An Autonomous Path Finding Robot Using Q-Learning

[1]Madhu Babu V, [2] Vamshi Krishna U, [3] Shahensha.SK
[1]Dept. of Mechanical Engineering, [2] [3] Dept. of Electronics & Communication Engineering
Rajiv Gandhi University of Knowledge Technologies, R.K Valley
[1]accesstomadhu@gmail.com, [2]vamsy.krishna39@gmail.com, [3]iiitshahensha@gmail.com

Abstract— **The main objective of this paper is to develop an autonomous robot that show the use of Q-learning for navigation in a complete unknown environment. This will calculate the shortest path from current state to goal state by analyzing the environment through captured images. Further the captured images will be processed through image processing and machine learning techniques. The proposed method also takes care of the obstacles present in that environment. Initially, the unknown environment will be captured using a camera. Obstacle detection method will be applied on it. Then the grid based map obtained from vision based obstacle detection method will be given to Q-Learning algorithm which will be further made live with motion planning.**

Keywords—Autonomous Robots, Machine learning, Q-learning, Reinforcement learning, Path planning, Motion planning.

I. INTRODUCTION

Nowadays, world is running towards automation. When it comes to task completing robots like vacuum cleaning robots, drones and many other, the path and motion planning will be a biggest task and it is the inevitable thing of automation [1][4]. The main aim of these robots is that it should reach its goal state form current state to accomplish the task. The main problem comes when it has to reach the goal state from the current state by consuming less energy and time and of course by the best feasible path. If the robot is planning path in a known environment, path planning itself completes the task of us but most of the times the environment will be unknown which makes our task even more tangled and cumbersome [10]. In order to reach goal state, and more importantly where no maps of that environment are available, this algorithm would be a more useful and handy solution for both path and motion planning [4]. Even the same can be magnified for robots like mars rovers where the robot should face a completely unknown environment [10].

The problem is to plan the safest path first and it should a have a perfect motion plan execution to accomplish its task. Initially people were used range finder like ultrasonic sensors, IR sensors, Laser range finders and many other to detect the obstacle [7] while planning the path. But with pinging this problem with vision based techniques the problem will be melted into a simple one [1]. With vision based system the robot can study the total environment with a single capture of image [1]. Before this image processing techniques many algorithms were used to study the environment [7]. Even the same can also implied to study the environment with the sensors described above. Doing all this will charge much time and energy. To overcome all the problems with the traditional methods such as path finding using range finders we have developed a method based on vision based navigation system which helps the robot in both path planning and motion planning [1][5].

Earlier, the robots were programmed with the consideration of each and every aspect which may create noise while achieving its goal [3]. The main drawback comes when the supervisor misses some aspect. To get over this draw back somehow an artificial intelligence should be given to the robot in order to be conscious about the environment. There are many techniques involved in getting the same. In this paper we are describing one of those approaches.

The method described in this paper aims to calculate the path more effectively and efficiently. The proposed method was experimented on smooth surfaces and it can be extended to even more knotty surfaces with stereo imaging. The images captured from camera which will be further processed through Image processing techniques like edge detection, blurring, thresholding etc [11]. We used open CV for Image processing, path calculator and we used an approach of reinforcement learning for the same.

Open CV is an open source library containing many algorithms for both image and video analysis and also it is one of the robust, user friendly, multi-platform supporting media for image processing. We can get vibrant community to share and solve the problems.

Reinforcement learning is a sub area of machine learning [2]. We adopt for the Q learning algorithm which is model free reinforcement learning technique [2]. This algorithm can be used to find the best action policy which gives the highest reward to the action which is the best for reaching goal state. This algorithm follows Markoniv Decision Process MDP [2]. In this there will be an action value function that will ultimately give the best action from the given state to reach goal state by following optimal policy.

II. OBSTACLE DETECTION

In recent times, image processing techniques are proving their mark on all fields like automation and many applications including security and authentication. Recognition and detection of obstacles is one of the challenging tasks. An autonomous robot should capture the environment in order to detect the object to avoid the collision while reaching the goal [1] [8][9]. The capturing and processing of the images is very essential character of an autonomous robot.

The first step of the algorithm is to capture the environment and then it should be processed for obstacles. After capturing the image fig 1(a), it has to be converted into a gray scale image. The conversion color image (first captured image) to gray scale image will reduce the information quantity will be reduces to one third of color images. After conversion the total image will be converted into black, white and grey pixels. This is so important that natural environment images are very difficult to analyze. This obstacle detection method uses basic concepts of image processing named edge detection and edge dilation along with thresholding [11].

The main purpose of edge detection is to reduce the amount of data in an image when only structural information is needed for further image processing. In this algorithm, we adopt for Cany edge detector. It is developed by F.Canny in 1986 [11] . The Cany edge detection algorithm will work on 5 separate steps. Smoothening, Finding gradients, Non-maximum suppression, Double thresholding and Edge tracking by hysteresis [11]. Smoothening is nothing but eliminating the noise by blurring the image. The image will be smoothened by Gaussian filter with a feasible standard deviation σ (1.4). After smoothening the edge algorithm will finds where the grayscale intensity of image changes the most. To determine those gradients in x and y direction after applying sobel-operator. The computed gradients will indicate the edges. After that the blurred edges should be converted to sharp edges, this will be done by preserving local maxima in gradient image and deleting everything else. Then double thresholding will be applied because even after suppressing some false pixels will be there because of color variations or surface roughness. Edge pixels stronger than high threshold will be marked as strong, which are weaker than low threshold will be suppressed and the middle pixels will be marked as weak. For our case high and low thresholds are 200 and 100. Finally edge tracking by hysteresis will be done by considering BLOB- analysis. The marked stronger pixels will be there in the final image but the weaker pixels will processed through Binary Large Object analysis. In this, the blob which having at least of one stronger pixel will preserved other will be suppressed. After suppressing, final images with true edges will be resulted where in the resulted image fig1 (b), black represents free space for path and motion planning and white will be the edges of obstacles. Once edges were found then the edges were dilated for the safe path planning.

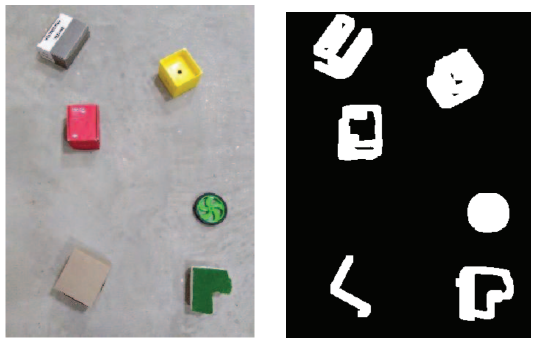

(a) (b)

Fig 1: a.Region of Interest , b. Image after edge detection with dilation

III. REINFORCEMENT LEARNING

The most common and complex problem in navigation is path planning [6][1]. This path planning will take care of energy, time and distance. This path planning will calculate the best shortest and obstacle collision free path [8][9]from the current state to goal state. To take care of all these we adopt for a machine learning approach called Reinforcement Learning [2]. The following lets you know in detail about this.

A. *Background*

Reinforcement learning is a sub area of machine learning [2]. It concerns about how an agent ought to take an action which indirectly maximizes its long term reward. Basically reinforcement learning will have five basic parts named like agent, action, reward, state, and environment. An agent is the learner or decision maker. Action is the decision made by the agent. Reward is the term which reflects the decision. State is nothing but where the agent is being located and finally environment is the area which the agent interacting with. The relationship between all these are shown in fig 2.

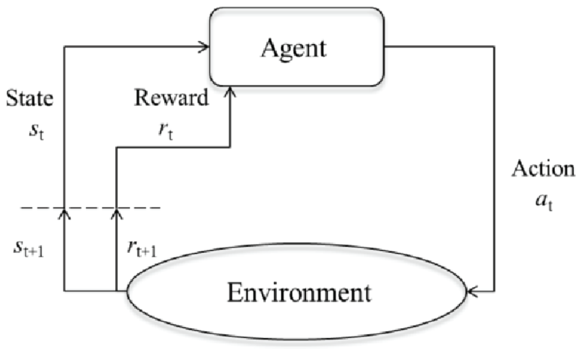


Fig 2. Architecture of Reinforcement Learning

The agent updates Q-values by state S_t and reward for that state r_t at the time t. The agent at current state S_t takes an action a_t , on interacting with the environment a new state S_{t+1}, based on the new state(either good / worst) a reward r_{t+1} , and will be resulted. Now then the resulted state after taking action S_{t+1} will be S_t . The agent will accumulate experience in the name of rewards. Based on the state and the experience the agent will take actions to reach its desired state. After long term learning the agent updated Q table. Finally the agent will have the sufficient experience to take the correct decision which will result in long term highest reward.

B. *Q-Learning Algorithm*

In reinforcement learning methods, Q learning is a model free method based on dynamic programming. Q learning allows the agent to explore and exploit these hidden markoniv environment for an optimized policy to take an action which results in long term reward. The Q learning algorithm will flows as follows.

```
Initialize Q(s,a) arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derive from Q (e.g. ε-greedy)
    Take action a, observe r,s'
    $Q(s,a) \leftarrow Q(s,a)+\alpha\,[\, r + \gamma*\max(Q(s,a)-Q(s,a)]$
    $S \leftarrow S'$
  Until s is terminal
```

IV. PROPOSED METHOD

Before applying this method, one should need to construct a state space S, action space a, and reward function r.

A. *State space (S)*

The agent should be able to track itself by positioning its current and the goal states. Thus we need an interlinked system which is otherwise called as state space. In our case the environment is the image came out of vision based obstacle detection system. For easy handling and calculation, the image is divided into a simple grid like fig 3, each point where vertical and horizontal lines meet is termed as node. All these nodes are combinely termed as state space and every node will be a state.

The size of the state space also plays an important role in learning. This will affect the learning quality and spending time of learning. The complex state space will cost more learning time. So, splitting up the state space is very important aspect in order to get a matured analysis about the environment.

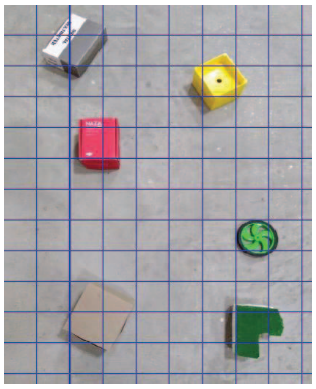


Fig 3: State Space (Grid)

B. *Action space (A)*

The designing of action space is a critical issue. This action space is responsible for state transition. On some iterations the Q-values will be updated and that will output a feasible action to the environment. In learning state itself, this action space is very important. When there is need of precise actions the action space should be complex as shown in fig 4 below. The number of actions from a state also matters in learning quality and time. If the number of actions increases, the agent gets more mature but it leads to high time complexity. So, the number of actions are split up based on the requirement of maturity level required for the environment.

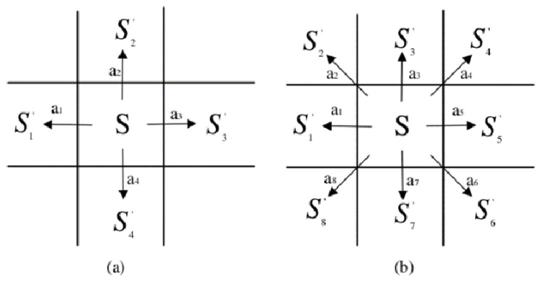


Fig 4: a. action space with 4 actions b. action space with 8 actions

C. Reward Function (R):

In reinforcement learning, the reward function will be used to determine whether the action taken by the agent is good or worst from the current state. In accordance with these reward values, Q-values will be stored as experience. In long term learning, the agent uses the accumulated experience values to derive an optimized policy for that environment. In the learning process the reward function is set as below:

- If agent reaches out of image
  Reward (r) = -100
- else if next state S' is the goal state
  Reward (r) = 100
- else
  Reward (r) = -1

V. PATH PLANNING

The algorithm proceeds as initialization of Q(s,a) , get the current state S, take an action based on policy derived from Q(s,a) (greedy policy), get an immediate reward 'r' depending on the action taken, update Q-values based on the formula shown in algorithm and then eventually new state will be transformed as current state in order. After completion of learning the resulted path is shown in fig 5.

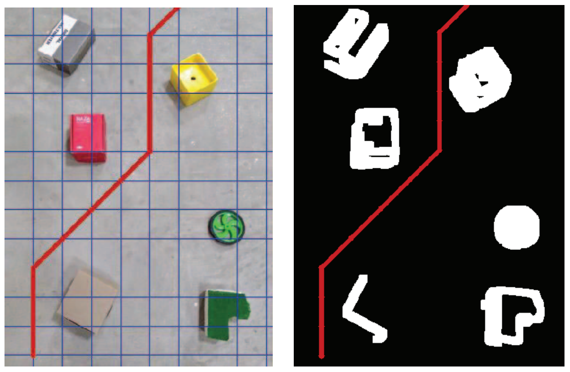


Fig 5: Shortest path calculated by Q-Learning in ROI and Threshold image

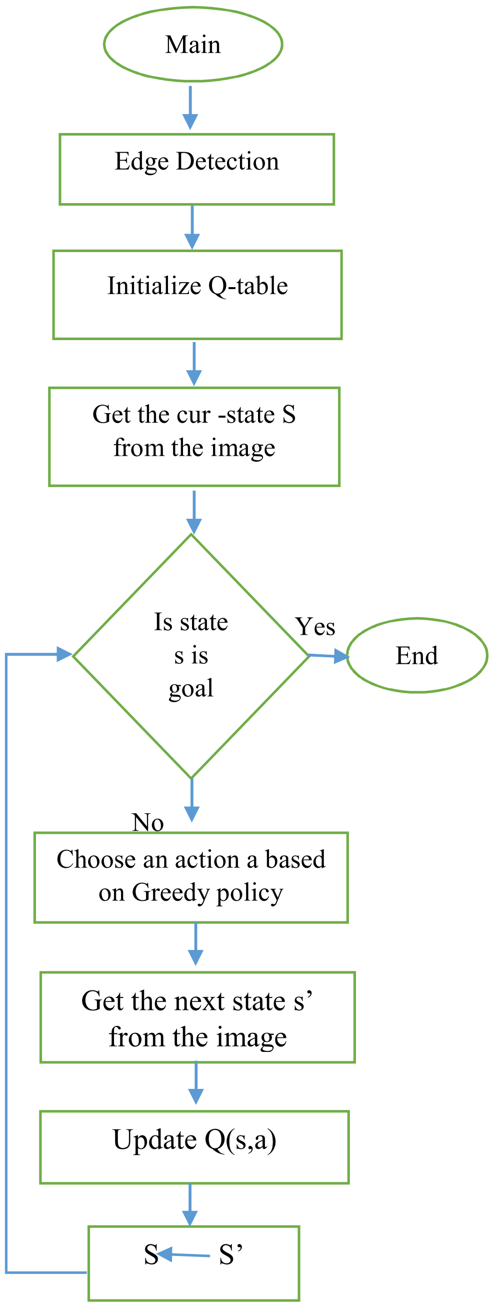

Main
Edge Detection
Initialize Q-table
Get the cur -state S from the image
Is state s is goal
Yes
End
No
Choose an action a based on Greedy policy
Get the next state s' from the image
Update Q(s,a)
S ← S'

Fig 6. Flow chart of Learning Process

VI. MOTION PLANNING

After path planning, motion planning is also an important case. Before motion planning, the environment image will be fed to the base station which processes the image. After calculating the path from previous section, we will be having the node actions to move (left, right, up, down, corners etc.). These actions will be fed to the robot. In our case, both goal state and starting states were given manually. The ROI will be divided into grid with a unit box of size 43 X 43 pixels which will be scaled later using the node lengths and height with which an image of environment is taken. The Scaling factor depends on the ratio of actual distance between two nodes to the number of coordinates between them.

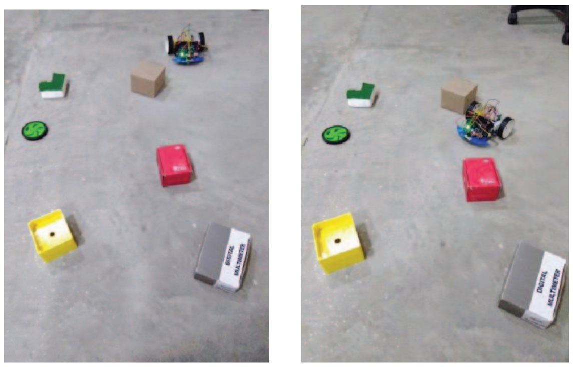
A. Hardware Design

Sensors play a vital role in making autonomous robots. Automated Guided Vehicles (AGV) also come into this category. The method which was discussed above will not be suitable for curved paths. To attain all possible states perfectly, there should be an accurate method for calculating distances.

Though there are many complex instrumentation, sensors and techniques for the calculation discussed above, keeping in mind of the cost-feasibility of robot, a general and very basic method of calculating distances is adopted in this using IR sensors.

Black marks are placed on white wheels symmetrically such that adjacent black marks should be separated by equal angles between them. IR transmitters and receivers are placed accordingly so that if a black mark is traced, the output from sensors are counted. Depending on the count values, the distance traversed can be calculated by using the count of alternate zeros and ones multiplied by the circumference of wheel. Eventually, the rotation of wheels has to be ceased if a particular distance is reached. By this we can adjust diagonal and side distances depending on the action taken by the robot.

Now, based on the further action to be taken, the robot takes a turn to face the corresponding direction which can also be implemented using the sensor readings. The total process repeats among straight traversing of path and setting of direction accordingly until goal is reached. The motion scenario shown in fig 7.



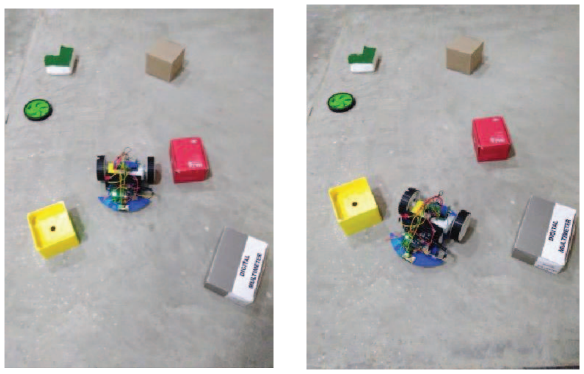


Fig 7. Motion Scenarios after Path planning

VI. CONCLUSION

In this paper, path and motion planning for a robot were described to make it autonomous in unknown environments. These were achieved using image processing and reinforcement learning techniques. The robot will be equipped with a camera on top which captures the snaps of the unknown environment. Further, those will be processed through canny edge detector for tracing free space and obstacle location. Then, Q learning is applied on it where the total image will be normalized like a grid. This grid-based approaches are easy to implement and it allows fast memory access. Implementation and discussion about a feasible path to reach goal through Q-values were made. Once path was found, robot clears those obstacles to reach goal state without obstacle collision and motion planning was made through an intuitive technique.

VII. FUTURE WORK

In this paper, we adopt basic method of edge detection for tracing obstacles on smooth surfaces which discards the height of obstacles.

By theory, if the surface has any unidentified intensity variations makes it to detect as an obstacle. The third coordinate of depth matters in real time environments for even more optimal paths. To overcome this, stereo image processing techniques can be done to get depth of the images. Once depth was found, we can simply discard the obstacles which have depth below a certain range. Then, this discussed Q learning can be applied on those for easy navigation in even more knotty environments.

VIII. REFERENCES

[1] Heramb Nandkishor Joshi and Prof. J.P. Shinde , "*An Image Based Path Planning And Motion Planning for Autonomous Robot*" International Journal of Computer Science and Information Technologies, Volume 5(4),2014,ISSN :4844-4847

[2] R.S.Sutton and A.G. Barto, "*Reinforcement Learning An Introduction*", MIT Press, Cambridge 1998

[3] Sandeep Konam, Mansoor Basha Patan, Renuka Lakshmi Dasari "Multi-Functional Real Time Path Programmable Robot ",proc IEEE international Conference on Vehicular Electronics and Safty (ICVES),December 16-17,2014,Hyderabad India.

[4] Xavier P_erez, Cecilio Angulo, Sergio Escalera "*Vision-based Navigation and Reinforcement Learning Path Finding for Social Robots*" September 3,2010.

[5] Claudiu Popirlan, Mihai Dupac "*An Optimal Path Algorithm for Autonomous Searching Robots*" Annals of University of Craiova, Math. Comp. Sci. Ser. Volume 36(1), 2009, Pages 37–48 ISSN: 1223-6934.

[6] Stephen J. Tobias, A. Antonio Arroyo "*Autonomous Path finding*", 2000 in proc. Florida Conference on Recent Advances in Robotics May 4-5, 2000, Florida Atlantic University.

[7] Amit Konar, Senior Member, IEEE, Indrani Goswami Chakrabarty, Sapam Jitu Singh, Lakshmi C.Jain and Atulya K.Nagar, "*A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot*" in IEEE Transactions on systems, Man and Cybernetics: Systems Vol. 43 No 5, September 2013

[8] Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "*Adaptive evolu-tionary planner/navigator for mobile robots,"* IEEE Trans. Evol. Comput.,vol. 1, no. 1, pp. 18–28, Apr. 1997

[9] Z. Bien and J. Lee, "*A minimum-time trajectory planning method fortwo robots*,"IEEE Trans. Robot. Autom., vol. 8, no. 3, pp. 443–450,Jun. 1992.

[10] M. Gerke and H. Hoyer, *"Planning of optimal paths for autonomous agents moving in inhomogeneous environments*," in Proc. 8th Int. Conf.Adv. Robot., Jul. 1997, pp. 347–352.

[11] John Canny. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-8(6):679–698, Nov. 1986