Diyala Journal of Engineering Sciences

ISSN 1999-8716

Printed in Iraq

Vol. 05, No. 01, pp. 13-24, June 2012

OPTIMUM SHORT PATH FINDER FOR ROBOT USING Q-LEARNING

Mohannad Abid Shehab Ahmed
Assistant Lecturer
Electrical Department, Engineering College, Al-Mustansirya University
(Received:27/9/2009 ; Accepted:3/1/2011)

ABSTRACT:- Programming robots is a useful tedious task, so there is growing interest in building robots which can learn by themselves. This paper describes the Reinforcement Learning and teaching approach like Queue Learning (Q-Learning) to be implemented for robotics technology environment navigation and exploration. Q – Learning algorithm is one of the widely used online learning methods in robotics; it is simple, efficient, and not need to complex process as in adaptive system. The aim of this work is to empower the agent to learn a certain goal directed navigation strategy and to generate a shortest path in static environment which contain static obstacles; it uses one of the important intelligent search methods the "heuristic". It makes a necessary modification for the search algorithm to suit the way of solving the problem. In our approach of learning from demonstration, the robot learns a reward function from the demonstration and a task model from repeated attempts (trials) to perform the task. A simplified reinforcement learning algorithm based on one-step Q-Learning that is optimized in speed and memory consumption is proposed and implemented in Visual Basic language (VB). The robot can be built using stepper motors and any available microcontroller like 89c52 with its driver circuit to utilize of their matching.

Keywords: Reinforcement Learning, Q-Learning, Navigation, Robot, microcontroller.

1. INTRODUCTION

In goal-directed navigation tasks, an autonomous moving agent has solved its task when having a certain reached location in space. Reinforcement Learning (RL) is frequently applied to such tasks, because it allows an agent to autonomously adapt its behavior to a given environment. It has proven to be an effective approach especially in conditions of uncertainty. However, in large and in continuous state spaces RL methods require extremely long training times. The navigating agent learns a strategy that will bring it to the goal from every position within the world, that is: It learns to select an action for every given observation of the given environment, but usually this strategy cannot be transferred to other scenarios, because knowledge of the underlying structure of the state space is not explicitly acquired [1].

The state space is a tow dimensional array with cell as a starting point and cell as a goal there may also be cells occupied by obstacles.

The first goal of this work is to provide a representation that leads to a small and discrete state space which enables fast and robust learning of a navigation strategy in a continuous, non-homogeneous world. The second goal is to explicitly model structural elements of the environment within this representation mapping to enable the agent to re-use

learned strategies in structurally similar areas within the same world and also being able to transfer learned strategies to other unknown environments.

An autonomous robot can be considered as physical device which performs a task in a dynamic and unknown environment without external help, the environment has been represented with all its components (static obstacles, dynamic obstacles, robot, and mobile robot) in area of two dimensions.

In many research areas like robotics, the computer program have come to play an important role as scientific equipment to provide the powerful devices for the virtual experiment as opposed to physical ones which are difficult to design, costly or even worse dangerous. As robots are used to perform increasingly difficult tasks in complex and unstructured environment they become more of the challenge to reliably program their behavior, there are many variation for a robot control program but they are often too complex to be adequately managed with static behavior hand coded by a programmer robotic system sometimes have undesirable limitation in their robustness efficiency or competence [2].

Today's robotic machines are faster, cheaper, more repeatable and reliable. More attention is being given by industry to robots vision and motion controls. New areas of usage are emerging for service robots, remote manipulators and automated guided vehicles. Ideally, all intelligent robots:

- Move dexterously, smoothly, precisely, using multiple degrees of coordinated motion and do something like a human but that a human now doesn't have to do.
- Have sensors that permit them to adapt to environmental changes.
- Learn from the environment or from humans without mistakes.
- Perform automatically, tirelessly, accurately.
- Diagnose their own problems, and repair them.
- Can reproduce, not biologically but by robots making robots.
- Can be used in industry for variety of applications.
- They mimic expert human responses.

2. NAVIGATION PLANNING

The task considered within this work is a goal-directed navigation task. A robot is requested for finding the optimum and best paths from start to goal points with identifying the obstacles on its trajectories, the navigational planning problem persists in both static and dynamic environment in a static environment the position of obstacles is fixed while in a dynamic environment the obstacles may move at arbitrary directions with varying speeds which are lower than the maximum speeds of the robot in order to avoid collision between them. A robot has to generate a navigational plan to a given environment between predefined starting and goal points, the robots environment may include many obstacles and thus finding the shortest path without touching the obstacles in many cases is an extremely complex problem, robots are equipped with various types of sensors to recognize the world around them, in most of the commercial robots where navigation is required in a closed environment (like factory) ultrasonic sensors are adequate for long range navigation of the order of 100 meter or more laser finders are occasionally employed in robots further for determining the shape and structure of the objects.

The term planning refers to the generation of the sequences of action in order to reach a given goal state in the navigational planning problem. Optimization problem is concerned with identification of the paths having shortest distance with optimum time needs for traversal searching between the starting and the goal point. Depending on the types of planning problem the robots environment is represented by a specified formalization such as tree graph, portioned blocks and trajectories. Thus in most cases before presenting a technique for handling the planning problems it must represent the environment by some structure.

The navigation can be developed according to "manual development" which depends on human hands, and "autonomous development" which depends on the artificial intelligent subjected to the environment mapping [3].

The proposed robot of this project is not provided with any sensor and works with autonomous development. It depends on the submitted environment mapping which has (start, goal, obstacles (static type)) without human intervention.

3. Q - Learning

The original Queue Learning is a simple incremental algorithm developed from the theory of dynamic programming for delayed reinforcement learning. In Q-Learning, policies and the value function are represented by a two-dimensional lookup table indexed by state-action pairs.

The robot task is formalized as a Markov Decision Process (MDP) [4]. An MDP is a group (S, A, T, R). S is a finite set of states s, some of which may be terminal states. A is a finite set of actions a, whose availability may depend on the state.

T: S × A × S → [0, 1] defines the state transition function that describes the probability p (s′|s, a) that the system will move from state s to s′ after performing the action a ∈ A.

R: S × A × S → IR defines the expected immediate real valued reward r(s, a, s′) when action (a) is taken in state s and the transition to s′ is made [4].

Formally, S_t is the state at time t with action a_t, the immediate action takes a reward r which is received by learner of the environment state space to give a correct translation to next state S_{t+1}. The Q-learn policy is:

$$Q(S_t, a_t) = R(S_t, a_t) + \gamma . \sum_{y} P_{sy}(a) . V(y) \qquad \ldots\ldots (1)$$

Equation (1) is concerned with exploration purpose; this policy is used to guide the exploration. With probability γ, the agent explores the search space, and with probability 1- γ, it goes along the best path already found.

The Q-learn algorithm works with

$$Q(S_t, a_t) = (1 - \alpha) . Q(S_t, a_t) + \alpha (r_{t+1} + \gamma . V(y)) \qquad \ldots\ldots (2)$$

Equation (2) is concerned with optimum path finding purpose.

Where:

$Q(S_t, a_t)$: is the expected discount return by taking action (a_t) at state (S_t). $R(S_t, a_t)$: is the immediate real value reward with action (a_t) at state (S_t).

$$\mathbf{V(y)} = max_{b \in actions} Q(y, b) = max . Q(S_{t+1}, a') \qquad \ldots\ldots (3)$$

α : is the constant step size parameter (learning rate), .

γ : is the discount factor, which indicates the influence of the future rewards on the current state, $\gamma \in [0\ 1]$, usually γ is closer to 1 [2],[5].

r : is the instant reward received after action at is performed.

$$r = \begin{cases} 1 & reward \\ 0 & don't\ care \\ -1 & punshiment \end{cases}$$

Q is improved gradually and the agent learns to maximize the future rewards.

It is important to note that the Q-Learning method does not specify what actions the agent should take at each state as it updates its estimates. In fact, the agent may take whatever actions it pleases. This means that Q-Learning allows arbitrary experimentation while at the same time preserving the current best estimate of state s' values.

In order to find a short path, the agent must explore the search space (e.g., maze) actively at the beginning. But as the exploration proceeds, it assumes the agent can find a good (short) path, so the agent should gradually focus on the path that has been found.

4- Q – Learning Software

The following steps are implemented to perform the Q – Learning algorithm as:

- Put the agent at the start point.
- While (not find goal) do:
  - Initiate the values of α and γ, (α = 0.99, γ = 0.9).
  - Use the Q-learn policy to select an action a_t
  - Take the action to get (S_t, a_t, r_t, s_{t+1}).
- Apply the Q – Learning equation to update Q – value.

$$Q(S_t, a_t) \leftarrow (1 - \alpha).Q(S_t, a_t) + \alpha(r_{t+1} + \gamma.max.Q(S_{t+1}, a'))$$

- If goal is reached then
- Change the value of γ and put the agent back at the starting point.
- Stop

The above steps can be clearly showed in figure (1) .

The state space is a two dimensional array with cells as the starting point, goal and obstacles, the agent can move from a cell to any one of the 8 direct neighbors with the restriction that the agent cannot move out of the state space or into a cell occupied by an obstacles the task of the agent is to find a good path from the starting point to the goal efficiently, it can understand the principle of Q-Learning utilizing the operation of agent (robot) in state space (map) as shown in block diagram of figure (2).

Q-Learn can formulate and program the problem with well defined states, actions, reward, and the Q-function, each one can be explained briefly as:

States:

The state of the robot is defined by four attributes:

- **Robot Heading**

  The heading of the robot is any one of the 8 direct neighbors, in figure (*) the heading of robot can be represented by the arrows

- **Target Bearing**

  Same as the state of heading, target bearing is a range from (0-3) where:
  0: target at front
  1: target at right side
  2: target at behind
  3: target at left side

- **Hit with Wall**

  When a robot collides with one side of the wall, two discrete values are used to represent this state, '1' represents not hit the wall and '-1' represents hit the wall.

- **Hit by obstacle**

  When a robot hit by an obstacle, the state is defined by '-1' while '1' represents not hitting is occur.

Different combination of the above four attribute will contribute to a state. For a robot, there are total of 128 (8 x 4 x 2 x 2) possible states.

Actions:
Six actions are defined in this project:
- Move ahead
- Move back
- Move ahead and Turn Left
- Move ahead and Turn Right
- Move back and Turn Left
- Move back and Turn Right

So, the state and action pair take a total number of entry = 128 x 6 = 768 entry

Reward:

A reward is received after the execution of certain action. It is determined from the change of states after the execution of an action. The reward of the robot is changed in the following situation [6].

- The reward of the robot is increased when no collision with wall or obstacle
- Negative reward for a collision with the wall
- Negative reward for a collision with obstacle

Q-FUNCTION

The Q-function Q(s, a) is represented in a table with the use of two dimension array, each entry of the array is the Q-value of the corresponding state and action pair.

The total number of entry is 768 after the end of each round, the values are stored in a file restricts the creation of size equal or less than 2Kbyte, this suitable capacity allows us to implement the robot using any 8-bit or 16-bit microcontroller like 8051, 8052,…etc.

In this work, the Q-Learning program is built using Visual Basic language (VB) and its GUI environment is as shown below in figure (4), it consists of variable dimension maze, start, goal and obstacle points, it is required to identify the best path from start to goal (target) avoiding colliding the wall and the obstacles and calculate the required robot navigation real time.

Two examples are given here,

First, let a simple maze with 8×8 cells with no obstacle as shown in figure (5), where the start point is at node 50 and the target is at node 6.

The Q-Learning Robot needs to 58 seconds to complete its search successfully for finding the best shortest path which is the green line above and its operation curve is in figure (7) which shows changes of the received rewards during the 58 second time.
Secondly, let a maze with 10×10 cells with four obstacles as shown in figure (6)

The Q-Learning Robot needs to about 100 seconds to complete its search successfully for finding the best shortest path and the operation curve is shown in figure(8) which shows changes of the received rewards during the 100 second time.

From the obvious figures (5) and (7), it is clear that the Q-table actions and states was randomly generated at the beginning of each experiment where several trails were carried out in order to find the best learning performance.

At the starting, most of these trails fail to get the path and as a result heavy negative rewards are generated and it still growth exponentially due to standard Boltzman Exploration Function which assign probabilities of action selection p(s,a) as:

$$p(s,a) = \frac{e^{\frac{Q^{+}(s,a)}{T}}}{\sum_{b} e^{\frac{Q^{+}(s,b)}{T}}} \quad \ldots\ldots (4)$$

Where: T is the Boltzman temperature, is the variable to control exploration function.

After passing certain amount of time depending on the state space dimension and number of obstacles the Q-Learning is speed up and the positive rewards will be generated, at the end the number of trails will be reduced and the system converges to stable behavior. Variations in the received rewards at some steps are because the robot is in different situations of the maze so the robot has to change the direction many times; also the variation is increased as the number of obstacles increased.

It is cleared that the last transitions of agent in figure (7) is point 36 → 37 → 28 → 39 rather than the straight path 36 → 37 → 38 → 39, this is because the Q-Learning policy try to deviate from the obstacle with same path transition cost.

CONCLUSIONS AND FUTURE WORK

Robotics technology has become widely accepted and is now being used expensively in industrial, service, and educational fields.

In this work, a robot programming that navigates in state space to get an optimum path with obstacle avoidance is implemented using Queue learning algorithm which is one of the most efficient and effective Artificial Intelligence representations.
The following are some conclusions that reached along this project:

1. The mobile robot cannot operate in any environment without an efficient navigation system that modifies mobile robot performance as stand alone.

2. The navigation system should supply the mobile robot with the ability of hill and obstacle avoidance and the ability of moving from point to another with the shortest path between them.

3. The representation used to achieve the above fundamental must be flexible easy and has clear operations and steps.

4. A mobile robot navigation system using queue learning algorithm behavior makes the mobile robot movement and task achieving more safe and reliable.

5. Instead of using the adaptive systems that need to complex algorithms, it can use the simple Q-Learning to achieve many optimal requirements.

6. It can expand the operation of Q-Learning generation in the Integrated Circuit (IC) fabrication where it needs to best path in many process as in IC metallization process.

7. It can expand the operation of Q-Learning generation in the networking or power system distribution where it needs to shortest paths between the system nodes.

In the future it might work to solve the problem of the mobile robot navigation system in dynamic obstacles environments with its two types (modeled & non- modeled), and taking the effects of the mobile robot speed and acceleration concerned with its movement and path planning.

Two solutions are currently exploring, combining the Q-learning coordination mechanism with other machine learning algorithms which can adjust behavioral parameters in response to environmental pressures, and improving the Q-Learning calculations by reducing the number of action and state trials and iterations through adapting efficient numerical methods like sparse matrix concept.

REFERENCES

1. Lutz Frommberger, "A Generalizing Spatial Representation for Robot Navigation with Reinforcement Learning", Universitat Bremen, 2007.

2. Ulrich Nehmzow, "Mobile robotics a practical introduction", 2nd Edition, Department of Computer Science, The University of Essex, 2003.

3. The Internet Encyclopedia, http://en.Wikipedia.org/wiki/Q-Learning.

4. Bram Bakker, Viktor Zhumatiy, Gabriel Gruener, and Jurgen Schmidhuber, "Quasi-Online Reinforcement Learning in Robots", Informatics Institute, University of Amsterdam, the Netherlands, 2004.

5. Vanden Berghen Frank, "Q-Learning", IRIDIA Unversit Libre de Bruxelles, 2003.

6. William D. Smart and Leslie Pack Kaelbling, "Effective Reinforcement Learning for Mobile Robots", 2002.

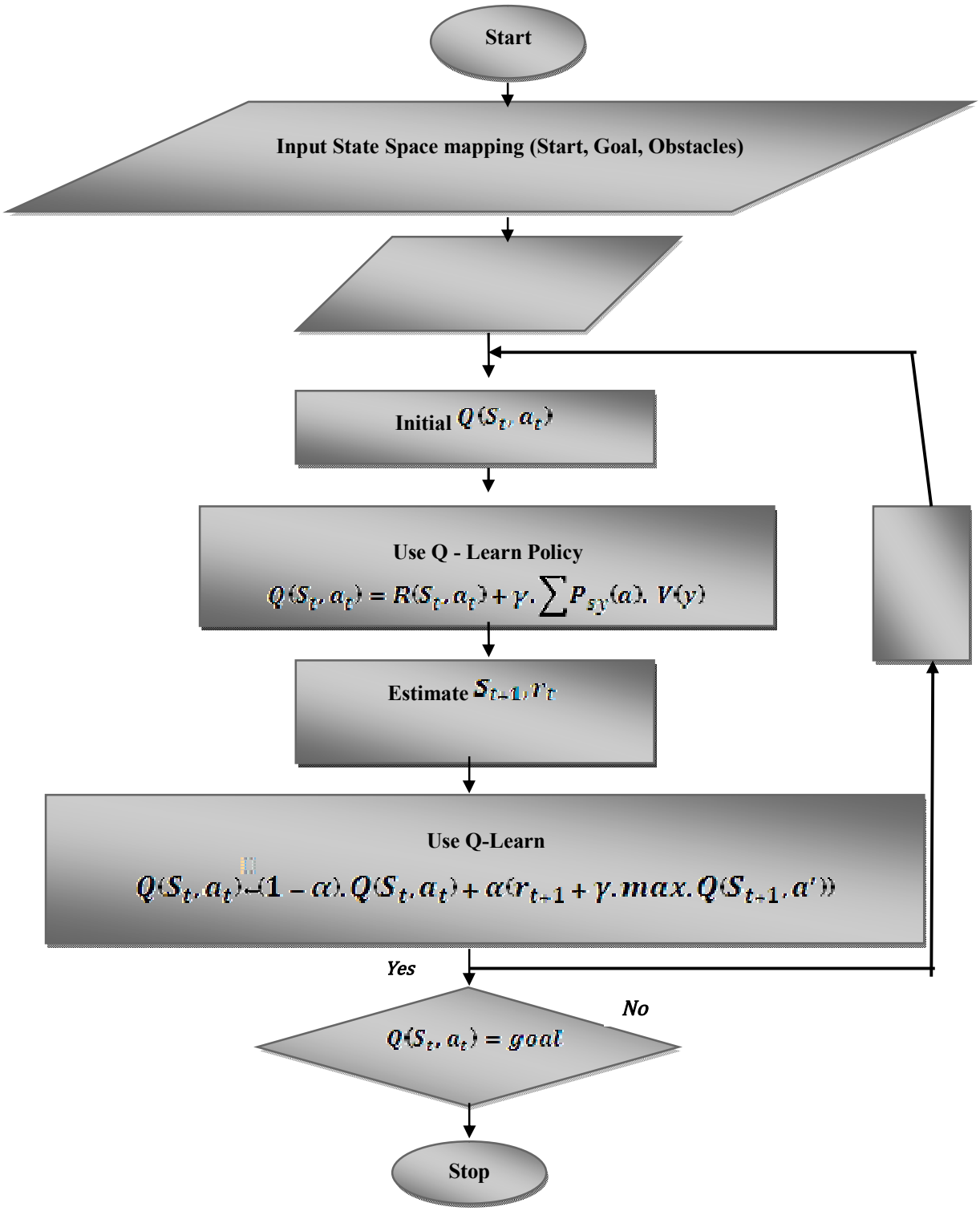

Fig.(1): Q-Learning flow chart steps.

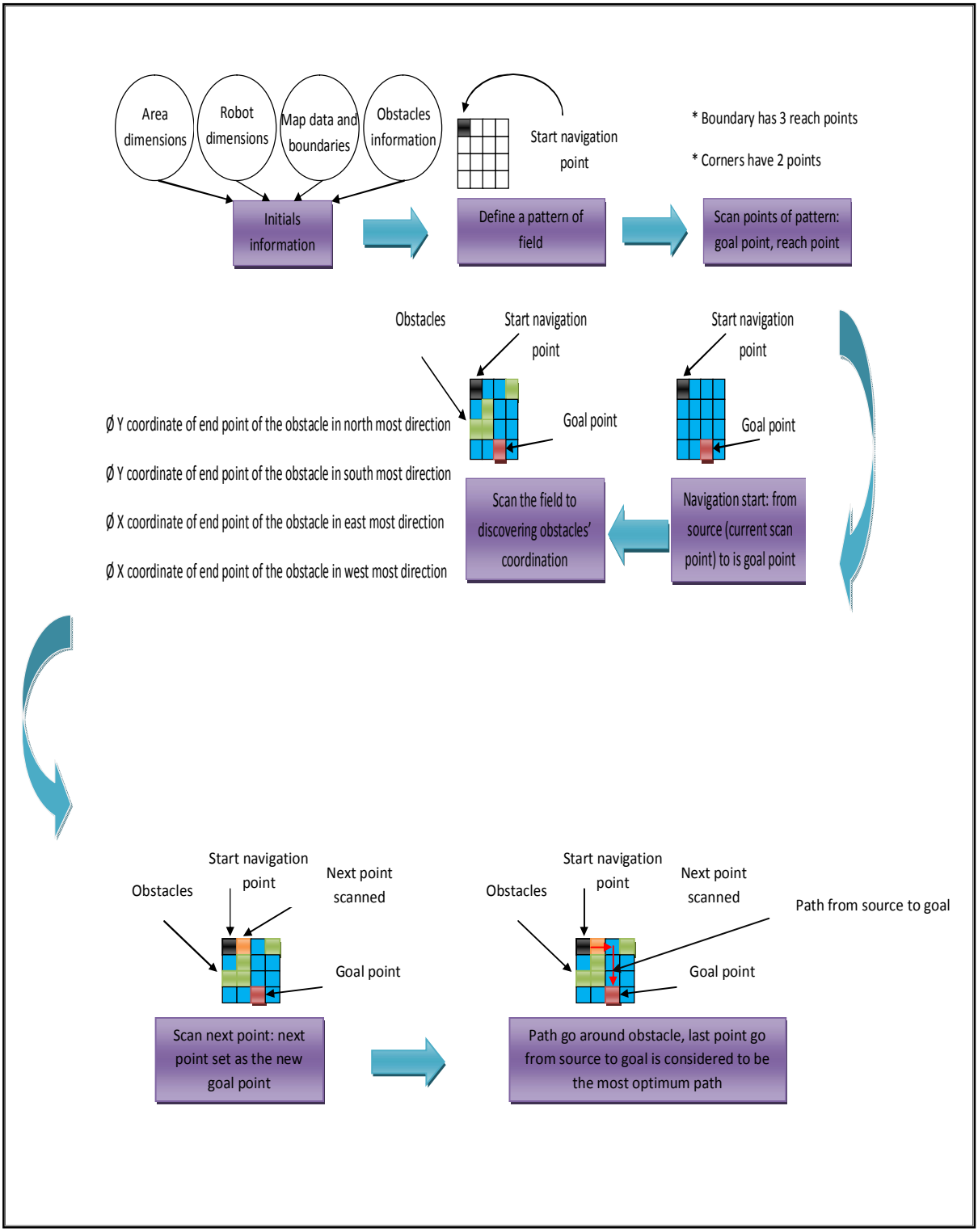

Fig.(2): Block diagram operation of agent (robot) in state space (map)

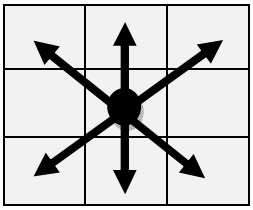

Fig.(*): shows the possibilities of heading of robot.

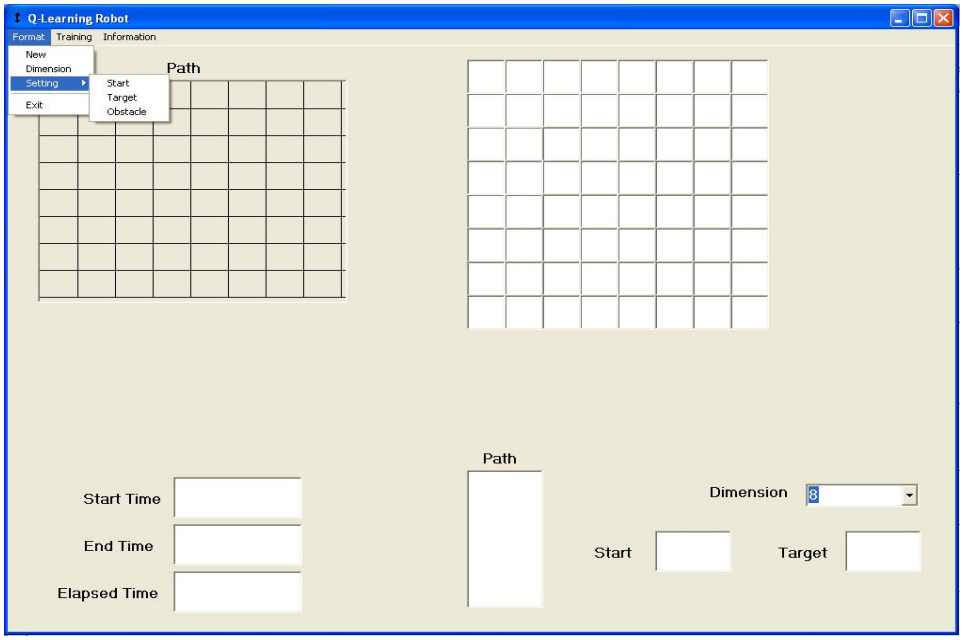

Fig.(4): GUI Q-Learning environment.

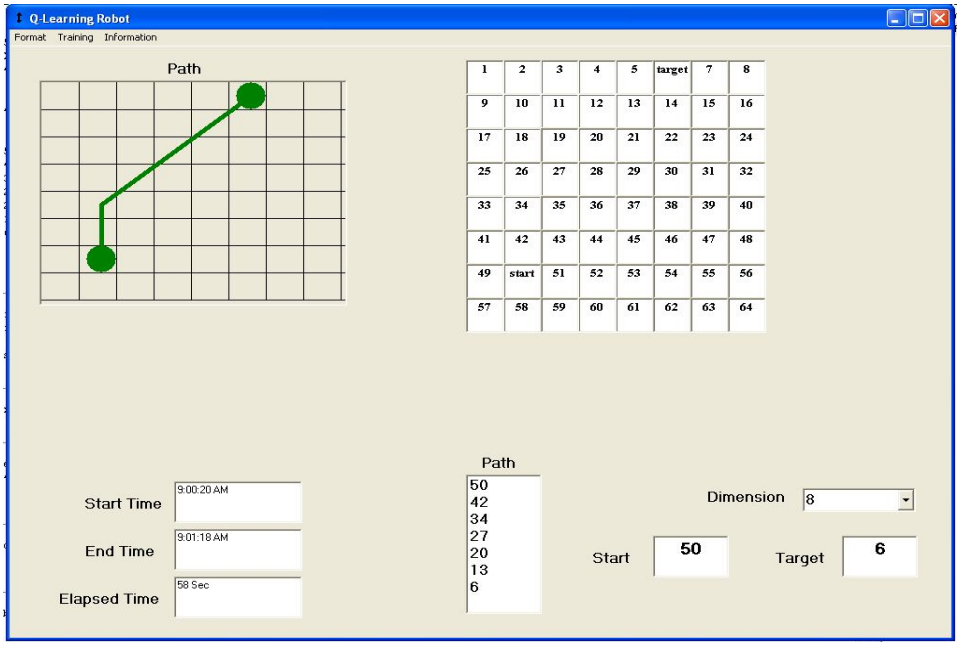

Fig.(5): Example1 maze (8×8)**.**

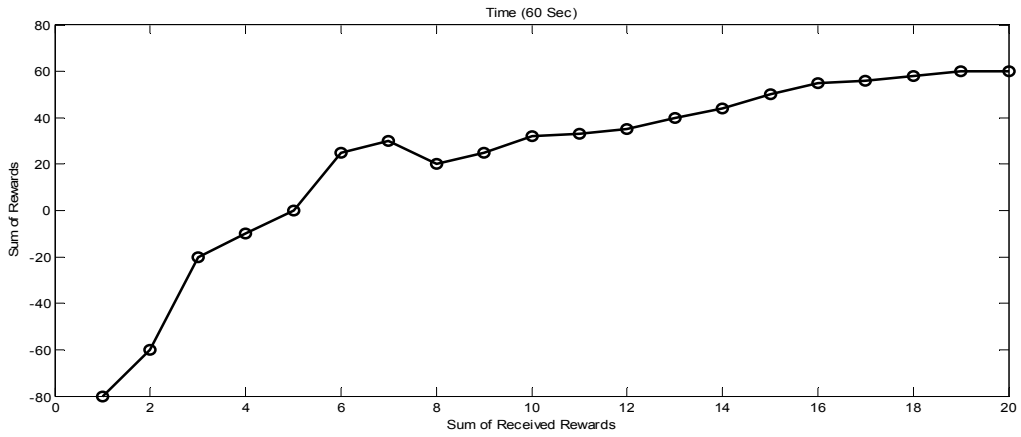

Fig.(6): Sum of Rewards for example1.

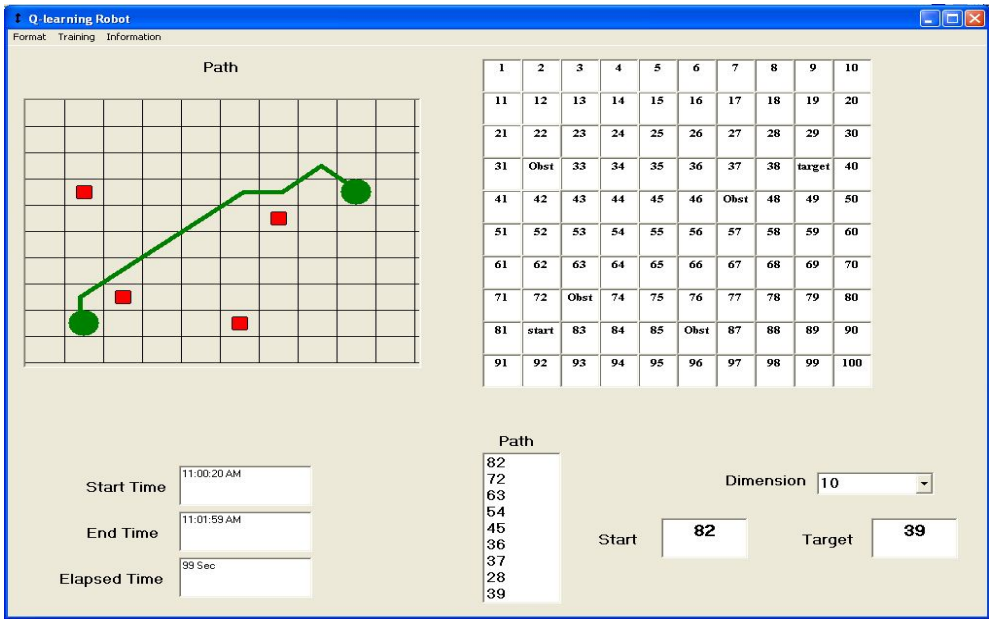

Fig.(7): Example2 maze (10×10).

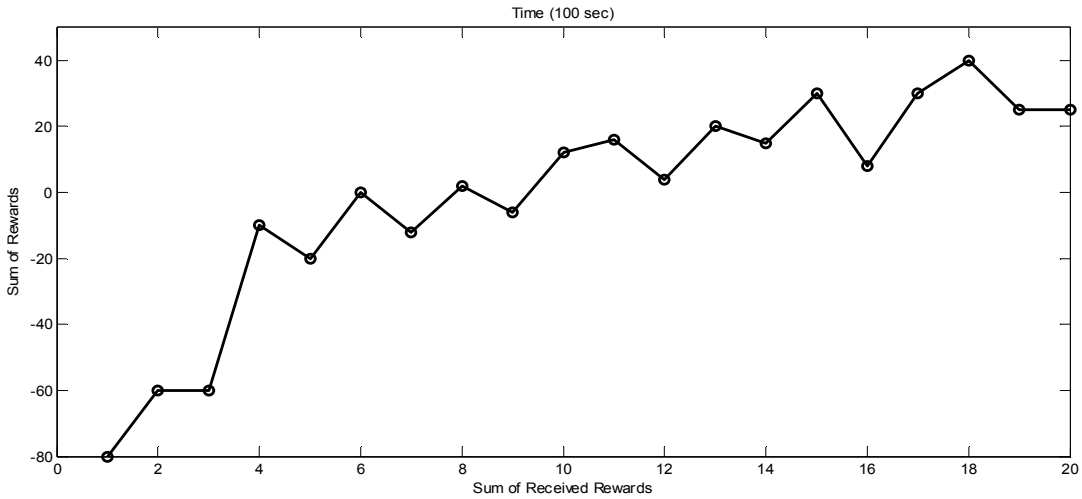

Fig.(8): Sum of Rewards for example2.

اكتشاف المسار الاقصر للربوت باستخدام طريقة Q-Learning

م.م. مهند عبد شهاب احمد

قسم الهندسة الالكترونية_ جامعة المستنصرية

الخلاصة

برمجة الروبوتات من المهام الصعبة والمفيدة ، لذلك هناك اهتمام متزايد في بناء الروبوتات ذات التعلم الذاتي. هذا المشروع يتناول طرق التعلم الرصينة التي يتعين تنفيذها لتكنولوجيا الروبوتات في الملاحة والاستكشاف. تعتبر طريقة تعلم ال Q واحدة من الطرق الواسعة الاستعمال وذات تعلم حقيقي مع الوقت وهي طريقة سهلة وكفوءة ولا تحتاج إلى عمليات معقدة او معالجة متكررة كما في الانظمة المتكيفة.الهدف من هذا العمل هو لتمكين المستخدم مثل الروبوت لاكتشاف الهدف من خلال استراتيجية الملاحة بحيث يتمكن الروبوت من سلوك اقصر المسارات بين البداية والهدف مع تجنب الاصطدام بالعوائق الثابته ، تم استخدام "طريقة الارشاد" في البحث (التعلم الذاتي) مع تعديل ظروف العمل لتلائم خوارزمية البحث ونمط حل المشكلة. الطريقة المستعملة هي تعليم الروبوت عن طريق مبدا الاستحقاق الاحسن للمسار المستكشف من عدة محاولات. اقترحت خوارزمية سهلة وذات خطوات مستقرة عدديا لعمل طريقة تعلم ال Q لتكافى سرعة وحجم ذاكرة معالج المتحكم المستخدم في تصميم الروبوت مثل المتحكم ٥٢c٨٩.