A United Framework of Controller and Behavior Planner: Iterative-Regrouping MPPI

Yihan Li

October 8, 2025

1 Introduction

Sampling-based MPC methods like MPPI(Model Predictive Path Integral) has been widely used in trajectory generating problems recently due to its flexibility and interference resistance, especially in scenarios like autonomous racing. However, with the existence of obstacle vehicles, different behavior tendencies could appear in the overtaking process, and the zig-zag between different behavior tendencies in different time steps could result in stuckness in such scenarios. Therefore, this project focuses on making behavior planning in car racing scenarios via a model-based method by constructing a united framework of controller and behavior planner through MPPI with regrouping policy during the iterative sampling process. The proposed framework aims on working out the overtaking tendency of the ego vehicle and build a close-loop grouping and warm-start logic, which is also a pre-stage work for constructing a model-based interactive framework which can be implied in scenarios involving multiple ego vehicles.

2 Methods

In this section, the proposed Iterative-Regrouping MPPI will be introduced step-by-step starting by the initial formulation of the MPPI method[1].

2.1 MPPI Formulation

MPPI assumes that we do not have direct control over the input variable v_t but rather that v_t is a random vector generated by a white-noise process with density function:

$$\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}_t, \Sigma) \tag{1}$$

Define the input sequences and the related mean value as:

$$\begin{aligned}(\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{T-1}) &= V \in \mathbb{R}^{m\times T},\\ (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}) &= U \in \mathbb{R}^{m\times T}.\end{aligned} \tag{2}$$

Then we define the probability density function for V as:

$$q(V \mid U, \Sigma) = Z^{-T} \exp\left(-\frac{1}{2}\sum_{t=0}^{T-1}(\mathbf{v}_t - \mathbf{u}_t)^\top \Sigma^{-1}(\mathbf{v}_t - \mathbf{u}_t)\right). \tag{3}$$

where $Z = ((2\pi)^m|\Sigma|)^{\frac{1}{2}}$. Then the optimization problem is given as:

$$U^* = \arg\min_{U\in\mathcal{U}} \mathbb{E}_{Q_{U,\Sigma}}\left[\phi(\mathbf{x}_T) + \sum_{t=0}^{T-1}\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t)\right]. \tag{4}$$

where $\phi(\mathbf{x}_T)$ is the terminal cost, and

$$\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) = c(\mathbf{x}_t) + \frac{\lambda}{2}\left(\mathbf{u}_t^\top \Sigma^{-1}\mathbf{u}_t + \boldsymbol{\beta}_t^\top \mathbf{u}_t\right). \tag{5}$$

$c(\mathbf{x}_t)$ here is the state-dependent cost, and $\frac{\lambda}{2}\left(\mathbf{u}_t^\top \Sigma^{-1}\mathbf{u}_t + \boldsymbol{\beta}_t^\top \mathbf{u}_t\right)$ here is the control cost. Defining the free-energy of the system and the KL-Divergence between the base distribution and the nominal distribution as:

$$\mathcal{F}(S, p, \mathbf{x}_0, \lambda) = -\lambda \log\left(\mathbb{E}_{\mathbb{P}}\left[\exp\left(-\frac{1}{\lambda}S(V)\right)\right]\right). \tag{6}$$

$$\mathbb{D}_{\mathrm{KL}}\left(Q_{U,\Sigma} \,\|\, Q_{\bar{U},\Sigma}\right) = \frac{1}{2}\sum_{t=0}^{T-1}\left(\mathbf{u}_t^\top \Sigma^{-1}\mathbf{u}_t + \boldsymbol{\beta}_t^\top \mathbf{u}_t + c_t\right). \tag{7}$$

We could turn the optimization problem into:

$$U^* = \arg\min_{U\in\mathcal{U}}\left(\mathbb{E}_{\mathbb{Q}^*}\left[\sum_{t=0}^{T-1}(\mathbf{v}_t - \mathbf{u}_t)^\top \Sigma^{-1}(\mathbf{v}_t - \mathbf{u}_t)\right]\right). \tag{8}$$

And we can get the optimal input by:

$$\mathbf{u}_t^* = \mathbb{E}_{\mathbb{Q}^*}[\mathbf{v}_t] \,\forall t \in \{0, 1, \ldots, T-1\}. \tag{9}$$

Then the optimal input could be fetched through importance sampling:

$$\begin{aligned} w(V) &= \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}\left(S(V) + \lambda\sum_{t=0}^{T-1}(\hat{\mathbf{u}}_t - \tilde{\mathbf{u}}_t)^\top \Sigma^{-1}\mathbf{v}_t\right)\right),\\ \mathbf{u}_t &= \mathbb{E}_{Q_{\hat{U},\Sigma}}\left[w(V)\mathbf{v}_t\right].\end{aligned} \tag{10}$$

The following steps of the proposed method is based on this basic formulation of MPPI.

2.2 Iterative MPPI

Based on the basic formulation of MPPI, the sampling process can also be performed in iterative manners.[2] In this method, the sampling process is performed for several cycles till the trajectory converges or the iteration number reaches the upper limit. At the end of every iteration cycle in the same time step, the mean value and the covariance matrix are updated according to the sampling result and trajectory cost for the next sampling cycle. The full algorithm is shown in Algorithmn 1.

Algorithm 1 Model Predictive Optimized Path Integral Control

$\mathbf{F}, \mathbf{g}$: System state transition model
K: Number of samples
T: Prediction horizon length
L: Maximum number of AIS iterations
$\mathbf{U} \in \mathbb{R}^{mT}, \Sigma \in \mathbb{R}^{m \times mT}$
ϕ, c, λ, α: Cost function/parameters $\psi_1, \psi_2, \ldots, \psi_p$:AIS parameters

$\mathbf{x}_0 \leftarrow$ GetStateEstimate()
$\mathbf{U}' \leftarrow \mathbf{U}, \Sigma' \leftarrow \Sigma$
for $\ell \leftarrow 1$ to L **do**
  for $k \leftarrow 1$ to K **do**
    Sample $\mathcal{E}_k$ from $\mathcal{N}(0, \Sigma')$
    for $t \leftarrow 1$ to T **do**
      $\mathbf{x}_t \leftarrow \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{g}(\mathbf{u}'_{t-1} + \mathcal{E}^k_{t-1}))$
      $s_k \leftarrow c(\mathbf{X}) + \phi(\mathbf{X}) + \lambda(1-\alpha)\mathbf{U}'^{\top}\Sigma^{-1}(\mathcal{E}_k + \mathbf{U}' - \mathbf{U})$
    end for
  end for
  if $\ell < L$ **then**
    $\mathbf{U}', \Sigma' \leftarrow$ PerformAIS$(\mathbf{U}', \Sigma', \mathbf{S}, \psi_1, \psi_2, \ldots, \psi_p, \ell)$
  end if
  $\rho \leftarrow \min(\mathbf{S})$
  $\eta \leftarrow \sum_{k=1}^{K} \exp\left(-\frac{1}{\lambda}(s_k - \rho)\right)$
  for $k \leftarrow 1$ to K **do**
    $w_k \leftarrow \frac{1}{\eta}\exp\left(-\frac{1}{\lambda}(s_k - \rho)\right)$
    $\mathbf{U} \leftarrow \mathbf{U} + w_k(\mathcal{E}_k + \mathbf{U}' - \mathbf{U})$
  end for
end for
SendToActuators$(\mathbf{u}_0)$
for $t \leftarrow 1$ to $T-1$ **do**
  $\mathbf{u}_{t-1} \leftarrow \mathbf{u}_t$
end for
$\mathbf{u}_{T-1} \leftarrow$ Initialize$(\mathbf{u}_{T-1})$

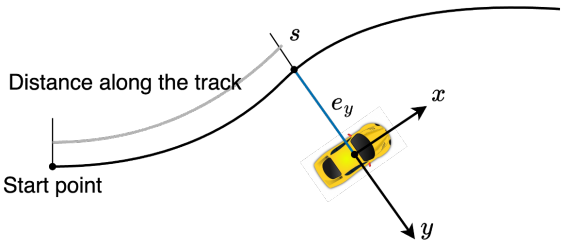


Figure 1: The coordinate used for plannings.

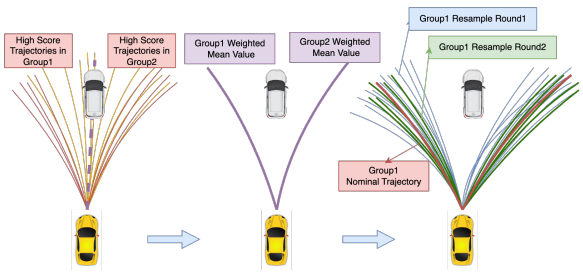


Figure 2: Demonstration of the open-loop regroup policy.

2.3 Iterative-Regrouping MPPI

With respect to the iterative MPPI, the regrouping policy is added to the racing strategies in this project, which is shown in Fig 2

In the first sampling cycle of every time step, based on the distribution of the low-cost trajectories, the proposed method separates these low-cost trajectories into different groups lying in the feasible region of the ego vehicle and estimates the mean value and covariance matrix of each group, in which the low-cost trajectories are defined as a user-defined portion of trajectories in the descending-sorted trajectory samples. The trajectory costs are made up of control cost and state cost. The control cost is the same as the basic formulation of MPPI, and the state cost includes the velocity reward, obstacle cost, track boundary cost and vibration cost. According to this setting, for example, in the scenario of one static obstacle, the distribution of the two groups would be on the both sides of the obstacle vehicle.

In the next iteration cycle, a resample process is started in every group based on the the mean value and covariance matrix calculated in last iteration, which will lasts till the trajectories in one group converge or hit the upper bound of the iteration number. The covariance matrix and mean value of each group is updated by cross-entropy method in each sampling cycle. An optimal trajectory then is chosen from the optimal trajectories of each group based on its vibration scale with respect to the optimal trajectory from last time step and their trajectory costs related to future states, actions and reachability.

In the next time step, the sampling is warm-started based on the optimal trajectory from the last time step and then repeat the process above. This resample policy could work out the behavior tendency, reduce the vibration and improve the coverage and optimality of the algorithm. The open-loop regrouping process is shown in Fig 2, and the whole close-loop logic is shown in Algorithmn 2

3 System Set-up

The planning in this system is done in curvilinear coordinate, and the system state and inputs are defined as the following:

$$x = \left[v_x, v_y, w_z, e_\psi, s, e_y\right]^\top \quad \text{and} \quad u = \left[\delta, a\right]^\top,$$

where w_z, v_x, v_y are the vehicle's yaw rate, longitudinal and lateral velocities, s is the distance the vehicle travelled along the centerline of the track, and e_ψ and e_y are the heading angle and lateral distance error between the vehicle and the centerline of the track, as shown in Fig 1. Finally, δ and a are the steering and acceleration commands.

The vehicle dynamics used for the roll-out from the sampled inputs to nominal states is:

$$x_{t+1} = Ax_t + Bu_t \tag{11}$$

The matrix A and B are from the regression of the history dataset of the vehicle states and inputs. The history dataset is obtained by running simple controllers on the vehicle in tracking tasks.

In the simulation part, the true physical dynamics is ap-

Algorithm 2 Model Predictive Path Integral Control with Grouping Strategy

Given: $\mathbf{F}, \mathbf{g}$: System state transition model
K: Number of samples
T: Prediction horizon length
L: Maximum number of AIS iterations
N, N': Number of groups for thermal diffusion and real-time splitting
$\mathbf{U} = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1}\} \in \mathbb{R}^{mT}$: Initial input sequence
$\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N$: Input sequences for thermal diffusion
$\Sigma \in \mathbb{R}^{mT \times mT}$: Covariance matrix
ϕ, c, λ, α: Related cost function parameters

while task not completed **do**
  $\mathbf{x}_0 \leftarrow$ GetStateEstimate()
  if regrouped in the last timestep **then**
    $\mathbf{U}'_1, \mathbf{U}'_2, \ldots, \mathbf{U}'_N \leftarrow \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N$
  else
    $\mathbf{U}', \mathbf{U}'_1, \mathbf{U}'_2, \ldots, \mathbf{U}'_N \leftarrow \mathbf{U}$
  end if
  for $\ell \leftarrow 1$ to L **do**
    if $\ell == 1$ **then**
      if regrouped in the last timestep **then**
        for $n \leftarrow 1$ to N **do**
          for $k \leftarrow 1$ to K/N **do**
            $\mathbf{x} \leftarrow \mathbf{x}_0$
            Sample $(\epsilon_0^k, \ldots, \epsilon_{T-1}^k) \sim \mathcal{N}(0, \Sigma)$
            for $t \leftarrow 1$ to T **do**
              $\mathbf{u}_{t-1}^n \leftarrow \mathbf{u}_{t-1}^{n'} + \epsilon_{t-1}^k$
              $\mathbf{x} \leftarrow \mathbf{F}(\mathbf{x}, \mathbf{g}(\mathbf{u}_{t-1}^n))$
            end for
            $s_{(k-1)\frac{K}{N}+k} \leftarrow TrajctoryCost()$
          end for
        end for
      else
        for $k \leftarrow 1$ to K **do**
          $\mathbf{x} \leftarrow \mathbf{x}_0$
          Sample $\mathcal{E}_k = (\epsilon_0^k, \ldots, \epsilon_{T-1}^k) \sim \mathcal{N}(0, \Sigma)$
          for $t \leftarrow 1$ to T **do**
            $\mathbf{u}_{t-1} \leftarrow \mathbf{u}'_{t-1} + \epsilon_{t-1}^k$
            $\mathbf{x} \leftarrow \mathbf{F}(\mathbf{x}, \mathbf{g}(\mathbf{u}_{t-1}))$
          end for
          $s_k \leftarrow TrajctoryCost()$
        end for
      end if
    else
      for $n \leftarrow 1$ to N' **do**
        if group not converged **then**
          for $k \leftarrow 1$ to K/N' **do**
            Similar computation as above
          end for
        end if
        Perform convergence check and adjust $\mathbf{U}'$
      end for
    end if
  end for
  $\mathbf{U} \leftarrow$ SelectOptimalTrajectory$(\mathbf{U}'_1, \mathbf{U}'_2, \ldots, \mathbf{U}'_N)$
  SendToActuators$(\mathbf{u}_0)$
  for $t \leftarrow 1$ to $T-1$ **do**
    $\mathbf{u}_{t-1} \leftarrow \mathbf{u}_t$
  end for
  $\mathbf{u}_{T-1} \leftarrow$ Initialize$(\mathbf{u}_{T-1})$
end while

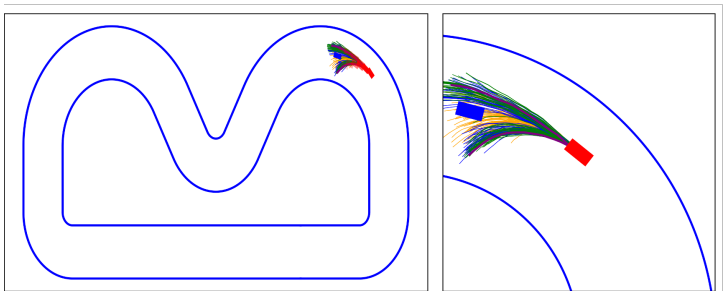


Figure 3: Open-loop Iterative-regrouping MPPI test result, where different colors in the sampled trajectories represents different cycles of samples.

plied as the following[3]:

$$\dot{e}_\psi = w_z - \frac{v_x \cos(e_\psi) - v_y \sin(e_\psi)}{1 - \kappa(s) e_y} \kappa(s), \tag{12}$$

$$\dot{s} = \frac{v_x \cos(e_\psi) - v_y \sin(e_\psi)}{1 - \kappa(s) e_y}, \tag{13}$$

$$\dot{e}_y = v_x \sin(e_\psi) + v_y \cos(e_\psi), \tag{14}$$

which is discretized as:

$$e_{\psi_{k+1}} = e_{\psi_k} + dt \left(w_{z_k} - \frac{v_{x_k} \cos(e_{\psi_k}) - v_{y_k} \sin(e_{\psi_k})}{1 - \kappa(s_k) e_{y_k}} \kappa(s_k) \right), \tag{15}$$

$$s_{k+1} = s_k + dt \left(\frac{v_{x_k} \cos(e_{\psi_k}) - v_{y_k} \sin(e_{\psi_k})}{1 - \kappa(s_k) e_{y_k}} \right), \tag{16}$$

$$\dot{e}_y = e_{y_k} + dt \left(v_{x_k} \sin(e_{\psi_k}) + v_{y_k} \cos(e_{\psi_k}) \right). \tag{17}$$

where $\kappa(s)$ is the curvature of the centerline of the track at the curvilinear abscissa s and dt is the simulation time step.

4 Results

4.1 MPPI vs Iterative MPPI in Tracking

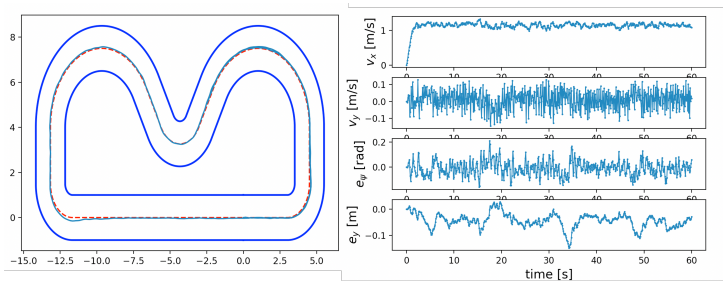
The tracking task is first used to test the MPPI and iterative MPPI controller to figure out the effect of adding iterative sampling strategy and to find evidence to support the following iterative structure in Iterative-regrouping MPPI.

In Fig 4, compare Fig 4a and Fig 4c, we know that by adding iteration cycles, the tracking error e_y can be reduced; Compare Fig 4b and Fig 4c, we know that when the sample number in every time step is fixed, using iterative structure instead of sampling all the trajectories once can help improve the behavior.

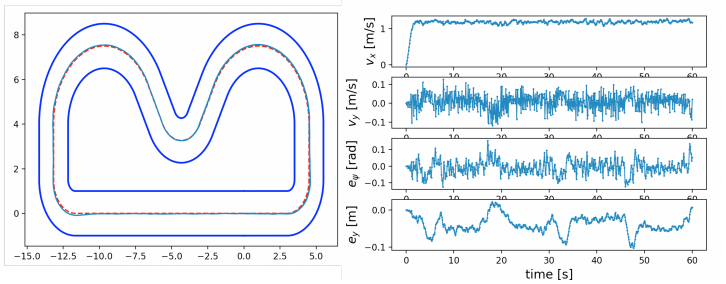
4.2 Overtaking Behavior in Iterative MPPI and Iterative-Regrouping MPPI

The open-loop regrouping result shown in Fig 3, which is related to Fig 2 in the algorithm. It shows that the open-loop regrouping process can successfully figure out the feasible regions of the ego vehicle with different behavior tendencies.

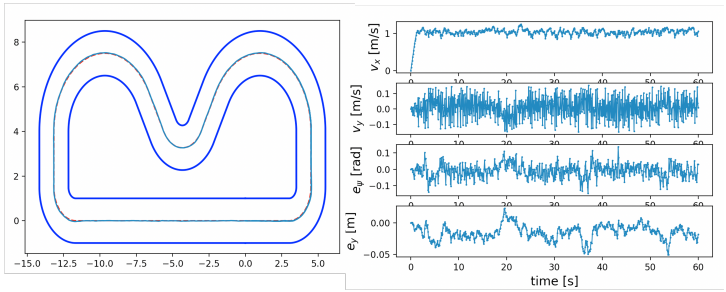
Fig 5 and Fig 3 shows the overtaking process of the ego vehicle with Iterative MPPI and Iterative-regrouping MPPI controllers with one static obstacle, which give a intuiitive sence of the effect of the regrouping policy in avoiding the stuckness in overtaking behaviors. Table 1 shows the overtaking success rate of different MPPI controllers in 50 tests, which shows that the regrouping policy can help improve the overtaking success rate.



(a) MPPI tracking with 100 samples in every time step.



(b) MPPI tracking with 100 samples in every time step.



(c) Iterative MPPI tracking with 100 samples in every sampling cycle and 3 cycles in each time step.

Figure 4: Centerline-tracking task results across MPPI and Iterative MPPI

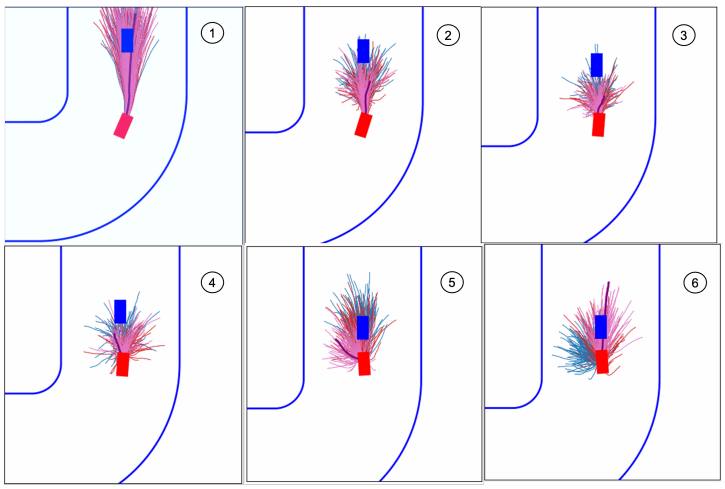


Figure 5: Close-loop Iterative MPPI overtaking snapshots, where different colors in the sampled trajectories represents different cycles of samples.

Table 1: Comparison of Controllers in Overtaking Test

Controller	Success	Failure	Success Rate
MPPI	22	28	44%
Iterative MPPI	29	21	58%
Iter-Regroup MPPI	32	18	64%

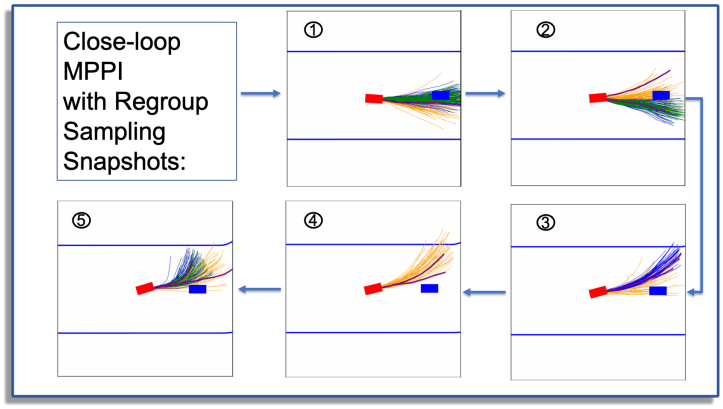


Figure 6: Close-loop Iterative-Regrouping MPPI overtaking snapshots, where, different colors in the sampled trajectories represents different cycles of samples.

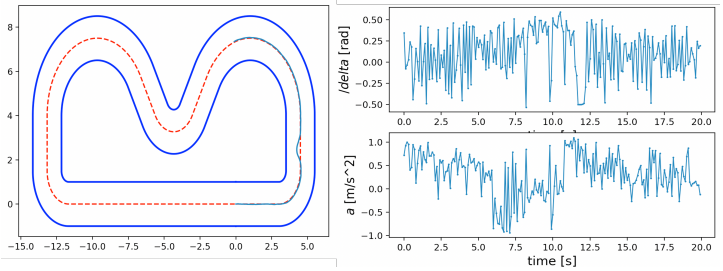


Figure 7: Notice how LaTeX automatically numbers this figure.

5 Conclusion and Discussion

Though the regrouping policy can help figure out the behavior tendency and improve the overtaking success rate, the improvement is not obvious now. And as shown in Fig 7, the vibration of the inputs calculated by iterative MPPI now is still obvious enven with low-pass filters, which could influence the robustness of the algorithm. In order to further solve this problems, more sampling numbers and iterations should be implied in future works on GPU, and different sampling strategies and covariance matrix update policies should be explored to improve the behavior. Furthermore, when the algorithm becomes robust enough with fine-tuning, the ego vehicle should be test in environments with randomly generated dynamic obstacles to perform the overtaking behavior.

References

[1] 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.

[2] 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)*, 2023, pp. 3182–3188.

[3] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: A predictive control approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2020.