SVG-MPPI: Enhancing Complexity Navigation In Autonomous Control

Kartik Virmani, Kaitian Chao, Kaihan Xu, Sebastian Tseng, Navjot Singh Chahal
Team 4
ESE 6150 - Autonomous Vehicle Racing
University of Pennsylvania
{kvirmani, chaokt, kaihanxu, sebastian, nschahal}@seas.upenn.edu

***Abstract*—This project investigates an advanced autonomous vehicle control strategy based on Stein Variational Guided Model Predictive Path Integral (SVG-MPPI) control. SVG-MPPI enhances the standard MPPI algorithm by incorporating mode-seeking behavior through a modified Stein Variational Gradient Descent (SVGD) method. This allows the controller to focus on a single, high-quality trajectory within a multimodal action distribution—crucial for tasks like precise path tracking and reliable obstacle avoidance. The approach retains MPPI's real-time, sampling-based optimization benefits while significantly improving performance in complex, high-speed navigation scenarios.**

I. Introduction

Traditional Model Predictive Control (MPC) methods—and their stochastic counterparts like Model Predictive Path Integral (MPPI)—are effective for short-term trajectory optimization but rely heavily on manually tuned hyperparameters such as the temperature parameter, control noise covariance, and prediction horizon. While MPPI is capable of evaluating and selecting low-cost trajectories from sampled control sequences, it struggles in dynamic or multimodal environments without continuous re-tuning. This limitation becomes especially problematic in time-critical and safety-sensitive applications like autonomous racing, where rapid changes in environment or task constraints can lead MPPI to average over multiple suboptimal solutions, reducing responsiveness and increasing collision risk.

To overcome this, SVG-MPPI introduces a mode-seeking mechanism that uses Stein Variational Gradient Descent (SVGD) to guide sampling toward the most promising solution mode. By adaptively focusing on a single trajectory and estimating a local covariance, SVG-MPPI improves robustness and responsiveness without requiring manual tuning during runtime.

II. Challenges with MPPI

Model Predictive Path Integral (MPPI) control is a sampling-based stochastic optimal control method that solves trajectory optimization problems without requiring gradient-based solvers or linearization of dynamics. Its simplicity and parallelizability make it appealing for real-time autonomous control. However, several core design assumptions in MPPI limit its effectiveness in complex or multimodal environments.

A. How MPPI works

Model Predictive Path Integral (MPPI) control is a stochastic optimal control method designed to handle nonlinear, non-differentiable systems. It models the system as a discrete-time dynamical process:

$$x_{\tau+1} = F(x_\tau, v_\tau), \quad \tau \in \{0, \dots, T-1\} \tag{1}$$

where:

$$x_\tau \in \mathbb{R}^n \quad \text{is the system state,} \tag{2}$$
$$v_\tau \in \mathbb{R}^m \quad \text{is the control input,} \tag{3}$$
$$F(\cdot) \quad \text{is the transition function,} \tag{4}$$
$$T \quad \text{is the planning horizon.} \tag{5}$$

The control inputs are modeled as samples from a Gaussian prior:

$$v_\tau \sim \mathcal{N}(u_\tau, \Sigma_\tau), \quad \Sigma_\tau = \mathrm{diag}(\sigma_\tau^0, \dots, \sigma_\tau^{m-1}) \tag{6}$$

MPPI then samples a set of K control trajectories $V = \{V_k\}_{k=0}^{K-1}$, where each trajectory is defined as a sequence of control inputs:

$$V_k = \{v_\tau^{(k)}\}_{\tau=0}^{T-1}$$

Each trajectory is evaluated using a cost function:

$$S(V_k) = \phi(x_T^{(k)}) + \sum_{\tau=0}^{T-1} c(x_\tau^{(k)}) \tag{7}$$

The optimal control sequence is updated using importance sampling with weights:

$$w(V_k) = \frac{\exp\left(-\frac{1}{\lambda} S(V_k)\right)}{\sum_{j=0}^{K-1} \exp\left(-\frac{1}{\lambda} S(V_j)\right)} \tag{8}$$

Finally, the control inputs are updated as a weighted average of the noise perturbations:

$$u_\tau \leftarrow u_\tau + \sum_{k=0}^{K-1} w(V_k) \cdot \epsilon_\tau^{(k)} \tag{9}$$

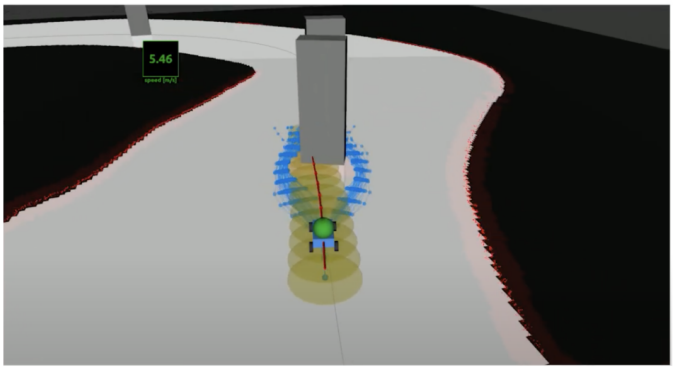
B. Core Limitation: Forward KL Divergence

MPPI implicitly minimizes the Forward Kullback-Leibler (FKL) divergence between the optimal action distribution Q* and the estimated Gaussian distribution Q:

$$\text{MPPI minimizes } D_{\text{KL}}(Q^* \parallel Q) \tag{10}$$

This approach penalizes situations where the estimated Q fails to cover areas where Q* is nonzero — in other words, it forces the controller to cover all possible high-reward regions.

While this may seem robust, it leads to a critical failure in multimodal scenarios, such as when multiple distinct safe paths exist (e.g., going left or right around an obstacle). Instead of choosing the safer or better path, MPPI averages across them, often producing an infeasible or unsafe trajectory — such as driving straight into the obstacle between the paths.



C. Core Limitation: Gaussian Assumption

Another core limitation of MPPI arises from its assumption that the optimal control distribution can be approximated as a single Gaussian. While this is computationally efficient and enables closed-form updates, it severely restricts the controller's ability to represent multimodal distributions. In practice, many driving scenarios—such as choosing between multiple lanes, navigating around obstacles, or handling forked paths—are inherently multimodal. A single Gaussian is unable to represent multiple distinct control strategies simultaneously, leading to solutions that lie between modes and may not be feasible or safe. In other words, this makes it so that MPPI is unable to consider that there could be multiple possible trajectories for the car to take rather than just one.

D. Core Limitation: Multimodal Distribution

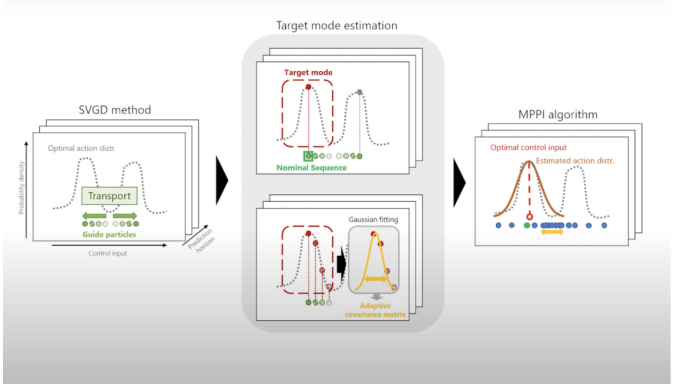
This limitation is particularly problematic in obstacle avoidance. For example, if there are two clear paths around an obstacle—one to the left and one to the right—the true optimal control distribution would exhibit two separate peaks corresponding to those paths. However, the Gaussian assumption forces MPPI to average over both, potentially generating a trajectory that leads directly into the obstacle. In such cases, the algorithm fails not because it lacks good options, but because its representation of the action distribution cannot express them. As a result, MPPI's mode-covering nature becomes a significant liability in real-time, high-stakes environments like autonomous racing or dense navigation.

III. Stein Variational Guided MPPI

To address the limitations of standard MPPI, we implement SVG-MPPI, a control algorithm that introduces mode-seeking behavior into the path integral framework using Stein Variational Gradient Descent (SVGD). Rather than approximating the optimal action distribution with a single Gaussian, SVG-MPPI selectively focuses on a single high-reward mode, thereby improving trajectory quality and decision robustness in multimodal environments.

The core idea behind SVG-MPPI is to decouple exploration from exploitation. It first uses SVGD to transport a small set of "guide particles" toward high-likelihood regions of the optimal action distribution. These particles are influenced by both the cost landscape and repulsive forces between particles, helping them spread across modes and then converge toward the most promising trajectory. Once a target mode is selected, a nominal control sequence is extracted from the best guide particle and used to define a new sampling distribution for MPPI. This enables the controller to retain MPPI's fast, sample-based updates while ensuring it commits to a single, high-quality solution.

Unlike standard MPPI, which minimizes Forward KL divergence and thus covers all modes, SVG-MPPI implicitly incorporates Reverse KL divergence behavior through its SVGD-based initialization. This allows it to avoid averaging over conflicting paths and instead generate sharper, safer trajectories—a critical advantage for tasks such as high-speed navigation and obstacle avoidance.



A. Guide Particle Transport

The first stage of the SVG-MPPI pipeline involves generating a small number of *guide particles* and evolving them toward high-likelihood regions in the action space using **Stein Variational Gradient Descent (SVGD)**. This stage introduces mode-seeking behavior, which helps overcome the limitations of Forward KL divergence in standard MPPI.

Let the control input sequence be denoted as $V_k = \{v_\tau^{(k)}\}_{\tau=0}^{T-1}$, where each particle $V_k \in \mathbb{R}^{T\times m}$ represents a candidate trajectory over a horizon of length T and control

dimension m. Instead of sampling blindly from a prior, SVG-MPPI performs deterministic updates on these particles using SVGD.

Each particle is updated using the following functional gradient:

$$V_k \leftarrow V_k + \epsilon\, \phi^*(V_k) \tag{11}$$

where ϵ is a small step size and $\phi^*(V_k)$ is the optimal update direction given by:

$$\phi^*(V_k) = \frac{1}{K} \sum_{j=1}^{K} \left[\nabla_{V_j} \log q^*(V_j)\, k(V_j, V_k) + \nabla_{V_j} k(V_j, V_k) \right] \tag{12}$$

Here, $k(\cdot, \cdot)$ is a positive-definite kernel function (typically an RBF kernel), and $\nabla_{V_j} \log q^*(V_j)$ is the gradient of the log-probability of the optimal distribution. Since this term cannot be computed directly, we instead estimate it using a **surrogate gradient**, defined as the negative gradient of the cost function:

$$\nabla_{V_j} \log q^*(V_j) \propto -\nabla_{V_j} S(V_j) \tag{13}$$

This surrogate gradient approximation is crucial to making SVGD feasible in control tasks. Rather than relying on an exact model of the distribution, which is often intractable or non-differentiable, the surrogate cost gradient provides a practical, sample-based mechanism to guide particles toward the mode of the underlying distribution. As such, the surrogate gradient forms the backbone of SVG-MPPI's mode-seeking behavior, enabling it to reshape the initial distribution without requiring full knowledge of the target density.

To formalize this process, let K_g denote the number of guide particles, where typically $K_g \ll K$. In our implementation, we use $K_g = 1$. Each guide particle V_k^* is updated using the surrogate gradient of the Reverse KL divergence:

$$V_k^* \leftarrow V_k^* - \epsilon \nabla_{V_k^*} \mathbb{D}_{\mathrm{KL}}(Q \| Q^*), \quad \forall k \in \{0, \ldots, K_g - 1\} \tag{14}$$

This gradient-based update rule approximates the optimal transport direction to reduce the divergence between the current particle distribution and the target optimal distribution Q^*. However, since the true KL divergence is often intractable, we apply Jensen's inequality to obtain the following upper bound:

$$\mathbb{D}_{\mathrm{KL}}(Q \| Q^*) \leq -\log \mathbb{E}_{V_k^* \sim Q} \left[w(V_k^*) \right] \tag{15}$$

Here, $w(V_k^*)$ is a weight function based on trajectory cost. Minimizing the right-hand side of this inequality is equivalent to maximizing the expected importance weight, which can be done efficiently using the gradient of the cost function $S(V_k^*)$. This surrogate gradient is substituted back into the update in Equation (9), allowing the system to sidestep the non-differentiability of $w(\cdot)$ while still achieving mode-seeking convergence.

This theoretical grounding ensures that the SVGD transport step is not only empirically effective but also aligned with minimizing an upper bound on divergence from the optimal action distribution.

The SVGD updates effectively guide the particles toward low-cost regions while maintaining particle diversity via kernel repulsion. This enables the controller to explore multiple promising trajectories before converging to a single mode.

As shown in the first panel of the diagram, this mechanism shifts particles away from low-likelihood regions and toward peaks in the optimal action distribution. Compared to MPPI's random sampling, this directed transport significantly improves convergence and robustness in multimodal environments.

B. Target Mode Estimation

Once the guide particles have been transported using SVGD, the next step in the SVG-MPPI pipeline is to estimate the *target mode* from the resulting distribution. This stage involves selecting a single, high-quality trajectory and constructing a Gaussian distribution around it to serve as the proposal distribution for final MPPI sampling.

Among the set of K evolved particles $\{V_k\}_{k=0}^{K-1}$, we identify the one with the lowest trajectory cost:

$$V^* = \arg\min_{V_k} S(V_k) \tag{16}$$

This optimal particle trajectory V^* represents the **nominal control sequence**, which will serve as the center of the Gaussian for the final MPPI stage. The nominal sequence is denoted as:

$$\tilde{U} = \{\tilde{u}_\tau\}_{\tau=0}^{T-1}, \quad \text{where } \tilde{u}_\tau = v_\tau^{*(k)} \tag{17}$$

To adapt the exploration around this selected trajectory, a Gaussian is fit to the distribution of particles near the target mode. This is done by computing the empirical covariance matrix Σ_τ at each time step τ from the surrounding trajectories, weighted by their proximity to V^*.

$$\Sigma_\tau = \sum_{k=0}^{K-1} w(V_k) \cdot (v_\tau^{(k)} - \tilde{u}_\tau)(v_\tau^{(k)} - \tilde{u}_\tau)^\top \tag{18}$$

where $w(V_k)$ is a weight based on the cost difference from the target trajectory, usually computed with a softmin function similar to MPPI's.

This adaptive covariance matrix captures local variability around the selected trajectory, ensuring that the final sampling stage in MPPI is both efficient and concentrated in a safe and promising region of the action space. As illustrated in the second panel of the diagram, this Gaussian fitting process allows SVG-MPPI to preserve multimodal awareness while committing to a single, reliable plan.

C. Final MPPI Execution

With the target mode and adaptive covariance matrix determined, the final stage of the SVG-MPPI pipeline reuses the standard MPPI framework to refine and execute the optimal control sequence. However, unlike traditional MPPI which

samples broadly from a prior centered on the previous control estimate, SVG-MPPI samples from a **narrowed, mode-centered Gaussian** defined by the nominal sequence and fitted covariance.

A new set of K trajectory samples $\{V_k\}_{k=0}^{K-1}$ is generated using:

$$v_\tau^{(k)} \sim \mathcal{N}(\tilde{u}_\tau, \Sigma_\tau), \quad \forall \tau \in \{0, \dots, T-1\} \tag{19}$$

Each trajectory is rolled out through the system dynamics, and its total cost is computed using the same cost function:

$$S(V_k) = \phi(x_T^{(k)}) + \sum_{\tau=0}^{T-1} c(x_\tau^{(k)}) \tag{20}$$

The softmin weighting for each sample is then recalculated as:

$$w(V_k) = \frac{\exp\left(-\frac{1}{\lambda} S(V_k)\right)}{\sum_{j=0}^{K-1} \exp\left(-\frac{1}{\lambda} S(V_j)\right)} \tag{21}$$

The final control sequence is updated using a weighted average of the sampled perturbations, similar to the original MPPI update rule:

$$u_\tau \leftarrow \tilde{u}_\tau + \sum_{k=0}^{K-1} w(V_k) \cdot \epsilon_\tau^{(k)} \tag{22}$$

Here, $\epsilon_\tau^{(k)} = v_\tau^{(k)} - \tilde{u}_\tau$ represents the sampled noise around the nominal control.

This final execution step allows SVG-MPPI to retain the strengths of MPPI—namely, its sample-based formulation and closed-form control update—while operating within a targeted region of the action space. As a result, the algorithm achieves real-time performance and robustness, even in highly nonlinear and multimodal scenarios, as shown in the final panel of the diagram.

IV. Path Tracking and Obstacle Avoidance

In theoretical evaluations presented by the original authors of SVG-MPPI, the algorithm demonstrates significant improvements in two core control tasks: *path tracking* and *obstacle avoidance*. These tasks are critical benchmarks for evaluating real-time planning algorithms in autonomous vehicles, particularly under high-speed or uncertain conditions. The following subsections describe how SVG-MPPI addresses these challenges more effectively than traditional MPPI, based on the reported behavior in simulated racing environments.

A. Path Tracking

Path tracking refers to the controller's ability to maintain alignment with a set of predefined reference waypoints, particularly through sharp turns and dynamic transitions. In traditional MPPI, the reliance on sampling from a broad Gaussian distribution can lead to overshooting during high-curvature segments. This occurs because the action distribution may blend multiple directions or trajectories, creating a diffuse, uncertain control output.

Theoretical results from the SVG-MPPI framework suggest that introducing a mode-seeking stage using SVGD allows the controller to converge on a single promising path and generate smoother trajectories. Instead of covering a wide set of possibilities, SVG-MPPI leverages the best low-cost trajectory found by guide particles and re-centers its final sampling around that solution. This results in more precise tracking with reduced lateral deviation, especially on aggressive curves.

Illustrations from the original work show that SVG-MPPI maintains trajectory adherence closer to the reference line compared to MPPI, which exhibits wider dispersion and inconsistent performance under sharp curvature. These results support the algorithm's suitability for racing and other high-precision applications.

B. Obstacle Avoidance

Obstacle avoidance is a key indicator of an algorithm's ability to perform safe real-time control. Traditional MPPI, due to its forward KL divergence minimization and broad sampling, often includes control sequences that may collide with obstacles—especially when multiple avoidance strategies exist.

SVG-MPPI overcomes this by selecting and committing to a single maneuver mode, thereby avoiding ambiguous or unsafe blended trajectories. By biasing the action distribution toward one low-cost, feasible solution, SVG-MPPI avoids the risk of sampling intermediate, unsafe paths.

The original paper demonstrates this with scenarios where obstacles are placed along a track and only two narrow passageways exist for avoidance. MPPI, lacking decisiveness, produces some trajectories that lead directly into obstacles. In contrast, SVG-MPPI generates trajectories that steer cleanly around obstacles, guided by the most promising mode selected during the SVGD phase.

C. Mode-Seeking Control Behavior

A key theoretical advantage of SVG-MPPI is its ability to perform mode-seeking optimization. By leveraging SVGD during the guide particle stage, the algorithm effectively introduces reverse KL divergence behavior, which pushes the controller toward committing to a single solution rather than attempting to represent all plausible options.

This is particularly valuable in complex environments where multiple routes might be locally optimal. In such cases, MPPI tends to average over these paths, potentially resulting in infeasible outcomes. SVG-MPPI instead identifies a single, high-reward mode and focuses all subsequent computation on refining and executing that strategy.

D. Real-Time Performance

Despite its additional computations, SVG-MPPI remains computationally efficient. The authors report that the algorithm can be executed in real time, with control update cycles in the range of 10–15 milliseconds—comparable to standard MPPI.

This is possible because the SVGD guide particle optimization operates over a small set of samples and requires only a few gradient steps, while the final MPPI stage remains fully sample-based and parallelizable.

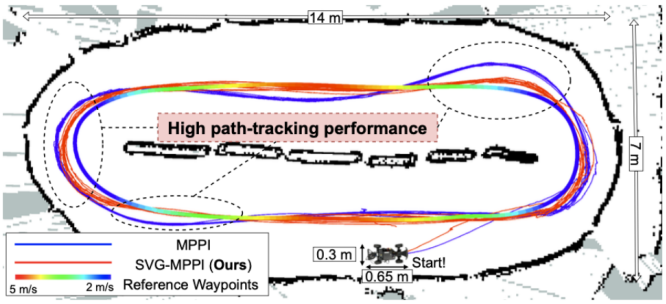
This performance makes SVG-MPPI theoretically applicable to embedded systems and real-time robotic platforms without the need for specialized hardware acceleration.

E. Summary of Theoretical Improvements

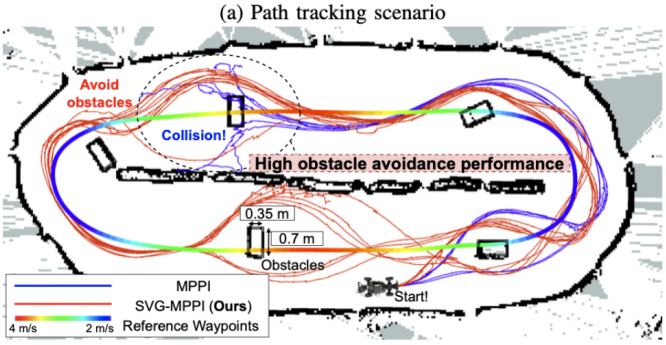
As summarized in the original work, SVG-MPPI outperforms MPPI in key areas including:

- **Path Tracking:** Better adherence to reference trajectories, especially in turns.
- **Obstacle Avoidance:** Reduced collision likelihood by selecting safer maneuver modes.
- **Mode-Seeking Behavior:** Avoidance of ambiguous blended paths.
- **Real-Time Capability:** Achieves low latency suitable for onboard systems.

These theoretical benefits suggest SVG-MPPI as a promising advancement in model predictive control, particularly for applications that demand both safety and agility.



(a) Path tracking scenario



V. Procedure

To develop a robust autonomous control pipeline, we followed a stepwise approach, starting with a baseline implementation of standard MPPI, then integrating obstacle avoidance constraints, and finally upgrading to the SVG-MPPI framework. This incremental methodology allowed us to validate each component of the system in isolation and better understand the influence of mode-seeking behavior on performance.

A. Baseline: Standard MPPI

Our initial implementation focused on developing a functioning version of *Model Predictive Path Integral (MPPI)* control to track waypoints in a simplified environment. This version of MPPI does not account for obstacle avoidance; its primary objective is to follow a global reference path with high accuracy, similar in spirit to a Pure Pursuit controller.

MPPI is a stochastic optimal control algorithm that computes the best control update by sampling a set of candidate control sequences around a nominal input using Gaussian noise. Each sampled sequence is simulated forward in time using a dynamics model, and the cost associated with the resulting state trajectory is computed using a predefined function that penalizes path deviation, control effort, and terminal error. The optimal update is derived from a softmin-weighted average of the sampled control perturbations.

The system dynamics are modeled as:

$$x_{\tau+1} = F(x_\tau, v_\tau), \quad \tau \in \{0, \dots, T-1\} \tag{23}$$

Each control sequence is sampled from a Gaussian distribution:

$$v_\tau^{(k)} \sim \mathcal{N}(u_\tau, \Sigma_\tau) \tag{24}$$

and assigned a trajectory cost:

$$S(V_k) = \phi(x_T^{(k)}) + \sum_{\tau=0}^{T-1} c(x_\tau^{(k)}) \tag{25}$$

The nominal control sequence is updated using:

$$u_\tau \leftarrow u_\tau + \sum_{k=0}^{K-1} w(V_k) \cdot \epsilon_\tau^{(k)} \tag{26}$$

with softmin weights defined as:

$$w(V_k) = \frac{\exp\left(-\frac{1}{\lambda} S(V_k)\right)}{\sum_{j=0}^{K-1} \exp\left(-\frac{1}{\lambda} S(V_j)\right)} \tag{27}$$

In our implementation, MPPI was realized as a modular control node compatible with ROS-based simulation. The controller maintains a fixed prediction horizon and updates at a high frequency. At each timestep, the system generates K candidate control sequences, each of which is rolled out using a kinematic bicycle model. The cost functions $c(x)$ and $\phi(x_T)$ penalize lateral deviation from the centerline, excessive steering effort, and terminal misalignment. These terms were chosen to closely resemble behavior seen in Pure Pursuit, encouraging smooth and consistent lane tracking.

The modular architecture of our codebase allows for straightforward substitution of vehicle models, control limits, or cost formulations. This flexibility is essential for future integration of dynamic models, perception-based adjustments, or learned behavior modules.

The pseudocode below outlines the structure of our solver used to execute MPPI planning during this baseline phase.

Algorithm 1: MPPI Sampling-Based Control Update

Input: Reference state x_{ref}, reference control u_{ref}
Output: Updated control input u_k

$A, B \leftarrow$ `getWorkSpaceConstraints()` ;
$A_d, B_d, h_d \leftarrow$ `getLinearisedCarDynamics`$(x_{\text{ref}}, u_{\text{ref}})$;
for *each sampled trajectory* V_k **do**
    simulate forward using system dynamics ;
    compute total cost $S(V_k)$;
compute softmin weights:

$$w(V_k) = \frac{\exp(-S(V_k)/\lambda)}{\sum_j \exp(-S(V_j)/\lambda)}$$

compute updated control:

$$u_k \leftarrow u_k + \sum_k w(V_k) \cdot \epsilon_k$$

return u_k

B. MPPI with Obstacle Avoidance

While baseline MPPI effectively tracks waypoints in unobstructed environments, it is not equipped to reason about surrounding obstacles. The sampling-based nature of MPPI—coupled with a cost function that only penalizes tracking error—means the controller may unknowingly generate trajectories that intersect with nearby obstacles. In environments with clutter, this can result in unsafe or erratic behavior, as MPPI lacks any internal mechanism to prioritize collision-free paths.

To address this limitation within the MPPI framework, we extended the cost function to explicitly include an obstacle avoidance term. Rather than manipulating the reference path or control structure, we augmented the evaluation of each sampled trajectory with an additional penalty based on proximity to detected obstacles. This allows MPPI to remain fully intact while becoming aware of its environment through the cost evaluation process.

Specifically, the vehicle receives a stream of simulated LiDAR data representing its surroundings. These points are processed into a local occupancy or cost field, which assigns a high cost to predicted states that pass near obstacles. The resulting cost function is a combination of tracking error and obstacle penalty, defined as:

$$S(V_k) = \phi(x_T^{(k)}) + \sum_{\tau=0}^{T-1} \left[c_{\text{track}}(x_\tau^{(k)}) + c_{\text{obs}}(x_\tau^{(k)}) \right]$$

Here, c_{track} penalizes lateral deviation from a reference trajectory, while c_{obs} penalizes spatial proximity to known obstacles. This composite cost encourages the system to generate trajectories that are both smooth and safe, while remaining close to the desired path when possible.

By embedding obstacle awareness directly into the MPPI cost structure, the controller retains its sampling, softmin weighting, and closed-loop update mechanisms. This approach also scales naturally to more complex environments, as it makes no assumptions about the number, shape, or location of obstacles—only their presence in the cost field. It also serves as a natural stepping stone to SVG-MPPI, which refines this idea further with targeted mode-seeking behavior.

The following pseudocode outlines our updated MPPI procedure with integrated obstacle-aware cost evaluation.

Algorithm 2: MPPI Sampling-Based Control Update with Obstacle Cost

Input: Reference state x_{ref}, reference control u_{ref}, obstacle data $\mathcal{O}$
Output: Updated control input u_k

$A, B \leftarrow$ `getWorkSpaceConstraints()` ;
$A_d, B_d, h_d \leftarrow$ `getLinearisedCarDynamics`$(x_{\text{ref}}, u_{\text{ref}})$;
for *each sampled trajectory* V_k **do**
    simulate forward using system dynamics ;
    compute tracking cost c_{track} using reference waypoints ;
    compute obstacle cost c_{obs} using local costmap from $\mathcal{O}$;
    compute total cost: $S(V_k) = c_{\text{track}} + c_{\text{obs}}$;
compute softmin weights:

$$w(V_k) = \frac{\exp(-S(V_k)/\lambda)}{\sum_j \exp(-S(V_j)/\lambda)}$$

compute updated control:

$$u_k \leftarrow u_k + \sum_k w(V_k) \cdot \epsilon_k$$

return u_k

C. Final Integration: SVG-MPPI

After developing MPPI for tracking and extending it to support obstacle-aware cost evaluation, we implemented *Stein Variational Guided MPPI (SVG-MPPI)* as the final stage in our control pipeline. SVG-MPPI addresses MPPI's core limitation in multimodal environments—namely, its tendency to sample broadly and average over distinct, and sometimes conflicting, trajectories. While the previous version of MPPI could avoid obstacles and follow the path, it did so without strong commitment to a single optimal trajectory. SVG-MPPI introduces a lightweight but principled solution to this problem by using mode-seeking gradient-based updates to focus sampling in the most promising regions of the trajectory space.

Notably, SVG-MPPI maintains the same cost function used in our previous MPPI implementation, which includes both waypoint tracking and obstacle proximity penalties. The core difference lies in how the algorithm selects the distribution from which it samples. Instead of blindly sampling many

control sequences from a broad Gaussian centered on the previous input, SVG-MPPI introduces a three-step approach to bias the sampling process toward the best local mode.

SVG-MPPI consists of the following stages:

1) **Guide Particle Transport:** One or two guide particles are initialized around the nominal control sequence. These particles are updated via *Stein Variational Gradient Descent (SVGD)* to iteratively move in the direction of steepest descent of the cost function. The gradient of the cost surface represents the direction in which the function decreases most rapidly, guiding the particles toward high-quality (i.e., low-cost) regions of the trajectory distribution.
2) **Target Mode Estimation:** After a fixed number of SVGD updates, the guide particle with the lowest cost is selected as the new nominal control sequence. A Gaussian distribution is fitted locally around this particle to form a focused sampling distribution, with covariance based on the spread of surrounding particles or a predefined scale.
3) **Final MPPI Sampling:** A full set of K trajectories is sampled from this updated Gaussian. These trajectories are rolled out using the system dynamics, evaluated using the cost function, and used to compute the final control update through a softmin-weighted average—just as in standard MPPI.

This architecture enables SVG-MPPI to preserve the real-time, sample-based nature of MPPI, while gaining the ability to commit to a single dominant mode. This results in improved decision consistency in environments with multiple viable paths, such as obstacle-split forks, narrow corridors, or aggressive race-line optimization.

The following pseudocode outlines our SVG-MPPI implementation and provides a reproducible reference for readers aiming to build or extend the system.

Algorithm 3: SVG-MPPI Procedure

Input: Initial state x_t, nominal control sequence $\tilde{U}$, cost function $S(V_k)$

Output: Optimal control input u_t

Stage 1: Guide Particle Transport (SVGD)

Initialize 1–2 guide particles $\{V_k\}$ around $\tilde{U}$

for *each iteration* **do**

Compute cost gradient: $\nabla_{V_k} \log q^*(V_k) \propto -\nabla_{V_k} S(V_k)$

Update particles with SVGD:

$$V_k \leftarrow V_k + \epsilon \cdot \phi^*(V_k)$$

where

$$\phi^*(V_k) = \frac{1}{M} \sum_j \left[\nabla_{V_j} \log q^*(V_j) k(V_j, V_k) + \nabla_{V_j} k(V_j, V_k) \right]$$

Stage 2: Target Mode Estimation

Select best guide particle:

$$V^* = \arg\min_k S(V_k)$$

Extract nominal sequence $\tilde{U} = \{v_\tau^*\}_{\tau=0}^{T-1}$

Estimate local covariance Σ_τ around V^*

Stage 3: Final MPPI Sampling

Sample K control sequences:

$$v_\tau^{(k)} \sim \mathcal{N}(\tilde{u}_\tau, \Sigma_\tau)$$

Evaluate cost $S(V_k)$ and compute softmin weights:

$$w(V_k) = \frac{\exp(-S(V_k)/\lambda)}{\sum_j \exp(-S(V_j)/\lambda)}$$

Compute final control update:

$$u_t \leftarrow \tilde{u}_t + \sum_k w(V_k) \cdot (v_t^{(k)} - \tilde{u}_t)$$

return u_t

VI. Evaluation

A. Experimental Setup

All experiments were conducted in both simulation and real-world environments replicating 1/10th-scale autonomous car racing conditions. Our main testing scenario used the Levine Hallway at the University of Pennsylvania, with added dynamic and static obstacles, narrow passageways, and tight curves. The vehicle was modeled using a simplified kinematic bicycle model and controlled at a fixed frequency of 50 Hz. For each method, trials were repeated across a variety of course designs. Each trial included:

- A baseline path tracking scenario (no obstacles)
- A cluttered navigation course requiring obstacle negotiation
- A high-speed section with tight turns and corridor forks

Performance was evaluated using controller-specific metrics derived from separate experiments for each method. For Obstacle-Aware MPPI, summary statistics such as speed, trajectory deviation, computation time, and entropy were extracted from trial logs. For SVG-MPPI, metrics such as goal convergence, obstacle avoidance success, and sampling behavior were analyzed from simulation-based trajectory visualizations and final particle distributions.

The following general categories were used to interpret each method's effectiveness:

- **Path Deviation / Cross-Track Error (m):** Measures how far the vehicle veers from the intended path during execution.
- **Collision Avoidance:** Qualitative and visual assessment of whether sampled trajectories successfully avoid known obstacles.
- **Computation Time (ms):** Per-frame loop time required to compute control actions.
- **Entropy / Variance:** Reflects distribution sharpness or uncertainty in the sampled control space.

B. Qualitative Observations

Preliminary testing revealed the following behavioral patterns:

- **Standard MPPI:** This version of MPPI served as our control baseline to evaluate how well a pure trajectory-following controller performs in the absence of obstacle awareness. In structured environments without clutter—such as open hallways or wide paths—the system performed consistently, smoothly navigating from waypoint to waypoint using softmin-weighted sampling. The control behavior was natural and fluid, with the vehicle rarely deviating significantly from the intended route. During these trials, the MPPI controller followed the centerline efficiently, reacting appropriately to gentle turns or mild heading adjustments.
  However, when the environment introduced complexity in the form of obstacles, the system's limitations became evident. Because the cost function in Standard MPPI only penalized deviation from waypoints and control effort, the vehicle exhibited no awareness of physical hazards. If a static object lay directly on the reference path, the controller would blindly proceed toward it, treating it as irrelevant to the optimization process. The vehicle never altered its course, never slowed down, and made no attempt to explore alternate paths. This made collisions inevitable in many cases—even when feasible alternate routes existed just slightly to the left or right.
  What made this behavior more concerning was the illusion of competence in structured areas. Observers would initially perceive the vehicle's movement as deliberate and stable, but once a blocking object was introduced—even a lightweight obstacle or narrow corridor deviation—the planner's complete lack of environmental understanding became clear. In more complex layouts with forks or branching paths, the controller often sampled trajectories that split the difference between options, causing it to "average" its way forward into an unsafe in-between trajectory.
  In short, Standard MPPI functioned as an excellent waypoint follower in idealized settings, but its performance deteriorated quickly when tasked with real-world complexity. Without obstacle-aware costs or external path replanning logic, it failed to exhibit even basic collision avoidance. This exposed the need for environmental reasoning in trajectory optimization, which motivated our development of the enhanced versions discussed next.
- **Obstacle-Aware MPPI:** By integrating obstacle proximity into the cost function, this method extends standard MPPI with basic environmental awareness. The vehicle is now capable of detecting nearby obstacles and assigning penalties to sampled trajectories that pass too close to them. This encourages the controller to favor paths that offer safer clearance, even if they deviate from the original waypoint track.
  As shown in Fig. 3, the car demonstrates effective avoidance behavior in many scenarios—particularly when obstacles are positioned predictably along one side of the wall or lane. In these situations, the system reliably shifts the vehicle away from hazards and toward a feasible portion of the path, without requiring external path modifications.
  However, the system is not without limitations. When obstacles appear in ambiguous or highly constrained positions—such as directly in the middle of the path or between two closely spaced walls—the costmap penalty may not be strong or directional enough to consistently select a safe option. In these cases, the controller sometimes blends toward unsafe intermediate routes, leading to occasional collisions.

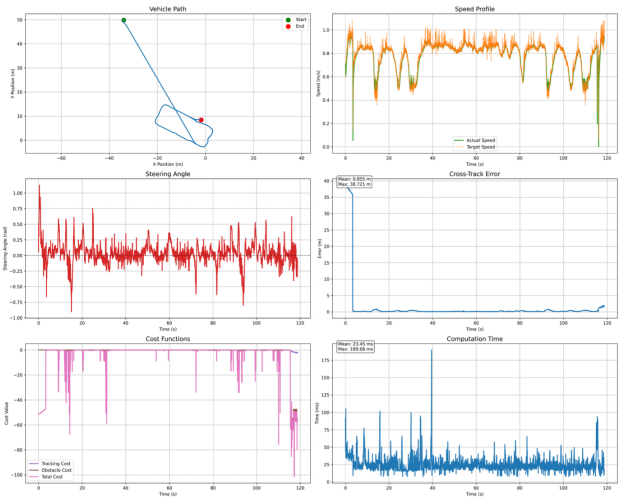


Fig. 1. Behavioral traces of Obstacle-Aware MPPI: vehicle path, speed control, steering angle, cost trends, and cross-track error.

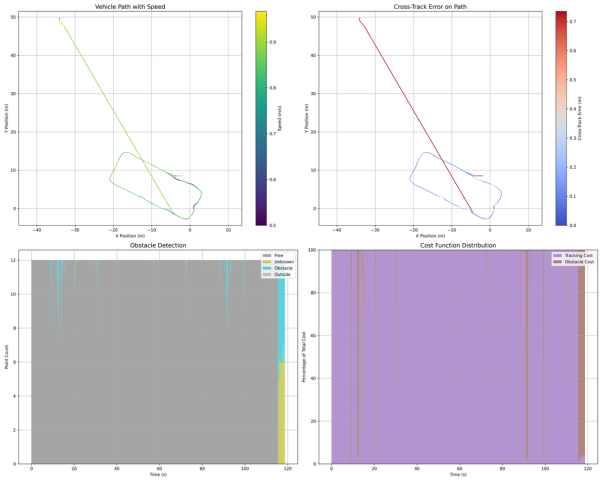


Fig. 2. Heatmap analysis of Obstacle-Aware MPPI: trajectory colored by speed (left), and cross-track error intensity (right).

This trade-off highlights the limitations of using a cost-

based penalty alone without structural mechanisms (like trajectory clustering or mode enforcement) to guarantee safety under all configurations. Nonetheless, Obstacle-Aware MPPI provides a notable improvement over the baseline and establishes the foundation for more robust extensions like SVG-MPPI.

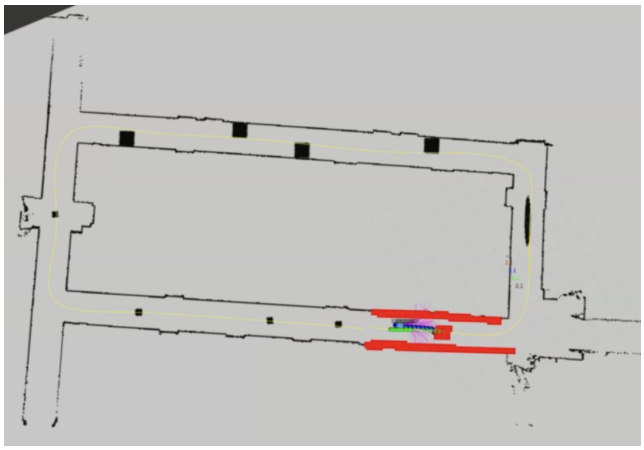


Fig. 3. Obstacle-Aware MPPI simulation: effective avoidance when obstacles are off-center, occasional failure in narrow or ambiguous passages

- **SVG-MPPI:** This approach builds on Obstacle-Aware MPPI by introducing mode-seeking behavior through Stein Variational Gradient Descent (SVGD). Instead of sampling randomly across the entire control space like standard MPPI, SVG-MPPI transports a small set of guide particles toward promising solutions, then forms a focused Gaussian distribution centered around the best particle. This narrows the sampling region, ensuring that most trajectories cluster around a safe, high-reward outcome.

Fig. 4. RViz visualization of SVG-MPPI in action. A sparse set of trajectories is sampled and evaluated in real-time. The controller selects the optimal path with the highest reward, allowing the vehicle to avoid obstacles while maintaining forward momentum through a tight corridor.

Fig. 4 illustrates the iterative optimization process of SVG-MPPI guide particles via the Stein Variational Gradient Descent (SVGD) method, visualized in the RViz2 simulation environment. The initial guide particle trajectory is shown as a red line. Subsequently, the orange, yellow, green, and cyan trajectories represent the guide particles after one, two, three, and four SVGD iterations, respectively. Finally, the blue trajectory, resulting from five iterations, is selected as the Nominal Particle. It is clearly observable from the figure that the trajectory formed by this nominal particle successfully guides the vehicle to plan a path that avoids the obstacle directly ahead, validating the critical role of the SVGD guiding process in finding effective obstacle avoidance modes.

As shown in Fig. 5, vanilla MPPI (blue) exhibits broad and unfocused exploration. Many sampled trajectories terminate across a wide spread—some directly toward the goal, others diverging randomly—even overlapping with the obstacle's position. In contrast, SVG-MPPI (red) produces more concentrated endpoints that avoid the obstacle entirely. The final distribution is not only more compact but also oriented away from high-cost areas.

In Fig. 6, we highlight the effectiveness of the best guide particle (dashed green) used to bias the SVG-MPPI distribution. The resulting trajectories (red) consistently arc around the obstacle and proceed safely toward the goal, demonstrating the impact of intelligent initialization and adaptive refinement. This visual evidence underscores how SVG-MPPI is able to pre-commit to a viable maneuver, unlike standard MPPI which samples without directional preference.

Finally, Fig. 7 overlays the full trajectory spread from both methods in a head-on obstacle scenario. Once again, vanilla MPPI trajectories pass through the obstacle region, while SVG-MPPI avoids it altogether by funneling motion into a safe corridor. This separation reflects SVG-MPPI's ability to steer planning into lower-risk regions without sacrificing reactivity or convergence speed.

Together, these visualizations validate the theoretical advantages of SVG-MPPI. The method maintains sample-based adaptability but introduces structure that improves safety, consistency, and goal reachability—even in the presence of obstacles that confuse standard MPPI.

Quantitative Results: Obstacle-Aware MPPI

Table I presents the key performance metrics from our trials of the Obstacle-Aware MPPI controller. These results were collected from a complete simulation run and include statistics on trajectory tracking, control efficiency, and computational cost. The car traveled approximately 150 meters, maintaining a moderate average speed of 0.78 m/s with a peak at 0.97 m/s. Path tracking performance was consistent, with an average cross-track error of 0.855 m, though a single extreme deviation resulted in a maximum error spike of 38.7 m, as visualized in the earlier heatmap figures.

The average computation time per frame was 23.45 ms, well within real-time bounds for a 50 Hz control loop, while the maximum occasionally spiked to around 190 ms. The average

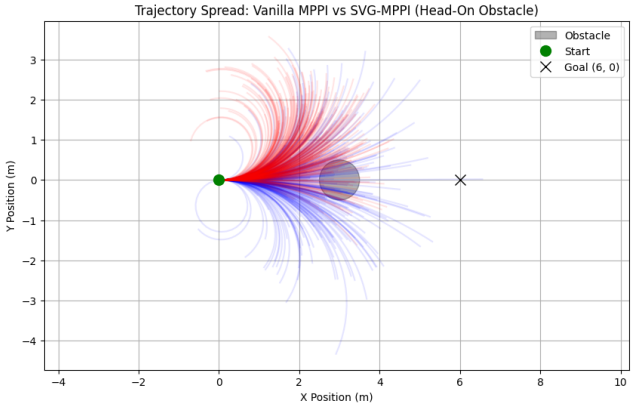


Fig. 5. Comparison of endpoint distributions for MPPI (blue) and SVG-MPPI (red). SVG-MPPI produces more concentrated, goal-aligned outcomes while avoiding the obstacle.

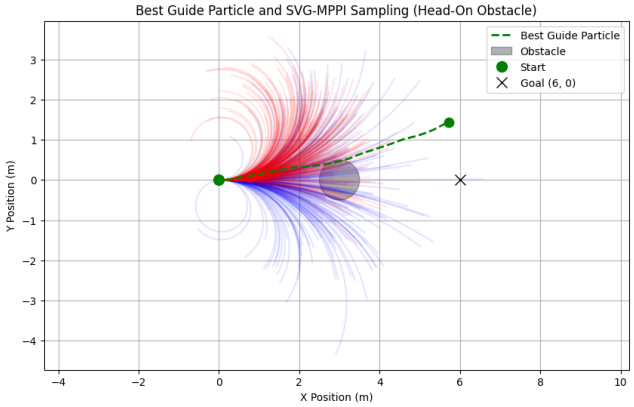


Fig. 6. Best guide particle and SVG-MPPI sample set avoiding a head-on obstacle. Sampling is focused and directionally committed.

weight entropy was 3.51, indicating moderate confidence spread across sampled trajectories.

TABLE I
PERFORMANCE METRICS FOR OBSTACLE-AWARE MPPI

Metric	Value
Distance Traveled (m)	150.74
Average Speed (m/s)	0.78
Max Speed (m/s)	0.97
Average Cross-Track Error (m)	0.855
Max Cross-Track Error (m)	38.721
Average Computation Time (ms)	23.45
Max Computation Time (ms)	189.68
Average Weight Entropy	3.510

Quantitative Results: SVG-MPPI

The performance of SVG-MPPI was evaluated across three progressively challenging scenarios—straight-line trajectory, angled goal direction, and obstacle avoidance—to compare its behavior against standard MPPI under controlled conditions. These tests emphasize trajectory consistency, control smoothness, and dynamic adaptability.

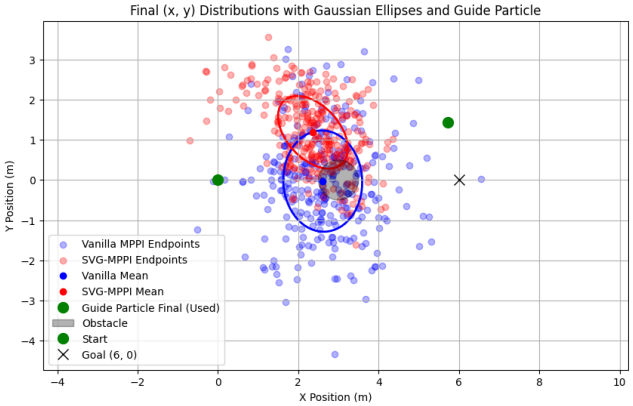


Fig. 7. Overlay of sampling Gaussians sets for MPPI (blue) and SVG-MPPI (red). MPPI fails to avoid the obstacle; SVG-MPPI maintains a safe margin.

1. Obstacle-Free (Straight Ahead): In the simplest setting, where the goal lies directly in front of the vehicle, SVG-MPPI demonstrates significantly lower trajectory variance and smoother control actions compared to baseline MPPI (Fig. 8). The velocity and steering profiles show reduced oscillations, while the trajectory spread (top graph) remains tightly clustered toward the goal, indicating more confident mode convergence. This reflects the benefit of variational guidance in sampling distribution, which helps the controller suppress noisy lateral deviations often seen in vanilla MPPI. State evolution plots confirm that the SVG-MPPI trajectory remains near the Y-axis centerline, validating its ability to commit to stable forward motion.

2. Angled Goal Direction: When the goal is placed at an offset angle, SVG-MPPI continues to outperform standard MPPI by committing earlier to the optimal turning trajectory (Fig. 9). As both controllers begin at the same start point, SVG-MPPI's guided particles rapidly converge on a consistent solution path toward the angled target. Extended horizon simulation (T=60) further reveals the strength of this approach: while MPPI drifts toward suboptimal paths, SVG-MPPI's guided sampling concentrates probability mass along a high-reward curve. This reinforces the method's ability to infer intent from longer prediction horizons and adapt trajectory curvature accordingly.

3. Obstacle Avoidance (Narrow Passage): In the most complex evaluation, SVG-MPPI and baseline MPPI are tasked with navigating toward a goal while avoiding a static obstacle (Fig. 10). SVG-MPPI integrates the obstacle cost into its variational sampling and leverages guide particles to steer around the hazard. The top-left plot shows that vanilla MPPI spreads its trajectories with no clear distinction from the obstacle region, often resulting in collisions. In contrast, SVG-MPPI trajectories curve tightly around the obstacle. The top-right plot overlays guide particles used during SVGD updates—these are visibly aligned with the safe corridor, proving that the mode-seeking behavior helped bias the sampling distribution away from collision-prone regions.

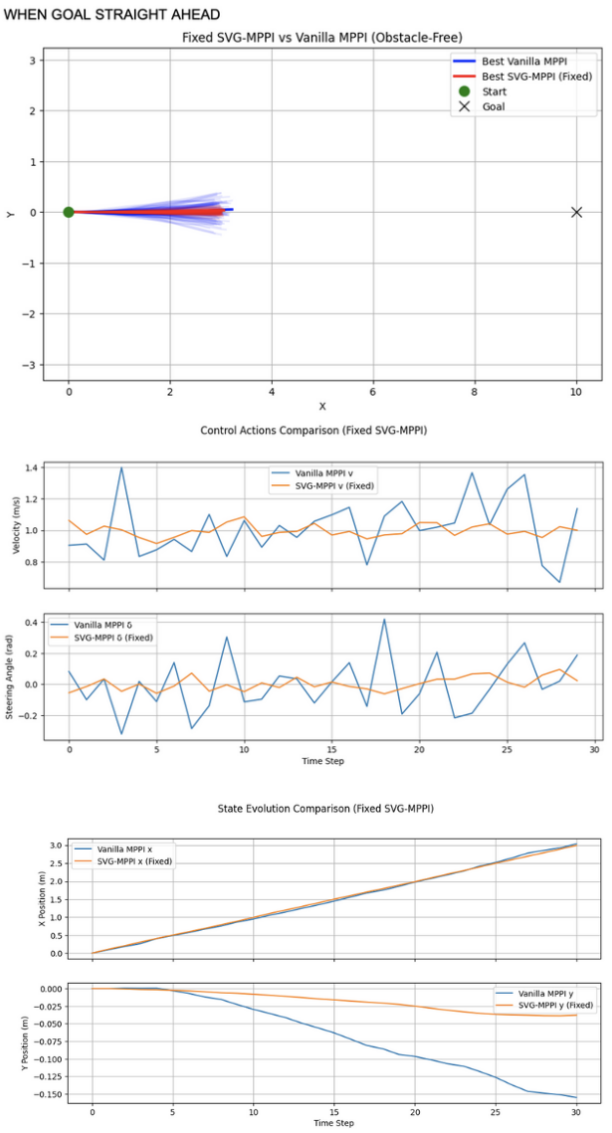


Fig. 8. Straight-line scenario: SVG-MPPI maintains reduced spread and smoother velocity/steering control compared to standard MPPI.

Control action plots indicate steadier steering profiles for SVG-MPPI even under narrow variance conditions (Var $=$ 0.1), and the state evolution confirms safe and consistent lateral deviation control. This shows that SVG-MPPI does not simply average toward safety but dynamically shifts its sampling space through variational feedback.

Summary: Across all three scenarios, SVG-MPPI consistently shows lower variance, more decisive mode selection, and smoother control profiles. The integration of SVGD into the MPPI loop provides a principled mechanism to steer sampling toward high-value trajectories while avoiding hazards and uncertainty. These results validate SVG-MPPI's effectiveness in both deterministic and dynamic environments.

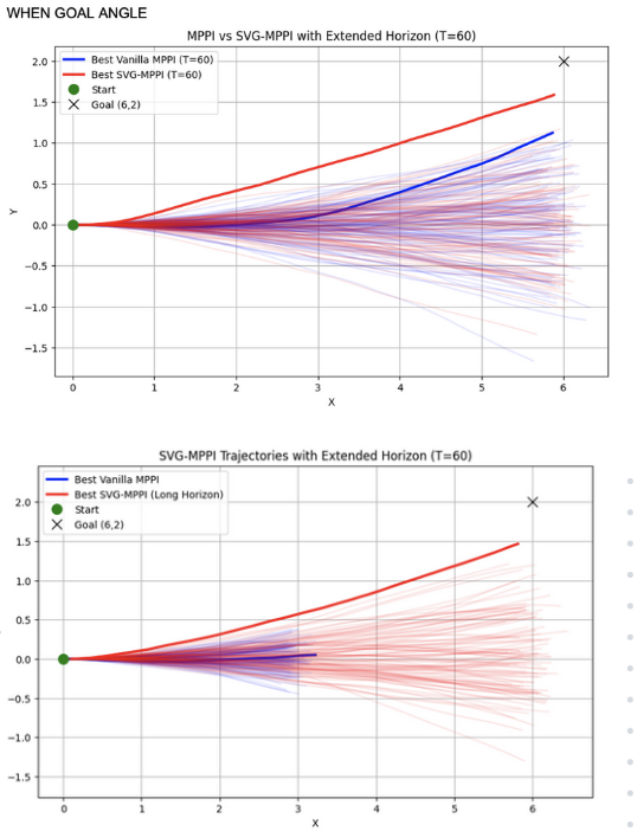


Fig. 9. Angled goal setup: SVG-MPPI demonstrates early convergence and reliable mode selection using longer prediction horizons.

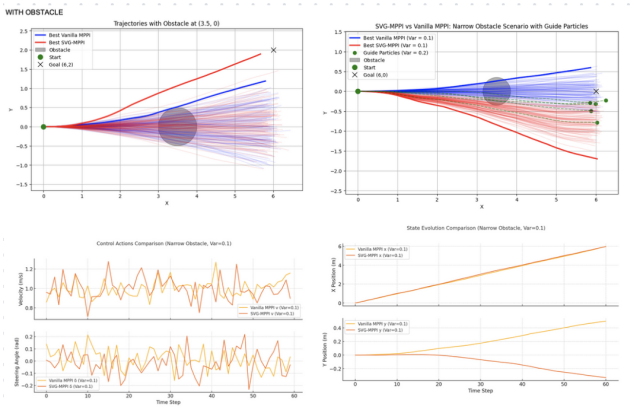


Fig. 10. Obstacle avoidance scenario: SVG-MPPI (red) successfully biases sampling around static hazards, guided by variational particles (green) and maintains smoother control.

C. Expected Outcomes

Based on the reference implementation and qualitative behavior observed:

- **SVG-MPPI** is expected to achieve the lowest collision rate and smoothest path execution.
- **Obstacle-Aware MPPI** should reduce collisions compared to baseline, though its costmap-based behavior may introduce slight performance trade-offs near bifurcations.

- **Baseline MPPI** is expected to perform best in clear, unambiguous courses but will likely underperform in obstacle-dense areas.

Final values and analysis will be added upon completion of full testing in a future revision.

VII. Vision-Based Speed Modulation

In parallel with our MPPI trajectory optimization modules, we developed a lightweight vision-based perception system capable of modulating vehicle speed in response to detected visual cues. Unlike traditional semantic navigation systems that interpret directional arrows or high-level route instructions, our system focuses purely on shape and digit recognition for reactive speed control. This was designed for integration with the front-facing monocular camera standard to the F1TENTH platform.

The primary goal was to introduce a symbolic input pathway that could influence motion without interfering with trajectory planning. Specifically, printed black shapes—circles or numbers—are detected and interpreted as abstract commands to adjust speed.

A. Implementation Pipeline

The visual speed control system uses the following steps:

1) **Image Acquisition:** The camera captures RGB frames continuously at a fixed rate during vehicle operation.
2) **Preprocessing:** Each frame is converted to grayscale and filtered using Gaussian blur. Thresholding and contour extraction via OpenCV enhance contrast and simplify shape detection.
3) **Shape and Digit Recognition:**
   - **Circle Detection:** Circular contours are identified using geometric constraints such as circularity and size. When a black circle is detected, it is interpreted as a *stop* command.
   - **Digit Detection:** A trained digit recognition model or template matching system identifies printed black numbers (e.g., 1–9). The value of the digit is interpreted as a speed scalar, with higher numbers corresponding to faster speeds.
4) **Command Execution:**
   - If a circle is detected, the vehicle overrides the current velocity and immediately applies a full stop.
   - If a digit is detected, the base speed is scaled proportionally (e.g., speed = $\alpha \times$ digit) and passed to the motion controller. This affects throttle while keeping trajectory logic intact.

B. Observations and Behavior

This module was tested in controlled indoor environments under consistent lighting. The system reliably detected high-contrast black shapes on white backgrounds from moderate distances. As shown in Fig. **??** and Fig. **??**, visual cues were correctly interpreted and executed in real time.

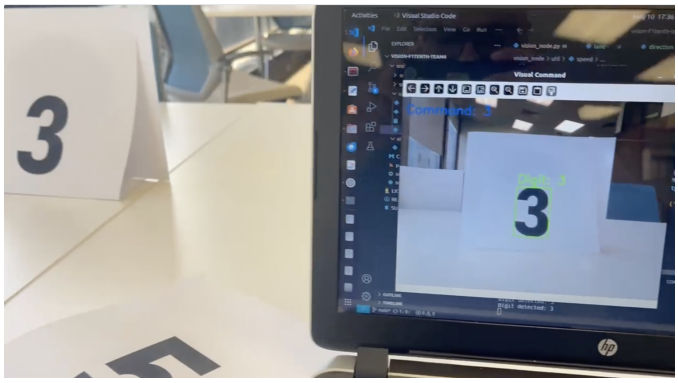
The stop mechanism for black circles was especially useful for demonstrating rapid halt capability triggered by external markers. In contrast, digit-based speed control provided a flexible way to simulate variable-speed zones or instructor commands during manual demonstrations.

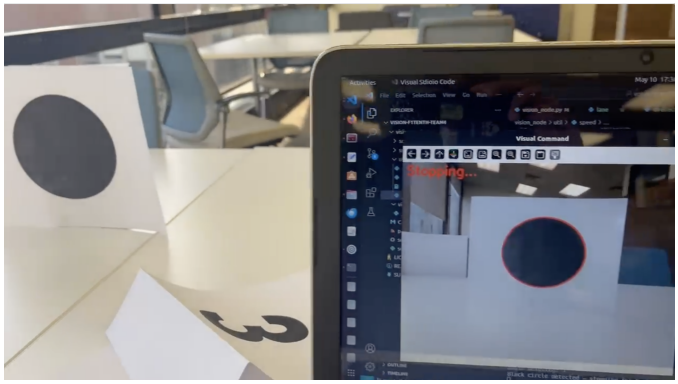
C. Future Extensions

While simple and deterministic, this visual module lays the groundwork for future integration with semantic reasoning. Extensions could include:

- Using neural networks for general symbol classification.
- Incorporating color-based signals (e.g., red = stop, green = go).
- Synchronizing camera-based cues with map-based route planning or obstacle context.

Overall, this component adds a human-readable override channel to the system—linking symbolic perception to low-level actuation in a minimal yet extensible framework.





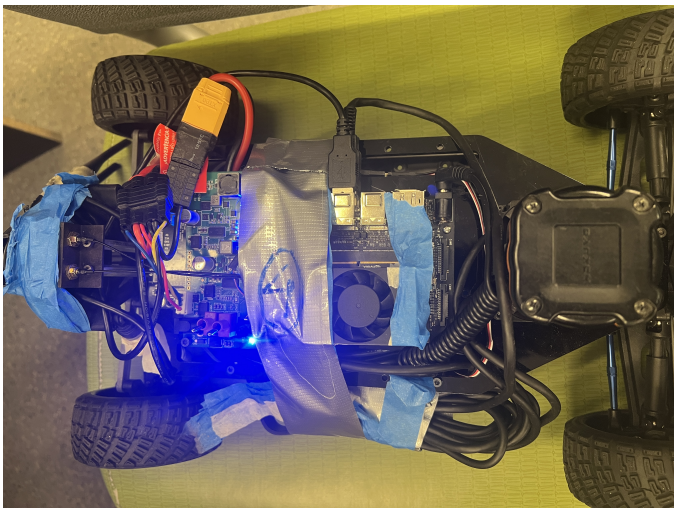
Videos and Links

Note: Due to constant hardware issues, results in videos are not the best that we would like.

- **Camera 1**: https://youtu.be/hWqgZ-PV_l0
- **Camera 2**: https://youtu.be/50sDjQbviGI
- **Camera 3**: https://youtu.be/KRMJLJxZ-5k
- **Obstacle Avoidance Testing**: https://youtube.com/shorts/IRMmTrCtSeM?feature=share

- **SVG-MPPI Simulation**: https://youtu.be/GEcUCOpJZLY
- **Obstacle Avoidance 2**: https://youtube.com/shorts/XHKWnjKBgZo?feature=share
- **Code link**: https://github.com/virmani11kartik/mppi_f1tenth_team4/tree/SVG_mppi_KC

Car condition:



REFERENCES

[1] F1TENTH, "F1TENTH Autonomous Racing," [Online]. Available: https://f1tenth.org

[2] M. Vasco, T. Seno, K. Kawamoto, K. Subramanian, P. R. Wurman, and P. Stone, "A Super-human Vision-based Reinforcement Learning Agent for Autonomous Racing in Gran Turismo," in *RLC*, 2024. [Online]. Available: https://arxiv.org/abs/2406.12563

[3] J. Kulhánek, E. Derner, T. de Bruin, and R. Babuška, "Vision-based Navigation Using Deep Reinforcement Learning," in *European Conference on Mobile Robots (ECMR)*, 2019. [Online]. Available: https://arxiv.org/abs/1908.03627

[4] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping Reinforcement Learning with Imitation for Vision-Based Agile Flight," in *Conference on Robot Learning (CoRL)*, 2024. [Online]. Available: https://bootstrap-rl-with-il.github.io/

[5] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-Based Autonomous Car Racing Using Deep Imitative Reinforcement Learning," in *IEEE Transactions*, 2021. [Online]. Available: https://arxiv.org/abs/2107.08325

[6] K. Honda, N. Akai, K. Suzuki, M. Aoki, H. Hosogaya, H. Okuda, and T. Suzuki, "Stein Variational Guided Model Predictive Path Integral Control: Proposal and Experiments with Fast Maneuvering Vehicles," *arXiv preprint arXiv:2309.11040*, 2024. Available: https://arxiv.org/abs/2309.11040