Application of Model Predictive Path Integral Controller in Autonomous Driving: A Simulation Study

Honglin Guo

School of Electrical Engineering and Artificial Intelligence, Xiamen University Malaysia-Xiamen University Malaysia, 43900 Sepang, Selangor, Malaysia
ait2109082@xmu.edu.my

Abstract. In the realm of autonomous driving technology, significant strides have been made, yet challenges persist. This paper aims to explore the effectiveness of applying a Model Predictive Path Integral (MPPI) controller in autonomous driving technology. The advancement of autonomous driving technology has become one of the hotspots in the automotive industry, with one of its core challenges being the design of control systems capable of safely navigating various complex environments. In this experiment, we selected a scenario with circular obstacles to simulate the obstacle situations that may be encountered on urban roads. The vehicle model takes into account the dynamic characteristics and constraints of the vehicle itself to more realistically simulate actual driving conditions. By demonstrating the navigation and obstacle avoidance capabilities of the MPPI controller in dynamic environments, we hope to prove the potential value of this controller in future autonomous driving scenarios and provide valuable references and insights for the further development of autonomous driving technology.

Keywords: Application; Model Predictive Path Integral; Autonomous driving; Simulation

1 Introduction

The advent of self-driving technology has catalyzed transformative developments in intelligent transportation systems, heralding a new era in the automotive industry. Autonomous navigation, a cornerstone of self-driving technology, necessitates sophisticated path planning and precise control mechanisms to ensure safe and efficient vehicle operation [1, 2, 3]. As the demand for autonomous vehicles capable of navigating diverse and complex environments continues to rise, the imperative for advanced control algorithms becomes increasingly pronounced [4]. In response to this imperative, the Model Predictive Path Integral (MPPI) controller emerges as a highly promising solution for autonomous driving applications. This controller embodies a fusion of model prediction and sample-based optimization techniques, enabling agile and adaptive vehicle maneuvering amidst dynamic and uncertain environments. By iteratively sampling the input space and optimizing trajectory cost functions [5], the

Y. Wang (ed.), *Proceedings of the 2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2024)*, Advances in Computer Science Research 115,
https://doi.org/10.2991/978-94-6463-540-9_46

MPPI controller facilitates effective navigation of nonlinear systems while robustly accounting for environmental uncertainties.

Against this backdrop, this paper endeavors to provide a comprehensive exploration of the application of the MPPI controller in autonomous driving contexts. Specifically, the focus lies in its integration within an autonomous vehicle framework, where it assumes the pivotal role of guiding the vehicle along a predefined reference path while accommodating dynamic vehicle constraints and navigating environmental obstacles. The real-time computation of optimal control inputs empowers the vehicle to seamlessly adapt to evolving surroundings, evade obstacles, and maintain precise path tracking.

In delineating the background and current state-of-the-art, it is pertinent to underscore the challenges inherent in autonomous navigation, including but not limited to the complexities posed by diverse road geometries, varying traffic conditions, and unpredictable obstacles. Traditional control methodologies often struggle to reconcile these challenges, highlighting the need for innovative approaches that can adeptly navigate such complexities.

In this vein, the MPPI controller presents a paradigm shift in autonomous vehicle control, offering a potent amalgamation of predictive modeling and adaptive optimization. By harnessing the power of predictive analytics and iterative optimization, MPPI controllers demonstrate a remarkable capacity to enhance vehicle safety and adaptability in dynamically changing environments.

Through a series of rigorous experiments and simulations, this paper aims to elucidate the robustness and efficacy of MPPI-based control systems in addressing the multifaceted challenges of autonomous navigation.

2 Literature Review

Model Predictive Control (MPC) and its variant MPPI are both powerful control models used in various fields to optimize system performance. While MPC focuses on deterministic optimization of control inputs, MPPI introduces stochasticity through random sampling to evaluate performance across multiple trajectories and account for system uncertainty [6].

In the context of autonomous driving, where real-time decision-making and adaptability are crucial, MPPI offers significant advantages. By exploring diverse potential system behaviors through random sampling and weighted evaluation, MPPI can enhance the adaptability, robustness, and safety of autonomous driving systems. This allows vehicles to navigate complex and uncertain road conditions more intelligently and flexibly, improving overall system performance [7].

One of the key advantages of MPPI over traditional MPC lies in its ability to handle uncertainty and optimize control strategies based on stochastic evaluation. This is particularly beneficial in dynamic and unpredictable environments, where the system may encounter unexpected obstacles or variations in road conditions. By considering multiple possible trajectories and evaluating performance probabilistically, MPPI

enables autonomous vehicles to make more informed and adaptive decisions in real-time, leading to improved safety and efficiency in driving scenarios.

Furthermore, the random sampling approach of MPPI allows for a more computationally efficient solution compared to traditional MPC, as it generates multiple control trajectories and selects the optimal input based on a weighted average of these trajectories [8, 9]. This can be especially beneficial in applications with strict real-time requirements and sensitivity to computational complexity, such as autonomous driving systems where quick and accurate decision-making is essential.

Overall, the application of MPPI in autonomous driving systems holds great promise for enhancing system performance and safety in complex and uncertain driving environments. By integrating stochastic evaluation and diverse trajectory exploration, MPPI can help vehicles adapt to changing conditions, avoid obstacles, and navigate challenging scenarios with improved intelligence and agility [10]. This makes MPPI a valuable tool for advancing the capabilities of autonomous driving technology and ensuring safe and efficient transportation in the future.

3 Methodology

3.1 Theory of MPPI

The MPPI algorithm is an advanced MPC method designed to address control problems in nonlinear systems. Unlike traditional MPC algorithms, MPPI employs a stochastic sampling-based optimization technique, which enables it to effectively handle systems with complex dynamics and high-dimensional state spaces.

The core principle of MPPI involves generating control sequences by optimizing a cost function over a finite time horizon. The process can be summarized into the following steps:

- Control Sequence Sampling: At each time step, MPPI randomly samples a set of possible control sequences from a control distribution. These sequences are typically generated based on prior experience and current state information.
- Performance Evaluation: For each sampled control sequence, MPPI evaluates its performance over a future time horizon. This evaluation is typically based on a predefined cost function that considers control objectives, system constraints, and other problem-specific factors.
- Sequence Weighting: Based on the performance of each control sequence, MPPI assigns weights to them. Generally, sequences with better performance are assigned higher weights, while those with poorer performance are assigned lower weights.
- Generation of New Control Sequences: The weighted control sequences are combined to generate a new control sequence. This process can be as simple as weighted averaging or can involve more complex methods such as Monte Carlo Tree Search.
- Iterative Optimization: The above steps are repeated until a predefined stopping criterion is met. This typically includes reaching a maximum number of iterations,

achieving satisfactory performance levels, or meeting specific convergence criteria.

One of the key features of the MPPI algorithm is its ability to handle uncertainty in the system. By sampling from the control distribution, MPPI naturally takes into account model uncertainty, sensor noise, and other external disturbances, thereby enhancing the robustness of the system. MPPI is capable of striking a balance between exploration and exploitation. By flexibly designing the control distribution, MPPI can simultaneously exploit the current best control sequences while exploring potential new ones, thereby adapting better to environmental changes and learning the complex dynamics of the system.

3.2 Implementation procedure

The Vehicle class is initialized with parameters such as wheelbase, maximum steering angle, maximum acceleration, reference path, time step (delta_t), and visualization settings.

Initialization Parameters wheel_base: The distance between the front and rear axles of the vehicle [m].

vehicle_width: The width of the vehicle [m].

vehicle_length: The length of the vehicle [m].

max_steer_abs: The maximum absolute value for the front tire angle [rad].

max_accel_abs: The maximum absolute longitudinal acceleration of the vehicle [m/s^2].

ref_path: An array representing the reference path for the vehicle in the global frame.

obstacle_circles: An array representing obstacle positions and radii in the form [obs_x, obs_y, obs_radius].

delta_t: The time step for simulation [s].

visualize: A boolean indicating whether to enable visualization.

The dynamics of the vehicle follow the Kinematic Bicycle Model. This is a simplified vehicle model that describes the motion using a combination of translation and rotation. This class is designed to create an environment for simulating the motion of a vehicle.

Visualization Three key functions related to the simulation and visualization of a vehicle: reset, update, and show_animation.

The reset function is responsible for resetting the environment to its initial state. It initializes or resets the state variables (position, orientation, and velocity) and clears the animation frames. For self.visualize_flag, it will set up the figure for visualization.

The update function is responsible for updating the state variables of the vehicle based on control inputs (u) and the Kinematic Bicycle Model. It also records animation frames by calling the append_frame function.

The show_animation function uses Matplotlib's ArtistAnimation to create an animation from the recorded frames and displays it.

3.3 Key Components of MPPI Controller

This controller aims to navigate a vehicle along a predefined reference path by optimizing control inputs based on a cost function. The MPPI algorithm employs a stochastic sampling approach to explore the control input space and generate an optimal trajectory.

Initialization The class is initialized with various parameters and settings necessary for the MPPI algorithm and the vehicle dynamics. Key parameters include the prediction horizon (T), the number of samples for trajectory generation (K), and parameters governing the exploration-exploitation trade-off (param_alpha, param_lambda, param_a).

Control Input Calculation This method calculates the optimal control input for the vehicle given the current state (observed_x). It utilizes the MPPI algorithm by sampling noise (epsilon), generating control input sequences with noise (v), and evaluating the cost function for each trajectory. The algorithm balances exploration and exploitation to find an optimal solution.

Sampling Noise This method generates random noise (epsilon) based on a multivariate normal distribution. The size of the noise matrix is determined by the number of samples (K), the prediction horizon (T), and the dimension of the control input vector (dim_u).

Clamping Control Input This method ensures that the generated control input is within the specified bounds, limiting the steering angle and acceleration to avoid unrealistic values.

Cost Functions The cost functions are defined to quantify the performance of the system. The stage cost (_c) evaluates the deviation from the reference path at each time step, and the terminal cost (_phi) assesses the final state of the system. These costs contribute to the overall cost function used in optimization.

Nearest Waypoint Calculation This method finds the closest waypoint on the reference path to the current vehicle position. It is crucial for updating the state of the system and evaluating the cost functions.

State Update This method calculates the next state of the vehicle based on the current state and the applied control input. It utilizes the kinematic bicycle model to update the position, orientation, and velocity of the vehicle.

Weight Computation This method calculates the weights assigned to each sample trajectory based on their respective costs. The weights are used to determine the contribution of each trajectory to the final control input.

Moving Average Filter A moving average filter is applied to smooth the generated control input sequence. This filtering helps in achieving a more stable and continuous trajectory.

An additional Python file has been crafted to mirror the existing code structure while introducing a specialized obstacle class. The inclusion of the obstacle class augments the realism and complexity of the test scenario, thereby facilitating further enhancement and validation of the path-tracking controller. In real-world driving contexts, vehicles often encounter obstacles necessitating navigation. By integrating an obstacle class, the simulation emulates the challenges confronted by a path-tracking controller in a dynamic environment.

The 'obstacle_circles' parameter within the Vehicle class characterizes circular obstacles in the simulation, each defined by its position (obs_x, obs_y) and radius (obs_radius). These obstacles are visualized in both the primary view and the mini-map view. The controller now must account for these obstacles and adjust the vehicle's trajectory to evade potential collisions.

4 Results and Discussion

For the simulation, it is set to run every 0.05 seconds for a total of 1000 steps. The path is sourced from a file ('ovalpath.csv') and visualized using matplotlib. The focal point is the car, with defined attributes such as wheelbase, maximum steering angle, and maximum acceleration. The simulation commences with the car's initial position, orientation, and velocity (Figure 1 and Figure 2). The MPPI controller, akin to a stage director, prepares with specific configurations. These configurations aid in predicting the future, determining the number of paths to consider, and striking a balance between exploration and exploitation. The introduction of noise adds an element of unpredictability to the process, enriching the simulation experience.

As the simulation progresses, the MPPI controller continuously issues directives to the vehicle based on its current position. We observe the vehicle's responses, documenting crucial details such as optimal moves, action sequences, and the trajectory. At each iteration, the behavior of the vehicle is scrutinized. If the vehicle successfully reaches the designated endpoint of the planned path, the simulation terminates. However, in case of anomalies, such as premature deviation from the planned trajectory, the simulation halts and notifies accordingly. The culmination involves an engaging animation illustrating the vehicle's journey. Further analysis of experimental results reveals that under typical conditions, the MPPI controller adeptly guides the vehicle along the planned path. Nonetheless, it is noted that in certain scenarios, environmental changes or system uncertainties may cause deviations from the expected trajectory. In such instances, the MPPI controller demonstrates prompt

responsiveness by adjusting control strategies to ensure the vehicle's safe arrival at its destination.

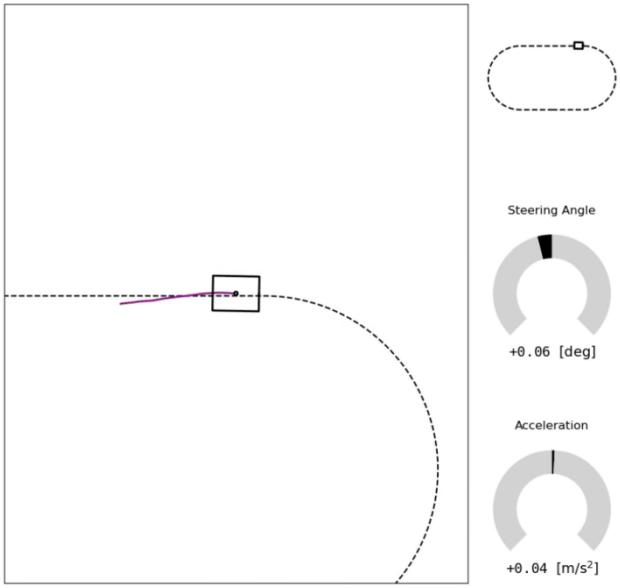

Fig.1 Without Obstacle

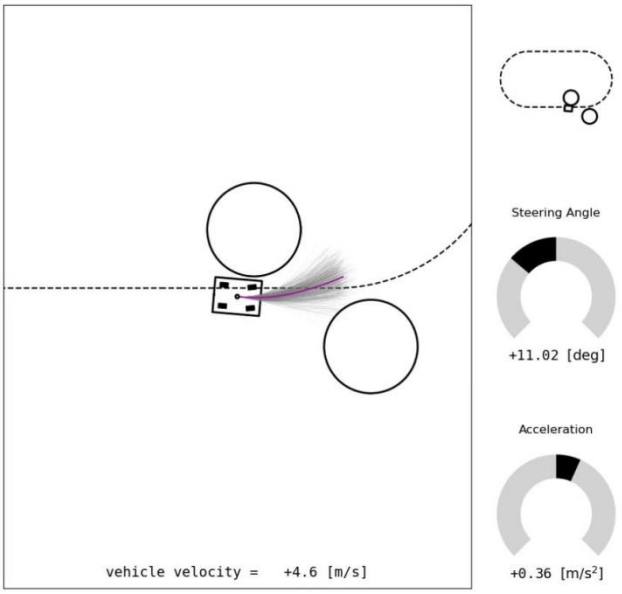

Fig.2 With Obstacle

In summary, the script presents an engaging performance wherein the vehicle and MPPI controller collaborate synergistically. Analogous to observing a choreographed dance, the animation elucidates how the controller navigates the vehicle through the convoluted path, while the analysis of experimental results validates the efficacy and robustness of the MPPI algorithm in practical applications.

5 Conclusion

This paper demonstrates the successful implementation and effectiveness of the MPPI controller in autonomous vehicle navigation. Utilizing a Python-based simulation incorporating the Kinematic Bicycle Model, the study showcases the robustness and adaptability of the MPPI controller in dynamic driving environments. The pivotal achievement lies in the successful deployment of the MPPI controller, showcasing its capability to maintain the vehicle on a predefined path and navigate obstacles, thereby emphasizing its potential for real-world autonomous driving applications.

The evaluation of the controller's response to unexpected situations, facilitated by the introduction of obstacles in the simulation, is crucial for advancements in autonomous driving. Detailed visualizations and assessments provide a comprehensive understanding of the controller's operations and interactions with the vehicle and surroundings. The simulations highlight the adeptness of the MPPI controller in handling uncertainties and nonlinear dynamics intrinsic to autonomous driving, owing to its stochastic approach that provides adaptive and predictive control capabilities distinct from traditional deterministic methods. This adaptability is essential for ensuring the safe and efficient navigation of autonomous vehicles in complex real-world scenarios.

While the findings are promising, it is imperative to acknowledge that the simulated environment presents an oversimplified representation of real-world driving conditions. Future research should focus on testing the MPPI controller in diverse and more realistic scenarios, integrating complex vehicle models and real-world data to further validate its efficacy.

In essence, this project underscores the pivotal role of advanced control algorithms like MPPI in advancing autonomous driving technology. The insights gleaned from the simulations can guide the development of more sophisticated and reliable autonomous driving systems, expediting their integration into everyday use.

References

1. Vu T., Moezzi R., Cyrus J., et al. Model predictive control for autonomous driving vehicles. Electronics, **10** (21): 2593, (2021) .
2. Rokonuzzaman M., Mohajer N., Nahavandi S., et al. Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking. IEEE Access, **9**: 128233-128249, (2021).
3. Musa A., Pipicelli M., Spano M., et al. A review of model predictive controls applied to advanced driver-assistance systems. Energies, **14**(23): 7974, (2021).

4. Tang L., Yan F., Zou B., et al. An improved kinematic model predictive control for high-speed path tracking of autonomous vehicles. IEEE Access, **8**, 51400-51413, (2020).
5. Ye B. L, Wu W., Ruan K, et al. A survey of model predictive control methods for traffic signal control. IEEE/CAA Journal of Automatica Sinica, **6**(3), 623-640,(2016).
6. Park S., Oh K., Jeong Y., et al. Model predictive control-based fault detection and reconstruction algorithm for longitudinal control of autonomous driving vehicle using multi-sliding mode observer. Microsystem Technologies, **26**(1), 239-264,(2020).
7. Chowdhri N., Ferranti L., Iribarren F S., et al. Integrated nonlinear model predictive control for automated driving. Control Engineering Practice, **106**, 104654, (2021).
8. Pan, J., Zhu, J., Yan, Q., Li, C., & Chen, X. A review of model predictive control for autonomous driving. IEEE Access, 106173-106186,(2020).
9. Lin, P., Kang, Z., Jia, Y., & Li, K.. Model predictive control for autonomous driving using the nonlinear kinematic bicycle model. IFAC-PapersOnLine, **52**(11), 93-98, (2019).
10. Wang, Z., Jia, X., Wu, J., Zhang, C. Dynamic obstacle avoidance in autonomous driving using model predictive control. IEEE Transactions on Vehicular Technology, **68**(7), 6901-6912, (2018).