MMP++: Motion Manifold Primitives With Parametric Curve Models

Yonghyeon Lee, *Member, IEEE*

***Abstract*—Motion manifold primitives (MMP), a manifold-based approach for encoding basic motion skills, can produce diverse trajectories, enabling the system to adapt to unseen constraints. Nonetheless, we argue that current MMP models lack crucial functionalities of movement primitives, such as temporal and via-points modulation, found in traditional approaches. This shortfall primarily stems from MMP's reliance on discrete-time trajectories. To overcome these limitations, we introduce motion manifold primitives++ (MMP++), a new model that integrates the strengths of both MMP and traditional methods by incorporating parametric curve representations into the MMP framework. Furthermore, we identify a significant challenge with MMP++: performance degradation due to geometric distortions in the latent space, meaning that similar motions are not closely positioned. To address this, isometric motion manifold primitives++ (IMMP++) is proposed to ensure the latent space accurately preserves the manifold's geometry. Our experimental results across various applications, including two-DoF planar motions, seven-DoF robot arm motions, and SE(3) trajectory planning, show that MMP++ and IMMP++ outperform existing methods in trajectory generation tasks, achieving substantial improvements in some cases. Moreover, they enable the modulation of latent coordinates and via-points, thereby allowing efficient online adaptation to dynamic environments.**

***Index Terms*—Autoencoders, isometric representation learning, manifold, movement primitives, Riemannian geometry.**

I. Introduction

DEVELOPING "good" mathematical models for representing basic motion skills continues to be a central focus in the literature on learning from demonstration [1], [2], [3]. In this article, we adopt the view that a good model should be capable of generating diverse trajectories that can complete the given task. Moreover, it should be easily adaptable to a new, unseen constraint. For instance, if an unseen obstacle blocks the initially planned path, the model should enable a robot to avoid that obstacle while still accomplishing the task. We aim to train such a model using multiple demonstration trajectories. Challenges often arise from the small dataset size, high dimensionality of the trajectory data, and the multimodality of data distribution.

Manuscript received 26 March 2024; revised 28 July 2024; accepted 7 August 2024. Date of publication 15 August 2024; date of current version 28 August 2024. The work was supported by CAINS through the Center for AI and Natural Sciences at Korea Institute for Advanced Study, KIAS Individual under Grant AP092701. This article was recommended for publication by Associate Editor S. Song and Editor J. Kober upon evaluation of the reviewers' comments.

The author is with the Center for AI and Natural Sciences (CAINS), Korea Institute for Advanced Study (KIAS), Seoul 02455, South Korea (e-mail: ylee@kias.re.kr).

Digital Object Identifier 10.1109/TRO.2024.3444068

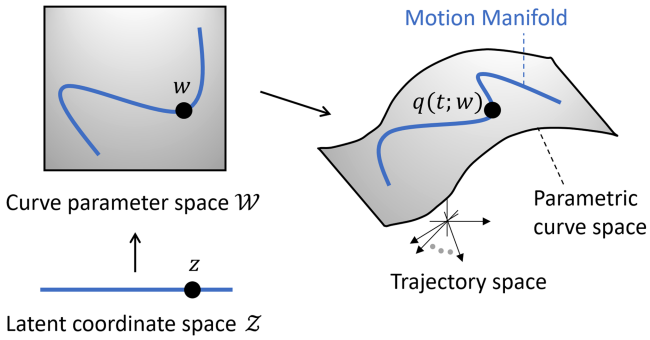

Fig. 1. *MMP++:* A latent coordinate space $\mathcal{Z}$ is mapped to a subspace of the curve parameter space $\mathcal{W}$; the parameter space $\mathcal{W}$ is mapped to a subspace of the infinite-dimensional trajectory space. The motion manifold and parametric curve space are visualized as a curve and surface, not because their actual dimensions are one and two, but only to indicate the relative size relationships of their dimensions.

Adopting the motion manifold hypothesis [4], [5]—which assumes that a set of high-dimensional trajectory data lies on some lower dimensional manifold—recent motion manifold primitives (MMP) framework provides motion primitive models that can encode and generate, for a given task, a continuous manifold of trajectories each of which is capable of accomplishing the task [6], [7]. This framework has demonstrated promising results in addressing the aforementioned challenges, effectively reducing the data dimensionality and capturing multimodality. In particular, adjusting the low-dimensional latent coordinate values enables the adaptation of trajectories to unseen environments.

These manifold-based models, however, lack some of the desired functionalities of movement primitives found in other conventional methods, such as DMP [8], [9], ProMP [10], and VMP [11]. These functionalities include: 1) temporal modulation to enable faster or slower execution of the movement and 2) modulation of via-points (e.g., start and goal points) given new task constraints. The fundamental reason for the absence of such functions in the MMP framework is its reliance on discrete-time trajectory representations. In contrast, conventional movement primitives often employ parametric models for trajectory representation. For example, one of the simplest forms of these models is the linear basis function model, for a configuration space $\mathcal{Q} = \mathbb{R}^n$, expressed as follows:

$$q(\tau; w) = \sum_{i=1}^{B} \phi_i(\tau) w_i \quad \text{for} \quad \tau \in [0, 1] \tag{1}$$

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

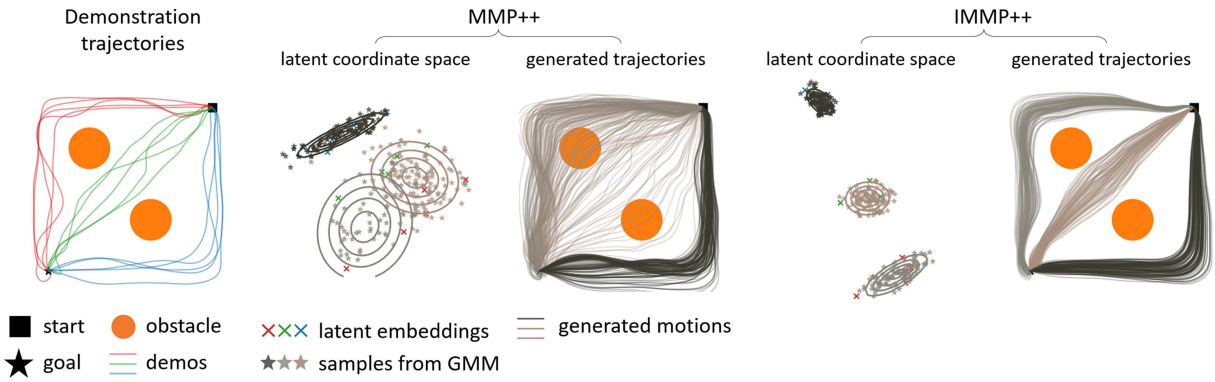

Fig. 2. *Left:* There are 15 demonstration trajectories (red, green, and blue trajectories) that travel from the start to the goal, avoiding the obstacle. *Middle and Right:* MMP++ and IMMP++ learn 2-D manifolds in the curve parameter space and produce 2-D latent coordinate spaces. Latent values of the demonstration trajectories are visualized in the latent coordinate spaces, marked as $\times$. GMMs of three components are fitted in the latent spaces, and the sampled points are visualized as stars $*$. The corresponding generated trajectories are also visualized.

where $\{\phi_i(\tau)\}_{i=1}^{B}$ is a set of some scalar-valued basis functions in [0,1] and $w_i \in \mathbb{R}^n, i = 1, \ldots, B$ are curve parameters. Temporal modulation can be achieved by modifying τ as a function of time t, and adding some structures to the basis function $\phi_i(\tau)$ can enforce constraint satisfaction [e.g., $\phi_i(\tau) := \tau(1-\tau)b_i(\tau)$ enforces $\phi_i(0) = \phi_i(1) = 0$ for any function $b_i(\tau)$].

In this article, we propose applying the MMP framework to the parametric curve representations of trajectories, thus simultaneously tackling the challenge of dimensionality and achieving the desired functionalities, denoted as motion manifold primitives++ (MMP++) (see Fig. 1). In addition, parametric curve representations in MMP++ lead to several other advantages. First, motions have bounded accelerations and jerks, avoiding sudden and abrupt changes. Second, the dimension of the parametric curve space—which is equal to the dimension of the parameter space—is generally much smaller than the dimension of the discrete-time trajectory data space, reducing the complexity of the subsequent motion manifold learning problem. The vanilla MMP++, a naive application of the MMP framework to the curve parameter space, however, sometimes results in a *geometrically distorted* latent coordinate space—where similar motion data are not positioned close to each other—leading to a generation of motions that violate the task constraint. For example, consider an example shown in Fig. 2. We learn a 2-D manifold and its latent coordinate space, by using the red, green, and blue demonstration trajectories and their parametric curve representations. Then, we fit a Gaussian mixture model (GMM) using the latent values of the trajectories. As illustrated in Fig. 2 (*middle*), due to the geometric distortion in the latent coordinate space, the same color trajectories are not located close enough to each other. Consequently, the red and green latent points are not correctly clustered by the GMM, and many of the generated motions collide with the obstacle, failing to accomplish the task.

In this article, adopting the isometric regularization method from [12], we propose to learn *a geometry-preserving latent coordinate space*, so that similar trajectories can be located nearby in the latent space. To employ this method in our context, we need to specify a *Riemannian metric* for the curve parameter space that serves as the basis for determining the notion of closeness in the parameter space. We propose a *CurveGeom Riemannian metric* for the curve parameter space that reflects the geometry of the trajectory space, given a parametric curve model that satisfies some mild regularity conditions. We call this framework *isometric motion manifold primitives++ (IMMP++)*; see Fig. 2 (*right*).

In the first part of our experiments, we focus on Euclidean configuration space cases and use affine curve models, similar to those in ProMP [10] and VMP [11]. This induces constant CurveGeom metrics and leads to simpler implementations of the isometric regularization. Experiments involving two-DoF planar obstacle-avoiding motions and seven-DoF collision-free motions of a robot arm confirm that our manifold-based methods, MMP++ and IMMP++, outperform conventional movement primitives in trajectory generation. Notably, we verify that the modulation of latent coordinates and via-points leads to diverse trajectory generation and enables the online adaptation of trajectories in dynamic environments. In the second part, we demonstrate how to extend our framework to SE(3) trajectory data with the water-pouring demonstration trajectory dataset.

II. Related Works

A. Movement Primitives

Movement Primitives are mathematical models that encode and generate motions or trajectories. Conventional methods for movement primitives can be roughly divided into two categories as follows. 1) Dynamical system-based approaches, such as dynamic movement primitives [8], [9], [13], [14], [15], [16] and stable dynamical systems [17], [18], [19], [20], [21], [22]. 2) Parametric or nonparametric probabilistic modeling of trajectories [10], [11], [23]. Each of these models possesses its own characteristics. Dynamical system-based methods typically ensure the stability of the resulting closed-loop systems. Approaches based on stable autonomous dynamical systems provide temporal and spatial robustness to perturbations. Parametric curve models offer adaptability in terms of temporal modulation and changes in constraints (e.g., goal points).

A primary challenge in most of these methods is the limited adaptability to diverse situations (e.g., when unforeseen obstacles or new constraints emerge), which is mainly due to

their design for encoding and producing a single trajectory for a given task [7]. This becomes problematic when this trajectory becomes infeasible by unforeseen environmental changes, such as the sudden appearance of an obstacle. While dynamical system-based approaches can incorporate mechanisms, such as obstacle avoidance potential functions [24], [25], [26], [27], these adaptations may inadvertently breach other task-related constraints. Therefore, for motion primitives to be truly adaptable, a strategy that can encode various trajectories for the same task is critical. Our work adopts the MMP framework [6], [7] to encode and generate diverse trajectories or even a continuous manifold of trajectories, producing highly adaptable primitives, simultaneously inheriting advantages of the parametric curve models utilized in [10] and [11].

B. Manifold-Based Movement Primitives

Recently, manifold-based representations of basic motion skills have shown promising results in encoding diverse motions and producing adaptable primitives. These approaches can be categorized into two types. The first approach attempts to learn a submanifold in the configuration space $\mathcal{Q}$, a manifold of configurations, where the latent value z maps to a configuration in $\mathcal{Q}$ [28]. To generate a trajectory, this approach computes a geodesic connecting two points in $\mathcal{M}$.

The second type learns a low-dimensional manifold of trajectories, referred to as a motion manifold, where each point in the latent space z corresponds to a trajectory $q(t) \in \mathcal{Q}$ [6], [7]. Our method extends the latter approach. While existing methods map the latent point z to a discrete-time trajectory representation $(q_1, \ldots, q_T)$, we use parametric curve models $q(t, w)$, in which the latent point z is mapped to the curve parameter w. As a result, our extended version can also be interpreted as learning a submanifold in $\mathcal{Q}$, similar to the first type approach, since (t, z) is now mapped to a point $q(t, w(z))$ in $\mathcal{Q}$.

MMP can produce diverse trajectories, and their ability to adapt to unseen obstacles by finding a latent value that generates a collision-free trajectory has been verified [7]. However, due to the discrete nature of trajectory representation, they have limited adaptability to a dynamically changing environment. In our extended version, trajectories are parameterized by time, enabling online adaptation to dynamic environments.

C. Manifold Learning and Latent Space Distortion

An autoencoder framework and its variants have received a lot of attention as effective methods to learn the manifold and its coordinate chart simultaneously, including but not limited to [12], [29], [30], [31], [32], [33], [34], [35], [36], and [37]. Of particular relevance to this article, a geometric perspective on autoencoders has been eloquently presented in [5]. In this article, a significant aspect of concern is the presence of the geometric distortion within the latent space of the autoencoder, as highlighted in [4], [5], [12], [34], and [38]. A recent regularization method [12] has developed a method to find the one that minimizes the geometric distortion, i.e., preserves the geometry of the data manifold and the latent coordinate space. While Euclidean metric is assumed in [12], in this work, we propose to use a pullback Riemannian metric for the curve parameter space that reflects the geometry of the curve space.

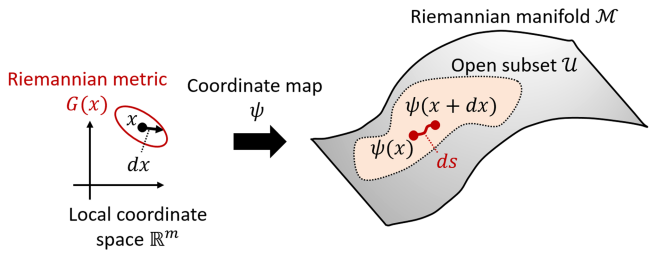

Fig. 3. Local coordinate system for an m-dimensional Riemannian manifold $\mathcal{M}$. The Riemannian metric at coordinates x, $G(x)$, is visualized as a red equidistant ellipse that is $\{y \in \mathbb{R}^m \mid (y-x)^T G(x)(y-x) = \text{constant}\}$.

III. Geometric Preliminaries

In this section, we review some basic concepts in differential geometry that serve as cornerstones for our method. We refer to standard differential geometry textbooks for more details [39], [40].

A. Riemannian Manifolds

A smooth manifold $\mathcal{M}$ equipped with a positive-definite inner product on the tangent space at each point is called a *Riemannian manifold*, and the family of inner products is called a *Riemannian metric*. Given an m-dimensional Riemannian manifold $\mathcal{M}$ and its local coordinates $x \in \mathbb{R}^m$—when using local coordinates, we implicitly assume there exists a local coordinate map $\psi : \mathbb{R}^m \to \mathcal{U} \subset \mathcal{M}$—the Riemannian metric at x can be expressed as an $m \times m$ positive-definite matrix denoted by $G(x) \in \mathbb{R}^{m \times m}$; see Fig. 3. This defines several geometric notions on $\mathcal{M}$, such as lengths, angles, and volumes. For example, given an infinitesimal displacement vector $dx \in \mathbb{R}^m$, its squared length is defined as follows:

$$ds^2 = dx^T G(x) dx = \sum_{i,j=1}^{m} g_{ij}(x) dx^i dx^j \tag{2}$$

where $\{g_{ij}(x)\}$ is an index notation of the matrix $G(x)$ and $dx = (dx^1, \ldots, dx^m)$.

B. Immersion and Embedding

Consider two differentiable manifolds, an m-dimensional manifold $\mathcal{M}$ and n-dimensional manifold $\mathcal{N}$, with their respective coordinates $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$. A differentiable mapping $\mathrm{f} : \mathcal{M} \to \mathcal{N}$ is an *immersion* if its differential is an injective function at every point in $\mathcal{M}$. Representing the mapping in local coordinates as $f : \mathbb{R}^m \to \mathbb{R}^n$, equivalently, f is an immersion if its Jacobian matrix

$$J(x) := \frac{\partial f}{\partial x}(x) \in \mathbb{R}^{n \times m} \tag{3}$$

has constant rank equal to $\dim(\mathcal{M}) = m$ at every point. Intuitively, for f to be an immersion, the output manifold dimension n must be greater or equal to the dimension of $\mathcal{M}$. A smooth *embedding* is an injective immersion $\mathrm{f} : \mathcal{M} \to \mathcal{N}$ such that $\mathcal{M}$ is

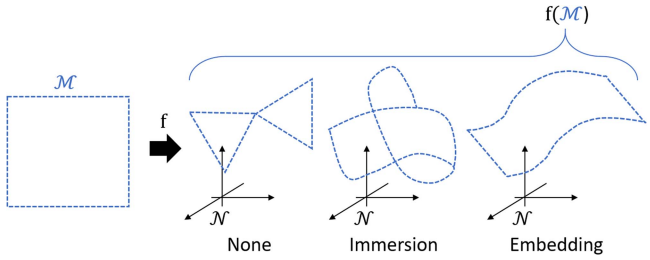

Fig. 4. Illustration of immersion and embedding between two manifolds $\mathcal{M}$ and $\mathcal{N}$.

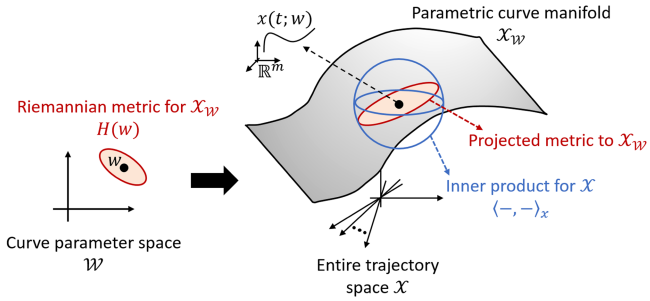

Fig. 5. Illustration of Riemannian geometry of the parametric curve manifold $\mathcal{X}_\mathcal{W}$.

diffeomorphic to its image $\mathrm{f}(\mathcal{M}) \subset \mathcal{N}$.[1] Specifically, when the domain manifold is compact, a smooth embedding is equivalent to an injective immersion. Then, $\mathrm{f}(\mathcal{M})$ is called an *embedded manifold* in $\mathcal{N}$; see Fig. 4.

C. Riemannian Geometry of Parametric Curves

In this article, we are particularly interested in Riemannian geometry of manifold of parametric curves. This section introduces how to define a Riemannian metric for the parametric curve manifold; see Fig. 5.

Let $\mathcal{M}$ be an m-dimensional Riemannian manifold with its local coordinates $x \in \mathbb{R}^m$ and Riemannian metric $G(x)$. Consider a smooth curve in $\mathcal{M}$ expressed as $x : [0, T] \to \mathbb{R}^m$ in coordinates, where its velocity norm is defined as $\|\dot{x}\| := \sqrt{\dot{x}^T G(x) \dot{x}}$. The space of all smooth curves is considered an infinite-dimensional function space $\mathcal{X}$ with an inner product defined as follows: $\langle v, w \rangle_x := \int_0^T v(t)^T G(x(t)) w(t)\, dt$ for two square-integrable functions $v, w : [0, T] \to \mathbb{R}^m$ (i.e., $\mathcal{X}$ is a Hilbert space).

Of particular relevance to this article is a parametric curve $x(t; w)$ where $w \in \mathcal{W}$ denotes the parameter of the curve and $\mathcal{W} \subset \mathbb{R}^n$. Consider the set of all parametric curves $\mathcal{X}_\mathcal{W} := \{x(\cdot; w) \in \mathcal{X} \mid w \in \mathcal{W}\}$. This space is a n-dimensional smooth manifold under the following conditions.

Proposition 1: Suppose a curve $x(t; w)$ is smooth in both t and w and $x(t; \cdot) : \mathcal{W} \to \mathcal{X}$ is injective, i.e., if $x(t; w_1) = x(t; w_2)$ for all $t \in [0, T]$, then $w_1 = w_2$. Let $w = (w^1, \ldots, w^n)$ and $v = (v^1, \ldots, v^n) \in \mathbb{R}^n$, if

$$\sum_{i=1}^{D} \frac{\partial x(t; w)}{\partial w^i} v^i = 0 \quad \Rightarrow \quad v = 0 \tag{4}$$

for all $w \in \mathcal{W}$ and $\mathcal{W}$ is compact, then $\mathcal{X}_\mathcal{W}$ is an n-dimensional smooth manifold.

Proof: A smooth map $x(t; \cdot) : \mathcal{W} \to \mathcal{X}$ is an injective immersion by (4). Since $\mathcal{W}$ is compact, the mapping is an embedding (i.e., $\mathcal{X}_\mathcal{W}$ is an embedded manifold in $\mathcal{X}$). ■

Given a parametric curve manifold $\mathcal{X}_\mathcal{W}$ embedded in $\mathcal{X}$, the inner product $\langle \cdot, \cdot \rangle_x$ in $\mathcal{X}$ can be naturally projected into $\mathcal{X}_\mathcal{W}$. This leads to—by treating $\mathcal{W}$ as a local coordinate space for $\mathcal{X}_\mathcal{W}$—the Riemannian metric in $\mathcal{X}_\mathcal{W}$ expressed in the parameter space $\mathcal{W}$, denoted by $H(w) = \{h_{ij}(w)\}$. Specifically, the squared length of an infinitesimal displacement $dw \in \mathbb{R}^n$ is

$$\begin{aligned} ds^2 &= \sum_{i,j} h_{ij}(w) dw^i dw^j \\ &= \int_0^T \left\langle \sum_i \frac{\partial x(t; w)}{\partial w^i} dw^i, \sum_j \frac{\partial x(t; w)}{\partial w^j} dw^j \right\rangle_x dt \\ &= \sum_{i,j} \left(\int_0^T \frac{\partial x(t; w)^T}{\partial w^i} G(x(t; w)) \frac{\partial x(t; w)}{\partial w^j}\, dt \right) dw^i dw^j. \end{aligned} \tag{5}$$

Therefore

$$H(w) = \int_0^T \left(\frac{\partial x(t; w)}{\partial w} \right)^T G(x(t; w)) \frac{\partial x(t; w)}{\partial w}\, dt \tag{6}$$

where $\frac{\partial x(t;w)}{\partial w} \in \mathbb{R}^{m \times n}$. This method of metric construction follows the standard procedure in differential geometry. A similar procedure can be found in the construction of Fisher information Riemannian metrics in statistical manifolds [37], [41].

D. Isometry and Coordinate-Invariant Distortion Measure

Consider two Riemannian manifolds, an m-dimensional manifold $\mathcal{M}$ with local coordinates $x \in \mathbb{R}^m$ and metric $G(x)$ and an n-dimensional manifold $\mathcal{N}$ with local coordinates $y \in \mathbb{R}^n$ and metric $H(y)$. And let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a differentiable mapping between two manifolds expressed in local coordinates.

We call f an *isometry* if it preserves geometric structures between two spaces, i.e., preserves distances, angles, and volumes. Specifically, let $J(x) = \frac{\partial f}{\partial x}(x)$, if

$$G(x) = J(x)^T H(f(x)) J(x) \tag{7}$$

at x, then f is called a local isometry at x (to see why, compare $dx^T G(x) dx$ and $dy^T H(y) dy$ where $dy = J(x) dx$). If this condition is satisfied at all points in $\mathcal{M}$, then f is a (global) isometry; see Fig. 6.

Sometimes, it is too stringent to find an isometry between two spaces and better to ignore the scale of distances [12]. A mapping f that satisfies the relaxed condition

$$G(x) = cJ(x)^T H(f(x)) J(x) \tag{8}$$

[1] A manifold $\mathcal{A}$ is diffeomorphic to another manifold $\mathcal{B}$ if there exists a differentiable map between $\mathcal{A}$ and $\mathcal{B}$ such that its inverse exists and is differentiable as well.

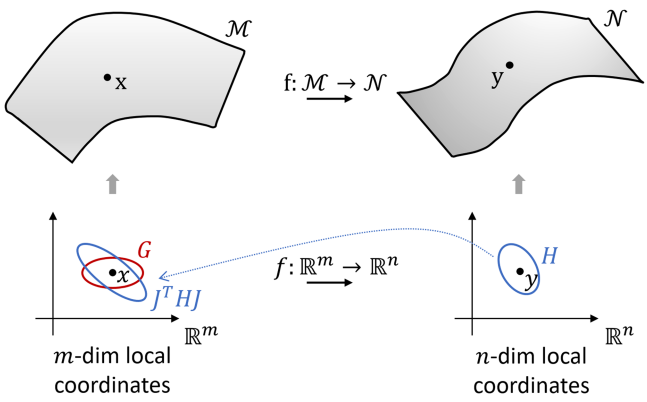

Fig. 6. Illustration of a mapping between two Riemannian manifolds. If $G = J^T HJ$, i.e., the red and blue ellipses coincide, at all points in $\mathbb{R}^m$, then the mapping f is a global isometry.

for all x and for some positive scalar c is called a *scaled isometry*. It preserves, angles and scaled distances.

There is a family of coordinate-invariant Riemannian *distortion measures*, each of which is a functional of a mapping f that measures how far f from being an isometry [42]. Let λ_i be eigenvalues of $J^T HJG^{-1}$. One example is

$$\int_{\mathcal{M}} \|\lambda_i(x) - 1\|^2 \sqrt{\det G(x)} dx. \tag{9}$$

This measure is coordinate-invariant[2], and note that if $\lambda_i(x) = 1$ for all x, then the measure is zero and f is an isometry.

A family of *relaxed distortion measures* quantifies how far f from being a scaled isometry [12] within the support of a positive finite measure ν in $\mathcal{M}$. One of them is

$$\int_{\mathcal{M}} \| \frac{\lambda_i(x)}{\frac{1}{\nu(\mathcal{M})} \int_{\mathcal{M}} \frac{1}{m} \sum_{i=1}^m \lambda_i(x) d\nu(x)} - 1\|^2 d\nu(x). \tag{10}$$

This measure is coordinate-invariant, and note that if $\lambda_i(x) = c$ for all x in the support of ν for some positive scalar c, then the measure is zero and f is a scaled isometry in ν [i.e., (8) is satisfied at all points in the support of ν].

Given a probability measure P in $\mathcal{M}$, restricting the relaxed distortion measure (10) to the support of P and removing the additive constant, it is proportional to

$$\mathcal{R}(f; P) := \frac{\mathbb{E}_{x\sim P}[\mathrm{Tr}\left((J^T HJG^{-1})^2\right)]}{\mathbb{E}_{x\sim P}[\mathrm{Tr}\left(J^T HJG^{-1}\right)]^2}. \tag{11}$$

Given this trace-based expression, we can employ the Hutchinson stochastic trace estimator, i.e., $\mathrm{Tr}(A) = \mathbb{E}_{v\sim\mathcal{N}(0,I)}[v^T Av]$, which facilitates an efficient implementation of isometric regularization. Further details can be found in [5] and [7].

[2]To see why, consider a pair of coordinate transformations $\psi : x \mapsto x'$ and $\phi : y \mapsto y'$. Let $\Psi = \frac{\partial \psi}{\partial x}$ and $\Phi = \frac{\partial \phi}{\partial y}$, then the Riemannian metrics transform via $G \mapsto \Psi^{-T} G \Psi^{-1}$ and $H \mapsto \Phi^{-T} H \Phi^{-1}$, while the Jacobian transforms via $J \mapsto \Phi J \Psi^{-1}$. Therefore, $J^T HJG^{-1} \mapsto \Psi^{-T} J^T HJG^{-1} \Psi^T$ and the eigenvalues remain unchanged.

IV. Isometric Motion Manifold Primitives++

In this section, we begin with the limitations of the existing MMP framework that relies on discrete-time trajectory representations and propose MMP++, applying the MMP framework to the continuous-time parametric curve representations. Then, we adopt the isometric regularization technique [12] with our proposed CuveGeom Riemannian metrics, and propose IMMP++.

Throughout, we will consider an n-dimensional Riemannian configuration manifold $\mathcal{Q}$ with its coordinates $q \in Q \subset \mathbb{R}^n$ and the metric $G(q) = \{g_{ij}(q)\}$.

A. Discrete-Time MMPs and Their Limitations

MMP framework learns a continuous manifold of trajectories, providing a mapping that maps a lower dimensional latent value to the discrete-time trajectory of a fixed length [6], [7]. A trajectory data are considered as a sequence of configurations denoted by $(q_1, \ldots, q_T)$ with a fixed length T and treated as an element of the high-dimensional trajectory space $Q^T := Q \times \cdots \times Q$.

Assuming the given demonstration trajectory dataset $\mathcal{D}_{\mathrm{traj}} = \{(q_1^i, \ldots, q_T^i)\}_{i=1}^N$ lies on some lower dimensional manifold, an autoencoder framework is adopted to learn this manifold and its coordinates. An autoencoder consists of an encoder $g : Q^T \to \mathcal{Z}$ and a decoder $f : \mathcal{Z} \to Q^T$, where $\mathcal{Z} = \mathbb{R}^m$ is a latent coordinate space. These two mappings are optimized to minimize the following trajectory reconstruction loss: denoting the reconstructed trajectory by $(\hat{q}_1^i, \ldots, \hat{q}_T^i) = f(g(q_1^i, \ldots, q_T^i))$:

$$\frac{1}{N} \frac{1}{T} \sum_{i=1}^N \sum_{t=1}^T d_Q^2(q_t^i, \hat{q}_t^i) \tag{12}$$

where $d_Q(\cdot, \cdot)$ is some distance metric in Q.

Minimizing (12) makes $\mathcal{D}_{\mathrm{traj}}$ approximately lie on the image of the decoder function f. As discussed in Section III-B, if f is injective, the Jacobian of f is m everywhere, and f is diffeomorphic to its image, then the image of the decoder can be considered as an m-dimensional manifold embedded in Q^T. Then, the mappings g, f with $\mathcal{Z}$ take the role of the coordinate chart. Consequently, an autoencoder can be interpreted as learning the motion manifold and its coordinates, simultaneously.

In practice, g and f are approximated using deep neural networks with smooth activation functions. By setting m sufficiently low, much lower than $\dim(Q^T) = nT$, empirical results imply that f converges to satisfy the above-mentioned conditions without enforcing them. However, choosing a large m may drop the rank of the Jacobian of f [5].

However, the nature of discrete-time trajectory representation leads to multiple limitations, as listed below. First, because the reconstructed trajectories may not be smooth, additional smoothness regularization must be added to the reconstruction loss (12), requiring weight tuning. Second, temporal modulation is not possible, meaning we cannot modulate the speed of the trajectory. More importantly, generating a configuration at an arbitrary specified time is not possible, which makes one of our key applications—online iterative replanning introduced later in Algorithm 1—inapplicable where the time variable needs to be continuously modified. Finally, given some constraints on curves

(e.g., via-points), generated trajectories cannot be enforced to satisfy these constraints. We may introduce an additional regularization term to the loss (12); however, this not only requires additional weight tuning but also does not guarantee the satisfaction of the constraints. Furthermore, smooth modulation of the trajectory is not allowed, such as by changing via-points, which limits the model's adaptability. For example, if we modulate a configuration at the first time step q_1, since there is no continuity between adjacent points, the other points q_t for $t > 1$ will remain unchanged.

B. Motion Manifold Primitives++

This section proposes MMP++, an extension of the MMP framework to the continuous-time parametric curve models. We use a phase variable $\tau \in [0, 1]$, and our main subject of interest is a parametric curve model

$$q : [0, 1] \times \mathcal{W} \to Q \quad \text{s.t.} \quad q(\tau, w) \in Q. \tag{13}$$

Then, given any monotonically increasing function with time $\tau(t)$, a timed-trajectory $q(\tau(t); w)$ can be constructed with a desired velocity profile $\frac{d}{dt}q(\tau(t); w) = \dot{\tau}_t \frac{\partial}{\partial \tau} q(\tau; w)$. This is called a temporal modulation.

Specifically, we focus on a particular class of curve models, an affine curve model, that is expressed as follows:

$$q(\tau; w) = \psi(\tau) + w\phi(\tau) \tag{14}$$

where $\psi(\tau) \in \mathbb{R}^n$, $\phi(\tau) \in \mathbb{R}^B$, and $w \in \mathbb{R}^{n \times B}$. This class includes models from ProMP [10] and VMP [11]. In VMP, $\psi(\tau)$ is referred to as an elementary trajectory, and $w\phi(\tau)$ is termed a shape modulation.

We assume that we are provided with multiple demonstration trajectories for a given task, each of which is a sequence of time-configuration pairs $((t_1, q_1), \ldots, (t_L, q_L))$. In the preprocessing step, we fit each demonstration trajectory to the affine curve model (14) and find w. Specifically, we set $\tau_{\text{linear}}(t) = \frac{t}{t_L}$ and consider $\Delta_i := q_i - \psi(\tau_{\text{linear}}(t_i))$. Then, we want to find w that best fits the trajectory, i.e., $\min_w \sum_{i=1}^{L} \|\Delta_i - w\phi(\tau_{\text{linear}}(t_i))\|$. Assuming $L > B$, there is a closed-form solution

$$w^* = \Delta\Phi^T(\Phi\Phi^T)^{-1} \tag{15}$$

where the data matrix $\Delta = (\Delta_1, \ldots, \Delta_L) \in \mathbb{R}^{n \times L}$ and basis matrix $\Phi = (\phi(\tau_{\text{linear}}(t_1)), \ldots, \phi(\tau_{\text{linear}}(t_L))) \in \mathbb{R}^{B \times L}$.

Suppose we are given curve parameters fitted to the demonstration trajectories, denoted by $\{w_1, \ldots, w_N\}$ where $w_i \in \mathbb{R}^{n \times B}$ is a curve parameter fitted to an ith trajectory. Adopting [6] and [7], we use an autoencoder framework to learn the manifold and its coordinates. An autoencoder consists of an encoder $g : \mathcal{W} \to \mathcal{Z}$ and a decoder $f : \mathcal{Z} \to \mathcal{W}$, where $\mathcal{W} = \mathbb{R}^{n \times B}$ is the curve parameter space and $\mathcal{Z} = \mathbb{R}^m$ is a latent coordinate space. These two mappings are optimized to minimize the following standard autoencoder reconstruction loss:

$$\frac{1}{N} \sum_{i=1}^{N} \|w_i - f(g(w_i))\|_F^2 \tag{16}$$

where $\|\cdot\|_F$ is the Frobenius norm.

Minimizing (16) makes $\{w_i\}_{i=1}^N$ approximately lie on the image of the decoder function f. As a result, f maps the latent coordinate space $\mathcal{Z}$ to a manifold in the curve parameter space $\mathcal{W}$, and then the parametric curve model $q(\tau; \cdot)$ maps this manifold to a manifold of continuous-time trajectories, i.e., the motion manifold, in the trajectory space, as visualized in Fig. 1.

Once f, g are fitted, we train a latent space distribution using the encoded data $\{g(w_i)\}_{i=1}^N$. To capture the multimodality of the distribution, we use a GMM, yet any other distribution fitting methods can be used. We call this framework MMP++, where "++" is added to distinguish it from vanilla MMP that uses discrete-time trajectory representations.

One might question why, instead of adopting a two-step approach to learn the autoencoder and latent density separately, we do not utilize the variational autoencoder (VAE) framework directly [30]. While using a VAE is a feasible approach, we have found that separating manifold learning from density learning proves more effective. This is because, in some instances, the KL divergence in VAEs—which penalizes differences between prior and posterior distributions in the latent space—can adversely affect the quality of reconstructions.

Another question that may arise is why we do not fit a density model directly in the curve parameter space $\mathcal{W}$. The primary reason is the high-dimensionality of the curve parameter space. For example, if the configuration space dimension is 7 and the number of basis functions $B = 30$, then the dimensionality of $\mathcal{W}$ is 210. Learning a density directly in this high-dimensional space is often challenging and, as shown in our later experiments, performs worse than our methods that learn densities in much lower dimensional latent spaces. More importantly, the high dimensionality of the curve parameter space makes it unsuitable for fast adaptation, such as the online iterative replanning introduced later in Algorithm 1. This is because the high dimensionality of $\mathcal{W}$ 1) requires a lot of data points for accurate density estimation[3], which is not the case in our situation and 2) makes the optimization insufficiently fast.

C. Isometric Regularization

The MMP++ often produces a geometrically distorted latent coordinate space $\mathcal{Z}$. Adopting [12], we would like to minimize the distortion between $\mathcal{Z}$ and $f(\mathcal{Z}) \subset \mathcal{W}$ by adding the relaxed distortion measure (11) of the decoder mapping $f : \mathcal{Z} \to \mathcal{W}$ to the reconstruction loss function.

We consider the latent space $\mathcal{Z}$ as a Riemannian manifold assigned with the identity metric $I \in \mathbb{R}^{m \times m}$, i.e., an m-dimensional Euclidean space. To apply the isometric regularization [12], we should be able to interpret the output space $\mathcal{W}$ as a local coordinate space for the embedded manifold $\mathcal{X}_\mathcal{W} = \{\psi(\tau) + w\phi(\tau) \in \mathcal{X} \mid w \in \mathcal{W}\}$ (see Section III-C). The space $\mathcal{X}_\mathcal{W}$ is an nB-dimensional smooth manifold, if $\phi_1(\tau), \ldots, \phi_B(\tau)$ are linearly independent:

Proposition 2: Suppose $\phi_1(\tau), \ldots, \phi_B(\tau)$ are linearly independent, i.e., if $a_1\phi_1(\tau) + a_2\phi_2(\tau) + \cdots + a_d\phi_d(\tau) = 0$ for all

[3]Any consistent estimator for p-times differentiable d-dimensional density functions converges at a rate of at most $n^{-\frac{p}{2p+d}}$ where n is the number of samples [43]. Therefore, when d is large, the rate is very slow.

$\tau \in [0,1]$, then $(a_1, \ldots, a_d) = 0$. Then, the affine curve model (14) satisfies the injective immersion condition in Proposition 1.

Proof: Suppose $\psi(\tau) + w_1\phi(\tau) = \psi(\tau) + w_2\phi(\tau)$ for all τ, which implies that $(w_1 - w_2)\phi(\tau) = \sum_{j=1}^{B}(w_1 - w_2)^{ij}\phi_j(\tau) = 0$ for all τ and i. By the linearity, $w_1 = w_2$; the injectivity is proved. Now, suppose $\sum_{i,j} \frac{\partial q(\tau;w)}{\partial w^{ij}} v^{ij} = 0$, which implies that $\sum_j v^{ij}\phi_j(\tau) = 0$ for all τ and i. Similarly, by the linearity, $v = 0$; thus the mapping $w \mapsto w\phi(\tau)$ is an immersion. ■

In existing movement primitives [10], [11], one of the standard methods for constructing $\phi(\tau)$ involves normalizing scalar-valued functions $b_i(\tau), i = 1, \ldots, B$: $\phi_i(\tau) = b_i(\tau)/\sum_{j=1}^{B} b_j(\tau)$. With this construction, if $\{b_i\}$ is linearly independent and $\sum_j b_j(\tau) > 0$ for all τ, then $\{\phi_i\}$ is linearly independent as well. We can construct such $\{b_i\}$ with the following proposition.

Proposition 3 ([44, Corollary of Proposition 4.]): Let $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a positive function and $\{c_1, c_2, \ldots, c_B\}$ be a finite set of mutually distinct points. Define $b_i(\tau) = K(\tau, c_i)$. Then, $\{b_i\}$ is linearly independent if and only if the matrix $(K(c_i, c_j))_{i,j=1,\ldots,B}$ is positive definite.

Consider the most standard choice of $b_i(\tau)$ for stroke-based movements, the Gaussian basis functions $b_i^G(\tau) := \exp(-\frac{(\tau - c_i)^2}{2h})$ where h defines the width of basis and c_i the center for the ith basis. According to Proposition 3, if $\{c_1, c_2, \ldots, c_B\}$ is mutually distinct, then $\{b_i^G\}$ is linearly independent, because Gaussian kernel is positive definite.

For a smooth manifold $\mathcal{X}_{\mathcal{W}}$, we can now define a Riemannian metric expressed in coordinates $w \in \mathcal{W}$ using (6). Since our parameter $w \in \mathbb{R}^{n \times B}$ is a matrix and has two indices $\{w^{ij}\}$, the Riemannian metric has four indices $h_{ijkl}(w)$ such that

$$ds^2 = \sum_{i,k=1}^{n} \sum_{j,l=1}^{B} h_{ijkl}(w) dw^{ij} dw^{kl} \tag{17}$$

for $dw \in \mathbb{R}^{n \times B}$. Accordingly, we define a CurveGeom Riemannian metric in $\mathcal{W}$ as follows.

Definition 1: A **CurveGeom Riemannian metric** for $\mathcal{X}_{\mathcal{W}}$ expressed in $\mathcal{W}$ is

$$h_{ijkl}(w) = \int_0^1 \frac{\partial q(\tau; w)^T}{\partial w^{ij}} G(q(\tau; w)) \frac{\partial q(\tau; w)}{\partial w^{kl}} \, d\tau \tag{18}$$

for $i, k = 1, \ldots, n$ and $j, l = 1, \ldots, B$.

Given an affine curve model, the metric further simplifies to the following expression.

Proposition 4: Suppose $q(\tau; w) = \psi(\tau) + w\phi(\tau)$, then the CurveGeom Riemannian metric is

$$h_{ijkl}(w) = \int_0^1 \phi_j(\tau) g_{ik}(q(\tau; w)) \phi_l(\tau) \, d\tau \tag{19}$$

for $i, k = 1, \ldots, n$ and $j, l = 1, \ldots, B$.

Proof: Let us denote by $q = (q^1, \ldots, q^n)$. Then, $\frac{\partial q^a(\tau;w)}{\partial w^{ij}} = \frac{\partial}{\partial w^{ij}}(\sum_{ab} w^{ab}\phi_b) = \sum_b \delta_i^a \delta_j^b \phi_b = \delta_i^a \phi_j$. Therefore, the metric is $h_{ijkl} = \int \sum_{a,b} g_{ab} \delta_i^a \phi_j \delta_k^b \phi_l \, d\tau = \int g_{ik} \phi_j \phi_l \, d\tau$. ■

Now, we can compute the relaxed distortion measure of the decoder mapping $f : \mathcal{Z} \to \mathcal{W}$ using (11). Since the metric for $\mathcal{Z}$ is the identity, we only need to compute $J^T H J$. Unlike the case in (11), the Jacobian of our decoder $f = (f^{ij})_{i=1,\ldots,n, j=1,\ldots,B}$ is not a matrix, but has three indices $\frac{\partial f^{ij}}{\partial z^a}$ where $a = 1, \ldots, m$. Therefore, instead of $J^T H J$, we can write it as follows:

$$\bar{h}_{ab}(z) = \sum_{i,k=1}^{n} \sum_{j,l=1}^{B} \left(\frac{\partial f^{ij}}{\partial z}(z) \right)^T h_{ijkl}(f(z)) \frac{\partial f^{kl}}{\partial z}(z) \tag{20}$$

where $\frac{\partial f^{ij}}{\partial z}(z) \in \mathbb{R}^{1 \times m}$. We let $\{\bar{h}_{ab}(z)\}$ be an index notation of an $m \times m$ matrix $\bar{H}(z)$. If $\bar{H}(z) = cI$ for some positive scalar c for all z, then f is a scaled isometry.

We consider a latent space probability measure P and finally define the relaxed distortion measure as

$$\mathcal{R}(f; P) := \frac{\mathbb{E}_{z \sim P}[\mathrm{Tr}\left(\bar{H}(z)^2\right)]}{\mathbb{E}_{z \sim P}[\mathrm{Tr}\left(\bar{H}(z)\right)]^2}. \tag{21}$$

Following [12], sampling from P is done by $\delta z_i + (1 - \delta) z_j$ where δ is uniformly sampled form $[-\eta, 1 + \eta]$ (we set $\eta = 0.2$ throughout) and $z_i = g(w_i)$ and $z_j = g(w_j)$ with $w_i, w_j \sim \{w_i\}_{i=1}^{N}$. The final loss function is

$$\frac{1}{L} \sum_{i=1}^{L} \|w_i - f(g(w_i))\|^2 + \alpha \mathcal{R}(f; P) \tag{22}$$

where α is a regularization coefficient. Together with the density model fitted in the latent coordinate space, we call this framework IMMP++.

As a special case, if $\mathcal{Q}$ is Euclidean space, i.e., $g_{ij}(w)$ is equal to the Kronecker delta δ_{ij}, then the metric formula and isometric regularization term can be further simplified. Plugging it into (19), the metric is simplified to

$$h_{ijkl} = \delta_{ik} \int_0^1 \phi_j(\tau) \phi_l(\tau) \, d\tau. \tag{23}$$

We note that it does not depend on w; therefore, we do not need to compute it in every iteration of the gradient descent during autoencoder training. This greatly reduces the computational cost in isometric regularization. Specifically, the matrix $\bar{H}(z)$ becomes

$$\bar{h}_{ab}(z) = \sum_{i=1}^{n} \left(\frac{\partial f^i}{\partial z}(z) \right)^T \Phi \frac{\partial f^i}{\partial z}(z) \tag{24}$$

where $f^i = (f^{i1}, \ldots, f^{iB}) \in \mathbb{R}^B$, $\frac{\partial f^i}{\partial z}(z) \in \mathbb{R}^{B \times m}$, and $\Phi = (\int_0^1 \phi_j(\tau)\phi_l(\tau) \, d\tau)_{j,l=1,\ldots,B} \in \mathbb{R}^{B \times B}$ is constant.

V. Experiments

In Section V-A and V-B, we assume Euclidean configuration spaces Q and focus on examples with fixed initial and final points $q_i, q_f \in Q$, therefore we use the via-point affine curve model from VMP [11]

$$q(\tau; w) = (1 - \tau) q_i + \tau q_f + w\phi(\tau) \tag{25}$$

where $\phi_i(\tau) = \tau(1 - \tau) \, b_i^G(\tau)/\sum_j b_j^G(\tau)$ (it is trivial to show the linear independence of ϕ_i; see Proposition 3). To give hard constraints of initial and final points to $q(\tau; w)$, we multiply $\tau(1 - \tau)$ to the original basis term from Zhou et al.'s [11] work.

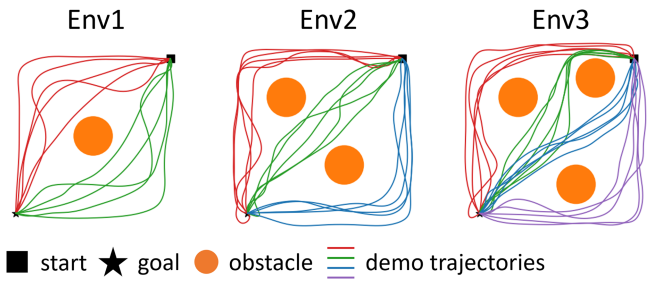

Fig. 7. Three environments with training demonstration trajectories.

One might wonder whether it is always necessary to design parametric curve models in an environment-specific manner. We note that the parametric curve model (14), with the choice of $\psi(\tau) = 0$ and $\phi_i(\tau) = b_i^G(\tau)/\sum_j b_j^G(\tau)$, is sufficiently expressive to model any smooth, arbitrary complex trajectory with a sufficiently large B. Thus, no special modeling is required if we do not enforce any constraints on the curve. If we have desired constraints, we can, in some cases, design suitable parametric models that guarantee these constraints are satisfied. For example, if we want $q(\tau)$ to pass through (τ_i, q_i) and $(\tau_i, \dot{q}_i)$ for $i = 1, \ldots, N$, we can design $\psi(\tau)$ to be the lowest order polynomial that satisfies (τ_i, q_i) and $(\tau_i, \dot{q}_i)$ for all i, and set $\phi(\tau) = \Pi_i(\tau - \tau_i)^2 b_j^G(\tau)/\sum_j b_j^G(\tau)$.

We compare MMP++, IMMP++, and VMP [11]. We utilize GMMs to fit latent density models for MMP++ and IMMP++ unless otherwise specified. While the distribution of w is assumed to be Gaussian in vanilla VMP, it has limitations in capturing multimodal distributions. To ensure a fair comparison, we also implement VMP with GMM. When sampling new trajectories from fitted distributions, we reject the samples with likelihood values lower than the threshold value η – where η is set to be the minimum value among the likelihood values of the training trajectories.

In Section V-C, we consider the position-orientation space

$$\mathrm{SE}(3) := \{(p, R) \mid p \in \mathbb{R}^3, R \in \mathrm{SO}(3)\}$$

where $\mathrm{SO}(3) := \{R \in \mathbb{R}^{3\times 3} \mid R^T R = I, \det(R) = 1\}$ is the group of rotation matrices; p and R denote the position and orientation, respectively. In the position space $\mathbb{R}^3$, we use the affine curve model $p(\tau; w)$ in (25). In the orientation space $\mathrm{SO}(3)$, we use a tailored parametric curve model $R(\tau; w)$, of which details will be given later in the corresponding section. In comparison to the MMP with discrete-time trajectory representations in [7], we show that our MMP++ enables additional modulations of initial and final positions and orientations.

A. Planar Obstacle-Avoiding Motions

We consider three different environments with different numbers of obstacles and obstacle-avoiding trajectories; see Env1, Env2, and Env3 in Fig. 7. The number of total training trajectories for each environment setting is 10, 15, and 20, respectively. The number of basis $B = 20$ for $\phi(\tau)$. In MMP++ and IMMP++. The latent space dimension is search among 2, 3, 4, and 5 The number of GMM components is set to be 2, 3, and 4 for Env1, Env2, and Env3, respectively. A generated trajectory is considered successful if it does not collide with the obstacles.

TABLE I
AVERAGES AND STANDARD DEVIATIONS OF THE SUCCESS RATES WITH FIVE TIMES RUN WITH DIFFERENT RANDOM SEEDS; THE HIGHER THE BETTER

	Env1	Env2	Env3
VMP (Gaussian)	76.42 ± 1.34	65.76 ± 1.60	38.24 ± 1.18
VMP (GMM)	97.12 ± 0.51	97.52 ± 0.30	98.14 ± 0.24
MMP++ (ours)	99.48 ± 0.38	88.58 ± 0.60	97.76 ± 1.46
IMMP++ (ours)	**100.00 ± 0.00**	**99.90 ± 0.12**	**99.22 ± 0.19**

The best results are marked in bold.

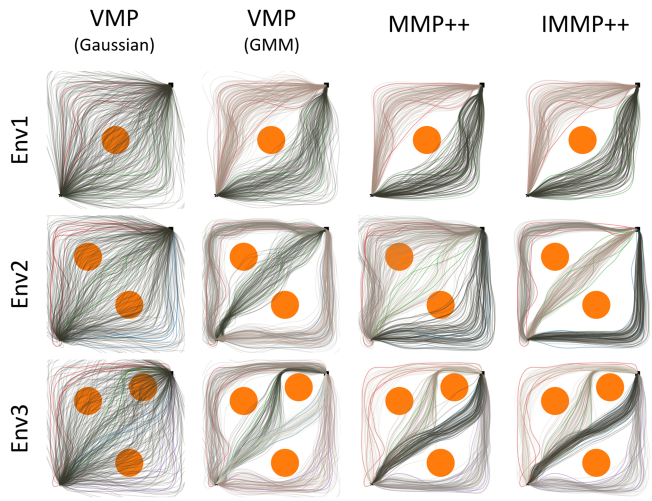

Fig. 8. Trajectories generated by trained models. The GMM component numbers are 2, 3, and 4 for Env1, Env2, and Env3, respectively (samples from the same GMM component are assigned the same color).

As shown in Table I and Fig. 8, the IMMP++ performs the best compared to the other methods. Sometimes, MMP++ fails significantly because GMM fits wrong clusters due to the geometric distortions in the latent coordinate spaces; see Fig. 2 for example latent spaces of MMP+ and IMMP++.

B. Seven-DoF Robot Arm Collision-Free Motions

In this section, we consider collision-free point-to-point motions of a seven-DoF Franka Emika Panda robot arm in an environment shown in Fig. 9 (*left*). Two types of demonstration trajectories are given, each with ten joint space trajectories—in Fig. 9 (*right*), SE(3) and $\mathbb{R}^3$ forward kinematics results of them are visualized for simplicity—where the initial and final configurations are identical as q_i and q_f in Fig. 9 (*left*). In the demo type 1, there are two clusters of trajectories, whereas in the demo type 2 a set of trajectories forms an 1-D manifold, a trajectory manifold embedded in the trajectory space. The number of basis $B = 20$ for $\phi(\tau)$. In MMP++ and IMMP++, the latent space dimensions are 2. The number of GMM components is 2. Exceptionally, for MMP++ and IMMP++ applied to the demo type 2, we use the kernel density estimator (KDE) instead of the GMM (we will provide the reason later).

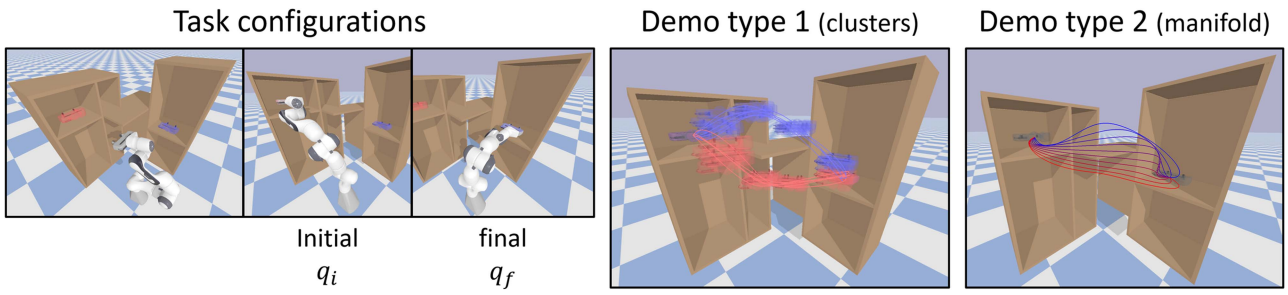

Fig. 9. *Left:* A seven-DoF robot arm needs to move from the joint configuration q_i to q_f without colliding to the environment. *Right:* Two types of demonstration trajectories, where 7-D joint space trajectories are given as demonstration data (only the end-effector's SE(3) or $\mathbb{R}^3$ trajectories are visualized). In demo type 1, two clusters of trajectories are given, and, in demo type 2, a 1-D manifold of trajectories is provided.

TABLE II
AVERAGES AND STANDARD DEVIATIONS OF THE SUCCESS RATES WITH FIVE TIMES RUN WITH DIFFERENT RANDOM SEEDS; THE HIGHER THE BETTER

Demo	VMP (Gaussian)	VMP (GMM)	MMP++ (ours)	IMMP++ (ours)
type 1	46.1 ± 4.37	97.1 ± 1.66	98.6 ± 0.20	**99.5 ± 0.77**
type 2	79.4 ± 2.94	83.0 ± 3.78	**99.3 ± 0.68**	98.2 ± 0.68

The best results are marked in bold.

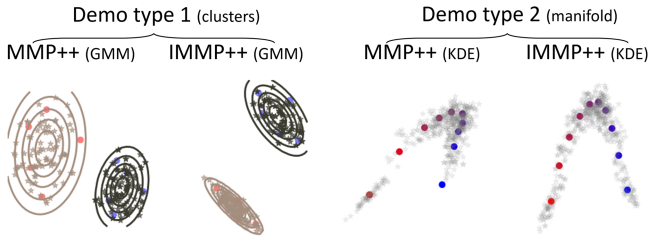

Fig. 10. 2-D latent spaces of our manifold-based methods (MMP++ and IMMP++) are illustrated (circular points are encoded points and star-shaped points are generated samples). Demonstration trajectories in Fig. 9 (*right*) are encoded into these latent spaces, with colors indicating correspondence (e.g., a red trajectory is encoded into a red point).

Table II shows the averages and standard deviations of the success rates. Overall, MMP++ and IMMP++ show the best performances. In particular, VMPs show far inferior results than our manifold-based methods given a manifold of demonstrations in the demo type 2. This is because neither Gaussian nor GMM models are suitable for capturing the continuous manifold structure of the density's support. Fig. 10 shows the 2-D latent spaces of the manifold-based methods. For demo type 1, the distance between clusters in the latent space of IMMP+ is greater than that in the latent space of MMP++. Even though MMP++ successfully captures correct clusters in this particular dataset, for more complex datasets, it is more likely to fail in capturing correct clustering structures.

For demo type 2, we note that 2-D encoded latent points form an 1-D manifold; it should be considered as a single connected manifold component. Given this manifold support, we found that it is sub-optimal to use the GMM for fitting a distribution. Hence, we use a nonparametric method, KDE that can more accurately estimate our latent space density function

$$p(z) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\pi|H_i|^{1/2}} \exp\left(-\frac{(z-z_i)^T H_i^{-1}(z-z_i)}{2}\right) \tag{26}$$

where $\{z_i\}_{i=1}^N$ is the set of encoded latent points and $H_i, i = 1, \ldots, N$ are positive-definite matrices. Let $K(z_i, z_k) = \exp(-\|z_i - z_k\|^2/h)$; for this example, we construct $H_i = \Sigma_i^2$ where

$$\Sigma_i = \frac{\sum_{k=1}^N K(z_i, z_k)(z_i - z_k)(z_i - z_k)^T}{\sum_{k=1}^N K(z_i, z_k)}. \tag{27}$$

One may ask if using such a nonparametric method in the curve parameter space $\mathcal{W}$ of VMP directly can lead to a better performance than using GMM. Unfortunately, this is challenging due to the high-dimensional nature of $\mathcal{W}$. It is worth noting that this approach is feasible in MMP++ and IMMP++ because we utilize sufficiently low-dimensional latent spaces.

Next, we qualitatively show the latent coordinates and via-points modulation results of IMMP++ trained with the demo type 2 (manifold) in Fig. 11. Note that we have the model $q(\tau; f(z)) = (1-\tau)q_i + \tau q_f + f(z)\phi(\tau)$ with the trained decoder function $w = f(z)$, where modulations in the latent coordinate value z and the initial and final configurations q_i and q_f lead to corresponding changes in the resulting trajectory $q(\tau)$. In Fig. 11 (*upper*), a continuous change in the latent value z leads to a smooth transition of $q(\tau)$ from an upward-moving path to a downward-moving path. As shown in Fig. 11 (*middle* and *lower*), continuous changes in q_i and q_f induce smooth changes of $q(\tau)$.

Finally, we show the adaptability of our primitive models in the presence of unseen constraints and propose a novel online iterative replanning algorithm. We will consider (z, τ) as a state. Let T be a total time length. First of all, suppose we do not have additional unseen constraints. We start with a random initial $z \sim p(z)$ and $\tau = 0$. Then, as the time δt passes, z remains constant and τ is updated to $\min(\tau + \delta t/T, 1)$. At each time step, we use $q(\tau; f(z))$ as a desired joint configuration and run a position controller with the control frequency f_c. This procedure is just a position tracking control given an initially planned trajectory $q(\tau; f(z))$.

Now, consider a situation where an obstacle that blocks the initially planned trajectory appears and moves as the time flows,

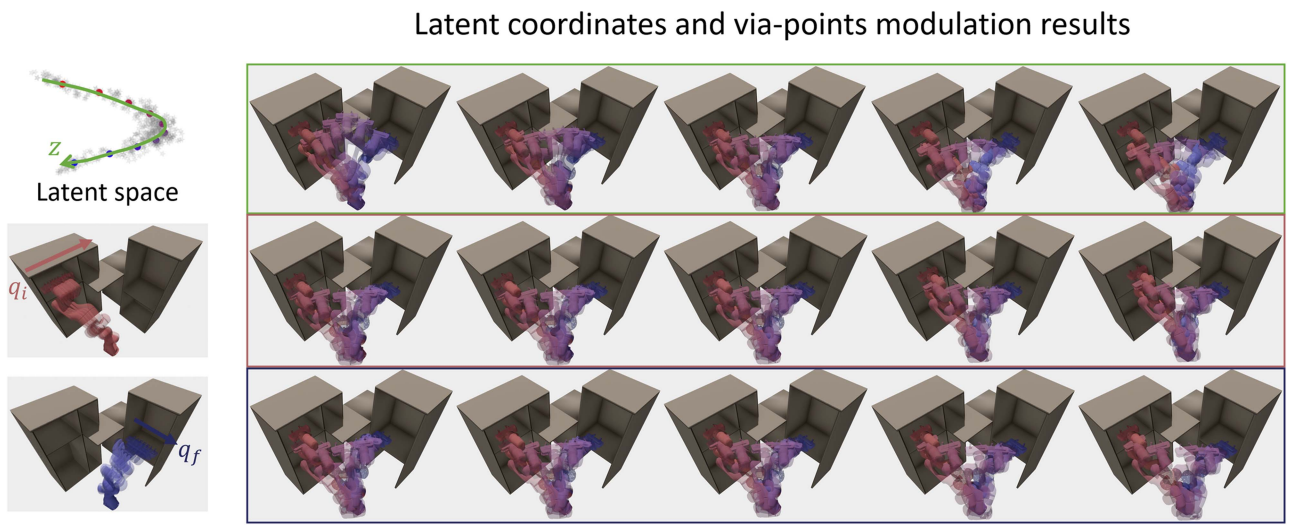

Fig. 11. Modulations of latent coordinates z and initial and final configurations q_i and q_f in IMMP++ $q(\tau; f(z)) = (1-\tau)q_i + \tau q_f + f(z)\phi(\tau)$. *Upper:* A continuous change in z results in a smooth transition of $q(\tau)$. *Middle:* Changes in q_i induce smooth transitions of $q(\tau)$. *Lower:* Changes in q_f induce smooth transitions of $q(\tau)$.

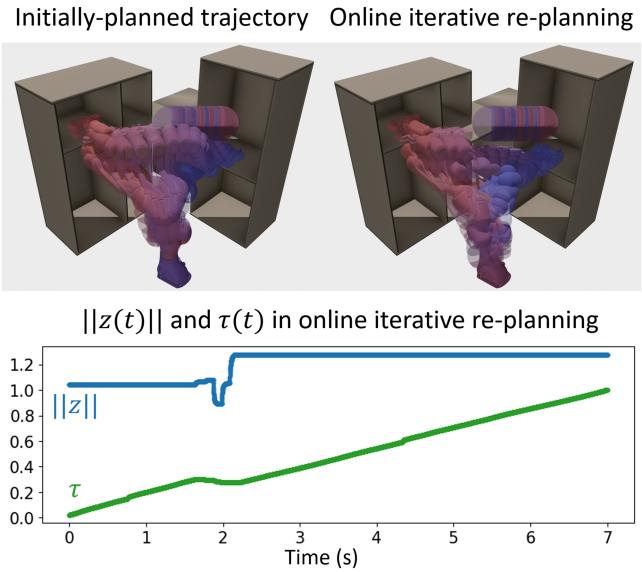

Fig. 12. *Upper left:* Initially planned trajectory is blocked by the unseen obstacle during training. *Upper right:* The robot iteratively replans its trajectory whenever it is predicted to collide with the obstacle. *Lower:* $\|z\|$ and τ as functions of time t in online iterative replanning (*Upper right*), where $T = 5$, $t_w = 1$, $f_c = 1000$, and $f_p = 10$.

as shown in Fig. 12. We assume constraints are given as inequality equations $C(q) \leq 0$ in the configuration space $\mathcal{Q}$, which can change dynamically in time. Suppose our current state is (z, τ). We set a time window t_w and if $C(q(\bar{\tau}; f(z)) > 0$ for some $\tau < \bar{\tau} < \tau + t_w/T$, we re-plan the trajectory, i.e., update and find a new desired state (z', τ'), with the replanning frequency $f_p < f_c$. Given a new desired state (z', τ'), we will update $z \leftarrow z + k(z' - z)$ and $\tau \leftarrow \tau + k(\tau' - \tau)$ with positive gain k at control frequency f_c for time $1/f_p$, and then decide whether we will re-plan (z, τ) again or not. Considering this update rule, in the replanning step, we find (z', τ') by solving the following optimization problem:

$$
\begin{aligned}
&\min_{(z',\tau')} \ \|z - z'\|^2 + \alpha\|\tau - \tau'\|^2 \\
&\text{such that } (1)\ C(q(\bar{\tau}'; f(z')) \leq 0 \\
&\qquad\qquad \tau' < \bar{\tau}' < \tau' + t_w/T \\
&\qquad\quad (2)\ \ \log p(z') \geq \epsilon \\
&\qquad\quad (3)\ \ C(q(\tau_\eta; f(z_\eta)) \leq 0 \ \ \&\ \ \log p(z_\eta) \geq \epsilon \\
&\qquad\qquad z_\eta = \eta z + (1-\eta)z' \\
&\qquad\qquad \tau_\eta = \eta\tau + (1-\eta)\tau' \\
&\qquad\qquad \eta \in [0, 1] \\
&\qquad\quad (4)\ \ \tau' \in [\tau - \delta, \tau]
\end{aligned}
\tag{28}
$$

where the weight $\alpha > 0$ is set to be big enough to prioritize updating z.

1) Enforces the planned trajectory from (z', τ') to satisfy the constraints for time t_w.
2) Prevents z' from being out-of-distribution in the latent space (ϵ is the log probability density threshold; we set it to be the minimum value among the log likelihood values of the training data).
3) Guarantees the linear path connecting the current and next states from (z, τ) to (z', τ') satisfies the constraints and lies within the in-distribution region in the latent space.
4) Allows $\tau' \in [\tau - \delta, \tau]$ – where δ is the max travel-back time—in case no feasible z' exists when $\tau' = \tau$. A pseudocode for our online iterative replanning algorithm is provided in Algorithm 1.

We can efficiently solve the optimization (28) thanks to the low-dimensionality of the optimization variable (z', τ'). In the example in Fig. 12, we use IMMP++ trained with the demo

Algorithm 1: Online Iterative Trajectory Re-planning with MMP++.

Input: MMP++ decoder $w = f(z)$, parametric curve $q(\tau; w)$, latent density $p(z)$, inequality constraint $C(q) \leq 0$, time window t_w, total time length T, gain $k > 0$, control frequency f_c, re-planning frequency f_p, and optimization solver SOLVE (28)

Initialization: $z \sim p(z)$, $\tau = 0$, $c_v = \text{False}$

while *True* **do**
- Once every $1/f_p$ seconds:
- **if** $C(q(\bar{\tau}; f(z))) > 0$ *for some* $\bar{\tau} \in [\tau, \min(\tau + t_w/T, 1)]$ **then**
  - $(z_g, \tau_g) \leftarrow$ SOLVE (28)
  - $c_v = \text{True}$
- **else**
  - $c_v = \text{False}$
- **end**
- Once every $1/f_c$ seconds:
- **if** c_v **then**
  - $z \leftarrow z + k(z_g - z)$
  - $\tau \leftarrow \tau + k(\tau_g - \tau)$
- **else**
  - $z \leftarrow z$
  - $\tau \leftarrow \min(\tau + 1/(f_c T), 1)$
- **end**
- Position control input: $q(\tau; f(z))$

end

TABLE III
COMPUTATION TIMES FOR PLANNING VIA RRT-CONNECT [45] AND IMMP++ ARE MEASURED WITH THREE TIMES RUN. WE CONSIDER A SHELF-ONLY ENVIRONMENT, THE SHELF WITH ONE SPHERICAL OBSTACLE, AND THE SHELF WITH TWO SPHERICAL OBSTACLES; SEE FIG. 13. COMPUTATIONS ARE PERFORMED USING AN AMD RYZEN 9 5900X 12-CORE PROCESSOR AND AN NVIDIA GEFORCE RTX 3090. ALL CODE IS IMPLEMENTED IN PYTHON

	RRT-connect [45]	IMMP++ (ours)
Shelf	$1.97 \sim 4.81$ s	0.006 ± 0.001 s
Shelf + one obstacle	$60.41 \sim 134.36$ s	0.039 ± 0.001 s
Shelf + two obstacles	$45.97 \sim 204.56$ s	0.077 ± 0.002 s

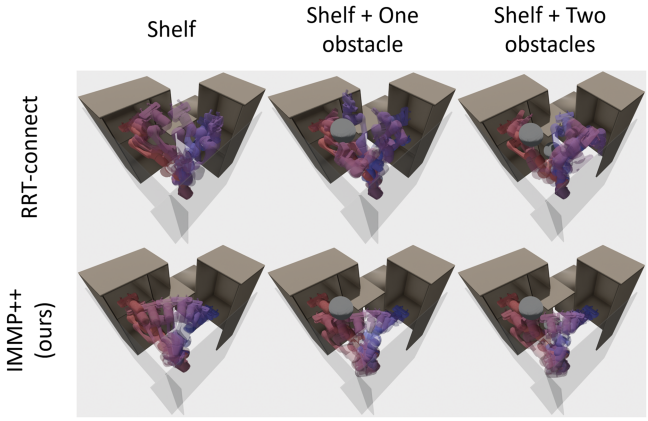

Fig. 13. Comparison of collision-free paths planned by RRT-connect [45] and IMMP. Two hypothetical walls, shown as transparent gray, are considered to limit the robot's workspace to the area close to the shelf.

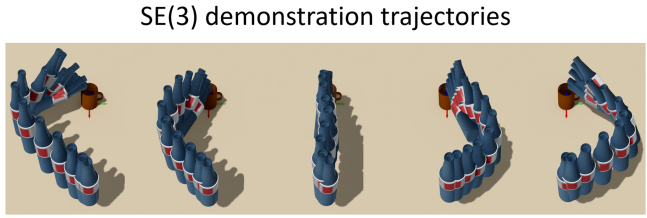

Fig. 14. Water-pouring SE(3) trajectory dataset.

type 2 (manifold). As visualized in Fig. 12 (*upper right*), given a dynamically changing constraint (i.e., moving spherical obstacle), the robot iteratively replans its trajectory whenever it is predicted to collide with the obstacle within the time interval t_w. A gradient-free sampling-based optimization approach is fast enough and shows a successful online iterative replanning result. Fig. 12 (*lower*) shows $\|z\|$ and τ as functions of time, indicating when (z, τ) is replanned. It can be confirmed that at around $t = 2$ the robot predicted collisions with the obstacle within $t_w = 1$ and replanned (z, τ) multiple times.

One can reasonably ask, if we have access to the robot kinematics and the geometry of the environment, what is the advantage of using our method compared to conventional sampling-based collision-free path planning methods, such as RRT variants? Our method offers two important advantages. First, our MMP already contain paths that are collision-free for a part of the environment, so only the newly introduced environmental constraints need to be considered during planning. For example, in Fig. 11, the paths in the learned motion manifold do not collide with the shelf. Therefore, when planning within the motion manifold, exact knowledge of the shelf's geometry is unnecessary. While existing sampling-based planning methods require knowledge of all constraints, we only need to consider the new environmental constraints, such as obstacles shown in Fig. 12.

Second, a more significant advantage is that, while conventional RRTs are typically not fast enough for online dynamic planning, our method is very fast. Table III shows the computation times for planning using RRT-connect [45] and our IMMP++ in the shelf-only environment and with one and two sphere obstacles, as visualized in Fig. 13. RRT-connect takes several seconds even in the simple shelf-only environment, and the time increases significantly as the complexity of the collision-free configuration space grows due to obstacles. In contrast, our method is relatively much faster. The trajectory sampling in the motion manifold itself takes 0.006 s, and the collision checking for obstacles takes only a few milliseconds. Our current implementation is in Python and is suboptimal. We anticipate that implementing it in a more optimal language would result in even faster performance. Although the RRT algorithm could also be optimized for speed, it is unlikely to become fast enough and suitable for online dynamic planning.

However, these advantages of our algorithm do not come for free. Our method also has limitations. If the learned MMP are too small, meaning the generated trajectories are not sufficiently

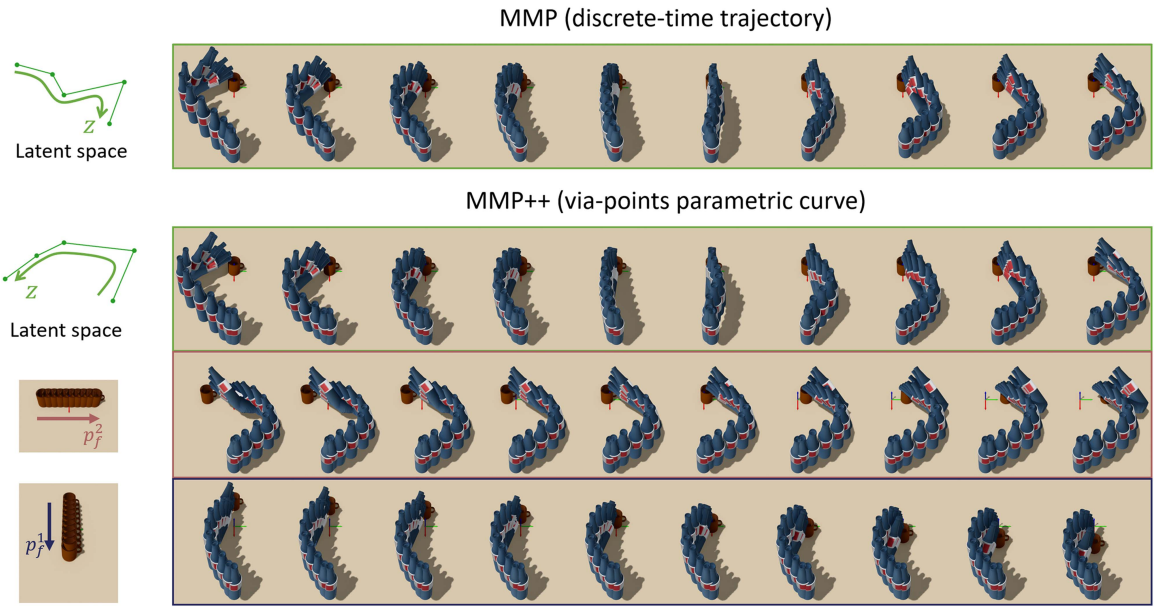

Fig. 15. Modulation of MMP trained to generate discrete-time trajectories and MMP++ trained to generate curve parameters of the trajectories. We can modulate latent values for both methods and generate smooth transitions of the SE(3) trajectories. In MMP++ (*middle* and *lower*), given continuous changes in the cup position, the water-pouring trajectories undergo smooth transitions.

diverse, a collision-free path may not exist on the motion manifold when excessive environmental constraints are introduced. The diversity of the learned motion manifold heavily depends on the diversity of the demonstration data, making the creation of good demonstration trajectories a very important issue.

C. MMP++ for SE(3) Trajectory Data

In this section, we show how to extend our MMP++ framework to SE(3) trajectory data. We will denote the position by $p \in \mathbb{R}^3$ and the orientation by 3×3 rotation matrix $R \in \mathbb{R}^{3\times 3}$. Let $\phi_i(\tau) = \tau(1-\tau)\, b_i^G(\tau) / \sum_j b_j^G(\tau)$ for $i = 1, \ldots, B$. For the position trajectory, we use the via-point model

$$p(\tau; w_p) = (1-\tau)p_i + \tau p_f + w_p \phi(\tau) \tag{29}$$

where $w_p \in \mathbb{R}^{3\times B}$ is the position curve parameter and $p_i, p_f \in \mathbb{R}^3$ are initial and final points. For the orientation trajectory, we use the following parametric model:

$$R(\tau; w_R) = R_i \exp(\tau \log(R_i^T R_f)) \exp([w_R \phi(\tau)]) \tag{30}$$

where $w_R \in \mathbb{R}^{3\times B}$ is the orientation curve parameter, $R_i, R_f \in \mathrm{SO}(3)$ are initial and final orientations, and $\exp(\cdot)$, $\log(\cdot)$ are matrix exponential and logarithm, and

$$[(w^1, w^2, w^3)] = \begin{bmatrix} 0 & -w^3 & w^2 \\ w^3 & 0 & -w^1 \\ -w^2 & w^1 & 0 \end{bmatrix}$$

is a skew symmetric matrix [46]. The orientation via-point model in (30) is constrained to have the initial and final orientations R_i and R_f.

Consider the water-pouring SE(3) trajectory dataset in Fig. 14. Our goal is to train an MMP++ with this dataset. First, note that the five trajectories in the dataset have the same initial SE(3) pose (p_i, R_i) but have different final SE(3) poses (p_f, R_f). Therefore, in addition to (w_p, w_R), the final SE(3) pose (p_f, R_f) is added to the curve parameter. To emphasize this, we denote the parametric curves by $p(\tau; w_p, p_f)$ and $R(\tau; w_R, R_f)$. Thus, our autoencoder g, f is trained by using $\{(w_p, w_R, p_f, R_f)_i\}_{i=1}^5$ where each of these parameters is fitted to the demonstration trajectory.

To construct neural network encoder and decoder, some cares need to be taken since $R_f \in \mathrm{SO}(3)$ is not a vector data. For the encoder $g : (w_p, w_R, p_f, R_f) \mapsto z$, we can flatten R_f to a 9-D vector and treat (w_p, w_R, p_f, R_f) as a $6B + 12$-dimensional vector input. For the decoder $f : z \mapsto (w_p, w_R, p_f, R_f)$, the orientation output R_f should satisfy the rotation matrix constraints. To enforce this condition, we let $f : z \mapsto (w_p, w_R, p_f, w_f)$ where $w_f \in \mathbb{R}^3$ and map $w_f \mapsto \exp([w_f]) \in \mathrm{SO}(3)$.

For autoencoder training, we may use the L2 reconstruction loss in the parameter space $(w_p, w_R, p_f, R_f) \in \mathbb{R}^{6B+12}$ as in (16). However, we found that this loss does not suitably capture the difference between the training and reconstructed SE(3) trajectories, resulting in an autoencoder with poor reconstruction qualities. Therefore, we use the following loss function: let $(\hat{w}_p, \hat{w}_R, \hat{p}_f, \hat{R}_f) = f \circ g(w_p, w_R, p_f, R_f)$ and (p_τ, R_τ) be the SE(3) demonstration trajectory—where τ denotes the normalized time parameter (i.e., for a total demonstration time length T, $\tau = t/T$)—, denoting the trajectory dataset by $\mathcal{D} = \{(p_\tau, R_\tau)_i\}_{i=1}^5$, the new reconstruction loss is

$$\sum_{(p_\tau, R_\tau)\in\mathcal{D}} \int_0^1 \|p_\tau - p(\tau; \hat{w}_p, \hat{p}_f)\|^2 + \beta \|\log(R_\tau^T R(\tau; \hat{w}_R, \hat{R}_f))\|_F^2 \, d\tau \tag{31}$$

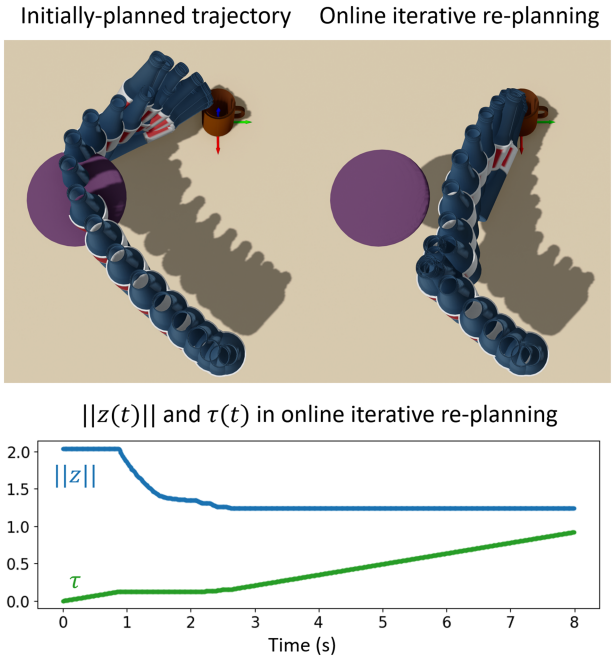

Fig. 16. *Upper-left:* Initially planned trajectory is blocked by the unseen obstacle during training. *Upper-right:* We iteratively replan the bottle's SE(3) trajectory whenever it is predicted to collide with the obstacle. *Lower:* $\|z\|$ and τ as functions of time t in online iterative replanning (*Upper-right*), where $T = 7$, $t_w = 0.2$, $f_c = 100$, and $f_p = 10$.

where $\beta > 0$ and the integration over τ is approximately computed by the discrete sum.

Fig. 15 shows the modulation results of the MMP trained with discrete-time trajectories as in [7] and MMP++ with our parametric curve models. Both methods enable modulation of the latent value z, producing continuous transitions of water-pouring trajectories from left to right. Furthermore, MMP++ enables modulation of the initial and final poses p_i, R_i, p_f, R_f. As shown in Fig. 15 (MMP++: *middle* and *lower*), MMP++ produces pouring trajectories that adapt to changes in the cup position.

Finally, we apply the online iterative replanning algorithm using the water-pouring MMP++ in the presence of unseen obstacle. We use the replanning algorithm in Algorithm 1. Similar to the robot arm experiment with the demo type 2 (manifold), we use the KDE density model (26). While the initially planned SE(3) trajectory of the bottle collides with the purple spherical obstacle, as shown in Fig. 16 (*upper left*), the bottle's SE(3) trajectory is replanned online when it is predicted to collide with the obstacle and successfully avoids the collision, as shown in Fig. 16 (*upper right*). Fig. 16 (*lower*) shows $\|z\|$ and τ as functions of time, indicating when (z, τ) is replanned. It can be confirmed that, from $t = 1$ to $t = 2.5$, collisions with the obstacle within the time interval $t_w = 0.2$ are predicted and (z, τ) is replanned multiple times.

VI. Conclusion and Discussion

We have proposed a novel model of movement primitives based on the motion manifold hypothesis, named MMP++. This model synergizes the strengths of the existing MMP with those of conventional parametric curve representation-based primitive models, achieving enhanced motion generation accuracy and adaptability. Moreover, to overcome the geometric distortion issue in MMP++, we have introduced a CurveGeom Riemannian metric for the parametric curve space and presented IMMP++. This approach ensures that the latent space preserves the geometric structure of the motion manifold, which, in some instances, leads to significantly improved density fitting results within the latent space.

We highlight that the low dimensionality of the latent parametrization of the motion manifold has led to multiple advantages. This allows us to exploit nonparametric density fitting techniques, such as kernel density estimation, in the latent space, which are typically challenging to apply in high-dimensional spaces, for example, directly in the curve parameter space $\mathcal{W}$. In addition, replanning has been easily accomplished by finding a new latent value z, formulated as an optimization problem that is efficiently solvable thanks to the low dimensionality of the latent space.

We have extended MMP++ to accommodate matrix Lie group data, specifically SO(3), by designing a parametric curve model that utilizes the exponential map, logarithm map, and group action. Given that the water-pouring SE(3) trajectory dataset forms a connected manifold without a clustering structure, MMP++—even without isometric regularization—has shown satisfying performance. However, in instances where we encounter a dataset with clustering structures, where a distorted latent space could result in poor density fitting, as verified in the other two-DoF and seven-DoF robot experiments, employing isometric regularization alongside an appropriate Riemannian metric can offer a solution. Although developing isometric regularization for matrix Lie group data falls outside the scope of this article, it represents an interesting direction for future research.

Although our focus has been on via-point models, we can adopt other parametric curve models with stronger inductive biases. Even more sophisticated movement primitives, such as those based on dynamical systems, can be used, for example, by making the parameters of the dynamical systems the outputs of the neural network, as done in [47]. These can potentially be combined with the autoencoder-based manifold learning technique. These combinations will yield a variety of manifold-based motion primitives capable of generating diverse motions, demonstrating high adaptability in previously unseen dynamic environments.

Finally, we note that our current MMP are not conditioned on vision inputs, such as images, point clouds, or sequences of images, so the learned manifold of trajectories is applicable only to the trained environment, such as the specific shelf in Fig. 11. If the MMP model could generalize to environments it has not encountered during training, it would be extremely powerful. For example, if arbitrary shelf geometries were given as vision inputs, and the model could generate a manifold of diverse trajectories conditioned on those inputs, it would significantly enhance its applicability. Adopting techniques (e.g., network architectures) from recent LfD methods for visuomotor policy learning [48], [49] is a feasible direction to extend our framework to generate a manifold of trajectories conditioned on vision

inputs. For example, the decoder can be modified to take a vision feature y along with the latent variable z, so that the decoder maps (z, y) to the curve parameter w. Although this would require a relatively large amount of data paired with visual observations, our core idea remains applicable.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[2] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *Int. J. Robot. Res.*, vol. 42, no. 13, pp. 1133–1184, 2023.

[3] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, 2018, Art. no. 17.

[4] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: On the curvature of deep generative models," in *Proc. Int. Conf. Learn. Represenations*, 2018.

[5] Y. Lee, "A geometric perspective on autoencoders," 2023, *arXiv:2309.08247*.

[6] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy, "Task-conditioned variational autoencoders for learning movement primitives," in *Proc. Conf. Robot Learn.*, 2020, pp. 933–944.

[7] B. Lee, Y. Lee, S. Kim, M. Son, and F. C. Park, "Equivariant motion manifold primitives," in *Proc. 7th Annu. Conf. Robot Learn.*, 2023, pp. 1199–1221.

[8] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control—A unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.

[9] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.

[10] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Adv. Neural Inf. Process. Syst.*, vol. 26, 2013.

[11] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter-and extrapolation capabilities," in *2019 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4301–4308.

[12] Y. Lee, S. Yoon, M. Son, and F. C. Park, "Regularized autoencoders for isometric representation learning," in *Proc. Int. Conf. Learn. Representations*, 2022.

[13] A. Pervez, A. Ali, J.-H. Ryu, and D. Lee, "Novel learning from demonstration approach for repetitive teleoperation tasks," in *2017 IEEE World Haptics Conf.*, 2017, pp. 60–65.

[14] Y. Fanger, J. Umlauft, and S. Hirche, "Gaussian processes for dynamic movement primitives with application in knowledge-based cooperation," in *2016 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3913–3919.

[15] J. Umlauft, Y. Fanger, and S. Hirche, "Bayesian uncertainty modeling for programming by demonstration," in *2017 IEEE Int. Conf. Robot. Automat.*, 2017, pp. 6428–6434.

[16] A. Pervez, Y. Mao, and D. Lee, "Learning deep movement primitives using convolutional neural networks," in *2017 IEEE-RAS 17th Int. Conf. Humanoid Robot. (Humanoids)*, 2017, pp, 191–197.

[17] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[18] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven Lyapunov candidates," in *2013 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1216–1222.

[19] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robot. Auton. Syst.*, vol. 70, pp. 1–15, 2015.

[20] C. Blocher, M. Saveriano, and D. Lee, "Learning stable dynamical systems using contraction theory," in *2017 14th Int. Conf. Ubiquitous Robots Ambient Intell.*, 2017, pp. 124–129.

[21] V. Sindhwani, S. Tu, and M. Khansari, "Learning contracting vector fields for stable imitation learning," 2018, *arXiv:1804.04878*.

[22] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[23] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 833–852, 2019.

[24] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proc. Humanoids 2008-8th IEEE-RAS Int. Conf. Humanoid Robots*, 2008, pp. 91–98.

[25] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *2009 IEEE Int. Conf. Robot. Automat.*, 2009, pp. 2587–2592.

[26] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Auton. Robots*, vol. 32, pp. 433–454, 2012.

[27] M. Ginesi, D. Meli, A. Calanca, D. Dall'Alba, N. Sansonetto, and P. Fiorini, "Dynamic movement primitives: Volumetric obstacle avoidance," in *2019 19th Int. Conf. Adv. Robot.*, 2019, pp. 234–239.

[28] H. Beik-Mohammadi, S. Hauberg, G. Arvanitidis, G. Neumann, and L. Rozo, "Learning Riemannian manifolds for geodesic motion skills," in *Proc. Robotics: Sci. Syst.*, 2021.

[29] Y. Lee, H. Kwon, and F. Park, "Neighborhood reconstructing autoencoders," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 536–546, 2021.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Representations*, 2014.

[31] A. Creswell, Y. Mohamied, B. Sengupta, and A. A. Bharath, "Adversarial information factorization," 2017, *arXiv:1711.05175*.

[32] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Int. Conf. Mach. Learn.*, 2011, pp. 833–840.

[33] Y. Lee and F. C. Park, "On explicit curvature regularization in deep generative models," in *Proc. Topological, Algebr. Geometric Learn. Workshops*, 2023, pp. 505–518.

[34] P. Nazari, S. Damrich, and F. A. Hamprecht, "Geometric autoencoders-what you see is what you decode," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 25834–25857.

[35] S. Yoon, Y.-K. Noh, and F. Park, "Autoencoding under normalization constraints," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12087–12097.

[36] C. Jang, Y. Lee, Y.-K. Noh, and F. C. Park, "Geometrically regularized autoencoders for non-Euclidean data," in *Proc. 11th Int. Conf. Learn. Representations*, 2022.

[37] Y. Lee, S. Kim, J. Choi, and F. Park, "A statistical manifold framework for point cloud data," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 12378–12402.

[38] H. Shao, A. Kumar, and P. T. Fletcher, "The Riemannian geometry of deep generative models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 428–4288.

[39] M. P. Do Carmo and J. F. Francis, *Riemannian Geometry*, vol. 6. Berlin, Germany: Springer, 1992.

[40] M. Fecko, *Differential Geometry and Lie Groups for Physicists*. New York, NY, USA: Cambridge Univ. Press, 2006.

[41] S.-I. Amari, *Information Geometry and Its Applications*, vol. 194. Berlin, Germany: Springer, 2016.

[42] C. Jang, Y.-K. Noh, and F. C. Park, "A Riemannian geometric framework for manifold learning of non-Euclidean data," *Adv. Data Anal. Classification*, vol. 15, no. 3, pp. 673–699, 2021.

[43] C. J. Stone, "Optimal rates of convergence for nonparametric estimators," *Ann. Statist.*, vol. 8, no. 6, pp. 1348–1360, 1980.

[44] V. I. Paulsen and M. Raghupathi, *An Introduction to the Theory of Reproducing Kernel Hilbert Spaces*, vol. 152. New York, NY, USA: Cambridge Univ. press, 2016.

[45] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. 2000 ICRA. Millennium Conf., IEEE Int. Conf. Robot. Automat., Symposia Proc.*, 2000, pp. 995–1001, vol. 2.

[46] K. M. Lynch and F. C. Park, *Modern Robotics*. New York, NY, USA: Cambridge Univ. Press, 2017.

[47] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural dynamic policies for end-to-end sensorimotor learning," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 5058–5069, 2020.

[48] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proc. Robot.: Sci. Syst.*, 2023.

[49] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto, "Behavior generation with latent actions," in *Proc. Int. Conf. Learn. Representations*, 2024.

Yonghyeon Lee (Member, IEEE) received the B.S. and Ph.D. degrees in mechanical engineering from Seoul National University, Seoul, South Korea, in 2018 and 2023, respectively.

Since 2023, he has been an AI Research Fellow with the Center for Artificial Intelligence, Korea Institute for Advanced Study, Seoul. His research interests include representation learning, robot learning, deep learning, dynamics, planning and control, vision and image processing, and geometric machine learning.