One-Step Model Predictive Path Integral for Manipulator Motion Planning Using Configuration Space Distance Fields

Yulin Li, Tetsuro Miyazaki and Kenji Kawashima

Abstract— Motion planning for robotic manipulators is a fundamental problem in robotics. Classical optimization-based methods typically rely on the gradients of signed distance fields (SDF) to impose collision-avoidance constraints. However, these methods are susceptible to local minima and may fail when the SDF gradients vanish. Recently, Configuration Space Distance Fields (CDF) have been introduced, which directly model distances in the robot's configuration space. Unlike workspace SDF, CDF are differentiable almost everywhere and thus provide reliable gradient information. On the other hand, gradient-free approaches such as Model Predictive Path Integral (MPPI) control leverage long-horizon rollouts to achieve collision avoidance. While effective, these methods are computationally expensive due to the large number of trajectory samples, repeated collision checks, and the difficulty of designing cost functions with heterogeneous physical units. In this paper, we propose a framework that integrates CDF with MPPI to enable direct navigation in the robot's configuration space. Leveraging CDF gradients, we unify the MPPI cost in joint-space and reduce the horizon to one step, substantially cutting computation while preserving collision avoidance in practice. We demonstrate that our approach achieves nearly 100% success rates in 2D environments and consistently high success rates in challenging 7-DOF Franka manipulator simulations with complex obstacles. Furthermore, our method attains control frequencies exceeding 750 Hz, substantially outperforming both optimization-based and standard MPPI baselines. These results highlight the effectiveness and efficiency of the proposed CDF-MPPI framework for high-dimensional motion planning.

I. INTRODUCTION

Motion planning is one of the most active research areas in robotic control, with extensive efforts devoted to achieving efficient and collision-free robot motions. Many studies [1], [2] formulate this problem as an optimization problem, where the signed distance field (SDF) is used to represent obstacle proximity and its gradient is employed to guide the optimization toward collision-free trajectories. In practice, classical approaches such as CHOMP [2] precompute SDF on a voxel grid via the Euclidean Distance Transform (EDT), while TrajOpt [1] repeatedly queries distances using GJK [3] between convex bodies. These computations are often implemented on CPUs in standard libraries, which is generally sufficient for static environments and moderate-scale problems. Recent work has explored GPU acceleration of both SDF construction and GJK queries to handle dynamic scenes and large-scale planning more efficiently, though they can still be computationally expensive due to frequent voxel updates and the iterative nature of distance queries.

More recent works, such as [4], which adopt regression-based formulations, and [5], which leverage neural networks, demonstrate that SDF can be efficiently fitted and evaluated
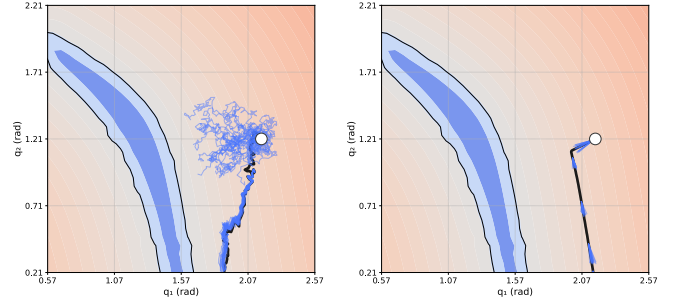

Fig. 1. Comparison between the original MPPI and our proposed method in the two-link robot configuration space. The irregular shapes on the left represent obstacles, blue thin lines denote sampled trajectories at each step, the black line indicates the resulting trajectory, and the white dot marks the start configuration. **Left:** the original MPPI samples long-horizon trajectories at each step. **Right:** our method leverages CDF to sample one-step horizon trajectories at each step.

on GPU. These approaches enable faster and more accurate SDF computation. However, as in previous studies, gradient information is required to achieve collision-free motion. For robotic arms, this introduces two issues: (i) the SDF gradient may vanish near obstacles [6], and (ii) the distribution of SDF values in the configuration space is highly non-uniform [7].

To overcome these limitations, sampling-based methods have emerged as a viable alternative. Their key advantage lies in the fact that they do not require gradient information, thereby avoiding issues such as the vanishing-gradient problem in SDF. Representative approaches include Rapidly-exploring Random Trees (RRT) [8] and Probabilistic Roadmaps (PRM) [9]. However, these methods are often inefficient due to their reliance on extensive random sampling.

Model Predictive Control (MPC) has also gained widespread popularity in robotic control [10], [11], but traditional MPC formulations still require gradient information, which limits their applicability in high-dimensional and non-differentiable environments.

Model Predictive Path Integral (MPPI) control was introduced in [12], integrating the sampling paradigm with the predictive principles of MPC. By updating the sampling distribution iteratively, MPPI improves sampling efficiency while eliminating the need for optimization solvers. Its extension to robotic manipulators was presented in [13], demonstrating promising results. In this framework, collision-free trajectories are obtained by penalizing predicted collisions within the cost function, rather than relying on gradient information, thereby mitigating the vanishing-gradient issue. Nevertheless, two key challenges remain. First, the cost function design in [13] is intricate, as it involves heterogeneous

components that are not directly comparable. Second, the method is computationally expensive: at each time step, a large number of finite-horizon trajectory rollouts must be simulated, each requiring frequent collision checks with obstacles. This significantly increases computation time and imposes heavy demands on GPU resources [5].

Configuration Space Distance Fields (CDF), proposed in [7], provide a promising alternative. Unlike traditional SDF defined in the workspace, CDF directly construct distance fields in the configuration space, thereby addressing both the non-uniformity and vanishing-gradient problems. CDF thus offer an attractive framework for robot motion planning, as their gradients can be directly exploited for navigation in configuration space [7]. However, optimization-based methods that rely on CDF are susceptible to local minima, which may result in navigation failures. Moreover, such optimization problems typically require repeated iterations, making the computation slow especially in high-dimensional configuration spaces.

In this work, we integrate the MPPI framework with CDF and propose a simple yet efficient algorithm that addresses several limitations of existing methods. Our goal is to control an n-DOF robotic arm to perform fast and collision-free motions.

1) **Avoiding local optima:** By eliminating optimization-based procedures, our method reduces the risk of convergence to local minima.

2) **Unified and simplified cost function:** Leveraging the property that CDF provide well-distributed, non-vanishing gradients in the configuration space, we redesign the MPPI cost function. In particular, we employ angular navigation to normalize and unify the scale of different cost terms.

3) **Faster computation:** The redesigned cost function enables one-step trajectory sampling instead of requiring long-horizon rollouts, thereby significantly reducing computation time. Most computations can be efficiently executed on GPU (Fig. 1 illustrates a comparison between our method and the original MPPI).

II. RELATED WORKS

A. Model Predictive Path Integral

The MPPI framework operates as follows. For a discrete-time robotic system at time step t, the control input $\mathbf{u}$ represents the joint velocity. The input $\mathbf{u}$ is sampled from a policy $\pi_t = \prod_{h=0}^{H-1} \pi_{t,h}$ where H is the prediction horizon and $\pi_{t,h}$ is a Gaussian distribution with mean $\boldsymbol{\mu}_{t,h}$ and covariance $\boldsymbol{\Sigma}_{t,h}$.

At each time step t, the MPPI algorithm proposed by [13] draws a batch of N control sequences $\mathbf{u}_{i=1..N}^{h=1..H}$ from the current policy π_t. For each sampled sequence, the corresponding cost $c_{i,h}$ is computed from the state trajectories $x_{i,h}$ obtained by propagating the dynamics of the robot using the sampled controls.

The Gaussian policy parameters $\pi_{t,h} = \mathcal{N}(\boldsymbol{\mu_{t,h}}, \boldsymbol{\Sigma}_{t,h})$ are

then updated according to the following weighted rules:

$$\boldsymbol{\mu}_{t+1,h} \leftarrow (1 - \alpha_\mu)\boldsymbol{\mu}_{t,h} + \alpha_{\boldsymbol{\mu}} \frac{\sum_{i=1}^{N} w_i \mathbf{u}_{i,h}}{\sum_{i=1}^{N} w_i}, \tag{1a}$$

$$\boldsymbol{\Sigma}_{t+1,h} \leftarrow (1 - \alpha_{\boldsymbol{\Sigma}})\boldsymbol{\Sigma}_{t,h} + \alpha_{\boldsymbol{\Sigma}} \frac{\sum_{i=1}^{N} w_i(\mathbf{u}_{i,h} - \boldsymbol{\mu}_{t,h})(\mathbf{u}_{i,h} - \boldsymbol{\mu}_{t,h})^\top}{\sum_{i=1}^{N} w_i}. \tag{1b}$$

where $\alpha_{\boldsymbol{\mu}}$ and $\alpha_{\boldsymbol{\Sigma}}$ are filtering coefficients. The weight for each sample is defined as $w_i = \exp\left(-\frac{1}{\beta}\hat{C}_i\right)$, where β is a temperature parameter for the cost. The cost value $\hat{C}(\mathbf{x}_t, \mathbf{u})$ is computed as

$$\hat{C}(\mathbf{x}_t, \mathbf{u}) = \sum_{h=0}^{H-2} \gamma^h c(\mathbf{x}_{n,h}, \mathbf{u}_{n,h}) + \gamma^{H-1}\hat{c}(\mathbf{x}_{n,H-1}, \mathbf{u}_{n,H-1}), \tag{2}$$

where γ is the discount factor, $c(\cdot)$ is the stage cost, and $\hat{c}(\cdot)$ denotes the terminal cost, and n is the each trajectory. This weighting scheme biases the sampling toward lower-cost control sequences.

The stage cost $c_{\mathbf{x},\mathbf{u}}$ is composed of multiple terms, including joint-limit avoidance, self-collision and collision avoidance, contingency stopping, and manipulation cost:

$$c_{\mathbf{x},\mathbf{u}} = \alpha_s c_{stop} + \alpha_m c_{mani} + \alpha_c c_{col} + \alpha_j c_{joint}. \tag{3}$$

Here, $\alpha_s, \alpha_m, \alpha_c$, and α_j are weighting coefficients for the respective cost components. Further details can be found in [13].

B. Configuration Space Distance Fields

We first define the SDF of the robot, denoted by f_s, as a function that assigns a signed distance between a point in the workspace $\mathbf{p}$ and the robot at configuration $\mathbf{q}$:

$$f_s(\mathbf{p}, \mathbf{q}) = \pm \min_{\mathbf{p}' \in \mathcal{R}(\mathbf{q})} \|\mathbf{p} - \mathbf{p}'\|, \tag{4}$$

where $\mathcal{R}(\mathbf{q})$ represents the set of points occupied by the robot at configuration $\mathbf{q}$, and $\mathbf{p}'$ is the closest point on the robot to $\mathbf{p}$. The CDF, denoted by f_c, measures the minimum distance from a robot configuration $\mathbf{q}$ to the set of configurations that are in contact with obstacles in the workspace [7]:

$$f_c(\mathbf{p}, \mathbf{q}) = \min_{\mathbf{q}'} \|\mathbf{q} - \mathbf{q}'\|, \tag{5}$$

subject to the constraint that $f_s(\mathbf{p}, \mathbf{q}') = 0$, where $\mathbf{q}'$ denotes a configuration in contact with the obstacle and $\mathbf{p}$ is a corresponding contact point in the workspace. In practice, $f_c(\mathbf{p}, \mathbf{q})$ can be encoded using a neural network. The CDF possesses several desirable properties. In particular, it satisfies the eikonal equation $\|\nabla_{\mathbf{q}} f_c(\mathbf{p}, \mathbf{q})\| = 1$ almost everywhere in the configuration space. This property makes CDF especially well-suited for motion planning, since the gradient $\nabla_{\mathbf{q}} f_c$ naturally points in the locally optimal escape direction away from the nearest obstacle, thereby providing a direct guidance signal for collision avoidance.

III. METHOD

We aim to control an n-DOF robotic arm to perform fast, collision-free motions. In this section, we present a method that combines MPPI and CDF to achieve this goal.

The robot dynamics are modeled as:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \Delta t \cdot \mathbf{u}. \tag{6}$$

where $\mathbf{q}_t$ be the robot configuration at time t, $\mathbf{q}_{t+1}$ the configuration at the next time step, $\mathbf{u}$ is the joint input, and Δt is the discrete control time step.

A. Improved Model Predictive Path Integral with an Angle-Based Cost Function

We propose an *angle-based* cost function that unifies the cost terms into a common unit. Let $\mathbf{q}_f$ the goal configuration, and $\mathbf{p}$ is the obstacle point in the work space. The proposed cost is defined as:

$$c(\theta_1, \theta_2) = \alpha_1 \theta_1 + \alpha_2 \theta_2, \tag{7}$$

where $\alpha_1, \alpha_2 > 0$ are scalar weights. $\theta_1 \in [0, \pi]$ is the angle between the obstacle normal direction at $\mathbf{q}_t$ and the motion vector $\mathbf{q}_{next} = (\mathbf{q}_{t+1} - \mathbf{q}_t)$, while $\theta_2 \in [0, \pi]$ is the angle between the goal vector $\mathbf{e}_t = (\mathbf{q}_f - \mathbf{q}_t)$ and the motion vector $\mathbf{q}_{next}$. When $\theta_2 = 0$, the robot moves directly toward the goal; $\theta_2 = \pi$ indicates motion directly away from it. And θ_2 can be computed as:

$$\theta_2(\mathbf{q}_t, \mathbf{q}_{t+1}) = \arccos \frac{\mathbf{q}_{next} \cdot \mathbf{e}_t}{\|\mathbf{q}_{next}\| \|\mathbf{e}_t\|}, \tag{8}$$

For the θ_1, we calculated from

$$\theta_1(\mathbf{q}_t, \mathbf{q}_{t+1}) = \begin{cases} 0, & \text{if } \theta_1^\star < \frac{\pi}{2} \text{ or} \\ & \quad f_c(\mathcal{P}, \mathbf{q}_t) \geq d_{\text{act}} \text{ or} \\ & \quad f_c(\mathcal{P}, \mathbf{q}_t) \geq \|\mathbf{e}_t\|, \\ \theta_1^\star, & \text{otherwise.} \end{cases} \tag{9}$$

$$\theta_1^\star = \arccos \frac{\mathbf{q}_{next} \cdot \nabla_{\mathbf{q}} f_c(\mathbf{p}, \mathbf{q}_t)}{\|\mathbf{q}_{next}\| \|\nabla_{\mathbf{q}} f_c(\mathbf{p}, \mathbf{q}_t)\|}, \tag{10}$$

The obstacle normal direction is obtained from the gradient of the collision CDF, $\nabla_{\mathbf{q}} f_c(\mathbf{p}, \mathbf{q}_t)$, which points towards the optimal escape direction from the nearest obstacle. In our implementation, θ_1 is set to zero under the following conditions:

- **Angle condition:** If $\theta_1 < \pi/2$, the motion naturally moves away from the obstacle, and collision will not occur (see Eq. 9 and Fig. 2(Left)).
- **Distance threshold:** If the distance to the obstacle $f_c(\mathcal{P}, \mathbf{q}_t)$ exceeds a specified activation threshold d_{act} obtained from the CDF, since obstacle avoidance is unnecessary when the robot is already far from the obstacle.
- **Goal proximity:** If the distance to the obstacle is larger than the distance to the goal, i.e., $f_c(\mathcal{P}, \mathbf{q}_t) \geq \|\mathbf{e}_t\|$, as illustrated in Fig. 2(Right) and Fig. 3, it is more efficient to move directly towards the goal rather than proactively avoiding the obstacle.
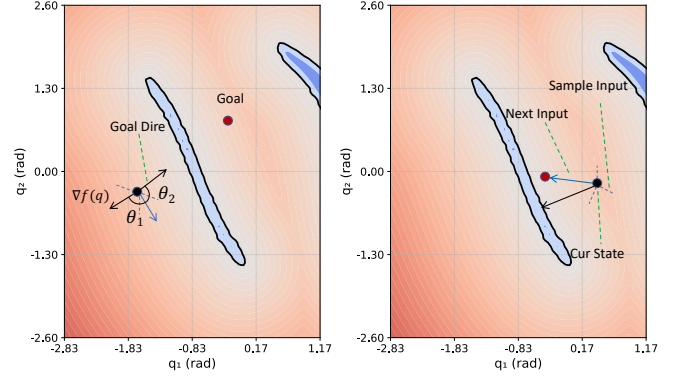

Fig. 2. Two-link robot in the configuration space. Irregular shapes represent obstacles, and the black dot denotes the current state of the robot. **Left**: The blue arrow indicates the actual motion direction of the robot at the next step, forming two angles with the CDF gradient and the goal state. **Right**: Illustration showing why the obstacle-avoidance angle cost is set to zero when the robot–obstacle distance exceeds the robot–goal distance.
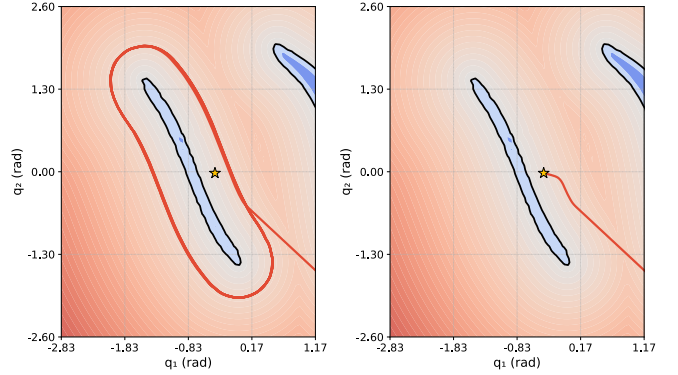

Fig. 3. **Left**: If the robot–obstacle distance exceeds the robot–goal distance while the obstacle cost remains active, the robot may circle around the obstacle. **Right**: When the obstacle cost is set to zero under the same condition, the robot converges to the goal more directly. The star marker indicates the goal configuration.

This design offers two advantages: (1) all cost terms are expressed in the same unit (angle), making the weight selection more intuitive; (2) obstacle avoidance and goal-seeking behaviors are directly captured in a single unified metric. In MPPI, a batch of N control sequences is sampled from π_t, and each sequence is rolled out over a prediction horizon H to compute the total cost. With the proposed angle-based cost, collision risk and goal progress can be evaluated locally at the current time step, without simulating the full horizon. Therefore, we modify the sampling policy to generate only N single-step control inputs ($H = 1$) from π_t, compute $\mathbf{q}_{next}$, and evaluate the cost directly. This reduces computation while retaining effective obstacle avoidance and goal-directed motion.

For the remaining cost terms in Eq. 3, the manipulability term c_{mani} is omitted, as our planning is conducted directly in the configuration space and thus does not suffer from the low-manipulability issue reported in [4]. For the collision cost c_{col}, we also treat the robot itself as an obstacle in the CDF function to account for self-collisions. For the joint-limit cost c_{joint}, we employ a simple projection strategy to

avoid joint-limit violations, which will be detailed in the next section. With these considerations, the cost function in Eq. 2 can be rewritten as

$$\hat{C}(\mathbf{x}_t, \mathbf{u}) = \hat{c}(\mathbf{x}_n). \qquad (11)$$

B. One-step MPPI for Configuration Space Motion Planning

To handle joint limits, we simply project $\mathbf{u}$ into feasible control set,

$$\mathcal{U}_{\text{step}}(\mathbf{q}) = \{\mathbf{u} \mid \mathbf{q} + \Delta t\, \mathbf{u} \in [\mathbf{q}_{\min}, \mathbf{q}_{\max}]\}. \qquad (12)$$

Equivalently, this yields the elementwise bounds

$$\mathbf{u}_{\min}(\mathbf{q}) = \frac{\mathbf{q}_{\min} - \mathbf{q}}{\Delta t}, \qquad \mathbf{u}_{\max}(\mathbf{q}) = \frac{\mathbf{q}_{\max} - \mathbf{q}}{\Delta t}. \qquad (13)$$

The projection of any control $\mathbf{u} \in \mathbb{R}^n$ onto $\mathcal{U}_{\text{step}}(\mathbf{q})$ is the elementwise clipping

$$\text{proj}_{\mathcal{U}_{\text{step}}(\mathbf{q})}(\mathbf{u}) = \text{clip}\big(\mathbf{u}; \mathbf{u}_{\min}(\mathbf{q}), \mathbf{u}_{\max}(\mathbf{q})\big), \qquad (14)$$

i.e., for each coordinate $i = 1, \ldots, m$,

$$\big[\text{proj}_{\mathcal{U}_{\text{step}}(\mathbf{q})}(\mathbf{u})\big]_i = \min\Big\{\max\big(u_i, [\mathbf{u}_{\min}(\mathbf{q})]_i\big), [\mathbf{u}_{\max}(\mathbf{q})]_i\Big\}. \qquad (15)$$

We have discussed the main idea of our method in the previous sections. The complete algorithm is presented in Algorithm 1. Regarding the choice of hyperparameters, the relative magnitudes of α_1 and α_2 determine the trade-off between obstacle avoidance and goal-seeking. If a safer behavior is desired, α_1 should be set larger than α_2 to penalize motions that approach obstacles more heavily. In Algorithm 1 (line 5), the CDF value $\min_{\mathbf{p}'} f_c(\mathcal{P}, \mathbf{q}_t)$ is obtained by evaluating the distance between the current configuration $\mathbf{q}_t$ and all obstacle points in the point cloud $\mathcal{P}$, selecting the minimal distance, and computing the corresponding gradient $\nabla_{\mathbf{q}} f_c(\mathcal{P}, \mathbf{q}_t)$.

IV. NUMERICAL SIMULATIONS

In this section, we report simulation results on a two-link robot and a Franka Emika robot to evaluate the proposed method. The experiments are designed to (i) demonstrate the necessity of updating both the mean and covariance of the sampling distribution, (ii) show that naïve sampling without distribution updates is insufficient, (iii) assess the effectiveness of our cost-function design, and (iv) compare our approach against representative baselines. Finally, we provide a qualitative analysis to further validate the method. All experiments were performed on a workstation with an AMD Ryzen 7000 CPU and an NVIDIA RTX 4070 GPU.

A. 2-DoF Two-Link Robot

We first evaluate the proposed method on a 2D two-link robot with link lengths $l_1 = l_2 = 2$ and joint limits $q \in [-\pi, \pi]$ for each joint. Two circular obstacles are placed at positions $(2.3, -2.3)$ and $(0.0, 2.45)$, each with a radius of 0.3. The start configuration is $\mathbf{q}_s = [2.1, 1.2]$, and the goal configurations are $\mathbf{q}_f = [-2.1, -0.9]$ and $[-0.5, 0.0]$, and control time step to $\Delta t = 0.01s$.

We compare our method with the following baselines: **Random Sampling**: At each time step, N random control

---

Algorithm 1 One-step MPPI for Configuration Space Motion Planning

---
Input: Initial configuration $\mathbf{q}_s$; goal configuration $\mathbf{q}_f$; obstacle point cloud $\mathcal{P}$; joint limits $(\mathbf{q}_{\min}, \mathbf{q}_{\max})$; time step Δt; number of samples N; angle-based cost weights (α_1, α_2); obstacle threshold distance d_{act}. MPPI parameters.

Output: Collision-free joint trajectory $\mathbf{q}_{0:T}$
1: Initialize $\mathbf{q}_0 \leftarrow \mathbf{q}_s$
2: Initialize Gaussian policy $\pi_0 = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$
3: **for** $t = 0$ **to** $T - 1$ **do**
4:     Sample N control inputs $\{\mathbf{u}_i\}_{i=1}^N \sim \pi_t$ {one-step $(H = 1)$}
5:     Compute CDF value $\min_{\mathbf{p}'} f_c(\mathcal{P}, \mathbf{q}_t)$ and its gradient $\nabla_{\mathbf{q}} f_c(\mathcal{P}, \mathbf{q}_t)$
6:     **for** $i = 1$ **to** N **do**
7:         Predict $\mathbf{q}_{t+1}^{(i)} = \mathbf{q}_t + \Delta t \cdot \mathbf{u}_i$
8:         Compute angles $\theta_1^{(i)}$, $\theta_2^{(i)}$ using Eq. 8 Eq. 9 and Eq. 10.
9:         Evaluate cost $c^{(i)} = \alpha_1 \theta_1^{(i)} + \alpha_2 \theta_2^{(i)}$
10:     **end for**
11:     Update policy parameters $\boldsymbol{\mu}_{t+1}$, $\boldsymbol{\Sigma}_{t+1}$ using Eq. 1a and Eq. 1b
12:     Set the next control input $\mathbf{u}_{\text{des}} \leftarrow \boldsymbol{\mu}_{t+1}$
13:     Calculate the next state $\mathbf{q}_{t+1} \leftarrow \mathbf{q}_t + \Delta t\, \text{proj}(\mathbf{u}_{\text{des}})$
14:     **if** $\|\mathbf{q}_{t+1} - \mathbf{q}_f\| < allow\_range$ **then**
15:         **break**
16:     **end if**
17: **end for**

---

inputs are sampled, the next configuration is computed, and the state with the lowest cost (Eq. 7 is selected. **Proposed Method**: As described in Algorithm. 1.

The results are shown in Fig. 4(Left). For the random sampling baseline, since it does not exploit information from previous steps, the trajectory initially exhibits a tendency to avoid obstacles but ultimately collides with them. In contrast, our proposed cost function not only ensures successful obstacle avoidance but also generates high-quality trajectories. This validates both the effectiveness of our cost function design and the rationale for deactivating the obstacle-avoidance penalty when the robot is sufficiently far from obstacles.

We further provide a comprehensive comparison with other baseline methods: the original MPPI [13] and the QP-based method from [7], solved using IPOPT [14]. The start and goal configurations are the same as in the previous experiment, with an additional test case for the QP method (labeled QP 1), where the start is $[2.5, 0.5]$ and the goal is $[-2.0, 0.3]$. The cost function used in the 2-DoF MPPI experiment is defined as

$$c_{x,u} = \alpha_g c_{\text{goal}} + \alpha_c c_{\text{col}} + \alpha_j c_{\text{joint}} + \alpha_s c_{\text{stay}}. \qquad (16)$$

where c_{goal} denotes the terminal cost measuring the error between the predicted final configuration and the target configuration $\mathbf{q}_f$, while c_{stay} penalizes trajectories that remain stationary over time [4].
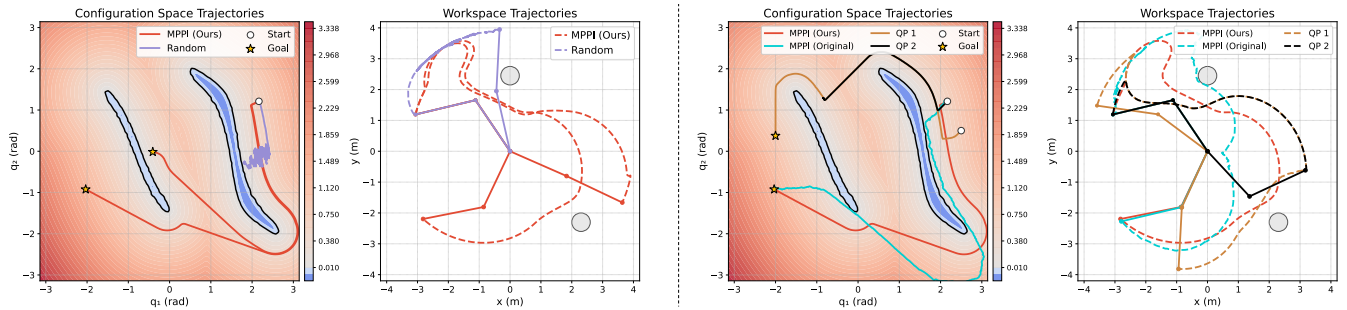
Fig. 4. Comparison of different methods in both configuration space and workspace trajectories. The background color encodes the CDF value. **Left**: Trajectories from the same start configuration but different goals, used to validate the effectiveness of our cost function design. A purely random baseline, which samples controls independently at each step without policy updates, highlights the necessity of distribution updates in MPPI. **Right**: Comparison with MPPI and an optimization-based method. For QP2 (black), the start and goal are identical to those used in MPPI (Original) and MPPI (Ours), but the trajectory converges to a local optimum and fails to progress. In contrast, QP1, with different start and goal states, successfully reaches the target.

TABLE I
COMPARISON OF DIFFERENT METHODS IN THE 2D TWO-LINK ROBOT

Method	Success Rate (%)	Path Length	Avg. Steps
MPPI (Ours)	99.6	5.17	213
MPPI (Original) [13]	71.2	4.40	155
QP (IPOPT) [7]	64.1	3.75	107

The trajectories in configuration space are shown in Fig. 4(Right). The QP-based method [7] occasionally fails to reach the goal for certain start–goal pairs, as the control input oscillates between two states, leading to stagnation and convergence to a local optimum (QP 2, with the same start and goal as MPPI (Ours) and MPPI (Original)). For comparison, in QP 1 with a different start–goal pair, the method successfully reaches the goal. The original MPPI produces trajectories that often run close to obstacles and joint limits, and the resulting paths are generally less smooth.

We also compare the methods using three evaluation metrics: success rate, average number of steps to reach the goal, and average path length, where the path length is defined as

$$L(\mathbf{q}_{0:T}) = \sum_{t=0}^{T-1} \left\| \mathbf{q}_{t+1} - \mathbf{q}_t \right\|_2, \quad (17)$$

measured in radians in the configuration space. All methods are tested under the same random seed with identical sampling rules for start and goal states, repeated 500 times. The results are summarized in Table. I. Our proposed method achieves the highest success rate. Although it requires more steps and produces slightly longer paths, this can be attributed to its ability to avoid local optima and handle longer trajectories effectively. The failure cases mainly occur when the start configuration is very close to an obstacle, causing collisions before the distribution has been sufficiently updated. In contrast, the original MPPI may still result in collisions with obstacles or violations of joint limits due to the difficulty of appropriately balancing the weights among different cost terms.

B. 7-DOF Franka Robot

In this section, we compare our method with the QP approach from [7] under two scenarios.
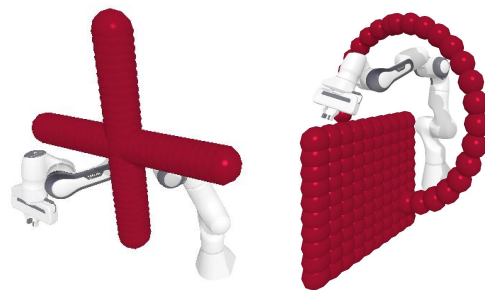

Fig. 5. Experimental setups for motion planning with the 7-DOF Franka robot. **Left**: Scenario 1 with 60 cross-shaped obstacles. The start and goal configurations are fixed, and the obstacle is shifted in each trial such that the linear input between start and goal results in collisions. **Right**: Scenario 2 with 123 obstacles composed of wall and ring structures. The start and goal configurations are randomly sampled, and their linear input also results in collisions.

In the first scenario, shown in Fig. 5(Left), we follow the experimental setup of [4]. The start and goal configurations are fixed, while the cross-shaped obstacle is shifted in each trial. The robot is commanded using incremental steps of the form

$$\mathbf{u} = k(\mathbf{q}_f - \mathbf{q}_s). \quad (18)$$

where k is a small positive coefficient. We refer to this update rule as the *linear dynamics* model. This setup ensures that, across 100 trials, the linear input between the start and goal consistently leads to collisions with the obstacle. In the second scenario, shown in Fig. 5(Right), both the start and goal configurations are randomly sampled. Again, 100 trials are conducted, and for each start–goal pair the linear input defined in Eq. 18 also results in collisions.

The results are summarized in Table II. Our method achieves consistently high success rates. The path length results indicate that, for the QP method, longer trajectories are more likely to become trapped in local optima. An example of the QP method being trapped in a local optimum, leading to oscillatory inputs, is illustrated in Fig. 6. In terms of computational efficiency, our approach achieves an average of 776 Hz in Scenario 1 (with 60 obstacles) and 718 Hz in Scenario 2 (with 123 obstacles).

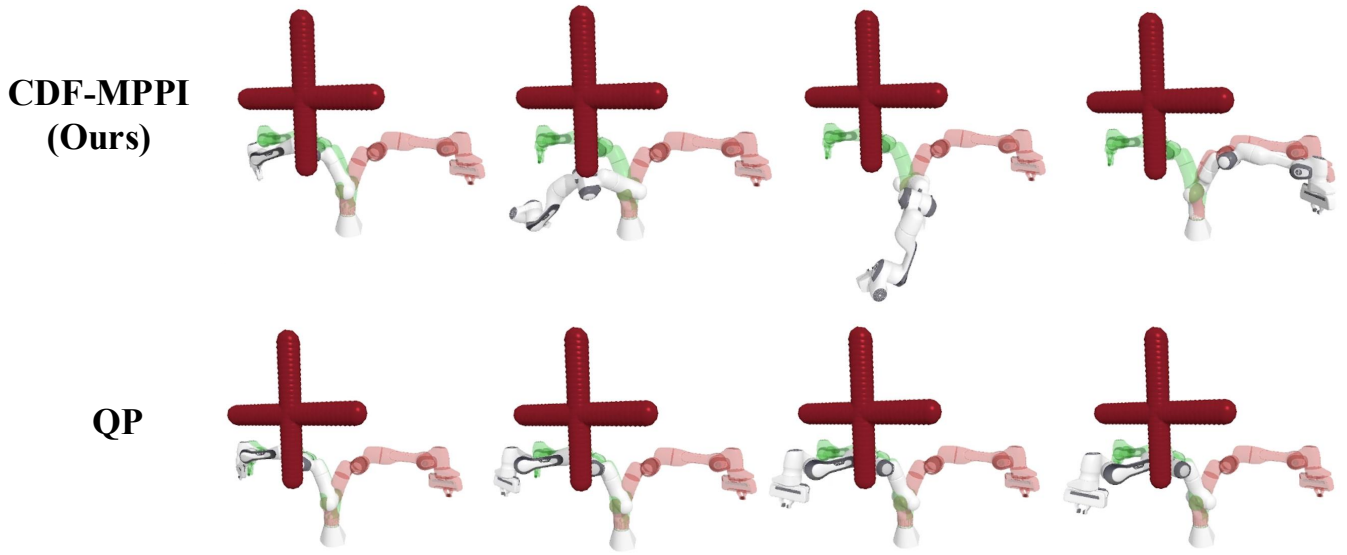We further compare the computation frequency with the

Fig. 6. Comparison between the QP method and MPPI (ours) in Scenario 1. The green marker indicates the start posture, while the red marker indicates the goal posture. **Top**: Our method successfully avoids the cross obstacle and reaches the goal. **Bottom**: The QP method becomes trapped near the obstacle, exhibiting oscillatory behavior around it and failing to make further progress due to local optimality.

TABLE II

COMPARISON OF DIFFERENT METHODS IN THE 7-DOF FRANKA ROBOT

(TWO SCENARIOS)

Method	Scenario 1			Scenario 2		
	Succ. (%)	Len.	Hz	Succ. (%)	Len.	Hz
MPPI (Ours)	100	4.57	776(59)	86	7.85	718(54)
QP (IPOPT) [7]	13	4.30	222(30)	14	6.74	228(24)

MPPI method from cuRobo [15]. We do not directly compare our method with the original MPPI in the configuration space, primarily because it is challenging to design a suitable cost function. The original MPPI framework is tailored for end-effector tracking, where multiple joint configurations correspond to the same end-effector pose due to the redundancy of the 7-DOF Franka robot. However, when the terminal cost is defined in the configuration space, the solution becomes unique, which makes it difficult for MPPI to converge. Therefore, we only compare the control frequency in this case. Specifically, we provide MPPI with the end-effector poses corresponding to the target joint configurations as its input. Under this setup, MPPI achieves an average of **61 Hz** in Scenario1 and **28 Hz** in Scenario2. As the number of obstacles increases, collision-checking time also grows. In contrast, [15] reported that MPPI can reach up to 500 Hz on an RTX 4090 when the number of obstacles is relatively small. Compared to this, our method sustains significantly higher control frequencies while maintaining near-perfect success rates.

V. CONCLUSION

We have proposed a method that integrates CDF with MPPI. Our approach effectively leverages the complementary strengths of both CDF and MPPI.

The proposed method achieves control frequencies exceeding 750 Hz with consistently high success rates, significantly outperforming the QP baseline, which is prone to local optima. Compared to the standard MPPI framework, our approach simplifies the cost function, unifies its units, and exploits the gradient information provided by CDF to shorten the prediction horizon, thereby attaining substantially faster control frequencies.

Future work will focus on further optimizing the CDF representation. One direction is to extend CDF beyond point-based sampling in the workspace. In our current experiments, we approximate obstacles using the centers of bounding spheres; a promising extension would allow CDF to compute the shortest distance in configuration space to entire spherical regions. Another direction is to improve robustness near obstacles. When the CDF values are small—i.e., when the manipulator is close to obstacles—collisions can occur due to the highly non-convex nature of the configuration space. Developing strategies that enable safe initialization and reliable avoidance in such regions is an important avenue for future research.

VI. APPENDIX

The QP formulation used in this paper follows [7]:

$$\mathbf{u}_k^* = \arg\min_{\mathbf{q},u} \ e(\mathbf{q}_k)^\top H e(\mathbf{q}_k) + \mathbf{u}_k^\top R \mathbf{u}_k, \quad (19)$$

subject to

$$\mathbf{q}_{k+1} = A\mathbf{q}_k + B\mathbf{u}_k, \quad (20)$$

$$\mathbf{q}_k \in \mathcal{Q}, \ \mathbf{u}_k \in \mathcal{U}, \quad (21)$$

$$-\nabla_q f_c(\mathbf{p},\mathbf{q})\,\mathbf{u}_k \Delta t \le \ln\big(f_c(\mathbf{p},\mathbf{q}) + \gamma\big). \quad (22)$$

where $\mathbf{q}_k$ and $\mathbf{u}_k$ denote the state and control input at time step k, H and R are positive definite weighting matrices for tracking error and control effort, $e(\mathbf{q}_k) = \mathbf{q}_k - \mathbf{q}_f$ is the configuration error, $f_c(\mathbf{p},\mathbf{q})$ is the CDF, Δt is the time step,

γ is a safety margin, and $\mathcal{Q}, \mathcal{U}$ are admissible sets of state and control.

For MPPI, the cost function in Eq. 16 more detailed,

$$c_{\text{goal}} = \|\mathbf{q}_H - \mathbf{q}_f\|_2, \tag{23}$$

$$c_{\text{col}} = \sum_{h=1}^{H} \max(0, -d_h), \tag{24}$$

$$c_{\text{joint}} = \sum_{h=1}^{H} \|\max(0, \mathbf{q}_{\min} - \mathbf{q}_h) + \max(0, \mathbf{q}_h - \mathbf{q}_{\max})\|_2^2, \tag{25}$$

$$c_{\text{stay}} = \frac{1}{\|\mathbf{q}_H - \mathbf{q}_0\|_2 + \varepsilon}. \tag{26}$$

Here, $\mathbf{q}_h$ denotes the configuration of a sampled trajectory at prediction step h within the horizon H. The term d_h represents the signed distance evaluated by the CDF function at step h, where negative values indicate collision with either external obstacles.

A. Hyperparameters

The hyperparameters used in our experiments for the 2D two-link robot are summarized in Table III, and those for the 7-Dof Franka robot are listed in Table IV. The MPPI hyperparameters follow [15], while the QP settings are taken from the official implementation provided in the GitHub repository of [7]. For the scenario illustrated in Fig. 5, the start and goal configurations are $q_0 = [-1.57, 0.40, 0.00, -1.2708, 0.00, 1.8675, 0.00]^\top$ and $q_f = [1.57, 0.40, 0.00, -1.2708, 0.00, 1.8675, 0.00]^\top$, respectively.

TABLE III

HYPERPARAMETERS USED IN THE EXPERIMENTS FOR 2D TWO-LINK ROBOT

Method	Parameter	Value
QP	H	diag$(100, 35)$
	R	diag$(0.01, 0.01)$
	$\mathcal{Q}$	$[-\pi, \pi]$
	$\mathcal{U}$	$[-3, 3]$
	γ	0.6
MPPI	α_g	10
	α_c	100
	α_j	100
	α_s	10
	Number of samples N	200
	Horizon H	50
	β	2.0
	γ	1.0
	$\alpha_{\boldsymbol{\mu}}$	0.5
	$\alpha_{\boldsymbol{\Sigma}}$	0.3
Ours (Algorithm 1)	α_1	20
	α_2	10
	d_{act}	0.5
	N	200
	β	1.0
	$\alpha_{\boldsymbol{\mu}}$	0.5
	$\alpha_{\boldsymbol{\Sigma}}$	0.5

TABLE IV

HYPERPARAMETERS USED IN THE EXPERIMENTS FOR 7-DOF FRANKA ROBOT

Method	Parameter	Value
QP	H	diag$(150, 190, 80, 70,$
		$70, 90, 100)$
	R	$0.01 \cdot I_7$
	$\mathcal{Q}$	Standard joint limits of the Franka Panda robot
	$\mathcal{U}$	$[-2.7, 2.7]$
	γ	0.6
Ours (Algorithm 1)	α_1	20
	α_2	10
	d_{act}	1.0
	N	200
	β	1.0
	$\alpha_{\boldsymbol{\mu}}$	0.5
	$\alpha_{\boldsymbol{\Sigma}}$	0.5

REFERENCES

[1] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[2] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 489–494.

[3] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[4] Y. Li, Y. Zhang, A. Razmjoo, and S. Calinon, "Representing robot geometry as distance fields: Applications to whole-body manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 15 351–15 357.

[5] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2022.

[6] ——, "Reactive collision-free motion generation in joint space via dynamical systems and sampling-based mpc," *The International Journal of Robotics Research*, vol. 43, no. 13, pp. 2049–2069, 2024.

[7] Y. Li, X. Chi, A. Razmjoo, and S. Calinon, "Configuration space distance fields for manipulation planning," in *Robotics: Science and Systems (RSS), 2024*, 2024.

[8] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[9] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[10] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1168–1175.

[11] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[12] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 1714–1721.

[13] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 750–759.

[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[15] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos *et al.*, "curobo: Parallelized collision-free minimum-jerk robot motion generation," *arXiv preprint arXiv:2310.17274*, 2023.