International Journal of Future Generation Communication and Networking
Vol. 13, No. 2, 2020 pp.1340-1347

Reinforcement Learning Based Autonomous Path Finding Robot for Dynamic Environments

Rajasekaran Thangaraj[1], Sivaramakrishnan Rajendar[2], Premkumar Murugiah[3], Vidhya Kandasamy[4], Rahul R[5]

[1,2,3,4,5]*KPR Institute of Engineering and Technology, Coimbatore*
[1]rajasekaran30@gmail.com, [2]sivaraamakrishnan2010@gmail.com,
[3]premkumarmurugiah1395@gmail.com, [4]vidhyakandasamy@gmail.com
[5]rahulram30122000@gmail.com

Abstract

In real world the path identification for autonomous robot in dynamic environment is difficult process. However, this paper introduce a reinforcement learning based double DQN learning model for autonomous robot to handle the problem of dynamic path identification. The experiment is performed to evaluate the effectiveness of the algorithm using the simulation tool Udacity. The fundamental idea is to determine the problem as state-action problem and discover the value for each state in its workspace. The performance of the model is examined in two different scenarios such as normal path and hill path. The results of the experiment confirm that the model shows better performance in normal path with an accuracy of 92% than hill path which is of 86%. Simulation results confirms that the reinforcement learning based model for autonomous robot provide better accuracy in dynamic environment for normal path than the hill path.

Keywords*: Path Finding, Reinforcement Learning Algorithm,Udacity simulator,Autonomous Robot*

1 Introduction

The autonomous robots creation is one of the important goal in robotics and its most important problems is a motion planning in the environment. The motion planning direct the robot from the original position (initial configuration) to the destination (goal configuration) without collision owing to the obstacle [1]. The motion planning consist of two category such as path planning and trajectory planning. The path planning comprises obstacles in the path without any collision in an environment. The robot path identification in the environment depends on the sensor input from the robot manipulator [2].The trajectory planning involves the velocity and time of the robot for identifying path to reach the goal state. There are various strategies available for solving the path finding problems. Some of them need the workspace to be two-dimensional and therefore the objects to be a plane figure. The foremost common strategies is square measure supported road-map, cell decomposition and potential fields. The main drawback of these approaches is that the majority of them turn out collision-free plane figure line methods and this type of geometric methods are suitable to avoid the collisions however not applied for non-holonomic execution of the robot [3]. The techniques to create this method with non-holonomic constraints on the far side of the agent and therefore the atmosphere, will determine four main sub-elements of a reinforcement learning system: a policy, a reward function, a value function, and optionally, environmental model. A policy defines the behavior of the agent's at a particular time, a reward function indicates the goal of reinforcement learning, a value function specifies smartness within the long-standing time, and the model behavior in the environment [4]. This approach relies on reinforcement learning and is impressed on potential fields' strategies. [5]. Subsequently, to plan the path within the environment collisions with degree optimum policy and it's worth perform square measure found by a reinforcement learning algorithm. This approach will avoid the matter of path generation and non-holonomic constraints, and provides a model of the environment wherever a path from any configuration to the goal

ISSN: 2233-7857 IJFGCN

configuration is found[6]. In this paper, reinforcement learning based autonomous path finding method is proposed employing potential fields methods. Section 2 discuss about the reinforcement learning algorithm, Section 3 presents the hardware and software required for implementation and methodology used is discussed in Section 4.The experimental results are presented in the section 5 and section 6 provide the conclusion.

2 Reinforcement Learning

In recent years, Reinforcement Learning (RL) is most popularly used in the fields such as robotics, intelligent decision making and decision analysis.RL is the real time learning method in which robot perform action based on the input received through environment interaction. Moreover, robot decides the action according to signal obtained from the environment and find the best movement over trial and error method[7].RL is based on the reward value which is received by interacting with the undefined environment. The agent is employed to enhance the reward value obtained from the surrounding environment. The robot can identify the optimal path even in the dynamic environment using RL method [8].

2.1 Background Knowledge

The environment would lead the state to the following state and additionally offer the state reward supported by the action. RL helps to learn to make and modify the policy supported a series of state-action-reward procedure.

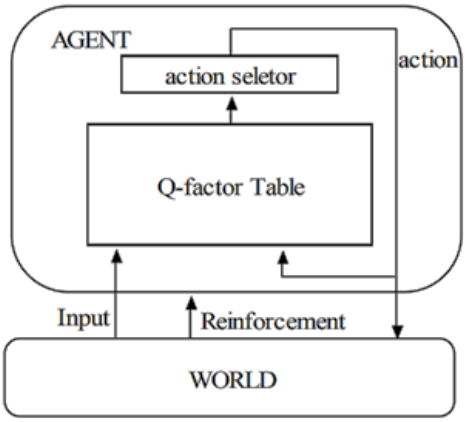

Figure.1 Structure of Reinforcement Learning Model

Some of the important term and notation used in figure 1 and the following paragraph are specified here.

- State set S = f s_1, s_2, s_3, ... g
- Action set A = f a_1, a_2, a_3, ... g
- State transition probability function T = T(s^0 j s, a) = P[S_{t+1}=s^0 j S_t = s, A_t = a]. This means the probability of transition from state s to s^0 when taking action a.
- Reward function r = r(s,a) = E[R_{t+1}j S_t = s, A_t = a]. This means the expected value of reward given s and a.
- Policy = (a j s) = P [A_t = a j S_t = s]. This means the probability of choosing a given the s.
- Discount factor 2 [0; 1]

ISSN: 2233-7857 IJFGCN

2.2 Learning Process

The agent learns from the reward for every action and it tends to assign every action a Q value and decision this action-value operate q(s, a). If the agent is at state s and with four decisions of action, then it tend to get four Q value for this state s. The Q square measure at the start position set to zero and may be updated by the reward that it incline to simply encounter.Moreover, it will take the longer term reward into consideration just in case it would be fooled by the present reward [9]. It is close to one, it means that the long term reward is sort of as necessary because of the current reward. It will categorical the mixture of current reward and therefore the future reward by the subsequent equation:

$$q\pi(s,a) = r(s,a) + \gamma Ps0 \in ST(s0 \mid s, a)\ Pa0 \in A\pi(a0 \mid s0)\ q\pi(s0, a0) \quad (1)$$

State s0 represents next state relative to the current state s. The action a0 goes to be designated before the agent extremely in state s0, therefore it want to require an expectation of Q (s0; a0) which is wherever the summation and take the role. Likewise, it tend to aren't certain what state s0 can the agent enter once taking action at a state s, therefore it want T (s0 js; a) to represent the chance and take every prospect into consideration. Once the agent steps in most of the states and tries most of the actions in every state, it will construct an associate instruction map to demonstrate the standard of taking a selected action given a selected state [10]. In the end, the agent will choose for the best value of action at every state which might presumably to guide. It can update q(s; a) through off-policy or on-policy wherever the update rule is going to be per next section. What distinguishes these 2 policy is however it tend to update q(s; a). For the off-policy, it tend to use the max q(s0; a0) and r(s; a) to update the q(s; a). For the on-policy, it tend to use q(s0; a0) and r(s; a) to update the q(s; a). Value operate v(s) may be operating to a grade selected state by taking all the Q value during this state into consideration.

$$v\pi(s) = P\ a \in A\pi(a \mid s)\ q\pi(s, a) \quad (2)$$

3 Materials and Methodology

3.1 Experimental setup

ROG Strix GL553 comes with Windows 10 pre-installed and features a 7th-generation Intel® Core™ i7 quad-core processor, NVIDIA® GeForce® GTX 1050 graphics and full Microsoft® DirectX® 12 (GPU) required for running our model.

3.3 Training and Autonomous Mode

Udacity Self-Driving Car Simulator is used in this work. The simulator has the interface which support manual driving and autonomous driving. In the training mode, the car is operated manually to record the driving behaviour. Then, the recorded image data are used to train the model. Each driving instruction contains a steering angle and an acceleration throttle, which changes the car's direction and the speed (via acceleration).Moreover, in the autonomous mode, the model is tested to see how well the model learned to drive the car without dropping off the road. Each driving instruction contains a steering angle and an acceleration throttle, which changes the car's direction and the speed (via acceleration). As this happens, our model generate the new image data frames at real time.

3.4 Deep Reinforcement Learning with Double DQN-Learning

In this work, neural network based reinforcement learning is applied with Double DQN algorithm for autonomous path planning. The practicality of a neural network is to map the state s to q(s;). The target is generated by another neural network known as a target network ruled by zero that contains a similar design. Currently, it has a tendency to get two neural networks with an identical structure ruled by parameters and zero separately. However, zero is at first traced from throughout coaching in one episode (agent begin from the beginning to the end), it are going to copy to zero for each N steps. In alternative words, it don't update zero at intervals these N steps. The equation of target is:

ISSN: 2233-7857 IJFGCN

$$Yt \equiv rt+1 + \gamma q(St+1, \text{argmax } aq(St+1, a; \theta t), \theta 0\ t) \quad (3)$$

The 1st input St+1 into the network and opt for the action a corresponds to the output vector denoted by argmax(St+1; a; t). At the same time, it input St+1 into the zero network and acquire another output vector. It have a tendency to figure the target Yt by combining the present reward rt+1 and therefore the alphabetic character worth from the zero net-work q(St+1; a; t0)[11,12]. The algorithm for training is shown below.

Algorithm : Double DQN Algorithm

Input: D -empty replay buffer; θ-initial network parameter; θ^t -copy of θ N_r-replay buffer max size; N_b-training batch size; N-target network update frequency

{

for (*episode* $e \in$ { *1, 2,...,M*}){

initialize frame sequence x ← ();

for ($t \in$ { *0, 1,...*}) {

Set state $s \leftarrow$ x, sample action $a \sim \pi B$;

Sample next frame x^t from environment s given (s,a) and receive reward r, and append x^t to x;

if |x| > N_f then delete oldest frame x^{tmin} from x end;

Set $s^t \leftarrow$ x, and add transition tuples (s, a, r, s^t) to D, replacing the oldest tuple if $|D| \geq Nr$;

Construct target values, one for each of the N_b tuples: Define

$$a^{max}(s^t;\theta) = argmax\ t\ q(s^t,a^t;\theta)$$

$$y_j = \begin{cases} r & \text{if } s^t \text{ is terminal} \end{cases}$$

Do gradient descent step with loss $''y_j - q(s, a;\theta)''^2$;

Replace target parameters $\theta^t \leftarrow \theta$ every N

}

4 Architecture for agent learning

4.1 Observation (Ot)

Observation made by the agent is taken as angles of joints in which the agent takes the decision and process to the real world entity i.e. environment. Here the path gets observed by the camera sensor and decides the path to travel.

4.2 Reward (Rt)

Consider the previous situation of having a decision to go either left or right and have a rough idea about the two paths, abstractly know about the path in each way and the estimated time it would take if you chose that path. Each reward value decides whether it is a Positive or Negative reward value.

ISSN: 2233-7857 IJFGCN

4.3 Action (At)

Actions are taken based on the reward values in which decision are made according to the angels produced by agent during action time.

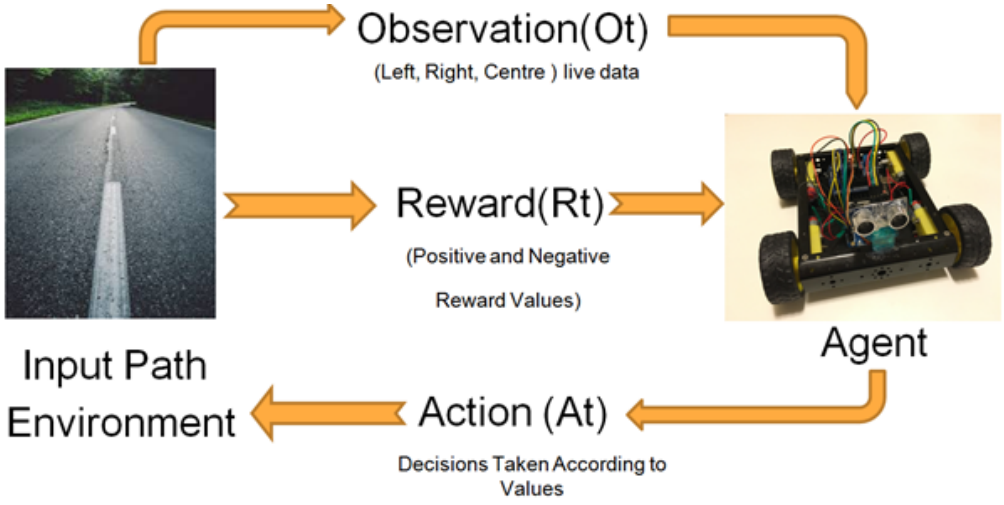

Figure 2. Agent Learning Model

5 Experiments and Results

The performance of the proposed model is measured by driving the car in normal and hill path. Initially, the model developed using RL is implemented using simulator used to drive the car manually for the normal path. While driving the car the model generated the training data from the environment. Four operations such as drive forward, left, right and brake or reverse are applied to control the car to drive in the right path. The data produced from three camera sensor is lively processed and taken as a input for RL learning techniques and behaves according to the environment inputs The live feeds obtained during the training phase are shown in the Fig.5. The driving action performed and its angles are depicted in Table.1.Table.2 compares the training time and memory speed obtained by the model running in CPU and GPU. Similarly, the model is trained for hill path and the training data are recorded. Subsequently, the model is tested for autonomous mode in both normal and hill path.Fig.3 and Fig.4 shows the autonomous mode of car driving in normal and hill path. The model achieved driving accuracy of 92% for normal path and 86% in hill path as shown in Fig.6.

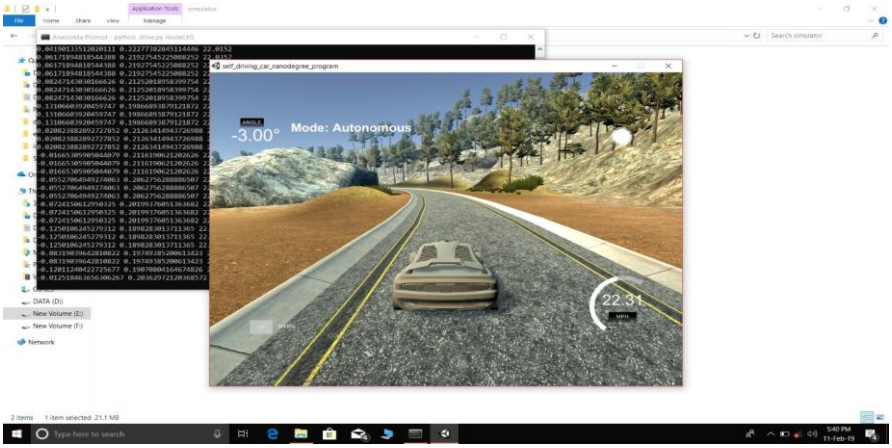

Figure 3. Simulation for normal path

ISSN: 2233-7857 IJFGCN

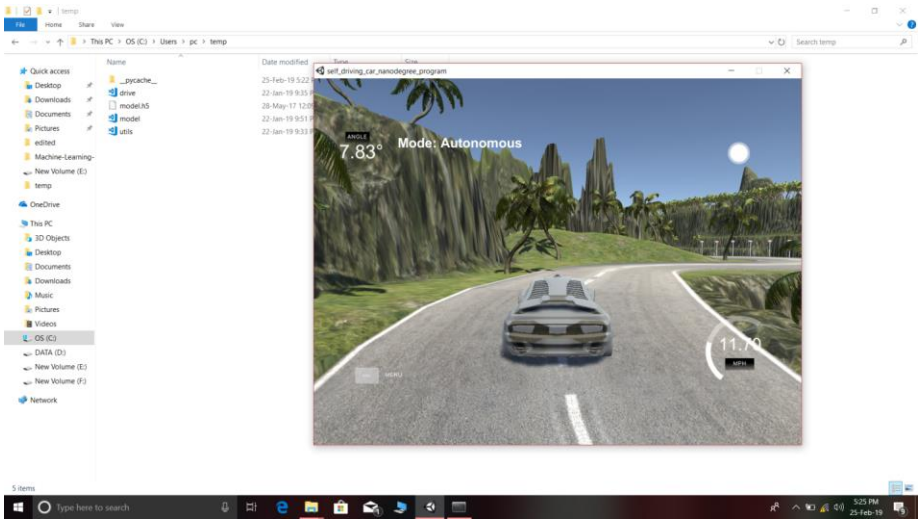

Figure 4. Simulation for hill path

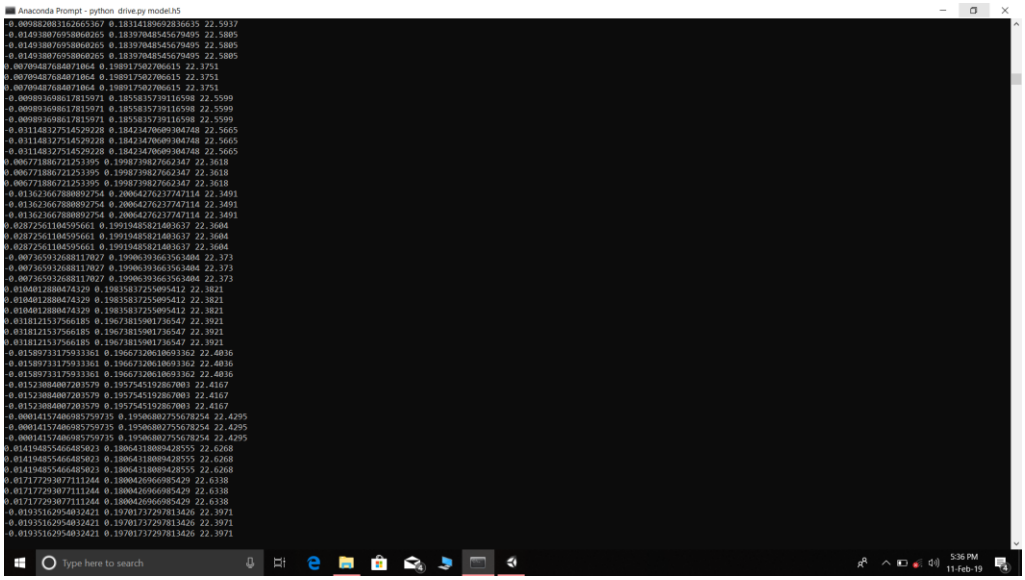

Figure 5. Live data feeds

Table 1 Angels and Actions takes in path

Angel	Action
-0.028 to -3.67(approx.)	Left

ISSN: 2233-7857 IJFGCN

6.23 to 10.09(approx.)	Right
0.43 to 2.29(approx.)	Forward

Table 2 Comparison of training time taken for CPU and GPU

Parameters	CPU	GPU
Training Time	8 Hours	4.5 Hours
Memory Speed	750 Mbps (Depends on Architecture)	4 Gbps (Depends on Architecture)

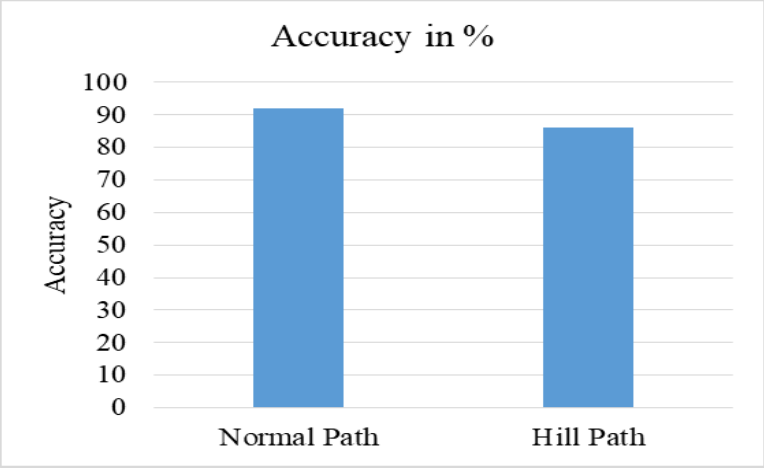

Figure 6 Accuracy comparison for normal and hill path

6 Conclusion

In this paper, a reinforcement learning algorithm is introduced to identify the path for autonomous robot in dynamic environment. The path generated by the reinforcement algorithm is executed by the robots and produces more accurate results. The results are simulated using Udacity simulator and the performance of the model is evaluated in normal and hill path. The experimental results show that the model provides a better accuracy of 92% for normal path and providing an accuracy of 86% for hill path. Simulation results show that the reinforcement learning based model for autonomous robot provide better accuracy for normal path even for the dynamic environment than the hill path. Using reinforcement learning method path can be identified without need of prior knowledge about the environment.

ISSN: 2233-7857 IJFGCN

References

[1] Tamar, A., Wu, Y., Thomas, G., Levine, S., & Abbeel, P. (2016). Value iteration networks. In Advances in Neural Information Processing Systems (pp. 2154-2162).

[2] Gerke, M., and Hoyer, H., Planning of Optimal paths for autonomous agents moving in inhomogeneous environments, in: Proceedings ofthe 8th Int. Conf. on Advanced Robotics, July 1997, pp.347-352.

[3] Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Thirtieth AAAI conference on artificial intelligence.

[4] Dung-Yi, Chao, Purdue University West Lafayette, Indiana 47906. Reinforcement Learning on Route Planning through Google map for Self-driving System.

[5] Zhang, Q., Li, M., Wang, X., & Zhang, Y. (2012). Reinforcement Learning in Robot Path Optimization. JSW, 7(3), 657-662.

[6] Jiang, B., Bishop, A. N., Anderson, B. D. O., & Drake, S. P. (2014). Path planning for minimizing detection. IFAC Proceedings Volumes, 47(3), 10200-10206.

[7] Qian Zhang, Ming Li, Xuesong Wang. Yong Zhang, "Learning in Robot Path Optimization", Journal of software, vol. 7, no. 3, March 2012.

[8] Sutton, R. S. and Barto, A. G., "Reinforcement Learning: An introduction". The MIT Press.1998.

[9] Sutton, R. S., & Barto, A. G. (2011). Reinforcement learning: An introduction.

[10] Hachour, O. (2008). Path planning of Autonomous Mobile robot. International journal of systems applications, engineering & development, 2(4), 178-190.

[11] Masehian, E., & Sedighizadeh, D. (2007). Classic and heuristic approaches in robot motion planning-a chronological review. World Academy of Science, Engineering and Technology, 23(5), 101-106.

[12] Konar, A., Chakraborty, I. G., Singh, S. J., Jain, L. C., & Nagar, A. K. (2013). A deterministic improved Q-learning for path planning of a mobile robot. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(5), 1141-1153.

ISSN: 2233-7857 IJFGCN