arXiv:2412.17462v2 [cs.RO] 15 Jan 2026

Sampling-Based Constrained Motion Planning with Products of Experts

Journal Title
XX(X):1–23
©The Author(s) 2026
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Amirreza Razmjoo[1,2], Teng Xue[1,2], Suhan Shetty[1,2], and Sylvain Calinon[1,2]

Abstract

We present a novel approach to enhance the performance of sampling-based Model Predictive Control (MPC) in constrained optimization by leveraging products of experts. Our methodology divides the main problem into two components: one focused on optimality and the other on feasibility. By combining the solutions from each component, represented as distributions, we apply products of experts to implement a *project-then-sample* strategy. In this strategy, the optimality distribution is projected into the feasible area, allowing for more efficient sampling. This approach contrasts with the traditional *sample-then-project* and naïve *sample-then-reject* method, leading to more diverse exploration and reducing the accumulation of samples on the boundaries. We demonstrate an effective implementation of this principle using a tensor train-based distribution model, which is characterized by its non-parametric nature, ease of combination with other distributions at the task level, and straightforward sampling technique. We adapt existing tensor train models to suit this purpose and validate the efficacy of our approach through experiments in various tasks, including obstacle avoidance, non-prehensile manipulation, and tasks involving staying in a restricted volume. Our experimental results demonstrate that the proposed method consistently outperforms known baselines, providing strong empirical support for its effectiveness. Sample codes for this project are available at https://github.com/idiap/smpc_poe

Keywords

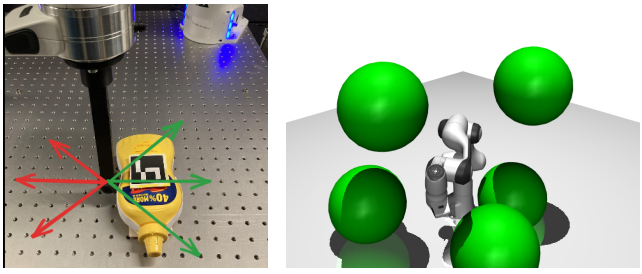
Sampling-based Motion Planning, Constrained Motion Planning, Products of Experts, Tensor-Train Decomposition

1 Introduction

For a robot to navigate and interact effectively within its environment, it must plan its motion while taking surrounding conditions into account. In most cases, the robot must satisfy various constraints imposed by physics, the environment, or its own mechanical limitations. Adhering to these constraints is often what makes motion planning and control a challenging problem in robotics.

In recent years, optimization-based methods, such as optimal control (OC) (Todorov and Jordan 2002) and reinforcement learning (RL) (Sutton and Barto 2018), have led to significant advancements in robotics. These methods are generally categorized into gradient-based and sampling-based approaches. Sampling-based methods, which are gradient-free, are particularly useful in scenarios such as contact-rich manipulation (Pezzato et al. 2023) and obstacle avoidance (Bhardwaj et al. 2022), where the system gradient is often non-smooth due to factors such as sudden environmental contact.

These methods have been widely applied in trajectory optimization (Orthey et al. 2023), including well-known techniques like Rapidly-Exploring Random Trees (RRT) (Kuffner and LaValle 2000). Their gradient-free nature makes them suitable for applications where computing gradients is difficult or infeasible. However, as these methods are only probabilistically complete, they may require extensive sampling. To address this, a common strategy is to reduce the planning horizon and focus on local decision-making. In this context, sampling-based model predictive (SMPC) approaches such as (Williams et al. 2017b; Bhardwaj et al. 2022) have been developed to improve responsiveness and adaptability. However, a primary challenge is the lack of efficient sampling mechanisms to capture

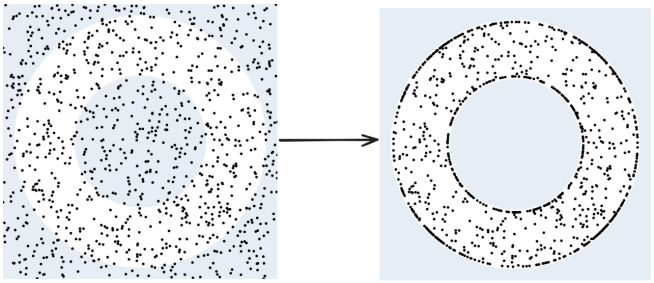


(a) Planar Pushing (b) Whole-body Obstacle Avoidance

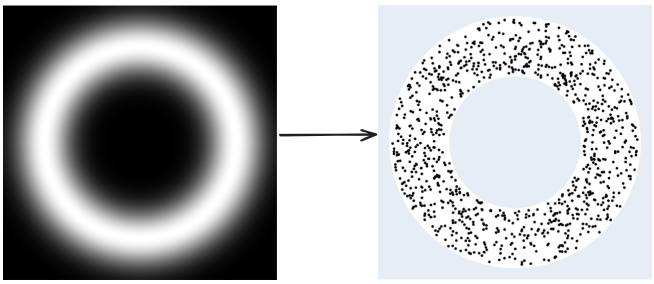
Figure 1. In sampling-based Model Predictive Control (SMPC), many samples can be discarded due to inefficiencies or infeasibility. In (a), only the green samples effectively influence the movement of the object, while the actions indicated by the red arrows have no impact on the environment. (b) illustrates the importance of selecting samples that do not collide with obstacles.

[1]Idiap Research Institute, Martigny, Switzerland
[2]École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Corresponding author:
Amirreza Razmjoo,
Centre du Parc, Rue Marconi 19, CH-1920 Martigny, Switzerland
Email: amirreza.razmjoofard@epfl.ch



(a) Sample-then-project



(b) Project-then-sample

Figure 2. An illustrative example highlighting the difference between two sampling strategies from a uniform distribution that meets the task constraints outlined in Fig. 3: (a) *sample-then-project* and (b) *project-then-sample*. In (a), many samples cluster on the boundaries, leaving significant areas within the feasible set unexplored, whereas (b) promotes a more thorough exploration. Note that the difference between the two approaches would become more pronounced if the feasible space were narrower.

essential information, particularly in cases involving constraints, sparse cost or reward functions. For instance, Fig. 1-a shows a robot trying to push a mustard bottle toward a target, but certain samples can lead it away from the object. Similarly, in Fig. 1-b, many samples can collide with the obstacles, resulting in less informative or rejected samples. In this article, we address the challenge of efficient sampling in the presence of constraints by proposing a novel approach to enhance sampling efficiency.

A basic approach to incorporating constraints into sampling-based methods is to generate samples from a distribution naively without considering constraints and subsequently reject those that violate the constraints. This *sample-then-reject* strategy can be highly inefficient in tightly constrained problems, leading to a large number of discarded samples. This inefficiency has motivated the development of numerous alternative approaches in the literature that improve the sampling process to reduce the number of rejected samples. These methods can be categorized into three main approaches. The first category includes projection-based methods (Mirabel et al. 2016; Jinkyu Kim and Park 2016), which sample first and then project into the feasible region. However, projection is not always straightforward: some methods (Pardo et al. 2015; Suh et al. 2011) require iterative computation of system gradients to perform the projection, which can be problematic when gradients are difficult to obtain—such as in non-prehensile manipulation. As shown in Fig. 2, even with an efficient projection mechanism (often difficult to implement), samples can accumulate near the boundary of the feasible region, leading the system to potentially converge toward that boundary. The second approach involves penalizing constraint violations (Bhardwaj et al. 2022), serving as a data-driven, iterative alternative to projection. However, this method does not guarantee feasible solutions before convergence and may require multiple iterations to achieve feasibility. The third approach, similar to Power and Berenson (2023) and Sacks and Boots (2023), learns a distribution that promotes sampling of feasible and optimal actions or states. While this approach can be effective, it is often complex to implement and computationally expensive (similarly to reinforcement learning methods).

One idea to solve these complexities can be to use several experts. Leveraging multiple expert models to simplify complex tasks is a widely adopted strategy in the machine-learning community. This is commonly implemented either through *mixture of experts* (MoE) (Jordan and Jacobs 1993), which combines outputs in a manner analogous to an *OR* operation, or *products of experts* (PoE) (Hinton 1999), which integrates outputs similarly to an *AND* operation. In our framework, since the objective is to identify solutions that are both optimal *AND* feasible, we adopt the *products of experts* paradigm. Beyond their inherent conceptual differences, PoE offers distinct advantages that make it particularly well-suited to our framework. Specifically, PoE enables each expert to focus on their own decision variables and objectives independently, simplifying and streamlining the learning process for each expert. For instance, in the example presented in this article, the expert tasked with optimizing for action optimality can concentrate solely on action variables (as in methods like MPPI), while another expert can focus on both action and state variables, or even exclusively on state variables, prioritizing feasibility without concern for optimization. By combining their outputs, PoE identifies the intersection of their respective distributions, yielding solutions that satisfy all variables and objectives simultaneously. In contrast, MoE typically combines solutions as a weighted average, potentially failing to account for such considerations.

Similarly, in other works on SMPC, such as (Pan et al. 2024), researchers have employed multiple experts to define the overall distribution. In these approaches, each expert specifies a distribution over the full trajectory, and the experts are combined first, either through a summation of cost terms or a product of distributions, before jointly solving the resulting problem. In contrast, our method decomposes the problem differently. The feasibility expert is learned independently and then combined with the optimality distribution. By taking the product of these two distributions, we reduce the probability of the optimality distribution in the infeasible area, thereby improving the efficiency of the sampling process. The optimality distribution can be modeled as a Gaussian distribution using methods such as Model Predictive Path Integral (MPPI), so this part of the problem is computationally efficient and does not require any learning. By introducing a feasibility expert and integrating it with optimality considerations, we obtain the *project-then-sample* strategy, which efficiently focuses the search

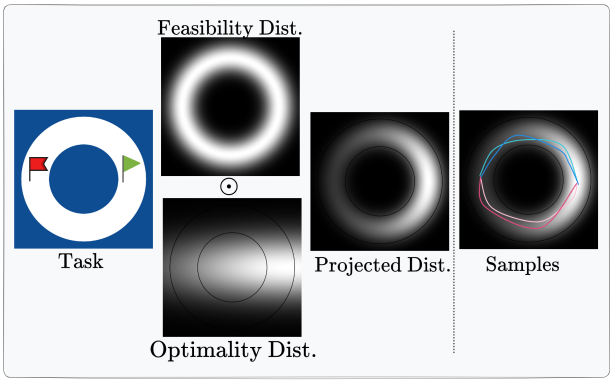


Figure 3. An illustrative example of using products of experts. The agent must navigate from the red flag to the green flag while remaining in the white area and avoiding the blue one. The problem is divided into two parts: one models the feasibility distribution, and the other models the optimality distribution. These are combined to project the optimality distribution into the feasible area (Project first), from which trajectories are then sampled (Then sample).

on the feasible region of the optimization problem, thereby improving overall efficiency.

Fig. 3 illustrates our proposed approach, which splits the main problem into two sub-problems: the **optimality problem**, focusing on optimization, and the **feasibility problem**, focusing on constraints. We compute distributions for both subproblems and then combine them with the products of experts approach to create a distribution that is both feasible and optimal. This framework effectively projects the optimality distribution into the feasible region. A key remaining challenge is how to learn these typically complex, multimodal distributions and to sample from them efficiently.

There are three main paradigms for combining experts at the solution level in the literature: Gaussian experts (Calinon 2016; Hinton 1999), energy-based models (Gkanatsios et al. 2023), and diffusion-based models (Yang et al. 2023; Wang et al. 2024). Gaussian experts offer closed-form inference but are inherently restrictive in the kinds of multimodal distributions they can represent. Energy-based models are more flexible but impose heavy sampling burdens (e.g., Langevin dynamics or MCMC). Diffusion-based approaches significantly reduce sampling complexity, yet typically require large demonstration datasets. Critically, these models often fail to consider feasibility when they are performing sampling during the denoising process, leading to the rejection of many samples at each step. In this paper, we show that by integrating a dedicated feasibility expert, we can improve its reliability before sampling and boost per-step sampling efficiency throughout the diffusion process.

In this article, we demonstrate how different experts can be effectively integrated using tensor train (TT) decomposition. Our previous work demonstrated the potential of this method for learning and sampling from desired distributions (Shetty et al. 2024a; Xue et al. 2024a). We extend these findings by applying TT in the context of products of experts. We compare it to a baseline relying on normalizing flows (NFs) (Dinh et al. 2016), inspired by (Power and Berenson 2023), to show the advantages of TT over NFs, as well as to discuss the differences between a *project-then-sample* strategy compared to a *sample-then-project* or pure *sample-then-reject* strategies.

In summary, our contributions are threefold:

- Enhancing the sampling efficiency of SMPC methods by introducing the feasibility expert and combining it with the optimality distribution using products of experts (i.e., *project-then-sample* strategy). Particularly, the feasibility distribution is learned separately and then combined with the optimality one before sampling. This enables more effective exploration of the feasible set and extends applicability to problems where gradient calculation is challenging.
- Introducing an effective approach to leverage products of experts through TT density estimation, a non-parametric distribution capable of learning multimodal distributions. This approach can also leverage existing straightforward sampling algorithms, enabling efficient exploration of complex distribution spaces.
- Conducting a series of experiments in both simulation and real-world settings to validate the effectiveness of our proposed method compared to baselines.

The remainder of this paper comprises a literature review on various aspects related to our work in Section 2 and a concise yet essential background on the employed methods in Section 3. Our proposed method is detailed in Section 4, and the results are presented in Section 5. Finally, we discuss different aspects of the method in Section 6 and conclude the paper in Section 7.

2 Related work

Gradient-Based vs Sampling-based MPC: Analytical solutions are often impractical for optimization problems in robotics, especially with non-convex, non-linear cost functions and constraints arising from kinematic chains and dynamics. Gradient-based methods like iLQR (Todorov and Jordan 2002; Diehl et al. 2005) are widely used, offering scalability and local solution updates. However, they face issues such as local minima, and undefined gradients in tasks like contact-rich manipulation (Pang et al. 2023; Dafle et al. 2014). These limitations are critical in Model Predictive Control (MPC), where fast solutions are essential.

To address these challenges, sampling-based methods have gained attention. They bypass the need for gradients, can exploit GPU-based simulators for parallel sampling (Liang et al. 2018), and may reduce sensitivity to local minima by approximating smoothed gradients (Pang et al. 2023). This makes them promising alternatives for complex robotic tasks involving discrete decisions (Chavan-Dafle et al. 2020).

Sampling-based MPC (SMPC): Methods such as Rapidly-Exploring Random Tree (RRT) (Kuffner and LaValle 2000) and Probabilistic Roadmap Planner (PRM) (Kavraki et al. 1996) for trajectory optimization, as well as approaches such as Cross-Entropy Motion Planning (CEM) (Kobilarov 2012) and Model Predictive Path Integral (MPPI) (Williams et al. 2017b) have shown the effectiveness of sampling strategies in addressing MPC problems. Recent findings by DeepMind (Howell et al. 2022) suggest that even

for a challenging task such as quadruped walking, selecting the best sample suffices to achieve the objective. However, these advantages are limited to tasks without substantial constraints.

Constrained Sampling-based Motion Planning: Given that sampling-based methods offer probabilistic completeness, they may require extensive data, posing challenges to finding constraint-free solutions for highly constrained problems. Projection methods are widely used in sampling-based methods, particularly within RRT, to ensure constraints are met and improve sampling efficiency (Kingston et al. 2018). Prior research shows that this approach enables probabilistic completeness (Berenson 2011), ensuring that the feasible set will be fully explored over time. However, finding an analytical solution for projection remains challenging under most constraints. In specific cases, projection operators can be locally defined using the tangent space and a nullspace projector (Pardo et al. 2015) or by direct sampling within the tangent space (Suh et al. 2011). Nevertheless, this approach has limitations, as it only captures local constraint behavior and fails with non-differentiable constraints (e.g., contact-rich manipulation). To address this, sampling-based methods often use gradient-based techniques to approximate the projection (Stilman 2010; Yakey et al. 2001). Some works, such as (Stilman 2010), frame projection as a secondary optimization problem. An alternative is to relax constraints by introducing penalty terms within the main objective function (Bhardwaj et al. 2022). However, both approaches face limitations, including increased computational demands and the risk of converging to infeasible regions.

This article contends that the above approaches may not be the most efficient, even with an analytical projection solution. Standard projection-based sampling may guide exploration toward regions with limited feasibility (e.g., boundaries), yielding biased solutions and insufficiently exploring the feasible set. A more effective approach could be to directly model the distribution of the feasible and optimal set and sample exclusively from this distribution, concentrating exploration solely within feasible regions. Recent advances in generative models have facilitated this, allowing feasibility-aware sampling strategies (Sacks and Boots 2023; Power and Berenson 2023). However, these methods, akin to RL, often require learning highly complex distributions, which can be data-intensive and computationally costly.

Our approach seeks to combine the advantages of both projection and generative methods. We propose a two-stage process: first, we train generative models to approximate the distribution of the feasible region. Rather than projecting individual samples, we then project an estimated optimality distribution into this feasible region using a product of experts, enhancing both the efficiency and focus of sampling within feasible areas. Our approach has two main benefits. Compared to learning a distribution that captures both feasibility and optimality, we focus on the simpler task of feasibility alone and allowing the learned model to be used across tasks with similar feasibility sets. Furthermore, compared to projection-based methods, our approach *project-then-sample* promotes better exploration within the feasible set.

Product of Experts: In SMPC, it is common practice to define the overall cost function or solution distribution by combining multiple terms, typically through summation or a product of distributions (Pan et al. 2024). Each term is designed to address a specific aspect of the problem, such as optimality or constraint satisfaction, and the final solution is obtained by solving the joint problem defined by their combination.

In most existing approaches, this combination occurs at the definition level, meaning that the full problem is formed first and then solved using techniques such as gradient descent, Langevin dynamics, or other sampling-based methods. While this is effective, it tends to reduce flexibility, as the individual structures of the constituent terms are lost once they are merged.

In contrast, Hinton's original formulation of the *product of experts* (Hinton 1999) suggests an alternative: solving each expert individually and then combining their solutions. By *solution*, we refer to a sampleable distribution, such as a Gaussian, that does not require further optimization or inference steps. This approach preserves the modularity of each expert, enables the use of specialized solvers, and simplifies sampling from the final distribution without re-solving the full joint problem. In this regard, each expert represents a relatively simple distribution focusing on specific parameters or tasks. These experts are subsequently combined to yield a more refined and sharper distribution. This methodology has found applications in robotics, including the fusion of sensory information (Pradalier et al. 2003), the fusion of robot motion learned in multiple coordinate systems (Calinon 2016), and across task spaces (Pignat et al. 2022; Mühlbauer et al. 2024). Our work extends these prior approaches by incorporating the product of experts framework within an MPC paradigm.

There are three main approaches in the literature for combining multiple experts at the solution level: (1) **Gaussian experts** (Calinon 2016; Hinton 1999), (2) **Energy-based models** (Gkanatsios et al. 2023), and (3) **Diffusion-based models** (Yang et al. 2023; Wang et al. 2024). Each of these methods offers different advantages and trade-offs, which we discuss further to motivate our use of the TT representation.

Compared to these methods, Gaussian distributions are significantly more limited than TT and other baselines, primarily due to the parametric nature of Gaussian models. Energy-based models (LeCun et al. 2006), which can be viewed as a more general form of Gaussian distributions, indeed hold greater potential. However, according to our experience, sampling from these models typically relies on Langevin dynamics or methods such as MCMC, which can become a significant computational bottleneck. In contrast, TT does not suffer from this issue and allows for efficient and straightforward sampling.

Diffusion models (Ho et al. 2020) (or similar variations such as Flow-Matching (Lipman et al. 2023)) address the sampling challenges present in energy-based models to a significant extent. These models can be broadly categorized into two types: data-based (Wang et al. 2024) and model-based (Pan et al. 2024). Data-based methods require the optimality distribution to be explicitly learned in diffusion

form, which typically demands a substantial number of high-quality demonstrations—a requirement not assumed in our work.

Moreover, recent works such as (Xue et al. 2025; Pan et al. 2024) have proposed innovative strategies to represent diffusion models in a data-free manner. Their approach, which bears resemblance to annealed MPPI methods, combines multiple experts at the function level. In their pipeline, different candidate solutions are sampled first, followed by a rejection step to discard infeasible samples at each denoising step. By contrast, our approach merges experts at the solution level. This allows us to reshape the sampling distribution before sampling, which is a more proactive method than relying solely on post-hoc sample rejection. Concretely, we inject the feasibility expert's density into every denoising (sampling) step of the diffusion process, producing candidates that are inherently more likely to satisfy constraints.

3 Background

3.1 Model Predictive Path Integral

The primary aim of this methodology is to tackle an optimal control problem associated with a dynamic system described by $\boldsymbol{x}_{t+1} = f_{\text{dyn}}(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\omega}_t)$ within a receding horizon H. Here, $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ and $\boldsymbol{u}_t \in \mathbb{R}^{d_u}$ denote the system state and action at time t, respectively, while $\boldsymbol{\omega}_t$ represents a stochastic noise element in the system. This objective is achieved by minimizing a predetermined cost function as

$$c(\hat{\boldsymbol{X}}_t, \hat{\boldsymbol{U}}_t) = \sum_{h=0}^{H-1} c_h(\hat{\boldsymbol{x}}_{t+h}, \hat{\boldsymbol{u}}_{t+h}) + c_H(\hat{\boldsymbol{x}}_{t+H}). \tag{1}$$

Here, $\hat{\boldsymbol{X}}_t = [\hat{\boldsymbol{x}}_t^\top, \hat{\boldsymbol{x}}_{t+1}^\top, \dots, \hat{\boldsymbol{x}}_{t+H}^\top]^\top$ represents the state trajectory, and $\hat{\boldsymbol{U}}_t = [\hat{\boldsymbol{u}}_t^\top, \hat{\boldsymbol{u}}_{t+1}^\top, \dots, \hat{\boldsymbol{u}}_{t+H-1}^\top]^\top$ denotes the sequence of action commands. The hat symbol $\hat{(\cdot)}$ indicates that these actions are part of the planned, but not yet applied, control sequence. Additionally, c_h and c_H denote stage and terminal cost values, respectively. Here, the constraints are formulated inside the cost function. The objective is to determine a set of optimal action commands $\boldsymbol{U}_t^*$ that minimizes this cost function while taking into account the dynamics of the system, specifically

$$\boldsymbol{U}_t^* = \underset{\hat{\boldsymbol{U}}_t}{\text{argmin}}\ c(\hat{\boldsymbol{X}}_t, \hat{\boldsymbol{U}}_t). \tag{2}$$

Williams et al. (Williams et al. 2017a,b) demonstrated that the solution for this optimal control problem can be effectively approximated using a forward approach, diverging from the conventional dynamic programming method. This discovery opened avenues for leveraging Monte Carlo estimation in approximating optimal actions through sampling-based methods. Furthermore, Wagener et al. (2019) showed that if the system motion can be modeled as a Gaussian distribution such as

$$\pi_\theta(\hat{\boldsymbol{U}}_t) = \prod_{h=0}^{H-1} \pi_{\theta_h}(\hat{\boldsymbol{u}}_{t+h}) = \prod_{h=0}^{H-1} \mathcal{N}(\hat{\boldsymbol{u}}_{t+h} | \boldsymbol{\mu}_{t+h}, \boldsymbol{\Sigma}_{t+h}), \tag{3}$$

then its parameters can be approximated as

$$\boldsymbol{\mu}_t = (1-\gamma_t)\hat{\boldsymbol{\mu}}_t + \gamma_t \sum_{i=1}^{N} w_i \hat{\boldsymbol{U}}_t^{(i)}, \tag{4}$$

where N denotes the number of samples, $\hat{\boldsymbol{\mu}}_t$ represents the preceding mean value, and γ_t is the step size. The term w_i is a weighting factor expressed as

$$w_i = \frac{e^{-\frac{1}{\beta} c(\hat{\boldsymbol{X}}_t^{(i)}, \hat{\boldsymbol{U}}_t^{(i)})}}{\sum_{j=0}^{N} e^{-\frac{1}{\beta} c(\hat{\boldsymbol{X}}_t^{(j)}, \hat{\boldsymbol{U}}_t^{(j)})}}, \tag{5}$$

where β is a temperature parameter. Following the calculation of the parameters, the initial action $\boldsymbol{u}_t^*$ can be sampled, applied to the system, and iteratively repeated until the task is successfully completed.

3.2 Tensor Train Decomposition

A d-th order tensor, denoted as $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, serves as an extension of vectors (1-D) and matrices (2-D) to higher dimensions (d-D). Its configuration is defined by a tuple of integers $\boldsymbol{n} = (n_1, \dots, n_d)$. Each element in the tensor, denoted as $\boldsymbol{\mathcal{P}_i}$, is identified by $\boldsymbol{i} = (i_1, \dots, i_d)$ as a set of d integer variables representing the index in each dimension. The index set of the tensor is expressed as $\mathcal{I} = \{\boldsymbol{i} = (i_1, \dots, i_d) : i_k \in \{1, \dots, n_k\}, k \in \{1, \dots, d\}\}$.

We can express a function, denoted as $p : \Omega_{\boldsymbol{x}} \subset \mathbb{R}^d \to \mathbb{R}$, as a tensor by assuming a rectangular domain for the function, such as $\Omega_{\boldsymbol{x}} = \Omega_{x_1} \times \Omega_{x_2} \times \cdots \times \Omega_{x_d} = \times_{k=1}^{d} \Omega_{x_k}$. This involves discretizing each input interval $\Omega_{x_k} \subset \mathbb{R}$ into n_k elements. The continuous set $\mathcal{X} = \{\boldsymbol{x} = (x^{i_1}, \dots, x^{i_d}) : x^{i_k} \in \Omega_{x_k}, i_k \in 1, \dots, n_k\}$ can be easily mapped to the corresponding index set $\mathcal{I}_{\mathcal{X}} = \{\boldsymbol{i} = (i_1, \dots, i_d)\}$. The $\boldsymbol{i}$-th element in the tensor $\boldsymbol{\mathcal{P}}$, denoted by $\boldsymbol{\mathcal{P}_i}$, corresponds to the function value with continuous variables, such that $\boldsymbol{\mathcal{P}_i} = p(\boldsymbol{x})$. We overload the terminology and define $\boldsymbol{\mathcal{P}_x} = \boldsymbol{\mathcal{P}_i}$ where $\boldsymbol{i}$ is the corresponding index of $\boldsymbol{x}$. Interpolation techniques between specific nodes of the tensor $\boldsymbol{\mathcal{P}}$ can be used to approximate the value $p(\boldsymbol{x})$ for any $\boldsymbol{x} \in \Omega_{\boldsymbol{x}}$.

To represent a tensor efficiently for a high-dimensional function, we employ a tensor factorization method that enhances storage efficiency by utilizing factors with a reduced number of elements. Among various methodologies, we adopt the Tensor Train (TT) decomposition approach in this study, which utilizes a set of third-order tensors known as *TT-cores* to compactly encode a given tensor. In the TT format, a d-th order tensor $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is expressed using a tuple of d third-order tensors $(\boldsymbol{\mathcal{P}}^1, \dots, \boldsymbol{\mathcal{P}}^d)$. The dimensions of the TT-cores are denoted as $\boldsymbol{\mathcal{P}}^k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ with $r_0 = r_d = 1$. As illustrated in Fig. 4, the computation of the $\boldsymbol{i}$-th element of the tensor in this format involves a straightforward multiplication of matrix slices obtained from the TT-cores as

$$\boldsymbol{\mathcal{P}_i} = \boldsymbol{\mathcal{P}}^1_{:,i_1,:} \boldsymbol{\mathcal{P}}^2_{:,i_2,:} \cdots \boldsymbol{\mathcal{P}}^d_{:,i_d,:}, \tag{6}$$

where $\boldsymbol{\mathcal{P}}^k_{:,i_k,:} \in \mathbb{R}^{r_{k-1} \times r_k}$ denotes the i_k-th matrix slice of the third-order tensor $\boldsymbol{\mathcal{P}}^k$. The multiplication of these TT-cores yields a scalar variable. The determination of these cores, which have been demonstrated to exist for any

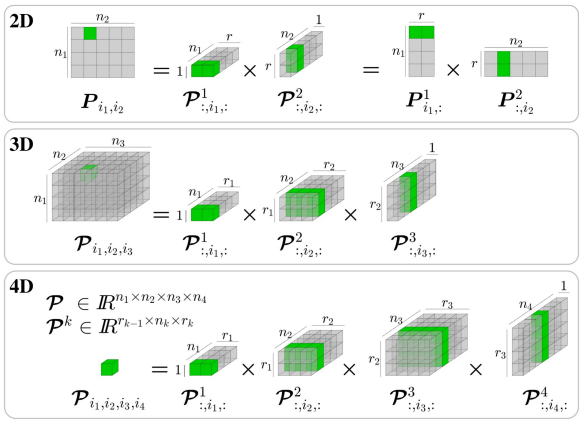


Figure 4. TT decomposition extends matrix decomposition techniques to higher-dimensional arrays. In the TT format, accessing an element of a tensor involves multiplying the chosen slices (represented by green-colored matrices) of the core tensors (factors). The illustration provides examples for a 2nd order, 3rd order, and a 4th order tensor. This picture is adopted from Shetty et al. (2024a) with permission from the authors.

given tensor consistently (Oseledets 2011), can be achieved through various methods, such as TT-SVD (Oseledets 2011) and TT-cross (Savostyanov and Oseledets 2011).

The *TT-rank* of the tensor in the TT representation is defined by the tuple $\boldsymbol{r} = (r_1, r_2, \ldots, r_{d-1})$. We refer to $r = \max(r_1, \ldots, r_{d-1})$ as the maximal rank. Similar to the matrix analogy, a smaller r leads to a less precise approximation, while a larger r necessitates greater storage capacity.

Utilizing this methodology, the ultimate outcome is the approximation of the continuous function p, expressed as

$$p(x_1, \ldots, x_d) \approx \boldsymbol{P}^1(x_1) \cdots \boldsymbol{P}^d(x_d). \tag{7}$$

Here, $\boldsymbol{P}^k(x_k)$ with $k \in \{1, \ldots, d\}$ is derived by interpolating within each of the cores. Employing advanced interpolation techniques enables a coarser discretization, thereby reducing storage requirements during the learning phase.

3.3 Tensor-Train Distribution

The TT approximation technique has been previously applied to optimization problems (Batsheva et al. 2023; Shetty et al. 2024a). In this approach, the problem of minimizing a cost function is first transformed into the task of maximizing an unnormalized probability density function $p(\cdot)$, represented as a tensor $\boldsymbol{\mathcal{P}}$ in TT format. Subsequently, a corresponding probability distribution, referred to as the TT distribution, is constructed as

$$\Pr(\boldsymbol{x}) = \frac{|\boldsymbol{\mathcal{P}}_{\boldsymbol{x}}|}{Z}, \quad \boldsymbol{x} \in \Omega_{\boldsymbol{x}}, \tag{8}$$

where Z denotes the corresponding normalization constant. Leveraging the separable structure of the TT model allows for the efficient generation of exact samples from the TT distribution, eliminating the need to compute the normalization factor Z. Additional details about the sampling process can be found in the Appendix. Moreover, as illustrated in (Shetty et al. 2024a), direct calculations of marginal and conditional distributions can be derived from the TT-cores.

For simplicity in notation, we employ $\boldsymbol{\mathcal{P}}(\boldsymbol{x})$ to denote the tensor element corresponding to $\boldsymbol{x}$ and $\boldsymbol{\mathcal{P}}(\boldsymbol{y_2}|\boldsymbol{y_1})$ to express the conditional probability of $\boldsymbol{y}_2 = [x_{j+1}, \ldots, x_d]$ given $\boldsymbol{y}_1 = [x_1, x_2, \ldots, x_j]$. It is important to emphasize that the latter is directly computed from $\boldsymbol{\mathcal{P}}$.

4 Methodology

4.1 Problem Definition

In a constrained SMPC problem, the aim is to find an optimal sequence of actions $\hat{\boldsymbol{U}}_t = [\hat{\boldsymbol{u}}_t^\top, \hat{\boldsymbol{u}}_{t+1}^\top, \ldots, \hat{\boldsymbol{u}}_{t+H-1}^\top]^\top$ at time step t over a horizon H. This sequence should optimize a cost function as

$$\begin{aligned} \boldsymbol{U}_t^* = \underset{\hat{\boldsymbol{U}}_t}{\operatorname{argmin}} \sum_{h=0}^{H-1} c_h(\hat{\boldsymbol{x}}_{t+h}, \hat{\boldsymbol{u}}_{t+h}, \boldsymbol{\theta}) + c_H(\hat{\boldsymbol{x}}_{t+H}, \boldsymbol{\theta}), \\ \text{s.t. } \boldsymbol{f}(\hat{\boldsymbol{x}}_{t+h+1}, \hat{\boldsymbol{u}}_{t+h}, \boldsymbol{\theta}) \leq \boldsymbol{0}, \quad \forall h \in \{0, 1, \ldots, H-1\}. \end{aligned} \tag{9}$$

Here, $\boldsymbol{x}_t$ denotes the system state at time step t, $\boldsymbol{f}(\cdot) \in \mathbb{R}^w$ represents the vector of w constraints, and $c_h(\cdot)$ and $c_H(\cdot)$ denote the stage and terminal costs, respectively. $\boldsymbol{\theta}$ denotes task-dependent terms such as obstacle locations, target positions, etc. For clarity, we omit these terms in the remainder of the paper; however, they are still incorporated in our experiments, as detailed in the Appendix. It is important to note that the constraint set may incorporate prior knowledge that we expect the system to follow. For instance, in a pushing task, one constraint could ensure that actions taken at each step influence the object and prevent the robot from moving away from it beyond a certain threshold, as illustrated in Fig. 1-a.

4.2 Products of Experts

Similarly to other SMPC methods, we assume that the solution to this problem can be modeled as a distribution $p(\hat{\boldsymbol{U}}_t|\boldsymbol{x}_t)$. Furthermore, we assume that this distribution can be expressed as a product of two other distributions

$$p(\hat{\boldsymbol{U}}_t|\boldsymbol{x}_t) = p^o(\hat{\boldsymbol{U}}_t|\boldsymbol{x}_t) p^f(\hat{\boldsymbol{U}}_t|\boldsymbol{x}_t). \tag{10}$$

The first distribution, p^o, captures the dynamic aspect of the task and ensures optimality (i.e., the optimality distribution, Expert 1), while the second distribution, p^f, represents the feasible area (i.e., the feasibility distribution, Expert 2).

Expert 1: We can solve (9) using the MPPI method by incorporating the constraint within the cost function. This approach enables the system to gradually converge to the feasible area, although it may lead to time-consuming iterations and the rejection of numerous samples due to high-cost values. This expert addresses the following problem:

$$\begin{aligned} \boldsymbol{U}_t^* = \underset{\hat{\boldsymbol{U}}_t}{\operatorname{argmin}} \sum_{h=0}^{H-1} c_h(\hat{\boldsymbol{x}}_{t+h}, \hat{\boldsymbol{u}}_{t+h}) + \\ c_f(\boldsymbol{x}_{t+h+1}, \boldsymbol{u}_{t+h}) + c_H(\hat{\boldsymbol{x}}_{t+H}), \end{aligned} \tag{11}$$

where $c_f(\cdot)$ denotes the cost term associated with constraint violations in $\boldsymbol{f}$. Although Expert 1 is primarily responsible

for modeling the optimality distribution, the inclusion of the additional cost term $c_f(\cdot)$ offers several advantages. First, by assigning significantly higher values to this term relative to others, infeasible samples are effectively assigned zero weight via (5), thereby removing them from consideration. Secondly, it allows us to express the objective in a more general form, consistent with how similar formulations are structured in related literature, thereby making Expert 1 directly comparable to prior approaches. Third, this added term provides flexibility to account for certain constraints that may emerge during task execution but are not explicitly captured by Expert 2 (described later). Finally, from a practical standpoint, incorporating $c_f(\cdot)$ into Expert 1 can enhance efficiency by guiding the optimization using relevant constraint information, even when a separate feasibility expert is also present. In this paper, we follow prior works (Jankowski et al. 2023; Bhardwaj et al. 2022; Power and Berenson 2023) by assigning a large weight to this term in order to more effectively reject infeasible solutions. Bhardwaj et al. (2021) provide a detailed analysis of how increasing this penalty improves feasibility under tight constraints.

The solution to (11) using the MPPI method can be formulated as

$$p^o(\hat{\boldsymbol{U}}_t) = \prod_{h=0}^{H-1} \mathcal{N}(\hat{\boldsymbol{u}}_{t+h} | \boldsymbol{\mu}_{t+h}, \boldsymbol{\Sigma}_{t+h}), \tag{12}$$

which represents a Gaussian distribution at each time step, characterized by the mean $\boldsymbol{\mu}_{t+h}$ and covariance matrix $\boldsymbol{\Sigma}_{t+h}$.

Expert 2: Handling constraints exclusively through a rejection mechanism, for example by assigning high values to the c_f term, is generally inefficient. It wastes computational resources by generating and then discarding infeasible trajectories. Our proposed method addresses this by introducing a *feasibility expert* that proactively biases the sampling distribution toward feasible regions. This improves the quality of samples before they are evaluated and potentially rejected, leading to several key benefits as discussed in this paper.

To modify the sampling distribution, we introduce another distribution $p^f(\cdot)$ that focuses on the feasible set $\mathcal{F}$: $\{(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}} \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \leq 0\}$. We define a function p_{feas} to capture this feasible area as

$$p^f(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \propto p_{\text{feas}}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) = \exp\Big(- \frac{\text{ReLU}\Big(\boldsymbol{f}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)\Big)}{\lambda}\Big), \tag{13}$$

where λ is a temperature parameter. Various methods, including different generative models, can be employed to convert this function into a proper distribution. This article chooses the approach outlined in (Shetty et al. 2024a). This method discretizes the function (13) and transforms it into an unnormalized distribution. The resulting distribution can be represented using tensor cores (as defined in Sec. 3.2), enabling a lower memory footprint and seamless integration with the other experts, along with a straightforward sampling method from the new distribution.

After converting the constraint function (13) into a distribution in TT format, we can express it as

$$p^f(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \rightarrow \boldsymbol{\mathcal{P}}_f(\hat{\boldsymbol{u}}_t, \hat{\boldsymbol{x}}_t) = \prod_{n=0}^{d_x} \boldsymbol{\mathcal{P}}^n_{:,i_{\hat{x}_n},:} \prod_{m=0}^{d_u} \boldsymbol{\mathcal{P}}^{d_x+m}_{:,i_{\hat{u}_m},:}, \tag{14}$$

where $\hat{x}_n$ and $\hat{u}_m$ are respectively n-th and m-th value of the state $\hat{\boldsymbol{x}}_t$ and action $\hat{\boldsymbol{u}}_t$ vectors. As discussed in (Shetty et al. 2024a), it is possible to derive the conditional distribution in TT format $\boldsymbol{\mathcal{P}}_f(\hat{\boldsymbol{u}}_{t+h} | \hat{\boldsymbol{x}}_{t+h})$ from the joint distribution obtained in (14). Consequently, assuming that this distribution is time-independent (though it can be readily extended to a time-dependent version by introducing time as an additional variable into the system), we can define the second expert as

$$p^f(\hat{\boldsymbol{U}}_t | \boldsymbol{x}_t) \propto \prod_{h=0}^{H-1} \boldsymbol{\mathcal{P}}_f(\hat{\boldsymbol{u}}_{t+h} | \hat{\boldsymbol{x}}_{t+h}). \tag{15}$$

4.3 TT-PoE-MPPI

By substituting the values derived in (15) and (12) into (10), we can express the distribution as

$$p(\hat{\boldsymbol{U}}_t | \boldsymbol{x}_t) \propto \prod_{h=0}^{H-1} \mathcal{N}(\hat{\boldsymbol{u}}_{t+h} | \boldsymbol{\mu}_{t+h}, \boldsymbol{\Sigma}_{t+h}) \boldsymbol{\mathcal{P}}_f(\hat{\boldsymbol{u}}_{t+h} | \hat{\boldsymbol{x}}_{t+h}). \tag{16}$$

Next, we focus on a specific time step h, where the sampling can be performed from the distribution

$$p^h(\hat{\boldsymbol{u}}_{t+h} | \boldsymbol{x}_t) \propto \mathcal{N}(\hat{\boldsymbol{u}}_{t+h} | \boldsymbol{\mu}_{t+h}, \boldsymbol{\Sigma}_{t+h}) \boldsymbol{\mathcal{P}}_f(\hat{\boldsymbol{u}}_{t+h} | \hat{\boldsymbol{x}}_{t+h}). \tag{17}$$

We can rewrite the conditional distribution as

$$p^h(\hat{\boldsymbol{u}}_{t+h} | \hat{\boldsymbol{x}}_{t+h}) \propto \mathcal{N}(\hat{\boldsymbol{u}}_{t+h} | \boldsymbol{\mu}_{t+h}, \boldsymbol{\Sigma}_{t+h}) \boldsymbol{\mathcal{P}}_f(\hat{\boldsymbol{u}}_{t+h}, \hat{\boldsymbol{x}}_{t+h}). \tag{18}$$

This product can be computed efficiently by introducing a new tensor, $\boldsymbol{\mathcal{P}}_{\mathcal{N}}(\hat{\boldsymbol{u}}_{t+h})$, which represents the tensor form of the $\mathcal{N}(\cdot)$ function. Consequently, the product can be expressed as

$$p^h(\hat{\boldsymbol{u}}_{t+h} | \hat{\boldsymbol{x}}_{t+h}) \propto \boldsymbol{\mathcal{P}}_{\mathcal{N}}(\hat{\boldsymbol{u}}_{t+h}) \boldsymbol{\mathcal{P}}_f(\hat{\boldsymbol{u}}_{t+h}, \hat{\boldsymbol{x}}_{t+h}). \tag{19}$$

Although this product can be computed for any distribution, if we assume the covariance matrix is diagonal (i.e., independent actions in different dimensions), $\boldsymbol{\Sigma}_{t+h} = \text{diag}(\boldsymbol{\sigma}^2_{t+h})$, the computation becomes more efficient, allowing it to be performed directly at the core level. This assumption is common (Power and Berenson 2023; Sacks and Boots 2023) and is crucial as it enables efficient multiplication of these distributions.

It is important to note that the cost value is calculated via (9), and thus the sample weights in (5) are computed by considering the actions across all times and dimensions collectively. Consequently, their influence on one another is captured in the desired action command. However, the independence assumption simplifies the learning and sampling process. Given this assumption, the elementwise

Algorithm 1 TT-PoE-MPPI

Require: $\boldsymbol{x}_t, \boldsymbol{\mu}_{t,\dots,t+H-1}, \boldsymbol{\sigma}_{t,\dots,t+H-1}$
Return: $\boldsymbol{u}_t^*$
$\hat{\boldsymbol{x}}_t \leftarrow \boldsymbol{x}_t$
for h from 0 to $H-1$ **do**
  Calculate $\boldsymbol{\mathcal{P}}_{\mathcal{N}_m}$ ▷ Eq. (21)
  $\boldsymbol{\mathcal{P}}_h \leftarrow$Product of the distributions ▷ Eq. (20)
  $\hat{\boldsymbol{u}}_{t+h}^0, \dots, \hat{\boldsymbol{u}}_{t+h}^N \leftarrow$Sample N actions
  $\hat{\boldsymbol{x}}_{t+h+1}^0, \dots, \hat{\boldsymbol{x}}_{t+h+1}^N \leftarrow$Forward Dynamics
end for
$c^0, \dots, c^N \leftarrow$Calculate cost values
$\boldsymbol{\mu}_t \leftarrow$ update the mean value ▷ Eq. (4)
$\boldsymbol{u}_t^* \leftarrow \boldsymbol{\mu}_t$

multiplication can be independently performed at each TT-core as

$$p^h(\hat{\boldsymbol{u}}_{t+h}|\hat{\boldsymbol{x}}_{t+h}) \propto \boldsymbol{\mathcal{P}}_h \approx \prod_{n=0}^{d_x} \boldsymbol{\mathcal{P}}^n_{:,i_{\hat{x}_n},:} \prod_{m=0}^{d_u} \left(\boldsymbol{\mathcal{P}}^{d_x+m} \odot \boldsymbol{\mathcal{P}}_{\mathcal{N}_m}\right)_{:,i_{\hat{u}_m},:}, \quad (20)$$

where $\odot$ denotes elementwise multiplication, and $\boldsymbol{\mathcal{P}}_{\mathcal{N}_m}$ has the same dimension as $\boldsymbol{\mathcal{P}}^{d_x+m}$, and its elements can be described by

$$\boldsymbol{\mathcal{P}}_{\mathcal{N}_{m:,i_{\hat{u}_m},:}} = \frac{1}{\sigma^m_{t+h}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\hat{u}_m - \mu^m_{t+h}}{\sigma^m_{t+h}}\right)^2\right), \quad (21)$$

where μ^m_{t+h} and σ^m_{t+h} represent the values of $\boldsymbol{\mu}_{t+h}$ and $\boldsymbol{\sigma}_{t+h}$, respectively, corresponding to the m-th dimension. The schematic process for this step at time $t+h$ is presented in Fig. 5.

Once we determine the desired distribution, we employ the same procedure as in SMPC methods. We sample a set of actions, evaluate their associated cost values through a forward path, update the mean values accordingly, and iterate through this process until the task is completed. The complete procedure is outlined in Algorithm 1.

4.4 Decomposable feasible distribution

The same idea can be applied to facilitate the learning of more complex feasibility distributions from simpler ones. For instance, without loss of generality, consider a scenario where the feasible area can be defined by two constraints, $f_1(\boldsymbol{x}_t, \boldsymbol{u}_t)$ and $f_2(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Instead of learning the feasible set for both constraints simultaneously, one could learn them separately as $\boldsymbol{\mathcal{P}}^{f_1}$ and $\boldsymbol{\mathcal{P}}^{f_2}$. This approach not only simplifies the problem but also increases efficiency. In some parts of the task, we may only need to consider one of the constraints, eliminating the need to relearn both.

The feasible set for the entire problem can be easily computed by combining the two separate (TT) models as

$$\boldsymbol{\mathcal{P}}^f = \boldsymbol{\mathcal{P}}^{f_1} * \boldsymbol{\mathcal{P}}^{f_2}, \quad (22)$$

where the symbol $*$ denotes the product of the two TT models. To calculate this product, we do not need access to the entire tensor, as it can be efficiently computed at the core level. Libraries such as tntorch (Usvyatsov et al. 2022) can facilitate this process.

As a side note, when the feasible regions of two constraints overlap, we can use a relaxed combination via summation rather than multiplication as

$$\boldsymbol{\mathcal{P}}^f = \boldsymbol{\mathcal{P}}^f_1 + \boldsymbol{\mathcal{P}}^f_2, \quad (23)$$

where $+$ denotes arithmetic operations between two TT models. While a product-of-experts formulation is often preferred, the additive combination retains the same high-probability overlap of feasible regions and thus remains effective at guiding the sampler toward the feasible set. Moreover, this relaxed *sum-of-experts* requires fewer ranks and can be more practical in high-dimensional settings.

Fig. 6 demonstrates two examples of leveraging multiple expert models to address complex tasks. On the left, one expert models the joint angles for achieving a specific orientation, while another focuses on positions along a desired vertical line. The resulting combined distribution respects both constraints and is more sparse compared to the individual models, as evident from the corresponding matrices shown in the figure. On the right, one expert focuses on obstacle avoidance, while another models the task of reaching the target. The combined distribution ensures that the target is reached while avoiding obstacles, effectively balancing both objectives. This approach highlights the power of combining simpler expert models to manage complex scenarios. Furthermore, the modularity of this method allows constraints to be selectively applied. For instance, in the right example, the obstacle avoidance model can be prioritized for intermediate trajectory points, while the target-reaching distribution is only necessary for the endpoint.

Moreover, this concept extends beyond simple AND operations. Indeed, we can also consider other logical behaviors such as OR operations. Without delving into further details, we can express the distribution that satisfies at least one of the constraints f_1 or f_2 as

$$\boldsymbol{\mathcal{P}}^f = \boldsymbol{\mathcal{P}}^{f_1} + \boldsymbol{\mathcal{P}}^{f_2} - \boldsymbol{\mathcal{P}}^{f_1} * \boldsymbol{\mathcal{P}}^{f_2}, \quad (24)$$

These operations can further assist in solving more complex problems by focusing on simpler ones, allowing us to efficiently utilize the learned models instead of relearning them for each situation.

5 Results

We applied our methodology to several tasks, verifying the effectiveness of the proposed approach by conducting a comparative analysis with other baselines, namely MPPI, Proj-MPPI (when possible) and NFs-PoE-MPPI. In the MPPI baseline, constraints are integrated directly into the main cost function (i.e., as relaxed constraints), which aligns with the most widely accepted approach in the literature employing this method. In Proj-MPPI, each sample generated by the MPPI method was initially projected into the feasible region before being processed in the MPPI pipeline. The NFs-PoE-MPPI method leverages normalizing flows (NFs) (Dinh et al. 2016) to learn the feasibility distribution, taking inspiration from FlowMPPI (Power and Berenson 2023). To learn this distribution with NFs, we used the same amount of data as for TT, ensuring that all

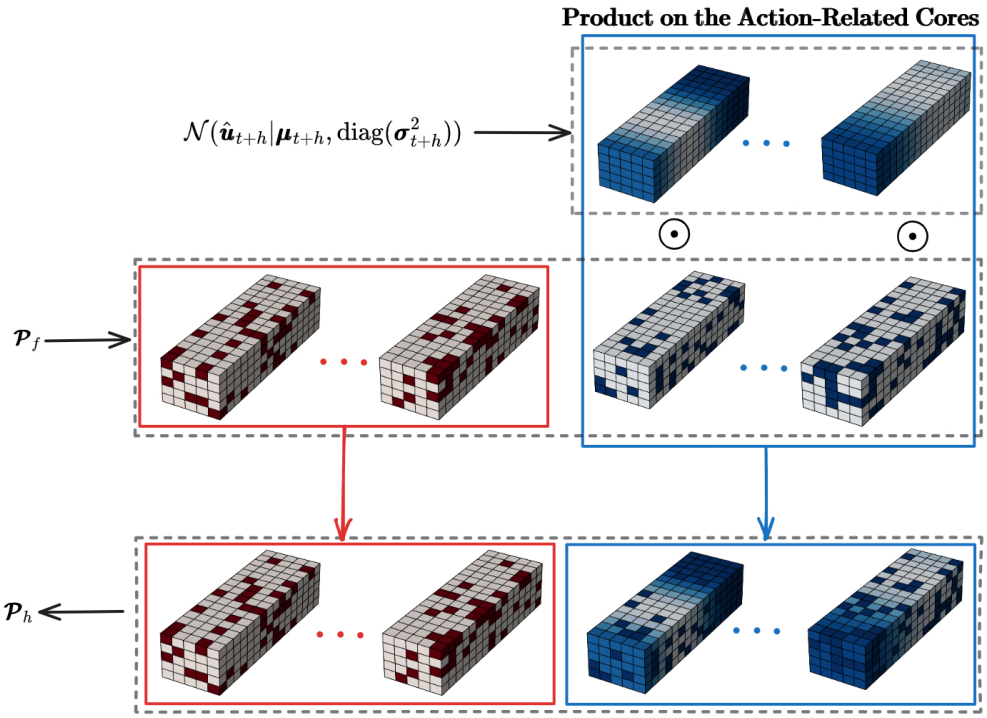


Figure 5. The pipeline employed in this paper integrates two distributions. Red, and blue colors indicate the cores related to the state and action variables, respectively. Initially, the optimality distribution is transformed into TT-cores, followed by elementwise multiplication with the feasibility distribution cores corresponding to the action space. Subsequently, a new distribution is generated using the updated cores. In the illustration, white boxes represent elements equal to 1, while darker boxes denote elements equal to 0. It is important to note that this visualization is simplified for clarity: in actual scenarios, the cores would not strictly be 0 or 1.

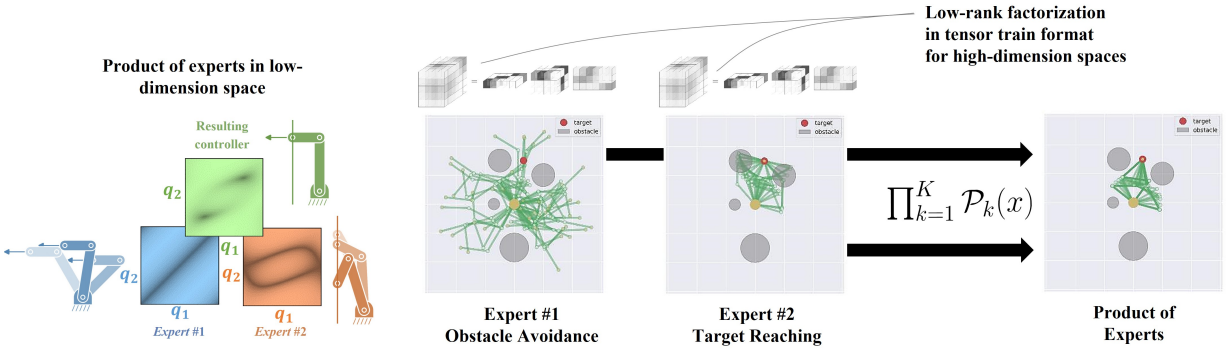


Figure 6. Examples of utilizing multiple experts to extract the feasible set. *Left*: Combining one expert that aligns the end-effector horizontally and another that positions the end-effector on a desired vertical line. The resulting distribution satisfies both constraints. The matrices show the corresponding configuration distribution with the darker values showing the desired configurations. *Right*: Similarly, combining models for obstacle avoidance and target reaching generates configurations (green lines) that meet both goals.

data points were positive. We maintained consistency in the number of data points across methods to enable a fair comparison. Details regarding the hyperparameters used in various learning methods across different tasks are provided in Table 7.

The core problem, particularly the optimality distribution, is identical for all baselines. Specifically, each method employs a high-penalty term with the same value within the cost function to reject infeasible samples. This mechanism is critical for the constrained tasks considered in this paper; without it, even a few infeasible samples that score well on the unconstrained cost could degrade performance. This approach is consistent with prior work (Jankowski et al. 2023; Bhardwaj et al. 2022; Power and Berenson 2023). When comparing baselines, we therefore focus on evaluating the unique components of each method, such as the feasibility expert in PoE-MPPI, rather than the shared rejection mechanism. This separation allows us to independently assess the contribution of these additional components.

The comparison is based on three metrics: the success rate, the total cost, and the number of steps required to reach the goal. To ensure fair comparison despite the varying complexities of different task parameters, we normalize the results of each baseline method to those of the MPPI method. We then report the mean for the logarithms of these normalized values. Thus, negative values indicate better performance relative to the MPPI method.

Additionally, we conduct experiments with different sampling regimes, specifically with 16 samples for a low-sampling regime, 64 samples for a medium-sampling regime, and 512 samples for a high-sampling regime. Further information regarding the experiments, including the definition of the cost function and hyperparameter details, is available in the Appendix.

Finally, we demonstrate that our framework scales to high-dimensional problems and can seamlessly integrate with alternative optimality distributions, such as the diffusion-based DIAL-MPC (Xue et al. 2025).

5.1 Obstacle Avoidance (PNGRID)

The PNGRID task, as utilized in prior studies like (Sacks and Boots 2023), is an instance of the widely studied obstacle-avoidance problem in the SMPC field. This task, illustrated in Fig. 7-a, involves a 2D environment where an agent must navigate from a randomly selected start point to a randomly assigned target while avoiding static obstacles marked in orange. For this task, the feasibility distribution captures the obstacle-free areas, directing sampling toward viable paths.

Table 1 presents a comparative analysis of system performance across baseline approaches, based on averages from 100 trials with random start and target points. All Proj-MPPI, NFs-PoE-MPPI and TT-PoE-MPPI demonstrate performance improvements over the MPPI method, indicated by negative values, consistent with prior findings (Power and Berenson 2023; Sacks and Boots 2023). Notably, the TT-PoE-MPPI method consistently outperforms the other methods, particularly in terms of success rate within the medium to low sampling regime. However, as the number of samples increases, both NFs-PoE-MPPI and TT-PoE-MPPI exhibit improved performance in terms of step efficiency and cost values. In contrast, Proj-MPPI's performance aligns more closely with MPPI under high-sampling conditions than in the lower and medium sampling regimes. This trend highlights the competitive advantage of the TT-PoE-MPPI and NFs-PoE-MPPI approaches across varying sampling densities.

5.2 Tracking Tube (e.g., surface cleaning or painting)

The proposed method is also applied to a task where the robot's end-effector must traverse a restricted volume (e.g., a spherical or a sinusoidal manifold with some margins, as illustrated in Fig. 8). Such tasks are prevalent in robotics. For instance, when a robot needs to clean or paint a surface or follow a specific path. In this setup, we control the robot's end-effector position in the task space, ensuring it stays within a designated shaded area. However, the feasible region for this task is substantially restricted relative to the entire workspace, making it challenging to sample feasible actions. We employed two different volumes: one between two spheres and another between two sinusoidal waves (Fig. 8). The evaluation results, shown in Table 1, reveal that the proposed method significantly improves the task success rate, especially in low-sampling scenarios. Moreover, our method achieves faster convergence with lower associated costs. As shown in the table, Proj-MPPI exhibits behavior similar to MPPI in terms of success rate (and even lower

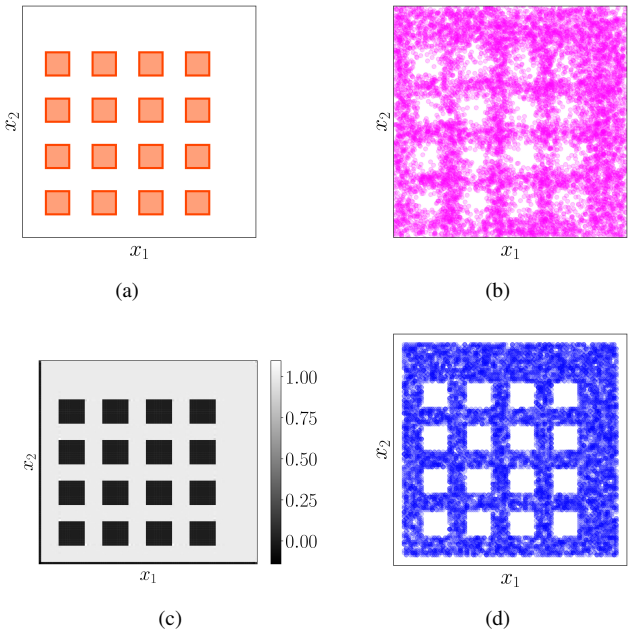


Figure 7. PNGRID Configuration. (a) Depiction of the primary environment, with orange rectangles representing obstacles. (b) Samples generated from a GMM model using $k = 400$ Gaussian components, trained with expectation-maximization (EM) algorithm. (c) Distribution learned through the TT-SVD method, and the samples generated from this distribution are depicted in (d). As evident from the results, the TT-SVD method precisely captures the feasible set, with the generated samples exhibiting a higher concentration within the feasible area compared to those generated by the GMM model.

in the sphere case) and number of steps, while achieving significantly lower cost values. This outcome is likely due to the projection step, which ensures that nearly all samples remain within the designated area, leading to reduced costs. However, this projection mechanism appears to make the system more inclined to stay within its current location rather than actively exploring new solutions, which may contribute to its lower success rate on the sphere manifold compared to MPPI. In contrast, NFs-PoE-MPPI shows markedly lower performance than TT-PoE-MPPI, for reasons that are discussed in detail in Section 6.

In this experiment, our objective was to evaluate the performance of our systems on equality constraints, which in this case correspond to a manifold. However, we acknowledge that the TT method, which is a volumetric representation, cannot directly enforce strict equality constraints. To address this limitation, we evaluated the performance of our approach on a relaxed version of the problem by introducing margin boundaries around the manifold, effectively creating a "soft" constraint region for analysis. We then evaluate how the pipeline behaves as this margin is progressively reduced, and benchmark it against an intrinsic-sampling approach (i.e., Riemannian log/exp maps), which guarantees constraint satisfaction intrinsically by sampling in the tangent space first, then mapping to the task space. For certain manifolds, like a sphere, these mappings can be computed analytically (see Appendix for details).

Table 1. Comparison of various methods across different tasks employing different sampling regimes. Values are normalized to the MPPI method. Hence, the number of steps and cost values for the MPPI method are not provided. Negative values indicate superior performance compared to the MPPI method, with bold formatting indicating superior results.

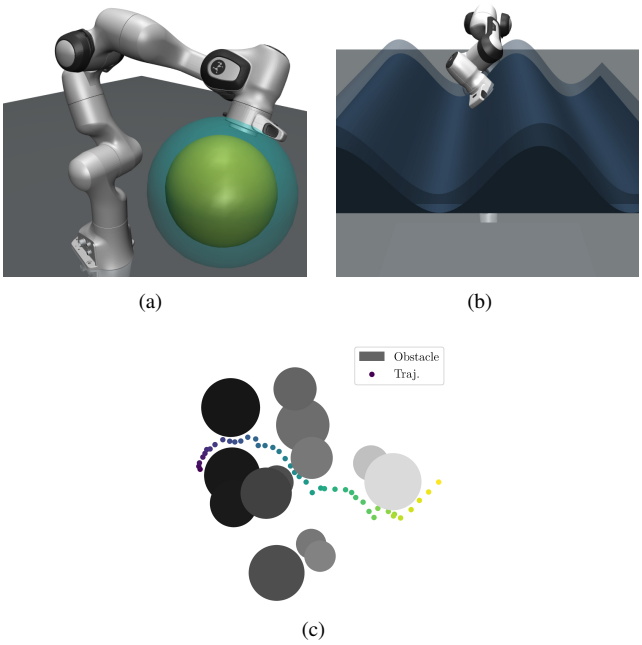
Task	#samples	MPPI	Proj-MPPI			NFs-PoE-MPPI			TT-PoE-MPPI (proposed)		
		succ.	succ.	$\log(\#\text{steps}_n)$	$\log(\text{cost}_n)$	succ.	$\log(\#\text{steps}_n)$	$\log(\text{cost}_n)$	succ.	$\log(\#\text{steps}_n)$	$\log(\text{cost}_n)$
PNGRID	16	46%	70%	−0.07	−0.19	74%	−0.03	−0.04	**96%**	**−0.81**	**−0.35**
	64	81%	92%	−0.06	−0.09	90%	−0.10	−0.16	**100%**	**−0.69**	**−0.90**
	512	93%	96%	−0.06	−0.09	97%	−0.13	−0.16	**100%**	**−0.65**	**−0.78**
Sphere. Man.	16	73%	73%	−0.11	−0.09	75%	0.01	−0.57	**100%**	**−0.32**	**−1.08**
	64	93%	77%	0.03	−1.57	87%	0.20	−0.68	**100%**	**−0.39**	**−4.38**
	512	**100%**	78%	0.05	−1.89	**100%**	−0.26	−1.97	**100%**	**−0.89**	**−3.08**
Sin. Man.	16	31%	38%	−0.09	−1.08	38%	−0.15	−0.08	**100%**	**−1.58**	**−1.86**
	64	44%	44%	−0.09	−0.87	50%	−0.41	−0.40	**100%**	**−2.37**	**−2.03**
	512	55%	68%	−0.30	−0.68	65%	−0.52	−0.14	**100%**	**−1.92**	**−2.40**

Table 2 presents the results of this comparison. We evaluate our proposed method against the Riemannian log/exp maps approach (specifically for a spherical manifold) and a vanilla-MPPI method. As shown in Table 2, when the margin exceeds the TT discretization (2 cm), our TT-PoE-MPPI method outperforms both standard MPPI and the Riemannian mapping approach. We attribute this to the fact that intrinsic sampling in the tangent space incurs inefficiencies, particularly when shifting between time steps and averaging, whereas the TT representation captures the volumetric structure more globally. As the margin shrinks, however, the intrinsic method ultimately yields the best performance, although our method remains superior to vanilla MPPI.

These results suggest that TT-PoE-MPPI can accelerate the dynamic sampler under equality constraints, but that pure intrinsic sampling may be preferable for very tight margins. Moreover, TT-PoE can be combined with intrinsic approaches: once an intrinsic transform renders the problem unconstrained, additional constraints (e.g., joint limits or other inequalities) can still be enforced by defining the TT experts in the transformed space. Note tha the optimality distributions for the intrinsic sampler and for other methods inhabit different spaces in this experiment. Hence, as the margin shrinks to zero, the unconstrained intrinsic sampler can serve as the ground-truth solution to the constrained problem. Furthermore, our product-of-experts framework consistently improves sampling efficiency without imposing additional structural assumptions, such as that the constraint space forms a Riemannian manifold or that the constraints are affine in the action variables, as required by Carius et al. (2021).

5.3 Non-prehensile Manipulation (Planar-pushing)

An other task that we consider is a pushing task (Mason 1986; Xue et al. 2023), a type of non-prehensile, contact-rich manipulation. These tasks are inherently challenging as they require the robot to handle a range of contact models, with hybrid dynamics that complicate the use of gradient-based methods. This hybrid nature arises from various factors, including the geometry of objects, the contact modes, as well as the task level. In this work, we focus specifically on the challenges related to geometry and contact mode, while task-level hybridization has been more extensively explored within task and motion planning (TAMP) frameworks (Holladay et al. 2023).



(a) (b) (c)

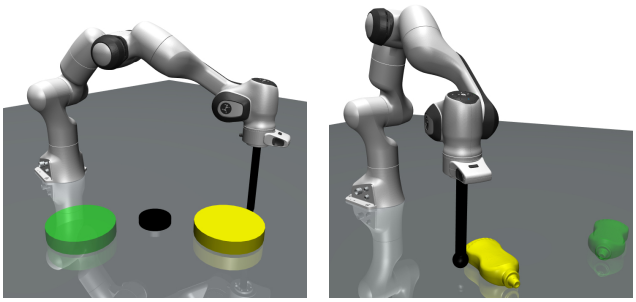
Figure 8. Different environmental setups: (a) sphere tube tracking, (b) sinusoidal wave tube tracking, and (c) Online obstacle-avoidance. In (c), the obstacles become more apparent as the robot approaches them; lighter colors indicate obstacles that appear later in the task.

Previous studies have attempted to address the hybrid nature of contact through unified contact modes (Anitescu 2006; Manchester and Kuindersma 2020; Posa et al. 2014; Yunt and Glocker 2006) or by employing mixed-integer programming (MIP) to jointly optimize hybrid and continuous variables (Aceituno and Rodriguez 2022). However, these approaches struggle with geometric hybridness, and solving complex non-linear, non-convex problems with MIP can be computationally intensive. SMPC offers an effective alternative (Pezzato et al. 2023), as it avoids reliance on gradients. Furthermore, as demonstrated by Pang et al. (2023), the inherent ability of sampling-based methods to smooth functions proves advantageous in these hybrid and contact-rich scenarios.

Using the method developed in this work, the optimality expert can focus on pushing an object toward a goal while adapting to environmental changes, such as the introduction of obstacles that were not present in the training data. The

Table 2. Comparison of different methods on sphere tube-tracking task across varying margin sizes. The intrinsic MPPI method (MPPI solved directly on the spherical manifold using log/exp maps) reformulates the problem as unconstrained and, in the limit as the margin $\to 0$, serves as the *ground truth* for the other baselines. Values are normalized to the MPPI method. Hence, the number of steps and cost values for the MPPI method are not provided. Negative values indicate superior performance compared to the MPPI method, with bold formatting indicating superior results.

Margin size	MPPI	TT-PoE-MPPI			Intrinsic MPPI		
	succ. rate	succ. rate	$\log(\#\text{steps}_n)$	$\log(\text{cost}_n)$	succ. rate	$\log(\#\text{steps}_n)$	$\log(\text{cost}_n)$
5 [cm]	98%	**100%**	**−0.98**	**−2.94**	**100%**	−0.72	−2.55
3 [cm]	91%	**100%**	**−1.06**	**−4.03**	**100%**	−0.81	−3.78
1 [cm]	78%	92%	−0.02	−4.91	**100%**	**−0.37**	**−5.73**



(a) (b)

Figure 9. Different environment setups for pushing various objects: (a) a cylinder and (b) a mustard bottle. The obstacle is shown in black, while the green objects represent the target locations.

feasibility distribution is designed to model the efficiency of the actions taken. As mentioned earlier, certain actions are inconsequential if they lead the robot away from the object or otherwise fail to move the object meaningfully. In an offline phase, we gather data on various state-action combinations, labeling them as efficient if the robot moves toward the object or if the action causes the object to relocate. Specifically, $p_{\text{feas}}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ in (13) returns 1 if the action $\boldsymbol{u}_t$ in state $\boldsymbol{x}_t$ is effective and 0 otherwise. This approach is similar to embedding a cost function that encourages the robot to approach the object. However, we demonstrate that our method is more efficient than relying on a penalty term within the cost function.

In this experimental setup, we employed three objects: a cylinder (Fig. 9-a), a box, and a mustard bottle sourced from the YCB dataset (Calli et al. 2017) (Fig. 9-b). Additionally, for the cylinder scenario, we introduced an obstacle into the environment to assess the efficacy of the methods in navigating around obstacles. It is important to note that the feasibility distribution is unaware of the obstacle in the environment (not considered in the training), as addressing this situation falls within the responsibility of the optimality distribution.

The results for these experiments are shown in Table 3. As can be seen, our approach exhibits enhanced performance when compared to both NFs-PoE-MPPI and MPPI methodologies, demonstrated through superior cost values and a reduced number of steps. Moreover, the proposed method attains a superior success rate compared to other baseline methods.

The proposed method was further applied to the real robot for pushing a mustard bottle using only 32 samples which is very low compared to the commonly used number of samples in other SMPC methods. We chose this low number deliberately to show the capability of our method in increasing the efficiency of samples. Selected frames from this experiment are illustrated in Fig. 10. The rationale behind selecting the mustard bottle lies in its complex shape compared to other objects, thereby serving as a demonstrative example that the shapes of the objects involved do not constrain our method. The objective in this scenario is to guide the object back to the goal location (transparent shape in Fig. 10), while the object undergoes a 90-degree rotation. The receding horizon controller employed allows the system to effectively handle external disturbances, as demonstrated in the accompanying video. Disturbances were introduced to the system between Fig. 10-c and 10-d. Notably, the planner adeptly manages this situation, guiding the object towards the target.

5.4 Online Obstacle Avoidance

One significant advantage of using TT-SVD to learn the TT distribution is its capability to process the entire dataset in milliseconds if all data points are available. This is especially beneficial when the robot leverages point cloud data online to identify free space in the environment. This rapid processing allows the system to adapt its distribution quickly if new environmental changes are detected, enabling it to sample updated, obstacle-free trajectories efficiently. To evaluate this capability, we designed a task in which a point-mass robot moves from the left to the right side of the environment. As it progresses, new obstacles appear, shown as lighter regions in Fig. 8-c. Table 4 presents the results of this experiment, demonstrating that the proposed method consistently outperforms the MPPI method across all measured criteria. Notably, the performance gap between the two methods widens as the number of samples increases.

5.5 Higher Dimensions and Alternative Optimality Distributions

Previous experiments addressed low-dimensional problems using TT-SVD and benchmarked against vanilla MPPI. Here, we demonstrate that our method scales to higher-dimensional settings and can be combined with alternative optimality distributions. We evaluate our method on a task in which a Franka Emika robot must traverse an environment with whole-body obstacle avoidance, where

Table 3. Comparison of various methods across different tasks employing different sampling regimes. Values are normalized to the MPPI method; hence, the number of steps and cost values for the MPPI method are not provided. Negative values indicate superior performance compared to the MPPI method, with bold formatting indicating superior results.

Task	#samples	MPPI	NFs-PoE-MPPI			TT-PoE-MPPI		
		succ. rate	succ. rate	log(#steps$_n$)	log(cost$_n$)	succ. rate	log(#steps$_n$)	log(cost$_n$)
Push. Cir.	16	89%	**96%**	−0.19	−0.22	95%	**−0.35**	**−0.41**
	64	96%	99%	−0.27	−0.30	**100%**	**−0.38**	**−0.45**
	512	97%	**100%**	−0.24	−0.27	**100%**	**−0.38**	**−0.45**
Push. Box	16	**100%**	**100%**	0.29	0.28	**100%**	**−0.17**	**−0.21**
	64	**100%**	**100%**	−0.06	−0.08	**100%**	**−0.25**	**−0.29**
	512	**100%**	**100%**	0.25	0.23	**100%**	**−0.01**	**−0.03**
Push. Mus.	16	70%	73%	0.12	0.11	**89%**	**−0.21**	**−0.25**
	64	75%	90%	−0.01	−0.02	**91%**	**−0.33**	**−0.34**
	512	75%	90%	0.01	−0.01	**91%**	**−0.26**	**−0.28**

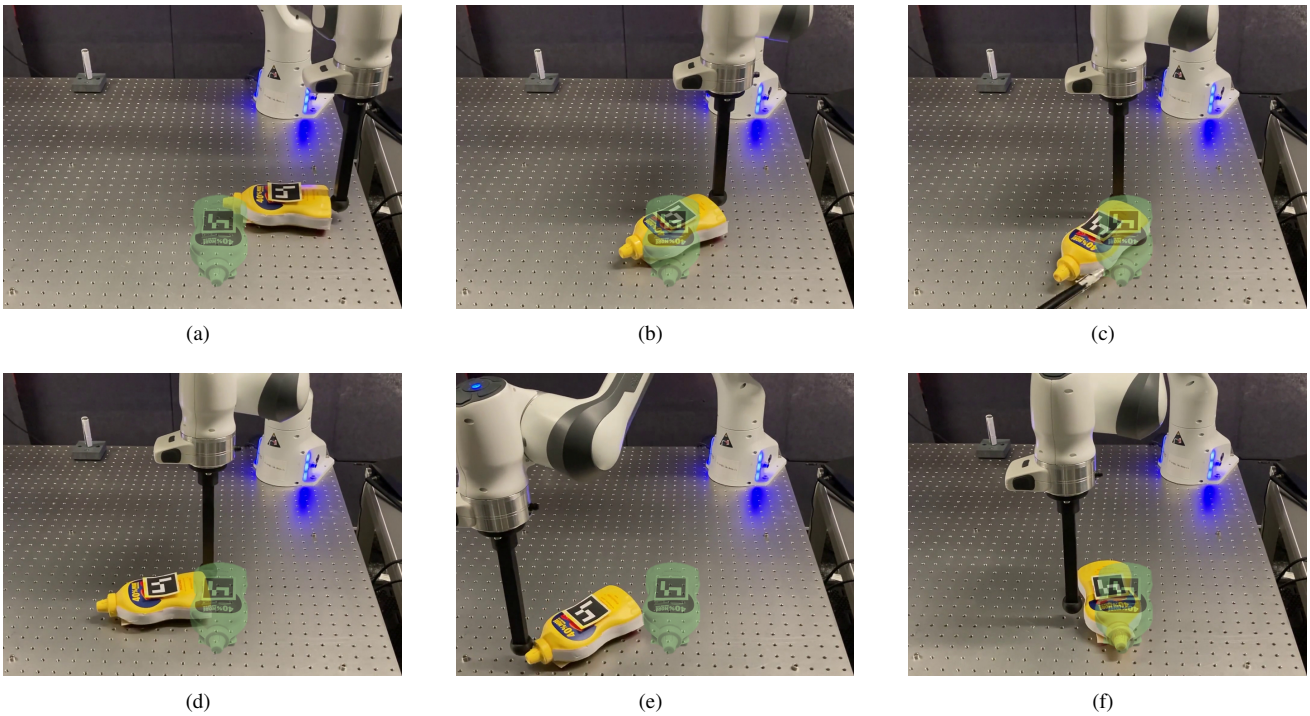


Figure 10. Frames selected from the real robot experiment showing the process of pushing a mustard bottle towards the target (shown as the semi-transparent picture). An external disturbance was introduced to the system at frame (c). However, the planner effectively navigates this scenario, successfully guiding the object toward the target.

Table 4. Comparison of TT-PoE-MPPI with MPPI in an online obstacle avoidance task across 100 random obstacle locations. Values are normalized to the MPPI method. Thus, steps and cost values for MPPI are omitted. Negative values indicate superior performance, with bold formatting highlighting the best results.

#samples	MPPI	TT-PoE-MPPI		
	succ. rate	succ. rate	log(# steps$_n$)	log(cost$_n$)
16	73%	**84%**	**−0.09**	**−0.13**
64	73%	**86%**	**−0.31**	**−0.44**
512	69%	**89%**	**−0.46**	**−0.66**

obstacle locations vary across scenes (as illustrated Fig. 1-b).

Since TT-SVD becomes impractical for the large tensors required, we instead employ TT-cross with a fixed maximum rank $r_{\max} = 20$. This low-rank approximation reconstructs the full tensor from a subset of entries. Our approach can be applied in two ways:

1. Augmented State Conditioning: Augment the state with all obstacle positions and, at inference time, condition on the current robot state $\boldsymbol{x}_t$ and obstacle locations $\boldsymbol{x}^o_{1:N_o}$. We define the joint feasibility distribution as

$$p^f\left(\boldsymbol{x}^o_{1:N_o}, \boldsymbol{x}_t, \boldsymbol{u}_t\right) = \left(1 - \exp(-\text{dist}(\boldsymbol{x}_t)/0.5)\right) \left(1 - \exp(-\text{dist}(\boldsymbol{x}_{t+1})/0.5)\right), \quad (25)$$

where $\text{dist}(\cdot)$ is the minimum distance between the robot body and obstacles computed via the Robot Distance Field (RDF) in (Li et al. 2024). By using a smooth, distance-based likelihood rather than a binary collision indicator, we favor configurations that keep the robot at a safe distance and produce a lower-rank function amenable to TT-cross learning.

Table 5. Comparison of different methodologies in whole body obstacle avoidance task across 100 random obstacle locations and initial and target configurations using 1024 sample size. Values are normalized to the MPPI method. Thus, steps and cost values for MPPI are zero. Negative values indicate superior performance, with bold formatting highlighting the best results.

Method	succ. rate	$\log(\#\text{ steps}_n)$	$\log(\text{cost}_n)$
MPPI	30%	0.00	0.00
TT-PoE-MPPI	53%	−0.10	−0.09
TT-PoE-MPPI-prod	50%	−0.14	−0.11
DIAL	88%	−0.35	−0.36
TT-PoE-DIAL	**89%**	**−0.50**	**−0.53**

2. Additive Feasibility Mixture: Learn a separate feasibility tensor $\mathcal{P}^{f_i}$ for only one obstacle, then combine them as

$$\mathcal{P}^{f} = \sum_{i=1}^{N_o} \mathcal{P}^{f_i}.$$

The additive mixture remains smoother with a lower rank representation, which is more amenable to the TT-cross method. We denote this variant as *TT-PoE-MPPI-prod*.

We further evaluate our framework against DIAL-MPC (Xue et al. 2025), an annealed MPPI variant that adapts its covariance using diffusion-inspired updates (Ho et al. 2020; Chi et al. 2024). Although DIAL-MPC is significantly more efficient than vanilla MPPI, it still relies on rejection sampling to eliminate infeasible trajectories—because its experts are fused at the function level and have no direct awareness of the feasible solution space. By contrast, our solution-level fusion lets us steer the sampler toward valid regions before any draws are discarded. To highlight this advantage, we embed DIAL-MPC within our TT-PoE pipeline (denoted *TT-PoE-DIAL*) and show that this hybrid consistently outperforms both standalone DIAL-MPC and our original TT-PoE-MPPI in terms of sampling efficiency and overall task success.

Table 5 reports success rate, total cost, and steps-to-goal for each method. Integrating TT with both MPPI and DIAL-MPC increases success rates while reducing cost and required steps. Notably, the TT-PoE-MPPI-prod variant matches or exceeds the performance of TT-PoE-MPPI, validating the effectiveness of learning and summing individual feasibility distributions within a product-of-experts framework. We define task success as reaching the target in fewer than 25 steps with total cost below 10^6. Further experimental details are provided in the Appendix.

6 Discussion

The primary objective of this article is to enhance the sampling distribution employed in SMPC through the application of products of experts in the solution space. As anticipated, all methods with TT-PoE pipeline (NFs-PoE-MPPI, TT-PoE-MPPI, and TT-PoE-DIAL) exhibit superior performance compared to their conventional counterparts. However, the performance of NFs-PoE was not significantly better than that of Proj-MPPI, and in all experiments, TT-PoE consistently outperformed NFs-PoE by a substantial margin. In the subsequent sections, we will delve deeper into the reasons behind this performance, explaining the advantages of using TT-distribution over NFs models and identifying its limiting factors.

6.1 TT-distribution vs Normalizing flow

The rationale behind choosing normalizing flows (NFs) over other generative approaches like autoencoders (Schmidhuber 2015) or generative adversarial networks (GANs) (Goodfellow et al. 2014) lies in the NFs' unique ability to map samples from the latent space to the main space and vice versa. Such bidirectional mapping is crucial when combining multiple distributions in a product, ensuring they operate within the same space.

Nevertheless, similar to other latent space learning methods, the acquired latent space lacks *intuitiveness* (Losey et al. 2022). In this context, *intuitive* refers to the observed relation between data points in the latent space, not necessarily aligning with their distance in the main spaces. For instance, when two proximate data points exhibit similar distances to an objective point in latent space, while in task space, one is positioned near the nominal solution and the other is distant. This phenomenon is illustrated in Fig. 11, obtained from a model trained for the PNGRID task, where the nominal action is presumed to be $[-0.5, -0.5]$ at state $[-1, -0.9]$, and the Gaussian model utilized for the MPPI method possesses identical variance in both directions. At this point, only velocity constraints (ranging between -1 and 1) must be active, with no other constraints in effect.

To generate samples from normalizing flows (NFs), the process involves mapping the nominal state-action back into the latent space (referred to as nominal-latent point), assuming a Gaussian distribution around it, combining it with the learned feasible distribution for the task, and sampling from the resulting distribution. The nominal latent point is depicted in dark blue in Fig. 11-a, with other samples represented in distinct colors corresponding to their distance from the nominal latent point (darker blue indicates proximity, while darker red signifies greater distance). These samples are then mapped back into the task space, shown in Fig. 11-b, where the colors in both images correspond, i.e. each sample maintains the same color in both representations. Notably, all samples fall within the -1 to 1 range in both directions, representing the action limits (see Appendix for experiment details).

Observing the task space samples, it is evident that they are not entirely clustered around the nominal action $[-0.5, 0.5]$. Additionally, the distribution of samples in both directions appears dissimilar. This discrepancy arises due to the nonlinear function employed to revert the data back to the task space, which may have varying effects in different directions. This suggests that the covariance matrix in the latent space must be meticulously chosen to ensure the desired distribution upon reverting the data to the task space. From the perspective of obtaining random samples, this might not seem like a significant issue, as there are indeed samples close to the nominal solutions. However, the bias introduced in NFs can render the MPPI solution inefficient, which we believe is a key factor contributing to the inferior performance of this method compared to our proposed method, as well as to the MPPI method in certain cases.

In contrast, samples obtained from the TT-distribution, as illustrated in Fig. 11-c, are centered around the desired nominal point and exhibit a consistent distribution in both directions. This is attributed to the fact that in the TT-distribution, the product of the two distributions is executed in the task space without the need to map the data to the latent space and back. This preserves the original distribution shape, contributing to increased efficiency. An alternative methodology involves the application of techniques like GMMs directly in the task space. While this approach holds promise in addressing the previously mentioned issue, a challenge arises from its parametric nature, a topic that will be further explored in subsequent discussions.

Another challenge associated with normalizing flows is that, despite enhancing the probability of positive samples in the latent space, it does not entirely eliminate the probability of negative samples. This stems from the method relying on the Gaussian distribution, a parametric distribution wherein the probability values decrease smoothly without abruptly reaching zero. This issue is also prevalent in other distribution learning methods such as GMMs. To illustrate this point, we model the distribution of the feasible set of the PNGRID area using GMM. Fig. 7-b displays approximately 10,000 samples generated from the GMM model, which employed 400 Gaussian components. The results show that the model fails to efficiently capture the boundaries, compared to TT-distribution samples depicted in Fig. 7-d. Additionally, it is noteworthy that the training process for the GMM model required significantly more time: approximately 21 minutes, compared to around 5 minutes for normalizing flows and just milliseconds for TT-SVD.

This issue also affects the product distribution when combining it with the one obtained from MPPI. If the two distributions (optimality and feasibility) do not have overlapping high-probability areas, such that the nominal trajectory is close to regions of infeasibility (low-probability areas), the system will consider the low-probability areas of both distributions as possible candidates.

In contrast, the TT approach mitigates this problem to a greater extent. As a non-parametric distribution model, TT does not require probabilities to change smoothly, allowing it to capture sharp transitions in the distribution more effectively. Furthermore, the TT model takes both positive and negative samples into account, aiming to assign a probability of zero to the infeasible areas and one to the feasible areas. Consequently, it acts as a filter to eliminate the possibility of sampling from an infeasible distribution. This means that after combining the two distributions, the infeasible areas will ideally have zero probability, regardless of the characteristics of the optimality distribution.

6.2 Low-Rank Representation

As highlighted in prior works such as (Shetty et al. 2024a), one of the features that enhance the efficiency of TT methods is their ability to represent high-dimensional tensors in low-rank formats. The rank of a tensor can be understood analogously to that of a matrix, defined as the number of independent fibers in the tensor. This low-rank representation plays a pivotal role in our process for several reasons.

First, it allows the entire tensor to be stored in a significantly more memory-efficient manner. Second, techniques such as TT-cross (Savostyanov and Oseledets 2011) exploit this property to learn the TT representation of a tensor without requiring access to all its elements, reducing computational overhead. Third, this low-rank structure enables faster and more efficient sampling strategies (compared to sampling directly from the whole tensor), an advantage leveraged extensively in our proposed method. Finally, it allows operations like tensor products to be executed directly at the core level, bypassing the need to reconstruct or manipulate the full tensor.

To clarify why low-rank representations make such operations efficient, it is essential to examine the relationship between tensor rank and the separability of a function's variables. When a function is converted into a tensor representation, each edge in the resulting tensor corresponds to one argument of the function. Without delving into excessive detail, it can be demonstrated that the rank of a tensor representing a function is inversely related to the separability of the function's variables. For instance, a fully separable function, such as $f(x, y) = f_x(x)f_y(y)$, results in a rank-1 tensor, irrespective of the specific forms or complexity of $f_x(\cdot)$ and $f_y(\cdot)$. This separability is also evident at the core level, where each tensor core corresponds to a distinct argument of the function. This property was crucial to our approach. When combining the optimality and feasibility distributions, we manipulated only the cores associated with the action commands, leaving other cores unchanged.

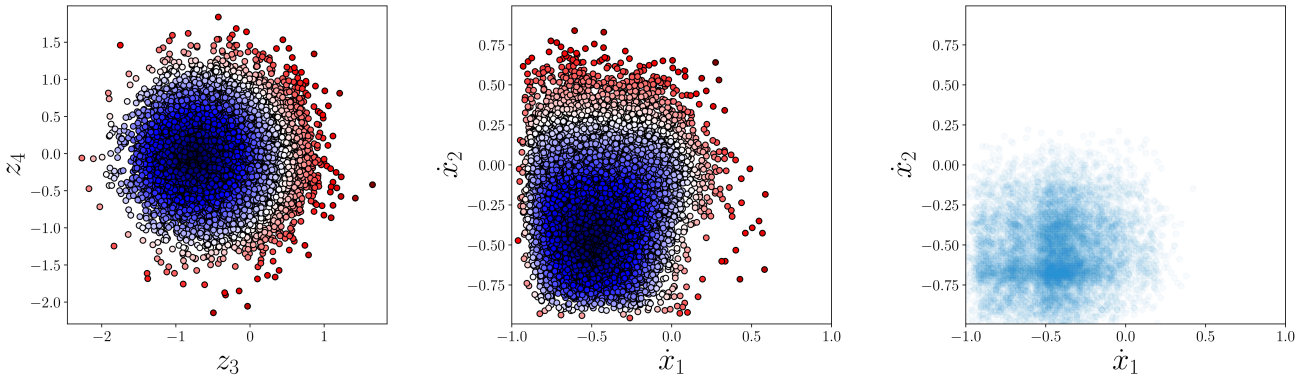
6.3 Computation time

As outlined in Alg. 1, our method adds only two extra operations (lines 3–4) and modifies the sampling step (line 5) compared to MPPI. Table 6 shows how these changes affect computation time. Although MPPI achieves faster overall runtimes, our TT-sampling times are on par with other baselines, making it suitable for online planning.

In its current Python implementation, TT-sampling scales linearly with the number of action variables—e.g., the whole-body obstacle-avoidance task uses 3.5× more—because it relies on a Python library (i.e., tntorch) with a for-loop inside, and lacks multithreading. By contrast, MPPI can leverage highly optimized native libraries and its state-independent Gaussian distribution to improve the computation time significantly. By migrating performance-critical TT sections to C/C++ and enabling multithreading, we estimate TT's sampling time could drop below 1 ms. Until those optimizations are implemented, the timings reported here reflect the current implementation. All measurements were recorded using an NVIDIA RTX 3090 GPU and a 12-core CPU.

6.4 Limitations

Despite its strengths, our approach has scalability limits: while we've demonstrated performance on moderately high-dimensional systems (which can be sufficient for a wide range of robotic tasks), it does not extend readily to very high dimensions (e.g., over 50). Several avenues could address this. First, a multi-resolution discretization scheme (e.g.,



(a) Latent space samples (b) Samples mapped to the task space (c) Samples from TT-distribution

Figure 11. Samples gathered with different methods. In (a) and (b), the color gradient represents the proximity of the samples to the mean action of MPPI, with darker blue indicating closer proximity and darker red indicating greater distance. It is evident that samples generated in the latent space are distorted when translated back to the task space: they are not centered around the desired action and exhibit varying spans across dimensions despite having the same variance. Conversely, samples generated via the TT method are centered around the desired action and exhibit consistent spans in both dimensions.

Table 6. Average computation times (in milliseconds) for each method on the PNGRID (P) and whole-body obstacle avoidance (W) tasks. MPPI's runtimes were virtually identical across both tasks and are therefore reported once. Note that computation times for related methods (e.g., DIAL and TT-PoE-DIAL) can be inferred directly from this table. Although our proposed method is slower than MPPI, it is on par with the other techniques and, as shown in the performance tables, yields a significant improvement over MPPI.

Method	Product [ms]	Conditioning [ms]	Sampling [ms]	Mapping [ms]	Projection [ms]	Total [ms]
NFs-PoE-MPPI (P)	0.03 ± 0.00	-	0.48 ± 0.02	$2.50 \pm 0.07(\times 2)$	-	5.51 ± 0.16
MPPI	-	-	0.48 ± 0.02	-	-	$\mathbf{0.48 \pm 0.02}$
Proj_MPPI (P)	-	-	0.48 ± 0.02	-	5.02 ± 26.46	5.50 ± 26.48
TT-PoE-MPPI (P)	0.02 ± 0.00	0.04 ± 0.00	2.67 ± 0.02	-	-	2.73 ± 0.02
TT-PoE-MPPI (W)	0.13 ± 0.01	1.23 ± 0.10	5.92 ± 0.31	-	-	7.28 ± 0.42

using coarse grids when the robot is far from obstacles and finer grids when it nears them) can dramatically shrink the effective discretization size and enable higher-dimensional problems. Second, leveraging cumulative learning to inform the TT-cross algorithm may accelerate convergence and improve accuracy. Finally, integrating TT models with complementary learning techniques to discover compact latent representations could further reduce dimensionality and allow TT methods to operate more efficiently in even larger state spaces.

Alternatively, recent advances in generative modeling techniques, such as diffusion-based models (Ho et al. 2020) and energy-based models (LeCun et al. 2006), present opportunities for further exploration. These methods could be extended for use within the MPC framework discussed in this work, as our PoE-MPPI approach is independent of TT methods and can be adapted to incorporate these generative approaches. In this work, we have already demonstrated the mutual benefits of combining our pipeline with DIAL-MPC (a data-free, diffusion-inspired sampler), and future work could similarly integrate other generative approaches to further improve sampling efficiency and constraint satisfaction.

It is also worth noting that while generative methods typically rely on demonstration data to train individual experts, this requirement can be challenging when exploring the entire feasibility region, as demonstrations may not sufficiently cover all possible scenarios. By contrast, the TT approach does not depend on demonstration data and instead operates on functions. This design choice has its advantages for our purpose but may also introduce limitations for certain applications. In some applications, a hybrid strategy that combines data-driven and function-based experts could offer the best of both worlds.

There are two potential avenues for enhancing the proposed method. Firstly, the system's capability for long-horizon planning could be improved by incorporating an additional expert that considers environmental contexts. These methods can be regarded as supplementary experts and seamlessly integrated with other methodologies through the products of experts method elucidated in this paper, potentially leading to increased efficiency. Secondly, recent findings (Shetty et al. 2024b; Xue et al. 2024b) demonstrate the applicability of TT models for solving approximate dynamic programming and deriving policy models for robots in TT form. Their approach involves selecting the most optimal or robust action from the model, but we can modify the pipeline by sampling from the TT model and incorporating additional experts, as discussed in this paper. This modification enables a more flexible pipeline, distributing tasks among different experts and mitigating the reliance on achieving convergence to the most optimal solutions.

7 Conclusion

In this article, we presented a novel approach to enhance the efficiency of sampling-based model predictive control by leveraging products of experts in the solution space and discussing the introduction of a feasibility expert.

We illustrated the efficacy of this concept, which differs from traditional projection methods (the *sample-then-project* strategy or tangent-space-related works) by first projecting the optimality distribution into the feasible area and then sampling from it (the *project-then-sample* strategy). This approach increases the likelihood of each sample being accepted and provides a more diverse set of samples within the boundaries, rather than focusing solely on the boundaries. Additionally, it is well-suited for tasks where computing the gradient function is challenging, such as non-prehensile manipulation. It also establishes a flexible framework to integrate solutions from different methods, each with distinct objectives and decision variables. This capability allows us to reshape the sampling distribution before drawing samples, enabling more targeted and efficient exploration. We demonstrated how this advantage can be leveraged across different sampling-based frameworks, including MPPI and DIAL-MPC, resulting in improved sampling efficiency and task performance in both cases.

We demonstrated that this idea can be realized using tensor train (TT) distribution models, which allow for seamless integration of different distributions, including solutions from other sampling-based methods. We compared the advantages of TT-distributions to another distribution learning method, specifically normalizing flows. The significant benefits of TT-distributions lie in their non-parametric nature, their ability to facilitate product operations with other distributions at the task level, and their straightforward sampling approaches. We illustrated these advantages across multiple tasks.

To address the current limitations of our method in higher dimensions, future research could investigate the potential of initially learning a low-dimensional latent space and subsequently leveraging the TT-PoE method. Another promising avenue involves incorporating additional experts with additional modalities (e.g., demonstrations or other optimality distributions . Similarly, some policies learned in the field of reinforcement learning, such as (Shetty et al. 2024b; Xue et al. 2024b), can be viewed as distributions and could function as experts to enhance and modify the proposed method in this study.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the State Secretariat for Education, Research and Innovation in Switzerland for participation in the European Commission's Horizon Europe Program through the INTELLIMAN project (https://intelliman-project.eu/, HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENSO project (https://sestosenso.eu/, HORIZON-CL4-Digital-Emerging Grant 101070310), as well as by the China Scholarship Council (grant No. 202106230104).

ORCID iDs

Amirreza Razmjoo https://orcid.org/0000-0003-3826-6608
Teng Xue https://orcid.org/0009-0001-7414-3958,
Suhan Shetty https://orcid.org/0000-0002-7550-9368,
Sylvain Calinon https://orcid.org/0000-0002-9036-6799

References

Aceituno B and Rodriguez A (2022) A Hierarchical Framework for Long Horizon Planning of Object-Contact Trajectories. In: *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. IEEE, pp. 189–196.

Anitescu M (2006) Optimization-based simulation of nonsmooth rigid multibody dynamics. *Mathematical Programming* 105: 113–143.

Batsheva A, Chertkov A, Ryzhakov G and Oseledets I (2023) PROTES: Probabilistic Optimization with Tensor Sampling. *arXiv preprint arXiv:2301.12162* .

Berenson D (2011) *Constrained manipulation planning*. Carnegie Mellon University.

Bhardwaj M, Sundaralingam B, Mousavian A, Ratliff N, Fox D, Ramos F and Boots B (2021) Further experimental results: Manipulation-mpc. Online at manipulation-MPC project site, hosted via Google Sites. URL https://sites.google.com/view/manipulation-mpc/further-experimental-results#h.ud1wkx4676yl. Accessed: 2025-07-30.

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

Calinon S (2016) A Tutorial on Task-Parameterized Movement Learning and Retrieval. *Intelligent Service Robotics* 9(1): 1–29. DOI:10.1007/s11370-015-0187-9.

Calli B, Singh A, Bruce J, Walsman A, Konolige K, Srinivasa S, Abbeel P and Dollar AM (2017) Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research* 36(3): 261–268.

Carius J, Ranftl R, Farshidian F and Hutter M (2021) Constrained stochastic optimal control with learned importance sampling: A path integral approach. *The International Journal of Robotics Research* 41(2): 189–209.

Chavan-Dafle N, Holladay R and Rodriguez A (2020) Planar in-hand manipulation via motion cones. *The International Journal of Robotics Research* 39(2-3): 163–182.

Chi C, Xu Z, Feng S, Cousineau E, Du Y, Burchfiel B, Tedrake R and Song S (2024) Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *The International Journal of Robotics Research* .

Dafle N, Rodriguez A, Paolini R, Tang B, Srinivasa S, Erdmann M, Mason M, Lundberg I, Staab H and Fuhlbrigge T (2014) Extrinsic dexterity: In-hand manipulation with external forces. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. pp. 1578–1585.

Diehl M, Bock HG, Diedam H and Wieber PB (2005) Fast direct multiple shooting algorithms for optimal robot control. In: *Fast Motions in Biomechanics and Robotics*.

Dinh L, Sohl-Dickstein J and Bengio S (2016) Density estimation using Real NVP. In: *International Conference on Learning*

Representations (ICLR).

Dolgov S, Anaya-Izquierdo K, Fox C and Scheichl R (2020) Approximation and sampling of multivariate probability distributions in the tensor train decomposition. *Statistics and Computing* 30: 603–625.

Gkanatsios N, Jain A, Xian Z, Zhang Y, Atkeson C and Fragkiadaki K (2023) Energy-based Models are Zero-Shot Planners for Compositional Scene Rearrangement. In: *Robotics: Science and Systems*.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y (2014) Generative adversarial nets. *Advances in neural information processing systems* 27.

Hinton GE (1999) Product of experts. In: *Proc. Intl Conf. on Artificial Neural Networks. (ICANN)*.

Ho J, Jain A and Abbeel P (2020) Denoising Diffusion Probabilistic Models. *arXiv preprint arxiv:2006.11239* .

Holladay R, Lozano-Pérez T and Rodriguez A (2023) Robust planning for multi-stage forceful manipulation. *The International Journal of Robotics Research* .

Howell T, Gileadi N, Tunyasuvunakool S, Zakka K, Erez T and Tassa Y (2022) Predictive sampling: Real-time behaviour synthesis with mujoco. *arXiv preprint arXiv:2212.00541* .

Jankowski J, Brudermüller L, Hawes N and Calinon S (2023) VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. pp. 10125–10131.

Jinkyu Kim IK and Park FC (2016) Randomized path planning for tasks requiring the release and regrasp of objects. *Advanced Robotics* 30(4): 270–283.

Jordan MI and Jacobs RA (1993) Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation* 6: 181–214.

Kavraki LE, Svestka P, Latombe JC and Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12(4): 566–580.

Kingston Z, Moll M and Kavraki LE (2018) Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems* 1(1): 159–185.

Kobilarov M (2012) Cross-entropy motion planning. *The International Journal of Robotics Research* 31(7): 855–871.

Kuffner JJ and LaValle SM (2000) RRT-connect: An efficient approach to single-query path planning. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, volume 2. pp. 995–1001.

LeCun Y, Chopra S, Hadsell R, Ranzato M and Huang F (2006) A tutorial on energy-based learning. *Predicting structured data* 1(0).

Li Y, Zhang Y, Razmjoo A and Calinon S (2024) Representing Robot Geometry as Distance Fields: Applications to Whole-body Manipulation. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. pp. 15351–15357.

Liang J, Makoviychuk V, Handa A, Chentanez N, Macklin M and Fox D (2018) Gpu-accelerated robotic simulation for distributed reinforcement learning. In: *Conference on Robot Learning*. PMLR, pp. 270–282.

Lipman Y, Chen RTQ, Ben-Hamu H, Nickel M and Le M (2023) Flow Matching for Generative Modeling. In: *The Eleventh International Conference on Learning Representations*.

Losey DP, Jeon HJ, Li M, Srinivasan K, Mandlekar A, Garg A, Bohg J and Sadigh D (2022) Learning latent actions to control assistive robots. *Autonomous robots* 46(1): 115–147.

Manchester Z and Kuindersma S (2020) Variational contact-implicit trajectory optimization. In: *Robotics Research: The 18th International Symposium ISRR*. Springer, pp. 985–1000.

Mason MT (1986) Mechanics and Planning of Manipulator Pushing Operations 5(3): 53–71.

Mirabel J, Tonneau S, Fernbach P, Seppälä AK, Campana M, Mansard N and Lamiraux F (2016) Density estimation using Real NVP. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Mühlbauer M, Hulin T, Weber B, Calinon S, Stulp F, Albu-Schäffer A and Silvério J (2024) A Probabilistic Approach to Multi-Modal Adaptive Virtual Fixtures. *IEEE Robotics and Automation Letters (RA-L)* 9(6): 5298–5305.

Orthey A, Chamzas C and Kavraki LE (2023) Sampling-Based Motion Planning: A Comparative Review. *Annual Review of Control, Robotics, and Autonomous Systems* 7.

Oseledets IV (2011) Tensor-train decomposition. *SIAM Journal on Scientific Computing* 33(5): 2295–2317.

Pan C, Yi Z, Shi G and Qu G (2024) Model-based Diffusion for Trajectory Optimization. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Pang T, Suh HT, Yang L and Tedrake R (2023) Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on Robotics* .

Pardo D, Neunert M, Winkler AW and Buchli J (2015) Projection based whole body motion planning for legged robots.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M and Duchesnay E (2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Pezzato C, Salmi C, Spahn M, Trevisan E, Alonso-Mora J and Corbato CH (2023) Sampling-based model predictive control leveraging parallelizable physics simulations. *arXiv preprint arXiv:2307.09105* .

Pignat E, Silvério J and Calinon S (2022) Learning from Demonstration using Products of Experts: Applications to Manipulation and Task Prioritization. *International Journal of Robotics Research (IJRR)* 41(2): 163–188. DOI:10.1177/02783649211040561.

Posa M, Cantu C and Tedrake R (2014) A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research* 33(1): 69–81.

Power T and Berenson D (2023) Variational Inference MPC using Normalizing Flows and Out-of-Distribution Projection. In: *Robotics science and systems*.

Pradalier C, Colas F and Bessiere P (2003) Expressing bayesian fusion as a product of distributions: Applications in robotics. In: *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, volume 2. pp. 1851–1856.

Sacks J and Boots B (2023) Learning Sampling Distributions for Model Predictive Control. In: *Conference on Robot Learning*. PMLR, pp. 1733–1742.

Savostyanov D and Oseledets I (2011) Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In: *The 2011*

International Workshop on Multidimensional (nD) Systems. IEEE, pp. 1–8.

Schmidhuber J (2015) Deep learning in neural networks: An overview. *Neural networks* 61: 85–117.

Shetty S, Lembono T, Löw T and Calinon S (2024a) Tensor Trains for Global Optimization Problems in Robotics. *International Journal of Robotics Research (IJRR)* .

Shetty S, Xue T and Calinon S (2024b) Generalized Policy Iteration using Tensor Approximation for Hybrid Control. In: *Proc. Intl Conf. on Learning Representations (ICLR)*.

Stilman M (2010) Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics* 26(3): 576–584.

Suh C, Um TT, Kim B, Noh H, Kim M and Park FC (2011) Tangent space RRT: A randomized planning algorithm on constraint manifolds. In: *IEEE International Conference on Robotics and Automation*. pp. 4968–4973.

Sutton RS and Barto AG (2018) *Reinforcement learning: An introduction*. MIT press.

Todorov E and Jordan MI (2002) Optimal feedback control as a theory of motor coordination. *Nature neuroscience* 5(11): 1226–1235.

Usvyatsov M, Ballester-Ripoll R and Schindler K (2022) tntorch: Tensor Network Learning with PyTorch.

Wagener N, Cheng C, Sacks J and Boots B (2019) An Online Learning Approach to Model Predictive Control. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Wang L, Zhao J, Du Y, Adelson EH and Tedrake R (2024) Poco: Policy composition from and for heterogeneous robot learning. *Proc. Robotics: Science and Systems (RSS)* .

Williams G, Aldrich A and Theodorou EA (2017a) Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics* 40(2): 344–357.

Williams G, Wagener N, Goldfain B, Drews P, Rehg JM, Boots B and Theodorou EA (2017b) Information theoretic MPC for model-based reinforcement learning. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. pp. 1714–1721.

Xue H, Pan C, Yi Z, Qu G and Shi G (2025) Full-Order Sampling-Based MPC for Torque-Level Locomotion Control via Diffusion-Style Annealing. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. pp. 4974–4981.

Xue T, Girgin H, Lembono T and Calinon S (2023) Demonstration-guided Optimal Control for Long-term Non-prehensile Planar Manipulation. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. pp. 4999–5005.

Xue T, Razmjoo A, Shetty S and Calinon S (2024a) Logic-Skill Programming: An Optimization-based Approach to Sequential Skill Planning. In: *Proc. Robotics: Science and Systems (RSS)*.

Xue T, Razmjoo A, Shetty S and Calinon S (2024b) Robust Manipulation Primitive Learning via Domain Contraction. In: *Proc. Conference on Robot Learning (CoRL)*.

Yakey JH, LaValle SM and Kavraki LE (2001) Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation* 17(6): 951–958.

Yang Z, Mao J, Du Y, Wu J, Tenenbaum JB, Lozano-Pérez T and Kaelbling LP (2023) Compositional Diffusion-Based Continuous Constraint Solvers. In: *Conference on Robot Learning*.

Yunt K and Glocker C (2006) Trajectory optimization of mechanical hybrid systems using SUMT. In: *IEEE International Workshop on Advanced Motion Control*. pp. 665–671.

A Appendix

A.1 TT-Sample

We demonstrate the application of TT-cores in the process of generating new data samples. For the sake of clarity, we omit the notation of Z, i.e. $Z = 1$, as it is not necessary for sampling from the specified distribution, thereby $\Pr(\boldsymbol{x}) = |\boldsymbol{\mathcal{P}}_{\boldsymbol{x}}|, \quad \boldsymbol{x} \in \Omega_{\boldsymbol{x}}$. Using a chain of conditional distributions, we can define

$$\Pr(x_1, \dots, x_d) = \Pr_1(x_1)\Pr_2(x_2|x_1)\cdots \\ \cdots \Pr_d(x_d|x_1, \dots, x_{d-1}),$$

where

$$\Pr_k(x_k|x_1, \dots, x_{k-1}) = \frac{s_k(x_1, \dots, x_k)}{s_{k-1}(x_1, \dots, x_{k-1})}$$

is the conditional distribution defined using the marginals

$$s_k(x_1, \dots, x_k) = \sum_{x_{k+1}} \cdots \sum_{x_d} \Pr(x_1, \dots, x_d),$$

with the assumption that $s_0 = 1$. Utilizing the definitions outlined above, we can proceed to generate samples $\boldsymbol{x} \sim \Pr(\cdot)$ by sampling from each conditional distribution sequentially. Each conditional distribution pertains solely to one variable, and in the discrete scenario, it can be interpreted as a multinomial distribution, wherein

$$x_k \sim \Pr_k(x_k|x_1, \dots, x_{k-1}), \ \forall k \in \{1, \dots, d\}.$$

This approach entails significant computational demands, as sampling x_k necessitates accessing the conditional distribution $\Pr_k$, which, in turn, mandates evaluating summations across multiple variables to ascertain the marginal s_k, which escalates exponentially with the dimensionality of the problem. The function separability, facilitated by the TT-model, can be leveraged to address this concern. Let $\boldsymbol{\mathcal{P}}$ denote the discrete counterpart of the function $\Pr(\cdot)$, represented as a tensor in TT format, with discretization set $\mathcal{X}$. If the TT model is composed of cores $(\boldsymbol{\mathcal{P}}^1, \dots, \boldsymbol{\mathcal{P}}^d)$, then we can express

$$\begin{aligned} s_k(x_1, \dots, x_k) &= \sum_{x_{k+1}} \cdots \sum_{x_d} \Pr(\boldsymbol{x}), \\ &\approx \sum_{x_{k+1}} \cdots \sum_{x_d} \boldsymbol{\mathcal{P}}_{\boldsymbol{x}}, \\ &= \sum_{x_{k+1}} \cdots \sum_{x_d} \boldsymbol{\mathcal{P}}^1_{:,x_1,:} \cdots \boldsymbol{\mathcal{P}}^k_{:,x_k,:} \boldsymbol{\mathcal{P}}^{k+1}_{:,x_{k+1},:} \cdots \boldsymbol{\mathcal{P}}^d_{:,x_d,:}, \\ &= \boldsymbol{\mathcal{P}}^1_{:,x_1,:} \cdots \boldsymbol{\mathcal{P}}^k_{:,x_k,:} \Big(\sum_{x_{k+1}} \boldsymbol{\mathcal{P}}^{k+1}_{:,x_{k+1},:} \Big) \cdots \Big(\sum_{x_d} \boldsymbol{\mathcal{P}}^d_{:,x_d,:} \Big), \end{aligned} \tag{26}$$

where $\sum_{x_k} \boldsymbol{\mathcal{P}}^k_{:,x_k,:}$ is the aggregation of all matrix slices within the TT-cores. Consequently, the TT format simplifies the expensive summation into one-dimensional summations.

In cases where identical summation terms recur across multiple conditionals $\Pr_k$, an algorithm known as Tensor Train Conditional Distribution (TT-CD) sampling (Dolgov et al. 2020) can be employed to efficiently obtain samples from $\Pr(\cdot)$.

A.2 Conditional TT

Let assume that $\boldsymbol{x}$ be composed of two parts as $\boldsymbol{x} = (\boldsymbol{y}_1, \boldsymbol{y}_2) \in \Omega_{\boldsymbol{x}}$, where $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ refers to the first d_1 and the last $d - d_1$ variables in $\boldsymbol{y}$, respectively. Our goal is to find $\Pr(\boldsymbol{y}_2|\boldsymbol{y}_1)$ of the TT distribution given in (8).

For a fixed value $\boldsymbol{y}_1 = \boldsymbol{y}_t$, we define a new TT model $\boldsymbol{\mathcal{P}}^{\boldsymbol{y}_t}$ over $\boldsymbol{y}_2$ by slicing the original cores:

$$\boldsymbol{\mathcal{P}}^{\boldsymbol{y}_t}_{\boldsymbol{y}_2} \;=\; \boldsymbol{\mathcal{P}}_{(\boldsymbol{y}_t, \boldsymbol{y}_2)} \quad \forall\, \boldsymbol{y}_2 \in \Omega_{\boldsymbol{y}_2}.$$

Equivalently, its cores become

$$(\boldsymbol{\mathcal{P}}^{\boldsymbol{y}_t})^k \;=\; \begin{cases} \boldsymbol{\mathcal{P}}^k_{:,\, y_{t_k},\, :}, & k = 1, \ldots, d_1, \\ \boldsymbol{\mathcal{P}}^k, & k = d_1 + 1, \ldots, d. \end{cases} \tag{27}$$

The conditional probability then follows as

$$\Pr(\boldsymbol{y}_2 \mid \boldsymbol{y}_1 = \boldsymbol{y}_t) \;=\; \frac{\left|\boldsymbol{\mathcal{P}}^{\boldsymbol{y}_t}_{\boldsymbol{y}_2}\right|}{Z_1} \quad \forall\, \boldsymbol{y}_2 \in \Omega_{\boldsymbol{y}_2}. \tag{28}$$

Finally, one can draw samples of $\boldsymbol{y}_2$ given $\boldsymbol{y}_1 = \boldsymbol{y}_t$.

A.3 Proj-MPPI

For this baseline, we aimed to project each sample into a feasible step before accepting it. Specifically, after sampling an action $\hat{\boldsymbol{u}}^i_t$ for sample i at state $\hat{\boldsymbol{x}}^i_t$, we project the action into $\bar{\boldsymbol{u}}^i_t = \alpha^* \hat{\boldsymbol{u}}^i_t$, where

$$\begin{aligned} \alpha^* &= \underset{\alpha \in [0,1]}{\operatorname{argmax}} \quad \alpha, \\ \text{s.t.} \quad & \boldsymbol{f}(\hat{\boldsymbol{x}}^i_{t+1}, \alpha \hat{\boldsymbol{u}}^i_t) \leq \boldsymbol{0}. \end{aligned} \tag{29}$$

This formulation performs a line search to determine the optimal scaling factor α^*, which identifies the best feasible action in the same direction as the sampled action while ensuring that constraints are satisfied.

A.4 NFs-PoE-MPPI

In this study, we compared our proposed method against normalizing flows (NFs) as a baseline for learning the static distribution. As discussed in the main text, NFs offer an advantage over other generative models by learning an invertible mapping function from a latent space to the primary space. This property enables the mapping of nominal solutions from the primary space to the latent space for sampling and subsequent remapping back to the primary space. However, it was noted in the main text that defining the nominal distribution in the latent space poses challenges.

To address this issue, we adopted an approach similar to what was outlined in (Power and Berenson 2023) for FlowMPPI. Specifically, for half of the samples, we performed standard sampling in the primary space (similar to the MPPI method). For the remaining samples, we initially transformed the nominal mean $\boldsymbol{\mu}^x_{t+h}$ from the task space to the latent space $\boldsymbol{\mu}^z_{t+h}$, assumed a Gaussian distribution around the nominal latent mean, i.e., $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}^z_{t+h}, \boldsymbol{\Sigma}^z_{t+h})$, combined it with the distribution learned using NFs, and then sampled from it. Finally, the samples were mapped back to the primary space. We predefined $\boldsymbol{\Sigma}^z_{t+h}$ to prioritize either the optimality or feasibity distribution based on its value. Given that the latent feasible space distribution has an identity covariance (with zero mean), a value of $\boldsymbol{\Sigma}^z_{t+h}$ greater than 1 in a specific dimension indicates a higher priority for the feasibility distribution, while a value less than 1 prioritizes the optimality distribution. The optimal value for $\boldsymbol{\Sigma}^z_{t+h}$ can vary depending on the task.

NFs were trained using the method described by Dinh et al. (2016). To learn the conditional distribution using NFs, we trained the mapping function for all state and action variables simultaneously, applying an identity transformation to the state variables. We employed the same amount of data as used for TT-SVD, with the parameters detailed in Table 7.

A.5 MPPI on the Sphere Manifold

For the intrinsic method used in Section 5.2, we adopt the exponential map to enable sampling in the tangent space of the sphere, followed by mapping the samples back to the manifold. At each timestep, the tangent space is defined at a point $\boldsymbol{x}_t \in \mathcal{M} = \left\{\boldsymbol{x} \in \mathbb{R}^3 \,\middle|\, \|\boldsymbol{x}\| = r\right\}$ on the sphere. Any valid action $\boldsymbol{u}_t \in \mathbb{R}^3$ must lie in this tangent space, and thus satisfy the orthogonality condition $\boldsymbol{x}_t^\top \boldsymbol{u}_t = 0$.

This tangent space is a 2D subspace of $\mathbb{R}^3$, and we generate actions $\boldsymbol{u}_t$ by first sampling in this lower-dimensional space: $\boldsymbol{u}^{2D}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. The mapping from the 2D tangent coordinates to the 3D ambient space is defined as

$$\boldsymbol{u}_t = \boldsymbol{T}(\boldsymbol{x}_t) \boldsymbol{u}^{2D}_t, \tag{30}$$

where the transformation matrix $\boldsymbol{T}(\boldsymbol{x}_t) = [\boldsymbol{e}_1^\top(\boldsymbol{x}_t), \boldsymbol{e}_2^\top(\boldsymbol{x}_t)]^\top$ is constructed from an orthonormal basis of the tangent space at $\boldsymbol{x}_t$

$$\boldsymbol{e}_1 = \frac{\boldsymbol{l} \times \boldsymbol{x}_t}{\|\boldsymbol{l} \times \boldsymbol{x}_t\|}, \quad \boldsymbol{e}_2 = \frac{\boldsymbol{e}_1 \times \boldsymbol{x}_t}{\|\boldsymbol{e}_1 \times \boldsymbol{x}_t\|}. \tag{31}$$

Here, $\boldsymbol{l}$ is an arbitrary vector chosen to avoid alignment with $\boldsymbol{x}_t$, ensuring a well-defined cross product

$$\boldsymbol{l} = \begin{cases} [1, 0, 0]^\top & \text{if } |\boldsymbol{x}_t^\top [1, 0, 0]^\top| \leq 0.9, \\ [0, 1, 0]^\top & \text{otherwise.} \end{cases} \tag{32}$$

After sampling actions and mapping them to the task space, the next state $\boldsymbol{x}_{t+1}$ is computed using the exponential map on the sphere

$$\boldsymbol{x}_{t+1} = \exp_{\boldsymbol{x}_t}(\boldsymbol{v}) = \cos\left(\frac{\|\boldsymbol{v}\|}{r}\right) \boldsymbol{x}_t + \sin\left(\frac{\|\boldsymbol{v}\|}{r}\right) \frac{r\, \boldsymbol{v}}{\|\boldsymbol{v}\|}, \tag{33}$$

where $\boldsymbol{v} = \boldsymbol{u}_t \cdot dt$ is the scaled action vector, and r is the radius of the sphere. This formulation ensures that the resulting state $\boldsymbol{x}_{t+1}$ remains on the sphere manifold.

A.6 DIAL-mpc and TT-PoE-DIAL

Inspired by the success of diffusion-based methods across various domains, Pan et al. (2024) proposed a novel MPPI variant for trajectory optimization. In their approach, each

Table 7. Hyperparameters employed for learning the feasibility distribution using both tensor train and normalizing flow methods across various tasks. Please note that the obstacle avoidance task refers to both PNGRID and the online obstacle avoidance task.

	Param.	Obstacle Avoidance	Push. Cyl.	Push. Box	Push. Mus.	Sphere man.	Sin. man.	Whole-body
TT	# State Discretization	100	10	10	10	25	25	50
	# Action Discretization	20	10	10	10	10	10	50
	State Refinement Scale	1	20	10	10	10	10	1
	Action Refinement Scale	10	20	10	10	10	10	1
NFs	# Coupling Layers	6	8	8	8	8	6	-
	Size of layers	256	256	256	256	256	256	-
	Batch size	4096	4096	8192	8192	8192	8192	-
	learning rate	0.0004	0.0002	0.0001	0.0001	0.0002	0.0004	-

Algorithm 2 TT-PoE-DIAL

Require: $\boldsymbol{x}_t, \boldsymbol{\mu}_{t,\dots,t+H-1}, \boldsymbol{\sigma}_{t,\dots,t+H-1}$
Return: $\boldsymbol{u}_t^*$
1: $\hat{\boldsymbol{x}}_t \leftarrow \boldsymbol{x}_t$
2: **for** n from 0 to N_{diffuse} **do**
3: **for** h from 0 to $H-1$ **do**
4: Update the covariance matrix ▷ Eq. (34)
5: Calculate $\boldsymbol{\mathcal{P}}_{\mathcal{N}_m}$ ▷ Eq. (21)
6: $\boldsymbol{\mathcal{P}}_h \leftarrow$Product of the distributions ▷ Eq. (20)
7: $\hat{\boldsymbol{u}}_{t+h}^0, \dots, \hat{\boldsymbol{u}}_{t+h}^N \leftarrow$Sample N actions
8: $\hat{\boldsymbol{x}}_{t+h+1}^0, \dots, \hat{\boldsymbol{x}}_{t+h+1}^N \leftarrow$Forward Dynamics
9: **end for**
10: $c^0, \dots, c^N \leftarrow$Calculate cost values
11: $\boldsymbol{\mu}_t \leftarrow$ update the mean value ▷ Eq. (4)
12: **end for**
13: $\boldsymbol{u}_t^* \leftarrow \boldsymbol{\mu}_t$

MPPI iteration is repeated N_{diffuse} times using different noise realizations. Their results (which we also observe in our own experiments) demonstrate that this method significantly outperforms vanilla MPPI and several other baselines. Building on this idea, Xue et al. (2025) further extended the method to a full MPC framework.

In their formulation, the noise covariance at each diffusion step is defined as

$$\boldsymbol{\Sigma}_{t+h}^n = \exp\left(-\frac{N_{\text{diffuse}}-n}{\beta_1 N_{\text{diffuse}}} - \frac{H-h}{\beta_2 H}\right)\boldsymbol{I}, \tag{34}$$

where β_1 and β_2 are temperature parameters controlling the diffusion process and the time horizon, respectively. For simplicity, we define two decay factors: $f_{\text{denoise}} = \exp\left(-\frac{1}{\beta_1 N_{\text{diffuse}}}\right)$ and $f_{\text{horizon}} = \exp\left(-\frac{1}{\beta_2 H}\right)$.

Algorithm 2 outlines how this diffusion-based MPPI method can be integrated into the TT-PoE pipeline. The only modifications required are the addition of lines 5 and 6, and replacing Gaussian sampling with the TT-based sampling approach described in Equation (20).

A.7 Experiments

Different settings used for different methods in various tasks are listed in Table 8. These variables include parameters like the length of the planning horizon H, the covariance matrices in the task space $\boldsymbol{\Sigma}_{\text{task}}$, the covariance matrix $\boldsymbol{\Sigma}_z$ for the latent space, the temperature variable β used in MPPI, the simulation time step dt, the maximum velocity $u_{\max}$, and the state limits $x_{\max}$. Task success is defined by meeting predefined criteria: completion within a specified number of steps $T_{\text{succ.}}$ and achieving a cost value below a predetermined threshold $c_{\text{succ.}}$. It is important to note that for the MPPI method, action commands are manually clipped to stay within feasible ranges while other methods handle this internally.

Upon computing the cost value for sample i, we proceed to normalize them using the minimum observed cost value among the samples, expressed as

$$c^i = \frac{\hat{c}^i}{c_{\min}}, \quad \text{s.t.} \quad c_{\min} = \min(\hat{c}^0, \dots, \hat{c}^N), \tag{35}$$

where the notation $\hat{(\cdot)}$ denotes the values of the cost functions before normalization. Additionally, we incorporate an extra step by assuming one of the samples to have zero action. This facilitates the system to halt in case all samples become infeasible. Further explanation regarding the employed cost functions in these experiments is provided in the following.

A.7.1 Obstacle avoidance: In these tasks (i.e., PNGRID and the online obstacle avoidance), our objective is to navigate towards a designated target point $\boldsymbol{p}_{\text{target}}$, while navigating around obstacles and staying within the environmental boundary. For addressing this task, we employed a single integrator point mass as our agent, resulting in a 4-dimensional state-action space. The TT-distribution for this function is depicted in Fig. 7-c for zero velocity. Notably, the region without obstacles exhibits a higher probability value compared to other areas. The occurrence of negative probabilities in some points is attributed to the imperfections in the TT-SVD approximation. While this could be mitigated by increasing the maximum rank value, which is set to 300 for this task, we observed satisfactory results with this configuration. The distribution is determined by discretizing the state space into 100 partitions and the action space into 20 partitions. However, the partition in the TT-cores corresponding to the action commands undergoes an additional refinement by interpolating between different nodes resulting in 10 times more partition. The samples generated by the model with zero velocity are illustrated in Fig. 7-d and can be contrasted with the samples obtained from a Gaussian mixture model (GMM) employing $K = 400$ Gaussian components, as shown in Fig. 7-b. The GMM model is trained with an EM algorithm using the Scikit-learn library (Pedregosa et al. 2011) in Python.

Table 8. Hyperparameters employed with different controllers. The obstacle avoidance task refers to both PNGRID and the online obstacle avoidance task.

	Obstacle Avoidance	Push. Cyl.	Push. Box	Push. Must.	Sphere man.	Sinus man.	Whole-body
H	15	32	20	20	50	30	15
$\boldsymbol{\Sigma}_{\text{task}}$	$0.125\boldsymbol{I}$	$0.3\boldsymbol{I}$	$0.05\boldsymbol{I}$	$0.05\boldsymbol{I}$	$0.1\boldsymbol{I}$	$0.05\boldsymbol{I}$	$0.05\boldsymbol{I}$
$\boldsymbol{\Sigma}_z$	$10\boldsymbol{I}$	$0.5\boldsymbol{I}$	$0.01\boldsymbol{I}$	$0.05\boldsymbol{I}$	$0.2\boldsymbol{I}$	$10\boldsymbol{I}$	-
β	0.05	0.05	0.05	0.05	0.05	0.05	0.01
$dt[s]$	0.1	0.05	0.05	0.05	0.1	0.1	0.1
$u_{\max}[\frac{m}{s}]$	1	1	1	1	1	1	Real Limits
$x_{\max}[m]$	1.25	1	1	0.4	0.25	0.5	Real Limits
$T_{\text{succ.}}$	100	500	500	400	400	200	25
$c_{\text{succ.}}$	10^{30}	10^4	10^4	10^3	3×10^5	10^{30}	10^6
N_{diffuse}	-	-	-	-	-	-	3
$\text{f}_{\text{denoise}}$	-	-	-	-	-	-	0.22
$\text{f}_{\text{horizon}}$	-	-	-	-	-	-	0.95

To assess the efficacy of a sample, we employ a performance evaluation as

$$\hat{c}_{\text{Obst. Avoid.}}(\hat{\boldsymbol{U}}_t) = \sum_{h=0}^{H-1} r_{t+h}\Big(10\ \|\hat{\boldsymbol{x}}_{t+h} - \boldsymbol{p}_{\text{target}}\|^2 + 10^{30}\ \text{coll}(\hat{\boldsymbol{x}}_{t+h}) + 10^{-3}\ \|\hat{\boldsymbol{u}}_{t+h}\|^2\Big) + 10^3\ r_{t+H}\|\hat{\boldsymbol{x}}_{t+H} - \boldsymbol{p}_{\text{target}}\|^2 \quad (36)$$

where $\text{coll}(\cdot)$ denotes a function that returns 1 if the point collides with an obstacle or exceeds the feasible area, and 0 otherwise. During the planning and learning phases of the feasibility distribution for the PNGRID task, we consider a 5-centimeter margin around obstacles to define collisions. In (36), r_{t+h} serves as an indicator to determine whether the agent has previously reached the target by time step $t+h$ or not. This mechanism prevents the controller from solely aiming for the target at the final time step. We consider the target as reached if the distance between the agent and the target point falls below 5 centimeters.

For Expert 2, we have utilized the following feasibility distribution

$$p_{\text{feas}}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \ = \ \mathbf{1}\big\{\hat{\boldsymbol{x}}_t \in \mathcal{X},\ \hat{\boldsymbol{x}}_{t+1} \in \mathcal{X},\ (\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_{t+1}) \cap \mathcal{O} = \emptyset\big\},$$

where, $\mathcal{X}$ denotes the allowable state-space (inside the boundaries), $\mathcal{O}$ the obstacle set, and $\mathbf{1}\{\cdot\}$ is the indicator function.

For the online obstacle avoidance task, we utilized the same values without accounting for any collision margins. The only additional modification was that the obstacle became detectable to the solver when the robot was within 40 centimeters horizontally from the center of obstacles, which had radii of up to 40 centimeters, selected randomly.

A.7.2 Planar-pushing: We have modeled the robot end-effector as a 2D point mass under the control of a velocity controller. For each object, we have utilized distinct parameters, detailed in Table 8, and is subjected to unique cost functions, elaborated upon subsequently. For Expert 2, however, all objects share the same feasibility distribution

$$p_{\text{feas}}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) = \mathbf{1}\Big\{\hat{\boldsymbol{x}}^{\text{obj.}} \in \mathcal{X} \wedge \Big(\Delta_o > 0\ \vee\ \Delta_d > 0\Big)\Big\},$$

where

$$\Delta_o = \|\hat{\boldsymbol{x}}^{\text{obj.}}_{t+1} - \hat{\boldsymbol{x}}^{\text{obj.}}_t\|,$$
$$\Delta_d = \|\hat{\boldsymbol{x}}^{\text{rob.}}_t - \hat{\boldsymbol{x}}^{\text{obj.}}_t\| - \|\hat{\boldsymbol{x}}^{\text{rob.}}_{t+1} - \hat{\boldsymbol{x}}^{\text{obj.}}_{t+1}\|.$$

Here, $\hat{\boldsymbol{x}}^{\text{obj}}$ and $\hat{\boldsymbol{x}}^{\text{rob}}$ denote the object and robot states, respectively. This distribution assigns high probability only to samples in which the object remains within the feasible region and either (i) the object moves ($\Delta_o > 0$) or (ii) the robot–object distance decreases ($\Delta_d > 0$).

Cylinder: The cost function designed for achieving a target point $\boldsymbol{p}_{\text{target}}$ in this task is expressed as

$$\hat{c}_{\text{cyl.}}(\hat{\boldsymbol{U}}_t) = \sum_{h=0}^{H-1} r_{t+h+1}\Big(10\ \|\hat{\boldsymbol{x}}^{\text{obj}}_{t+h+1} - \boldsymbol{p}_{\text{target}}\|^2 + 10^{-2}\ \text{coll_o}(\hat{\boldsymbol{x}}^{\text{obj}}_{t+h+1}) + 10\ \text{coll_b}(\hat{\boldsymbol{x}}^{\text{obj}}_{t+h+1}) + \text{coll_b}(\hat{\boldsymbol{x}}^{\text{rob}}_{t+h+1}) + 10^{-1}\ \text{SDF}^2(\hat{\boldsymbol{x}}^{\text{obj}}_{t+h+1}, \hat{\boldsymbol{x}}^{\text{rob}}_{t+h+1})\Big), \quad (37)$$

where functions $\text{coll_o}(\cdot)$ and $\text{coll_b}(\cdot)$ determine collision with the obstacles and the boundary, respectively, returning 1 in case of collision and 0 otherwise. Additionally, $\text{SDF}(\cdot)$ calculates the signed distance value between the robot and the agent. Similarly to the obstacle avoidance experiment, r_{t+h} serves as an indicator to verify whether the agent reached the target before the time step $t+h$. We define the target as reached when the distance between the agent and the target point is less than 5 centimeters.

Pushing a Mustard Bottle and a Box: We use the same cost function for these two tasks which aimed at reaching a target position $\boldsymbol{p}_{\text{target}}$ and desired yaw angle o^{d} as

$$\hat{c}_{\text{mus./Box}}(\hat{\boldsymbol{U}}_t) = \sum_{h=0}^{H-1} r_{t+h+1}\Big(10\ \|\hat{\boldsymbol{x}}^{\text{obj}}_{t+h+1} - \boldsymbol{p}_{\text{target}}\|^2 + (\hat{o}_{t+h} - o^d)^2 + 10^5\ e^{-\frac{h}{4}}\text{coll_b}(\hat{\boldsymbol{x}}^{\text{obj}}_{t+h+1}) + 10^5\ e^{-\frac{h}{4}}\text{coll_b}(\hat{\boldsymbol{x}}^{\text{rob}}_{t+h+1}) + 10\ \|\boldsymbol{x}^{\text{rob}}_{t+h+1} - \boldsymbol{x}^{\text{obj}}_{t+h+1}\|^2\Big) + 10^3\ r_{t+H}\Big(\|\hat{\boldsymbol{x}}^{\text{obj}}_{t+H} - \boldsymbol{p}_{\text{target}}\|^2 + (\hat{o}_{t+h} - o^d)^2\Big) + s\ \text{Ang_rel}(\hat{\boldsymbol{x}}^{\text{obj}}_t, \hat{\boldsymbol{x}}^{\text{rob}}_t, \hat{\boldsymbol{x}}^{\text{rob}}_{t+H}, \theta^d), \quad (38)$$

where $\hat{o}$ denotes the orientation of the object, and the remaining notation is the same as in the previous task.

Additionally, for boundary cost considerations, a margin of 10 and 16 centimeters is assumed during the training and planning phases, respectively. The key distinction between (38) and (37) lies in the final term, which evaluates the degree of rotation of the robot around the object compared to its initial position. Due to potential issues arising from the object geometry causing the robot to become stuck on one side (i.e., $s = 1$), this term is introduced to encourage the robot to rotate around the object by approximately $\theta^d = \pm\frac{\pi}{2}$ when such circumstances occur. The value of s is determined by

$$s = \begin{cases} 1, & \text{if } \exp\big(-|\text{dist}^{\min}_{t+H} - \text{dist}_t|\big) > 0.96 \\ 0, & \text{otherwise} \end{cases},$$

where dist_{t+h} represents the distance of the object to the target (disregarding orientation) at time step $t+h$, and the superscript $\min$ indicates the minimum distance among the samples. It is important to note that this cost is applied to all samples, penalizing instances where the system remains on one side of the object.

In this experiment, the task is deemed complete if the object is within a distance of less than 10 centimeters from the target, and the orientation difference from the desired angle is less than 0.2 radians.

A.7.3 Tube Tracking: In this task, our objective is to control the robot end-effector's position within a predefined area. In the sphere manifold scenario, the area is bounded by two spheres: the inner sphere has a radius of 15 cm, while the outer sphere has a radius of 20 cm. For the sinusoidal wave, the area is defined as the region inside $\{(x, y, z) \mid z \in 0.1\sin(5y\pi) \pm 0.03, \quad x \in [0.2, 1.0], \quad y \in [-0.5, 0.5]\}$. During both the learning and planning phases, collision detection incorporates a 1-cm margin to ensure a safety buffer.

The cost function designed for this task is expressed as

$$\hat{c}_{\text{Manifold}}(\hat{\boldsymbol{U}}_t) = \sum_{h=0}^{H-1} r_{t+h}\Big(10^2\ \|\hat{\boldsymbol{x}}_{t+h} - \boldsymbol{p}_{\text{target}}\|^2 + 10^4\ \text{coll}(\hat{\boldsymbol{x}}_{t+h}) + 10^{-3}\ \|\hat{\boldsymbol{u}}_{t+h}\|^2\Big) + 10^2\ r_{t+H}\|\hat{\boldsymbol{x}}_{t+H} - \boldsymbol{p}_{\text{target}}\|^2, \quad (39)$$

where the variables and terms retain the same definitions as previously described.

For Expert 2, we employ the following feasibility distribution

$$p_{\text{feas}}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) = \mathbf{1}\Big\{\hat{\boldsymbol{x}}_t \in \mathcal{X},\ \hat{\boldsymbol{x}}_{t+1} \in \mathcal{X},\ \hat{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_{t+1} \in \mathcal{M}_\epsilon\Big\},$$

where

$$\mathcal{M}_\epsilon = \{\,\boldsymbol{x} \mid \text{dist}(\boldsymbol{x}, \mathcal{M}) \le \epsilon\}$$

is an ϵ-margin around the manifold $\mathcal{M}$. This ensures that both the current and next states lie within the admissible state space $\mathcal{X}$ and remain close to $\mathcal{M}$ within the specified tolerance.

A.7.4 Whole-body Obstacle Avoidance: In this experiment, we control a Franka Emika robot in the presence of randomly placed spherical obstacles. The optimality distribution cost is

$$\hat{c}_{\text{wb}}(\hat{\boldsymbol{U}}_t) = \sum_{h=0}^{H-1} r_{t+h}\Big(10\,\|\hat{\boldsymbol{x}}_{t+h} - \boldsymbol{p}_{\text{target}}\|^2 + 10^6\,\text{coll_b}(\hat{\boldsymbol{x}}_{t+h}) + 10^6\,\exp\big(-\tfrac{\text{dist}(\hat{\boldsymbol{x}}_{t+h})}{0.01}\big) + 10\,\|\hat{\boldsymbol{u}}_{t+h}\|^2\Big) + 10^3\,r_{t+H}\|\hat{\boldsymbol{x}}_{t+H} - \boldsymbol{p}_{\text{target}}\|^2, \quad (40)$$

where all quantities are in joint space, $\text{dist}(\cdot)$ is computed via the Robot Distance Field (Li et al. 2024), and obstacles are five spheres of radius $0.2\,\text{m}$.

The feasibility expert uses a soft-penalty distribution:

$$p_{\text{feas}}\big(\boldsymbol{x}^o_{1:N_o},\, \boldsymbol{x}_t,\, \boldsymbol{u}_t\big) = \prod_{\tau\in\{t,t+1\}} \Big(1 - \exp\big(-\tfrac{\text{dist}(\boldsymbol{x}_\tau)}{\lambda}\big)\Big),$$

where $\lambda = 0.5$. This definition assigns low probability to states near obstacles, ensuring both the current and next robot configurations remain safely distant from all obstacles.

A.8 Real-robot Experiment

For the real robot experiment, we adapted the pipeline initially developed for simulating the pushing of a mustard bottle. However, we made an adjustment by reducing the number of horizons to 10. As a safety precaution, we constrained the robot's speed to one-twentieth of its simulated counterpart, with $u_{\max} = 0.05$ and $\boldsymbol{u}^{\text{real}} = \frac{\boldsymbol{u}^{\text{sim}}}{20}$. Additionally, each command is executed 20 times, pushing the object to the same target point as planned. After each set of executions, the robot refreshed its action commands and iterated the process until reaching the target.

In this experiment, we employed a 7-axis Franka robot, which we controlled using a Cartesian velocity controller operating at 1000 Hz. For the visual feedback, we utilized a RealSense D435 camera, operating at 30 Hz. Detailed results and observations from these experiments are available in the accompanying videos.