2023 IEEE International Opportunity Research Scholars Symposium (ORSS) | 979-8-3503-3700-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ORSS58323.2023.10161975

Energy Minimization in Overtaking for Autonomous Vehicles in a Bidirectional Environment

Joshua Chio[1], Daniel Geng[1], Lydia Han[1], Meghna Jain[1], Mi Zhou[1], Erik I. Verriest[1]

***Abstract*—In this article, we formulate an overtaking problem in a bidirectional dynamic highway environment. A Deep Deterministic Policy Gradient (DDPG)-based reinforcement learning method is used to learn an optimal policy for the Ego car with a customized environment. Moreover, the classical optimal control method is applied to solve a similar optimal control problem with two time-varying constraints. Simulations are provided to verify the performance of DDPG. The optimal policy obtained by the classical optimal control method is then used as a compare benchmark for the learning-based method. Model predictive path integral control is finally employed to handle a more dynamic environment and possible different driving modes of cars.**

***Index Terms*—autonomous cars, energy minimization, overtaking, reinforcement learning, optimal control**

I. Introduction

In the past decade, ample research has been conducted on autonomous cars. Most research problems focus on improving the "vision" ability of cars by improving the sensor and the computer vision algorithm. Motion planning algorithms, such as model predictive control and rapidly-exploring Random Tree (RRT) algorithm, are combined with this kind of machine vision to realize the aim of local optimal path routing [1]. Despite the ample research data, the study on the self-driving car targeting human-like behavior for cars, such as overtaking and yielding, is still lacking. However, in the real world, it is in these overtaking behaviors that the greatest risk factors are introduced. These risk factors call for a further understanding of related problems. Thus, in this study, we consider a car overtaking problem and focus on finding an optimal overtaking policy for an autonomous vehicle on a bidirectional road with incoming traffic.

Previous studies have been conducted on overtaking maneuvers under various scenarios and using different methods, some of them include model predictive control (MPC) approach [2], stochastic model predictive control (SMPC) [3], [4], and reinforcement learning [5]. For example, [2] explored an overtaking problem on a bidirectional lane with the simplified car model. The authors then used the MPC method to maximize velocity and minimize time spent in the adjacent lane; they also strove to decrease abrupt changes in velocity between consecutive time steps. In their study, even though the observation error was considered, the driving error was ignored. [4] presented a control algorithm for the autonomous overtaking problem using SMPC on a one-directional lane. The purpose of the control model is to track the longitudinal speed and lateral position references. To avoid the complexity of modeling, many researchers started to use reinforcement learning to solve this kind of problem. [5] explored the use of deep reinforcement learning to first teach a car to drive on a straight road and then train it to perform lane overtaking.

All of these studies provided insights into autonomous cars. However, there is still room for improvement in ensuring that the autonomous vehicle is energy-efficient, safe, and provides a comfortable experience for passengers. Overtaking, specifically, is one aspect of driving that requires calculation and skill. In this paper, we will discuss how an autonomous vehicle can safely perform overtaking maneuvers on a bidirectional two-lane highway by analyzing two methods - model-based optimal control and model-free reinforcement learning algorithm. Overall, our work aims to contribute to the development of safe and comfortable autonomous vehicles, which is crucial for the future of transportation.

The motivation behind using reinforcement learning is to train the vehicle to interact with and learn from its environment, instead of following a fixed policy for its behavior. This would allow the vehicle to navigate through unique situations it encounters on a highway. We use a similar approach as did the authors of [6]. Namely, we utilize the model-free Deep Deterministic Policy Gradient algorithm to incrementally train the car to conduct the overtaking maneuver. We will change the environment used in [6]–a multi-lane highway, to a bidirectional two-lane highway. A simplified problem will allow for a deeper study into how the agent can learn and perfect simple tasks.

The authors of [7], used a similar environment to the one we will be exploring - a bidirectional highway. However, they followed a more conventional mathematical approach by designing a Model Predictive Controller (MPC), which outputs a binary decision to conduct the overtaking maneuver and the ability to retract the decision based on sensory inputs.

II. Problem Formulated

The following car model is used in this article:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \\ \dot{v} = u \end{cases} \tag{1}$$

where (x, y) is the position of the car, θ is the steering angle of the car, v is the velocity, ω and u are the control inputs for the system. The problem is described as follows: there

[1]Meghna Jain, Daniel Geng, Joshua Chio, and Lydia Han, Mi Zhou, Erik I. Verriest are with the School of Electrical and Computer Engineering, Georgia Institute of Technology. Email: {mjain92, dgeng31, jchio, whan65,mzhou91, erik.verriest}@gatech.edu.

are two lanes and car “Ego” (self) needs to overtake the cars “Bobs” to reach its final position. However, it needs to take into consideration the cars “Alices” to avoid collision. Fig. 1 gives a more explicit description of the problem we want to solve. The following assumptions are made to simplify the problem:

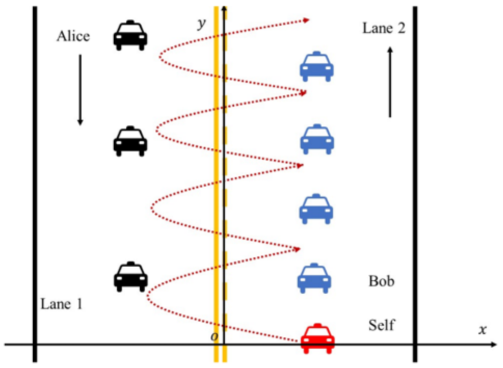

Fig. 1: Illustration of this overtaking problem

Assumption 1: The width and length of all cars are the same (compact cars). Only car Ego is to realize overtaking.

Assumption 2: The cars Alices and Bobs are moving with constant velocity v_A, v_B respectively, where $v_A \leq 0$ and $v_B \geq 0$. They will keep in the middle line of their lane driving. The Ego car moves initially with a velocity v_B.

III. Methods

Two main categories of methods exist for this problem, namely model-based and model free. In a model-based solution, information on the dynamics and environment is used in path planning. In this study, the model-based solution is essentially an optimal control problem. Optimal control guarantees optimal and accurate solutions but it relies heavily on the accuracy of the model and struggles to handle a complicated and dynamic environment. The reinforcement learning algorithm is chosen as the model-free solution for this study as it is highly suitable for problems with sequential decision-making. Its scheme is to learn through trial and error by interacting with different scenarios of the environment, which is generally expensive, if even available. This research studies both approaches simultaneously. It compares the reinforcement learning approach with the model-based optimal control method to highlight the advantages and limitations of both approaches in the given context. Specifically, comparisons are made to see how closely the results of reinforcement learning match that of optimal control.

A. Reinforcement Learning

Reinforcement learning is a model-free method in which an agent trains itself by interacting with the environment. DDPG is a special reinforcement learning algorithm that takes continuous action space and state space. A DDPG algorithm composes of four neural networks: actor network, critic network, target actor network, and target critic network. Further, DDPG uses neural networks to approximate the Q function and policy function. It follows a value-based policy where the goal is to maximize the cumulative reward of the agent. Fig. 2 shows the structure of this DDPG algorithm used in this study.

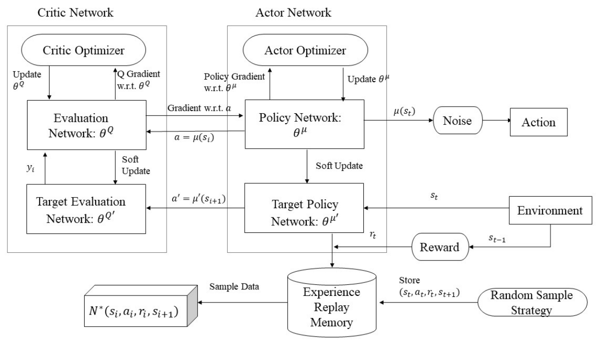

Fig. 2: DDPG Algorithm.

To be more specific, the DDPG algorithm utilizes a replay buffer to keep track of environment variables during the learning process. Our replay buffer is stored as a tuple (s, s_n, a, r, t) where s, s_n, a, r, t represent the current state, next state, action space, reward, and a Boolean terminal flag. The actor is able to pass in observations using this replay buffer to store previous information. This helps the agent learn from a diverse set of experiences and reduces the correlation between consecutive samples.

The observation space is designed as an array of size 9, i.e., $[x_e, y_e, \theta, v, x_b, y_b, x_a, y_a, t]$ that keeps track of the current state of all 3 cars in the environment.

1) x_e: X Position of car Ego
2) y_e: Y Position of car Ego
3) θ: Car Steering Angle
4) v: Car velocity on y-axis
5) x_b: X Position of car Bob moving with constant velocity
6) y_b: Y Position of car Bob moving with constant velocity
7) x_a: X Position of car 'Alice' moving with constant velocity
8) y_a: Y Position of car 'Alice' moving with constant velocity
9) t: Current time elapsed

Based on the current state and reward, our algorithm will output a control policy in velocity and a control policy in steering angle which the agent will use for its new trajectory. The action space dictates the scope of this input. The action space is (ω, u), where u is the change in velocity or acceleration and ω is the change in steering angle. The limits on action space span from -3 to 3, thus both the velocity and steering angle will only change within this range in each step.

In order to simulate a bidirectional highway, this project utilizes Gym from OpenAI, an open-source library to simulate

AI environments. Using this library and our algorithm, a two-lane bidirectional highway environment is constructed to train the actor. This is shown in Fig. 3. The green circle represents the car Ego and the two red circles are cars moving at constant velocities in opposite directions denoted by the corresponding arrows. The goal of the car Ego is to overtake the red car in front of it while avoiding both red cars during the maneuver.

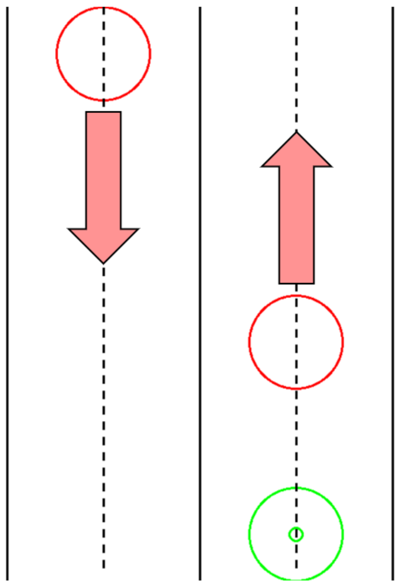
Fig. 3: The environment where the cars interact.

To train the agent in this environment, several factors are taken into consideration in order to create the desired path. The factors are broken up into a reward system where the agent receives positive and negative rewards based on its actions.

The first two reward distinctions are successful overtaking and crashing. The requirements for overtaking consist of the agent being ahead of the car in front of it and back in the same lane. Crashing occurs when the agent touches either of the constant cars. A successful overtaking adds 1000 to the reward while a crashing event deducted 1000. The other constraint is that the car has to stay within the bounds of the lanes. Going out of bounds in either direction also results in a deduction of 1000.

To minimize energy consumption, we define a reward function as follows:

$$r_{energy} = -\frac{1}{2}(u^2 + \omega^2).$$

Using the action space inputs, an energy cost is calculated. These two actions that are chosen by our DDPG algorithm signify the amount of energy a given step will use. Since the algorithm strives to maximize its reward, the negative prefactor compels the algorithm to minimize the energy cost.

Separate from the energy cost, there are a few additions to the reward function to encourage correct behavior. These experiments yield various best paths, as will be explored below.

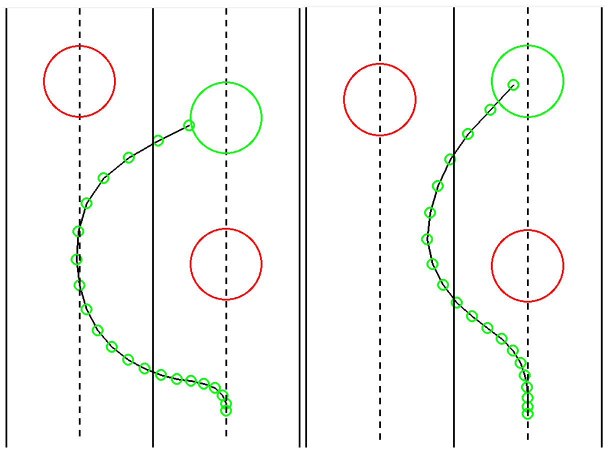
(a) Energy Cost: 44.100 (b) Energy Cost: 37.131

Fig. 4: The minimum cost path of two different reward systems after 5000 and 10000 episodes respectively. The green path is the path that the agent took to overtake the car in front of it.

As presented in Fig. 4a, this is the best path found after 10000 episodes using the original reward function without modifications. The final energy cost of this path is 44.100. However, this path is not the most optimal path, as it veers off a little too far in the left lane. A more optimal path would stay closer to the y-axis to conserve energy. To improve upon this iteration, a reward condition is introduced to help keep the agent further from the left lane. This condition is a function that penalizes the car as it drifts further left. The result of this addition is shown in Fig. 4b. The path is indeed tighter; thus the energy consumed is reduced to 37.131.

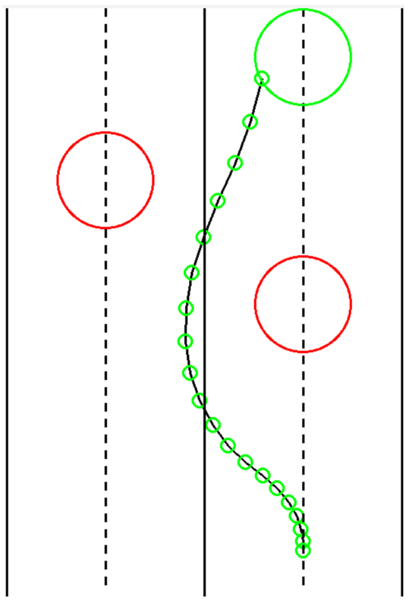
Fig. 5: The minimum cost path after a run of 10000 episodes using the final reward function. The energy cost of this path is 35.928.

The last modification to the reward function is to factor in the distance difference between the car Ego and the red car in front of it. This encourages the car Ego to overtake quicker. The result of this modification is shown in Fig. 5. This path is the most energy-efficient path found through the reinforcement learning approach.

B. Optimal Control

To the same problem, we also formulate this problem as the following optimal control problem:

$$\min_{u,\omega} \; J = \int_0^{t_f} L(x(t), u(t))\mathrm{d}t \tag{2}$$

$$s.t, \; \dot{s} = f(s, u) \tag{3}$$

$$(s - s_a)^2 \geq 2^2 \tag{4}$$

$$(s - s_b) \geq 2^2 \tag{5}$$

where $L(x(t), u(t))$ is the stage cost which we will define later, $s = (x, y)$ is the position of Ego car, $s_b = (x_b, y_b)$ and $s_a = (x_a, y_a)$ are the positions of cars Bob and Alice respectively. Based on our assumption, $x_b = 2$, $x_a = -2$, $y_b = V_B t$, $y_a = V_A t$ which means cars Bob and Alice are moving in their center lane with constant velocity 1 from two directions. Eqn. (3) is the car's dynamics introduced in Eqn. (1). Eqn. (4) and Eqn. (5) are the constraints that ensure that the Ego car has a safe distance from two other cars.

The optimal control method is built upon Pontryagin's Maximum Principle, which states the conditions required to take a dynamic system from one state to another while minimizing/maximizing some cost functions under certain constraints. In contrast to the model-free approach that explores all scenarios in one setting, optimal control studies different overtaking scenarios separately. In all of the cases studied, the constraint is the car(s) to be overtaken. ICLOCS2 (Imperial College London's MATLAB toolbox for Optimal Control) [8], is used to solve our defined problem.

In the experiments, all cars are modeled as a circle of radius one. This is a simplification of the real-world case, as non-convex optimization is significantly more difficult than its convex counterpart.

The first batch of experiments studies the most basic case where only stationary cars are present as the constraint. The objective is to minimize the energy cost

$$J = \int_0^{t_f} \omega^2 + u^2 \mathrm{d}t,$$

where ω is the steering angle and u is the acceleration of the car Ego, which is closely tied to petrol consumption. The Ego car travels in the positive $\pi/2$ direction on the coordinate axis from (2,0) to (2,10). The steering angle is constrained to $(0, \pi)$ and the velocity is within (-10,10). The final time of overtaking is set to 10 seconds. In each experiment, the position of the constraint is changed and the minimized trajectory cost is found. The results are tabulated in Table I. As can be seen in this table, when there is no obstacle, the car Ego can drive at the least cost by simply taking a straight line, which is in line with our life experience. The position of car Bob will influence the cost spent to achieve overtaking. When car Bob is in the middle of the initial position and terminal position (i.e., $(2, 5)$), the energy cost is the least.

	Ego Car Start Position	**Bob Car position**	**Cost**
1	(2,0)	None	1.2000
2	(2,0)	(2,2)	6.5748
3	(2,0)	(2,3)	2.4767
4	(2,0)	(2,4)	2.1121
5	(2,0)	(2,5)	2.0282
6	(2,0)	(2,6)	2.1121
7	(2,0)	(2,7)	2.4767
8	(2,0)	(2,8)	5.5839

TABLE I: Single Stationary Car Overtake.

The subsequent batch of experiments, shown in Table II, introduces an additional moving constraint. pos_B and pos_A refer to the starting position of cars Bob and Alice respectively. V_B and V_A refer to their corresponding velocities. A noteworthy behavior is that when the position of car Alice is sufficiently far away (as in runs 10, 11, and 12), its effect on the overtaking trajectory becomes negligible, hence their trajectory costs coincide with that of Experiment 3 as shown in Table II, in which no car Alice is present.

	Ego Car	**Car Bob**		**Car Alice**		**Trajectory Cost**
	Starting position	pos_B	V_B	pos_A	V_A	
1	(2,0)	(2,3)	0.3	None	N/A	2.0626
2	(2,0)	(2,3)	0.4	None	N/A	2.0292
3	(2,0)	(2,3)	0.5	None	N/A	2.1031
4	(2,0)	(2,2)	0.6	None	N/A	2.0688
5	(2,0)	(2,10)	-0.5	None	N/A	2.3495
6	(2,0)	(2,10)	-1.0	None	N/A	2.0282
7	(2,0)	(2,10)	-2.0	None	N/A	2.4106
8	(2,0)	(2,10)	-3.0	None	N/A	3.3082
9	(2,0)	(2,3)	0.4	(2,9)	0.4	2.0292
10	(2,0)	(2,3)	0.5	(-2,8)	-0.5	2.1031
11	(2,0)	(2,3)	0.5	(-2,9)	-0.5	2.1031
12	(2,0)	(2,3)	0.5	(-2,10)	-0.5	2.1031

TABLE II: Moving Car Overtake.

The final part of this section redefines the metric studied: instead of optimizing the cost function within a fixed time, the time is now set as a free variable and becomes the cost. Namely, the objective is redefined as

$$\min \; t_f$$

where t_f is the final time of an overtaking trajectory. The results are shown in Table III. As the table suggests, when there is only car Bob, the minimal time for overtaking occurs when the car Bob is in the middle of the initial position and terminal position. The addition of the car Alice does not influence the overtaking time in this deterministic environment since there is enough safe space for both cars.

IV. Model predictive path integral control

While the reinforcement learning-based method gets rid of the complexity of manual modeling, the training process takes a long time. What's more, in real applications, mechanical

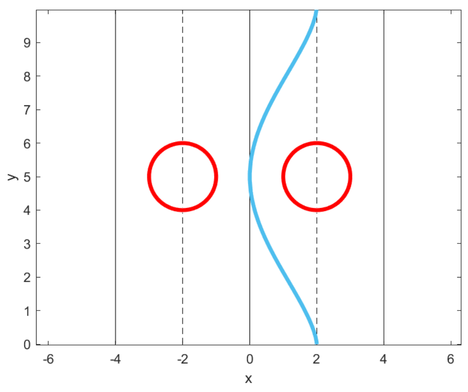

Fig. 6: Optimal trajectory with 2 stationary vehicles as constraints.

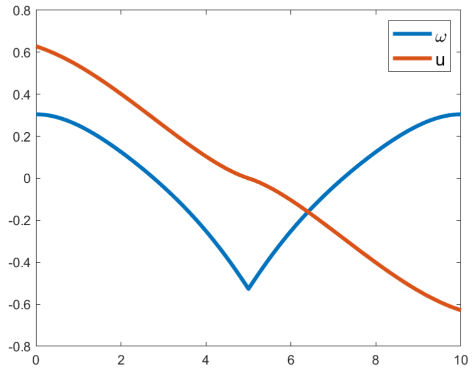

Fig. 7: Control variables u and ω to achieve optimal trajectory.

	Ego Car Starting Point	Car Bob	Car Alice	Cost (t_f)
1	(2,0)	None	None	2.0000
2	(2,0)	(2,2)	None	2.2952
3	(2,0)	(2,3)	None	2.0984
4	(2,0)	(2,4)	None	2.0848
5	(2,0)	(2,5)	None	2.0811
6	(2,0)	(2,6)	None	2.0848
7	(2,0)	(2,7)	None	2.0984
8	(2,0)	(2,8)	None	2.2944
9	(2,0)	(2,5)	(-2,3)	2.0811
10	(2,0)	(2,5)	(-2,7)	2.0811
11	(2,0)	(2,5)	(-2,5)	2.0811

TABLE III: Moving Cars Overtake.

damage is not irreversible. Thus, recent research mainly builds up a similar simulation environment and lets the car be trained in an offline mode. On the other hand, the disadvantage of optimal control, besides the difficulty of modeling, is that it cannot account for the variation of the environment and the driving errors of other cars. Thus, the study turns to algorithmic- and sampling-based solutions–such as Model Predictive Path Integral (MPPI) as a compromise of both aforementioned methods. MPPI, boasting higher flexibility [9], is chosen as the theoretical foundation of work in this phase.

This method develops control algorithms by employing the importance sampling of future trajectories. For a given number of time steps, the algorithm generates random disturbance vectors δu_k, applies them to the input u, and simulates their effect on the car's trajectory. Using a chosen cost function, each trajectory is assigned a cost $\tilde{S}_k$. With the cost, the algorithm updates the input vector with the following formula:

$$u(x(t),t) = u(x(t),t) + \sum_k \frac{e^{-\frac{1}{\lambda}\tilde{S}_k}\delta u_k}{e^{-\frac{1}{\lambda}\tilde{S}_k}}$$

where λ is a determined constant parameter. The lower the cost of a sample, the greater its weight in the input update calculation and the lower the cost of the ultimate optimal trajectory.

Using the open-source MPPI toolbox [10], and [11] as the theoretical foundation, experiment results are summarized in Tables IV and V. The control data from the deterministic experiments is taken as the initial control sequence and Gaussian white noise is added. The new control inputs u are modelled as random vectors with of the form

$$u_{stochastic} \sim \mathcal{N}(u_{determinstic}, \Sigma),$$

where Σ is the covariance matrix. Here, the inputs are no longer directly controlled. Instead, their arithmetic means are determined. In the following experiments, λ is set to 1, the time horizon 10 seconds, and the control noise covariance matrix diag(0.5,0.5). As in the deterministic case, the Ego Car's initial position is (2,0). For each scenario, data is collected for 100 runs. The trajectory data is then used to determine the occurrence of collisions. $\boldsymbol{\mu}$ is the arithmetic mean and $\boldsymbol{\sigma}$ is the standard deviation. $pos.$ is the coordinates of the initial position and $Cols.$ is the number of collisions. The success rate is calculated by

$$SuccessRate = \frac{100 - TotalCollisions}{100}.$$

The results for the stationary and moving experiments are tabulated in Tables IV and V respectively. As shown in Table IV, if only considering car Bob, the success rate first increases and then decreases as car Bob's initial position increases. With the addition of the car Alice, the success rate decreases, which means overtaking in a bidirectional lane is highly risky. The closer the two obstacles, the riskier the collision during overtaking. Table V illustrates the risk when both Alice and Bob are moving. The success rate decreases below 30% when both cars are moving with velocity 0.5 m/s.

Fig. 8 is the mean trajectory and standard deviation with two stationary vehicles and Fig. 9 is the corresponding mean and standard deviation of control variables with the influence of noise. The shading area is the risky area that needs to be taken into consideration when overtaking.

V. Conclusion and Future Work

In this paper, a DDPG framework for a car to conduct overtaking maneuvers in a specific environment was developed. This framework can be extended to cover more situations:

	Car Bob		Car Alice		Cost (Energy)		Success
	pos_B	Cols.	pos_A	Cols.	μ	σ	Rate
1	(2,2)	55	None	N/A	7.2979	0.1538	0.45
2	(2,3)	2	None	N/A	2.8802	0.1234	0.98
3	(2,4)	4	None	N/A	2.4818	0.1258	0.96
4	(2,5)	9	None	N/A	2.4105	0.1270	0.91
5	(2,6)	11	None	N/A	2.5136	0.1331	0.89
6	(2,7)	26	None	N/A	2.9210	0.1345	0.74
7	(2,8)	21	None	N/A	7.2946	0.2479	0.79
8	(2,5)	17	(-2,2)	0	2.3940	0.1091	0.83
9	(2,5)	10	(-2,3)	4	2.3932	0.1143	0.86
10	(2,5)	8	(-2,4)	74	2.3991	0.1194	0.18
11	(2,5)	12	(-2,5)	88	2.3995	0.1100	0.00
12	(2,5)	13	(-2,6)	75	2.4020	0.1041	0.12
13	(2,5)	6	(-2,7)	36	2.3851	0.1101	0.58
14	(2,5)	12	(-2,8)	3	2.3880	0.1018	0.85

TABLE IV: MPPI Stationary Obstacles.

	Car Bob			Car Alice			Cost (Energy)		Success
	pos_B	V_B	Cols.	pos_A	V_A	Cols.	μ	σ	Rate
1	(2,3)	0.3	4	None	N/A	0	2.4222	0.1134	0.96
2	(2,3)	0.4	13	None	N/A	0	2.4010	0.1273	0.87
3	(2,3)	0.5	41	None	N/A	0	2.4561	0.1001	0.59
4	(2,2)	0.5	44	None	N/A	0	2.4328	0.1161	0.56
5	(2,10)	-0.5	16	None	N/A	0	2.7861	0.1324	0.84
6	(2,10)	-1.0	8	None	N/A	0	2.4105	0.1270	0.92
7	(2,10)	-2.0	1	None	N/A	0	2.7557	0.1012	0.99
8	(2,10)	-3.0	0	None	N/A	0	3.7375	0.1102	1.00
9	(2,3)	0.4	13	(2,9)	0.4	3	2.3942	0.1112	0.84
10	(2,3)	0.5	30	(2,8)	-0.5	100	2.4573	0.1094	0.00
11	(2,3)	0.5	43	(2,9)	-0.5	56	2.4706	0.1168	0.01
12	(2,3)	0.5	31	(2,10)	-0.5	52	2.4358	0.1056	0.17

TABLE V: MPPI Moving Obstacles.

a vehicle could potentially be in environments with various numbers of surrounding cars along with their corresponding velocities and diverse interactions with the agent. There is also room for optimizing energy consumption during maneuvers while maintaining safety standards.

In terms of Model Predictive Path Integral control, this sampling-based method can be used as an approximation method of optimal control in real applications. However, the optimal control policy obtained using this method has risks of collision. This introduces greater unpredictability and thus calls for more intensive experimentation.

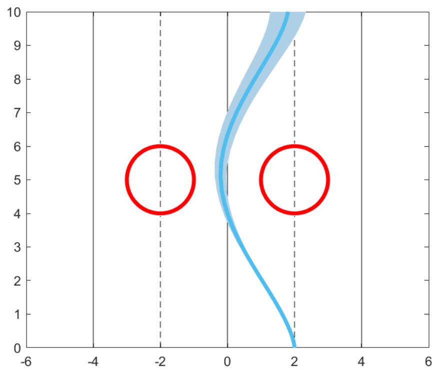

Fig. 8: Mean trajectory and standard deviation after MPPI algorithm of case with 2 stationary vehicles as constraints.

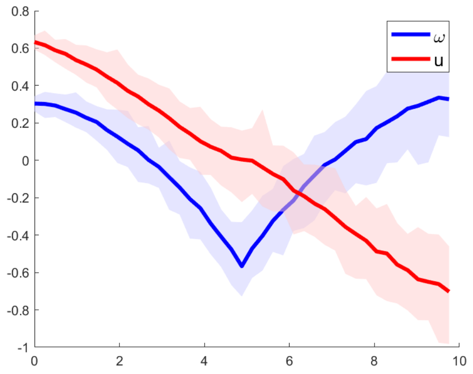

Fig. 9: Mean and standard deviation of control variables u and ω after MPPI algorithm.

References

[1] S. Feraco, S. Luciani, A. Bonfitto, N. Amati, and A. Tonoli, "A local trajectory planning and control method for autonomous vehicles based on the rrt algorithm," in *2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, 2020, pp. 1–6.

[2] F. M. Tariq, N. Suriyarachchi, C. Mavridis, and J. S. Baras, "Vehicle overtaking in a bidirectional mixed-traffic setting," in *2022 American Control Conference (ACC)*, 2022, pp. 3132–3139.

[3] N. A. Nguyen, D. Moser, P. Schrangl, L. del Re, and S. Jones, "Autonomous overtaking using stochastic model predictive control," in *2017 11th Asian Control Conference (ASCC)*, 2017, pp. 1005–1010.

[4] ——, "Autonomous overtaking using stochastic model predictive control," in *2017 11th Asian Control Conference (ASCC)*, 2017, pp. 1005–1010.

[5] M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1885–1890.

[6] ——, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1885–1890.

[7] F. M. Tariq, N. Suriyarachchi, C. Mavridis, and J. S. Baras, "Autonomous vehicle overtaking in a bidirectional mixed-traffic setting," in *2022 American Control Conference (ACC)*, 2022, pp. 3132–3139.

[8] Y. Nie, O. Faqir, and E. C. Kerrigan, "Iclocs2: Try this optimal control problem solver before you try the rest," in *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018, pp. 336–336.

[9] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control using covariance variable importance sampling," *CoRR*, vol. abs/1509.01149, 2015. [Online]. Available: http://arxiv.org/abs/1509.01149

[10] acxz, "acxz/mppi: Mppi in octave/matlab," Nov. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3539762

[11] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *Trans. Rob.*, vol. 34, no. 6, p. 1603–1622, dec 2018. [Online]. Available: https://doi.org/10.1109/TRO.2018.2865891