Journal of Engineering and Development, Vol. 15, No. 3, September (2011) **ISSN 1813-7822**

Real Time Mini-Robot using Improved Q-learning

Mohannad Abid Shehab Ahmed
Assistant Lecturer
Department of Electrical
College of Engineering
Al-Mustansirya University,
Baghdad- Iraq

Abstract

The task planning by a robot becomes easier when it has the requisite knowledge about its world and there is a self improving ability. In many artificial intelligent research areas like robotics navigation, path planning, and autonomous it needs to extract features precisely from environment to get the shortest path away from obstacles and even smooth this path. Choosing the path is being related to many variables like, the random of site and movement of obstacles, changing obstacle's speed, robot's size, and robot's speed variation. Scaling down robots to miniature size introduces many new challenges including memory and program size limitations, low processor performance, and low power autonomy. As a result to obvious, the simplified Q-learning tends to solve these problems as well as it learns the robot behavior on line and in real time. In this paper, numerical efficient methods (sparse reward function and directed explorer) are presented and added to the simplified type to get a self-improving on Q-Learning operations which involves the number of trial, task time and hazard, so it is natural to try to reduce the number of states, actions, and overall time. The overall analysis results in an accurate and numerically stable method for improving Q-learning.

Keyword : Reinforcement Learning, Q-Learning, Mobile Robot, 89c52 MCU.

الخلاصة

المهام المطلوبة من الروبوت تصبح سهلة عند وجود معرفة دقيقة عن بيئة العمل وهنالك امكانية تحسين ذاتي للروبوت. في اغلب ابحاث الذكاء الاصطناعي للروبوتات مثل الملاحة وتخطيط المسار والتحكم الذاتي نحتاج لاستخلاص خواص العمل للحصول على أقصر الطرق وايجاد المسارات بعيدا عن العوائق. ان اختيار المسار له علاقة بالعديد من المتغيرات مثل عشوائية المواقع , حركة العوائق ، تغيير سرعة العوائق ,حجم الروبوت ، والتباين في سرعة الروبوتات . لتقليص حجم الروبوتات هنالك عدة تحديات جديدة بما في ذلك تحديد حجم الذاكرة والبرنامج ، وخفض أداء المعالج والطاقة . ونتيجة لما سبق, طريقة تعلم ال Q المبسطة تميل لحل هذه المشاكل ، فضلا عن أنه يتم تعليم الروبوت في الوقت الحقيقي . في هذه المشروع، طرق عددية كفوءة مثل طريقة الاستحقاق في المصفوفات المتناثرة وطريقة الكشف المباشر قد استعملت لتحقيق التحسين في مبدا تعلم الـ Q من حيث عدد الحالات والإجراءات والوقت الكلي. العمل الكلي نتج عنه استعمال طرق عددية دقيقة ومستقرة لتحسين العمل .

1. Introduction

In many research areas like robotics, computer program have come to play an important role as scientific equipment, computer simulations provides a powerful devices for the virtual experiment as opposed to physical ones which are difficult to design costly or even worse dangerous as robot are used to perform increasingly difficult tasks in complex and unstructured environment, so it becomes 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[1] .

Artificial intelligence techniques offer an innovative way to overcome these limitations, in principle artificial intelligence techniques can allow a robot to successfully adapt its behavior in response to changing circumstance without the intervention of human, an autonomous robot is a device that is difficult to program because it must carry out its task in an environment at least partially unknown and unpredictable with which it interacts through sensors the actual effect of its actions might be very difficult to describe a priori due to the complex interactions of the robot's effectors with its physical environment, a good way of solving such problems would be create autonomous robot with the ability to acquire knowledge and get decision that is from direct interaction with their environments, the term "mental" refers to cognitive behavioral and other mental skills that are exhibited by humans higher animals and artificial systems [1].

It is often difficult for a programmer to translate knowledge about how to complete a task into terms that are useful or the robot. Robot sensors and actuators are very different from those of humans, and misconceptions about how they operate can cause control code to fail. The idea of providing some high-level specification of the task and using machine learning techniques to "fill in the details" is an appealing one [2].

Searches below show it can improve the Q-learning by different ways, these methods are summarized as : In work [3], the task of finding the optimal policy in Q-learning is transformed into search for an optimal solution in a combinatorial optimization problem. Then the hyper mutation mechanism from immune system is applied to the search procedure in order to keep the balance between exploration and exploitation. Through introducing the Adaptive Resonance Theory ART2 neural network in the Q-learning algorithm, Q- learning Agent in view of the duty which needs to complete to learn an appropriate incremental clustering of state-space model, so Agent can carry out decision making and a two-tiers online learning of state-space model cluster in unknown environment, without any priori knowledge, through interaction with the environment unceasingly alternately to improve the control strategies, increase the learning accuracy [4].

A Hierarchical Reinforcement Learning HRL method [5] is good method to overcome the "curse of dimensionality" problem, but when the state space is very large, using HRL method, even though the task carried out as hierarchical, the state space of subtask is also a quite large, then the convergence speed of subtask of learning the internal or external policy is quite slow. Reward function is a very important and necessary factor in RL system.

A well-designed reward function, not only can get more environmental information feedback, but also can reduce the number of useless repetition of acts of exploration, which allows you to make agent interact with the environment more quickly and deeply, and to accelerate the speed of learning.

Finally, algorithm that can extract features from the environment and get the heuristic information, which can be applied to the study by Agent in the form of reward function, which can accelerate the convergence speed significantly [6].

2. Q-Learning Theory

Probably the most widely used reinforcement learning method for robotic system is Q-Learning. This is largely due to its simplicity and the ease of transitioning from a state value function to an optimal control policy by choosing in every state the action with highest value. At every time step the robot receives the perceptual state 's', based on this information the robot chooses an action and execute it. The utility of this action is communicated to the robot through a scalar reinforcement value 'R'. The goal of the robot is to choose actions that, in the long run, maximize the sum of the reinforcement value [7]. At certain state, the best action can be selected by the use of Q-function is to have the largest Q-value at that state. After execution of the selected action, a reward is received and an update to the Q-value is needed. The update of the Q-value of an action 'a' at certain state 's' is performed by the following equation (1)

$$Q_{t+1}(S_t, a_t) = (1-\alpha).\, Q_t(S_t, a_t) + \alpha[R_t(S_t, a_t) + \gamma.\max. Q(S_{t+1}, a')] \ldots\ldots \quad (1)$$

Where:

$Q_{t+1}(S_t, a_t)$: new Q-value of an action 'a_t' at current state 's_t'.

$Q_t(S_t, a_t)$: old Q-value of an action 'a_t' at current state 's_t'.

$R_t(S_t, a_t)$: immediate reward received after perform of action 'a_t' at current state 's_t'.

$\max. Q(S_{t+1}, a')$: maximum Q- value of next state 's' as the future reward.

α: learning rate, positive constant smaller than 1.

γ: discount rate, positive constant smaller than 1. It used to weight immediate reward more heavily than distant future reward [8].

3. The Proposed Q-Learning Improvement Approaches

Reinforcement learning and Q-learning in particular, seems to be a natural choice of learning control policies on robots. Instead of designing a low-level control policy, it can design a much higher-level task description in the form of the reward function, R(s, a). Designing a sparse reward function is generally easier than designing the low-level mapping from observations to actions. Often, for robot tasks, rewards correspond to physical events in the world. It is easy to come up with simple reward functions for many tasks. For example, for an obstacle avoidance task, the robot might get a reward of 1 for reaching the goal, -1 for hitting (punishment) an obstacle, and 0 for don't care. In theory, this is all that is necessary for the robot to learn the optimal policy [2].

However, there are a number of problems with simply using standard Q-learning techniques, it can avoid most of these problems through developing more heuristic reward function, it can call this type by ***sparse reward function***. Sparse reward functions are zero everywhere, except for a few places (the obstacles and goal in the above example). The contrast with dense reward functions, which give non-zero rewards most of the time. A dense reward function for obstacle avoidance might be, for example, the sum of distances to the obstacles divided by the distance to the goal state. Dense reward functions give more information after each action, but are much more difficult to construct than sparse functions. The sparse reward transforms the original reward function to make it more informative during learning by adding small value (0-1) to original reward for the operation compatible with optimum policy function.

In this work, the state space is mostly be interested in sparse reward functions, since it is generally much simpler to perform and design, it is, therefore, forced to choose less arbitrary actions. As more information, in the form of rewards R(s, a) arrives Q-learning can iteratively improve its approximation of the value function. However, if no rewards are observed, the value function approximation will never change. This problem is compounded by sparse reward functions. If there are only a few rewards, and the state-action space is large, the chances of finding a reward by chance are very small indeed.

Exploration is an inherent aspect of any Reinforcement Learning RL method that does not assume an a priori model. Undirected exploration, which explores evenly around the currently policy, is often less efficient than directed exploration, which attempts to direct exploration towards "interesting" parts of the state-action space. It uses a combination/variation of two methods [9] which results in a directed exploration technique that is particularly effective for robot RL. It requires the storage of some extra information for each state-action pair, an exploration bonus is added to the estimated Q-values, which reflects the added value of selecting a particular state-action value for the sake of exploration, and action selection is done based on these Q-values plus exploration bonuses:

$$Q^{+}(s,a) = Q(s,a) + \mu\frac{\sqrt{m(s,a)}}{n(s,a)} \qquad \ldots\ldots (2)$$

Where μ is a constant, $m(s,a)$ is the number of time steps since action was last tried in state s, and $n(s,a)$ is the number of times action a was tried in state s at all. If n(s, a) = 0 it is replaced by a constant $\nu \in [0, 1)$. $Q^{+}(s,a)$ is used for standard Boltzmann exploration which assigns probabilities of action selection p(s, a) in proportion to values:

$$p(s,a) = \frac{e^{Q^{+}(s,a)/T}}{\sum_{\hat{a}} e^{Q^{+}(s,\hat{a})/T}} \qquad \ldots\ldots (3)$$

Where: T is the so-called temperature parameter constant

The net result is exploration that favors actions which are promising with respect to rewards (the effect of Boltzmann exploration), that favors actions that have not been tried often (the effect of n(s, a)). Once n(s, a) becomes very large, the exploration bonus vanishes [9]. The needing of high storage density can be overcome utilizing the real robot system connected to computer system or high storage PIC microcontroller.

4. Work Implementation:

The following procedure is to perform reinforcement learning in Robocar called mini-robot:

1. The robot obtains information from the environment. This information is converted to discrete values. The combination of these values can determine a state number.
2. Convert the above step to sparse matrix through estimate the non-effective states and zero their activity, here the reward will be customize to 1 (reward) and -1 (punishment)
3. To select an action, rather than get an action with highest Q-value. The equations (2) and (3) are used, these equations give the probability of picking an action 'a' under state 's'.

   Exploration for new policy continues with the use of equation (1)
4. Execute the selected action.
5. Obtain the reward from the change of the robot energy.
6. Determine the new state from the information of the environment.
7. Update the Q-value of the executed action at pervious state. With the use of reward and the maximum Q-value of the new state.

It can summarize our approach as compared to [3,4,5,6] in two main points as:

- Before selecting an action or give any reward the environment must be handled as a sparse matrix leads to eliminate many undesired paths and not need to be accessed by agent, then combining the reward functions of the robot with previous environment matrix to get the sparse reward function; in other words it means mixing "Artificial Intelligence" of Q-learning with "Algebraic Manipulation" of sparse.
- Apply the Direct Exploration method which attempts to increase the probabilities of action selection $p(s, a)$ that can reduce the required time.

With the use of these procedures, a Robocar "mini-robot" programming is implemented. The above steps can be summarized in the following flow chart shown in figure (1) below:

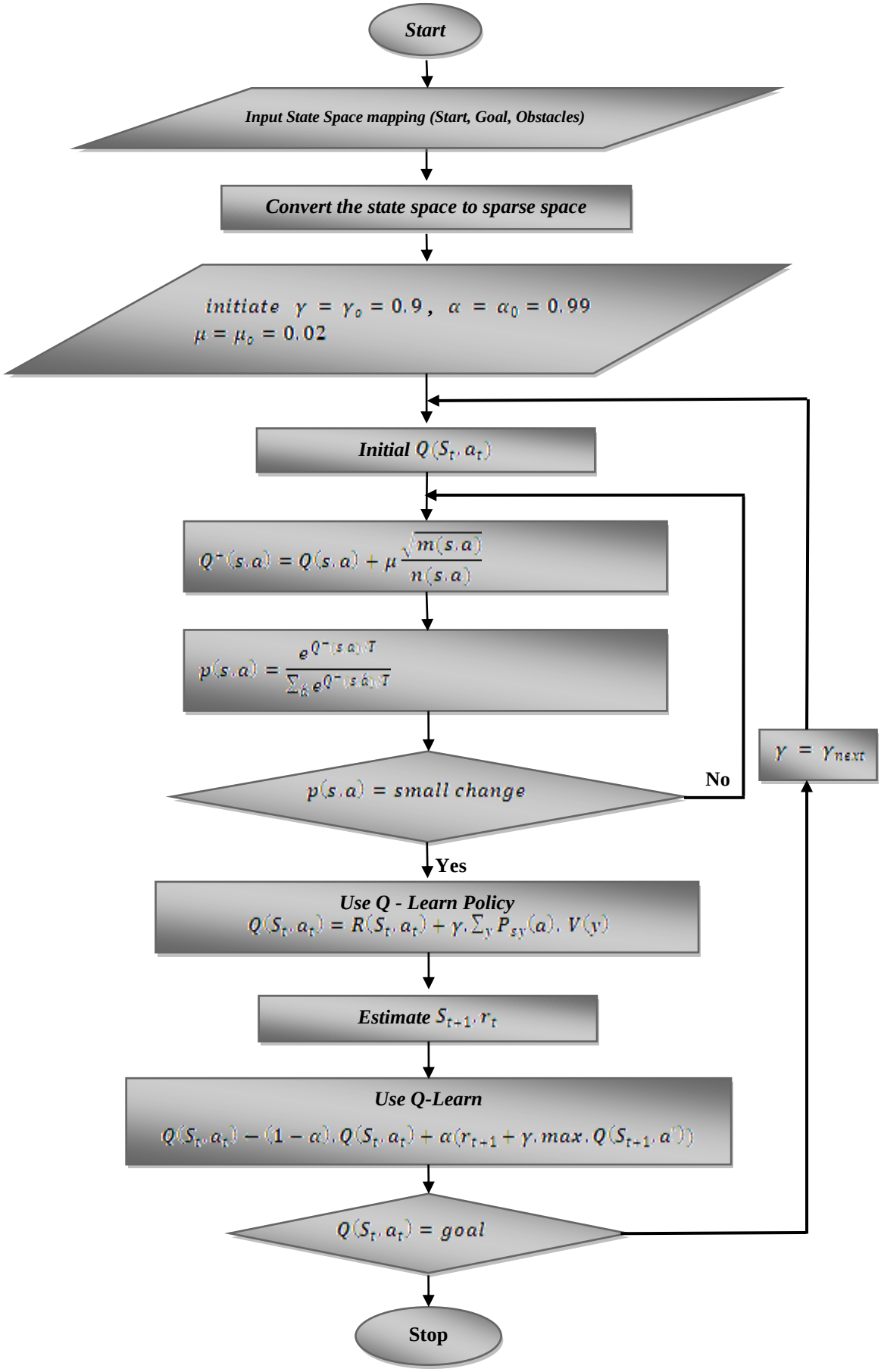


Figure (1): Improved Q-Learning Flow Diagram

5. Q – Learning Software

The Mini Robot program is build using Visual Basic language and it state space is a two - dimensional array with cells as starting point, target point, 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 target efficiently.
First, let a simple maze with 8×8 cells with no obstacle as shown in figure (2):

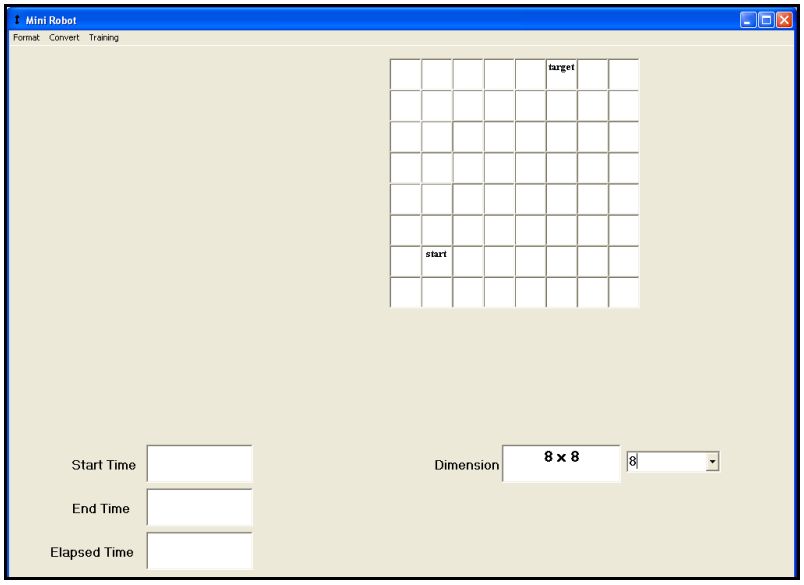


Figure (2): example 1 with maze (8×8)

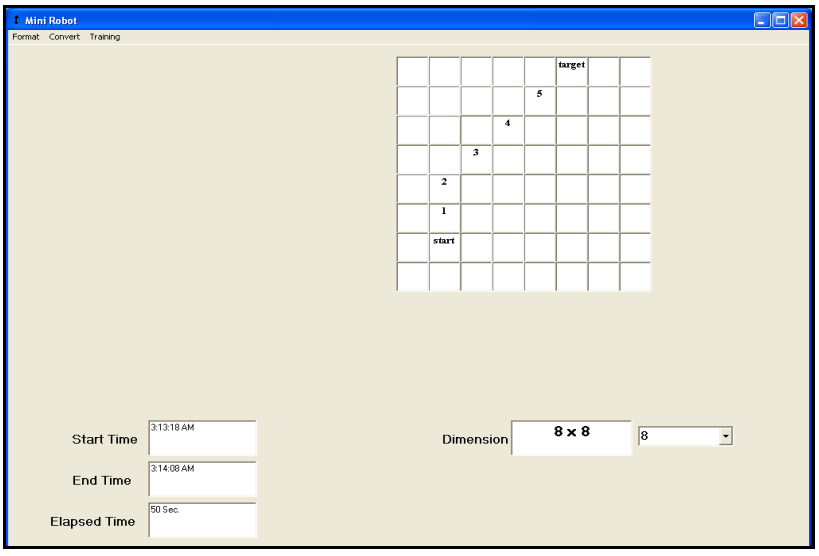


Figure (3): execution of example 1

The Mini Robot needs to 50 seconds to complete its search successfully for finding the best shortest path and its curve is of figure (4) which shows changes of the received rewards during the 50 second time.

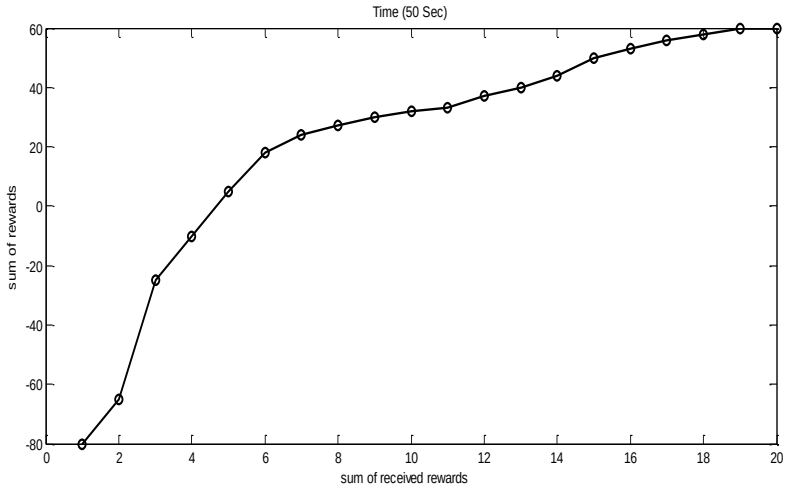


Figure (4): Sum of Rewards of example1

Secondly, let the maze with 10×10 cells with four obstacles as shown in figure (5) :

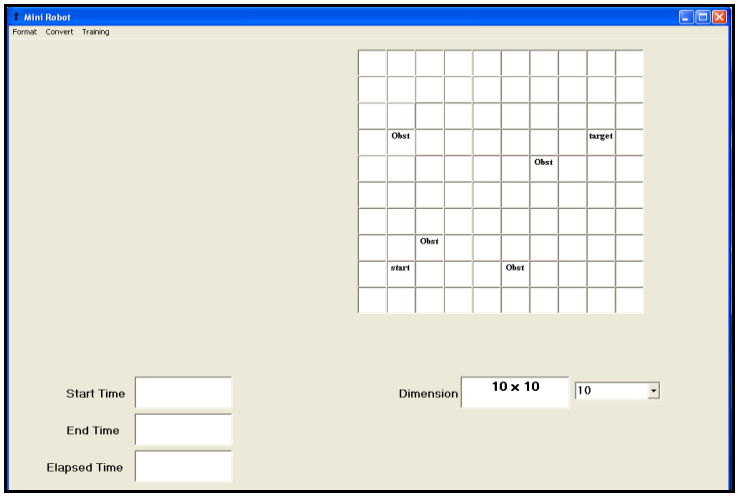


Figure (5): example 2 with maze (10×10)

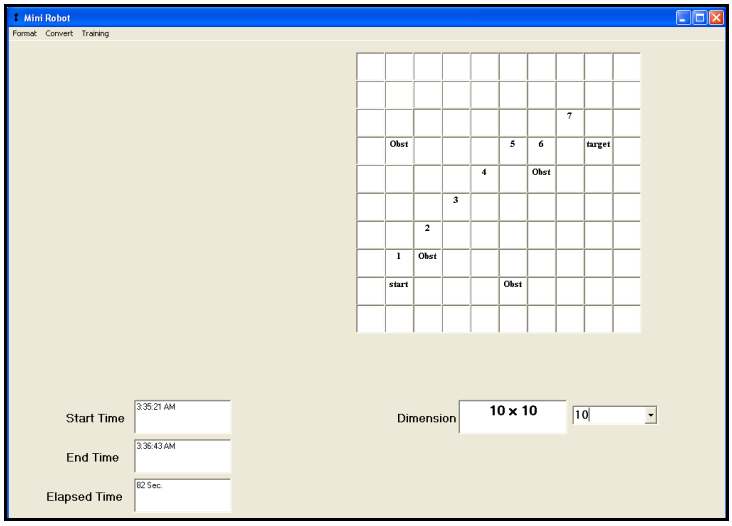


Figure (6): execution of example 2

The Mini Robot needs to 82 seconds to complete its search successfully for finding the best shortest path and its operation curve is in figure (7) which shows changes of the received rewards during the 82 second time.

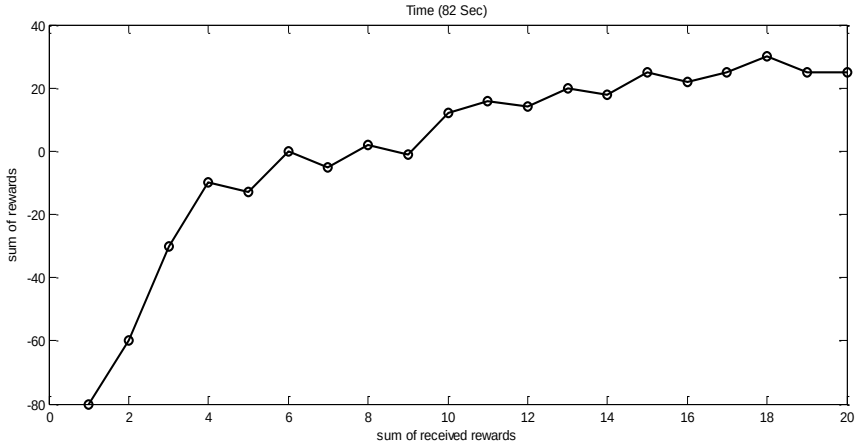


Figure (7): Sum of Rewards of example2

From the obvious figures 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 negative rewards are generated and the still growth exponentially due to standard Boltzman Exploration Function which assign probabilities of action selection p(s,a) as in equation (3), 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 generated, at the end the number of trails will be reduced and the system may converge 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, utilizing the improved Q-Learning leads to minimize these variations and the negative rewards especially at the medium of operation not at the beginning. Examples (1)&(2) were studied using simple Q-Learning [2][10], they result in same path result but with larger time than the improved type, as a result the improved method can improve the amount of time needed as the comparison shown in table (1)

Table (1)Time comparison for simplified and improved methods

Q-Learning Method	Example1 time	Example2 time
Simplified method	58 Sec	99 Sec
Improved method	50 Sec	82 Sec

The comparison of the simplified Q-learning [2][10] researches and the present improved type shows that the present work needs to less number of actions and iterations for small time as shown in the figure below

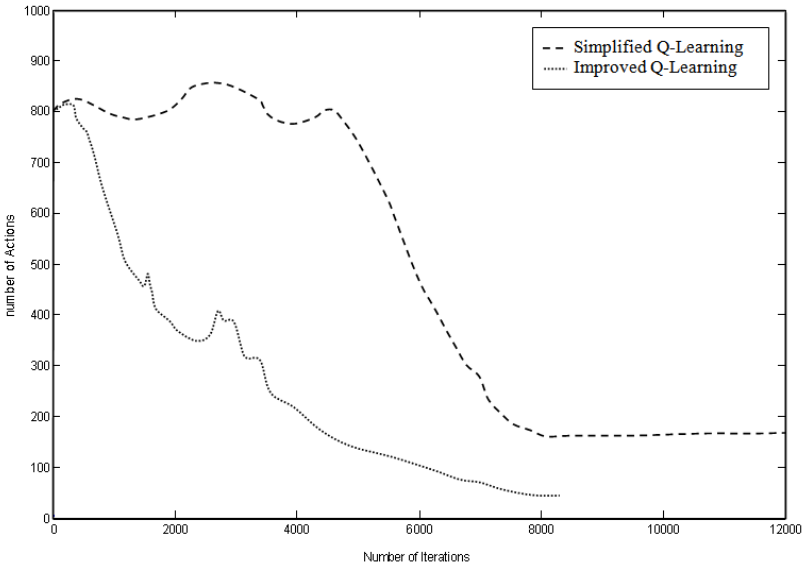


Figure (8): Number of actions .vs. iterations comparison for simplified and improved methods

6. Mini Robot and Microcontroller Hardware:

In this work real robocar "Mini-Robot" hardware is designed to study and implement the robot learning. Mini-Robot is equipped with 89c52 MCU microcontroller and its drive circuit, two stepper motors for locomotion; it has high energy battery to make it suitable for long time learning. The specification of MCU 89c52 is detailed in table (2) : The software core is composed of a simple real time operating system, which handles different time-critical tasks as:

1. Communication with controller to receive remote commands. Currently there are four:
   - Start learning.
   - Stop learning and behave based on the learned policy (for evaluation).
   - Save the learned policy to EEPROM so that it can be downloaded after learning.
   - Load the saved policy from EEPROM to continue the learning task in case the task is too long to be completed in one battery life cycle.
2. Proximity reading and state detection.
3. Action selection.
4. Control of right and left motors to do the selected action correctly.
5. Updating the policy and learning.
6. Computing statistics and quality measures and writing them to EEPROM for later evaluation purposes.

Table (2): 89c52 Microcontroller specification

1-	80C52 based architecture 40-pin DIP.
2-	8-Kbytes of on-chip Reprogrammable Flash Memory 256 x 8 RAM.
3-	Three 16-bit Timer/Counters.
4-	Full duplex serial channel.
5-	Boolean processor.
6-	Four 8-bit I/O ports, 32 I/O lines.
7-	Memory addressing capability – 64K ROM and 64K RAM.
8-	Program memory lock – Lock bits (3).
9-	Power save modes – Idle and power-down.
10-	Eight interrupt sources.
11-	Most instructions execute in 0.3 µs.
12-	CMOS and TTL compatible.
13-	Maximum speed: 40 MHz @ Vcc = 5V.
14-	Industrial temperature available.

Figures (9) & (10) below show a simple diagram of real time operation and schematic diagram :

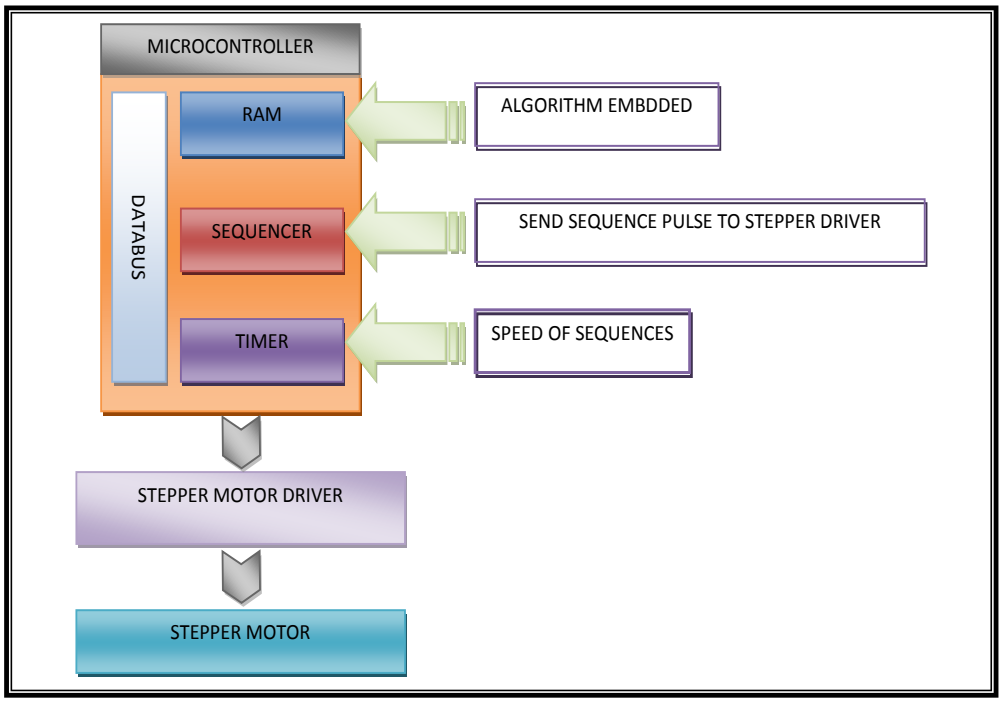


Figure (9): Real Time Simple Diagram

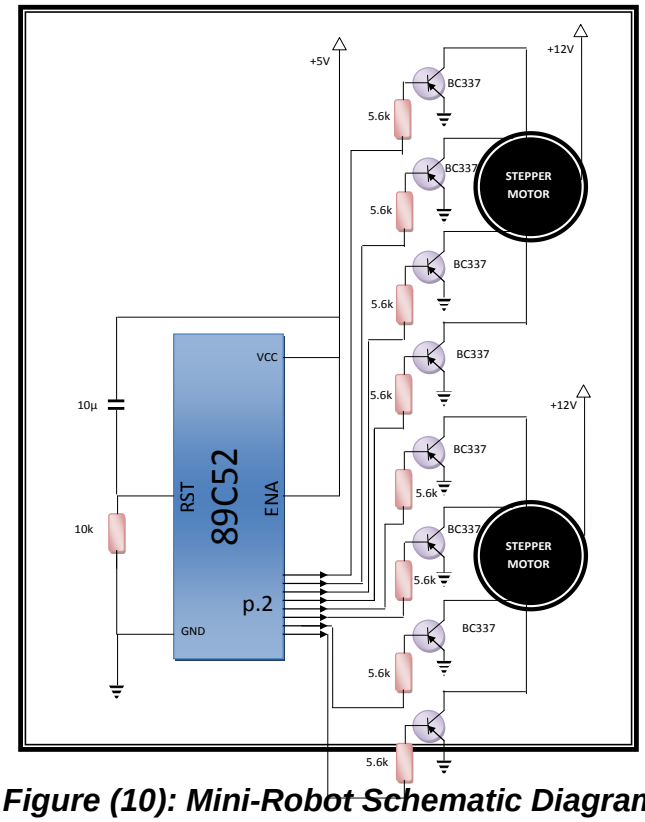


Figure (10): Mini-Robot Schematic Diagram

The designed Mini-Robot is as shown in figure (11):

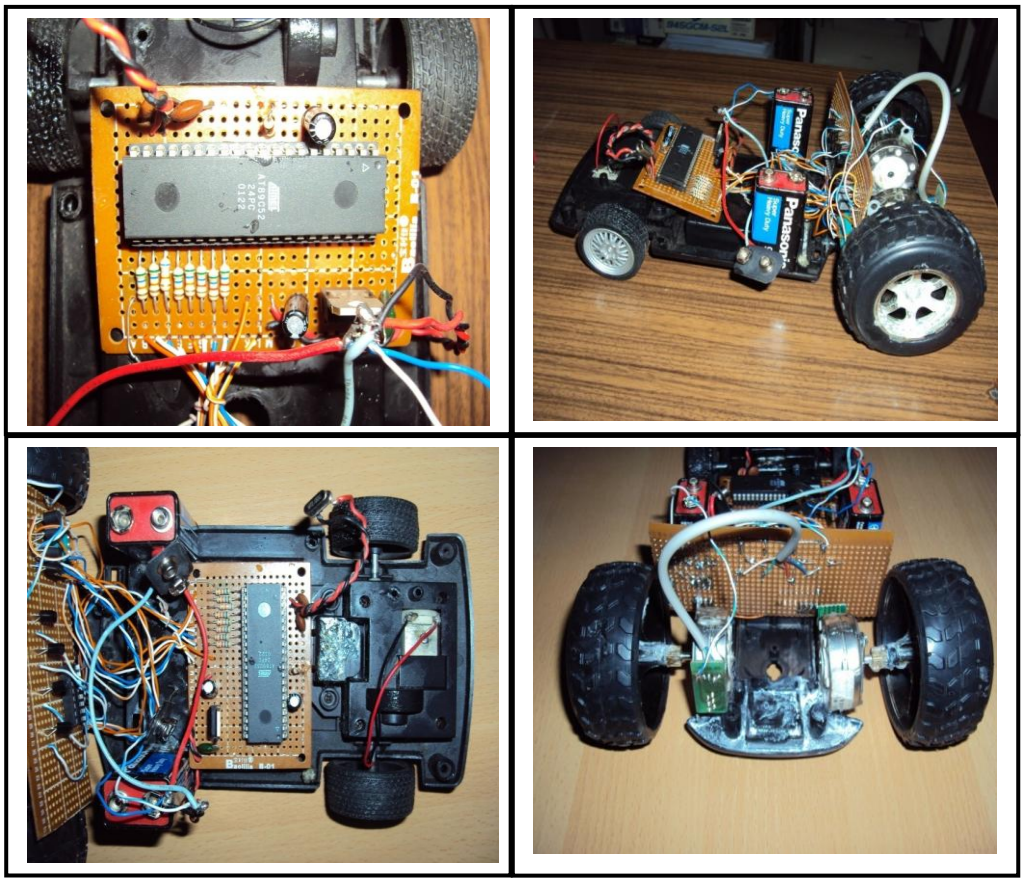


Figure (11): Mini-Robot Robocar

Conclusion and future works

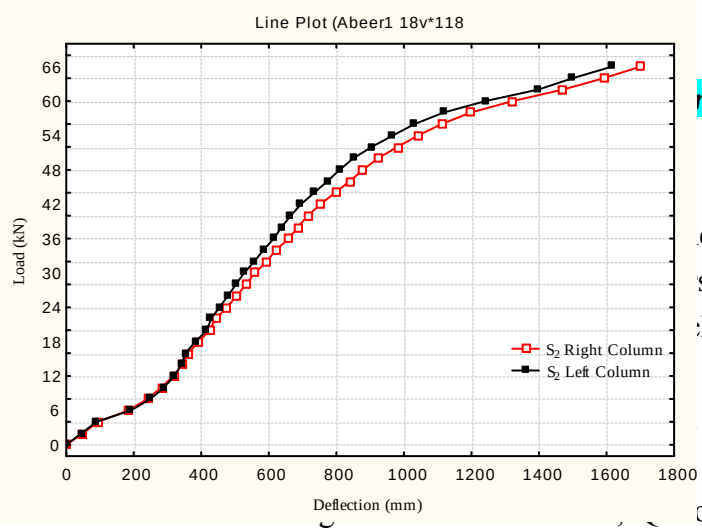
The previous works [2][10] show that Q-learning is a valid and good approach for robot behavioral in both simulation and real robot, so we proposed a simple and fast improving on reinforcement learning algorithm optimized with robot size, data, and program memory consumption.

The proposed representation generates a small and discrete state space. It results in a fast and stable learning of the given task and out performs metrical approaches that rely on function approximation in learning success, speed, stability, and number of collisions.

The reinforcement learning system passively observes the states, actions and rewards encountered by the robot, and uses this information to bootstrap its value- function approximation, it also demonstrated that reinforcement learning and teaching together is a promising approach to autonomous learning of robot behaviors. Since a Q-function indicates some kind of information about the applicability of a behavior (i.e., the larger the Q-value, the closer the goal), the integration of behaviors can benefit from the use of learned Q-functions.

Finally, the framework is capable of learning good control policies more quickly than moderately experienced programs. Future work will show that the acquired strategy can be used as background knowledge for new learning tasks in unknown environments and therefore allows for speeding up learning. It can also investigate how to learn two separate policies for goal-oriented and generally sensible behavior in a hierarchical learning architecture. Strategies learned in simulation will also be ported to a real robot platform.

With restricted state space, the robot cannot have an accurate and complete representation for its states so it may affect the learning of the correct policies. Use of neural network or fuzzy logic with reinforcement learning can approximate the Q-function and compress the size of table to small number of weights. It can also calculate output from any input. With large complex environment, Hierarchical Reinforcement learning algorithms can support our goals through divide the task into subtasks to get more feature extractions.



cs a practical introduction", 2nd Edition, Department sity of Essex, 2003.

k Kaelbling, "Effective Reinforcement Learning for

and Maher Sid-Ahmed, "An Improved Immune Q-

ong Gu, Liping Tang and Xinyu Qu, "Study on Q-learning Algorithm Based on ART2", 2010.

5. Qicui Yan, Quan Liu, Daojing Hu, "A Hierarchical Reinforcement Learning Algorithm Based On Heuristic Reward Function", 2010.
6. Jianhong Zhang, Ying Shi, and Xiaofei Xie, "The Q(λ) Algorithm Based on Heuristic Reward Function", 2010.
7. Eric Martinson, Alexander Stoytchev, and Ronald Arkin, "Robot Behavioral Selection Using Q-Learning", Mobile Robot Laboratory, College of Computing, Georgia Institutes of Technology, 2001.
8. Tsang Hin Hang, "Reinforcement Learning in Robocode", 2005.
9. Chris Gasket, "Q-Learning For Robot Control", Thesis Submitted for Degree of PhD., Department of Computer Systems Engineering, The Australian National University, 2002.
10. M. Carreras, P. Ridoa and A. El-Fakdi "Semi-Online Neural Q-Learning for Real-time Robot Learning", Las Vegas, USA, 2003.