θ we consider the robot location at any time instant is: $P=[P_x \;\; P_y]^T$, the target location is: $P_{tar}=[P_{tar-x} \;\; P_{tar-y}]^T$, and the closest obstacle location is: $P_{obs}=[P_{obs-x} \;\; P_{obs-y}]^T$. The distance between the robot and the target d_{r-t} is

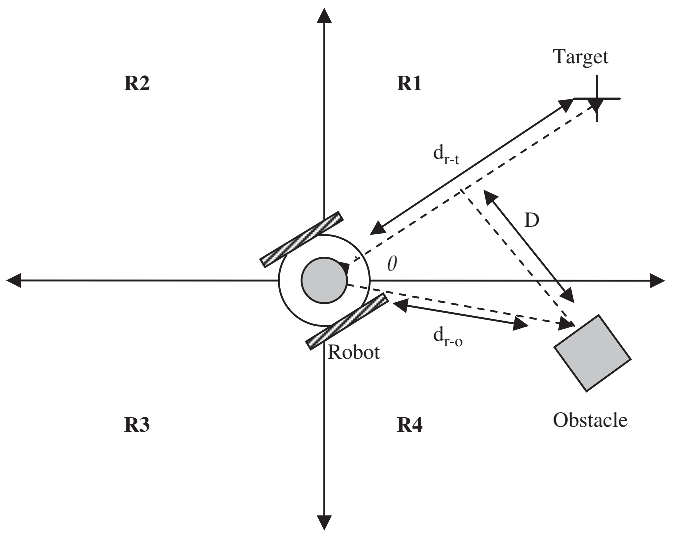


Fig. 1. The mobile robot environment containing one target and one obstacle.

given in the following equation:

$$d_{r-t}=\sqrt{(P_x-P_{tar-x})^2+(P_y-P_{tar-y})^2} \tag{1}$$

The distance between the robot and the obstacle d_{r-o} is given in the following equation:

$$d_{r-o}=\sqrt{(P_x-P_{obs-x})^2+(P_y-P_{obs-y})^2} \tag{2}$$

The equation of the robot to target line is

$$y-mx+b=0, \quad \text{where } m=\frac{P_y-P_{tar-y}}{Px-P_{tar-x}}, \quad b=-P_y+mP_x \tag{3}$$

Eq. (3) is based on finding the slope of the robot to target line (m). The distance between the obstacle and the robot to target line calculate based on the equation of the orthogonal distance between a point and a line is:

$$D=\frac{\|P_{obs-x}-mP_{obs-y}+b\|}{\sqrt{m^2+b^2}} \tag{4}$$

The angle θ can be calculated using Eqs. (2) and (4) as follows:

$$\theta=\sin^{-1}(D/d_{r-o}) \tag{5}$$

The range of θ is the interval $[0, 2\pi]$. Adding this angle will solve the problem when the target and the obstacle are in the same region, but may produce an infinite number of states. Moreover, the learning agent does not need to know the exact value for this angle. It only needs to have an idea about the range of this angle. To reduce the number of states, the interval is divided into eight angular regions:

$$G=\begin{cases} G1, \; \theta\in[0, \pi/4[\\ G2, \; \theta\in[\pi/4, \pi/2[\\ G3, \; \theta\in[\pi/2, 3\pi/4[\\ G4, \; \theta\in[3\pi/4, \pi[\\ G5, \; \theta\in[\pi, 5\pi/4[\\ G6, \; \theta\in[5\pi/4, 3\pi/2[\\ G7, \; \theta\in[3\pi/2, 7\pi/4[\\ G8, \; \theta\in\left[7\pi/4, 2\pi\right[\end{cases} \tag{6}$$

where G is the current angular region, and G1 through G8 are the possible values for this region. The eight regions are shown in Fig. 2.

Using these definitions the state at any time instance will be completely defined as a combination of the region of the target, the region of the closest obstacle and the range of the angle

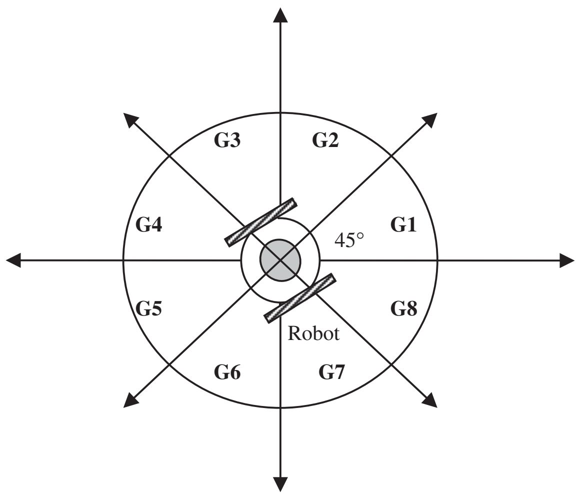


Fig. 2. The eight regions for the angle θ.

between the robot to target line and the robot to obstacle line:

$$s_t = (R_g, R_o, G_n), \quad n \in [1,8] \tag{7}$$

where s_t is the state of the environment at time instant t, R_g is the region of the goal or the target, R_o is the region of the closest obstacle and G_n is the angular region of the angle between the obstacle to target line and the robot to obstacle line.

After defining the states space, the allowed actions in each state should be specified. The robot is assumed to be of non-holonomic constraints, so three actions were defined for the robot movement: Move Forward, Turn Right and Turn Left. The three actions depend on the robot orientation. The orientation angle for robot is determined by the target location. The robot changes its orientation at each time instant so that it can move forward to the target in a straight line taking the shortest path. The turn left and turn right actions are used for obstacle avoidance. The turn degree to the left or right directions can vary depending on the situation and the required degree of avoidance.

5.2. The reward function

The reward function is an immediate evaluator for the action taken at a given state. It is an indicator of how good or how bad the performed action at a specific situation is. It is used to calculate the value function that will determine the behavior of the robot at different states. To define the reward function, the set of states were classified as follows:

- The set of states where the robot has a low or no possibility for collision with any of the obstacles in the environment are considered *Safe States* (*SS*).
- The set of states where the robot has a high possibility for collision with any of the obstacles in the environment are considered *Non-Safe States* (*NS*).
- Two terminate states are defined:
  a. The state when the robot reaches its goal is considered a *Winning State* (*WS*).
  b. The state when the robot collides with an obstacle is considered a loss or a *Failure State* (*FS*).

To make the robot navigate safely in its environment the reward function is defined as follows:

- The reward for moving from a *Non-Safe State* to a *Safe State* is $r=1$.
- The reward for moving from a *Safe State* to a *Non-Safe State* is $r=-1$.

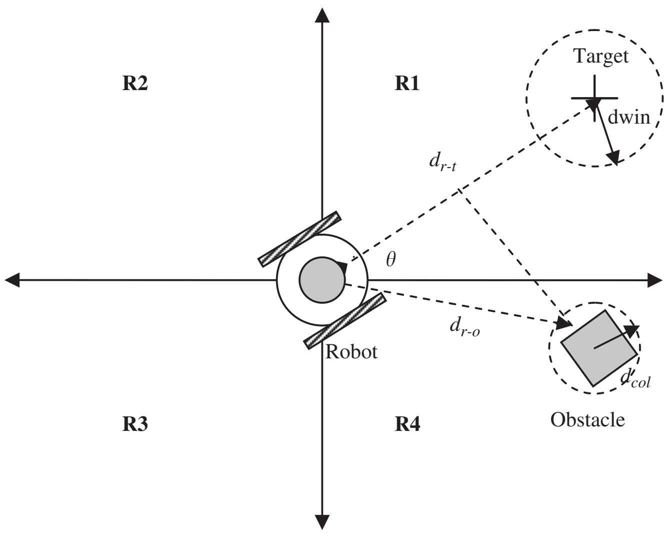


Fig. 3. The main distances for determining the robot state.

- The reward for moving from a *Non-Safe State* to a *Non-Safe State* but getting closer to the obstacle is $r=-1$.
- The reward for moving from a *Non-Safe State* to a *Non-Safe State* and getting away from the obstacle is $r=0$.
- The reward for moving to a *Winning State* is $r=2$.
- The reward for moving to *Fail State* is $r=-2$.

This reward function gives the robot the tendency to stay in a safe region until it reaches its winning state and keeps it away from obstacles.

The distances between the robot and obstacle d_{r-o} and the one between the robot and the target d_{r-t} determine the transition state of the robot. The radius of the *Non-Safe* region around the obstacle d_{min} can be determined depending on the nature of the environment and the obstacles. Also, the radius of the collision region d_{col} around the obstacle should be calculated, which determines the influence area of the obstacle. These distances are shown in Fig. 3.

The values of d_{min}, d_{col}, d_{r-t} and d_{r-o} together with the radius of the wining region around the target d_{win} which reflects the influence area of the target determine the current transition state of the robot as follows:

$$S = \begin{cases} WS, \ d_{r-t} \le d_{win} \\ SS, \ d_{r-o} > d_{min} \\ NS, \ d_{col} < d_{r-o} \le d_{min} \\ FS, \ d_{r-o} \le d_{\mathrm{col}} \end{cases} \tag{8}$$

where S is the new transition state the robot has entered currently. Based on this definition of the transition states, the reward function can be re-written as follows::

$$r = \begin{cases} 2, \ S \subset SS \to WS \\ 1, \ S \subset NS \to SS \\ -1, \ S \subset SS \to NS \\ -1, \ S \subset NS \to NS, d_{r-o}(n+1) < d_{r-o}(n) \\ 0, \ S \subset NS \to NS, d_{r-o}(n+1) > d_{r-o}(n) \\ -2, \ S \subset NS \to FS \end{cases} \tag{9}$$

5.3. The value function

In the Q-learning algorithm, the value function is stored in a table that contains the Q-values. The rows of the table are the different states the robot passes through while learning. The columns of the table are the actions performed by the robot. The table is initially set to all zeros. Then the table entries are filled while training the robot by updating the *Q*-values of the state–action pairs. The value is the immediate reward of the action taken at a state and the maximum previous reward for all actions taken at the new resulting state [10,21]:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma Max(Q, s_{t+1}) \tag{10}$$

where s_t is the state at time instant *t*, a_t is the action taken by the robot at this time instant. $r(s_t, a_t)$ is the immediate reward calculated as described in the previous section. $Max(Q, s_{t+1})$ is the maximum *Q*-value calculated for taking all possible actions on the new state at previous time. γ is the discount factor. The value function is calculated while training the robot by exposing him to different scenarios of a dynamic environment. The information stored in the table is used by the robot to reach its target when it is set to work in a real environment [23,24].

5.4. The policy

After training the robot in different environments with different conditions, the robot will have a policy that can be used for future navigations. The robot does not need to follow this policy if it is navigating a safe zone. The simple policy to follow while being in a *Safe State* is to change its orientation toward its target then to move forward. The policy stored in the Q-table is used when entering a Non-Safe region. In this case the robot will take the action with the highest *Q*-value for the current state.

5.4.1. The Q-learning training algorithm

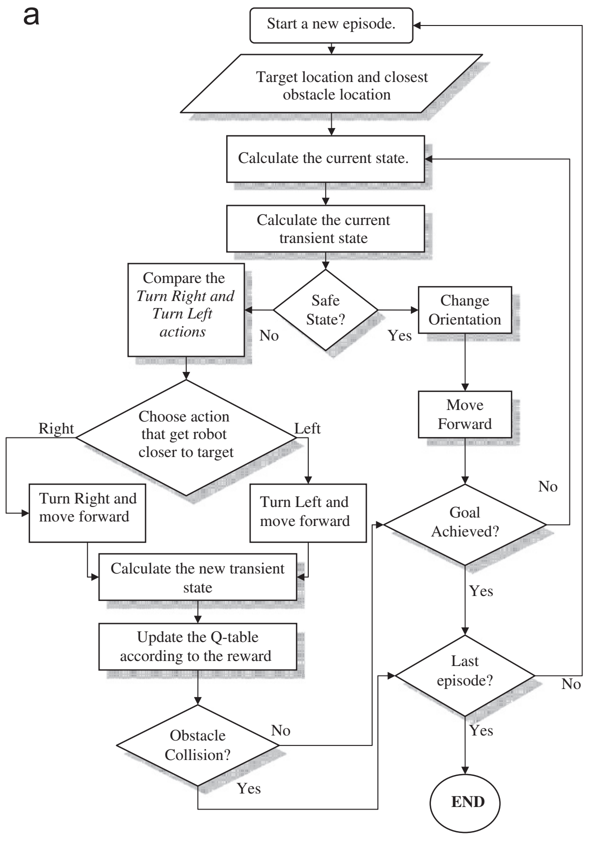
Training the mobile robot is done by exposing it to different scenarios of dynamic environments each of them is called an episode. In each episode different target motion is assumed. Also, a different number of obstacles are used in each episode. Some of the obstacles are assumed to be static and some are assumed to be moving. The locations of the static obstacles are chosen randomly in each episode. The movements of the dynamic obstacles are assumed to be random in all episodes. The greater the number of episodes used to train the robot, the better will be the performance of the robot for navigating the dynamic environment.

The Q-learning algorithm for training the mobile robot is shown in the flow chart of Fig. 4. As it can be seen from the flow chart, the training algorithm is repeated many times until all episodes have been performed. The episode starts by calculating the current state of the environment. The target and the obstacle locations are supplied to the robot through its sensors. This data combined with the robot current location is used to find the current state of the environment.

After that, the current state is checked, if it is a *Safe State* the robot changes its orientation toward the target location, and moves one step forward trying to reach the target in the shortest path. If this step leads to reaching the goal, the episode is finalized and a new episode is started. If the target has not been reached yet, the robot calculates the new current state and repeats the process.

If the current state is a *Non-Safe State* the robot compares the expected results from taking either action: *Turn Right* and *Turn Left* and chooses the one that will lead to a closer location to the target. Then, the robot checks the resulting transient state from taking the action and updates the Q-table accordingly. If the action leads to a *Winning State* which means the robot reached the target as a result of taking the action, the reward for the action will be 2 as mentioned previously. If the action leads to a *Safe State*, the reward will be 1. If the action leads to a *Non-Safe State* the reward is 0 and if it leads to a collision with the obstacle the reward is −1. The *Q*-value is then calculated as shown in Eq. (10) and the Q-table is updated accordingly.

a


Start a new episode.
Target location and closest obstacle location
Calculate the current state.
Calculate the current transient state
Safe State?
No
Yes
Compare the Turn Right and Turn Left actions
Change Orientation
Move Forward
Choose action that get robot closer to target
Right
Left
Turn Right and move forward
Turn Left and move forward
Goal Achieved?
No
Yes
Calculate the new transient state
Update the Q-table according to the reward
Last episode?
No
Yes
Obstacle Collision?
No
Yes
END

b

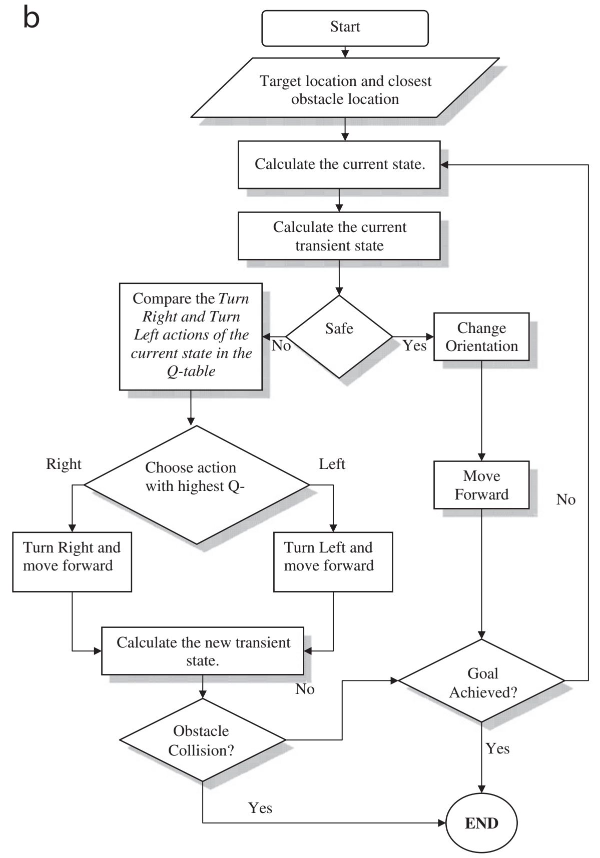

Start
Target location and closest obstacle location
Calculate the current state.
Calculate the current transient state
Compare the Turn Right and Turn Left actions of the current state in the Q-table
Safe
No
Yes
Change Orientation
Choose action with highest Q-
Right
Left
Move Forward
No
Turn Right and move forward
Turn Left and move forward
Calculate the new transient state.
No
Goal Achieved?
Obstacle Collision?
Yes
Yes
END

Fig. 4. (a) Robot Q-learning algorithm and (b) robot using Q-learning policy.

After that the robot continues its task calculating the new current state and transient state. If the last action leads to an end of an episode as in the case of a *Winning State* or a *Terminate State* the episode is terminated and a new episode is started. If it was the last episode, the whole learning process is terminated.

Each episode has a specific time slot. The robot needs to reach its target during this time slot. If the robot ran out of time and did not reach the target, the episode is terminated and a new episode is started.

The result from applying the previous algorithm is a policy stored in the Q-table. This policy is used by the robot for future navigation in any dynamic environment.

5.4.2. *Robot navigation algorithm using the learned policy*

After training the robot on different episodes of dynamic environment, the resulting policy can be used by the robot for any future navigation. The flow chart in Fig. 4 shows the robot behavior while following the learned policy.

As it can be seen from the chart the robot starts its navigation through the environment by finding its current state and its current transient state. If the current transient state is a *Safe State* the robot changes its orientation towards the target and moves one step forward. It continues doing that until the current transient state is *Non-Safe*. In this case the robot checks the row of the current state in the Q-table and builds its decision of turning right or left based on the stored *Q*-values. If the *Turn Right* action has a higher *Q*-value, the robot turns right in a specific angle then moves forward one step. On the other hand, if the *Turn Left* action has a higher *Q*-value, the robot turns left and moves one step forward. If both actions have the same *Q*-value, the robot will turn in either direction.

After that, the robot finds its new transient state. If it is a *Terminate State* then a collision is detected and the navigation process is terminated. If it is a *Winning State* then the target has been reached and the navigation process ends. If it is a *Safe State* or a *Non-Safe* State the navigation continues.

6. Simulation and results

An extensive simulation studies were carried to train the robot and to prove the effectiveness of the new method. This simulation was implemented using MATLAB software. Different scenarios for different situations were implemented and the results of these scenarios were used to assess the performance of the proposed Q-learning solution.

6.1. *Simulation scenarios*

The simulation is divided into two phases: The training phase and the testing phase. In the training phase different scenarios for different environments were implemented with different target and obstacle behaviors. The learning occurs while the robot navigates through the different states of each scenario. The learned information from the different scenarios is accumulated and stored in the Q-table as mentioned previously. After the learning phase the testing phase starts by exposing the robot to more complicated environments. The testing scenarios have bigger number of obstacles, some are static and some are dynamic. Some of the testing scenarios were designed for testing the algorithm for special common problems like the local minima problem. Some other testing scenarios were random ones. In these random scenarios, the locations of the static obstacles were chosen randomly and the motion of each dynamic obstacle is a random one as well. These test scenarios were used to assess the value of the information learned during the first phase of the simulation. Random testing scenarios were used more extensively for evaluating the efficiency of the use of Q-learning algorithm for mobile robot navigation in dynamic environment.

Before starting the presentation of the conducted simulation scenarios and their results some parameters should be determined to be used in the different scenarios. The first one is the angle of turn in the *Turn Left* and *Turn Right* actions. Throughout all our scenarios, this angle was set to 45^{o}.

The other parameters are the velocities for the robot, the target and the obstacles. The velocities for the target and the obstacles are varied according to each scenario and they are stated at their places in the scenarios. The robot velocity is constant and is set to 1 m/s throughout the test scenarios. The robot location at each time instant is determined by the following equation [18]:

$$\begin{bmatrix} p_{xn} \\ p_{yn} \end{bmatrix} = \begin{bmatrix} p_{x(n-1)} \\ p_{y(n-1)} \end{bmatrix} + v \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \Delta T \tag{11}$$

where p_{xn} is the location of the robot in the x direction at time instant (n), p_{yn} is the location of the robot in the y direction at time instant (n). $p_{x(n-1)}$ and $p_{y(n-1)}$ are the x and y components for the robot location at time instant $(n-1)$. The velocity of the robot v is constant, θ is the turn or orientation angle and $\mathbf{\Delta}T$ is the duration for each iteration.

6.1.1. *The training phase*

Several training scenarios were conducted. In each scenario different target and obstacle behavior were assumed. Each training scenario consists of the robot, the target and one obstacle. Following is a description for each scenario.

6.1.1.1. Training scenario (1). In the first scenario the target moved in a vertical motion starting at point $\begin{bmatrix} 70 & 50 \end{bmatrix}^{\mathrm{T}}$ and with a velocity $v_{tar} = \begin{bmatrix} 0 & -0.5 \end{bmatrix}^{\mathrm{T}}$. The obstacle started its motion at point $\begin{bmatrix} 20 & 5 \end{bmatrix}^{\mathrm{T}}$ and with velocity $v_{obs} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^{\mathrm{T}}$. In this scenario the robot started its motion at point $\begin{bmatrix} 10 & 10 \end{bmatrix}^{\mathrm{T}}$, and through applying the Q-learning algorithm the robot succeeded to reach the goal at point $\begin{bmatrix} 70 & 12 \end{bmatrix}^{\mathrm{T}}$ at time instant 76, as shown in Fig. 5a, the robot is shown as star sign, the goal as a cross sign, while the obstacle is denoted as a square sign.

At the beginning of the navigation process the robot was moving in a smooth path toward the goal oriented toward the target location and using its *Forward* action, until time instant 24. At that time the robot entered a *Non-Safe* region, where the obstacle was in region R4 and the goal in region R1. As the algorithm suggests the robot has the option of moving 45° left or right. It can choose either action randomly since it is the first time it encounters such a state it can choose the action that will get it closer to the target. The robot chose to move left, because it is the action that leads to a point closer to the target. This action also helped avoiding the obstacle, so the enforcement the robot gets is high and the *Q*-value of this action at this state is increased. Next time the robot will pass through this state it will choose to move left anyway.

6.1.1.2. Training scenario (2). In the second training scenario the target moved in a vertical motion starting its motion at point $\begin{bmatrix} 70 & 50 \end{bmatrix}^{\mathrm{T}}$ as in the first scenario. The velocity of the target $v_{tar} = \begin{bmatrix} 0 & 0.5 \end{bmatrix}^{\mathrm{T}}$. The obstacle started at point $\begin{bmatrix} 30 & 25 \end{bmatrix}^{\mathrm{T}}$ and its velocity $v_{obs} = \begin{bmatrix} -0.5 & -0.5 \end{bmatrix}^{\mathrm{T}}$. In this scenario the robot started its motion at point $\begin{bmatrix} 10 & 10 \end{bmatrix}^{\mathrm{T}}$, and through applying the

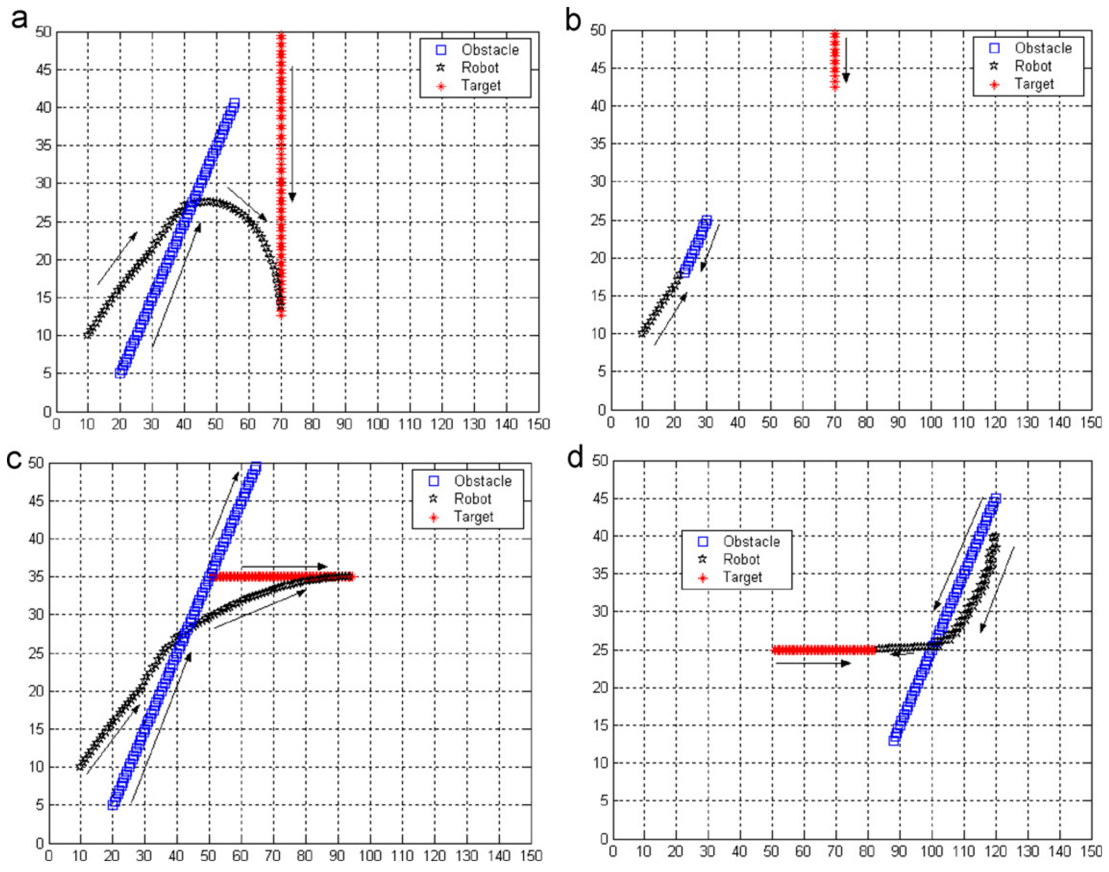


Fig. 5. Path for robot in: (a) the first training scenario, (b) the second training scenario, (c) the third training scenario and (d) the fourth training scenario.

Q-learning algorithm the robot failed to reach its target as shown in Fig. 5b.

At the beginning of the navigation process the robot was moving in a smooth path toward the target, until time 13. At that time the robot entered a *Non-Safe* region, where the obstacle was in region R1 and the goal was also in region R1. As the algorithm suggests the robot has the option of moving 45° left or right in order to avoid the obstacle and trying to get back to the safe region. Since it is the first time it encounters such a state it can choose either action randomly or as in our case it can choose the action that will get it closer to the goal. The robot chose to move left, because it is the action that leads to a point closer to the goal. This action produced a direct collision with the obstacle at time 15. As a result, the robot received a negative reinforcement for taking this action in this state and the *Q*-value for this state–action pair is decremented. Next time the robot will pass through this state it will choose to move right. Since the robot encountered a collision, the navigation process has been stopped and the training scenario was terminated.

6.1.1.3. Training Scenario (3). In the third scenario the target moves in a horizontal line starting at point $\begin{bmatrix}50 & 35\end{bmatrix}^{\mathrm{T}}$. The velocity of the target is $v_{tar}=[0.5 \quad 0]^T$. The obstacle velocity $v_{obs}=[0.5 \quad 0.5]^T$ and it started its motion at point $\begin{bmatrix}20 & 5\end{bmatrix}^{\mathrm{T}}$. Again, the robot started its motion at point $\begin{bmatrix}10 & 10\end{bmatrix}^{\mathrm{T}}$, and through applying the Q-learning algorithm the robot succeeded to reach the goal at point $\begin{bmatrix}94 & 35\end{bmatrix}^{\mathrm{T}}$ at time instant 88, as shown in Fig. 5c.

At the start of this scenario the robot was moving in a smooth path toward the target until time instant 23. At that instant the robot entered a non-safe region, where the obstacle was in region R4 and the goal in region R1. As the algorithm suggests the robot has the option of moving 45° left or right. The robot executed the action that will get it closer to the target, so it turned left. The robot received a positive reward for this action. Since it is not the first time it encounters such a state. It is the same state it encountered in scenario one the *Q*-value for this action–state pair is incremented again. The robot succeeded to avoid colliding with the obstacle. The robot will use this *Q*-value if it encounters this state in the testing phase.

6.1.1.4. Training Scenario (4). In the fourth scenario the target moved in a horizontal line starting at point $\begin{bmatrix}50 & 25\end{bmatrix}^{\mathrm{T}}$ with velocity $v_{tar}=[0.5 \quad 0]^T$. The obstacle velocity in this scenario is $v_{obs}=[-0.5 \quad -0.5]^T$ and it started its motion from point $\begin{bmatrix}120 & 45\end{bmatrix}^{\mathrm{T}}$. In this scenario the robot started its motion at point $\begin{bmatrix}120 & 40\end{bmatrix}^{\mathrm{T}}$ which is a close location to the obstacle. Using the Q-learning algorithm the robot succeeded to reach the goal at point $[83.5 \quad 25]^T$ at time instant 64, as shown in Fig. 5d.

In this scenario the robot started from a non-safe region. As it can be seen from the figure the robot bounced back and forth trying to reach the goal and in the same time trying to avoid the obstacle. In one step the robot leaves the non-safe region to the safe region. Then in the following step it enters again the non-safe region while trying to move toward its target in the shortest path. The bouncing situation for the robot persisted until point $\begin{bmatrix}100 & 25\end{bmatrix}^{\mathrm{T}}$ where it could avoid the obstacle completely then it started to move in a smooth line toward its goal.

The training for the robot can continue as much as it is needed with different goal and obstacle behaviors. The more we train the robot, the better results we expect to get in the future.

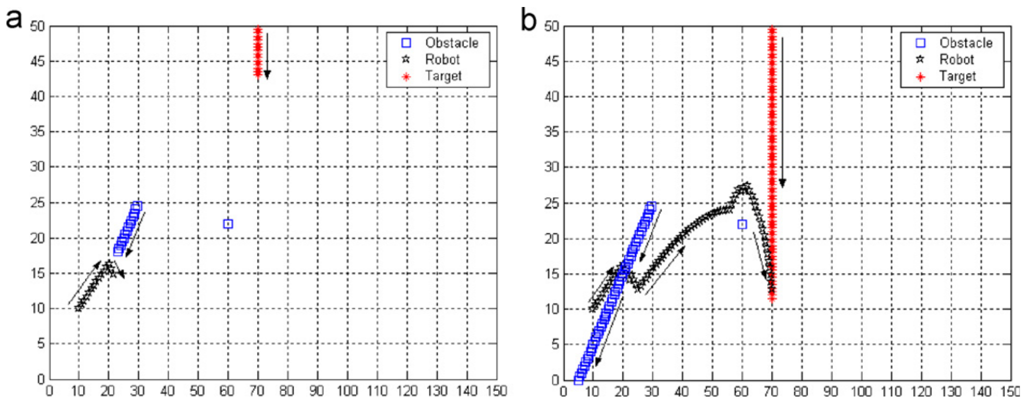


Fig. 6. Path for the robot in the first testing scenario: (a) until time instant $n=15$ and (b) time instant $n=77$.

6.2. The testing phase

After the training phase the testing phase started with more complex situations and scenarios. Different types of target motions were simulated and more obstacles were added in each scenario, some are static and some are dynamic. The location of the static obstacles has been chosen so as part of them can disturb the movement of the robot. For the dynamic obstacles, the starting point and the path followed by each obstacle have been chosen in away to disturb the movement of the robot. In the last testing scenario, the location of the static obstacles has been chosen randomly and also the paths followed by the dynamic obstacles were determined randomly using the random walk model. Following is a description of some of the test scenarios that we conducted to evaluate our new approach for mobile robot navigation.

It should be noted however, that the test scenarios can also be considered as training scenarios by themselves because the robot keeps updating its Q-table every time it navigates a new environment. Hence, the learned information keeps developing all the time.

6.2.1. Test Scenario (1)

The first testing scenario is the same as the second training scenario. The robot failed to reach its target in the second training scenario. The aim is to check whether the robot will succeed to reach its target after the training or not. As to make things a little bit difficult; we added a new static obstacle at the point $\begin{bmatrix}60 & 22\end{bmatrix}^T$. The result of this test scenario can be seen in Fig. 6a and b.

As it can be seen from Fig. 6b the robot succeeded to reach its target this time. This proves that the robot learned from his previous experience with this environment. The robot did not repeat the action that resulted in collision in training scenario. When the robot reached the *Non-Safe* region around the first obstacle it turned right this time instead of turning left as in the first time. Because it did learn that turning left in such a state leads to a collision, hence the *Turn-Left* action has been given a lower *Q*-value. When the robot faced the same state it picked the *Turn-Right* action because it is the one with the higher *Q*-value this time.

Fig. 6a shows that the robot managed to avoid the dynamic obstacle at time instant $n=15$. It was able to avoid the static obstacle as well. It can be clearly seen from Fig. 6b that the presence of the static obstacle forced the robot to pass through a longer path to avoid collision with this obstacle, but before the end of the simulation time the robot managed to reach its target at time instant $n=77$.

The path to the target should be shorter if the robot turned right instead of turning left when it reached the static obstacle. But at that time the target was at point $\begin{bmatrix}75 & 30\end{bmatrix}^T$ so the robot chose to turn left trying to reach the target. The robot has no information about the movement of the target and makes no assumptions. At each time the robot only look at the current location of the target. It does not look at the previous locations to estimate what the next new location will be.

6.2.2. Test Scenario (2)

The second test scenario is a random one. It contains six obstacles, two static and four dynamic obstacles. Obstacles 1 and 2 are the static ones located at the points $\begin{bmatrix}13 & 17\end{bmatrix}^T$ and $\begin{bmatrix}14 & 47\end{bmatrix}^T$, respectively. Obstacles 3, 4, 5 and 6 are dynamic obstacles. Their random paths are determined by the random walk model. The starting points for the dynamic obstacles and the location of the static obstacles were chosen randomly too. Obstacle 3 started its motion at point $\begin{bmatrix}72.1 & 33.3\end{bmatrix}^T$, Obstacle 4 started at point $\begin{bmatrix}65.9 & 34.7\end{bmatrix}^T$, Obstacle 5 started walk at point $\begin{bmatrix}80.2 & 28.5\end{bmatrix}^T$ and Obstacle 6 started its motion at point $\begin{bmatrix}96.2 & 43.4\end{bmatrix}^T$. The target motion in this scenario is a sinusoidal motion. It started moving at point $\begin{bmatrix}75 & 25\end{bmatrix}^T$ with a velocity $v_{tar}=[0.6,\ 3\times\sin(x)]^T$. The robot started its navigation at point $[10,\ 10]^T$ at a constant speed $v=1$ m/s.

The complete track of the robot in this environment is shown in Fig. 7a. The figure shows that the robot succeeded to reach its sinusoidal target and the duration for the navigation process was 83 time instances. The robot managed to avoid obstacle 1 as early as it started its motion as can be seen from Fig. 7b and then managed to avoid Obstacle 5 at the end of its track Fig. 7c. The other obstacles did not distract the motion of the target.

Fig. 7d shows another testing scenario with the same starting points for the target and the robot and the same velocity and motion for the target. The difference this time is that none of the obstacles passed in the way of the robot. In this scenario the robot needed 82 time instances to reach its target when it moved directly to the target without any distraction. If the time for this scenario was compared with the previous one, it can be clearly seen that the latency for the robot is only one time instant. In spite of the fact that the robot has to avoid two obstacles, the latency was very small. The previous discussion shows the efficiency of the new solution.

The path taken by the robot is not the shortest path even when it moved directly to the target without any distraction. But as we assumed, the robot has no idea about the motion of the target. It only can find the location and the velocity of the target at each time instant. This is the reason why the robot could not take a shorter path.

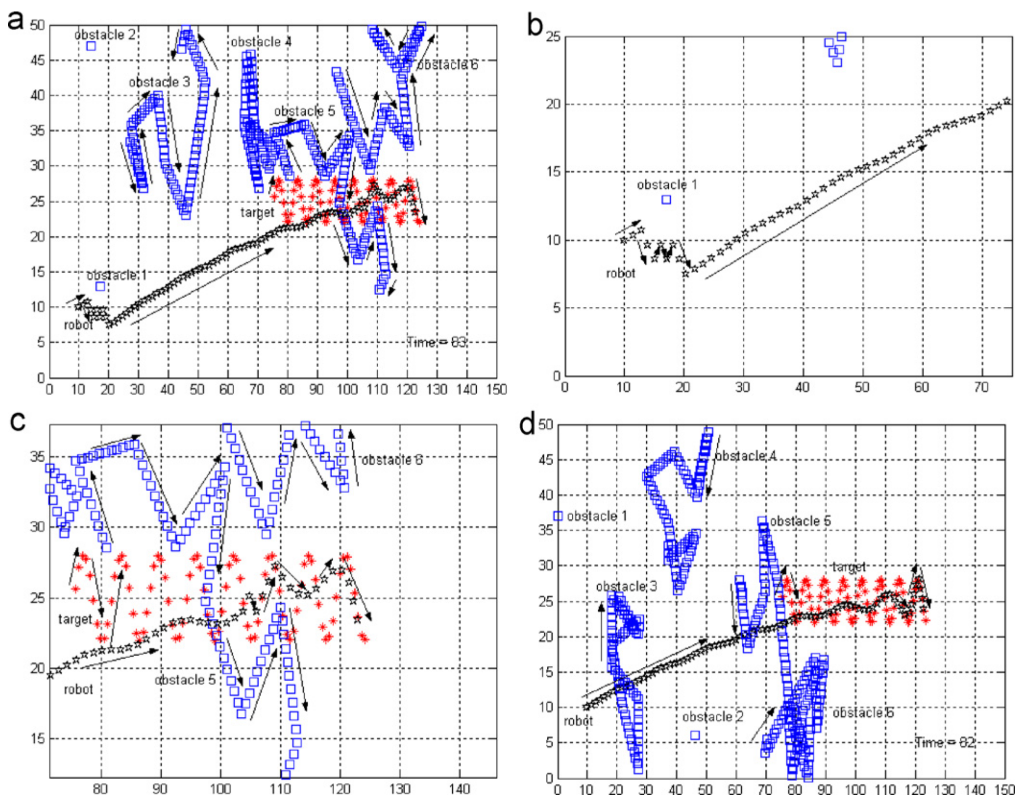


Fig. 7. (a) Path for the robot in the third testing scenario, (b) path for the robot in the third testing scenario displaying the avoidance of the first obstacle after time instant 3, (c) path taken in the third testing scenario displaying the avoidance of Obstacle5 and (d) path for the robot without any obstacle avoidance.

6.3. Hit/miss rates

Hit rate is the fraction of all trials the robot reaches its target successfully. Accordingly, the miss rate is the fraction of all trials the robot fails to reach its target. The path from the robot to the obstacle using the proposed Q-learning method is not unique and depends on the amount of training; since this training builds the experience for the robot to deal with different situations, it is expected that the more training the robot gets, the better its performance gets and hence the hit rate improves. In this section the effect of the training on the hit/miss rates will be covered.

The test scenarios that were used for studying the hit rate are random test scenarios. In these test scenarios, the target movement is sinusoidal. We have chosen this movement for the target because it is considered to be a complex one, where the target keeps changing its motion direction with time. The robot velocity in these testing scenarios is 2 m/s and the Turn Left and Turn Right angle is 45°. The location of the static obstacles and the starting point of the dynamic obstacles were chosen randomly and the motion of the dynamic obstacles was determined by the random walk model.

6.3.1. The effect of training

At the beginning, the effect of the amount of training on the hit and miss ratios will be investigated. To study this effect, the robot was trained using a varying number of random scenarios. Then the robot was tested using 500 test scenarios that were random scenarios too. The number of hits for each amount of training was counted, the miss rate was computed. The results of this study are shown in Table 1.

As it can be seen from the table, the miss rate decreases as the amount of training increases. At the beginning we started by only five training scenarios where the miss rate was 18.2%. After that we started to increase the amount of training until we reached 75 training scenarios, where the miss rate was 2%, which is a very small miss rate.

Table 1
The effect of training on the Hit/Miss ratios.

# of Training scenarios	#of Hits out of 500	Miss rate (%)
5	409	18.2
15	417	16.6
30	429	14.2
45	436	12.8
65	481	3.8
75	490	2
90	421	15.8
100	412	17.6

It can be also seen from the table that over training the robot with more than 75 scenarios increases the miss rate. This is due to over fitting that is a result of over training that causes the system concentrate on some cases. The relationship between the amount of training and the hit/miss rates is plotted in Figs. 8 and 9.

6.3.2. The effect of the number of obstacles

The other parameter was studied is the number of obstacles. We studied the effect of increasing the number of obstacles on the hit rate. The robot this time is trained for 50 times then it was tested using 500 random testing scenarios. We started by 3 obstacles 2 are dynamic and 1 is static. Then we started to increase the number of obstacles adding one dynamic obstacle and one static obstacle each time. In each trial we counted the number of hits and computed the miss rate. The results of this study are shown in Table 2.

From the table we can see that increasing the number of obstacles increases the miss rate. When the number of obstacles was 3 the miss rate was 2% which is a very low miss rate. The miss

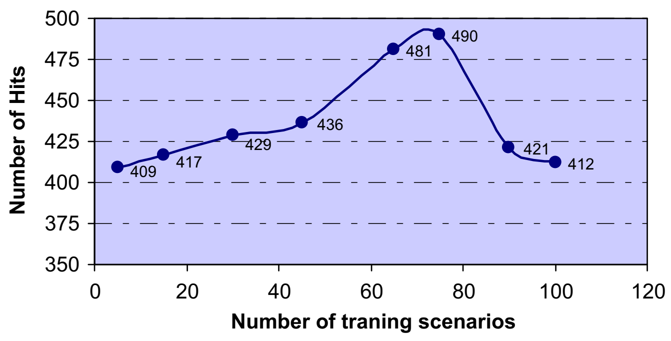


Fig. 8. The relationship between the amount of training and number of hits.

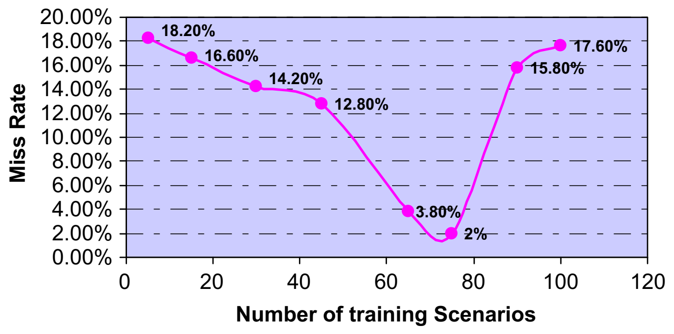


Fig. 9. The relationship between the amounts of training the miss rate.

Table 2
The effect of the number of obstacles on the Hit/Miss ratios.

# of obstacles	#of Hits out of 500	Miss rate (%)
3	490	2
5	488	2.4
7	484	3.2
9	483	3.4
11	407	16.28
13	378	24.4

rate started to increase with increasing the number of obstacles until it reached 24.4% when the number of obstacles was 13 which is not a desirable miss rate. This is a normal result because the increase of the number of obstacles gives the robot more difficult scenarios and increases the probability for collisions. Hence, if a large number of obstacles are expected in the working environment, the amount of training should be increased in order to decrease the miss rate.

The relationship between the number of obstacles and the Hit/Miss rates is plotted in Figs. 10 and 11. As it can be seen from the plots, the hit rate stays approximately constant as long as the number of obstacles is less than 10, then it starts to drop down if more obstacles were added to the robot environment. The miss rate was a very low miss rate as long as the number of obstacles is less than 10, then it started to increase until it reached 24.4%.

6.4. Comparison with the potential field method

The performance of the robot navigation method using Q-Learning was compared to the performance of the Potential Field Method. In the work of Ge and Cui [13], the authors present the application of the potential field method in dynamic environment. In this method, it is assumed that the robot moves

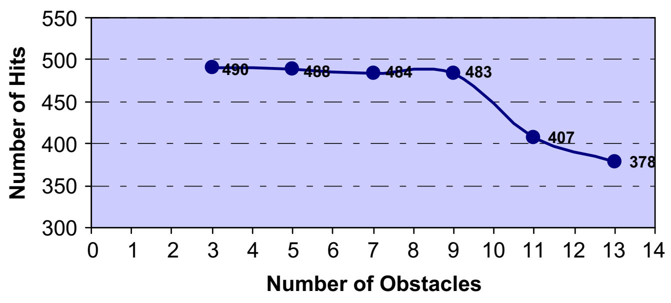


Fig. 10. The relationship between the number of obstacles and the number of Hits.

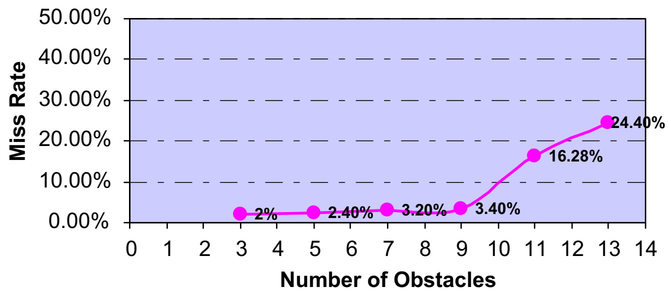


Fig. 11. The relationship between the number of obstacles and the miss rate.

under the effect of the forces of two potential fields. The attractive potential field generated by the target produces an attractive force (F_{att}) that attracts the robot to the position of the target all the time. The other field is the repulsive field generated by the obstacles. This field produces a repulsive force (F_{rep}) that moves the robot away from the obstacles. The robot moves under a virtual force which is the sum of the two forces (F_{total}) [13]

$$F_{total} = F_{att} + F_{rep} \tag{12}$$

The repulsive force is calculated as the sum of all repulsive forces generated by all the obstacles in the environment as follows [13]:

$$F_{rep} = \sum_{i=1}^{n} F_{repi} \tag{13}$$

where n is the total number of obstacles in the environment and F_{repi} is the repulsive force generated by the ith obstacle. Detailed information about the Potential Field Method for dynamic environment and the derivation of the equations can be found in [13].

In the above study a simulation was carried for testing the potential field method in a dynamic environment. A test scenario was applied for a certain environment that contains a moving target and six obstacles. Some of the obstacles were static and some were dynamic. In order to compare the new proposed artificial intelligent approach with the potential field method which is one of the most commonly used methods in the field of robotics for robot navigation the same testing scenario was repeated using the Q-learning algorithm.

The scenario assumes that the robot moves in a dynamic environment where the target moves at constant velocity, $v_{tar}=[0.1 \quad -0.05]^T$ starting its motion from point $[10 \quad 10]^T$. There are six obstacles in this environment: Obstacles 1, 2 and 3 are dynamic obstacles and obstacles 4, 5 and 6 are stationary obstacles. Obstacle 1 starts its motion from point $[5 \quad 0]^T$ and at constant velocity $[0.0 \quad 0.28]^T$. Obstacle 2 moves at velocity $[-0.29 \quad 0]^T$ starting from point $[9.4 \quad 4.5]^T$. Obstacle 3 moves at velocity $[-0.05 \quad -0.065]^T$ from point $[19 \quad 10]^T$. Obstacle 4 is

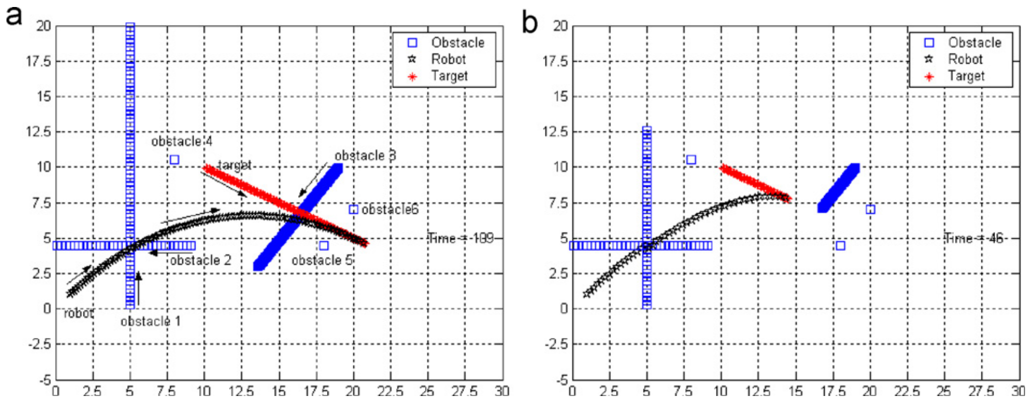


Fig. 12. (a) Path of robot, target and obstacles using Q-learning algorithm and with robot velocity=0.2 m/s and (b) path of robot, target and obstacles using Q-learning algorithm and with robot velocity=0.35 m/s.

centered at point $[8 \quad 10.5]^T$, Obstacle 5 is located at point $[18 \quad 4.5]^T$, and Obstacle 6 at point $\left[20 \quad 7\right]^T$. The robot started its motion from point $\left[1 \quad 1\right]^T$.

Fig. 12 shows the result for applying the same scenario using the Q-learning method. The only issue in applying the same scenario is in the robot speed. The speed for the robot using the potential field method is varying depending on the strength of the net potential field. On the other hand, the robot using the Q-learning method moves according to Eq. (9) using constant v_{rob}. However, in order to make a valid comparison we computed the average speed of the robot in the potential field scenario by dividing the total passed distance by time, and moved our robot in a speed slower than this average speed. In the potential field scenario the average speed for the robot was v_x=0.236 m/s and v_y=0.472 m/s. In our scenario we chose v_{rob}=0.2 m/s.

As it can be seen from Fig. 12, the robot succeeded in its mission to catch the target at point $[20.9 \quad 4.55]^T$. It took the robot 109 s to reach the target. This is approximately half the time need by the robot to reach the target using the potential field method. In this scenario the robot did not have to avoid any of the obstacles since it was very slow so that Obstacles 1 and 2 intersected and passed in their ways before the robot reach the intersection point. Also, Obstacle 3 passed before the robot gets close to it, so no avoidance occurred. This proves that in our scenario the robot moves at a much slower speed and yet it was able to accomplish its mission in half the time required for potential field method.

As a further step, the robot speed was increased to v_{rob}=0.35 m/s to make it face Obstacles 1 and 2 and make avoidance steps. Fig. 12b presents the result from applying this robot speed on the same scenario. As it can be seen the robot velocity is very close to the velocity of the robot in the potential field method. The robot in this scenario avoided Obstacle 1. Obstacle 2 was faster, and no avoidance occurred. The robot reached the target at time 46 s which is much shorter than the time required by the potential field method. This proves the efficiency of the new approach.

From the previous discussion, it can be seen that the new approach is a very efficient approach, and can accomplish the navigation process for a much shorter time. Moreover, the potential field method suffers from the local minimum problem when the robot, the target and the obstacle are on the same line and the obstacle is between the target and the robot Fig. 13 [13].

As it can be noticed from the figure, as the obstacle gets closer to the robot, then the repulsive force is greater than or equals the attractive force, so the total force will take the robot away from the target, or if the total force is zero, the robot stops. Using the potential field method the robot will wait until the obstacle changes its location and get away from the path to the target. On the other hand, the new Q-learning approach does not suffer from such a problem because it does not deal with forces. Fig. 14 shows the robot path when the robot, the obstacle and the target are on the same line; and the obstacle is between the robot and the target.

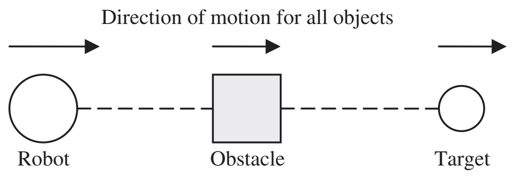


Fig. 13. Local minima problem [13].

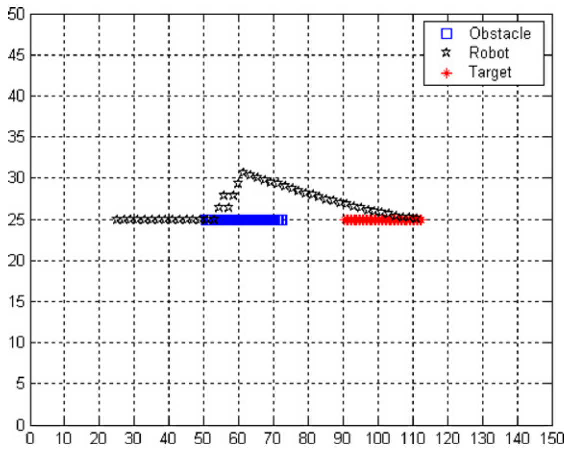


Fig. 14. Path of robot when the robot, obstacle and target are on the same line and obstacle is between the robot and the target.

In this figure, the target started its motion at point $\left[90 \quad 25\right]^T$ with velocity $[0.5 \quad 0]^T$, the obstacle started its motion at point $\left[50 \quad 25\right]^T$ and with velocity $[0.5 \quad 0]^T$ which is the same velocity of the target. The robot started its motion from point $\left[25 \quad 25\right]^T$ and the velocity of the robot v_{rob}=2 m/s.

As it can be seen the robot was able to avoid the obstacle and reach the target. From the figure we can see that the robot was moving in a straight line until second 15. At this time the robot entered the *Non-Safe* region of the moving obstacle, so the robot turned left. As a result for executing this action with the obstacle still moving forward, the robot entered the *Safe* region. The robot

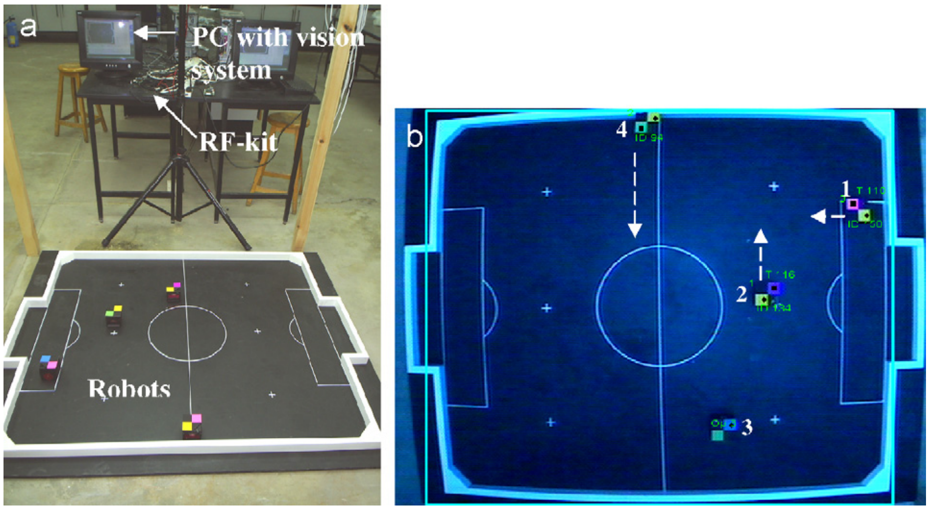


Fig. 15. Multi-robot experimental system: (a) robot system and (b) experiment layout.

changed its orientation toward the target. This action returned the robot to the *Non-Safe* region. The robot turned left again band moved in a straight line in this direction until second 21. At this time instant, the robot was at point $[61 \quad 30.6]^T$ and the obstacle was at point $\left[61 \quad 25\right]^{\mathrm{T}}$. The distance between the robot and the obstacle is 5.6 m, and as we stated before the boundary for the *Non-Safe* region is 5 m. Hence, this action returned the robot back to the *Safe* region so the robot changed its orientation toward the target. After that the robot did not enter the *Non-Safe* region again and it started heading toward the target until catching it at time 46 s.

It can be concluded from the previous discussion that the robot can deal with difficult situations and reach its target in a short time. It can be more efficient than the potential field method. In this section the robot was trained for 75 times before applying the comparison tests.

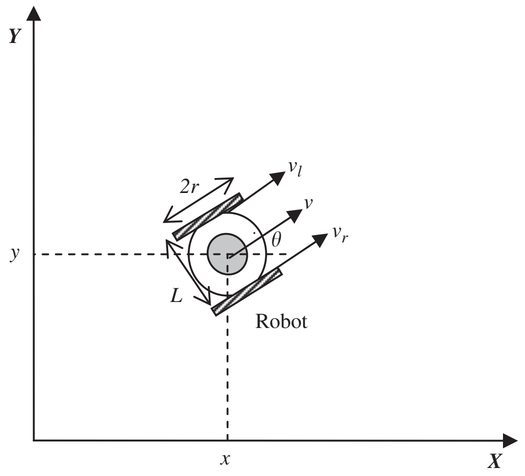


Fig. 16. Differential drive mobile robot.

6.5. Experimental results

For the experimental evaluation a multi-robot soccer system is considered as a dynamic environment, as shown in Fig. 15(a). The system mainly consists of the following parts: host computer, vision system, RF communication modules, and wheeled mobile robots.

The robot is a differential drive wheeled mobile one with inboard microprocessor, two DC-motors with encoders, and a RF communication module. The kinematics for the differential drive mobile robot under no slipping, pure rolling and no sliding constraints is

$$\dot{P} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{14}$$

where P is the posture vector as shown in Fig. 16, v is the robot translation velocity, and ω is the robot rotation velocity defined with respect to the robots center. The overall robot translation and rotation velocities are determined based on the two wheels rotation velocities as follows:

$$v = \frac{r}{2}(\omega_r + \omega_l) \tag{15}$$

$$\omega = \frac{r}{L}(\omega_r - \omega_l) \tag{16}$$

where the ω_r and ω_l are the robot right and left wheels angular velocity respectively, r is the nominal wheel radius, and L is the distance between the two wheels centers as shown in Fig. 16.

The robot is controlled to update its posture by determining the right and left wheels translation velocities (v_r and v_l). The proposed approach will drive the robot with the desired direction (θ_d) based on the selected state which minimizes the error:

$$\Delta\theta = \theta_d - \theta \tag{17}$$

The control command used to update the robot posture is based on proportional control as in Eqs. (18) and (19), where K is the proportional gain constant:

$$v_r = v + K\Delta\theta \tag{18}$$

$$v_l = v - K\Delta\theta \tag{19}$$

During the experimental evaluation for the proposed approach, the robot (1) should avoid obstacles (2) and (3) while tracking the moving target robot (4), as shown in Fig. 15b. Obstacle (2) is a moving one, while obstacle (3) is stationary. The white arrows indicate the movement direction for the robot, the target and the moving obstacle. Snapshots for the conducted experiment are shown in Fig. 17a–e.

As shown in Fig. 17a and b, the robot (robot 1) starts moving form its initial location moving toward the moving target

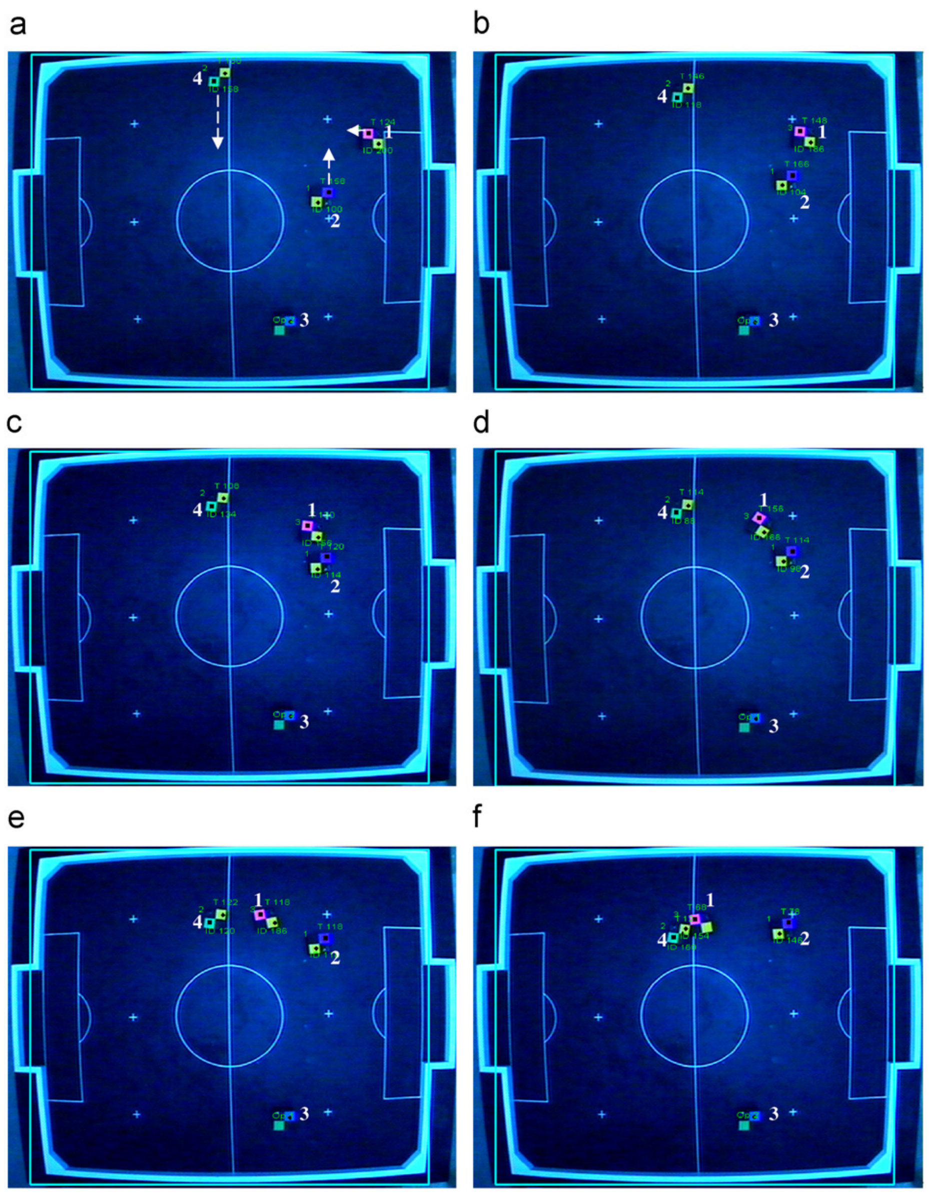


Fig. 17. Experimental evolution for the proposed approach.

(robot 4) within a defined safe state by the proposed approach. The robot entered a non-safe state in Fig. 17c where the robot is meeting a moving obstacle (robot 2), at this instant the robot was successfully able to update its position and avoid the obstacle as shown in Fig. 17c. After that the robot entered the safe sate again, it moved toward the target again to meet with it as shown in Fig. 17d and e. the conducted experiment has validated the new approach for mobile robot path planning in dynamic environments.

7. Conclusions

In this research a new approach for mobile robot navigation in dynamic environment was presented using the Q-learning algorithm. The Q-learning algorithm helped in solving the problem of motion planning without having a model for the environment since the environment is completely unknown for the robot. No previous constrains were assumed about the environment or about the target or obstacles movements. In order to be able to apply this algorithm in dynamic environment a new definition for the state space was created. This new definition tries to mimic the human reasoning for dealing with such unknown environments.

Different simulated test scenarios were conducted to test the robot ability to deal with different types of dynamic environments. Different types of target motions and different number of obstacles with different dynamicity patterns were implemented in the testing scenarios. The test scenarios demonstrated the robot ability to deal with different environmental situations and to reach its target all the time. The computed miss rate was 2%, the miss rate started to increase with increasing the number of obstacles until it reached 24.4% when the number of obstacles was 13. The performance of the robot navigation method using Q-Learning was compared to the performance of the potential field method in dynamic environments; using the new approach the robot can accomplish the navigation process for a much shorter time, approximately half the time needed by the robot to reach the target using the potential field method. The presented experimental results and conducted simulation scenarios reflects the strength of the proposed approach to be adapted for robot navigation in unknown dynamic environments.

References

[1] Minguez J, Montano L. Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios: real-time obstacle avoidance for fast mobile robots. Robotics and Autonomous Systems 2005;52:290–311.

[2] Chakravarthy A, Ghose D. Obstacle avoidance in a dynamic environment: a collision cone approach. IEEE Transactions on Systems, Man, and Cybernetics 1998;28(5):562–74.

[3] Bilgic T, Burhan I. Model-based localization for an autonomous mobile robot equipped with sonar sensors. IEEE International Conference on Systems, Man, and Cybernetics 1995;4:3718–23.

[4] Mucientes M, Iglesias R, Regueiro CV, Bugarin A, Carifiena P, Barro S. Fuzzy temporal rules for mobile robot guidance in dynamic environments. IEEE Transactions on Systems, Man, and Cybernetics 2001;31(3):391–8.

[5] Filliat G, Mayer J. Map based navigation in mobile robots: I. A review of localization strategies. Cognitive System Research 2003;4:243–82.

[6] Mayer J, Filliat G. Map based navigation in mobile robots: II. A review of map learning and path planning strategies. Cognitive System Research 2003;4: 283–317.

[7] Malki A, Lee J, Lee S. Vision based path planning for mobile robot using extrapolated artificial potential field and probabilistic obstacle avoidance. ASME International Mechanical Engineering Congress and Exposition 2002:133–9.

[8] Song K, Chang C. Reactive navigation in dynamic environment using a multisensor predictor. IEEE Transactions on Systems, Man, and Cybernetics 1999;29(6):870–80.

[9] Pratihar DK, Deb K, Chosh A. A genetic-fuzzy approach for mobile robot navigation among moving obstacles. International Journal of Approximate Reasoning 1999;20:145–72.

[10] Russell S, Norvig P. Reinforcement learning in: artificial intelligence a modern approach. 2nd ed.. New Jersey: Prentice-Hall; 2003 p. 763–88.

[11] Borenstein J, Koren Y. Real-time obstacle avoidance for fast mobile robots. IEEE Transactions on Systems, Man, and Cybernetics 1989;19:1179–87.

[12] Ge S, Cui Y. New potential functions for mobile robot path planning. IEEE Transactions on Robotics and Automation 2000;16:615–20.

[13] Ge S, Cui Y. Dynamic motion planning for mobile robots using potential field method. Autonomous Robots 2002;13:207–22.

[14] Tsourveloudis N, Valvanis K, Herbert T. Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic. IEEE Transactions on Robotics and Automation 2001;17.

[15] Joo M, Deng C. Obstacle avoidance of a mobile robot using hybrid learning approach. IEEE Transactions on Industrial Electronics 2005;52(3):898–905.

[16] Kim D, Kim J. A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer. Robotics and Autonomous Systems 2003;42:17–30.

[17] Hwang K, Tan S, Chen C. Cooperative strategy on adaptive Q-learning for robot soccer system. IEEE Transactions on Fuzzy Systems 2004;12(4):569–76.

[18] Watkins C. Learning from delayed rewards. PhD dissertation, King's College, 1989.

[19] Aranibar D, Alsina P. Reinforcement learning-based-path planning for autonomous robots ENRI: Encontro Nacional de Robótica Inteligente, 2004.

[20] Yang G, Chen E, An C. Mobile robot navigation using neural Q-learning. In: IEEE proceedings of the third international conference on machine learning and cybernetics, vol. 1(26–29), 2004. p. 48–52.

[21] Harmon M, Harmon S. Reinforcement learning: a tutorial, available via 〈http://citeseerx.ist.psu.edu〉, 1996.

[22] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research 1986;5:90–8.

[23] Smart W, Kaelbling L. Practical reinforcement learning in continuous spaces. In: Proceedings of the seventeenth international conference on machine learning, 2000. p. 903–10.

[24] Smart W, Kaelbling L. Effective reinforcement learning for mobile robots. In: Proceedings of the international conference on robotics and automation, 2002. p. 3404–10.

[25] Boem H, Cho H. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. IEEE Transaction on System, Man, and Cybernetics 1995;25:464–77.

[26] Yung N, Ye C. Self-learning fuzzy navigation of mobile vehicle. In: Proceeding of the international conference on signal processing, 1996. p. 1465–8.

[27] Er M, Zhou Y. Automatic generation of fuzzy inference systems via unsupervised learning. Neural Networks 2008;21(10).

[28] Er M, Deng C. Obstacle avoidance of a mobile robot using hybrid learning approach. IEEE Transaction on Industrial Electronics 2005;52:898–905.

[29] Macek K, Petrovic I, Peric N. A reinforcement learning approach to obstacle avoidance of mobile robots. In: 7th international workshop on advanced motion control, 2002. p. 462–6.

[30] Park J, Kim J, Song J. Path Planning for a robot manipulator based on probabilistic roadmap and reinforcement learning. International Journal of Control, Automation, and Systems 2007;5:674–80.

[31] Lin L. Scaling-up reinforcement learning for robot control. In: Proceedings of the 10th international conference on machine learning, 1993. p. 182–9.

[32] Park K, Kim Y, Kim J. Modified uni-vector field navigation and modular Q-learning for soccer robots. In: Proceedings of the 32nd international symposium on robotics, ISR, 2001.

[33] Ellery A. Environment–robot interaction—the basis for mobility in planetary micro-rovers. Robotics and Autonomous Systems 2005;51:29–39.

[34] Humphrys M. Action selection methods using reinforcement learning, PhD thesis, University of Cambridge, 1997.