2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM) | 978-1-6654-2094-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRoM54204.2021.9663449

Motion Planning of Autonomous Vehicles Using Obstacle-Free Q-Learning Path Generator and Model Predictive Control

Roozbeh Bazargani
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
roozbeh@aut.ac.ir

Mahboobe Shakeri Nadrabadi
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
m.shakeri@aut.ac.ir

Elnaz Firouzmand
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
efsh@aut.ac.ir

Iman Sharifi
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
imansharifi@aut.ac.ir

Heidar Ali Talebi
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
alit@aut.ac.ir

***Abstract*—One of the key concerns in developing autonomous vehicles is guaranteeing safety. This paper focuses on designing a safe and real-time path with the help of reinforcement learning, specifically Q-learning as a path generator. Furthermore, Model Predictive Control (MPC) is exploited to implement the generated path by considering the model of the vehicle and corresponding constraints. The path obtained by the Q-learning algorithm is optimal and obstacle-free. Moreover, the stability and Recursive Feasibility (RF) of the MPC is investigated by using terminal cost and constraints. In the proposed strategy, the non-convexity induced by obstacles in the environment is handled in the Q-learning-based path generator. Moreover, illustrative case studies will demonstrate the efficiency of the proposed motion planning algorithm in the field of autonomous vehicles.**

***Index Terms*—Q-learning, Model Predictive Control, Path Generator, Obstacle-Free, Reinforcement Learning, Autonomous Vehicles, Recursive Feasibility**

I. Introduction

In recent years, autonomous vehicles have been widely applied for various applications, from smart transportation to industrial automation. Therefore, motion planning and tracking of an autonomous vehicle is an attractive research area in robotics and control theory field [1]. Motion planning is the generation of a feasible and obstacle-free path for a vehicle to reach a desired target point in the planning workspace [2].

Generally, a motion planning module should consist of both a high-level path planner and a low-level controller to implement the appropriate commands. There is huge research in the literature on path planning algorithms. To name a few, Dijkstra [3], A^* [4] and RRT [5] are widely used for path planning. Furthermore, Artificial Intelligence (AI) can be used as a path planning algorithm for its inherent intelligence in designing actions in unknown environments. One of the famous methods for path planning in AI is reinforcement learning. Q-learning is a model free reinforcement learning, and there have been implementations where a combination of deep neural networks and Q-learning, in the name of Deep Q-networks, were able to drive a car [6].

Model Predictive Control (MPC) is an optimal control strategy based on numerical optimization, which widely used for constrained MIMO systems over the past decade [7]. MPC generates an optimal control input sequence over a receding horizon by solving an optimization problem at every sampling time. Afterward, only the first element of the optimal sequence is applied to the system. MPC might be used for implementing different scenarios in autonomous vehicles, such as trajectory tracking [8] or motion planning [9]. The feasibility and existence of a solution for MPC algorithm is based on the convexity of the planning domain, which would be lost by considering obstacles. On the other hand, the problem of obstacle avoidance is considered in the lowest level of motion planning hierarchy by introducing potential fields [10] or control barrier functions [11]. Most of these methods are prone to lost RF.

In this paper, we propose a novel motion planning algorithm in which an obstacle-free path is planned online, and MPC is responsible for implementing low-level commands for an autonomous vehicle. The benefit of our proposed approach is twofold; in AI-based path generation, lack of stability is a major issue since generalization of the learned states to all the possible states is not proved. As a result, a combination of AI and control engineering such as reinforcement learning and MPC has been used in recent years to reach more stable methods for autonomous vehicles [12] [13]. On the other hand, by incorporating the intelligence of Q-learning in generating the obstacle-free path, the non-convexity in solving numerical optimization problems in MPC in the presence of obstacles

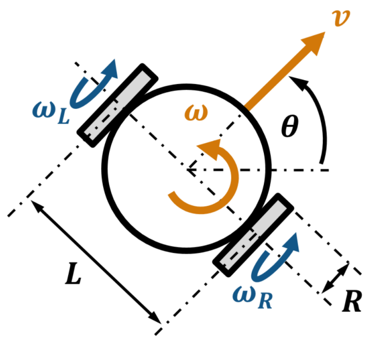

Fig. 1. Differential vehicle [15]

is relieved. Furthermore, we use the idea of adding terminal cost and constraint for guaranteeing stability and RF. We also enlarged the obstacles by considering neighbour states as obstacles. Despite treating these obstacles as real ones, virtual obstacles were defined that have less punishment in comparison to real obstacles in order to pass narrow passages in case of having no other path, or the other path is too costly and long.

The organization of the remainder of the paper is as follows. The modelling of a vehicle is presented in Section II. The overall scheme of the proposed method consisted of the Q-learning path generator, and the design of the model predictive controller is introduced in Section III. Simulation results are reported in Section IV. Finally, the conclusion and future works are given in Section V.

II. Vehicle Model

The differential model is used to represent the vehicle kinematics as illustrated in Fig. 1. Derivation of its kinematics is completely covered in [14]. Hence, the kinematic of the mobile robot is

$$\omega_L = \frac{1}{R}(v - \frac{\omega L}{2}), \quad \omega_R = \frac{1}{R}(v + \frac{\omega L}{2}) \tag{1}$$

where R is wheel radius, L is the distance between the wheels, v is the linear velocity of the robot, ω is the angular velocity, and ω_L and ω_R are the speed of left and right wheel, respectively.

To control the vehicle, the dynamic of the model in state space is acquired. First, we show the equations for $\dot{x}_v$, $\dot{y}_v$, and $\dot{\theta}_v$ as [16]

$$\begin{cases} \dot{x}_v = v\cos\theta_v \\ \dot{y}_v = v\sin\theta_v \\ \dot{\theta}_v = \omega \end{cases} \tag{2}$$

Considering $x = [x_v, y_v, \theta_v]^T$ (states) and $u = [v, \omega]^T$ (control input) near reference point as (x_r, u_r), the overall discretized system is [16]

$$\tilde{x}(k+1) = A(k)\tilde{x}(k) + B(k)\tilde{u}(k) \tag{3}$$

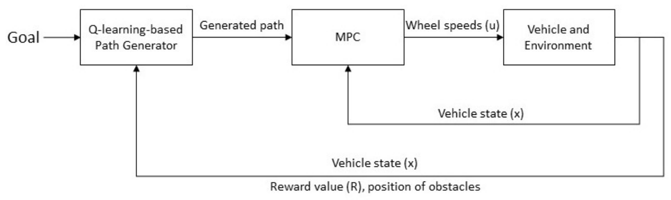

Fig. 2. Control Methodology flowchart

where $\tilde{x} \triangleq x - x_r$ and $\tilde{u} \triangleq u - u_r$ with

$$A(k) \triangleq \begin{bmatrix} 1 & 0 & -v_r(k)\sin\theta_r(k)T \\ 0 & 1 & v_r(k)\cos\theta_r(k)T \\ 0 & 0 & 1 \end{bmatrix}$$

$$B(k) \triangleq \begin{bmatrix} \cos\theta_r(k)T & 0 \\ \sin\theta_r(k)T & 0 \\ 0 & T \end{bmatrix}$$

In the next section, control methodology is discussed.

III. Control Methodology

The overall methodology is that by using a path generator, in our case Q-learning, the generated obstacle-free reference path will be passed to MPC, and the MPC will control the vehicle in order to follow the path. The Q-learning algorithm learns and provides the optimal path online and in real-time, which is its greatest advantage. In addition to the vehicle state (x), it also requires position of obstacles and reward function from the environment. It is worth mentioning, with the correct selection of reward function and definition of virtual obstacles, the algorithm was able to create a non-zigzag path and plan a safer path by passing obstacles at a logically farther distance from them. The control diagram can be seen in Fig. 2.

Now, each section of the controller will be discussed.

A. Path Generator: Q-learning

Q-learning is a model free reinforcement learning in which a robot tries to learn a policy in order to gain as much reward as possible. There is a dynamic method for Q-learning that has been used in this paper with some modifications due to our deterministic environment [17]. Moreover, choosing the best reward function for computing reward for each state and action is vital to get our desired performance from the Q-learning model.

Firstly, states and actions are defined. Every integer pair in the space, considered as an state ($S = \{(x,\ y) | x, y \in \mathbb{Z}\}$). Actions for transferring a vehicle from one state to the other are defined as

$$A = \{a \mid a \in \{[0\ 1]^T, [0\ \ -1]^T, [1\ 0]^T, [-1\ 0]^T, [1\ 1]^T, [-1\ 1]^T, [1\ \ -1]^T, [-1\ \ -1]^T\}\} \tag{4}$$

After exactly defining states and actions, we discuss the general formulas for implementing a Q-learning model [17]. If we define a probability of transferring to state y from state

x under policy of $\pi(x) \in A$ as $P_{xy}[\pi(x)]$, the value of the state x would be

$$V^{\pi}(x) \equiv \mathcal{R}_x(\pi(x)) + \gamma \sum_y P_{xy}[\pi(x)]V^{\pi}(y) \tag{5}$$

where γ is the discount factor, and $\mathcal{R}_x(\pi(x))$ is the immediate reward after acting on policy $\pi(x)$ on the state x. Therefore, the state values under the optimal policy is

$$V^{\pi^*}(x) = \max_a \{ \ \mathcal{R}_x(\pi(x)) + \gamma \sum_y P_{xy}[\pi(x)]V^{\pi^*}(y) \ \} \tag{6}$$

Since our environment is deterministic, meaning the next state (s') for every state and action pair (s, a) is determined, therefore we do not need a probability for showing the next state since $P_{s's}(\pi(s)) = 1$ and for each other state s", $P_{s"s}(\pi(s)) = 0$. As a result, our equations will be as

Equation (5) will change to

$$V^{\pi}(x) \equiv \mathcal{R}_x(\pi(x)) + \gamma V^{\pi}(y) \tag{7}$$

and Eq. (6) will become

$$V^*(x) \equiv V^{\pi^*}(x) = \max_a \{\mathcal{R}_x(a) + \gamma V^{\pi^*}(y)\} \tag{8}$$

Q-function evaluate the value of each state and action pair [17].

$$Q^{\pi}(x,\ a) = \mathcal{R}_x(a) + \gamma V^{\pi}(y) \tag{9}$$

Bellman equation, Eq. (10), is used to update Q-function [17].

$$Q_n(x,\ a) = (1-\alpha_n)Q_{n-1}(x,\ a) + \alpha_n[r_n + \gamma V_{n-1}(y_n)] \tag{10}$$

where α_n, and r_n are the learning rate, and reward at the step n, respectively.

After updating the Q-function, value of every state is computed by considering to choose the action that its pair with the state has the maximum value. Moreover, that action (b) becomes the optimal policy of that state.

$$V_{n-1}(y) \equiv \max_b \{Q_{n-1}(y,\ b)\} \tag{11}$$

In order to update, every interval, all the states are updated in a loop until the maximum change is less than a constant ϵ.

The next step is to define the reward function in a way that satisfies our expectations from the model. The goal of our Q-learning model is to minimize the travelled distance as well as travel duration between our start and end states. Having interval for recomputing the optimal policy, one can consider reward function as $\mathcal{R} = -1$ to minimize the duration of the travel. However, as Fig. 3 illustrates, this reward function can result in a zigzag path planning. The source of problem is that the robot does not see a difference between horizontal and vertical path, and diagonal path which has longer distance. Consequently, we have come up with a new reward function that also considers norm of proposed path for each action $a \in A$.

$$\mathcal{R} = -\|a\|_2 \tag{12}$$

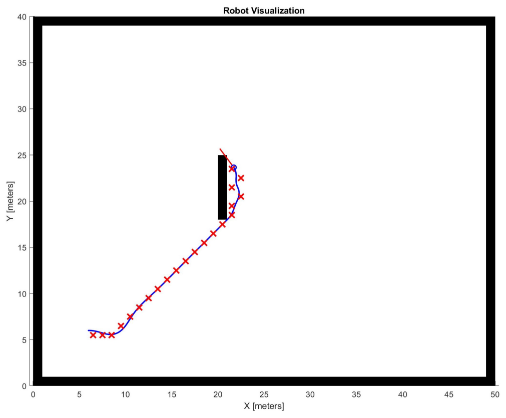

Fig. 3. Travelled path by Q-learning with $\mathcal{R} = -1$

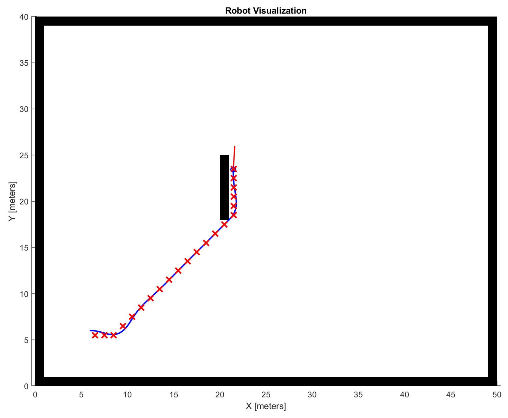

Fig. 4. Travelled path by Q-learning with $\mathcal{R} = -\|a\|_2$

It is worth mentioning that our reward function is independent of the state the vehicle belongs to it resulting in a simple, effective and low computation cost implementation. The robot planned and travelled path according to Eq. (12) are shown in Fig. 4. As we see, the proposed reward function results in a smooth straight path as it is desired.

In order to avoid obstacles, two types of obstacles for states are defined, real obstacles and virtual obstacles which are neighbour states to the real obstacles. On the one hand, considering only real obstacles may lead the vehicle crashing into the obstacles. On the other hand, treating neighbour states as the real obstacles may cause the vehicle not to consider any difference between the real object and the states around it. Therefore, by having virtual obstacles, we consider different penalties for real obstacles and virtual ones in order to be capable of passing narrow passages between the real obstacles. Finally, the reward function will define as

$$\mathcal{R}(s, a, s') = \begin{cases} r_r & s' \in \text{Real obstacles} \\ r_v & s' \in \text{Virtual obstacles} \\ -\|a\|_2 & \text{otherwise} \end{cases} \tag{13}$$

There are some tips to consider for choosing the value of the parameters.

Remark 1: r_r should be defined as the smallest number possible ($-\infty$) that depends on the system and size of bytes that has been used for a number due to overflows.

Remark 2: r_v should be defined small enough that in the edges, Q-learning decides to have a safer distance from obstacles. However, it should not be too negative since sometimes we want to go through narrow passages in order to not travel a very long path. It is important to note that smaller r_v prefer a safer approach and, thus, that other paths should be longer, so algorithm policy changes to passing a narrow passage.

Remark 3: A higher learning rate results in a faster adaptation to the changes in the environment and learning faster. However, choosing a large learning rate might result in instability of the model.

Remark 4: The discount factor parameter γ affects the rate of forgetting the importance of a reward taken n steps before by reducing its coefficient by γ^n [18].
Therefore, in order to be adaptable to sudden changes in the environment, we have chosen $\alpha_n = 0.9$ and $\gamma = 0.9$.

Now, we focus on implementing MPC to control the location of the vehicle by right and left wheel speeds in order to follow the generated path by Q-learning.

B. Controller: Model Predictive Control

Here, we use MPC to implement the control commands for the vehicle to follow the given path. Therefore, the goal is to minimize the tracking error with respect to the desired reference points of the path generated by the Q-learning algorithm.

As we discussed in Section I, in MPC algorithm, a numerical optimization problem is solved over a prediction horizon to minimize the desired cost function while satisfying a set of constraints. Moreover, only the first element of the obtained optimal control sequence is implemented at each sampling time; and this process is then repeated for the subsequent times. There are two aspects to consider while designing MPC: stability and RF. To ensure stability, one might use a terminal cost to compensate the effect of choosing a finite prediction horizon [19]. According to RF, feasibility of the optimization problem in terms of the satisfaction of constraints at each sampling time must be guaranteed [20]. One of the standard approaches to ensure RF is by introducing terminal cost and terminal constraint [19], [21]. In the following, we use the idea of introducing a terminal cost and constraint to ensure stability and RF, which is mostly not considered in the literature of AI-based motion planning algorithms.

Consider the vehicle with dynamical model as described by Eq. (2) is subject to state and input constraints

$$x \in \mathcal{X}, u \in \mathcal{U} \tag{14}$$

where $\mathcal{U} \in \mathbb{R}^m$ is compact, and $\mathcal{X} \in \mathbb{R}^n$ is closed.

The path generator plans the path online by generating a set of reference points. To implement MPC, we discretize and linearize Eq. (2) near each of the reference points. Therefore, the new set of state and input constraints $\tilde{\mathcal{X}}_r$ and $\tilde{\mathcal{U}}_r$ is resulted as

$$\begin{aligned} \tilde{\mathcal{X}}_r &= \mathcal{X} \oplus (-x_r) \\ \tilde{\mathcal{U}}_r &= \mathcal{U} \oplus (-u_r) \end{aligned} \tag{15}$$

which contains origin in interior, and $\oplus$ is Minkowski sum.

Consider $\tilde{u}_{i|k}$ and $\tilde{x}_{i|k}$ as the error input and state vectors at time i which are predicted at the sampling time k. Let N be the prediction horizon, then the constrained MPC problem for reaching reference points of the generated path is formulated as

$$\begin{aligned} &\min_{\tilde{\mathbf{u}}} \; J(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \\ \text{s.t. } &\tilde{x}_{i+1|k} = A_r \tilde{x}_{i|k} + B_r \tilde{u}_{i|k} \\ &\tilde{x}_{i|k} \in \tilde{\mathcal{X}}_r, \; \tilde{u}_{i|k} \in \tilde{\mathcal{U}}_r, \quad \forall k \in \mathcal{L}_{N-1} \\ &\tilde{x}_{N|k} \in \tilde{\mathcal{X}}_{r,f} \end{aligned} \tag{16}$$

where

$$J(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = \sum_{i \in \mathcal{L}_{N-1}} l(\tilde{x}_{i|k}, \tilde{u}_{i|k}) + J_{r,f}(\tilde{x}_{N|k}) \tag{17}$$

$$l(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = (1/2)(\tilde{\mathbf{x}}^T Q \tilde{\mathbf{x}} + \tilde{\mathbf{u}}^T R \tilde{\mathbf{u}}) \tag{18}$$

$$J_{r,f}(\tilde{\mathbf{x}}) = (1/2)\tilde{\mathbf{x}}^T P \tilde{\mathbf{x}} \tag{19}$$

in which, $\tilde{\mathbf{x}} = [\tilde{x}_{0|k}, x_{1|k}, \ldots, x_{N-1|k}]^T$ is the augmented state, and $\tilde{\mathbf{u}} = [\tilde{u}_{0|k}, u_{1|k}, \ldots, u_{N-1|k}]^T$ is the augmented input vector over a prediction horizon. Moreover, A_r and B_r are the state and input matrices computed as in Eq. (3) for every reference point along the desired path, and $\mathcal{L}_{N-1} = \{0, 1, \ldots, N-1\}$ determines the sequence of sampling times over a horizon. Furthermore, $\tilde{\mathcal{X}}_{r,f}$ denotes the terminal constraint, and $J(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ is the cost function of the embedded numerical optimization problem, where $l(\tilde{x}_{i|k}, \tilde{u}_{i|k})$ and $J_{r,f}(\tilde{x}_{N|k})$ are the stage and final cost, respectively. In addition, Q and P are weighting matrices for states, which must be positive semi-definite, and R is a design positive definite matrix corresponding to inputs.

Suppose $K \in \mathbb{R}^{m \times n}$ is such that $A_{cl} \triangleq A_r + B_r K$ is stable. Let $\mathcal{Z}$ be an invariant set for error; i.e. $\tilde{x} \in \mathcal{Z}$, satisfying $\tilde{x}^+ = A_{cl}\tilde{x}$ [19]. Therefore

$$A_k \mathcal{Z} \subseteq \mathcal{Z}. \tag{20}$$

Now, we present the theorem for ensuring stability and RF of MPC problem formulated in (16).

Theorem: [22] Consider the dynamical model of a vehicle as Eq. (2), and the corresponding error dynamic modeled by Eq. (3). Let us define $\mathcal{Z}$ to be an invariant set as defined in Eq. (20). According to Eq.(15), if $x_v(0) \in x_r \oplus \tilde{\mathcal{X}}_r$, and $u_v = \tilde{u}^o + u_r$, then the MPC problem formulated as (16) is asymptotically stable and the RF is guaranteed, if for $i \in \mathcal{L}_{N-1}$, the following axioms are held

1) $A_{cl}\tilde{\mathcal{X}}_{r,f} \subset \tilde{\mathcal{X}}_{r,f}, \; \tilde{\mathcal{X}}_{r,f} \subset \mathcal{Z}, \; K\tilde{\mathcal{X}}_{r,f} \subset \tilde{\mathcal{U}}_r$
2) $J_{r,f}(A_{cl}\tilde{x}_{i|k}) + l(\tilde{x}_{i|k}, K\tilde{x}_{i|k}) \leq J_{r,f}(\tilde{x}), \;\; \forall \tilde{x}_{i|k} \in \tilde{\mathcal{X}}_{r,f}$

where $\tilde{u}^o$ is the first element of the optimal control sequence obtained from Eq. (16), and $\tilde{\mathcal{X}}_{r,f}$ denotes the maximal positive invariant set computed from Eq. (20) [19].

Proof 1: The proof of the statements is straightforward based on the result of [22], [23].

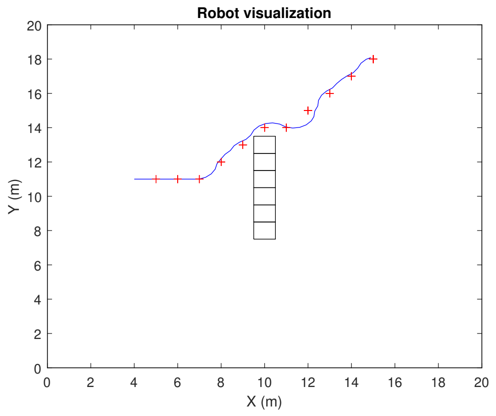

Fig. 5. Travelled path by only using real objects

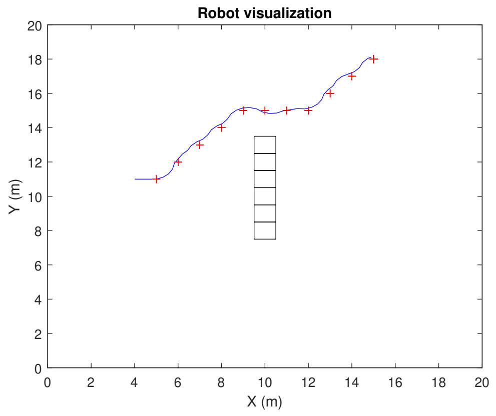

Fig. 6. Travelled path by using both real and virtual objects

IV. Result and Comparison

First, we declare the value of the parameters and then present and discuss the results.

A. Parameter Values

For the Q-learning values of $\alpha_n = 0.9$ and $\gamma = 0.9$ were chosen as has been already discussed in Sec. III-A. Moreover, the reward function was defined as Eq. (13) with $r_r = -10000$, and $r_v = -5$. Wheel radius(R) and wheel base(L) of vehicle are set as $0.05m$ and $0.2m$ respectively. Q and R matrices for MPC cost function are chosen as follow:

$$Q = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 0.8 \end{bmatrix}, \quad R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Prediction horizon $N = 15$, and sampling time is $T_s = 0.1$. The space S is considered as

$$S = \{(x, y) \mid 0 \le x, y \le 20\} \tag{21}$$

and we considered obstacles as unit squares with centers of Obs which is defined differently in the following scenarios.

In the next two sections, the importance of virtual obstacles and a comparison between virtual obstacles and considering real obstacles instead of them are experimented using the above values for the parameters.

B. Necessity of virtual obstacles

Real obstacles in this example are defined as

$$Obs = \{(x, y) \mid x = 10 \ \wedge \ y \in \{8, 9, ..., 13\}\} \tag{22}$$

and initial condition for state vector is $x = [4, 11, 0]^T$, and target is $x = [15, 18, 0]^T$.

The result of travelled path by vehicle with only considering real obstacles is demonstrated in the Fig. 5. It can be seen that without considering neighbour states of the real obstacle, the vehicle has passed very closely to them. However, with the help of virtual obstacles, the vehicle has travelled between the initial and goal states with a more safe distance from obstacles in Fig. 6.

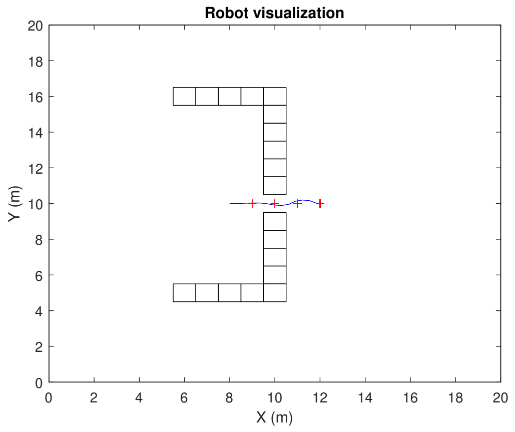

Fig. 7. Travelled path by adding virtual objects

C. Comparison of virtual with real obstacles

The location of real obstacles in this example is considered as the following

$$\begin{aligned} Obs = \{(x, y) \mid (x = 10 \ \wedge \ y \in \{3, 4, ..., 16\} - \{10\}) \\ \vee \ (y = \{3, 16\} \ \wedge \ x \in \{6, 7, ..., 10\})\} \end{aligned} \tag{23}$$

where its neighbour states once considered as virtual obstacles, and once as real ones. The vehicle starts from $x = [8, 10, 0]^T$ and the goal is to reach $x = [12, 10, 0]^T$. The actual distance between the two points is 4 and can be travelled by a narrow path crossing $(10, 10) \in S$. In the proposed method, neighbour states such as $(10, 10) \in S$ are considered as virtual obstacles, and Q-learning algorithm will decide whether to pass the virtual obstacle. Therefore, as it is illustrated in Fig. 7, the result is optimal by passing the narrow path. Although the other approach that is considering neighbour states same as real obstacles will result in a safer path, it can end up in a very long path, as it can be seen in Fig. 8. It is worth remembering, as in Remark 2, the coefficient r_v needs to be tuned in order to prefer a narrow path over a safe path with distance d.

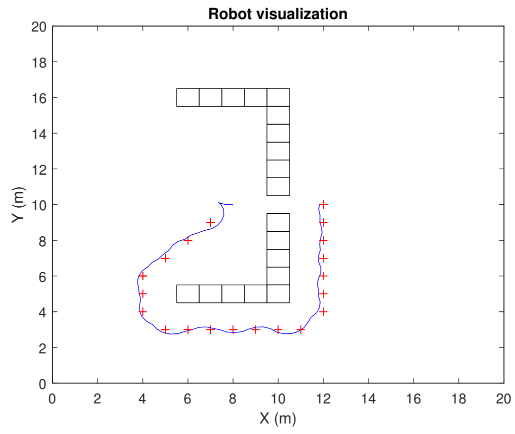

Fig. 8. Travelled path by adding real objects instead of virtual ones

V. Conclusion

In conclusion, the path planning method using obstacle-free Q-learning path generator and MPC was successful in both stability and optimality. Moreover, the implementation was in real-time, which is essential to control autonomous vehicles. Since considering states as an obstacle, the edges of the objects were far from the states where real obstacles are. Thus, neighbour states of the real obstacles were considered as virtual obstacles in which they have punishment but not as severe as the real obstacles. The importance of choosing reward function in order to have a smooth and straight path was discussed by showing the difference between $\mathcal{R} = -1$, and $\mathcal{R} = -\|a\|_2$. Finally, the optimization problem of MPC was always feasible and solved fast by the help path generator method that excluded all obstacle constraints and made the linear optimization problem simple and convex.

There are also some items that need attention and could be ideas for the future works:

- Some other options for making a reward function is to consider magnitude of jerk in order to have less jerk. As a result less pressure on the vehicle travelers.
- Defining a method to determine the value of punishment for virtual obstacles to prefer the narrow path over a path with distance d, by making a relation between d, and r_v since tuning could be time-consuming and needs a lot of experiments.
- Choosing the state locations with more intelligence or randomly in order to reduce the number of the states in the environment and adding more states in the critical sections such as near obstacles.

References

[1] Q. Yao, Y. Tian, Q. Wang, and S. Wang, “Control strategies on path tracking for autonomous vehicle: State of the art and future challenges,” *IEEE Access*, vol. 8, pp. 161 211–161 222, 2020.
[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.
[3] H. Wang, Y. Yu, and Q. Yuan, “Application of dijkstra algorithm in robot path-planning,” in *2011 second international conference on mechanic automation and control engineering*. IEEE, 2011, pp. 1067–1069.
[4] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, “Path planning with modified a star algorithm for a mobile robot,” *Procedia Engineering*, vol. 96, pp. 59–69, 2014.
[5] S. M. LaValle *et al.*, “Rapidly-exploring random trees: A new tool for path planning,” 1998.
[6] P. Wolf, C. Hubschneider, M. Weber, A. Bauer, J. Härtl, F. Dürr, and J. M. Zöllner, “Learning how to drive in a real world simulation with deep q-networks,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 244–250.
[7] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.
[8] F. Künhe, J. Gomes, and W. Fetter, “Mobile robot trajectory tracking using model predictive control,” in *II IEEE latin-american robotics symposium*, vol. 51. Citeseer, 2005.
[9] H. Guo, C. Shen, H. Zhang, H. Chen, and R. Jia, “Simultaneous trajectory planning and tracking using an mpc method for cyber-physical systems: A case study of obstacle avoidance for an intelligent vehicle,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4273–4283, 2018.
[10] K. Shibata, N. Shibata, K. Nonaka, and K. Sekiguchi, “Model predictive obstacle avoidance control for vehicles with automatic velocity suppression using artificial potential field,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 313–318, 2018.
[11] U. Rosolia and A. D. Ames, “Multi-rate control design leveraging control barrier functions and model predictive control policies,” *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1007–1012, 2020.
[12] Á. Fehér, S. Aradi, and T. Bécsi, “Hierarchical evasive path planning using reinforcement learning and model predictive control,” *IEEE Access*, vol. 8, pp. 187 470–187 482, 2020.
[13] R. Bazargani, “Implementing iot based path planning and platooning control of autonomous vehicles,” 2021.
[14] F. A. Salem, “Dynamic and kinematic models and control for differential drive mobile robots,” *International Journal of Current Engineering and Technology*, vol. 3, no. 2, pp. 253–263, 2013.
[15] M. S. C. T. (2021), “Mobile robotics simulation toolbox (https://github.com/mathworks-robotics/mobile-robotics-simulation-toolbox),” *GitHub*, August 4, 2021.
[16] W. F. Lages and J. A. V. Alves, “Real-time control of a mobile robot using linearized model predictive control,” *IFAC Proceedings Volumes*, vol. 39, no. 16, pp. 968–973, 2006.
[17] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[18] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.
[19] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.
[20] P. D. Christofides, R. Scattolini, D. M. de la Pena, and J. Liu, “Distributed model predictive control: A tutorial review and future research directions,” *Computers & Chemical Engineering*, vol. 51, pp. 21–41, 2013.
[21] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
[22] D. Q. Mayne, M. M. Seron, and S. Raković, “Robust model predictive control of constrained linear systems with bounded disturbances,” *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.
[23] S. V. Raković and W. S. Levine, *Handbook of model predictive control*. Springer, 2018.