Kinematic Control Of A 2D Differential Drive Unammed Ground Vehicle Using Model Predictive Path Integral Controller.

Azmyin Md. Kamal 89-579-4248[1]

***Abstract*— Control algorithms for Unmanned Ground Vehicles (UGVs) must manage complex state dynamics, sensor uncertainties, and environmental changes in real-time for safe autonomous operation. These algorithms require capabilities to work with non-convex and non-differentiable objective functions, generate control policies in real-time, and integrate various domain-specific constraints. The Model Predictive Path Integral (MPPI) controller, a stochastic optimal control approach, has gained attention in robotics for its effective real-time control of highly nonlinear systems. Its applications include high-speed maneuvering of small-scale rally cars, collision-free navigation of quadrotors in occluded environments, humanoid robot control and so on. While several recent MPPI methods have been proposed for UGV control, most have tested their approaches on four-wheel robots modeled using the simple bicycle model. Only a limited number of studies have evaluated the MPPI control strategy on four-wheel skid-steering robots utilizing differential drive kinematics.**

Motivated by the above, the aim of this project is to study the kinematic control of a 2D Unmanned Ground Robot modeled using the ideal-differential drive kinematics model. Two test paths of varied difficulty and object density was chosen for this experiment. For each timestep of 0.1 s and sampling 100 trajectories per step, the average RMSE for X, Y and Yaw axes was found to be 0.58 m, 0.32 m and 3.53 radians respectively. These results demonstrate MPPI's effectiveness in kinematic control of skid-steered robots. The complete source code to replicate this study is made publicly available at: `https://github.com/Mechazo11/mppi-python`

- unmanned ground vehicles, model predictive control, kinematic control.

I. Introduction

Field robots in real-life conditions are often required to perform complex tasks in a dynamically challenging environment [1]. At the same time, they are required to handle new observational data without incurring high-run time penalty while contending with potentially nontrivial system dynamics, non-differential, non-convex objective functions and input-output constraints [2].

Model Predictive Control (MPC) is a popular approach due to its inherent capability to handle non-linear dynamics [3]. This approach can be divided into two families. The first family of MPC formulations involves using linear model approximations with quadratic cost functions, sequential quadratic programming, interior point methods, or Koopman operators [4]. While these methods offer computationally inexpensive optimal solution [2], they have several limitations, chief among them the handling of non-convex, non-differentiable objective functions and constraints inherent in numerous real-world applications. These applications often involve a wide variety of situations, many of which cannot be known a priori, such as aggressive autonomous driving that requires fast lane keeping, obstacle avoidance [5] or real-time collison-free navigation in partially observerd 3D space of a quadrotor [6]. The other family of approaches consists of stochastic sampling methods that utilize Monte Carlo sampling for trajectory optimization. The Model Predictive Path Integral (MPPI) controller [7], first introduced in 2016 for aggressive race car driving, demonstrated real-time control using a neural network to represent the system. This was later extended with its information-theoretic formulation [5], the arbitrary choice of proposal distributions [3], and, more recently, efficient sampling coupled with learning-based ancillary controllers [1]. The general idea of MPPI is to sample a large number of control sequences (parallely on a GPU), compute cost of each sequence and then calculate a cost informed weighted average of the sampled control. The first control in the sequence is then executed with the remaining portion used in the next timestep as the mean of the proposal distribution.

In the motion planning and control literature for Unmanned Ground Robots, most work has focused on developing MPPI solutions for four-wheel vehicles with an Ackermann steering mechanism, whose kinematics can be accurately modeled using the simple bicycle model [1], [3], [5]. Skid-steer four-wheel robots, recognized for their high traction and payload capacity [8], have received little attention from MPPI researchers, with only a handful of studies testing MPPI control strategies on skid-steered robots [9].

Hence, this project aims to study the path-tracking performance of a four-wheel skid-steered robot kinematically modeled as an ideal 2D differential drive robot [10] using the Information-Theoretic MPPI (IT-MPPI) controller based on [5]. The problem is thus, formulated as a finite-horizon optimal control problem [7].

The objectives of this project are twofold:

- Control a four-wheel skid-steer 2D robot using an ideal differential drive kinematic model and IT-MPPI.
- Demonstrate robust path tracking and obstacle avoidance on oval and figure-eight trajectories.

To achieve these objectives, we extensively modified an existing open-source IT-MPPI project [11] to implement successful path tracking for a skid-steer robot modeled

*Corresponding author

[1]The author is with the Department of Mechanical & Industrial Engineering at Louisiana State University, Baton Rouge, LA 70803, USA. This report has been submitted in partial fulfillment of the requirements for the Final Project of the EE 7500 Model Predictive Control class instructed by Dr. Xiangyu Meng. `akamal4@lsu.edu`

with an ideal differential drive kinematic framework. The contributions of this report are twofold:

- Achieved reliable control of a four-wheel differential drive robot on two different 2D paths with varying difficulty and obstacles.
- Created an open-source Python project to reproduce the study's results.

The remainder of the paper is organized as follows: Section II presents a concise discussion on the formulation of the MPPI framework, differential drive kinematic model and the experimental procedure. In Section III, the experimental results are discussed. Section IV summarizes the findings and discusses future work.

II. Methodology

At first a very high-level overview of the IT-MPPI formulation is presented followed by the

A. Brief overview of IT-MPPI

The following is a high-level working methodology of IT-MPPI is based on [1], [2], [12]. Please go through the original papers for a thorough mathematical treatment on development of IT-MPPI.

Consider a generalized non-linear discrete-time dynamical system

$$\zeta_{t+1} = \mathbf{F}(x_t, v_t) \tag{1}$$

$$v_t \sim \mathcal{N}(u_t, \Sigma_t) \tag{2}$$

where $x \in \mathbb{R}^n$ is the state, $F(\cdot)$ is the state transition function, $v_t \in \mathbb{R}^m$ are the "colored noise" inputs based on the commanded inputs $u_t \in \mathbb{R}^m$ at timestep t, and T is the total number of timesteps in the horizon. $\Sigma_t \in \mathbb{R}^{m\times m}$ represents the covariance of the Gaussian noise associated with the input u_t [3]. Let $\mathbf{U} = [u_0, u_1, \ldots, u_{T-1}] \in \mathbb{R}^{mT}$ and if the combined covariance matrix Σ is defined as following

$$\mathbf{\Sigma} = \begin{bmatrix} \Sigma_0 & 0 & \cdots & 0 \\ 0 & \Sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{T-1} \end{bmatrix}, \quad \Sigma \in \mathbb{R}^{mT\times mT} \tag{3}$$

then input vector $\mathbf{V}$ is defined as a random vector sampled using

$$\mathbf{V} \sim \mathcal{N}(\mathbf{U}, \mathbf{\Sigma}) \tag{4}$$

For simplicity of explanation let $\mathbb{Q} = \mathbb{Q}_{\mathbf{V},\Sigma}$ represent a probability measure that characterizes the control input distribution.

IT-MPPI approach developed by Williams et al. [5], [7] first showed that, a for a set of noisy control inputs $\mathbf{V}$ the relation between the free-energy of the system to achieve state $\mathbf{X}$ to the divergence between distributions $\mathbb{Q}$ and base measure $\mathcal{P}$ is bounded by

$$\mathcal{F}(S, \mathbf{p}, \mathbf{X}, \lambda) \leq \mathbb{E}_{\mathbb{Q}}\left[S(\mathbf{V})\right] + \lambda \mathbb{KL}(\mathbb{Q}\|\mathbb{P}) \tag{5}$$

where $\mathbf{X} = [x_0, \mathbf{F}(x_0, v_0), \cdots] \in \mathbb{R}^{nT}$ are the predicted states starting from last known state x_0, $\mathbb{KL}(\mathbb{Q}\|\mathbb{P})$ denotes the KL-Divergence between the input and base measure probability distributions and $S(\mathbf{V}$ is the staging cost to execute control sequence $\mathbf{V}$. What Equation (5) signifies is that the free energy $\mathcal{F}$ of the control system is lower bounded by the expected cost for taking the controlled inputs $\mathbb{V}$. Now defining the finite-horizon optimal control problem as

$$\mathbf{U}^* = \arg\min_{\mathbf{V}\in\mathcal{U}} \mathbb{E}_{\mathcal{Q}}\left[\phi(\mathbf{X}) + \mathcal{L}(\mathbf{X}, \mathbf{V})\right] \tag{6}$$

Equation (5) can be rewritten as

$$\mathcal{F}(S, \mathbf{p}, \mathbf{X}, \lambda) \leq \mathbb{E}_{\mathbb{Q}}\left[\phi(\mathbf{X}) + \mathcal{L}(\mathbf{X}, \mathbf{U})\right]. \tag{7}$$

where $\phi(\mathbf{X})$ is the state dependent cost, $\mathcal{L}(\mathbf{X}, \mathbf{V})$ is the quadratic control cost and Equation (7) clearly shows the free energy provides a lower bound for the optimal control problem. Hence the core idea of IT-MPPI is to finding the optimal control distribution $\mathbf{U}^*$ that minimizes the quadratic minimization problem

$$\mathbf{U}^* = \arg\min_{\mathbf{U}\in\mathcal{U}} \mathbb{E}_{\mathbb{Q}*}\left[(\mathbf{V}-\mathbf{U})^\top \Sigma^{-1}(\mathbf{V}-\mathbf{U})\right] \tag{8}$$

where $\mathbb{Q}^*$ is the optimal control distribution in the sense that it achieves the lower bound [1].

Williams et al. [5] solved Equation (8) using **importance sampling** where from a Monte Carlo simulation of K samples, the weight of the k^{th} sequence, $\mathbf{V}_k$ can then be found as

$$w(\mathbf{V_k}) = \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}\left(S(\mathbf{V_k}) + \lambda(\hat{\mathbf{U}} - \widetilde{\mathbf{U}})^\top \Sigma^{-1}\mathbf{V}\right)\right) \tag{9}$$

where $\widetilde{\mathbf{U}}$ is the base distribution of control inputs and $\hat{\mathbf{U}}$ is a nominal input that for practical purposes, contains the history of last optimal control inputs and λ is **inverse temperature variable** that controls exploration vs exploitation of control sequence samples. Finally, the Equation (6) then takes the form

$$\mathbf{U}^* = \mathbb{E}_{\mathbb{Q}^*}[w(\mathbf{V_k})\mathbf{V_k}] \tag{10}$$

where Equation (10) gives the optimal information-theoretic control policy at the current timestep t

B. Kinematic model

Figure 1 shows the kinematic diagram [13] of a differential robot with the assumption that wheels on each side of the robot may be grouped as one wheel the same assumption depicted in ideal skid-steer drive robots [10].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{rcos(\theta)}{2} & \frac{rcos(\theta)}{2} \\ \frac{rsin(\theta)}{2} & \frac{rsin(\theta)}{2} \\ -\frac{r}{s} & \frac{r}{s} \end{bmatrix} \begin{bmatrix} \dot{\psi}_l \\ \dot{\psi}_r \end{bmatrix} \tag{11}$$

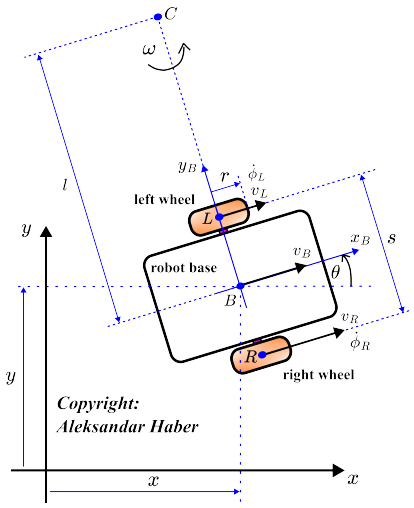

Fig. 1: Ideal differential drive kinematic model.

Here, x_B, y_B (m) are the coordinate of robot's CoM B in global frame X, Y. θ is its yaw (radians) and ω is the robot's angular velocity measured in the global coordinate frame. $\dot{\psi}_l, \dot{\psi}_r$ are the left and right angular wheel velocities (rad/s), v_B is the robot's linear velocity and r and s are respectively the radius and wheel center-to-center distances. Additionally, the relation between v_b, ω_b and the angular velocities of wheels can be compactly represented as

$$\begin{bmatrix} \dot{\psi}_l \\ \dot{\psi}_r \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ -\frac{r}{s} & \frac{r}{s} \end{bmatrix}^{-1} \begin{bmatrix} v_B \\ \omega \end{bmatrix}. \tag{12}$$

C. Implementation

We define the state vector $\zeta_t = [x_B, y_B, \theta]^T$ and control inputs as body twist [14] defined as $u_t = [v_B, \omega]^T$. For sake of simplicity, current timestep t is not appended with these notations. For an input u_t at timestep t, Equation (12) is used to solve for the wheel velocities and then from Equation (11) rate of changes in the three states re obtained. This is then fed into the Forward Kinematics model as shown in Equation (13) to obtain new state ζ_{t+1} using Euler integration where Δ_t is the predefined length between two timesteps.

$$\begin{aligned} x_{t+1} &= x_t + \dot{x}\Delta_t \\ y_{t+1} &= y_t + \dot{y}\Delta_t \\ \omega_{t+1} &= \omega_t + \dot{\omega}\Delta_t \end{aligned} \tag{13}$$

The high-level steps to implement IT-MPPI based control of the above 2D skid-steered robot is shown below

1) Step 1: Randomly sample K input sequences $V_k \mid k = \{1, 2, \cdots, K\}$.
2) Step 2: Predict future states and evaluate cost for each sample $S(V_k)$
3) Calculate weight for each sample sequence $w(V_k)$
4) Solve $\mathbf{U}^*$ using Equation (10)
5) Apply optimal control u_t^*, the first input in the optimized sequence.
6) Preserve $\mathbf{U}^*(2:K)$ for next timestep.

D. Experiment

The Python simulation environment was adopted from Mizuho Aoki's **simple_python_mppi** project [11]. However, extensive modifications have been made as part of this project, chief among them are:

1) Addition of Ideal Differential Drive kinematics model
2) Addition of new trajectory generation and testing framework
3) Numba [15] accelerator to speed up computing some of the steps.

To assign a weight to a sampled sequence, the staging cost was formulated as a simple quadratic function augmented with a penalty term if the robot's trajectory causes collision with an circular obstacle shown below

$$\begin{aligned} S(\mathbf{V_k}) = \sum_i^H \Big(& W_1(x_{ref} - x_{t+i})^2 \\ & + W_2(y_{ref} - y_{t+i})^2 \\ & + W_3(\omega_{ref} - \omega_{t+i})^2 \Big) \\ & + \mathcal{G}_c(\hat{\zeta}_{i=0}, \mathbf{O}) C_{penalty} \end{aligned} \tag{14}$$

where $W_i | i = [1, 2, \cdots, n]$ are predefined weights, $\mathbb{G}_c$ is a function that takes in the first predicted state vector $\hat{\zeta}_{i=0}$ of $\mathbf{V}_k$, $\mathbb{O}$ is a matrix that defines the position and radii of circular obstacles and $C_{penalty} = 1e^{10}$ is a very large penalty term.

Figure 2 shows the **oval** and **figure-eight** trajectories with circular obstacles tested in this experiment. Note that the heading angle (yaw) was normalized between 0 to 2π radians.

The simulation parameters for the experiment were set as follows: $\Delta_t = 0.1$ s, $K = 100$, prediction horizon $H = 20$, input constraints: $-5.0 \leq v_B \leq 5.0$ m/s, $-1.5 \leq \omega \leq 1.5$ rad/s. Objective function weights and covariance of Gaussian noise used are presented below

TABLE I: Weight and covariance matrix for two paths

Name	Weights	Covariance matrix
Oval	$[80, 1, 80]$	$\begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$
Figure-eight	$[100, 20, 60]$	$\begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$

All simulations where conducted on a computer with Ubuntu 22.04, Ryzen 5600x @ 3.2GHz, NVIDIA RTX 3080 GPU, and 32 GB RAM.

III. Results & Discussion

A. Results

Figure 2 shows the path taken by the 2D in comparison to the reference trajectory. Table II shows the root mean squared error for the robot's trajectory along with key simulation metrics.

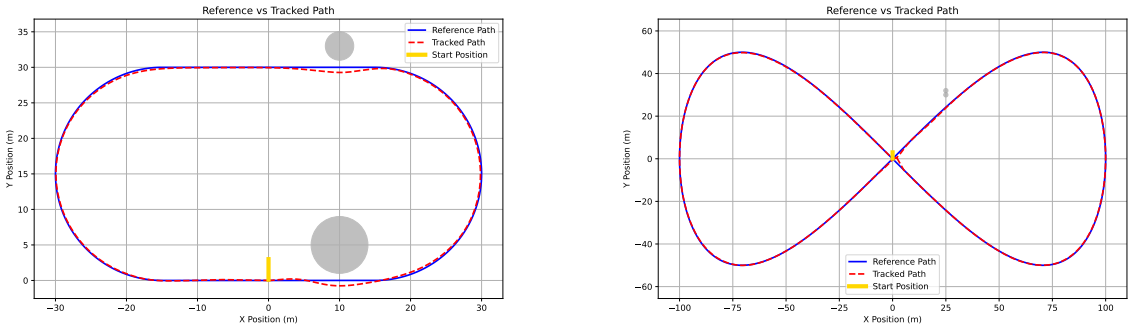

Fig. 2: Tracked trajectory (red) compared to the reference trajectory (blue) for oval and figure-eight paths. Zoom in for better view.

TABLE II: Results

Metric	Oval	Figure-8
RMSE_X (m)	0.85	0.32
RMSE_Y (m)	0.24	0.39
RMSE_ω (radians)	1.69	5.38
Simulation Steps (int)	760	1354
Mean v_B (m/s)	2.02	4.5
Mean ω (rad/s)	0.08	0.007

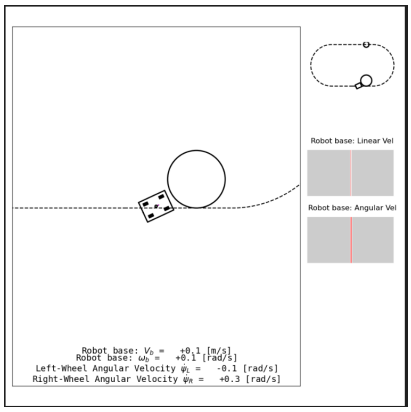

Fig. 3: Emergent behavior also demonstrating usage of previous timestep trapping robot into a “zero“ minima

B. Discussion

From Table II the RMSE error in all three dimensions states demonstrates high performance. From Figure 2 it is clear that the differential drive kinematics and the MPPI controller were implemented correctly since the robot was able to complete both trajectories. Additionally, the robot demonstrated correct maneuver around circular obstacles and in the direction of least required move thereby demonstrating the effectiveness of information-theoretic sampling even with as little as 100 samples. This is shown in the collage in Figure 4.

Figure 5 shows the evolution of optimal control action over time. In both cases no action cross 0.4 rad/s indicating the robot mostly held true to the reference trajectory. The larger spikes may be explained directional changes when avoiding obstacles or going through a circular section.

The MPPI controller exhibited an ”emergent” behavior, coming to a complete stop instead of colliding head-on with an obstacle, as shown in Figure 3. However, this behavior highlights a limitation of IT-MPPI: it can become trapped in a local minimum due to sampling exclusively from a unimodal Gaussian distribution centered on the previous distribution. In this scenario, since the previous distribution is nearly zero, the robot halts entirely instead of attempting to maneuver around the obstacle.

The average execution time per timestep is 0.236 seconds. This was expected as MPPI while accommodating non-convex objective functions and constraints requires a large number of Monte Carlo simulation to find the optimal control action. However, in practice, this problem has been solved with the help of parallel processing on GPU that enables real-time control of high speed dynamical systems [2], [7].

IV. Conclusion

This project investigated effectiveness of an Information-Theoretic Model Predictive Path Integral controller in evaluating the path-tracking performance of 2D Unmanned Ground Robot modeled as an ideal-differential drive kinematic system. Extensive testing an oval and a challenging figure-eight path revealed average RMSE values of 0.58 m, 0.32 m and 3.53 radians respectively for X, Y and yaw axes. The average linear and angular body velocities were found to be 3.26 m/s and 0.0435 rad/s respectively. The average execution time per timestep was 0.236 seconds. The controller not only demonstrated effective path tracking but also accurate obstacle avoidance at high speeds without requiring complex objective or constraint functions.

Future work may focus on parallelizing trajectory rollouts on a GPU using projects like CuPY [16], employ usage of multimodal arbitrary distributions coupled with ancillary controllers [1] or improving the mean and covariance of the Gaussian distribution using feedback from the system samples [3] to mitigate the limitation of getting trapped in local minima

ACKNOWLEDGMENT

The author would like to express sincere gratitude to Dr. Xiangyu Meng for his guidance and insights throughout the course of this work. Special thanks are extended to Mizuho

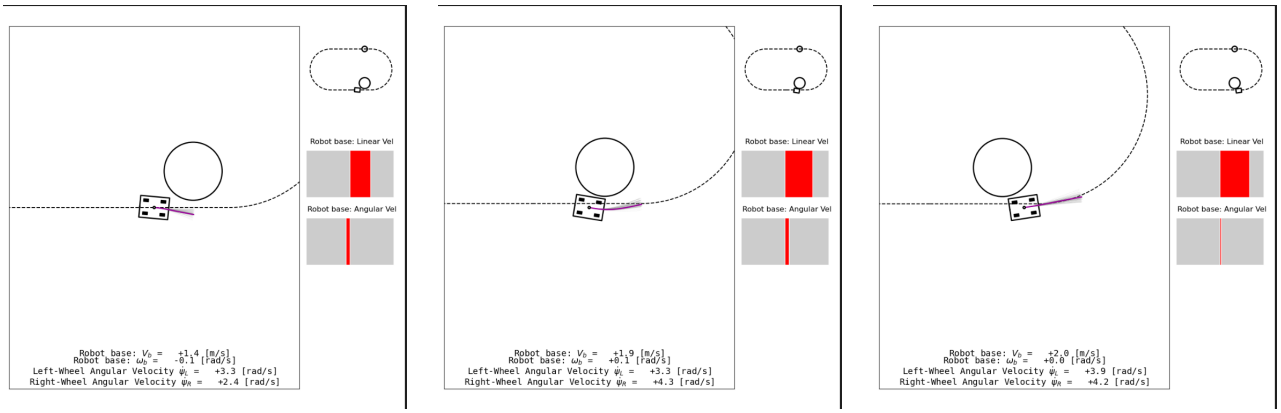

Fig. 4: Optimal path plan demonstrating obstacle avoidance. Path in purple indicates optimal trajectory while path in grey indicates the paths that were sampled but rejected by the update law. Zoom in for better view

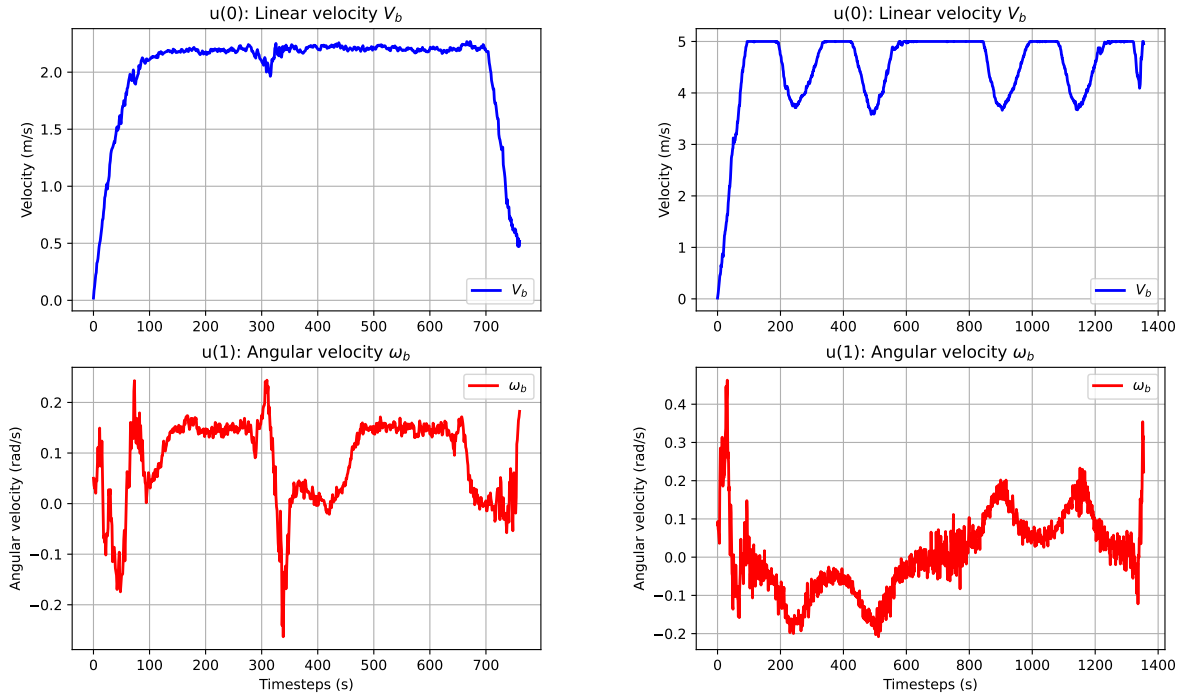

Fig. 5: Optimal control actions over time. Left column is for oval and right column is for figure-8 paths.

Aoki for providing the foundational MPPI implementation that was extensively modified for this project.

REFERENCES

[1] E. Trevisan and J. Alonso-Mora, "Biased-mppi: Informing sampling-based model predictive control by fusing ancillary controllers," *IEEE Robotics and Automation Letters*, 2024.

[2] B. Vlahov, J. Gibson, M. Gandhi, and E. A. Theodorou, "Mppi-generic: A cuda library for stochastic optimization," 2024. [Online]. Available: https://arxiv.org/abs/2409.07563

[3] D. M. Asmar, R. Senanayake, S. Manuel, and M. J. Kochenderfer, "Model predictive optimized path integral strategies," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3182–3188.

[4] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent advances in quadratic programming algorithms for nonlinear model predictive control," *Vietnam Journal of Mathematics*, vol. 46, no. 4, pp. 863–882, 2018.

[5] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[6] I. S. Mohamed, G. Allibert, and P. Martinet, "Model predictive path integral control framework for partially observable navigation: A quadrotor case study," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2020, pp. 196–203.

[7] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation*

(ICRA). IEEE, 2016, pp. 1433–1440.

[8] R. Khan, F. M. Malik, A. Raza, and N. Mazhar, "Comprehensive study of skid-steer wheeled mobile robots: Development and challenges," *Industrial Robot: the international journal of robotics research and application*, vol. 48, no. 1, pp. 142–156, 2021.

[9] A. Trivedi, S. Prajapati, A. Shirgaonkar, M. Zolotas, and T. Padir, "Data-driven sampling based stochastic mpc for skid-steer mobile robot navigation," *arXiv preprint arXiv:2411.03289*, 2024.

[10] D. Baril, V. Grondin, S.-P. Deschênes, J. Laconte, M. Vaidis, V. Kubelka, A. Gallant, P. Giguere, and F. Pomerleau, "Evaluation of skid-steering kinematic models for subarctic environments," in *2020 17th Conference on Computer and Robot Vision (CRV)*. IEEE, 2020, pp. 198–205.

[11] M. Aoki, "python_simple_mppi," https://github.com/MizuhoAOKI/python_simple_mppi, 2023, accessed: 2024-12-07.

[12] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1423–1430, 2021.

[13] A. Haber, "Clear and detailed explanation of kinematics equations and geometry of motion of differential wheeled robot (differential drive robot)," n.d., accessed: 2024-12-08.

[14] K. Lynch, *Modern Robotics*. Cambridge University Press, 2017.

[15] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.

[16] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, "Cupy: A numpy-compatible library for nvidia gpu calculations," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. [Online]. Available: http://learningsys.org/nips17/assets/papers/paper_16.pdf