Knowledge based Reinforcement Learning Robot in Maze Environment

Dr. D. Venkata Vara Prasad ME.Ph.D,
Professor,
Department Of Cse,
Ssn College Of Engineering.,
Kalavakkam-603 110.

Chitra Devi. J,
Be (Cse),
Ssnadar College Of Engg.,
Kalavakkam- 603 110.

Karpagam. P,
Be(Cse),
Ssnadar College Of Engg.,
Kalavakkam- 603 110.

Manju Priyadharsini. D
Be(Cse),
Ssnadar College Of Engg.,
Kalavakkam- 603 110.

ABSTRACT
A simple approach for knowledge based maze solving is presented for a mobile robot. The artificial intelligence concept like reinforcement learning technique is utilized by the robot to learn the new environment. The robot travels through the environment and identifies the target by following a set of rules. After reaching the target, the robot returns back through the optimum path by avoiding dead ends. For achieving this, the robot uses a line maze solving algorithm which uses a set of replacement rules to replace the wrong paths travelled with the correct ones. The algorithm for this maze solver is qualitative in nature, requiring no map of environment, no image Jacobian, no Homography, no fundamental matrix, and no assumption. The environment is accessible, deterministic and static. The working procedure of this project consists of line path following, mobile robot navigation, knowledge based navigation, reinforcement learning.

General Terms
Maze solving algorithm
1. Teach Phase
2. Replay Phase

Rules for maze solver
1. LSR Rule
2. Replacement Rule

1. INTRODUCTION
Robotics deals with computer-controlled machine that is programmed to move, manipulates objects and accomplish work while interacting with its environment. Robots are able to perform repetitive tasks more quickly, cheaply, and accurately than humans. The term robot originates from the Czech word robota, meaning "compulsory labor." It was first used in the 1921 play R.U.R. (Rossum's Universal Robots) by the Czech novelist and playwright Karel Capek. The word robot has been used since to refer to a machine that performs work to assist people or work that humans find difficult or undesirable.

The first industrial modern robots were the Unimates developed by George Devol and Joe Engelberger in the late 50's and early 60's. The first patents were by Devol for parts transfer machines. Engelberger formed Unimation and was the first to market robots. As a result, Engelberger has been called the 'father of robotics.' Modern industrial arms have increased in capability and performance through controller and language development, improved mechanisms, sensing, and drive systems.

The modern era of robotics begins around 1959, when John McCarthy and Marvin Minsky established the Artificial Intelligence lab at MIT. A couple years later, Heinrich Erst created the first modern robotic hand, and in 1962, Unimate, the first industrial robot, was created to perform repetitive or dangerous tasks on the assembly line of General Motors.

2. PROBLEM STATEMENT
In Military department, during terrorism attack, there is a need to identify the bombs placed by harmful attackers. A person is employed to carry out these action, which is highly a risky profession placing the challenge for livelihood. In order to overcome this risky job and to save the life of the Army, a maze solving robot has been designed and developed. This maze solver is useful in the environments where there is a threat to human life. The robot travels through the environment and identifies the target and returns back through the optimum path by avoiding dead ends.

The camera for this maze solving robot is reflection sensors. This sensors use LED as transmitters and photodiode as reflectors. The reflective nature of black and white colour is utilized in the robot. Black surface absorbs the light where as white is reflective in nature, thus emitting the light it received.

Identifying the target location and learning the path through reinforcement method are the actions performed by the robot. The goal of the robot is to detect the bomb spot and dismantle it. The environment is accessible, deterministic and static.

3. SCENARIO
The environment is in the form of a maze with black lines representing the path and having different kinds of junctions.

There is a starting point and target differentiated from other paths.
The robot will have IR sensors in the front to track the path. The sensor values are sent to the microcontroller which processes them and identifies the possible turns needed.
The controller then rotates the wheels by supplying appropriate values to the motors which drive the wheels. In case of decision making junctions, the microcontroller stores the junction in its memory for finding the optimum path later. Maze solving algorithm is used for this purpose.

The Figure 1 gives the blueprint of the sample maze.

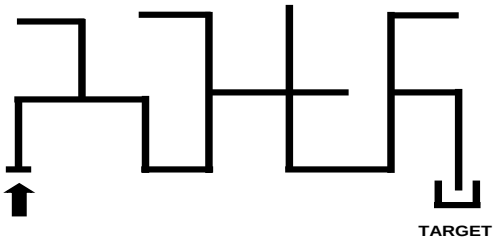

Figure 1 Blueprint of the maze

4. LITERATURE REVIEW

4.1 Overview

Mobile robots in human inhabited environments should operate safely and reliably. At the same time they are expected to minimize energy consumption, travel time and distance. Optimization of these parameters implies that the robot must be able to predict its behavior in a partially unknown and changing environment.

4.2 Line Detector

The line detector is an Infrared reflective sensor that can be attached to the front of the car. The sensor delivers three digital signals to the microcontroller to enable the car to follow a white line on a black background, or vice versa.

[3] examined path planning strategies in large partially unknown environments. The robot learned innovative routes to find reliable trajectories and optimize robot's behavior. The approach was verified to shortest path following in 6 different environments.
The analysis and optimal design of a millimeters scale omni-directional mobile microrobot has been presented in [5]. The microrobot is supported by two sets of special dualwheel structure, and is actuated by three electromagnetic micromotors with 2 mm diameter.

Based upon a novel concept called the funnel lane [13], the coodinates of feature points during the replay phase are compared with those obtained during the teaching phase in order to determine the turning direction. Increased robustness is achieved by coupling the feature coordinates with odometry information. The system requires a single off-the-shelf, forward-looking camera with no calibration (either external or internal, including lens distortion). Implicit calibration of the system is needed only in the form of a single controller gain.

4.3 Motion Planning Algorithms

The first step in mobile robot navigation is to create or to use a map and to localize itself in it. An autonomous agent has to have the following abilities: map learning or map creating, localization and path planning [12]. Swarm intelligence theory is proposed for motion planning of multi-robot systems. Multiple particles start from different points in the solutions space and interact to each other while moving towards the goal position. Swarm intelligence theory is a derivative-free approach to the problem of multi-robot cooperation which works by searching iteratively in regions defined by each robot's best previous move and the best previous move of its neighbors [2].

Robots use path planning algorithms to plan a path from start to goal. In dynamic environments the environment can change during path following. To avoid collisions with unknown obstacles robots use local re-planning. While classic Artificial Intelligence(AI) planners are used to produce the global path to the goal, local re-planners usually act reactively [3].

The roadmap approach to robot path planning [7] is one of the earliest methods. Since then, many different algorithms for building roadmaps have been proposed and widely implemented in mobile robots but their use has always been limited to planning in static, totally known environments.

4.4 Reinforcement Learning

Reinforcement learning [11] from delayed rewards has been applied to mobile robot control in various domains. The approach has been especially successful in applications where it is possible to learn policies in simulation and then transfer the learned controller to the real robot. However, applications involving learning on real robots are still relatively rare. In principle a mobile robot could learn any task from scratch by reinforcement learning, but learning of complex tasks can be very time consuming, so the researcher must find a way to speed up the learning.

Techniques for accelerating reinforcement learning on real robots include
(1) guiding exploration by human demonstration, advice or an approximate pre- installed controller
(2) using replayed experiences or models to generate "simulated" experiences
(3) applying function approximators for better generalization.

Function approximators also provide a means for dealing with continuous state and action spaces.

4.4.1 Q- Learning

Q-learning is a form of model-free reinforcement learning method based on stochastic dynamic programming. It provides robots with the capability of learning to act optimally in a Markovian environment.

In [6], they have proposed an approach which combined Fuzzy based CMAC(Cerebellar Model Arithmetic Computer) to calculate the weighting factors of the agents' state with Multi-agent Q-learning to generate a continuous action output command and have performed it in real robot applications.

4.4.2 Lazy Q Learning

Compared to single robot learning, cooperative learning adds the challenge of a much larger search space (combined individual search spaces), awareness of other team members, and also the synthesis of the individual behaviors with respect to the task given to the group. Over the years, reinforcement learning has emerged as the main learning approach in autonomous robotics, and lazy learning has become the leading bias, allowing the reduction of the time required by an experiment to the time needed to test the learned behavior performance. These two approaches have been combined together in what is now called lazy Q-learning [1], a very efficient single robot learning paradigm.

4.5 Different Environments

Mobile robots in human inhabited environments should operate safely and reliably. At the same time they are expected to minimize energy consumption, travel time and distance. Optimization of these parameters implies that the robot must be able to predict its behavior in a partially unknown and changing environment [3].

Mining accidents have occurred since the early days of mining. There were a total of 525 mining disasters (incidents with five or more fatalities) in both coal and metal/non metal mines from 1900 through 2007 in the United States, resulting in 12,823 fatalities. Most of these disasters involve mine rescue teams, which are specially trained to perform search and rescue operations in extremely hostile environments.

Robots have a great potential to assist in these underground operations, searching ahead of rescue teams and reporting conditions that may be hazardous to the teams. When explosive conditions exist or when heavy smoke or unstable ground conditions prevent team members from entering a mine, robots can become an invaluable tool [9].

Underground mine rescue presents a novel application of robotics. It is different from urban search and rescue in that it has a stronger focus on property recovery and requires manipulation, but it shares the same problems in the use of tethers and belays for operating on slopes and down voids. Mine rescue robot needs are most similar to law enforcement and bomb squad robots; however, mine rescue appears to require less manipulation overall. The most common type of subterranean robots, pipe crawlers, are too specialized for the diverse mine missions.

4.6 Obstacle Detection

Most collision avoidance methods do not consider the vehicle shape and its kinematic and dynamic constraints, assuming the robot to be point-like and omnidirectional with no acceleration constraints. The contribution of [4] is a methodology to consider the exact shape and kinematics, as well as the effects of dynamics in the collision avoidance layer, since the original avoidance method does not address them. This is achievable by abstracting the constraints from the avoidance methods in such a way that when the method is applied, the constraints already have been considered.

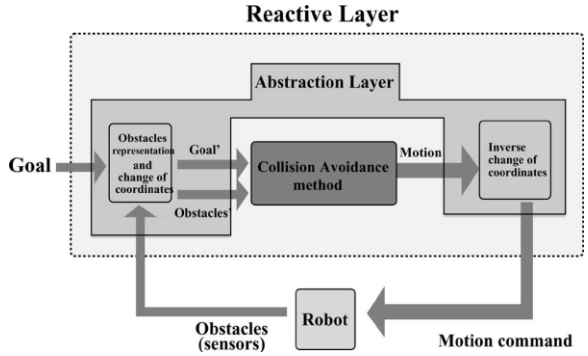

Figure 2 Collision Avoidance Method

The Figure 2 shows the collision avoidance method proposed in [4]. Abstraction layer abstracts shape, kinematics, and dynamics of the vehicle from the avoidance method. The idea is to understand the method as a "black-box" and to modify the representation of its inputs, so that they have implicit information about these restrictions.

4.7 Fuzzy Logic Controllers

Fuzzy Logic Controllers (FLC) are used to implement an efficient and accurate positioning of an autonomous car-like mobile robot, respecting final orientation. To accomplish this task, called "Oriented Positioning", two FLC have been developed: Robot Positioning Controller (RPC) and Robot Following Controller (RFC).

[10] presents a new technique used to perform the task of Oriented Positioning on a non-holonomic mobile robot, using two fuzzy logic controllers. The first controller used for Simple Positioning has been implemented and tested on simulation. Then, it is combined with a second controller, used for "Virtual Following", in order to accomplish the desired task

4.7.1 Combined Intelligent Control

The Combined Intelligent Control(CIC) [8] combines different methods and architectures in intelligence, distributed intelligence and control areas to reach a high level of intelligence by emphasizing their strengths and eliminating the weaknesses of each architecture. Natural environments are dynamic, nondeterministic and complex and if agents want to be efficient, they must have a high level of intelligence same as any natural creature. A robot must have the special abilities to do different goals and tasks and the CIC tries to provide these abilities. The CIC is a distributed structure. Thus, it can provide some abilities such as unlimited development, high fault tolerance, cooperation, ability to do various tasks, needless of accurate modelling of environment and other agents, making global decisions by local information.

5. DESIGN METHODOLOGY

Design methodology consists of designing six basic circuits. Signals are passed on from one circuit to another and necessary conversions are carried out by each circuit and thus enabling proper line flow. In sensor circuit, four IR emitter receiver pairs were used which distinguishes black line from white surface based on their reflective nature. The information from sensor circuit is passed on to comparator, which is mainly used for converting the analog input to digital output.

The digitalized output is then passed on to transistor circuit, which is needed for maintaining constant voltage through out the circuit. The signal is then passed on to programmed microcontroller circuit which then takes decision based on received input. It processes the given input and sends the corresponding output to motor driver circuit. Motor driver circuit is responsible for driving two independent motors, which is connected to each of its corresponding side. Each motor is operated independently by the motor driver, which is decided based on each motors input. Motor driver rotates the motor both in forward and reverse directions. By supplying same voltage to both of its terminals which is associated with motor driver, the motor can be stopped at dead ends.

The basic flow diagram is shown below in Figure 3.

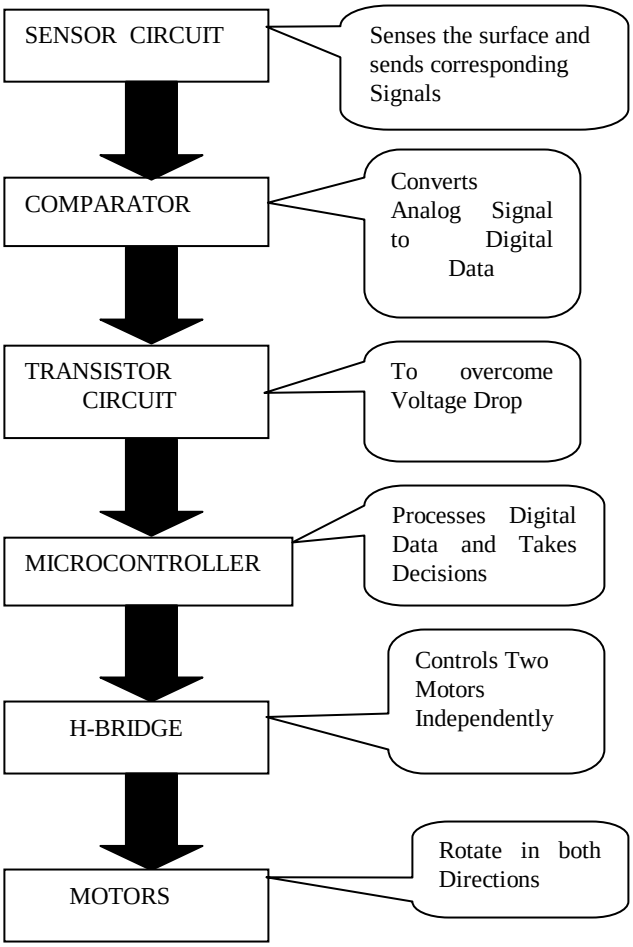

Figure 3 Basic Flow Diagram

5.1 Sensor Circuit Design

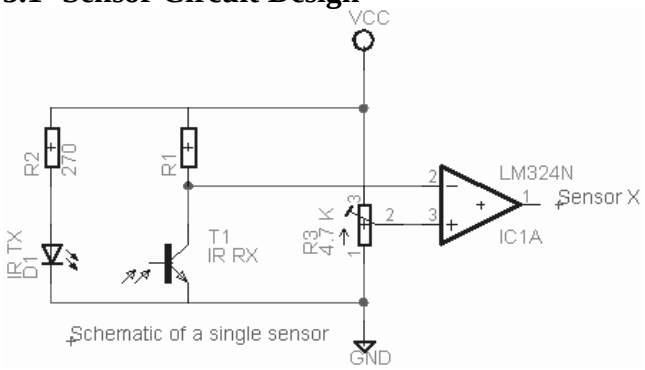

Figure 4 Single Sensor Circuit

A single sensor circuit is shown in Figure 4. It basically comprises of IR emitting diode, IR receiving transistor, resistors, potentiometer and an op-amp.

5.2 Analog to Digital Convertor

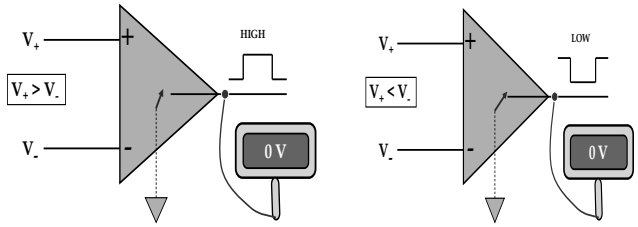

Figure 5 Operation of ADC

The operation of ADC is shown in Figure 5**.** Each op-amp consists of an inverting and a non-inverting input. Non-inverting input gets the signal from potentiometer, which is varied accordingly. To the inverting input, sensors output are given. When there is high signal at the non-inverting side output perceived is logic 1, similarly when there is low signal at the non-inverting side; logic 0 is received as output. Thus it digitizes the analog signal from sensors.

5.3 Transistor Based Circuit Design

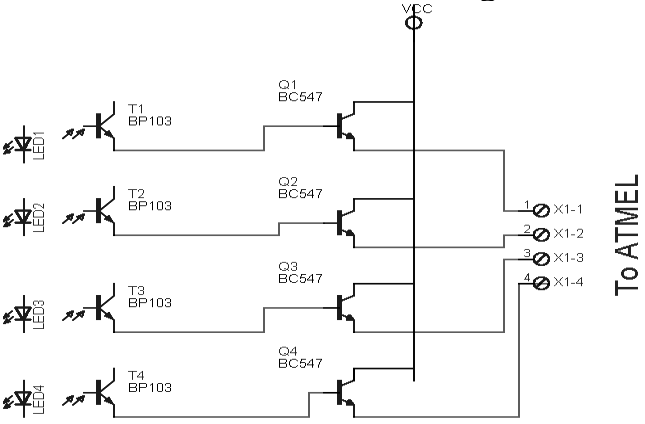

Figure 6 Transistor circuit

Figure 6 shows the circuit design based on transistors. Sensors are nothing but the combination of transmitters and receivers. Transmitters are usually LED's which emits constant 5V to the environment whereas Receivers are usually Transistors which absorbs the reflected voltage from the environment usually in the range 1-5V. Sometimes there might be the case where the transistors transmit the voltage of range 0-2V which is not accepted as input by the Atmel microcontroller. In order to raise the input up to 4V say, transistor circuit is used.

5.4 Microcontroller Based Circuit Design

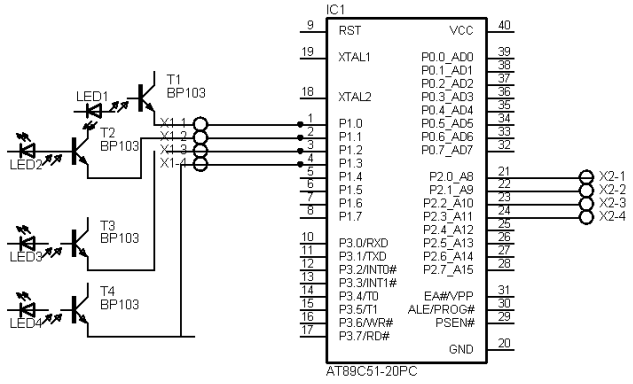

Figure 7 Microcontroller circuit

The Figure 7 shows the circuit design based on microcontroller AT89C51.LED and Transistor T are combined and termed as Sensors. The sensor circuit consists of 4 sensors say S1, S2, S3, S4. Inputs are taken from sensors and if required, the current flow can be modified using transistor circuit. Then the input is given to Port1 of Atmel microcontroller. The output is taken from Port2 (X2-1, X2-2, X2-3, X2-4), which is embedded to DC motor through H-Bridge circuit. Microcontroller is programmed with a C program [14] in order to sense the environment and act according to the given sensor input, thus rotating the wheels accordingly. Here Atmel [16] acts as the artificial intelligent agent.

5.5 Design of Motor Driver

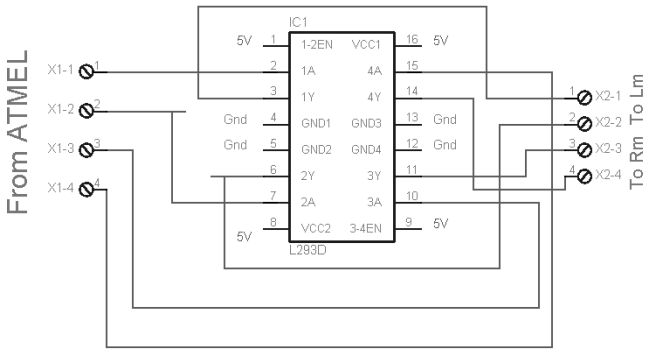

Figure 8 H-Bridge circuit pin connections

The Pin connections of L293D is shown in the Figure 8. H-bridge is used to drive the DC motor [15] at various voltages. The DC motor is driven with 6V. Outputs of Atmel microcontroller through Port2 is taken through the pins X1-1, X1-2, X1-3, X1-4 and given as input to H-bridge through pin2, pin7, pin10, pin15.

The pins X1-1, X1-2 are used to drive left motor Lm. The pins X1-3, X1-4 are used to drive right motor Rm. The output pins are X2-1, X2-2, X2-3, X2-4. The pins X2-1, X2-2 are connected to positive and negative terminals of Left DC motor respectively. The pins X2-3, X2-4 are connected to negative and positive terminals of Right DC motor respectively.

5.6 Integrated Circuit Design

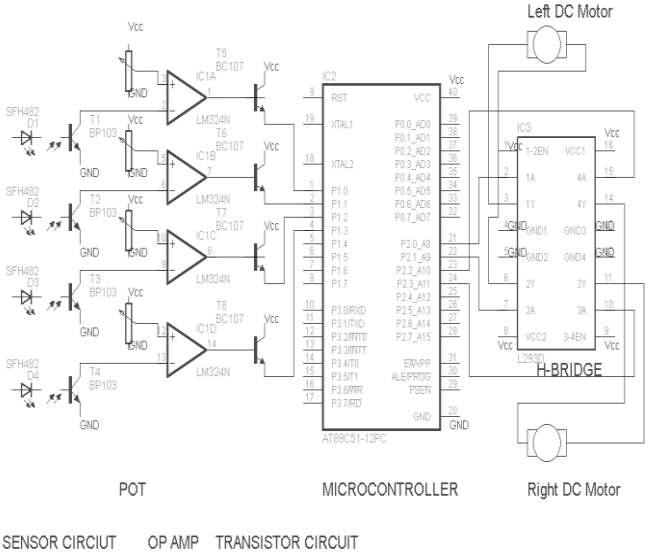

Figure 9 Integrated circuit

The entire circuit design is shown below in Figure 9. The integrated circuit consists of designing various individual circuits starting from sensor circuit to H-Bridge designing. Designing of these circuits are briefly discussed above. Analog signal from sensor circuit is passed on to comparator for digital conversion. Digital data is passed to microcontroller through transistor circuit for retaining its voltage. Master controller processes the data following the way it is programmed to. It runs the DC motors according to the input using H-Bridge circuit.

6. DRIVING METHODOLOGY

The various junctions identified according to the sensor input are tabulated in the Table 1.

S1	S2	S3	S4	DECISIONS
0	0	0	0	4 -junction (OR)T-junction
0	0	0	1	Only left (OR) Left and straight
1	0	0	0	Only right (OR)Right and straight
0	0	1	1	Take slight Left Turn
1	0	1	1	Take slight Left Turn
1	1	0	0	Take slight Right Turn
1	1	0	1	Take slight Right Turn
0	1	1	0	Target
1	1	1	1	Dead End

Table 1 Truth table for Junction Identification

S1- sensor1 , S2- sensor2 , S3-sensor3 , S4-sensor4

6.1 Motor Rotations

Motor rotations based on various turns are illustrated below in the Table 2.

POSSIBLE TURNS	LEFT MOTOR	RIGHT MOTOR	VALUES
4-junction(OR)T-junction	Stop	Forward	0x01
Only Left(OR)Left and Straight	Stop	Forward	0x01
Only Right	Forward	Stop	0x04
Right and Straight	Forward	Forward	0x05
Slightly Left	Reverse	Forward	0x09
Slightly Right	Forward	Reverse	0x06
Target	Stop	Stop	0x00
Dead End	Forward	Stop	0x04

Table 2 Truth Table for Motor Rotation

DENOMINATIONS (from left to right):

01 → FORWARD

10 → REVERSE

00 → STOP

7. MAZE SOLVING ALGORITHM

7.1 LSR Rule

LSR rule states Left direction has highest priority compared to straight and right directions. Similarly Straight has high precedence compared to Right. The precedence order is

Left > Straight > Right

7.2 Replacement Rule

The replacement algorithm is used to find the optimum path in a maze. This algorithm takes the turns stored during the teach phase and applies the replacement rules accordingly and corrects all the wrong turns. It identifies the wrong turns in the maze by finding the u turns made previously. Then it replaces the wrong turns made by the correct ones

7.2.1 Pseudo Code of Replacement Rule

It takes the turns stored during teach phase as values in 'res' array and 'i' as the iterator.
The replacement rules used are

LUL→ S
LUS→ R
SUL→ R

where L - Left turn, R - Right turn, S - Straight, U – U turn

```
void replay(String result)
{
        //declare i as int and assign it with 0
        while(result[i+1] is not null)
        {
                if(result[i]=='U')
                {
                  if(result[i-1]=='L' && result[i+1]=='L')
                        replace "LUL" as 'S';
                  else          if(result[i-1]=='L'          &&
result[i+1]=='S')
                        replace "LUS" as 'R';
                  else          if(result[i-1]=='S'          &&
result[i+1]=='L')
                        replace "SUL" as 'R';
                }
        }
}
```

7.3 Teach Phase

The teach phase consists of the robot following the black path and finding out the target. During this phase, the robot prefers left in case of junctions with left turn and prefers straight compared to right turns. At each junction where decision is to be made, the turn the robot takes is stored in the memory of the controller. The robot stores 'L' when it takes left and 'S' when it takes straight when a junction is encountered. In case of dead ends, it makes an u turn and stores 'U'. It does not store only Left and only Right turns.
When target is reached, the robot finishes the teach phase and proceeds to the replay phase.

7.3.1 Pseudo Code of Teach Phase

```
String Teach(Graph maze)
{
        Switch(path in maze)
        {
                case "Target":
                        //target reached.
                        Break;
                case "Dead end":
                        Take U turn;
                        Store 'U' in result array;
                        break;
                case "Left only":
                        Take left turn;
                        break;
                case "right only":
                        Take right turn;
                        break;
                case "Left or Straight junction":
                        Take left  turn;
                        Store 'L' in result array;
                        break;
                case "Left or Right junction":
                        Take left turn;
                        Store 'L' in result array;
                        break;
                case "Straight or Right junction":
                        Go Straight;
                        Store 'S' in result array;
                        break;
                case "Four way junction":
                        Take left turn;
                        Store 'L' in result array;
                        break;
                }
        replace(result);
        return result;
}
```

7.4 Replay Phase

During the replay phase, the robot will travel from the target to the starting point by travelling through the optimum path. The optimum path can be found by applying the replacement algorithm on the values stored during the teach phase.

The robot travels in the black path similar to the teach phase till it encounters a junction. Whenever a junction is encountered, it takes value from the stored memory and makes a turn.

7.4.1 Pseudo Code of Replay Phase

```
void replay(String result, Graph maze)
{
        while(result[i]!=null && path is a junction)
        {
                if(result[i]=='L')
                        Take left;
                else if(result[i]=='S')
                        Go Straight;
                else if(result[i]=='R')
                        Take right;
        }
}
```

8. WORKING

The working of the robot in solving the line maze is given as follows.

8.1 Line Path Following

This Robot follows a black maze by sensing the environment using IR sensors. It is programmed to follow the line perfectly with both the motors rotating in forward direction. Any deviations from the line will adjust the motor accordingly to ensure it follows the right path again. Two sensors are used for tracing the black line, and the left most, right most sensor are used for taking decision on junctions. The basic principle applied for following the line is the amount of reflection variation caused by white and black surfaces. It differentiates depending on the amount of reflection and follows the correct path.

8.2 Mobile Robot Navigation

Robot navigates with its sensors decision, the output from sensor circuit is given to microcontroller, which is programmed based on the input values. Given input is processed and corresponding output is given to H-bridge, to navigate the robot by rotating it in corresponding directions.

8.3 Knowledge Based Navigation

Knowledge based navigation is incorporated by means of LSR Rule. First Priority is given for left, second to straight and least priority to straight. Based on this knowledge, the robot navigates. In teaching phase, it uses this rule to navigate, it traverses through all paths including dead ends. When it encounters a dead end, it takes U-turn and traverses the same path again. It stores the values only in junctions, which is later used in replay phase. U-turn paths will be deleted and restored as different values in teaching phase which is later used by the robot for navigation in replay phase. In replay phase, LSR rule is not applied instead the replaced values are used to traverse the right path in an optimal way.

8.4 Reinforcement Learning

Artificial Intelligence concept of reinforcement learning is applied in replay phase. Reinforcement learning illustrates the concept of "learning from mistakes". In teaching phase, it learns the maze, it proceeds through dead ends. In replay phase, it solves the maze without dead ends in an optimal path, thus applying the concept of reinforcement learning. In replay phase LSR rule is not applied, instead the values stored in teaching phase are been replaced. The Only for decision making turns the values are stored for future use, for the only left/right turns no values are stored since it is of no use. Thus the robot follows an optimal path and reaches the target using reinforcement learning technique.

Replacement Rules used here are,

LUL→S
LUS→R
SUL→R

L-Left U-U turn R-Right S-Straight

If there is a U-turn, it implies those values should be replaced according to the replacement rules.

9. RESULTS

The snapshots of this project when the robot traverses through the line maze is shown here.

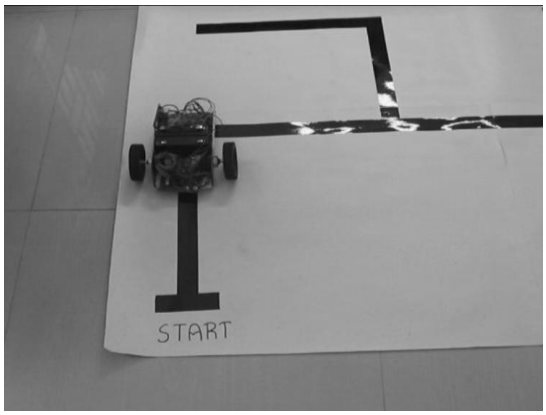

Figure 10 Robot in Initial Position

In the Figure 10, the robot starts from initial position in teach phase.

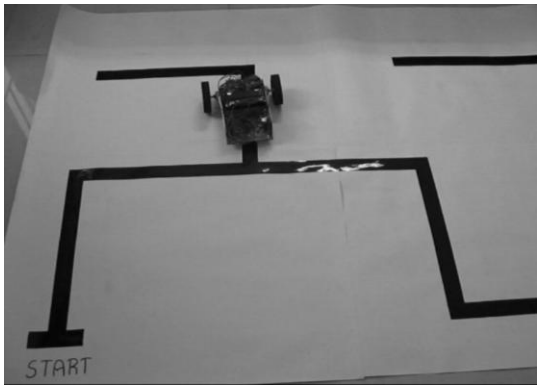

Figure 11 Robot Entering T – Junction

In the Figure 11, the robot makes an u - turn and enters into the T-Junction where it chooses the path to take. This occurs during the Teach phase.

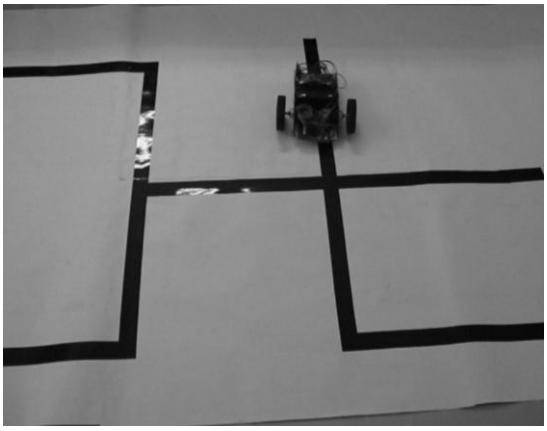

Figure 12 Robot crossing the Four Junction

In the Figure 12, the robot makes a left turn when it enters the Four Junction. This also occurs during the Teach phase.

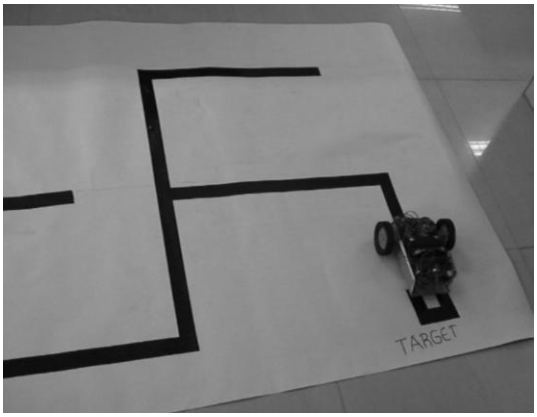

Figure 13 Robot Reaches The Target

The robot reaches the target and ends the Teach Phase. This is shown in the Figure 13.

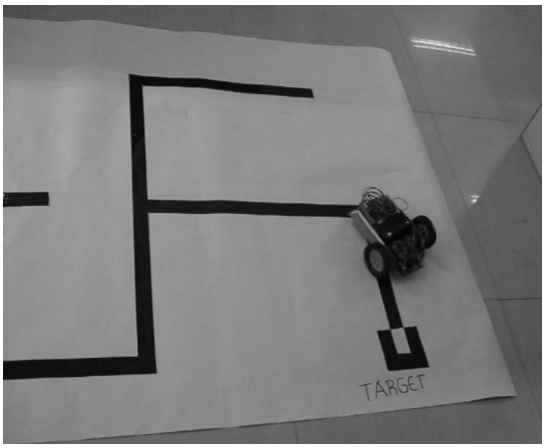

Figure 14 Robot Backtracking From Target

The robot enters the Replay phase in the Figure 14 and backtracks the path to reach the starting point in an optimum path.

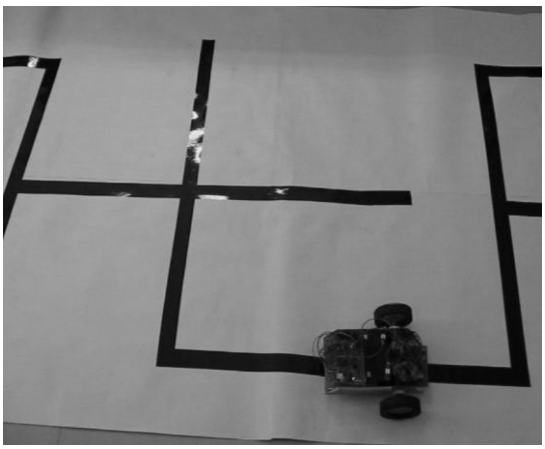

Figure 15 Robot Traversing the Optimum Path

In the Figure 15, the robot travels through the optimum path avoiding the dead ends to reach the starting point. This is done in the Replay Phase.

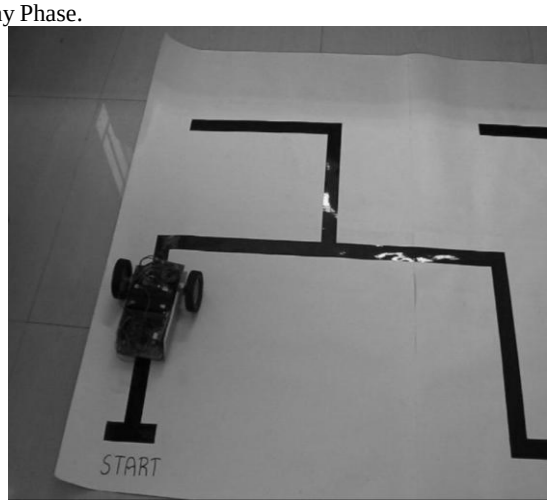

Figure 16 Robot Finishes Replay Phase

In the Figure 16, the robot reaches the starting point and finishes the Replay Phase by traversing the optimum path found through the replacement algorithm.

10. CONCLUSIONS AND FUTURE DIRECTIONS

This robot is mainly used for applications where human's intervention turns out to be danger. It finds the best and optimal path for solving a maze thereby reducing path cost in replay phase. As LDR sensors, IR sensors are not sensitive to ambient light thereby providing high efficiency.

Since the design is based on 6V power supply, problems such as voltage drop at many places due to rapid discharging of battery power are observed. Also memory capacity of the microcontroller wasn't too efficient to store multiple maze information's at any particular instant.

Due to limited memory capacity of Atmel AT89C51, the robot couldn't solve multiple mazes at one particular time. Chips of high capacity can be used to overcome this problem. Some times there might be a problem for this bot to track the path and decide due to voltage drop. Providing SMPS can turn out to be a better solution. Loops if available in maze is not resolved in this algorithm design. It can be overcomed by redesigning the algorithm to consider the path is a repeated one or not. Multiple targets in a single maze couldn't be solved with the current algorithm. Redesign of the algorithm with some user inputs like total number of targets available in the given maze and recalling the procedure for each target and storing the same may be the solution for this.

11. ACKNOWLEDGEMENTS

Authors are thankful to **Dr. S. Shiv Nadar**, Chairman of SSN Institutions, **Dr. Chitra Babu** Head of Department(CSE), faculty members of SSN Institutions and family and friends for valuable guidance and necessary testing facilities for this work.

12. REFERENCES

[1] Claude F. Touzet. 2004 "Distributed Lazy Q-learning for Cooperative Mobile Robots", pp. 5-13, International Journal of Advanced Robotic Systems, Volume 1 Number 1, ISSN 1729-8806

[2] Gerasimos G. Rigatos. 2008 "Multi-Robot Motion Planning Using Swarm Intelligence", International Journal of Advanced Robotic Systems, Vol. 5, No. 2

[3] Heero, K.; Aabloo, A. & Kruusmaa, M. 2005 "Learning Innovative Routes for Mobile Robots in Dynamic Partially Unknown Environments", pp. 209 - 222, International Journal of Advanced Robotic Systems, Volume 2, Number 3, ISSN 1729-8806

[4] Javier Minguez, *Associate Member*, *IEEE*, and Luis Montano, *Member*, *IEEE* 2009 " Extending Collision Avoidance Methods to Consider the Vehicle Shape, Kinematics, and Dynamics of a Mobile Robot" IEEE TRANSACTIONS ON ROBOTICS, VOL. 25, NO. 2.

[5] Jianghao Li; Zhenbo Li and Jiapin Chen. 2008 "Wheels Optimization and Vision Control of Omni-directional Mobile Microrobot", International Journal of Advanced Robotic Systems, Vol. 5, No. 2

[6] Kao-Shing, Hwang, Yu-Jen Chen, and Tzung-Feng Lin. 2008 "Q-learning in Multi-Agent Cooperation", IEEE International Conference on Advanced Robotics and its Social Impacts, Taipei, Taiwan, Aug. 23-25

[7] Maurizio Piaggio and Renato Zaccaria 1997 "Learning Navigation Situations Using RoadMaps", D.I.S.T. University of Genoa Via Opera Pia 13,I-16145 Genova, Italy

[8] Moteaal Asadi Shirzi, M. R. Hairi Yazdi and Caro Lucas. 2007 "Combined Intelligent Control (CIC) An Intelligent decision making algorithm", International Journal of Advanced Robotic Systems,Vol.4,No.1

[9] Murphy, R.; Kravitz, J.; Stover, S.; Shoureshi, R. 2009 "Mobile robots in mine rescue and recovery**",** Robotics & Automation Magazine, IEEE Volume 16, Issue 2, Pages: 91-103

[10] Noureddine Ouadah, Lamine Ourak and Farès Boudjema. 2008 "Car-Like Mobile Robot Oriented Positioning by Fuzzy Controllers", International Journal of Advanced Robotic Systems, Vol. 5, No. 3

[11] Shaker, M.R. Shigang Yue Duckett, T. 2009 Dept. of Comput. & Inf., Univ. of Lincoln, Lincoln, UK, "Vision-based reinforcement learning using approximate policy iteration", Advanced Robotics, ICAR 2009.

[12] Szoke, I. Lazea, G. Tamas, L. Popa, M. Majdik, A. 2009 "Path planning and dynamic objects detection", Advanced Robotics, ICAR 2009.

[13] Zhichao Chen and Stanley T. Birchfield. 2009 Senior Member, IEEE, "Qualitative Vision-Based Path Following", IEEE TRANSACTIONS ON ROBOTICS, VOL. 25, NO. 3.

[14] Vinod G. Shelake , Rajanish K. Kamat, Jivan S. Parab, Gourish M. Naik, "Exploring C for Microcontrollers - A Hands on Approach".