2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Spline-Interpolated Model Predictive Path Integral Control with Stein Variational Inference for Reactive Navigation

Takato Miura[1], Naoki Akai[1,2], Kohei Honda[1], and Susumu Hara[1]

***Abstract*— This paper presents a reactive navigation method that leverages a Model Predictive Path Integral (MPPI) control enhanced with spline interpolation for the control input sequence and Stein Variational Gradient Descent (SVGD). The MPPI framework addresses a nonlinear optimization problem by determining an optimal sequence of control inputs through a sampling-based approach. The efficacy of MPPI is significantly influenced by the sampling noise. To rapidly identify routes that circumvent large and/or newly detected obstacles, it is essential to employ high levels of sampling noise. However, such high noise levels result in jerky control input sequences, leading to non-smooth trajectories. To mitigate this issue, we propose the integration of spline interpolation within the MPPI process, enabling the generation of smooth control input sequences despite the utilization of substantial sampling noises. Nonetheless, the standard MPPI algorithm struggles in scenarios featuring multiple optimal or near-optimal solutions, such as environments with several viable obstacle avoidance paths, due to its assumption that the distribution over an optimal control input sequence can be closely approximated by a Gaussian distribution. To address this limitation, we extend our method by incorporating SVGD into the MPPI framework with spline interpolation. SVGD, rooted in the optimal transportation algorithm, possesses the unique ability to cluster samples around an optimal solution. Consequently, our approach facilitates robust reactive navigation by swiftly identifying obstacle avoidance paths while maintaining the smoothness of the control input sequences. The efficacy of our proposed method is validated on simulations with a quadrotor, demonstrating superior performance over existing baseline techniques.**

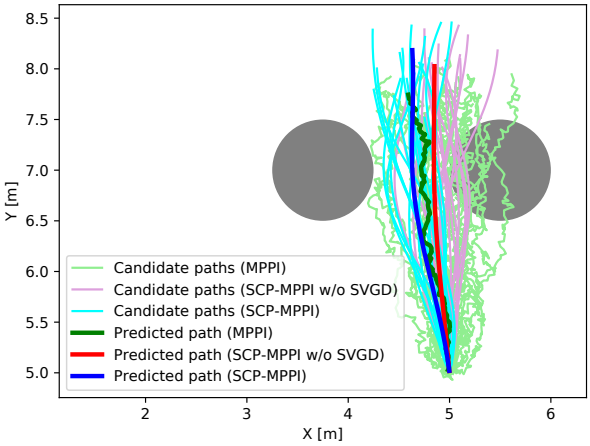

Fig. 1. A selected simulation result. The original MPPI's candidate paths (light green) are erratic due to the randomness of the injected Gaussian noise, leading to a jerky predicted path (green). To achieve a smoother solution, we employ spline interpolation in MPPI, which samples sparse control points and interpolates them using spline curves. This approach enables the generation of smoother candidate paths. However, some paths (light purple) collide with obstacles due to the scarcity of control points. To address this, we apply the SVGD method, enhancing MPPI with spline interpolation and SVGD, referred to as SCP-MPPI. This method improves the quality of collision-free interpolated samples (light blue). Consequently, SCP-MPPI is able to identify a smooth and collision-free solution (blue), even with sparse control points, resulting in improved computational efficiency.

I. Introduction

Unmanned Aerial Vehicles (UAVs) have garnered considerable interest for their capability to enhance a wide range of applications, particularly in unfamiliar and demanding settings. In these environments, characterized by numerous unexpected obstacles, UAVs must have the ability for reactive point-to-goal navigation, enabling them to reach designated destinations while bypassing unforeseen obstacles. To facilitate agile and reliable reactive navigation, local motion planners are tasked with identifying smooth and collision-free paths using a minimal number of random samples, thereby reducing computational demands.

Among various practical local motion planning methods, such as Dynamic Window Approach (DWA), Sampling-based Local Planner (SLP), and Cross-Entropy Method (CEM) [1]–[3], Model Predictive Path Integral (MPPI) control [4]–[6] stands out for its relative computational efficiency. MPPI, a sampling-based stochastic Model Predictive Control (MPC) framework [7], [8], is adept at handling the complexities of non-linear and non-differentiable environments, including UAV dynamics and cost maps. This method is capable of finding a smooth control input sequence by sampling a sufficiently large number of samples and taking their weighted average. In the MPPI algorithm, the variance of the random samples is crucial for improving collision avoidance, whereas solution smoothness necessitates spreading a "large enough" number of samples across a broad variance. However, a larger sample variance increases the number of samples required, which is constrained by hardware limitations. Therefore, finding a smooth solution with an insufficient number of samples due to these limitations poses a significant challenge.

To find a smooth solution even with the insufficient number of samples, one key idea is to utilize sample in-

*This work was supported by KAKENHI under Grant 23K03773.
[1]Takato Miura, Naoki Akai, Kohei Honda, and Susumu Hara are with the Graduate School of Engineering, Nagoya University, Nagoya 464-8603, Japan
[2]Naoki Akai is with the LOCT Co., Ltd., Nagoya 464-0805, Japan
E-mail of corresponding author: akai@nagoya-u.jp

terpolation between the time dimensions. That is, we can obtain a smooth solution by interpolating a sparse predicted control input sequence (hereafter referred as to *sparse control points*). However, the sparse control points decrease the predictive resolution of the dynamics, resulting in decreasing the number of feasible (*i.e.*, collision-less) samples, resulting in degrading of the smoothness and sample efficiency.

In this paper, we propose an enhanced Model Predictive Path Integral (MPPI) algorithm, named Sparse Control Points MPPI (SCP-MPPI), designed to improve the smoothness and collision-free performance of MPPI solutions. Our approach first integrates spline interpolation of control points into the conventional MPPI framework to generate smooth control input sequences, even when control points are sparse. While spline interpolation allows for smoother sample trajectories, it often results in a majority of samples being infeasible due to the reduced number of control points, for example, some predicted paths collide with obstacles. To mitigate collisions within the interpolated sequences, we subsequently apply the Stein Variational Gradient Descent (SVGD) method [9] following interpolation. SVGD enhances the number of feasible samples by transporting them in accordance with the gradient of the optimal distribution. Consequently, SCP-MPPI is capable of producing smooth and collision-free trajectories, demonstrating superior performance compared to the standard MPPI approach, as illustrated in Fig. 1.

To encapsulate the primary contributions of this study, we introduce a novel method based on the MPPI framework, named SCP-MPPI. This approach combines spline interpolation with the SVGD algorithm to facilitate smooth and collision-free navigation, even though sampled control inputs are sparse. The utilization of sparse control points significantly diminishes the computational complexity typically associated with the random sampling and evaluation processes necessary for identifying the optimal solution. Through experimental validation, we reveal the enhanced efficiency of our proposed method, achieving comparable performance levels with fewer control points. While traditional methods necessitate up to 150 points, our method demonstrates effectiveness with as few as 4 points. Moreover, SCP-MPPI surpasses other baseline methods in obstacle avoidance success rates and the average speed of the UAV, culminating in shorter flight durations to the designated target. These findings show the potential of our method to provide robust and agile control for the reactive navigation of UAVs.

II. Related work

Although MPPI is a promising framework for sampling-based MPC because of its relatively high sampling efficiency, it often struggles with the oscillation of the solution when the number of samples is small due to the limitation of computational resources. To overcome this limitation, how to improve the sample efficiency is an open issue. Existing approaches can be broadly categorized into the following two types:

A. Random Sampling in Different Spaces

One approach to finding a smooth control input sequence with fewer samples is to sample other spaces instead of control inputs. Kim *et al.* find smooth control input sequences for autonomous vehicles by sampling in lifted input space, *i.e.*, differential values of control inputs [10]. Although this approach is easy to apply, it penalizes the agile behavior of the vehicle, which may degrade obstacle avoidance performance. Another approach is to sample the random parameters in a geometric space. Some studies can find a smooth geometric path by sampling parameters of the geometric curves [11], [12], such as spline curves, or using a kernel function [13]. However, these methods cannot find actual control inputs and require a lower-level controller to track the optimized path.

B. Improving Quality of the Random Samples

The second approach to finding a more sample-efficient solution is to increase the number of low-cost and feasible random samples. Some studies propose ancillary controllers that induce the MPPI solution to far from infeasible area [14], [15]. These methods often struggle with the nonlinearity of the system to formulate the ancillary controllers. Other studies try to improve the prior distribution to reduce the infeasible samples [16]–[19].

As a more advanced method, variational inference techniques have been attracting attention to handle more representative distribution, as their prior [12], [20]–[22]. In particular, Stein Variational MPC (SV-MPC) [12] can transport the random samples directly to make them feasible and low-cost by using SVGD [9]. However, the computational complexity per samples is expensive, and requires reducing the control points.

Our proposal in this work restricts the sampling of control input sequences to lower dimensions and approximates the intermediate points using spline interpolation. This approach allows multi-dimensional control input sequence optimization problems to be approximated with lower-dimensional control input sequences, thus allowing more efficient trajectory sampling. By updating the sampled trajectories using the SVGD method, the proposed method achieves effective trajectory updates in response to the environment. The proposed approach is expected to overcome the problem of falling into local minima by exploiting SVGD's ability to promote convergence among solutions and to encourage the solutions to spread evenly over the entire posterior distribution, aided by the gradient information of the cost function.

III. ALGORITHM REVIEW

A. MPPI Algorithm Review

In this section, we briefly describe the basic algorithm of MPPI. The MPPI theory considers a general discrete-time dynamical system for the controlled object. The control input sequence $U = (\mathbf{u}_0,\ \mathbf{u}_1,\ \ldots,\ \mathbf{u}_{T-1})$ represents the control inputs from the current time point to future T time points and controls the system, where $\mathbf{u}$ is a control point. The MPPI algorithm models the system as, $\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \mathbf{v}_t)$, where

$\mathbf{x}_t \in \mathbb{R}^n$ represents the state of the system at time t, and $\mathbf{v}_t \in \mathbb{R}^m$ is the input to the system at time t. We assume that $\mathbf{F}$ can be nonlinear and non-differentiable. In practice, the control input sequence is subject to noise, and the actual control input sequence is represented as $V = (\mathbf{v}_0,\ \mathbf{v}_1,\ \ldots, \mathbf{v}_{T-1})$, where $\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}_t, \Sigma)$, and Σ is the covariance matrix of the Gaussian distribution.

MPPI aims to approximate the action distribution q using randomly generated K control input sequences $V^{(k)}$. The probability density function of the action distribution q is defined as,

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

where $Z = \left(\sqrt{(2\pi)^m |\Sigma|}\right)$ is the normalization term. To find the optimal U^*, MPPI solves the following stochastic optimization problem,

$$U^* = \underset{U}{\operatorname{argmin}}\, \mathbb{E}_q \left[\psi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \mathcal{L}(\mathbf{x}_t,\ \mathbf{u}_t)\right], \tag{2}$$

where $\psi(\cdot)$ is the terminal cost function, and $\mathcal{L}(\cdot)$ is the stage cost function. In the following, we desire the sequence cost of the kth trajectory as,

$$\tilde{S}_k = \psi(\mathbf{x}_T^{(k)}) + \sum_{t=0}^{T-1} \mathcal{L}(\mathbf{x}_t^{(k)},\ \mathbf{u}_t^{(k)}). \tag{3}$$

Since directly solving the stochastic optimization problem shown in equation (2) is challenging, MPPI instead aims to minimize the Kullback-Liebler (KL) divergence between q and the optimal optimality likelihood $\mathbb{Q}^*$:

$$U^* = \underset{U}{\operatorname{argmin}}\, \mathrm{KL}(q^* \| q). \tag{4}$$

To minimize the KL divergence, the right-hand term of equation (4) can be re-written as:

$$\begin{aligned} U^* &= \underset{U}{\operatorname{argmax}}\, \mathbb{E}_{q^*} \left[\frac{1}{2} \sum_{t=0}^{T-1} (\mathbf{v}_t - \mathbf{u}_t)^\top \Sigma^{-1} (\mathbf{v}_t - \mathbf{u}_t)\right], \\ &= \mathbb{E}_{q^*}[V^{(k)}]. \end{aligned} \tag{5}$$

The optimal control input sequence U^* is obtained by sampling $V^{(k)}$ from the optimal optimality likelihood $\mathbb{Q}^*$ and taking its expectation value in equation (5). However, since direct sampling from $\mathbb{Q}^*$ is not possible, MPPI approximates the optimal optimality likelihood using the importance sampling technique [23] as,

$$U^* = \mathbb{E}_q \left[\frac{q^*(V)}{q(V^{(k)})} V^{(k)}\right] \simeq \sum_{k=0}^{K-1} w\left(V^{(k)}\right) V^{(k)}, \tag{6}$$

$$w\left(V^{(k)}\right) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} \tilde{S}(V^{(k)} + \lambda \sum_{t=0}^{T=1} (\hat{\mathbf{u}}_t - \tilde{\mathbf{u}}_t)^\top \Sigma^{-1} \mathbf{v}_t)\right), \tag{7}$$

where $w(\cdot)$ is the weight function and $\hat{\mathbf{u}}_t$ represents the previous optimal solution, and $\tilde{\mathbf{u}}_t$ represents a nominal control input sequence applied to the system. The MPPI algorithm leverages the law of large numbers to minimize the KL divergence by averaging a sufficient number of weighted samples, obtaining the optimal solution without iterative solution updates.

B. SVGD Algorithm Review

To address the task of minimizing the KL divergence as shown in equation (4) and to identify an appropriate distribution q, we utilize the SVGD method. Similar to MPPI, SVGD aims to approximate the optimal distribution q^*. This method employs particles to represent the optimal action distribution q^*. In this work, the particles symbolize vectors of the difference of control input sequences, which are defined as $\Delta U = (\Delta \mathbf{u}_0,\ \Delta \mathbf{u}_1,\ \ldots, \Delta \mathbf{u}_{T-1})$. During the SVGD process, these particles are adjusted along the gradient direction of optimality likelihood p to converge with the optimal distribution q^*. The optimality likelihood p is a modeled version of the optimal distribution, formulated based on the optimal control problem. Specifically, the particle update process is performed iteratively based on following equations:

$$\Delta U^{(k)} \leftarrow \Delta U^{(k)} + \epsilon \Phi(\Delta U^{(k)}), \tag{8}$$

$$\Phi = \underset{\Phi \in \mathcal{F}}{\operatorname{argmax}} -\nabla_\epsilon \mathrm{KL}(q \| p \simeq q^*), \tag{9}$$

where the step size ϵ is provided as a hyperparameter, and $\mathcal{F}$ is a set of functions with bounded Lipschitz norms. Solving the variational problem shown in equation (9) involves the choice of a smooth function Φ, which affects the computational cost. However, to broaden $\mathcal{F}$, requiring an infinite number of basis functions makes solving the variational problem in Stein Discrepancy computationally expensive [24]. Therefore, the function Φ is considered to reside in the unit ball of a Reproducing Kernel Hilbert Space (RKHS). This RKHS takes the form of $\mathcal{H} = \mathcal{H}_0 \times \ldots \mathcal{H}_0$. $\mathcal{H}_0$ is a reproducing kernel Hilbert space that holds scalar-valued functions, and the kernel function used to characterize the properties of functions in this space is $k(\Delta U',\ \Delta U)$. This kernel function defines the relationship and inner product between different particles $\Delta U'$ and ΔU within $\mathcal{H}_0$, capturing the similarity between particles.

In other words, Φ is expressed as follows:

$$\Phi = \underset{\Phi \in \mathcal{H}}{\operatorname{argmax}} -\nabla_\epsilon \mathrm{KL}(q \| p). \tag{10}$$

The function Φ characterizes the optimal perturbation that maximally reduces the KL divergence. Based on this, particles are updated following the approximate steepest descent direction $\Phi(\Delta U)$:

$$\begin{aligned} \Phi(\Delta U) = \frac{1}{K} \sum_{j=0}^{K-1} \Big[& k(\Delta U^{(j)},\ \Delta U) \nabla_{\Delta U^{(j)}} \log(p(\Delta U^{(j)}) \\ & + \nabla_{\Delta U^{(j)}} k(\Delta U^{(j)},\ \Delta U)\Big]. \end{aligned} \tag{11}$$

The first term in equation (11) represents a force acting on the particle approximation of the optimality likelihood in

the direction of the gradient. This term utilizes the gradient of the optimality likelihood to attract particles towards the optimality likelihood.

On the other hand, the second term serves as a repulsive force. It pushes particles away from each other when they get too close. This repulsive term helps prevent particles from converging to local optima, allowing the method to avoid collapsing into a unimodal distribution.

Unlike conventional Monte Carlo methods, this approach uses gradient-based forces to control interactions between particles. Even when using only a single particle, this method aims to maximize $\log(p(\Delta U^{(j)}))$ using gradient information. This is because the repulsive term vanishes, resulting in $\nabla_{\Delta U} k(\Delta U,\ \Delta U) = 0$.

IV. SPARSE CONTROL POINTS MPPI

A. Overview of the Proposed Method

This subsection provides an overview of the proposed Sparse Control Points MPPI (SCP-MPPI) method. Figure 2 illustrates the overview of SCP-MPPI with and without the SVGD method. SCP-MPPI begins by sampling sparse control points from previous time steps, akin to the original MPPI process. These points are then spline-interpolated to generate smooth control input sequence candidates. However, due to the sparsity of control points, these spline-interpolated samples might not closely approximate the optimal action sequence distribution, potentially reducing solution quality. To enhance solution accuracy, we apply SVGD [9] to directly adjust the spline-interpolated samples, leading to improved outcomes. Notably, this process maintains lower computational complexity compared to SV-MPC [12] since it only modifies sparse control points. SCP-MPPI achieves a smooth, collision-free solution by computing the weighted average of these samples, as depicted in Fig. 2.

B. MPPI with spline interpolation

This subsection details how to incorporate spline interpolation of sparse control points into the MPPI algorithm. We refer this method to as SCP-MPPI w/o SVGD in this paper. This MPPI allows us to find smooth trajectories even when the control points are sparse.

In MPPI, the optimal control input sequence U^* is obtained via equation (6) and it is used as the reference control input sequence to obtain kth control input sequence. The kth control input sequence $U^{(k)}$ is obtained by adding the Gassusian noises to U^*. Then, the generated sequences are evaluated and the optimal sequence is obtained. MPPI iteratively performs these processes.

SCP-MPPI samples the sparse control input points $\tilde{U} \in \mathbb{R}^{m \times M}$ and adds Gaussian noises to them to generate kth sparse control points $\tilde{U}^{(k)}$, where M is the number of sparse control input. Spline interpolation is performed to the sparse control input points and the kth interpolated result is denoted as $U^{(k)}$ that is one control input sequence. The cost evaluation is executed using the interpolated sequence and the optimal sequence is obtained as the weighted average of them. $\tilde{U}^*$, that is the sparse control inputs generating the optimal input sequence, is extracted from the optimal sequence and is fed to the next inference.

C. Transport Interpolated Sparse Control Points by SVGD

SCP-MPPI w/o SVGD can navigate around obstacles but struggles to generate a path close to the optimal one when the initial action sequence distribution deviates from the optimal distribution. To address this, we incorporate the SVGD method, which iteratively refines the sampled M control inputs by updating the control input noises $\Delta\tilde{U}^{(k)}$ using equations (8) and (11). Following this update, spline interpolation is reapplied to the refined control inputs, and their cost is assessed with equation (7). The process concludes with the recalculation of weighted control inputs to determine the actual control input for the robot.

The SVGD method's strength lies in its ability to directly adjust solution candidates, *i.e.* particles, by aligning them with the posterior distribution. This capability ensures that particles, even those initially positioned far from the optimal actions, can be guided closer to them. Thus, SCP-MPPI becomes particularly effective in dynamic environments where the optimal distribution may rapidly change, such as in the presence of unforeseen obstacles.

The pseudo-code for SCP-MPPI is provided in Algorithm 1. Initially, sparse control points are sampled by adding the Gaussian noises and subjected to spline interpolation from line 5 to 7. At line 15, the SVGD method is applied to transport the sparse control inputs. Following this, spline interpolation is performed once more on the transported control inputs. The process culminates at line 28, where the actual control inputs are finalized, creating an optimized control input sequence.

V. EXPERIMENTS

A. Navigation Task

In this work, we used a simulator of a quadrotor and conducted obstacle avoidance experiments. In the simulation, the controlled quadrotor aims to reach a given goal in three types of *forest-like* environments filled with cylindrical obstacles, as shown in Fig. 3. In these environments, the cylindrical obstacles are set, but the quadrotor does not know their positions in advance. Hence, the quadrotor needs to reactively avoid them based on the sensing information from its onboard 2D LiDAR sensor.

B. Formulation of Optimal Control Problem

To apply the standard MPPI and SCP-MPPI to the navigation task, we implemented the following MPC formulation. We first model the dynamics of the controlled quadrotor with a point mass model as follows:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}_t \Delta t, \tag{12}$$

where $\mathbf{u}_t = (v_x, v_y, v_z)^\top$ is the control input vector. The length of the control input sequence, which is an important tuning parameter, was set to 150 in MPPI, that is 150 control points are involved in one control sequence, while it was set to 4 in SCP-MPPI. In SCP-MPPI, these 4 control points are

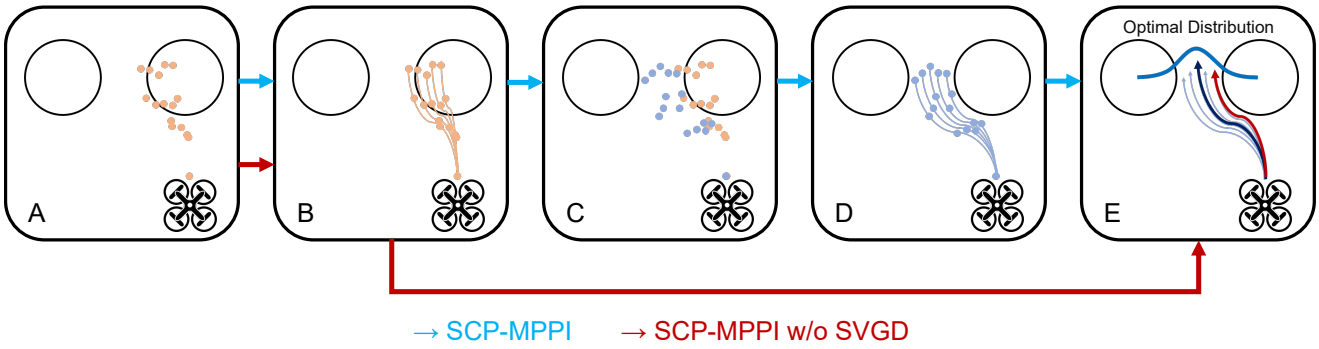

Fig. 2. The overview of SCP-MPPI with and without the SVGD method. (A) Initially, SCP-MPPI sparsely samples control points (illustrated as light red points) and (B) employs spline curves to interpolate these points, creating control input sequences (shown as light red lines). (C) If SVGD is applied, the sparse control points are adjusted (depicted as light blue points), followed by (D) another round of spline interpolation (resulting in light blue lines). (E) The process culminates in determining the optimal action distribution and generating a reactive navigation path (represented by a blue line). While SCP-MPPI without SVGD can navigate around obstacles (indicated by a red line), incorporating SVGD enables SCP-MPPI to approximate the global optimum more closely, even when the initial distribution is not close to the optimal solution.

interpolated into a control input sequence and makes 150 control points.

To allow the quadrotor to smoothly reach at a given goal while avoiding obstacles, we design the cost function $\tilde{S}_k$ as follows:

$$\tilde{S}_k = \sum_{t=0}^{T-1} \left[\Delta \mathbf{x}^\top Q \Delta \mathbf{x} + \frac{1}{2} \mathbf{u}_t^\top R \mathbf{u}_t + \frac{w_d}{d} + C(\mathbf{u}_t) \right], \quad (13)$$

where $\Delta \mathbf{x} = \mathbf{x}_t - \mathbf{x}_d$ represents the position error between the current position to the goal, d denotes the distance to the nearest obstacle, and Q, R, and w_d are weight parameters, respectively. $C(\cdot)$ is an indicator function of constraint conditions and is expressed as follows:

$$C(\mathbf{u}_t) = \begin{cases} 1 + w_v(\|\mathbf{u}_t\|_2 - \|\mathbf{u}_{\max}\|_2) & \text{if } \|\mathbf{u}_t\|_2 > \|\mathbf{u}_{\max}\|_2 \\ 0 & \text{otherwise} \end{cases}, \quad (14)$$

where w_v are weight parameters and $\mathbf{u}_{\max}$ is a constraint condition. We then define the optimality likelihood $p(\Delta\tilde{U})$ for the SVGD in (11) as follows

$$p(\Delta\tilde{U}) = ((\tilde{S}_k - \beta) + 1000)^{-1}, \text{ where } \beta = \min_{\Delta\tilde{U}} \tilde{S}_k, \quad (15)$$

and the kernel function in equation (11) is the following RBF kernel in this paper as

$$k(\Delta\tilde{U}^{(j)}, \Delta\tilde{U}) = \exp\left(-\frac{\|\Delta\tilde{U}^{(j)}, \Delta\tilde{U}\|_2^2}{\sigma} \right), \quad (16)$$

where σ is denoted as

$$\sigma = \frac{\text{median}(\|\Delta\tilde{U}^{(0)}\|^2, \|\Delta\tilde{U}^{(1)}\|^2, \ldots, \|\Delta\tilde{U}^{(K-1)}\|^2)}{\log K}, \quad (17)$$

where $\text{median}(\cdot)$ takes the median of the given value set. Since the cost function shown in equation (13) is not differentiable, we used a numerical differentiation for the gradient shown in equation (11).

TABLE I
SIMULATION RESULTS FOR EACH ENVIRONMENT
(SR: SUCCESS RATE, FT: FLIGHT TIME, AS: AVERAGE SPEED)

Environment (a)	SR [%]	FT [s]	AS [m/s]
SCP-MPPI	100	28.3	0.686
MPPI (K = 50)	100	31.7	0.541
MPPI (K = 1000)	100	27.3	0.606
SCP-MPPI w/o SVGD (K = 50)	100	28.7	0.668
SCP-MPPI w/o SVGD (K = 1000)	100	28.1	0.694
Environment (b)	**SR [%]**	**FT [s]**	**AS [m/s]**
SCP-MPPI	60	23.7	0.707
MPPI (K = 50)	0	-	-
MPPI (K = 1000)	0	-	-
SCP-MPPI w/o SVGD (K = 50)	20	23.9	0.685
SCP-MPPI w/o SVGD (K = 1000)	100	22.0	0.738
Environment (c)	**SR [%]**	**FT [s]**	**AS [m/s]**
SCP-MPPI	20	21.3	0.704
MPPI (K = 50)	0	-	-
MPPI (K = 1000)	0	-	-
SCP-MPPI w/o SVGD (K = 50)	0	-	-
SCP-MPPI w/o SVGD (K = 1000)	70	19.7	0.735

C. Simulation Results

Based on these conditions, we conducted simulation experiments comparing SCP-MPPI, SCP-MPPI w/o SVGD, and the standard MPPI. SCP-MPPI was tested with 50 sample trajectories, while SCP-MPPI w/o SVGD and the vanilla MPPI were tested with 50 and 1000 sample trajectories. The results are summarized in Table I, where K represents the number of sampled trajectories. The criteria for failure were defined as that the quadrotor collided against the obstacles or got stuck in local minima.

In SCP-MPPI, the success rate of obstacle avoidance has improved. This indicates that SCP-MPPI can generate diverse sample trajectories, allowing the controller to predict trajectories for obstacle avoidance. From the results of flight time and average speed, SCP-MPPI w/o SVGD predicts faster control inputs with fewer samples compared to the standard MPPI, enabling exploration of more distant points

Algorithm 1 Sparse Control Points MPPI

Require:
K: Number of samples;
T: Number of time steps;
M: Number of control points;
L: Number of iteration for SVGD;
$\tilde{U} = (\tilde{\mathbf{u}}_0,\ \tilde{\mathbf{u}}_1,\ \ldots,\ \tilde{\mathbf{u}}_{M-1})$: Initial control points;

while task not completed **do**
  $\mathbf{x}_{t_0} \leftarrow$GetState();
  for $k \leftarrow 0$ to $K-1$ **do**
    $\mathbf{x}_0^{(k)} \leftarrow \mathbf{x}_{t_0}$;
    $\Delta\tilde{U}^{(k)} \leftarrow (\Delta\tilde{\mathbf{u}}_0^{(k)},\ \Delta\tilde{\mathbf{u}}_1^{(k)},\ \ldots, \Delta\tilde{\mathbf{u}}_{M-1}^{(k)})$;
    $\tilde{U}^{(k)} \leftarrow (\tilde{\mathbf{u}}_0 + \Delta\mathbf{u}_0^{(k)},\ \tilde{\mathbf{u}}_1 + \Delta\mathbf{u}_1^{(k)},\ \ldots, \tilde{\mathbf{u}}_{M-1} + \Delta\mathbf{u}_{M-1}^{(k)})$;
    $U^{(k)} \leftarrow \mathrm{CubicSpline}(\tilde{U}^{(k)})$;
    for $t \leftarrow 0$ to $T-1$ **do**
      $\mathbf{x}_{t+1}^{(k)} \leftarrow \mathbf{F}(\mathbf{x}_t^{(k)},\ \mathbf{u}_t^{(k)})$;
      $\tilde{S}_k \leftarrow \tilde{S}_k + c(\mathbf{x}_t^{(k)},\ \mathbf{u}_t^{(k)})$;
    end for
  end for
  for $l \leftarrow 0$ to $L-1$ **do**
    for $k \leftarrow 0$ to $K-1$ **do**
      $\Delta\tilde{U}^{(k)} \leftarrow \Delta\tilde{U}^{(k)} + \epsilon\Phi(\Delta\tilde{U}^{(k)})$;
    end for
  end for
  for $k \leftarrow 0$ to $K-1$ **do**
    $U^{(k)} \leftarrow \mathrm{CubicSpline}(\tilde{U}^{(k)})$;
    for $t \leftarrow 0$ to $T-1$ **do**
      $\mathbf{x}_{t+1}^{(k)} \leftarrow \mathbf{F}(\mathbf{x}_t^{(k)},\ \mathbf{u}_t^{(k)})$;
      $\tilde{S}_k \leftarrow \tilde{S}_k + c(\mathbf{x}_t^{(k)},\ \mathbf{u}_t^{(k)})$;
    end for
  end for
  for $m \leftarrow 0$ to $M-1$ **do**
    $\tilde{\mathbf{u}}_m^* = \tilde{\mathbf{u}}_m + \left[\sum_{k=0}^{K-1} \frac{\exp(-\frac{1}{\lambda}\tilde{S}_k)\Delta\mathbf{u}_m^{(k)}}{\sum_{j=0}^{K-1}\exp(-\frac{1}{\lambda}\tilde{S}_j)}\right]$;
  end for
  $U^* \leftarrow \mathrm{CubicSpline}(\tilde{U}^*)$;
  SendCommand $(\mathbf{u}_0^*)$;
  Feed $\tilde{U}^*$ as next initial control points;
  Update the current state after receiving feedback;
  Check for task completion;
end while

and significantly reducing the probability of falling into local minima.

However, as shown in Table I, the flight time with 1000 samples is slower. This could be attributed to modeling errors introduced by using the velocity dynamics model shown in equation (12). In environment (a), the quadrotor maneuvers around obstacles, leading to drift and an increase in flight time. One possible solution is to use alternative dynamics models such as an acceleration dynamics model.

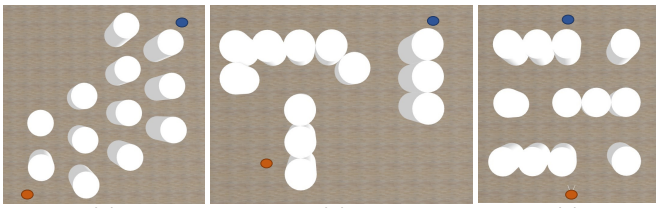

Fig. 3. Experimental environments. Each environment is filled with cylinders of a radius of 0.75 meters. The quadrotor departs from the orange circle and goes to the destination depicted by the blue circle.

Furthermore, SCP-MPPI demonstrates the capability to generate trajectories without getting stuck in local minima even with fewer samples, emphasizing its ability to avoid obstacles. This is due to SCP-MPPI's inherent property of transporting trajectories in a spread-out manner, resulting in reduced chances of collision with obstacles or falling into local minima.

Nevertheless, the superior performance of SCP-MPPI comes at the cost of increased computational overhead. In our implementation, the SCP-MPPI controller runs at 10 Hz, while MPPI and SCP-MPPI w/o SVGD controllers run at 34 Hz. All simulations were conducted on a desktop computer equipped with a Intel(R) Core(TM) i9-10980XE CPU @ 3.00 GHz.

The main reasons for the increased computational cost of SCP-MPPI are associated with trajectory transportation in each iteration of the SVGD algorithm shown in equations (8), (11) and numerical differentiation of the cost function. Possible solutions to address this issue include parallelizing computations on GPUs for speedup and switching to analytical differentiation of the cost function when possible.

VI. Conclusion

In this paper, we proposed the sparse control points model predictive path integral (SCP-MPPI), which outperforms the conventional MPPI method in challenging reactive navigation tasks. Our proposed approach employs flexible control input approximation using spline curves, enabling effective sampling even with sparse control points. Furthermore, we introduced an algorithm for high-precision inference, representing the posterior distribution as a set of multiple particles and optimizing it using Stein Variational Gradient Descent (SVGD). This approach allows us to improve sampling efficiency by transporting samples based on gradient-based information, in contrast to pure Monte Carlo sampling methods. This enhancement increases the feasibility of interpolated samples. The main contribution of this work lies in achieving smooth and collision-free motion even with sparse control points by combining spline interpolation and the SVGD algorithm.

ACKNOWLEDGMENT

This work was supported by KAKENHI under Grant 23K03773.

REFERENCES

[1] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[2] Thomas M. Howard, Colin J. Green, Alonzo Kelly, and Dave Ferguson. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25(6-7):325—-345, 2008.

[3] Zdravko Botev, Dirk Kroese, Reuven Rubinstein, and Pierre L' Ecuyer. Chapter 3. the cross-entropy method for optimization. *Handbook of StatisticsCEM*, 31:35–59, 12 2013.

[4] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. pages 1433–1440, 2016.

[5] Grady Williams1, Andrew Aldrich1, B. Goldfain, J. M. Rehg, B. Boots, and A. Theodorou. Information theoretic mpc for model-based reinforcement learning. *in IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[6] Grady Williams1, Andrew Aldrich1, Evangelos, and A. Theodorou. Information theoretic model predictive control:theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[7] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[8] Marcello Farina, Luca Giulioni, and Riccardo Scattolini. Stochastic linear model predictive control with chance constraints – a review. *Journal of Process Control*, 44:53–67, 2016.

[9] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Neural Information Processing Systems*, 2016.

[10] Taekyung Kim, Gyuhyun Park, Kiho Kwak, Jihwan Bae, and Wonsuk Lee. Smooth model predictive path integral control without smoothing. *IEEE Robotics and Automation Letters*, 7(4):10406–10413, 2022.

[11] Jacob Higgins, Nicholas Mohammad, and Nicola Bezzo. A model predictive path integral method for fast, proactive, and uncertainty-aware uav planning in cluttered environments. *arXiv preprint arXiv:2308.00914*, 2023.

[12] Alexander Lambert, Fabio Ramos, Byron Boots, Dieter Fox, and Adam Fishman. Stein variational model predictive control. 155:1278–1297, 2021.

[13] Baolin Liu, Gedong Jiang, Fei Zhao, and Xuesong Mei. Collision-free motion generation based on stochastic optimization and composite signed distance field networks of articulated robot. *arXiv preprint arXiv:2306.04130*, 2023.

[14] Isin M Balci, Efstathios Bakolas, Bogdan Vlahov, and Evangelos A Theodorou. Constrained covariance steering based Tube-MPPI. In *American Control Conference*, pages 4197–4202. IEEE, 2022.

[15] Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M Rehg, and Evangelos A Theodorou. Robust sampling based model predictive control with sparse objective information. In *Robotics: Science and Systems*, volume 14, page 2018, 2018.

[16] Ji Yin, Zhiyuan Zhang, Evangelos Theodorou, and Panagiotis Tsiotras. Trajectory distribution control for model predictive path integral control using covariance steering. In *International Conference on Robotics and Automation*, pages 1478–1484. IEEE, 2022.

[17] Jacob Sacks and Byron Boots. Learning sampling distributions for model predictive control. In *Conference on Robot Learning*, pages 1733–1742. PMLR, 2023.

[18] Dylan M Asmar, Ransalu Senanayake, Shawn Manuel, and Mykel J Kochenderfer. Model predictive optimized path integral strategies. In *International Conference on Robotics and Automation*, pages 3182–3188. IEEE, 2023.

[19] Ihab S Mohamed, Junhong Xu, Gaurav Sukhatme, and Lantao Liu. Towards efficient MPPI trajectory generation with unscented guidance: U-MPPI control strategy. *arXiv preprint arXiv:2306.12369*, 2023.

[20] Masashi Okada and Tadahiro Taniguchi. Variational inference MPC for Bayesian model-based reinforcement learning. In *Conference on robot learning*, pages 258–272. PMLR, 2020.

[21] Ziyi Wang, Oswin So, Jason Gibson, Bogdan Vlahov, Manan S Gandhi, Guan-Horng Liu, and Evangelos A Theodorou. Variational inference MPC using tsallis divergence. *Robotics: Science and Systems*, 2021.

[22] Lucas Barcelos, Alexander Lambert, Rafael Oliveira, Paulo Borges, Byron Boots, and Fabio Ramos. Dual online Stein variational inference for control and dynamics. *Robotics: Science and Systems*, 2021 (in print).

[23] Teun Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978.

[24] Jackson Gorham and Lester Mackey. Measuring sample quality with stein's method. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.