TEST Engineering & Management

November-December 2019
ISSN: 0193-4120 Page No. 245 – 251

An Examination on Motion Planning for Mobile Robot using Deep Q-learning with RPLidar Sensor

Manh Luong Tien[1], Yoon Young Park[*2], Se-Yeob Kim[3], Linh H.Ngo[4]

[1,2,3,4] *Department of computer engineering, Sun Moon University, Korea*
** Corresponding Author : Yoon Young Park, Email: yypark215@gmail.com*

Article Info
Volume 81
Page Number: 245 - 251
Publication Issue:
November-December 2019

Abstract : In this paper, we investigate about deep Q-learning for autonomous motion planning of mobile robot in indoor environment. Additionally, the proposed method utilizes an RPLidar sensor to deal with training issue. The deep Q-learning is adopted in this work, which is the combination between Q-learning and deep neural network. We propose a neural network model to learn the Q-function in order to solve the planning task, the proposed model is implemented in a simulation environment by using gazebo robot simulation. Our propose studies a new approach on motion planning for mobile robot. The new approach based on the deep Q-learning, which is start-of-art for robotic application. The approach does not based on a model of environment to navigation robot toward a given goal, therefore, it is independent with the scale of environment. As a result, the planning time is able to be real time to execute. Our model can directly map the range finder sensor data to a motion action such as the angular velocity, the result in section 3 points out our model well execute on a static indoor environment and successfully learn to reach a goal after some thousand training episodes. The research achieves two improvements. Firstly, the motion planning task is able to be real time planning comparing with the previous research. Secondly, the proposed model help robot reach to a given goal without the model of map.

Article History
Article Received: *3 January 2019*
Revised: *25 March 2019*
Accepted: *28 July 2019*
Publication: *22 November 2019*

Keywords: *Navigation, Obstacle avoidance, Deep Q-learning, RPLidar ,Deep neural network*

I. Introduction

Recently, the autonomous robot plays an importance role in both industry and human daily life. It is able to adopt in many applications such as logistic industry, search and rescue person, and take care of the elderly person. Therefore, the investigation of developing the autonomous system is mandatory to concrete fourth industrial revolution which is the most concerned of many governments. For accomplish the fully autonomous ability, the robot must possess the motion planning capability, that provides for robot executing by itself in order to achieve a given goal without any command from a human. In order to deal with this issue, there has been many researchers proposed many

November-December 2019
ISSN: 0193-4120 Page No. 245 – 251

approaches, nevertheless, the reinforcement learning approach achieves the success than the other approaches.

Indeed, the reinforcement learning approach benefits many advantages for an autonomous system. Firstly, the RL approach is able to facilitate robot perform reliably in an unseen environment where the robot has never either execution or training [1], [2]. This property is extremely crucial because of its reliability in unknown environment, which is difficult to achieve by the other recent methods. Secondly, the robot can perform real-time on onboard resource [3] since the RL method is model free that does not influent by the size of environment. The recent methods are based on the model based approach which creates a motion planner based on environment model. Consequently, when the scale of environment increases, the model of environment also increases that turn the model become more complicated, as a result the computation time take a long time to perform.

However, the traditional RL approach can only solve the simple problems which have limited input and output, and easily estimate the Q function from empirical data. In our research, the input data come from sensor, thus calculating Q function from empirical data is infeasible due to the various sensor data. To deal with this issue, an approach is call deep Q-learning is proposed to estimate the Q function by adopting a deep neural network. The deep neural network act as an approximate function that estimate Q function and the network is trained by temporal difference between the target network and the trainable network.

Consequently, in this paper, we develop a new approach for robot motion planning depending on the deep Q-learning. We solve the training issue for RL by using range finder sensor that is RPLidar. The training issue for RL such that the difference between the training environment and the running environment which can lead to bad performance. Subsequently, we build a deep neural network model to accomplish obstacle avoidance capability and perform well at both training environment and unknown environment. The remaining of paper is constructed as follow. In section 2, we give an overview of related work. In Section 3, the proposed method and the way to solve motion planning problem, the section 4 is the experiment and result. The last section is conclusion and future work.

II. Related work

In a decade ago, the motion planning task used to solve by using the motion planner which plans the robot action form on the environment model [4]-[6]. Those approaches plan the motion depending on a model of the environment which provides a state transition probability, and then adopt a searching algorithm to find the optimal path from the start point to destination. K. Sertac et al [4] introduced a new motion planning algorithm based on the RRT* algorithm, however, the new algorithm that is called anytime RRT* that solve the running time problem of traditional RRT* and easily converge to the optimal solution. The main idea of anytime RRT* is to execute a trajectories and generate a branch-and-bound tree in order to be adaptive in running time. The result shows that the new algorithm is faster than both RRT and RRT*.

The most traditional researches about motion planning are to focus on the static environment, however, in [5] proposed an approach to plan a robot in dynamic environment where the obstacle is not stationary. A new potential field method, is developed, which considers not only the relative position of robot respect to goal and surrounding obstacles but also the velocity of robot in order to be well suited for dynamic

environment. A virtual force is determined as a negative gradient of velocity that adjusts the velocity of robot to smoothly execute. The result shows that the new potential field is quite effective to adapt on dynamic motion planning.

Though the searching algorithm is quite successful to cope with the motion planning task, however, there is a considered issue such that the running time is too expensive, when the model of environment is extensive. Therefore, another approach is investigated that is model free for motion planning [7], [8]. The model free utilizes an approximate function to directly map the input data from sensor to the best action to execute. Lei Tai et al [7] developed a deep network for a mobile robot moving toward to a goal without the environment of model. The robot accomplishes the obstacle avoidance ability in an unknown environment through a depth camera putting rigid on top of robot. The neural network takes a raw image as input, after that predict a discrete action as the output. The result points out that the robot successfully achieves the obstacle avoidance in real world environment with the real-time planning. As a result, in this research we investigate the model-free approach for motion planning.

III. Methodology

3.1 Reinforcement learning

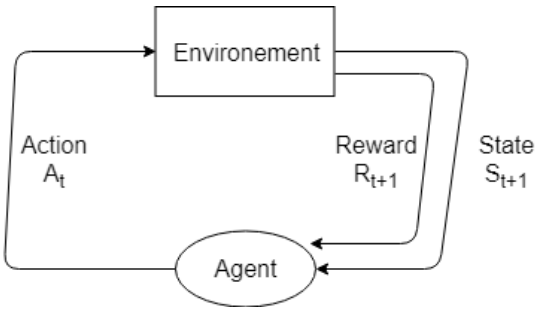

Figure 1. Reinforcement learning diagram

The reinforcement learning method is an machine learning categorize (Figure. 1), that an agent learns a policy to couple the state input s to the most appropriate output action a in an environment in order to maximize the accumulate reward function $Q(s, a)$. Typically, the Q-function is calculated from the experiment data which means the agent interacts with the environment under a random policy until the number of sample data is sufficiently huge to create a Q-table. The Q-function is determined as follow:

$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | s_t, a_t] \quad (1)$$

,where the π is the policy which the agent execute to generate data in Q-table, typically the policy is the random policy to estimate the Q value of a pair state-action. The $s_t and a_t$ are the state and action at time t, respectively. The value of Q-function is calculated by summing the total expectation reward from time step t+1 until time step T when the episode is terminated under the condition of state s_t and action a_t . γ is the discount factor which influence to future accumulate reward value, the value of γ is within the interval from 0 to 1. Hence, if the discount factor equals 0, the Q-value just takes immediate reward into account. On the other hand, if the discount factor equals 1, the Q-function takes the all future reward into account. Nevertheless, in practice, the discount reward usually is chosen $\gamma = 0.99$ to be trade-off between immediate reward and future reward.

Following the equation (1) and the Q-table, the calculating Q-value is still intractable, thus a trick is adopted to calculate Q-value that is called Bellman equation. The equation (1) can be rewritten:

$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) | s_t, a_t] \quad (2)$$

the equation (2) is able to solve by using dynamic programming. Indeed, the Q-value at

November-December 2019
ISSN: 0193-4120 Page No. 245 – 251

time step t can be computed by calculating the Q-value at time step t+1, that is the fundamental attribute of dynamic programming problem.

After achieving a Q-table and all the value of pair state-action, the agent executes a greedy policy instead of the random policy to maximize the accumulate reward.

$$\pi^{\varepsilon} = \max_a Q(s,a) \qquad (3)$$

the greedy policy always choose the action at time step t which is the maximum Q-value of a pair state-action. In case of Q-table, in each row of table, choose the highest value and then refer to the appropriate action, whenever the agent is transition to this state, it performs the chosen action to follow the greedy policy.

3.2 Proposed Method

As mentioned in previous section, the traditional RL uses a Q-function that depend on a Q-table in order to estimate the accumulate reward of pair of state-action, however, this approach is not efficient and become infeasible when the number of pair state-action is extremely huge. In this case, building a Q-table for calculating a Q-function take a long time and the numerous data sample, thus an approximate function is used to estimate the Q-function instead using Q-table. We rewrite the equation (1):

$$Q^{\pi}(s_t, a_t|\,\theta) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots |s_t, a_t] \qquad (4)$$

hereθ is the set of parameter of neural network such that $\theta = \{w_1,\ w_2,\ w_3, \dots,\ w_n\}$, and the policy is arbitrary policy because the deep Q-network is generated randomly. Adopted the Bellman equation in the equation (4), the new equation is:

$$Q^{\pi}(s_t, a_t|\,\theta) = E[R_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}|\,\theta)|s_t, a_t] \quad (5)$$

the deep Q-network is able to be trained by adopting the supervised machine learning approach. To reduce the number of pair state-action as the input for deep neural network, the input of network incorporates only the state and the output of network is a set of Q-value, each output is corresponding to a pair of input state and an action $a \in \{a_1, \dots, a_n\}$ with n is the number of action.

Based on the deep Q-learning approach, we proposed a deep neural network model to address the motion planning issue for mobile robot. The method utilizes an RPLidar sensor which is low-cost range finder sensor to collection observation data surrounding the robot. The observation data includes both the sensor data and the relative distance between robot and the destination. The purpose of deep Q-learning is to make a decision control in real time and avoid the obstacle while moving or exploring the environment. In order to accomplish those goals, the neural network is trained to be satisfied the equation:

$$Q^{\pi}(s_t, a_t|\,\theta) = \max \pi E[t' = tT\gamma t' - tRt'|st, at] \quad (6)$$

, where s_t and a are the state of robot from the RPLidar sensor and the action at the time t, respectively. The action a is chosen from a deep neural network, in which s_t is input and θ_i are the parameters of the neural network. Subsequently, γ is the discount factor for future reward, and $r_{t'}$ is the intermediate reward at time t' after performing an action a.

In our approach, the discount factor is $\gamma = 0.99$ as many RL practices. Since our purpose is motion planning, thus we keep the velocity of robot is stable $\vartheta = 0.3m/s$ in order to simplify the robot action. Therefore, for action decision is the angle of the robot $\varphi = [0, \pi]$, the angle is from 0 to π with the interval $\pi/2$, as a result, the robot action possesses 5 actions

November-December 2019
ISSN: 0193-4120 Page No. 245 – 251

corresponding to 5 angles of robot's head. We define an immediate reward function for each pair of state-action that is:

$$r(s,a) = \begin{cases} 1\ if\ d_{curr} > d_{goal} \\ 0\ if\ d_{curr} < d_{goal} \\ 200\ if\ d_{goal} < 0.02 \\ -200\ if a collision happen \end{cases} \quad (7)$$

where $r(s,a)$ is the immediate reward the robot receive when the robot is at the state s performing action a. d_{goal} is the distance between the robot and the goal after executing an action, and d_{curr} is the distance between the robot and the goal before executing an action. As can be seen from the equation (7), the robot will receive the highest reward if it reaches to the goal. On the other hand, if it hits to any obstacle, it will take a negative reward to punish those actions leading a collision. The robot will receive a little bit reward, if it moves close to the goal.

In order to learn the Q-function, we operate a deep neural network (as shown in Figure 2.) to estimate the Q-function, the input of network is the range finder data from RPLidar $s_t = \{\omega_i | 0 \leq i \leq 180\}$, where ω_i is the distance from robot to obstacle corresponding angle i. The deep Q-network is constructed by an input layer where the rangefinder data is fed into, subsequently, two full-connected hidden layers with ReLU activation function is operated to learn the pattern from input layer, each hiddenlayer own 124 neural units. Finally, the output layer own 18 neural units, correspond to 18 actions in

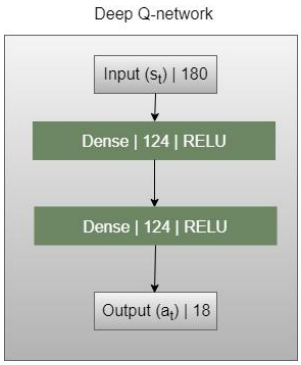

Figure 2. The proposed deep neural network for motion planning

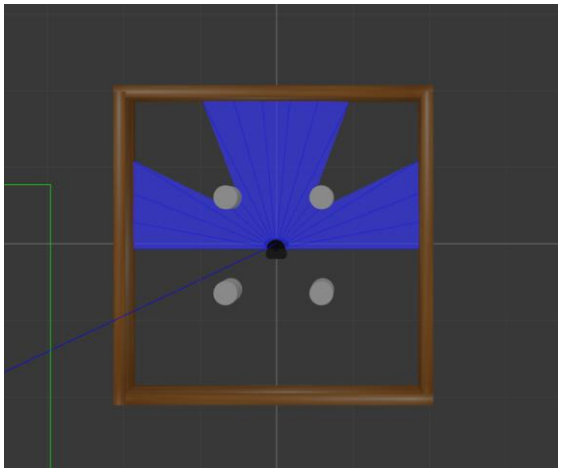

Figure 3. The simulation environment

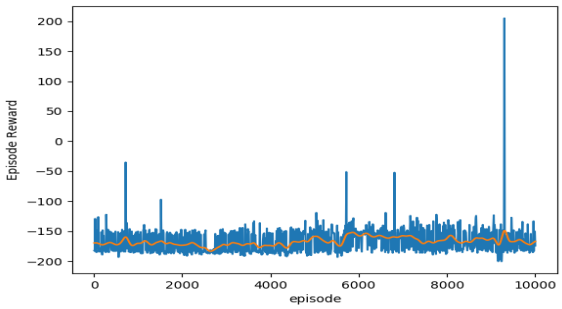

Figure 4. The accumulated reward of robot

$\varphi = [0, \pi]$. The suitable action for a state s_t is derived from neural network by forward propagation process, and the action which has the highest value from output layer is picked up in order to send to robot.

On the other hand, we use the experience replay buffer and a target network [4] to train our deep neural network. The target network has the similar network parameter with the

trainable network at initialization time, and then updating the target network parameter as follow:

$$\theta^{'} = \tau\theta^{'} + (1-\tau)\theta (8)$$

, where $\theta^{'}$ is the target network parameter, θ is the trainable network parameter, and $\tau \ll 1$.

Lastly, for training the deep neural network the trainable network is trained by using the temporal difference between the trainable network and the target network. The Q-network is trained based on the mean square error (MSE) as follow:

$$L_j(\theta_i) = E_{s,a,r,s'}[\left(y_j - Q(s,a;\theta)\right)^2] (9)$$

, where $y_j = r(s,a) + \max_{a'} Q(s',a';\theta^{'})$ with $Q(s',a';\theta^{'})$ is the prediction from Q target-network, action $a^{'}$ is an action that maximize the Q-value at the state s'. $Q(s,a;\theta_i)$is the Q-value of predecessor state s from Q-network.

IV. Result and Discussion

For the experiment, we use the turtlebot2 simulation [10] to implement the neural network model for motion planning. The simulation runs on ROS and gazebo simulator, which provides a bunch of tool box for testing the robotic system. The environment is set up such that the robot executes in a small square room with four cylinders as four state obstacles in Figure. 3. The robot is equipped a RPLIDAR sensor on its top and the sensing range of RPLIDAR sensor is between 0.12 meter to 3.5 meters. The robot plan to move toward a goal which is determined as a small red square, the goal is randomly located after the robot reaching the goal. To easily train the robot, we determine the episode, which is the time from robot initialization until it hits into an obstacle or after 200 move steps .

The parameter of neural network is chosen based on the empirical implementation such that learning rate $l = 0.0001$, the parameter of each layer is generated depending on [11], the parameter is randomly generated from a normalize distribution $N(0, \frac{2}{fan\_in})$, with fan_in is the number of input hidden unit of predecessor hidden layer. The batch_size for training is 32 data samples for each gradient descent execution. The planning time for an execution is approximate 300ms, that is well suited for robotic application in real world. As shown in Figure. 4, the chart shows the accumulated reward of robot during along 10000 episodes, from the first episode until the 3000th episode the reward is quite low due to the robot early hitting the obstacle. .However, from the 6000th episode the reward line cliff dramatically because the robot learned a policy to move toward goal and avoid the static obstacle.

V. Conclusion

In this work, we study a model free approach for motion planning for mobile robot. The proposed method adopts the deep reinforcement learning to solve the collision issue in indoor environment while planning the motion for the robot to reach a given goal. The experiment shows the method is able to be real-time planning which is crucial for many robot applications. Additionally, the robot is able to reach the goal after particular episodes. However, the method still has some drawbacks as the training time is too long and the model is offline which cannot improve during execution time. In the future work, we will investigate more to solve all the drawbacks and develop a navigation system to adopt on a real robot.

VI. Acknowledgment

This work was supported by National Research Foundation of Korea Grant Funded by the

Korean Government (NRF2016R1A2B4014223).

References

[1] L. Tai, G. Paolo and M. Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. The IEEE International Conference on Intelligent Robots and Systems (IROS), 31-36, 2017.

[2] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, C. Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. The IEEE International Conference on Robotics and Automation (ICRA), 1527-1533, 2017.

[3] A. Nagabandi, G. Kahn, R.S. Fearing, S. Levine. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. The IEEE International Conference on Robotics and Automation (ICRA), 7559-7566, 2018.

[4] S. Karaman, M.R. Walter, A. Perez, E. Frazzoli, S.J. Teller. Anytime Motion Planning using the RRT*. The IEEE International Conference on Robotics and Automation, 1478-1483, 2011.

[5] S.S. Ge, Y. Cui. Dynamic Motion Planning for Mobile Robots Using Potential Field Method. Auton. Robots, 207-222, 2002.

[6] J.A. Wolfe, B. Marthi, S.J Russell. Combined Task and Motion Planning for Mobile Manipulation. ICAPS, 254-257, 2010 May.

[7] L. Tai, S. Li, M. Liu. A deep-network solution towards model-less obstacle avoidance. The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2759-2764, 2016.

[8] M. Everett, Y.F. Chen, J.P. How. Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 3052-3059, 2018.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M.A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis. Human-level control through deep reinforcement learning. Nature, 518, 529-533, 2015.

[10] http://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#simulation

[11] K. He, X. Zhang, S. Ren, J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. The IEEE International Conference on Computer Vision (ICCV), 1026-1034, 2015.