Model Predictive Path Integral Control With Discrete Control Barrier Function

Anubhav Vishwakarma

Abstract— **This project addresses the problem of path tracking with collision avoidance using a gradient-free optimization approach: Model Predictive Path Integral (MPPI) Control, combined with a Discrete Control Barrier Function (CBF) incorporated as part of the rollout cost. The proposed method is validated in a simple environment with a single obstacle and a quadrotor agent. Performance is compared against a baseline LQR controller and MPPI without the CBF safety filter.**

I. Introduction

Autonomous vehicles are poised to become the new norm in the near future, and ensuring that they can reach their destinations both quickly and safely is of paramount importance. However, achieving reliable autonomous navigation is a complex challenge due to several inherent difficulties present in real-world environments. These challenges include:

- **High-dimensional state spaces:** The state space of an autonomous vehicle typically includes positions, velocities, orientations, sensor readings, and potentially the states of surrounding dynamic agents. This high dimensionality renders traditional dynamic programming approaches, such as solving the Hamilton-Jacobi-Bellman (HJB) equation, computationally intractable for real-time deployment. Methods like the Linear Quadratic Regulator (LQR), which rely on simplifying assumptions and low-dimensional representations, are often not suitable for such complex settings, especially when dynamics are nonlinear and interactions with the environment are rich- in a sense that offline policy will fail due to stochasticity in the environment.
- **Non-convex costs and constraints:** Real-world navigation tasks often involve non-convex cost functions (e.g., optimizing for energy efficiency while minimizing travel time) and non-convex constraints (e.g., obstacle avoidance, dynamic safety limits). These characteristics make the problem difficult to solve using standard convex optimization techniques. To address this, iterative methods that approximate the problem as a sequence of convex sub-problems are commonly used. Examples include Model Predictive Control (MPC) approaches leveraging iterative Linear Quadratic Regulator (iLQR) methods or Sequential Quadratic Programming (SQP). These algorithms work by linearizing the system dynamics and approximating the cost and constraint functions locally at each iteration, making it feasible to generate locally optimal control policies in real time.

Together, these challenges motivate the exploration of novel control and planning strategies that are both computationally efficient and capable of handling the non-convex, high-dimensional nature of autonomous driving environments.

A. Related Work

Before proceeding further, it is important to define the key concepts that this work builds upon.

Model Predictive Control (MPC): MPC is a receding horizon control approach that iteratively solves an optimal control problem over a finite horizon. It does so by linearizing the system dynamics and convexifying the cost and constraints. At each time step, only the first control input from the optimized sequence is applied to the system, and the process is repeated until the task is completed.

However, this iterative convexification can fall short when the underlying problem is highly non-convex. To address this, one must consider approaches that do not rely on convexity assumptions. Additionally, MPC has difficulty incorporating discontinuous cost functions, which are common in real-world applications.

Model Predictive Path Integral (MPPI) Control: MPPI is an extension of MPC that also solves an optimal control problem over a finite horizon, but in a gradient-free manner. It optimizes the cost via sampling, without requiring quadratic cost structures or smooth constraints as in methods like SQP or iLQR.

More specifically, MPPI seeks to minimize the distributional shift between an expert distribution (typically defined as an exponential over a reference trajectory) and a set of sampled trajectories using importance sampling—a Monte Carlo method that estimates expectations using a weighted average of samples drawn from a known distribution (usually Gaussian).

To elaborate, the sampled trajectories are generated by unrolling the system dynamics from an initial state using a sequence of control inputs perturbed with Gaussian noise. These controls are typically represented as a tensor of shape `[num_samples, horizon, control_dim]`.

Challenge with Control Barrier Functions (CBFs): Despite the advantages of MPPI, integrating CBFs into it presents challenges:

- CBFs require the barrier function to be continuous everywhere to certify safety.
- Computing the Jacobian of a neural network–based barrier function can be computationally expensive.

The equation below shows the continuous CBF condition for avoiding states in the unsafe set A [1]:

$$\max_{u \in \mathscr{U}} \nabla h(x)^T f(x,u) \geq -\alpha(h(x)), \quad \forall x \in A$$

This motivates the use of a Discrete Control Barrier Function (DCBF), which does not require gradient information.

Discrete Control Barrier Function (DCBF): The continuous CBF condition above can be reformulated for the discrete case as follows (see reference [2]):

$$B(x) < 0, \quad \forall x \in A,$$
$$B(x) \geq 0 \Rightarrow \max_{u \in U} [h(f(x,u)) - h(x)] \leq -\alpha(h(x))$$

II. Problem Formulation

The primary objective is for the quadcopter to follow a reference trajectory while safely avoiding obstacles. The dynamics of the planar quadcopter are defined as:

Planar Quadcopter Dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{y} \\ \dot{v}_y \\ \dot{\phi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_x \\ \frac{-(T_1+T_2)\sin\phi - C_D^v v_x}{m} \\ v_y \\ \frac{(T_1+T_2)\cos\phi - C_D^v v_y}{m} - g \\ \omega \\ \frac{(T_2-T_1)\ell - C_D^\phi \omega}{I_{yy}} \end{bmatrix} \tag{1}$$

The derivation of this model can be referenced in Figure [1], which shows the free-body diagram of the quadcopter.

Model Predictive Path Integral (MPPI) control seeks to minimize the Kullback–Leibler divergence between the optimal control distribution and the distribution of sampled control sequences. Formally:

$$U^* = \arg\min_{U \in \mathscr{U}} D_{\mathrm{KL}}(Q^* \parallel Q_{U,\Sigma})$$

where Q^* is the optimal trajectory distribution, and $Q_{U,\Sigma}$ is the distribution induced by sampling from a Gaussian density $q(V \mid U, \Sigma)$. The nominal control sequence $U \in \mathbb{R}^{m \times T}$ and sampled control sequences $V \in \mathbb{R}^{m \times T}$ are defined as:

$$(u_0, u_1, \ldots, u_{T-1}) = U, \quad (v_0, v_1, \ldots, v_{T-1}) = U + \varepsilon, \quad \varepsilon \sim \mathscr{N}(0, \Sigma)$$

As shown in [3], the optimal control sequence can be obtained by maximizing the log-likelihood of the sampled trajectories under the distribution $q(V \mid U, \Sigma)$:

$$U^* = \arg\max_{U \in \mathscr{U}} \mathbb{E}_{Q^*} [\log q(V \mid U, \Sigma)]$$

The Gaussian density is given by:

$$q(V \mid U, \Sigma) = Z^{-T} \exp\left(-\frac{1}{2} \sum_{t=0}^{T-1} (v_t - u_t)^\top \Sigma^{-1} (v_t - u_t) \right)$$

$$Z = ((2\pi)^m |\Sigma|)^{1/2}$$

According to [3], the optimal distribution over sampled trajectories is:

$$q^*(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} S(V) \right) p(V)$$

$$\eta = \int_{\mathbb{R}^{m \times T}} \exp\left(-\frac{1}{\lambda} S(V) \right) p(V) \, dV$$

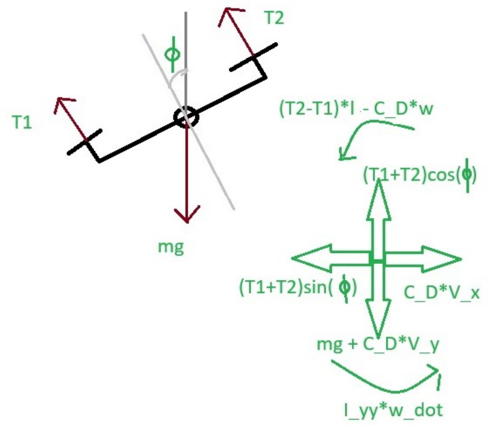

Fig. 1. Free body diagram and annotation of forces and momentum

where $S(V)$ is the trajectory cost, and $p(V) = q(V \mid \tilde{U}, \Sigma)$ is a base distribution centered at a previous nominal control $\tilde{U}$.

The solution to this optimization yields:

$$u_t^* = \mathbb{E}_{Q^*}[v_t], \quad \forall t \in \{0, 1, \ldots, T-1\}$$

Since sampling directly from Q^* is intractable, we employ importance sampling using the known distribution $Q_{\hat{U},\Sigma}$:

$$\mathbb{E}_{Q^*}[v_t] = \int \underbrace{\frac{q^*(V)}{q(V \mid \hat{U}, \Sigma)}}_{w(V)} q(V \mid \hat{U}, \Sigma) \, v_t \, dV$$

$$\approx \mathbb{E}_{Q_{\hat{U},\Sigma}}[w(V) v_t], \quad w(V) = \frac{q^*(V)}{q(V \mid \hat{U}, \Sigma)}$$

Substituting $q^*(V)$ into the weight expression yields:

$$w(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} \left(S(V) + \lambda \sum_{t=0}^{T-1} u_t^\top \Sigma^{-1} v_t \right) \right)$$

From this expression, it can be seen that the normalized importance weights resemble a softmax function applied over the negative cost of each trajectory. This form ensures that lower-cost rollouts are exponentially more influential during control updates.

III. Proposed Solution

As we have seen till now how mppi tries to solve the optimal control optimization problem, which comprises of state cost and control cost which is the KL divergence between previous optimal/nominal control and perturbed control with white noise:

$$\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[S(V)] + \lambda D_{\mathrm{KL}}(\mathbb{Q}_{U,\Sigma} \parallel P)$$

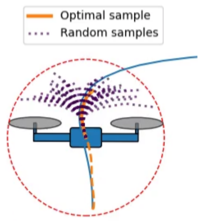

Fig. 2. MPPI Visualization

Algorithm 1 MPPI Algorithm for Path Tracking

```
Require:
  Number of trajectories to sample $K$;
  Planning horizon $H$;
  Number of optimization iterations $J$;
  Fixed noise sampling variance $\Sigma$
  Previous optimal plan $\mathbf{u}^*_{t-1} = \{(\mathbf{u}^*_{t-1})\}^{H}_{t'=0}$;
  Current state $x_t$;
  Dynamics model $f(x,u)$;
  Reference trajectory;

  while task not completed do
    $x_{\text{ref}} \leftarrow$ reference_trajectory
    for $j \leftarrow 0$ to $J-1$ do
      $\lambda \leftarrow \left(1 - \frac{1}{j+1}\right)$
      $\gamma \leftarrow \lambda \cdot (1-\alpha)$
      for $k \leftarrow 0$ to $K-1$ do
        $x \leftarrow x_0$
        $\varepsilon_k = \{\varepsilon^k_0, \dots, \varepsilon^k_{T-1}\}$, $\varepsilon^k_t \sim \mathcal{N}(0,\Sigma)$
        for $t \leftarrow 1$ to $H$ do
          $v_{t-1} \leftarrow u^*_{t-1} + \varepsilon^k_{t-1}$
          $x \leftarrow f(x, v_{t-1})$
          $S_k \mathrel{+}= c(x, x_{\text{ref}}) + \gamma \cdot u^T_{t-1}\Sigma^{-1} v_{t-1}$
        end for
      end for
      $Z \leftarrow \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda} S_k\right)$
      for $k \leftarrow 0$ to $K-1$ do
        $w^k \leftarrow \frac{1}{Z} \exp\left(-\frac{1}{\lambda} S_k\right)$
      end for
      for $t \leftarrow 0$ to $H-1$ do
        $u^{*j}_t \leftarrow u^{*j}_t + \sum_{k=0}^{K-1} w^k \cdot \varepsilon^k_t$
      end for
    end for
  end while
```

- We start the optimization process using the previous nominal control sequence as the base. Noise is injected into it as $\varepsilon^k = \{\varepsilon^k_0, \dots, \varepsilon^k_{T-1}\}$, where $\varepsilon^k_t \sim \mathcal{N}(0,\Sigma)$. This results in sampled control sequences $v_{t-1} = u^*_{t-1} + \varepsilon^k_{t-1}$, which are then passed through the dynamics $f(x, v_{t-1})$ to generate samples of the next states.
- These next states are used to compute the trajectory cost based on the deviation from the reference trajectory and the value of the barrier function. More positive values of the barrier function indicate safer next states.
- Additionally, the term $\gamma \cdot u^{*\top}_{t-1}\Sigma^{-1}v_{t-1}$ estimates the divergence of the sampled control from the nominal control. Larger deviations result in higher costs, discouraging overly exploratory or unsafe behaviors.
- Temperature plays a crucial role in balancing exploration and exploitation. At the beginning of the iterations, a higher temperature encourages exploration to discover promising controls. As iterations proceed, the temperature is decreased, leading the controller to exploit the best trajectories.
- After optimizing the control sequence over several iterations, the cost associated with each k^{th} sample is used to compute softmax-like weights. These weights determine the relative importance of each trajectory.
- The final control update is computed as a weighted sum of the injected noise, emphasizing lower-cost trajectories. This update is added to the previous nominal control sequence.
- After updating the optimal control sequence, the first control input is applied to the system, and the entire process is repeated, as in traditional MPC.

IV. Experimental Setup and Results

As this is a sampling-based method, one of the cornerstone questions is: what should the variance be for this highly agile quadrotor? Although sampling from a wider distribution allows for better exploration, sampling states too far from the goal may cause the robot to diverge and never converge to the optimal path. Hence, it is crucial to choose the covariance matrix Σ carefully.

For this problem, a variance in the range of $0.1 \cdot m \cdot g$ to $0.5 \cdot m \cdot g$ was found effective in finding the optimal control:

$$\Sigma = \begin{bmatrix} 0.5 \cdot m \cdot g & 0 \\ 0 & 0.5 \cdot m \cdot g \end{bmatrix}$$

One of the primary challenges was the system diverging from the reference path and failing to find an optimal solution. This occurred because a high weight was assigned to the cost computed via the barrier function. Consequently, if a high-variance sampled trajectory stayed far from the obstacle, it would receive a lower cost and be preferred — causing the robot to avoid obstacles excessively and diverge from the reference path.

A core objective of this work is computing an effective state cost function. Through experimentation, the following cost weightings were found to balance safety and tracking objectives:

$$\text{cost}_{\text{coeff}} = \{\text{CBF penalty} \times 5 + \text{Deviation} \times 2 + \text{Hover penalty} \times 1\}$$

A hover penalty was added to penalize control actions where the vertical thrust component fell below the quadrotor's weight.

The barrier function used is a Euclidean distance-based function between the robot and the obstacle centers, incorporating a safety padding given by:

$$r = \text{robot\_radius} + \text{obstacle\_radius}$$

For the control-invariant set formulation, a class-K function α was chosen as a linear, monotonically increasing function. The constant $a = 0.5$ was found to be a good balance for constraining the control input. This understanding is supported by the results observed in HW1 problem 5 [1], Figure [5].

Discrete Control Barrier Function:[2]

$$h(f(\mathbf{x},\mathbf{u})) - h(\mathbf{x}) \geq \alpha(h(\mathbf{x})), \quad \text{where} \quad \alpha(h(\mathbf{x})) = a \cdot h(\mathbf{x})$$

$$\text{D-CBF} = h(f(\mathbf{x},\mathbf{u})) - (1+a)h(\mathbf{x})$$

$$h(x) = (x - x_o)^2 + (y - y_o)^2 - r^2$$

The results (Figure [3] and Figure [4]) clearly demonstrate the strength of sampling-based optimization by comparing the tracking performance of MPPI and LQR. MPPI, when augmented with the CBF cost, effectively avoids high-cost regions near obstacles. However, because more weight is given to the CBF term than to the deviation term, MPPI with CBF shows a higher deviation from the reference trajectory compared to MPPI using only the deviation cost.

On the other hand, LQR performed suboptimally, even when collision constraints were incorporated while solving the Riccati equation.

V. Conclusions and Future Work

- This work does not consider the terminal value of the sampled trajectory. A more robust way to implement MPPI would be to inform each sampled rollout with the value function, which could be calculated using the BRT (problem: curse of dimensionality) or learned.
- Learn the Control Barrier Function from expert demonstrations, which would be easily incorporated with DCBF.
- It has been found that even after applying the convolution filter to the control before updating the optimal control (i.e., $w * \varepsilon$), this does not lead to an improved trajectory. It has been concluded that this phenomenon could be due to the shorter horizon.

References

[1] K. Leung, "Lecture material aa/me/ee 548 2025," *University of Washington*, 2025.

[2] E. Y. Y. C. F. P. T. Ji Yin, Oswin So, "Safe beyond the horizon: Efficient sampling-based mpc with neural control barrier functions," *arXiv preprint arXiv:2502.15006*, 2025.

[3] G. I. o. T. A. G. U. . E. A. T. Grady Williams; Paul Drews; Brian Goldfain; James M. Rehg School of Interactive Computing, College of Computing, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, 2018.

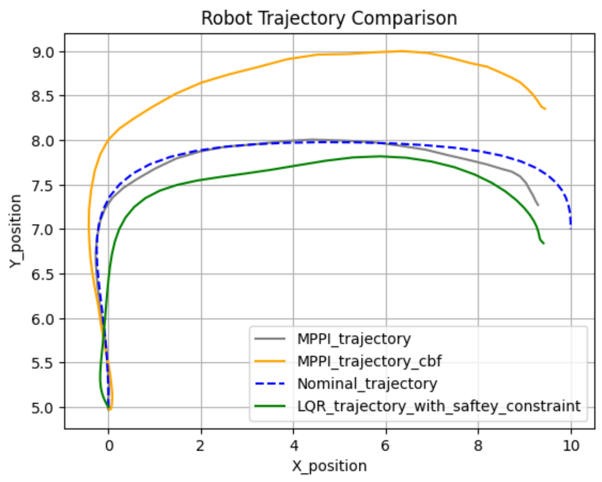

Fig. 3. Trajectory of differenc approach

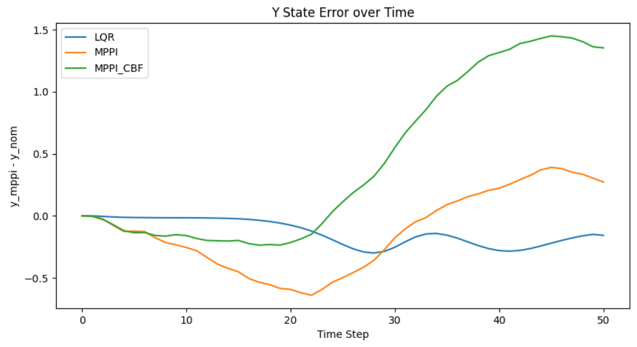

Fig. 4. Error over Y axis for different approach

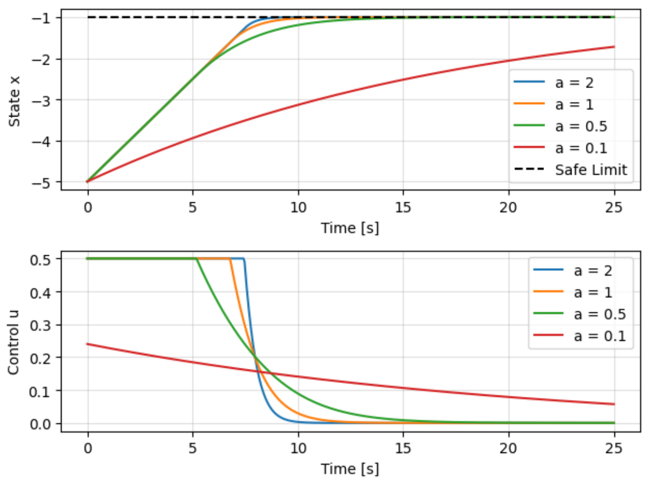

Fig. 5. K class function constant tradeoff