MeshDMP: Learning Motion on Surfaces for Flexible Manufacturing

Matteo Dalle Vedove[1,2], Fares J. Abu-Dakka[3], Luigi Palopoli[4], Daniele Fontanelli[1], Matteo Saveriano[1]

***Abstract*— An open problem in industrial automation is to reliably perform tasks requiring in-contact movements with complex workpieces, as current solutions lack the ability to seamlessly adapt to the workpiece geometry. In this paper, we propose a Learning from Demonstration approach that allows a robot manipulator to learn and generalise motions across complex surfaces by leveraging differential mathematical operators on discrete manifolds to embed information on the geometry of the workpiece extracted from triangular meshes, and extend the Dynamic Movement Primitives (DMPs) framework to generate motions on the mesh surfaces. We also propose an effective strategy to adapt the motion to different surfaces, by introducing an isometric transformation of the learned forcing term. The resulting approach, namely MeshDMP, is evaluated both in simulation and real experiments, showing promising results in typical industrial automation tasks like car surface polishing.**

I. Introduction

In recent years, industrial automation has witnessed growing interest in integrating robotic systems for repetitive tasks to increase productivity, enabling the human to face more cognitive-demanding tasks. Still, there are several open challenges in developing intelligent platforms that can perform motion on complex surfaces, such as polishing, grinding, and cleaning. If the geometry of the workpiece is simple and the job is not changing in time, an ad-hoc trajectory could be generated offline for the specific operation through traditional coding. Still, a majority of these tasks are performed by human operators, as they require a high online adaptation capability to cope with the complexity of the workpiece shape. Automating the latter tasks is challenging, especially in the context of flexible manufacturing, since it is not possible to rely on pre-defined trajectories. Therefore, the robot should inherit the human-like capability to adapt to the workpiece geometry, and generalise motion patterns to such a variety of shapes.

In this context, Learning from Demonstration (LfD) [1] emerges as a promising approach to transfer human expertise and knowledge to robotic systems by simply observing the human behaviour. Once a set of demonstrations is collected, either by visual observation or kinesthetic teaching [1], different mathematical frameworks can be used to encode such motions. By describing motion as stable dynamical systems, Dynamic Movement Primitives (DMPs) [2], [3] provide a compact representation that enables to generalise the learned motion to different initial and goal configurations, and execution speed, without model retraining or trajectory-shape modification. Their extension on Riemannian manifolds [4], [5], also enabled a wide variety of robotic applications requiring control of the whole end-effector pose, such as surface polishing [6]–[9].

Standard approaches treat the planning problem on curved surfaces without exploiting its intrinsic geometry. Instead, either 1) they assume a flat surface to perform bi-dimensional planning, and use low-level control strategies to ensure that the end-effector stays on the surface [9], or 2) they learn the motion in the operational space of the robot [6]. The main pitfall of these approaches is that they do not exploit the a-priori knowledge of the surface geometry, which can be obtained from CAD models or visual observations [10], thus limiting the generalisation capabilities of the motion on different surfaces.

To overcome this limitation, inspired by recent work on geometric DMPs [11], [12], we propose a novel formulation of differential operators on triangulated meshes that enables both motion policy learning and execution of end-effector paths on arbitrary surfaces. Differently from mesh-based Riemannian Motion Policies [13], where the surface is required to be isomorphic to a (closed) flat surface, our method does not constrain the surface topology, making the solution particularly suitable for industrial applications. To show the effectiveness of our approach, we propose experiments, both simulated and with real hardware, that highlight the generalisation capabilities of learned policies on different surfaces, with focus on the industrial application of polishing.

II. Preliminaries

A triangulated mesh is a discrete representation of a surface, composed of vertices and faces that form triangles. Formally, a triangulated mesh $\mathcal{M}$ can be regarded as a connected graph of n_v vertices, described by the set V, connected by edges $\mathcal{E}$ which will constitute n_f triangular faces, described by the set F. A mesh can be regarded as piecewise-linear approximation of a continuous surface, providing an efficient, yet flexible, representation for computational purposes.

In geometry, a differential manifold M is a locally Euclidean topological space where calculus can be applied. Smooth surfaces $\mathcal{S}$, such as a sphere or a torus, are 2-dimensional differentiable manifolds which provide the concept of curved plane, and over which we can perform precise

Co-funded by the European Union projects INVERSE (grant agreement no. 101136067) and MAGICIAN (grant agreement no. 101120731)

[1]Department of Industrial Engineering, Università di Trento, Trento, Italy. matteo.dallevedove@unitn.it

[2]DRIM, Ph.D. of national interest in Robotics and Intelligent Machines.

[3]Mechanical Engineering Program, Division of Engineering, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates.

[4]Department of Information Engineering and Computer Science, Università di Trento, Trento, Italy.

geometric computation.

With this work, we seek establishing an analogy that enables the use of discrete surface representations $\mathcal{M}$ to perform operations defined for smooth surfaces $\mathcal{S}$. For this reason, we must constraint $\mathcal{M}$ to be *manifold*, i.e., any edge in the polygonal mesh must be shared by at most two faces, and $\mathcal{M}$ must have a consistent normal vector field to the be *orientable* [14].

III. Proposed Approach

A. Differential operators on discrete manifolds

In this section, we extend the concept of some differential operators to discrete manifolds by first recalling their intuitive definition for bi-dimensional surfaces $\mathcal{S}$ embedded in $\mathbb{R}^3$, and the corresponding concept for discretised surfaces $\mathcal{M}$. Readers may refer to [14] for an in-depth treatment of differential geometry.

1) Geodesics: For any two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in M$, the curve $\gamma : [0, s] \subset \mathbb{R} \rightarrow M$, parameterised by arc length, with $\gamma(0) = \boldsymbol{x}_1$ and $\gamma(s) = \boldsymbol{x}_2$, is a geodetic if it minimises the distance along the manifold between these points. In the context of Riemannian geometry, where M is smooth, the geodesic γ is always defined, unique, and represents a smooth curve of length $|\gamma| = s$, with $|\cdot|$ being the Riemannian distance.

For discrete geometry, particularly polygonal meshes, different algorithms have been proposed to compute the geodesics between arbitrary points on a surface [15]–[17]. In a mesh $\mathcal{M}$, the geodesic between two points, $\boldsymbol{m}_1, \boldsymbol{m}_2 \in \mathcal{M}$, is a polyline modelled by a set of n_γ ordered segments:

$$\gamma = \left\{ \overline{\boldsymbol{p}_1 \boldsymbol{p}_2}, \ldots, \overline{\boldsymbol{p}_{n_\gamma - 1} \boldsymbol{p}_{n_\gamma}} \right\}, \tag{1}$$

where $\boldsymbol{p}_1 = \boldsymbol{m}_1$, $\boldsymbol{p}_{n_\gamma} = \boldsymbol{m}_2$, and each $\boldsymbol{p}_i \in \mathcal{M}$.

2) Tangent space: Let M be an n-manifold, given any point $\boldsymbol{x} \in M$ and a suitable local parameterisation $\boldsymbol{\phi} : \mathcal{U} \subset \mathbb{R}^n \rightarrow M$, such that $\boldsymbol{\phi}(\boldsymbol{u}_{\boldsymbol{x}}) = \boldsymbol{x}$, then a vector basis for the tangent space $T_{\boldsymbol{x}} M$ at the point $\boldsymbol{x}$, as in [18], is provided by

$$\left\{ \left. \frac{\partial \boldsymbol{\phi}}{\partial u_1} \right|_{\boldsymbol{u}_x}, \ldots, \left. \frac{\partial \boldsymbol{\phi}}{\partial u_n} \right|_{\boldsymbol{u}_x} \right\}.$$

In the case of smooth surfaces $\mathcal{S}$ embedded in $\mathbb{R}^3$, $T_{\boldsymbol{x}} \mathcal{S}$ simplifies to the 2-dimensional plane in $\mathbb{R}^3$ that is tangent to $\mathcal{S}$ at the point $\boldsymbol{x} \in \mathcal{S}$.

We can generalise this definition of the tangent plane for discrete manifolds. Let $\boldsymbol{m} \in \mathcal{M}$ be a point lying in a triangular face $\mathcal{T}_{\boldsymbol{m}}$, then the corresponding tangent space $T_{\boldsymbol{m}} \mathcal{M}$ is the plane that contains this triangle.

3) Parallel transport: The parallel transport is an isometric transformation that enables the expression of a vector $\boldsymbol{v} \in T_{\boldsymbol{x}_1} M$ in the tangent plane $T_{\boldsymbol{x}_2} M$ of another point. The general derivation is tied with the concept of parallel vector fields on geodesic curves, but in the case of smooth surfaces $\mathcal{S}$ embedded in $\mathbb{R}^3$, the parallel transport turns out to be a rotation transformation.

As discussed in Sec. III-A.1, we are able to construct the geodesic curve γ between any two points $\boldsymbol{m}_1, \boldsymbol{m}_2 \in \mathcal{M}$. It follows that the tangent vectors at the curve in $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are respectively

$$\boldsymbol{w}_1 = \frac{\overline{\boldsymbol{p}_1 \boldsymbol{p}_2}}{\|\overline{\boldsymbol{p}_1 \boldsymbol{p}_2}\|}, \qquad \boldsymbol{w}_2 = \frac{\overline{\boldsymbol{p}_{n_\gamma - 1} \boldsymbol{p}_{n_\gamma}}}{\|\overline{\boldsymbol{p}_{n_\gamma - 1} \boldsymbol{p}_{n_\gamma}}\|}.$$

Given the tangent vectors $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, the parallel transport of a vector $\boldsymbol{v} \in T_{\boldsymbol{m}_1} \mathcal{M}$ onto $T_{\boldsymbol{m}_2} \mathcal{M}$ is defined as

$$P_{\boldsymbol{m}_1 \rightarrow \boldsymbol{m}_2} \mathcal{M}(\boldsymbol{v}) : T_{\boldsymbol{m}_1} \mathcal{M} \rightarrow T_{\boldsymbol{m}_2} \mathcal{M} := \mathbf{R} \boldsymbol{v}, \tag{2}$$

where $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ is the unique rotation matrix that rotates $\boldsymbol{w}_1$ to $\boldsymbol{w}_2$.

4) Logarithmic map: The logarithmic map is a differential operator which enables the expression of a point $\boldsymbol{x}_2 \in M$ within the tangent space $T_{\boldsymbol{x}_1} M$ of $\boldsymbol{x}_1 \in M$. Such projection yields a vector $\boldsymbol{v} \in T_{\boldsymbol{x}_1} M$ whose direction is provided by the tangent vector $\dot{\gamma}(0)$ of the geodesic γ connecting $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$, and with magnitude equal to the length s of the geodesic itself.

By applying this definition on discrete manifolds, and recalling the definition (1) of the geodesic on a mesh, it follows that the logarithmic map of $\boldsymbol{m}_2$ in the tangent space of $\boldsymbol{m}_1$, denoted with $\mathrm{Log}_{\boldsymbol{m}_1}(\boldsymbol{m}_2) : \mathcal{M} \rightarrow T_{\boldsymbol{m}_1} \mathcal{M} \subset \mathbb{R}^3$, is

$$\mathrm{Log}_{\boldsymbol{m}_1}(\boldsymbol{m}_2) := \frac{\overline{\boldsymbol{p}_1 \boldsymbol{p}_2}}{\|\overline{\boldsymbol{p}_1 \boldsymbol{p}_2}\|} \sum_{i=1}^{n_\gamma - 1} \|\overline{\boldsymbol{p}_i \boldsymbol{p}_{i+1}}\|. \tag{3}$$

5) Exponential map: The exponential mapping can be regarded as the inverse operation of the logarithmic map. Given a vector $\boldsymbol{v} \in T_{\boldsymbol{x}_1} M$ defined in the tangent space of $\boldsymbol{x}_1 \in M$, such mapping yields a point $\boldsymbol{x}_2 \in M$ such that the corresponding geodesic curve γ has length $\|\boldsymbol{v}\|$ and its initial velocity $\dot{\gamma}(0)$ is directed as $\boldsymbol{v}$. In the case of smooth manifolds, one can obtain the geodesic curve by integrating a second order differential equation (ODEs) [19], thus retrieving $\boldsymbol{y}$.

While dealing with discrete manifolds, however, such definition cannot be applied. In fact, the ODEs to integrate depend on the Christoffel symbols whose values are undefined at non-differentiable points such as vertices and edges of the mesh. Still, by resorting to the intuitive concept of the exponential map, we can retrieve a numerical procedure, described in depth in Algorithm 1, to evaluate the corresponding value.

The exponential map $\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{v})$ aims at laying off a line of length $|\boldsymbol{v}|$ on the manifold M along the geodesic that passes through $\boldsymbol{x}$ with direction $\boldsymbol{v}$ [14]. The achieve this behavior, we propose an interative algorithm that, given an initial vector $\boldsymbol{v} = \boldsymbol{v}_0 \in T_{\boldsymbol{m}_0} \mathcal{M}$ applied in $\boldsymbol{m} = \boldsymbol{m}_0 \in \mathcal{M}$, computes $\hat{\boldsymbol{m}}_1$ as the displacement of $\boldsymbol{m}_0$ along $\boldsymbol{v}_0$. If such point lies within the face $\mathcal{T}_0 \subset \mathcal{M}$ where $\boldsymbol{m}_0$ was located, then it means that we found the actual projection of the vector in the mesh, thus the point corresponding to $\mathrm{Exp}_{\boldsymbol{m}}(\boldsymbol{v})$. If $\hat{\boldsymbol{m}}_1$ falls outside of $\mathcal{T}_0$, it means that the exponential mapping would be outside of such face; for this reason, we displace $\boldsymbol{m}_0$ by a vector $\tilde{\boldsymbol{v}}_0$ which enables to reach the edge of $\mathcal{T}_0$ at point $\boldsymbol{m}_1$. Let $\mathcal{T}_1$ be the other unique triangular face that shares the edge of $\mathcal{T}_0$ containing $\boldsymbol{m}_1$, we have to ensure

Algorithm 1 Exponential Map Approximation. $\mathcal{E}(\mathcal{T})$ are the edges of the triangle $\mathcal{T}$, while $\boldsymbol{n}_{\mathcal{T}}$ is the triangle normal.

Input: $\boldsymbol{m} \in \mathcal{M}$, $\boldsymbol{v} \in T_{\boldsymbol{m}}\mathcal{M}$

$\boldsymbol{m}_0 \leftarrow \boldsymbol{m}$, $\boldsymbol{v}_0 \leftarrow \boldsymbol{v}$
$\mathcal{T}_0 \leftarrow$ triangle containing $\boldsymbol{m}_0$
$k \leftarrow 0$
while $\|\boldsymbol{v}_k\| > 0$ **do**
  $\hat{\boldsymbol{m}}_{k+1} \leftarrow \boldsymbol{m}_k + \boldsymbol{v}_k$
  if $\hat{\boldsymbol{m}}_{k+1}$ inside $\mathcal{T}_k$ **then**
    return $\hat{\boldsymbol{m}}_{k+1}$
  end if
  $\boldsymbol{m}_{k+1} \leftarrow$ intersection$(\overline{\boldsymbol{m}_k \hat{\boldsymbol{m}}_{k+1}}, \mathcal{E}(\mathcal{T}_k))$
  $\mathcal{T}_{k+1} \leftarrow$ adjacent face of $\mathcal{T}_k$ sharing $\boldsymbol{m}_{k+1}$
  $\boldsymbol{v}_k^{\Delta} \leftarrow \mathrm{Log}_{\boldsymbol{m}_k}(\boldsymbol{m}_{k+1})$ {Eq. (3)}
  $\tilde{\boldsymbol{v}}_k \leftarrow \boldsymbol{v}_k - \boldsymbol{v}_k^{\Delta}$
  $\boldsymbol{\nu}_k \leftarrow \left(\mathbf{I}_{3\times 3} - \boldsymbol{n}_{\mathcal{T}_{k+1}} \boldsymbol{n}_{\mathcal{T}_{k+1}}^{\top}\right) \tilde{\boldsymbol{v}}_k$
  $\boldsymbol{v}_{k+1} = \frac{\boldsymbol{\nu}_k}{\|\boldsymbol{\nu}_k\|} \|\tilde{\boldsymbol{v}}_k\|$
  $k \leftarrow k + 1$
end while

Output: $\mathrm{Exp}_{\boldsymbol{m}}(\boldsymbol{v})$

that the vector $\boldsymbol{v}_1$ that will be used to displace $\boldsymbol{m}_1$, lies in the plane of $\mathcal{T}_1$. To do so, we take the part of the vector $\tilde{\boldsymbol{v}}_0 = \boldsymbol{v}_0 - \boldsymbol{v}_0^{\Delta}$ that has not been yet used for the displacement of $\boldsymbol{v}_0$, and apply a norm-preserving projection on the plane of $\mathcal{T}_1$. At this point, we can iterate the algorithm starting from $\boldsymbol{m}_1$ and displacing along $\boldsymbol{v}_1$, until we reach a point $\tilde{\boldsymbol{m}}_k$ that lies within a face of the mesh.

B. Mesh Dynamic Movement Primitive (MeshDMP)

To encode cyclic demonstrations on a surfaces, we extend the geometry-aware formulation of DMPs ($\mathcal{G}$-DMPs) proposed in [11] to periodic motions, and provide some insight on how to enable skill generalisation to arbitrary meshes. Note that we focus on periodic motions as they are common in industrial automation, but one can start from the approach in [12] and derive MeshDMP for point-to-point motions.

Periodic MeshDMPs consists of the following dynamic evolving on a manifold:

$$\begin{aligned} \nabla_{\boldsymbol{z}} \boldsymbol{z} &= \Omega \left(\alpha \left(\beta \, \mathrm{Log}_{\boldsymbol{y}}(\boldsymbol{g}) - \boldsymbol{z} \right) + \mathbf{T}(\boldsymbol{y}, \boldsymbol{z}) \boldsymbol{f}(\phi) \right), \\ \dot{\boldsymbol{y}} &= \Omega \boldsymbol{z}, \\ \Omega \dot{\phi} &= 1. \end{aligned} \tag{4}$$

Here, $\boldsymbol{y} \in \mathcal{M}$ represents the position state, $\boldsymbol{z} \in T_{\boldsymbol{y}}\mathcal{M}$ the (scaled) velocity, $\boldsymbol{g} \in \mathcal{M}$ the centre of the periodic motion, $\boldsymbol{f} : \mathbb{R} \to T_{\boldsymbol{y}}\mathcal{M}$ the forcing function, $\phi \in \mathbb{R}$ the phase variable, $\Omega \in \mathbb{R}^+$ a time scaling factor, and $\alpha, \beta \in \mathbb{R}^+$ coefficients associated to the linear dynamics. Similarly to [9], we assume that $\boldsymbol{f}$ is parameterised locally to the current state in the mesh manifold, thus an isometric transformation $\mathbf{T} : \mathcal{M} \times T_{\boldsymbol{y}}\mathcal{M} \to \mathbb{R}^{3\times 3}$ is required to project the forcing term in the global reference frame which is used to perform the numerical integration of (4). Note that previous work [11], [12] do not include this transformation as they consider smooth manifolds. As local frame, we propose to use the one whose first basis vector $\hat{\boldsymbol{i}}$ is aligned with the current velocity $\boldsymbol{z}$ of the MeshDMP, the third basis $\hat{\boldsymbol{k}}$ normal to the current face, and $\hat{\boldsymbol{j}}$ bi-normal to the other ones. The corresponding transformation is formally defined as

$$\mathbf{T}(\boldsymbol{y}, \boldsymbol{z}) = \begin{bmatrix} \dfrac{\boldsymbol{z}}{\|\boldsymbol{z}\|} & \boldsymbol{n}_{\boldsymbol{y}} \times \dfrac{\boldsymbol{z}}{\|\boldsymbol{z}\|} & \boldsymbol{n}_{\boldsymbol{y}} \end{bmatrix}. \tag{5}$$

The forcing function $\boldsymbol{f}$ is encoded as a linear combination of N Gaussian radial basis functions, i.e.,

$$\boldsymbol{f}(\phi) = \frac{\sum_{i=1}^{N} \Psi_i(\phi) \boldsymbol{w}_i}{\sum_{i=1}^{N} \Psi_i(\phi)} r, \tag{6}$$

wherein $\Psi_i(\phi) = \exp\left(h_i(\cos(\phi - c_i) - 1)\right)$, $\boldsymbol{w}_i \in \mathbb{R}^3$ are the weights to be learned from the demonstration, and $r \in \mathbb{R}^+$ is a scaling coefficient, that in this work we set to $\|\mathrm{Log}_{\boldsymbol{y}_0}(\boldsymbol{g})\|$, with $\boldsymbol{y}_0$ the starting configuration of the MeshDMP.

To learn the weights in (6), we need a demonstration $\mathcal{D}$, i.e., a set of N_s samples $\left\{\boldsymbol{y}_k, \dot{\boldsymbol{y}}_k, \nabla_{\dot{\boldsymbol{y}}_k}\dot{\boldsymbol{y}}_k\right\}_{k=1}^{N_s}$. By appropriately inverting (4), one can compute the desired forcing $\boldsymbol{f}_d$ at the different time-steps as

$$\boldsymbol{f}_{d,k}^{(w)} = \frac{\nabla_{\dot{\boldsymbol{y}}_k}\dot{\boldsymbol{y}}_k}{\Omega^2} - \alpha \left(\beta \, \mathrm{Log}_{\boldsymbol{y}_k}(\boldsymbol{g}) - \frac{\dot{\boldsymbol{y}}_k}{\Omega} \right), \tag{7}$$

$$\boldsymbol{f}_{d,k} = \mathbf{T}_k^{-1} \boldsymbol{f}_{d,k}^{(w)}. \tag{8}$$

Equation (7) is the common definition of the forcing term for rhythmic DMPs that leads to a forcing term $\boldsymbol{f}_{d,k}^{(w)}$ defined in the *world* reference frame. However, since we request the forcing term $\boldsymbol{f}_{d,k}$ to be defined locally to the demonstration, in (8) we perform an appropriate change of basis by considering $\mathbf{T}_k = \mathbf{T}(\boldsymbol{y}_k, \dot{\boldsymbol{y}}_k)$. Finally, given the set of forcing term $\mathcal{F} = \left\{\boldsymbol{f}_{d,k}\right\}_{k=1}^{N_s}$, one can simply solve a least-square problem to obtain the desired set of weights $\boldsymbol{w}_i$ of (6).

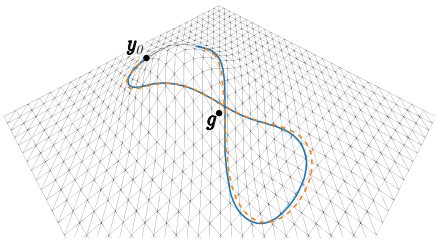
When learning on manifolds, we must ensure that the different states lie on the corresponding geometrical entity, i.e., $\boldsymbol{y}_k \in \mathcal{M}$ and $\dot{\boldsymbol{y}}_k, \nabla_{\dot{\boldsymbol{y}}_k}\dot{\boldsymbol{y}}_k \in T_{\boldsymbol{y}_k}\mathcal{M}$. In practise, to build $\mathcal{D}$, we propose to acquire position and velocity samples from the Cartesian space, i.e., to collect $\mathcal{C} = \left\{\boldsymbol{\chi}_k \in \mathbb{R}^3, \dot{\boldsymbol{\chi}}_k \in \mathbb{R}^3\right\}_{k=1}^{N_s}$. We then retrieve $\boldsymbol{y}_k$ by determining the closest point on the mesh $\mathcal{M}$ of the position $\boldsymbol{\chi}_k$, and use a norm-preserving projection of $\dot{\boldsymbol{\chi}}_k$ on the plane $T_{\boldsymbol{y}_k}\mathcal{M}$ to construct $\dot{\boldsymbol{y}}_k$. Finally, given the sampling period $dt \in \mathbb{R}^+$, the acceleration-like sample is built by numerically computing the covariant derivative as

$$\nabla_{\dot{\boldsymbol{y}}_k}\dot{\boldsymbol{y}}_k = \frac{P_{\boldsymbol{y}_{k+1} \to \boldsymbol{y}_k}(\dot{\boldsymbol{y}}_{k+1}) - \dot{\boldsymbol{y}}_k}{dt}. \tag{9}$$

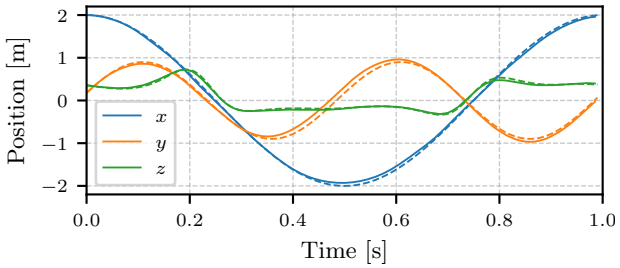
IV. Experimental Results

A. DMP learning and execution

To learn the forcing term of the MeshDMP (4), we generate synthetic demonstrations on non-flat surfaces $\mathcal{S}$ obtained as the graph of bi-variate functions $f : \mathbb{R}^2 \to \mathbb{R}$, i.e., $\mathcal{S}$ is described as the image of the function $\Pi(x, y) = (x, y, f(x, y))$ in the domain $\Theta = [a_x, b_x] \times [a_y, b_y]$. Defined a 2D differentiable curve $\eta : \mathbb{R} \to \mathbb{R}^2$, the demonstrations $\mathcal{D}$ are obtained by sampling uniformly the tri-dimensional

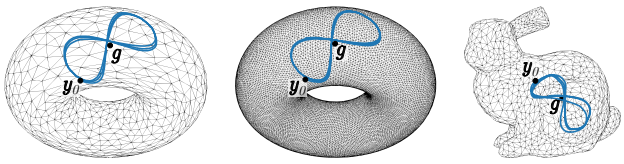


(a)



(b)

Fig. 1. (a) Demonstrated trajectory (orange) and path obtained by MeshDMP (blue) after learning the forcing term. (b) Position trajectory of the demonstration (dashed lines) and the one obtained through MeshDMP integration (solid lines).



(a) (b) (c)

Fig. 2. Execution of a MeshDMP learned on an 8-shaped trajectory on the flat surfaces, then generalised to a low (a) and high (b) polygonal density meshed torus, as well as on a simplified Stanford bunny (c).

curve $\rho = \Pi \circ \eta$. The mesh $\mathcal{M}$ approximating the surface is obtained by sampling $\mathcal{S}$ on a uniform grid in Θ.

An example of MeshDMP learning and execution is depicted in Fig. 1(a), where the demonstration consists of a 8-shaped trajectory projected on the graph of the function $f(x, y) = e^{-(x-1)^2 y^2} - 0.5e^{-(x+1)^2 y^2}$. Figure 1(b) reports the time-series plots of the Cartesian trajectories of both demonstration and learned demonstration, which scores a root mean-squared error $8 \cdot 10^{-2}$m.

Even though there's no formal way to asses the generalisation capability, we trained a MeshDMP on a 8-shaped trajectory, drawn in a flat surface, and then we integrate it on 2 surfaces with different topology, namely a torus, and the Stanford bunny, Fig. 2, clearly showing the generalisation of the proposed system in adapting to different meshes. We can also observe from Fig. 2(b) that using a higher quality mesh, leads to better performance of the algorithm since the approximation error of $\mathcal{M}$ is lower w.r.t. the nominal shape.

B. Execution on real hardware

To further prove the effectiveness of the proposed algorithm, we conducted an experiment simulating the wiping operation on industrial work-pieces; the video of the experiment can be found at `https://youtu.be/3Az6q5JfL94`.

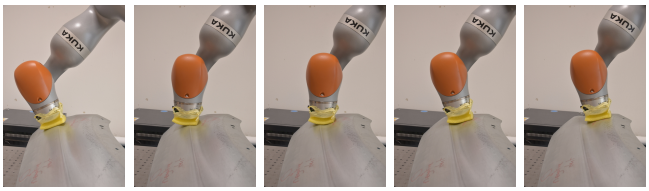


Fig. 3. End-effector configurations while traversing the crest present in the surface.

The proposed experiment is performed on a Kuka Iiwa14 polishing the front fender of a car, a piece with non-trivial curvature that makes traditional robot programming and LfD techniques hard to execute. Since MeshDMPs provide trajectories in the surface domain, we choose as set-point for the controller a reference system centred in the DMP-provided position, and z axis pointing inward the object.

Our first experiment shows the effectiveness of MeshDMPs in real-life scenarios, by learning an elliptical pattern in a flat surface, that then is executed on the surface of the fender. From a graphical user interface (GUI), the user can specify the centre position, i.e., the centre of the ellipse, and the initial position of the MeshDMP, and internally the algorithm determines the proper initial velocity and integrates the DMP for the requested time amount, and with custom cycle period. By planning the trajectory directly within the surface of the mesh, it follows that the end-effector can easily adapt to steep curvature changes in the mesh. An example is shown in Fig. 3, where the robot smoothly follows the surface normal when passing through the *crest* present in the fender.

The second experiment shows the trajectory generated by a MeshDMP learned on a circular pattern that is integrated while shifting the centre from $\boldsymbol{g}_{\text{start}}$ to $\boldsymbol{g}_{\text{final}}$ with a constant velocity, mimicking the human behaviour of polishing.

V. Conclusion

In this paper, we presented Mesh Dynamic Movement Primitive (MeshDMP), an approach to learn and execute motions on triangular meshes. MeshDMP treats the mesh as a discrete Riemannian manifold to perform geometry-aware learning and generalisation. To this end, we have defined differential operators on the discrete manifold, namely the logarithmic and exponential maps and the parallel transport. These operators are then used to fit DMPs that encode the motion on the manifold. We have also proposed an effective strategy to generalise the learned motion to different manifolds, by accounting for the different curvature through an isomorphic transformation which can be efficiently computed at run-time. MeshDMP has been evaluated in simulation and real experiments, where a robotic manipulator effectively performs polishing on a car fender. Future work will investigate and evaluate the use of MeshDMPs in correctly performing polishing and sanding operation in a industrial scenario.

REFERENCES

[1] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," *Springer handbook of robotics*, pp. 1995–2014, 2016.

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

[3] M. Saveriano, F. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[4] A. Ude, B. Nemec, T. Petrić, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 2997–3004.

[5] F. J. Abu-Dakka, M. Saveriano, and L. Peternel, "Periodic dmp formulation for quaternion trajectories," in *International Conference on Advanced Robotics*, 2021, pp. 658–663.

[6] M. D. Vedove, E. Lamon, D. Fontanelli, L. Palopoli, and M. Saveriano, "A passivity-based variable impedance controller for incremental learning of periodic interactive tasks," in *IEEE International Conference on Automation Science and Engineering*, 2024.

[7] F. Dimeas and Z. Doulgeri, "Progressive automation of periodic tasks on planar surfaces of unknown pose with hybrid force/position control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 5246–5252.

[8] E. Shahriari, A. Kramberger, A. Gams, A. Ude, and S. Haddadin, "Adapting to contacts: Energy tanks and task energy for passivity-based dynamic movement primitives," in *IEEE International Conference on Humanoid Robotics*, 2017, pp. 136–142.

[9] X. Xue, J. Dong, Z. Lu, and N. Wang, "A robotic learning and generalization framework for curved surface based on modified dmp," *Robotics and Autonomous Systems*, vol. 160, p. 104323, 2023.

[10] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," *Computer Graphics Forum*, vol. 36, no. 1, p. 301–329, 2016.

[11] F. Abu-Dakka, M. Saveriano, and L. Peternel, "Learning periodic skills for robotic manipulation: Insights on orientation and impedance," *Robotics and Autonomous Systems*, vol. 180, p. 104763, 2024.

[12] F. J. Abu-Dakka, M. Saveriano, and V. Kyrki, "A unified formulation of geometry-aware discrete dynamic movement primitives," *Neurocomputing*, vol. 598, p. 128056, 2024.

[13] M. Pantic, L. Ott, C. Cadena, R. Siegwart, and J. Nieto, "Mesh manifold based riemannian motion planning for omnidirectional micro aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4790–4797, 2021.

[14] M. P. Do Carmo, *Differential geometry of curves and surfaces*. Upper Saddle River, NJ: Pearson, Feb. 1976.

[15] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, "The discrete geodesic problem," *SIAM Journal on Computing*, vol. 16, no. 4, pp. 647–668, 1987.

[16] J. Chen and Y. Han, "Shortest paths on a polyhedron," in *SCG '90: Proceedings of the Sixth Annual Symposium on Computational geometry*. ACM Press, 1990, p. 360—369.

[17] S.-Q. Xin and G.-J. Wang, "Improving chen and han's algorithm on the discrete geodesic problem," *ACM Transactions on Graphics*, vol. 28, no. 4, 2009.

[18] M. Perdigao Carmo, *Riemannian Geometry*, 1st ed., ser. Mathematics: Theory & Applications. Secaucus, NJ: Birkhauser Boston, Jan. 1992.

[19] M. Kniely and W. Ring, "Riemannian methods for optimization in a shape space of triangular meshes," *Inverse Problems in Science and Engineering*, vol. 23, no. 6, pp. 1011–1039, 2015.