A Topological Extension of Movement Primitives for Curvature Modulation and Sampling of Robot Motion

Adrià Colomé · Carme Torras

Abstract This paper proposes to enrich robot motion data with trajectory curvature information. To do so, we use an approximate implementation of a topological feature named *writhe*, which measures the curling of a closed curve around itself, and its analog feature for two closed curves, namely the *linking number*. Despite these features have been established for closed curves, their definition allows for a discrete calculation that is well-defined for non-closed curves and can thus provide information about how much a robot trajectory is curling around a line in space. Such lines can be predefined by a user, observed by vision or, in our case, inferred as virtual lines in space around which the robot motion is curling.

We use these topological features to augment the data of a trajectory encapsulated as a Movement Primitive (MP). We propose a method to determine how many virtual segments best characterize a trajectory and then find such segments. This results in a generative model that permits modulating curvature to generate new samples, while still staying within the dataset distribution and being able to adapt to contextual variables.

This work has been carried out within the project CLOTHILDE ("CLOTH manIpulation Learning from DEmonstrations") funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Advanced Grant agreement No 741930). Research at IRI is also supported by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI MDM-2016-0656.

A. Colomé · C. Torras
Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Llorens i Artigas 4-6, 08028 Barcelona, Spain
Tel.: +34 9340 13383 Fax: +34 9340 15750
E-mail: {acolome, torras}@iri.upc.edu

Keywords Robot Motion · Imitation Learning · Topological Methods · Adaptive Systems

1 Introduction

Over the last years, robot motion learning has been approached by characterizing motion with different parametrizations, namely different types of movement primitives, and using direct Policy Search (PS) (Deisenroth et al (2013)) reinforcement learning methods to improve the robot's behavior from a certain initial motion, taught to the robot through either a sequence of points, kinesthetic teaching (Colomé and Torras (2018)) or other methods like motion capture or visual servoing (Jevtić et al (2018)). Such motion parametrizations are also modulated by external features.

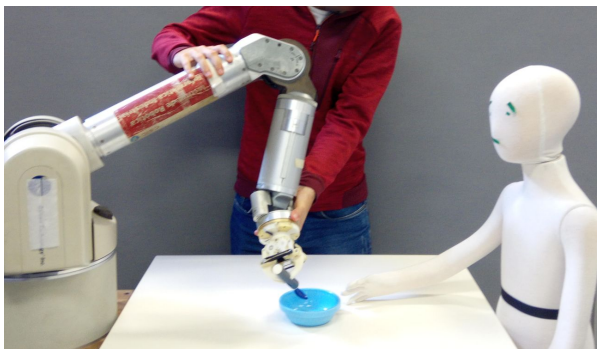

Fig. 1 A human kinesthetically teaching a robot to feed a mannequin. When adapting to a changing situation and generating new motions, it is preferable that the robot keeps a smooth motion, which can be better achieved by modulating curvature in trajectory samples.

A task example can be seen in Fig. 1, where a human kinesthetically teaches a robot how to feed a person -

substituted by a mannequin -. The locations of the food plate and the mannequin are variables to which the motion must be adapted. Moreover, fitting such motions and generating random samples from the learned dataset can sometimes result in rather unpredictable or erratic movements. While those samples can be useful from a dataset perspective, they are not well-suited for execution in a real robot in a human environment.

These initial motions taught are used by the robot to subsequently improve its behavior through trial-and-error given a certain reward function, until an optimal (or sub-optimal) policy representation is found. In certain cases, an initial learning can be performed offline through simulation and then be transferred to a real robotic platform. However, real-robot applications like manipulation of non-rigid objects are very difficult to model and, consequently, only real-robot executions can be used for learning. This issue has led roboticists to often move from big data RL to *micro-data* RL approaches, as stated by Mouret (2016). In the latter, the most important aspect of the policy representation and optimization is the real robot sample efficiency, as every sample is costly.

Within this context, Bayesian Optimization (BO) has been lately used, as in Chatzilygeroudis et al (2017), together with Dimensionality Reduction (DR) techniques (Colomé and Torras (2018)) to obtain more compact policy representations that allow to sample policies in a lower dimensional space and, therefore, to find an optimal solution with less samples. While these approaches often sacrifice the possibility of reaching a global optimum within the parameter space, local optimization has proved to be successful in many tasks (Deisenroth et al (2013)). Policy representation has then showed to be critical for an efficient reinforcement learning of a robot skill. Policies are often represented with Gaussian distributions, as in Canal et al (2018), Paraschos et al (2013). These parametric distributions capture the data moments and correlations well, but when generating samples from them, the randomness in high-dimensional spaces often produces robot motions that are not useful for optimization purposes. The well-known tradeoff between a precise fit of the data (many parameters) versus having a simpler representation with larger error (fewer parameters), presents here an additional dimension. Those samples generated with a simpler representation might not match the data well enough, while samples from larger-dimensional spaces often present higher-frequency (with large acceleration) motions, which are undesired for real robot executions. This fact often appears in kernel-based representations that do not optimize the kernel parameters - centers, widths -, such as DMPs and ProMPs, where two consecutive kernels can present large exploratory gradients of opposite signs, resulting in a fast, highly accelerated, robot motions. Using shape indicators such as how much a curve is curling around a certain reference to modulate the trajectory samples for learning, we can ensure that the trajectory samples are smoother overall.

Therefore, in this paper we propose to augment the robot motion data by capturing not only those Gaussian correlations, but also how much a trajectory is curling. These curvatures can then be sampled to generate more useful trajectories for both motion imitation and learning. The methodology, described in Sec. 3, consists in first calculating the trajectory curvatures, to then infer a number of virtual segments around which the trajectory is curling. We infer their number by applying a filtration method to the curvature radius which provides a number of candidate points that cut the trajectory into parts curling around different segments along the trajectory. These candidates are then refined through an optimization process, and the resulting parts of the trajectories are used to obtain the segments wrt. which the trajectory writhe is computed. Finally, a subsampling of the trajectory points permits evaluating its writhe wrt. those segments, yielding a writhe vector for each trajectory that can be used to augment the trajectory data encapsulated in a Movement Primitive.

Topological indexes such as writhe or the Gauss Linking Integral (GLI, reinterpreted by Ricca and Nipoti (2011)) have been used in Marzinotto et al (2014), and Zarubin et al (2013) for robot grasping, as well as for finding similarities in tangled postures between two bodies in Ho and Komura (2009). Such features were also used in robot motion by Ivan et al (2013) by examining the relative position between the robot and a curve defined by an object or the environment.The GLI was also applied to control humanoid robots in tasks involving *tangled interaction* between a robot's limbs and a manipulated object (Ho et al (2010)), as well as for data-driven inverse kinematics (Ho et al (2013)). Such works used topological coordinates to characterize the relative positioning of the interacting bodies, and finding a suitable inverse kinematics, after Ho and Komura (2007) had used similar concepts for motion planning in tangled situations. Koganti et al (2017) also uses such topological features to dress a mannequin with a shirt, while combining it with dimensionality reduction techniques and Gaussian Processes. Yuan et al (2019) have also used writhe for robotic manipulation, focused on inferring the relative positionings of bodies in human-robot interaction. Our approach is novel in the field of robot motion characterization in the sense that such writhe is not computed for extracting features between

robots and/or external bodies/agents, but rather to extract features of the robot's end-effector trajectory, relative to external geometrical shapes.

In Section 4, a generative model is built using a Gaussian distribution encompassing the joint variables of MPs weights - in particular, Probabilistic Movement Primitives - , contextual features, and the writhe vectors obtained. Such model allows to condition any of the three elements given any of the others. For example, given a context, sample the writhe and then generate a ProMP weight vector that results in a certain trajectory with such writhe and adapted to the given context.

Prior to the description of the method, we introduce some preliminaries in Section 2.

2 Preliminaries

In this section, we briefly introduce the concepts and methods used within this article. Throughout this paper, we will often refer to context variables, which are defined as observable features in the environment that affect the task being done. Context can be represented by means of real-valued variables for measurable features, or classified with an integer value. As an example, imagine a task of feeding two different types of food to a person, from two separate plates. Given a query from the person, indicating what food he/she wants, context could be represented as a vector containing the position of the food plates, the position of the person's mouth, and an integer variable indicating which food the person wants.

2.1 Probabilistic Movement Primitives (ProMPs)

ProMPs are a motion characterization that learns and encodes a set of similar motion trajectories that present time-dependent variances. Given a number of basis functions per DoF, N_f, ProMPs use time-dependent Gaussian kernels $\mathbf{\Phi}_t$ to encode the state of a trajectory, $\mathbf{\Phi}_t$ being the vector of normalized kernel basis functions (e.g., uniformly distributed Gaussian basis function over time). Thus, the position and/or velocity state vector $\mathbf{y}_t$ can be represented as

$$\mathbf{y}_t = \quad {}_t^T \, ! + \quad {}_y; \tag{1}$$

where ${}_t^T = I_d \otimes \mathbf{\Phi}_t^T$, I_d being the d-dimensional identity matrix and $\mathbf{\Phi}_t$ an N_f-dimensional column vector with the Gaussian kernels associated to one DoF at time t. Moreover, ${}_y \sim \mathcal{N}(0; \mathbf{\Sigma}_y)$ is a zero-mean Gaussian noise and the weights $!$ are also treated as random variables with a distribution

$$p(!) = \mathcal{N}(! \mid \quad {}_!; \quad {}_!): \tag{2}$$

This distribution can be fitted, given a set of demonstration trajectories ${}_j = \{\mathbf{y}_t^j\}_{t=1::N_t}$, $j = 1::N_d$, by obtaining the weights $!_j$ of each demonstration through least squares. Subsequently, the parameters of the distribution $= \{ {}_!; \quad {}_!; \quad {}_y\}$, ${}_y$ being the state covariance, are fitted by means of a maximum likelihood estimate, i.e., computing the sample mean and the sample covariance of $!$. Then the probability of observing $\mathbf{y}_t$ is:

$$\begin{aligned} p(\mathbf{y}_t; \ ) &= \int^{\mathsf{R}} \mathcal{N}(\mathbf{y}_t \mid \ {}_t^T \, !; \quad {}_y)\mathcal{N}(! \mid \quad {}_!; \quad {}_!)d! \\ &= \mathcal{N}\left(\mathbf{y}_t \mid \ {}_t^T \quad {}_!; \quad {}_y + \ {}_t^T \quad {}_! \quad {}_t\right) \end{aligned} \tag{3}$$

Due to their probabilistic nature, ProMPs can represent motion variability while keeping other MP properties such as rescalation and linear representation wrt. parameters. ProMPs also allow for other operations such as modulation via probabilistic conditioning and combination by product, as well as providing a model-based stochastic controller that reproduces the encoded trajectory distribution Paraschos et al (2013). ProMPs are capable of encapsulating variability and correlations throughout a trajectory.

2.2 Curvature computation

Throughout this paper, we will need to compute the curvature of a differentiable parametrized trajectory $\mathbf{y}(t)$. Such curvature can be obtained as:

$$= \frac{\|\dot{\mathbf{y}}_t \times \ddot{\mathbf{y}}_t\|}{\|\dot{\mathbf{y}}_t\|^3}; \tag{4}$$

$R = 1=$ being the Radius of curvature. Then, the center of curvature is:

$$\mathbf{C}_t = \mathbf{y}_t + R \cdot \mathbf{n}; \tag{5}$$

where $\mathbf{n}$ is the direction to the center of curvature and can be obtained for each point y_t: let $\mathbf{a} = y_t - y_{t\ 1}$, $\mathbf{b} = y_{t+1} - y_t$ and $\mathbf{c} = \mathbf{b} \times \mathbf{a}$. Then:

$$\mathbf{n} = \frac{\dot{\mathbf{y}}_t \times \mathbf{c}}{\|\dot{\mathbf{y}}_t \times \mathbf{c}\|}: \tag{6}$$

Note that, in the case of gathered data, we will compute the derivatives numerically with

$$\dot{\mathbf{y}}_t = \frac{\mathbf{y}_{t+dt} - \mathbf{y}_t}{dt}; \tag{7}$$

while in the case of ProMP fitting we can compute the derivative of $\mathbf{y}_t = \ {}_t^T \, !$ as $\dot{\mathbf{y}}_t = \ {}_t^{\dot{T}} \, !$.

2.3 Calculating the linking number and writhe

The writhe number is a quantity that describes how much a closed, simple curve $_1$ revolves around itself. It is very similar to the linking number, which describes how many times two closed curves $_1$; $_2$ wind around each other. In fact, both numbers are computed with the Gauss Linking Integral (GLI), being the linking number de ned as:

$$\text{link}(\ _1;\ _2) = \frac{1}{4} \int_{\ _1} \int_{\ _2} \frac{r_1 \quad r_2}{jr_1 \quad r_2 j^3} \ (dr_1 \quad dr_2); \quad (8)$$

and the writhe number using the same expression, but integrating through the same curve for dr_1 and dr_2. The highlight in Eq.(8) is that we can compute a linking (or writhe) value for any pair of segments, without integrating them throughout a closed curve. This provides us with information about how much a trajectory is curling around a given segment (or set of segments). As the calculations are the same, we do not distinguish between writhe and linking number throughout this paper.

Focusing on our application, we want to evaluate the linking integral between a trajectory and a given segment in space. To do that, we can partition our trajectory in small segments and evaluate the discretization of the integral in Eq. (8). Thus, given two segments AB and CD in a three-dimensional space, their writhe w can be obtained by de ning:

$$\begin{array}{l} n_1 = AC \quad AD \\ n_2 = BD \quad BC \\ n_3 = BC \quad AC \\ n_4 = AD \quad BD \end{array}; \quad (9)$$

and then computing writhe as Yuan et al (2019):

$$w = \text{sign}\,(AB(AC \quad CD))\,[\text{asin}\,\tfrac{n_1^T n_4}{kn_1^T n_4 k} + \text{asin}\,\tfrac{n_2^T n_3}{kn_2^T n_3 k} + \text{asin}\,\tfrac{n_3^T n_1}{kn_3^T n_1 k} + \text{asin}\,\tfrac{n_2^T n_4}{kn_2^T n_4 k}] \quad (10)$$

2.4 Conditional Gaussian Distributions

In order to build a generative model, we will be using a conditional Gaussian distribution. We de ne a random variable vector by appending context features s and the ProMP weights $!$ to the writhe values w:

$$x = \begin{bmatrix} w \\ s \\ ! \end{bmatrix} = N\left(@\begin{bmatrix} w \\ s \\ ! \end{bmatrix}; \begin{bmatrix} w & ws & w! \\ sw & s & s! \\ !w & !s & ! \end{bmatrix} A\right): \quad (11)$$

This distribution allows to condition any of the three variables given any of the others. In particular, we are interested in generating a trajectory - $!$ - given a current context s, which is assumed to be normally distributed, and a sampled or perturbed writhe w.

Let $a = \begin{bmatrix} w \\ s \end{bmatrix}$ and $_a = \begin{bmatrix} w \\ s \end{bmatrix}$, with a covariance $_a = \begin{bmatrix} w & ws \\ sw & s \end{bmatrix}$.

Then, the ProMP weights can be obtained by operating in Gaussian distributions (see Bishop (2006)):

$$p(! \ ja) = = N\left(! \ j\ ! + \ !a \ a^{1}(a \quad a);\ ! \quad !a \ a^{1} \ ^T!a\right) \quad (12)$$

Given these preliminary tools, in the following section we de ne the methodology for obtaining the writhe representation of robot motion data.

3 Writhe-Augmented ProMPs

In this section, we present a methodology for, given a certain dataset of N_k robot motion trajectories aiming at performing the same task in the Cartesian 3D space$f Y_k g$, each with an associated contextual feature value s_k that can be observed, adding the writhe of the trajectory wrt. certain segments that we can compute, which tell us how curved the trajectory is. In Section 3.1, we rstly average the data, removing contextual dependencies as much as possible in order to nd the segments from which to compute the writhe. Then, we nd such segments in Section 3.2, and compute their writhe values in Section 3.3. We will use this model as a generative model in Section 4. This section is developed for contextualized trajectories, although the procedure for context-free data is equivalent by just omitting contextual components.

3.1 Decontextualizing data

In order to augment a dataset of robot motion with writhe parameters, we need to nd certain segments, common for all trajectories, to evaluate the twisting of the trajectory wrt. them. Context-dependent trajectories such as pick and place, have strong dependencies on, for instance, the position of the object being picked. On the one hand, if we would pick di erent segments for each trajectory, those would not be comparable and, on the other hand, nding segments with all the raw data would result in averaging the e ect of each trajectory, which would not have a good result. Now, let us assume each trajectory Y_k can be tted to a parameter vector

$!_k$ by using Gaussian kernels as in Eq. (1). Then, we concatenate $!_k$ with the contextual variable s to obtain a vector $x = \begin{pmatrix} ! \\ s \end{pmatrix}$. From the data, we then have a set of x_k, for $k = 1::N_k$, that can be t into a joint distribution of $!$ and s: $p(x) = N(\ _x;\ _x)$:

$$p(x) = p\begin{pmatrix} ! \\ s \end{pmatrix} = N\begin{pmatrix} ! \\ s \end{pmatrix}; \begin{pmatrix} ! & !s \\ !s^T & s \end{pmatrix} : \quad (13)$$

From Eq. (13), we can remove the contextual dependency of the MPs weights (see Eq. (1)) bydeconditioning each trajectory k with weight vector $!_k$ from its context s_k. So by taking Eq. (12) we have:

$$!_k = \ _k + \ _{!s}\ _s^{-1}(s_k \ \ _s); \quad (14)$$

where the rst term $\ _k$ does not depend on the context, which is added by the second term $\ _{!s}\ _s^{-1}(s_k \ \ _s)$. Given that $!_k$ and the second term on the right side are obtained from the trajectory and context, respectively, we can extract the contextual dependency from the ProMP weights of a sample $!_k$ and obtain the decontextualized trajectory with ProMP weights $\ _k$:

$$_k = !_k \ \ _{!s}\ _s^{-1}(s_k \ \ _s); \quad (15)$$

yielding the decontextualized trajectory:

$$z_t^k = \ _t^T\ _k = y_t^k \ \ _t^T\ _{!s}\ _s^{-1}(s_k \ \ _s) \quad (16)$$

These decontextualized trajectories are in the same reference context, and are useful to obtain the reference segments for the writhe computations in the following subsection. To do so, we will rely on the mean decontextualized trajectory, i.e., the mean of , which generates a trajectory Y . In Fig. 2 we see an example of robot trajectories (taught by setting a robot in gravity compensation and moving it while recording its position) starting at a randomized point, touching a table and ending in a certain position. Both the table position and the ending positions are set as contextual variables, and the set of trajectories are decontextualized as seen in Fig. 3, where the mid-point and end-point have collapsed as a result of the decontextualization e ect. In this case, the decontextualization can also be seen as the quotient space of the trajectories given the two contextual points.

In the following subsection, we describe how to rstly obtain the reference segments and quantify the curling of a trajectory around them.

Fig. 2 Set of trained trajectories in the Cartesian space (scale in meters). Mean trajectory shown as a blue dotted line. These motions were taught kinesthetically for a feeding task, as illustrated in Fig. 1.

Fig. 3 Trained trajectories in the Cartesian space after decontextualization with Eqs. (15) and (16). Mean trajectory shown as a blue dotted line.

3.2 Finding segments to compute writhe

The aim of this section is to obtain a number of segments in the Cartesian space around which the trajectory curls (see Fig. 9). While other methods in literature can decompose curves into di erent parts with di erent shapes (as in Rosin and West (1995)), our application here is more speci c. After a long time working with robot motion learning, we found that many robotic tasks have geometrical/topological properties that are not considered, thus favouring pure numerical analysis. Some motions curl around visible or invisible lines in the

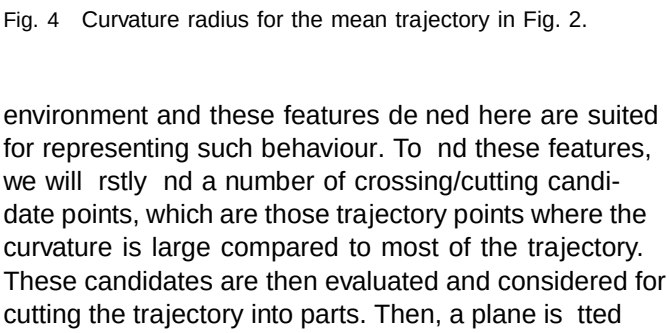

Fig. 4 Curvature radius for the mean trajectory in Fig. 2.

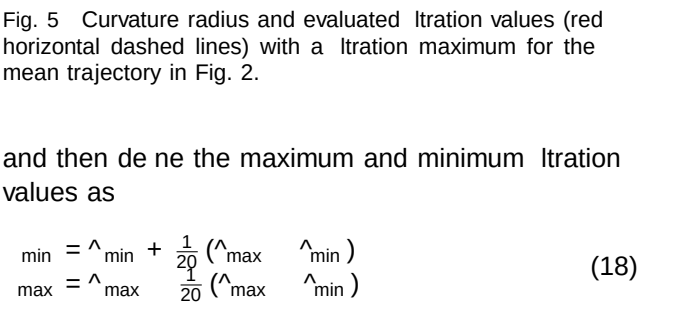

Fig. 5 Curvature radius and evaluated filtration values (red horizontal dashed lines) with a filtration maximum for the mean trajectory in Fig. 2.

environment and these features defined here are suited for representing such behaviour. To find these features, we will firstly find a number of crossing/cutting candidate points, which are those trajectory points where the curvature is large compared to most of the trajectory. These candidates are then evaluated and considered for cutting the trajectory into parts. Then, a plane is fitted to each trajectory part, whose normal vector combined with the mean center of curvature and mean radius of the trajectory part being considered, will form a segment.

We will firstly find the center of curvature of every point of the mean decontextualized trajectory Y obtained in the previous section. The curvature (and therefore the radius of curvature R) of the trajectory at point y_t is obtained with Eq. (4). Then, we can obtain the sequence of radius of curvature $R(t)$ at every timestep (see Fig. 4).

We detect parts of the trajectory in which the radius is very large. In many situations, this can be understood as a change in the trajectory, resulting in a different region for the centers of curvature. In the curvature radius plot (Fig. 4), we can already see that there is one clear peak of curvature, which corresponds to the point touching the table for the motion trajectories in Fig. 2.

We will then obtain a filtration for the curvature radius as seen in Fig. 5. To do so, we test, for filtration values $\in [\hat{}_{min}; \hat{}_{max}]$, where the minimum and maximum values $\hat{}_{min}$, $\hat{}_{max}$ of are given by the min and max radius of curvature of the trajectory, plus an additional 5% of the total radius span, i.e., we firstly find:

$$\begin{aligned} \hat{}_{min} &= \min_t(R(t)) \\ \hat{}_{max} &= \max_t(R(t)); \end{aligned} \tag{17}$$

and then define the maximum and minimum filtration values as

$$\begin{aligned} _{min} &= \hat{}_{min} + \tfrac{1}{20}(\hat{}_{max} - \hat{}_{min}) \\ _{max} &= \hat{}_{max} - \tfrac{1}{20}(\hat{}_{max} - \hat{}_{min}) \end{aligned} \tag{18}$$

For all the tested filtrations, we decompose the curvature radius plot in Fig. 4 into segments between each two consecutive points, and check if such segments cross the horizontal line corresponding to the filtration value (red dotted lines in Fig. 5). We store the data index in which there is a crossing, and while we increment the filtration value, we check if it stills crosses the same horizontal line until it stops doing so. At that point, we store this cross candidate c with its datapoint index, its maximum filtration in which it is crossed by the horizontal red line (as seen in Fig. 5). However, we only store those candidates whose maximum filtration value is at least the mean of all the curvature radius, plus one standard deviation. This prevents the algorithm from selecting too many candidates and thus building a too large set of candidates. After finishing the filtration, we gather all cross candidates c in the Crossing Set C, which will have a cardinal number $\#C$ equal to the total number of crossing candidates. Another example of this method can be seen in the trajectory in Fig. 6 which, looking at Figs. 7 and 8, seems to have two crossing candidates, marked in Fig. 6 as green points.

Given the set of all crossing candidates, after removing repeated occurrences that might occur, we evaluate which choice of the candidates suits best the trajectory. To that endeavour, we evaluate all the possible combinations within the power set of C, namely $P(C)$. Each element p of $P(C)$ consists of a certain number of point candidates to be the separation between parts of the trajectory curling around centers in different regions.

Fig. 6 Another example of a robot motion trajectory, with its extracted cutting points.

Fig. 7 Curvature radius for the mean trajectory in Fig. 6.

Then, we want to nd

$$p = \text{argmax}_{p \in P(C)} h(P(C)); \quad (19)$$

where h is a cost function de ned as:

$$h(p) = D(p) \left(1 + \frac{\#p}{M}\right); \quad (20)$$

where $\#p$ is the number of point candidates in p, M is the total number of point candidates found through ltration, i.e., $\#C$, and $D(p)$ is evaluated using the procedure summarized in Alg. 1. We rstly t a plane to every part j of the trajectory, given by the separation points in p, and then we calculate the squared sum of the distance from each point of that part of the trajectory to the plane. However, in tting the plane we wanted it to be the best tting both the trajectory part points and their centers of curvature, but close to transition areas, the centers of curvature may become outliers that negatively a ect the tting. Therefore, we rst perform a k-means clustering of the centers of curvature and nd the dominant cluster that will exclude

Fig. 8 Curvature radius with ltration maxima for the mean trajectory in Fig. 6.

Fig. 9 Writhe segments (black lines) computed with Alg. 2 for the trajectory data in Fig. 2. The perspective has been moved for a better view.

Algorithm 1 Evaluating $D(p)$ for $p \in P(C)$

1: Input: Partitioning of a trajectory by the points indicated in p, trajectory data Y and centers of curvature Q
2: Set $D(p) = 0$
3: for all $c_j \in p$ do
4: Use the part of the trajectory before point c_j : $Y^j = \{y_t; t = t_{j0}; \ldots; t_{j1}\}$ and their corresponding centers of curvature $Q^j = \{_t; t = t_{j0}; \ldots; t_{j1}\}$
5: Perform k-means on the centers of curvature and keep the dominant cluster cl.
6: Find the plane $_j$ tting the points in Y^j and Q^j_{cl} (centers of curvature in the dominant cluster)
7: $D_j = \sum_{t=t_{j1}}^{t=t_{j0}} \|y_t - \text{Proj}(y_t; _j)\|^2$
8: $D(p) = D(p) + D_j$
9: Output: $D(p)$, tting planes $= \{_j \mid \forall c_j \in C\}$, and clusterized centers and radius of curvature $Q_{cl} = \{Q^j_{cl} \mid c_j \in C\}$, $R_{cl} = \{R^j_{cl} \mid \forall c_j \in C\}$

such outliers. Then, we use this cluster with the trajectory points to t a plane $_j$. Additionally, we penalize the use of too many partitions of the trajectory with the term $\left(1 + \frac{\#p}{M}\right)$, which goes from 1 to 2.

Once we have found which is the best partitioning by solving Eq. (19), we can nally de ne the segment

Fig. 10 An oscillatory trajectory, consisting on a robot shaking an object, in a 3-dimensional space. A 3D plot (left), the X-axis projection (middle) and the Z-axis projection (right). In green, the points obtained by Alg. 2 to separate the trajectory into parts, given the filtration on curvature as in Fig. 11.

Algorithm 2 Find segment given plane, centers and radius of curvature

1: Input: For the dominant cluster found in Alg. 1, the centers of curvature $Q_{cl}^j = \{q_t; t = t_{j0}, \dots, t_{j1}\}$, radius of curvature $R_{cl}^j = \{R_t; t = t_{j0}, \dots, t_{j1}\}$ and fitting plane $_j$ for a given partitioning of the trajectory.
2: Compute the mean of the curvature centers $\bar{q}_j$
3: Compute the mean of the radius of curvature $\bar{R}_j$
4: Find the segment SEGM $_j$ with Eq.(21)

Fig. 11 Curvature radius with filtration maxima for the oscillatory trajectory in Fig. 10.

for each trajectory part j given the fitting planes $_j$, by calculating the mean point $\bar{q}$ of the centers of curvature $q^j = \{q_t; t = t_{j0}, \dots, t_{j1}\}$ corresponding to that trajectory part, and from there, the segment is defined as:

$$\text{SEGM}_j = \left\{ \bar{q}^j - \frac{\bar{R}_j}{2} n_j ; \bar{q}^j + \frac{\bar{R}_j}{2} n_j \right\}; \quad (21)$$

where n_j is the vector normal to the plane $_j$, and $\bar{R}_j$ is the mean radius of curvature for that part of the trajectory. Algorithm 2 summarizes this procedure, and we can see the resulting segments of the data in Figs. 2-5 in Fig. 9, where the perspective has been rotated to have a better view of them.

Figure 10 shows a more complex trajectory that is generated by a single sample, and two of its projections for a better perspective. Its curvature filtration plot is displayed in 11, and its optimization ends up with the points marked back in Fig. 10, showing the usefulness of the filtration for finding changes of direction. Note that the first trajectory peak has not been selected, as selecting it would increase complexity without providing much information, since it is a rather coplanar part of the trajectory as we observe in the Z-projection plot.

3.3 Computing writhe components

After obtaining the segments around which the trajectory is curling, we generate a writhe vector by comparing the trajectory against them. The general process is described in Alg. 3, where trajectories are sub-sampled as a sequence of M segments (always at least twice the number N_W of writhe segments obtained). This is done to prevent the method from having as many writhe components as samples in the trajectory. We empirically saw that a subsampling is enough to capture information for a generative model. Then, these M segments are compared with the N_W writhe segments with Eq. (10) and we generate an $\mathbb{R}^{M \cdot N_W}$ vector w.

4 Building a Generative Model

Using the segments obtained in the previous section, we will now build a generative model, as hinted in Section 2.4. For each trajectory sample k consisting of a series of time-stamped datapoints y_t^k and a corresponding context variable that we can measure s_k in the data, we will firstly compute the Gaussian kernel weights $! _k$ by maximum likelihood techniques as in Eq. (1). Then, by using Alg. 3, we obtain the writhe of each trajectory.

Algorithm 3 Obtaining trajectory writhe

1: Input: Data trajectories Y^k with their corresponding context s_k
2: for all $k = 1::N_{demos}$ do
3: Fit linear weights $!_k$ as for Eq.(1)
4: Decontextualize trajectories with Eq. (16) and obtain the mean decontextualized trajectory $\overline{Y}_{decon}$ Obtain mean trajectory
5: Compute curvatures q_t for the mean trajectory with Eqs. (4), (5)
6: Do ltration with curvatures and obtain cross candidates set C
7: Find best partitioning p with (19)
8: Obtain the writhe segments with Eq.(21)
9: for all $k = 1::N_{demos}$ do
10: Compute k-th trajectory writhe w with Eq.(10)

This results in having, for each sample k, three types of data: writhe values w_k, context variables s_k and kernel weights $!_k$. We append the three and create a random variable x as in Eq. (11), under the assumption that the context feature s and writhe w follow a Normal distribution (this is already assumed for $!$ in building such linear model as a ProMP):

$$x = \begin{bmatrix} w \\ s \\ ! \end{bmatrix} = N\left(\begin{bmatrix} w \\ s \end{bmatrix}; \begin{bmatrix} w & ws & w! \\ sw & s & s! \\ !w & !s & ! \end{bmatrix}\right): \quad (22)$$

The model obtained is a generative one in the sense that $!$ can be conditioned to a given context or a perturbation of the writhe, and then generate new trajectories Y through Eq. (1). The model is de ned for contextualized trajectories, although it can be used regardless of whether these contextual features exist or not.

One approach to using this model is to impose a change in the writhe of a certain part of the trajectory and then generate a new trajectory after obtaining the conditional $p(! \mid w)$ with Eq. (12). Then use Eq. (1) for obtaining the trajectory points. We applied this approach to the data from Fig. 2, and Figs. 12 and 13 show how the trajectory is perturbed by applying a step increase in di erent components. Note that in Fig. 13, the perturbation of the nal part of the trajectory is also correlated with a change in the rest of the trajectory. This is due to the non-diagonal correlation matrices in Eq. (22). The output trajectories are those that keep the rest of the writhe components equal - remind that the distribution of trajectories, given a writhe value, can have several free dimensions -, while only changing the last term of the writhe vector.

Another approach to use this generative model is, given a context s, to obtain the representation of the writhe distribution using Eq. (12), sample writhe w (or either take the mean $_{wjs}$), and proceed similarly as in the previous case. This results in an e cient sampling that provides trajectories which are smoother and meaningful. Both for the case without contextual variables (Fig. 14) and with contextual variables (Fig. 15).

Moreover, we used the model in Fig. 15 and generated 1000 random trajectories by sampling a context s (which would be the plate position as seen in Fig. 1) , and obtaining a weight vector with Eq. (12). For the same context valuess, we used the model in (11) and sampled the writhe vector given the context, to then sample the weight vector $!$ given the context and writhe. We computed the Mean Squared Acceleration (MSA), i.e.: the average squared norm of the trajectories' acceleration. We used such indicator as it gives an idea of how smooth trajectory samples are, and found that those trajectories sampled without the writhe components had a mean and standard deviation

Fig. 12 Writhe perturbations for the rst part of the trajectory vs the rst writhe segment computed with Alg. 2 for the trajectory data in Fig. 9. The perspective has been moved for a better view.

Fig. 13 Writhe perturbations for the nal part of the trajectory vs the second writhe segment computed with Alg. 2 for the trajectory data in Fig. 9. The perspective has been moved for a better view.

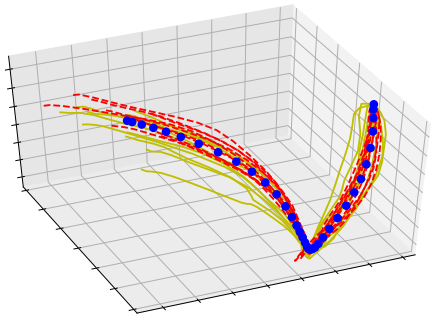

Fig. 14 Samples generated with our model without context dependency (in dashed red lines) and data (yellow). The writhe is sampled from a Gaussian distribution.

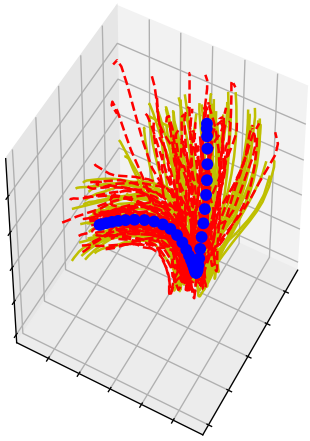

Fig. 15 Context-dependent samples generated with our model (in dashed red lines) and data (in yellow). The samples are conditioned on a certain context (mid- and end-points) and generated according to our model.

MSA of 20:22 ± 8:03, while using writhe modulation yielded 9:78 ± 3:81, which is a very significant drop in accelerations for the generated samples.

We also tested a learning scenario for the task in Fig. 1 and Figs. 12-15. Without the contextual dependency, we defined a new goal G for the location of the plate and defined a reward function for each trajectory as:

$$R = -\text{distance}(\ ; G) - MSA(\ )/50; \tag{23}$$

where distance is defined as the minimum Euclidean distance between the trajectory and the point G. We added an acceleration term with a factor of 1/50 that yielded a similar relevance for both terms in the reward for the early samples. Then, we performed an update of the model after every 20 samples, using a Policy Search algorithm named Relative Entropy Policy Search (REPS) (Peters et al (2010)), and updated the model 10 times. We performed this experiment 10 times by generating samples with and without writhe modulation, and the mean results with two standard deviations are shown in Fig. 16, where the blue learning curve (writhe modulation) yields a slightly better result than the baseline method. While the results do not show a significant improvement over the baseline in this application, it does show that smoother and more intuitive samples can be generated.

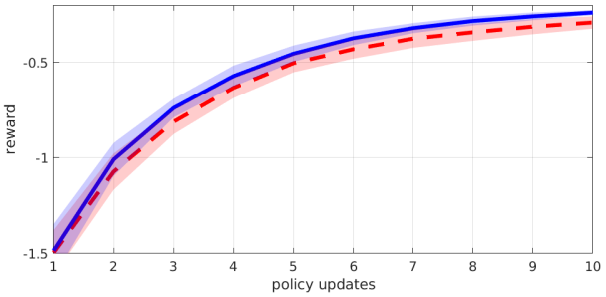

Fig. 16 Learning curve of a reinforcement learning application with (blue) and without (dashed red) writhe modulation. The plots show the average reward vs number of policy updates, averaged over a set of 10 experiments each, showing mean and two standard deviations

5 Conclusions and future work

This paper presents a very novel attempt at motion characterization by suggesting writhe as a useful feature for robot trajectory modulation. The derivations and results show that writhe can be successfully used for robot motion characterization and thus opens a new path for using topological features in robot motion learning and generation. While the applicability of this method is limited to those motions from which geometrical features can be extracted, the results show that writhe modulation is capable of providing samples with a significant reduction in accelerations, thus safer trajectories, while keeping or even slightly improving the efficiency of learning with them.

The proposed generative method is also robust to contextual perturbations, and we chose a conditional Gaussian distribution, despite having attempted to implement a Gaussian Processes approach directly mapping writhe and time to a Cartesian pose. Such approach fails with conventional Gaussian Processes (Rasmussen and Williams (2005)), mainly because the generalization required is too large. Thus, Bayesian Gaussian Processes is envisaged to be the next step in using the newly defined feature. Additionally, it can be combined with vision algorithms to find relevant geometries in the environment, as done by Pumarola et al (2017) with an application to SLAM. Finally, the methodol-

ogy can be applied with dimensionality reduction techniques, such as Gaussian Process Latent Variable Models (GPLVM) (Li and Chen (2016)) in order to obtain a reduced-dimension feature space for policy learning, as in the works of Koganti et al (2017), Koganti et al (2019) and Delgado-Guerrero et al (2020).

References

Bishop CM (2006) Pattern Recognition and Machine Learning (Information Science and Statistics). *Springer-Verlag*, Berlin, Heidelberg

Canal G, Pignat E, Alenyà G, Calinon S, Torras C (2018) Joining high-level symbolic planning with low-level motion primitives in adaptive hri: Application to dressing assistance. *IEEE International Conference on Robotics and Automation (ICRA)* pp 3273–3278

Chatzilygeroudis K, Rama R, Kaushik R, Goepp D, Vassiliades V, Mouret JB (2017) Black-box data-efficient policy search for robotics. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp 51–58

Colomé A, Torras C (2018) Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes. *IEEE Transactions on Robotics* 34(3):602–615

Deisenroth MP, Neumann G, Peters J (2013) A survey on policy search for robotics. *Foundations and Trends in Robotics* 2:1–142

Delgado-Guerrero JA, Colomé A, Torras C (2020) Sample-efficient robot motion learning using gaussian process latent variable models. *IEEE International Conference on Robotics and Automation (ICRA)* pp 314–320

Ho E, Komura T, Ramamoorthy S, Vijayakumar S (2010) Controlling humanoid robots in topology coordinates. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp 178–182

Ho E, Shum H, Cheung Ym, Yuen PC (2013) Topology aware data-driven inverse kinematics. *Computer Graphics Forum* 32(7):61–70

Ho ES, Komura T (2009) Indexing and retrieving motions of characters in close contact. *IEEE Transactions on Visualization and Computer Graphics* 15(3):481–492

Ho ESL, Komura T (2007) Planning tangling motions for humanoids. *IEEE-RAS International Conference on Humanoid Robots,* pp 507–512

Ivan V, Zarubin D, Toussaint M, Komura T, Vijayakumar S (2013) Topology-based representations for motion planning and generalization in dynamic environments with interactions. *The International Journal of Robotics Research* 32(9-10):1151–1163

Jevtić A, Colomé A, Alenyà G, Torras C (2018) Robot motion adaptation through user intervention and reinforcement learning. *Pattern Recognition Letters* 105:67–75

Koganti N, Tamei T, Ikeda K, Shibata T (2017) Bayesian nonparametric learning of cloth models for real-time state estimation. *IEEE Transactions on Robotics* 33(4):916–931

Koganti N, Shibata T, Tamei T, Ikeda K (2019) Data-efficient learning of robotic clothing assistance using bayesian gaussian process latent variable model. *Advanced Robotics* 33(15-16):800–814

Li P, Chen S (2016) A review on gaussian process latent variable models. *CAAI Transactions on Intelligence Technology* 1(4):366 – 376

Marzinotto A, Stork JA, Dimarogonas DV, Kragic D (2014) Cooperative grasping through topological object representation. *IEEE-RAS International Conference on Humanoid Robots,* pp 685–692

Mouret J (2016) Micro-data learning: The other end of the spectrum. *Computing Research Repository* abs/1610.00946

Paraschos A, Daniel C, Peters J, Neumann G (2013) Probabilistic movement primitives. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pp 2616–2624

Peters J, Mülling K, Altün Y (2010) Relative entropy policy search. *AAAI Conference on Artificial Intelligence* 24(1):1607–1612

Pumarola A, Vakhitov A, Agudo A, Sanfeliu A, Moreno-Noguer F (2017) PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines. In: *IEEE International Conference in Robotics and Automation (ICRA)*, pp 4503–4508

Rasmussen CE, Williams CKI (2005) Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). *The MIT Press*

Ricca R, Nipoti B (2011) Gauss' linking number revisited. *Journal of Knot Theory and its Ramifications* 10

Rosin PL, West GAW (1995) Nonparametric segmentation of curves into various representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(12):1140–1153

Yuan W, Hang K, Song H, Kragic D, Wang MY, Stork JA (2019) Reinforcement learning in topology-based representation for human body movement with whole arm manipulation. *IEEE International Conference on Robotics and Automation (ICRA)* pp 2153–2160