Article

The International Journal of Robotics Research
2024, Vol. 43(11) 1693–1710

DOI: 10.1177/02783649241233300
journals.sagepub.com/home/ijr


Path signatures for diversity in probabilistic trajectory optimisation

Lucas Barcelos[1], Tin Lai[1], Rafael Oliveira[2], Paulo Borges[2] and Fabio Ramos[1,3]

Abstract

Motion planning can be cast as a trajectory optimisation problem where a cost is minimised as a function of the trajectory being generated. In complex environments with several obstacles and complicated geometry, this optimisation problem is usually difficult to solve and prone to local minima. However, recent advancements in computing hardware allow for parallel trajectory optimisation where multiple solutions are obtained simultaneously, each initialised from a different starting point. Unfortunately, without a strategy preventing two solutions to collapse on each other, naive parallel optimisation can suffer from mode collapse diminishing the efficiency of the approach and the likelihood of finding a global solution. In this paper, we leverage on recent advances in the theory of rough paths to devise an algorithm for parallel trajectory optimisation that promotes diversity over the range of solutions, therefore avoiding mode collapses and achieving better global properties. Our approach builds on path signatures and Hilbert space representations of trajectories and connects parallel variational inference for trajectory estimation with diversity-promoting kernels. We empirically demonstrate that this strategy achieves lower average costs than competing alternatives on a range of problems, from 2D navigation to robotic manipulators operating in cluttered environments.

Keywords

path signature, trajectory optimisation, global optimisation

Received 21 July 2023; Revised 16 January 2024; Accepted 22 January 2024
Senior Editor: Tim Barfoot
Associate Editor: Christoffer Heckman

1. Introduction

Trajectory optimisation is one of the key tools in robotic motion, used to find control signals or paths in obstacle-cluttered environments that allow the robot to perform desired tasks. These trajectories can represent a variety of applications, such as the motion of autonomous vehicles or robotic manipulators. In most problems, we consider a *state-space model*, where each distinct situation for the world is called a *state*, and the set of all possible states is called the *state space* (LaValle, 2006). When optimising candidate trajectories for planning and control, two criteria are usually considered: *optimality* and *feasibility*. Although problem dependant, in general, the latter evaluates in a binary fashion whether the paths generated respect the constraints of both the robot and the task, such as physical limits and obstacle avoidance. Conversely, optimality is a way to measure the quality of the generated trajectories with respect to task-specific desired behaviours. For example, if we are interested in smooth paths, we will search for trajectories that minimise changes in velocity and/or acceleration. The complexity of most realistic robot planning problems scales exponentially with the dimensionality of the state space and is countably infinite. When focusing on motion planning, a variety of algorithms have been proposed to find optimal and feasible trajectories. These can be roughly divided into two main paradigms: sampling-based and trajectory optimisation algorithms.

Sampling-based planning (Gammell and Strub, 2021) is a class of planners which rely on a sampling procedure to compose paths. In many cases, these approaches will be *probabilistically complete* and *asymptotically optimal* with respect to given optimality criteria (Al-Bluwi et al., 2012). These approaches decompose the planning problem into a series of sequentially sampled next steps evaluated via a tree-based (LaValle and Kuffner, 2001) or graph-based (Jaillet and Simeon, 2008; Kavraki et al., 1996) approach. However,

[1]The University of Sydney, Sydney, NSW, Australia
[2]CSIRO, Pullenvale, QLD, Australia
[3]NVIDIA, Seattle, WA, USA

Corresponding author:
Lucas Barcelos, School of Computer Science, The University of Sydney, Building J12, 1 Cleveland Street, Darlington, Sydney, NSW 2008, Australia.
Email: lucas.barcelos@sydney.edu.au

most approaches are limited in their ability to encode kinodynamic cost like trajectory curvature (Heilmeier et al., 2020) or acceleration torque limits (Berntorp et al., 2014). In addition, despite the completeness guarantee, sampling-based planners are often more computationally expensive as the search space grows and can obtain highly varying results due to the random nature of the algorithms.

Trajectory optimisation algorithms (Gonzalez et al., 2016) use different techniques to minimise a cost functional that encourages solutions to be both optimal and feasible. The most direct optimisation procedure relies on a differentiable cost function and uses functional gradient techniques to iteratively improve the trajectory quality (Ratliff et al., 2009). However, many different strategies have been proposed. For example, one may start from a randomly initialised candidate trajectory and proceed by adding random perturbations to explore the search space and generate approximate gradients, allowing any arbitrary form of cost functional to be encoded (Kalakrishnan et al., 2011). The same approach can be used to search for control signals and a local motion plan concurrently (Williams et al., 2016). Finally, a locally optimal trajectory can also be obtained via decomposing the planning problem with sequential quadratic programming (Schulman et al., 2013). A drawback of these methods is that they usually find solutions that are locally optimal and may need to be run with different initial conditions to find solutions that are feasible or with lower costs Figure 1.

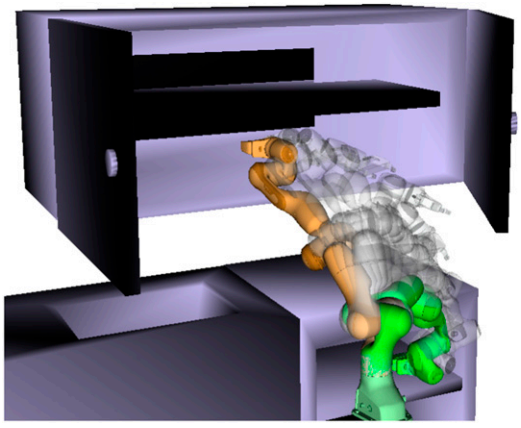

Figure 1. An episode of the *Kitchen* scene. Depicted is one of the collision-free paths found by SigSVGD on a reaching task using a 7-DOF Franka Panda arm on the MotionBenchMaker planning benchmark.

Our goal with the present work is to propose a new trajectory optimisation method to improve path diversity. More specifically, we focus on a class of algorithms that performs parallel trajectory optimisation of a batch of trajectories. This concurrent optimisation of several paths in itself already alleviates the proneness to local minima since many initial conditions are evaluated simultaneously. Nonetheless, we show how a proper representation of trajectories when performing functional optimisation leads to increased diversity and solutions with a better global property, either with direct gradients or Monte Carlo-based gradient approximations. For an illustrative example, refer to Figure 2.

Our approach is based on two cornerstones. On one hand, we use a modification of Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016), a variational inference method to approximate a posterior distribution with an empirical distribution of sampled particles, to optimise trajectories directly on a structured Reproducing Kernel Hilbert Space (RKHS).

The structure of this space is provided by the second pillar of our approach. We leverage recent advancements in rough path theory to encode the sequential nature of paths in the RKHS using a Path Signature Kernel (Kiraly and Oberhauser, 2019; Salvi et al., 2021). Therefore, we can approximate the posterior distribution over optimal trajectories with structured particles during the optimisation while still taking into account motion planning and control idiosyncrasies.

More concretely, the main contributions of this paper are listed below:

- We introduce the use of path signatures (Lyons, 2014) as a canonical feature map to represent trajectories over high-dimensional state spaces.

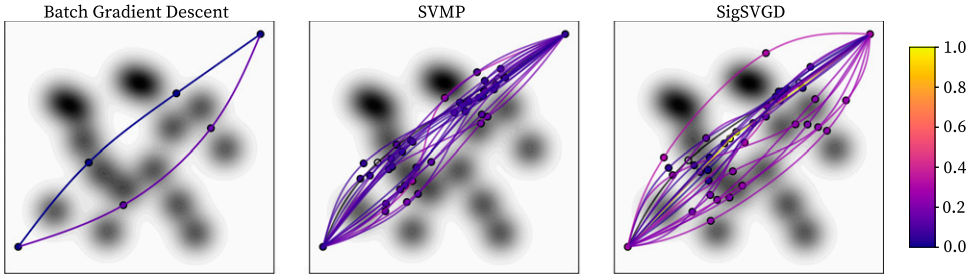

Figure 2. Qualitative analysis of 2D planning task. The plot shows the final 20 trajectories found with different optimisation methods. The colour of each path shows its normalised final cost. Note how all batch gradient descent trajectories converge to two modes of similar cost. Paths found by SVMP are already more diverse, but one of the gradient descent modes is lost. Note how when multiple trajectories converge to a single trough, the knots are pushed away by the repulsive force resulting in suboptimal solutions. Conversely, paths found by SigSVGD are diverse and able to find more homotopic solutions, including those found by BGD. Note also how paths are able to converge to the same trough without being repelled by one another since the repulsive force takes into account the entire trajectory and not exclusively the spline knot placement. That also allows for paths that are more direct and coordinated than SVMP.

- Next, we outline a procedure to incorporate the signature kernel into a variational inference framework for motion planning.
- Finally, we demonstrate through experiments in both planning and control that the proposed procedure results in more diverse trajectories, which aid in avoiding local minima and lead to a better optimisation outcome.

The paper is organised as follows. In Section 2, we review related work, contrasting the proposed method to the existing literature. In Section 3, we provide background on path signatures and motion planning as variational inference, which are the foundational knowledge for the method outlined in Section 4. Finally, in Section 5, we present a number of simulated experiments, followed by relevant discussions in Section 6.

2. Related work

Trajectory optimisation refers to a class of algorithms that start from an initial sub-optimal path and find a, possibly local, optimal solution by minimising a cost function. Given its broad definition, there are many seminal works in the area. One influential early work is Covariant Hamiltonian Optimisation for Motion Planning (CHOMP) (Ratliff et al., 2009) and related methods (Zucker et al., 2013; Byravan et al., 2014; Marinho et al., 2016). The algorithm leverages the covariance of trajectories coupled with Hamiltonian Monte Carlo to perform annealed functional gradient descent. However, one of the limitations of CHOMP and related approaches is the need for a fully-differentiable cost function.

In Stochastic Trajectory Optimisation for Motion Planning (STOMP) (Kalakrishnan et al., 2011), the authors address this by approximating the gradient from stochastic samples of noisy trajectories, allowing for non-differentiable costs. Another approach used in motion planning is quality diversity algorithms, at the intersection of optimisation and evolutionary strategies, of which Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is the most prominent (Hansen et al., 2003; Hämäläinen et al., 2020; Tjanaka et al., 2022). CMA-ES is a derivative-free method that uses a multivariate normal distribution to generate and update a set of candidate solutions, called individuals. The algorithm adapts the covariance matrix of the distribution based on the observed fitness values of the individuals and the search history, balancing exploration and exploitation of the search space. Because of its stochastic nature, it is ergodic and copes well with multi-modal problems. Nonetheless, it may require multiple initialisations, and it typically requires more evaluations than gradient-based optimisers (Hansen, 2016).

TrajOpt (Schulman et al., 2013), another prominent planner, adopts a different approach solving a sequential quadratic program and performing continuous-time collision checking. Contrary to sampling-based planners, these trajectory optimisation methods are fast but only find locally optimal solutions and may require reiterations until a feasible solution is found. Another issue common to these approaches is that, in practice, they require a fixed and fine parametrisation of trajectory waypoints to ensure feasibility and smoothness, which negates the benefit of working on continuous trajectory space. To address this constraint, in Marinho et al. (2016), the authors restrict the optimisation and trajectory projection to an RKHS with an associated squared-exponential kernel. However, the cost between sparse waypoints is ignored, and the search is still restricted to a deterministic trajectory. Another approach was proposed in GPMP (Dong et al., 2016; Mukadam et al., 2017; Mukadam et al., 2018) by representing trajectories as Gaussian Processes (GP) and looking for a *maximum a posteriori* (MAP) solution of the inference problem.

More closely related to our approach are Lambert and Boots (2021) and Yu and Chen (2022) which frame motion planning as a variational inference problem and try to estimate the posterior distribution represented as a set of trajectories. In Yu and Chen (2022), the authors modify GPMP with a natural gradient update rule to approximate the posterior. On the other hand, in Stein Variational Motion Planning (SVMP) (Lambert and Boots, 2021), the posterior inference is optimised using Stein variational gradient descent. This method is similar to ours, but the induced RKHS does not take into account the sequential nature of the paths being represented but rather treats paths similar to static vectors on a high-dimensional space. This approach leads to a diminished repulsive force and lack of coordination along the dimensions of the projected space.

In contrast, our approach—which we will refer to as Kernel Signature Variational Gradient Descent (SigSVGD)—uses the path signature to encode the sequential nature of the functional being optimised. We argue that this approach leads to a better representation of trajectories promoting diversity and finding better local solutions. To empirically corroborate this claim, we use Occam's razor principle and take SVMP as the main baseline of comparison since it more closely approximates our method.

We note that the application of trajectory optimisation need not be restricted to motion planning. By removing the constraint of a target state and making the optimisation process iterative over a rolling horizon, we retrieve a wide class of Model Predictive Controllers with applications in robotics (Barcelos et al., 2021; Barcelos et al., 2020; Lambert et al., 2020; Williams et al., 2016). Stein Variational MPC (SVMPC) (Lambert et al., 2020) uses variational inference with SVGD optimisation to approximate a posterior over control policies and more closely resembles SigSVGD. However, like SVMP, it too does not take into account the sequential nature of control trajectories, and we will illustrate how our approach can improve the sampling of the control space and promote better policies.

3. Background

3.1. Trajectory optimisation in robotics

Consider a system with state $\mathbf{x} \in \mathcal{X}$, and let us denote a *trajectory* of such a system as $X : [a, b] \to \mathcal{X}$, where $\mathcal{X}$ is an

appropriate Euclidean space or group. We shall use the notation X_t to denote the dependency on time $t \in [a, b]$. The trajectory X describes a *path* in $\mathcal{X}$, and we shall use the two denominations interchangeably. In trajectory optimisation, the goal is to find the optimal path X^* from a given starting state $\mathbf{x}_s$ to a certain goal state $\mathbf{x}_g$. This can be done by minimising a cost functional that codifies our desired behaviour $\mathcal{C} : \mathcal{P}_\mathcal{X} \to \mathbb{R}^+$, where $\mathcal{P}_\mathcal{X}$ is the Hilbert space of trajectories (King et al., 2013):

$$X^* := \arg\min_X \mathcal{C}(X), \text{s.t.} \, X_a = \mathbf{x}_s \, \text{and} \, X_b = \mathbf{x}_g. \tag{1}$$

Typically, $\mathcal{C}$ is a bespoke functional that includes penalties for trajectory non-smoothness, total energy, speed, and acceleration tracking, as well as length. To ensure that the solution is feasible and collision-free, additional equality and inequality constraints may also be included (Schulman et al., 2013). Alternatively, we can solve an unconstrained problem and include additional penalties to the cost functional as soft constraints (Ratliff et al., 2009; Zucker et al., 2013).

Finally, we draw the reader's attention to the fact that the problem stated in equation (1) can be viewed as an open-loop optimal control problem. If the solution can be found in a timely manner, the same problem can be cast onto a Model Predictive Control (Barcelos et al., 2021; Barcelos et al., 2020; Camacho and Alba, 2013) framework:

$$U^* := \arg\min_U \mathcal{C}(X, U), \text{s.t.} \, X_a = \mathbf{x}_s, \tag{2}$$

where $U : [a, b] \to \mathcal{U}$ is a path of control inputs on a given Euclidean space and the mapping to $\mathcal{X}$ is given by the dynamical system $\mathbf{f}$ such that $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$. That is to say, we now influence the path X indirectly through input U, and at any time t, the problem is solved for a finite interval. The closed-loop solution arises from applying only the first immediate control action before re-optimising the solution.

3.2. Path signature

A multitude of practical data streams and time series can be regarded as a path, for example, video, sound, financial data, control signals, handwriting, etc. The path signature transforms such multivariate sequential data (which may have missing or irregularly sampled values) into an infinite-length series of real numbers that uniquely represents a trajectory through Euclidean space. Although formally distinct and with notably different properties, one useful intuition is to think of the signature of a path as akin to a Fourier transform, where paths are summarised by an infinite series of feature space coefficients. Consider a path X traversing space $\mathcal{X} \subseteq \mathbb{R}^c$ as defined in Section 3.1. Note that at any time t, such path can be decomposed in $X_t = \{X_t^1, X_t^2, \ldots, X_t^c\}$. Now, recall that for a one-dimensional path X_t and a function f, the path integral of f along X is defined by:

$$\int_a^b f(X_t) \mathrm{d}X_t = \int_a^b f(X_t) \dot{X}_t \mathrm{d}t. \tag{3}$$

In particular, note that the mapping $t \to f(X_t)$ is also a path. In fact, equation (3) is an instance of the Riemann–Stieltjes integral (Chevyrev and Kormilitzin, 2016), which computes the integral of one path against another. Let us now define the *1-fold iterated* integral, which computes the increment of the ith coordinate of the path at time t as:

$$\mathrm{S}(X)_t^i = \int_{a<t_1<t} \mathrm{d}X_{t_1}^i = X_t^i - X_a^i, \tag{4}$$

and we again emphasise that $\mathrm{S}(X)_t^i$ is also a real valued path. Geometrically, the 1-fold iterated integral is the increment of this one-dimensional path over the entire time interval.

Since the 1-fold iterated integral is a path, we can apply the same iterated integral and define the *2-fold iterated* integral (Chen, 1954; 1977) as:

$$\mathrm{S}(X)_t^{i,j} = \int_{a<t_2<t} \mathrm{S}(X)_{t_2}^i \mathrm{d}X_{t_2}^j = \int_{a<t_1<t_2<t} \mathrm{d}X_{t_1}^i \mathrm{d}X_{t_2}^j. \tag{5}$$

To draw a geometrical parallel once again, for a single-dimensional path, the signature would be equivalent to the square of the increment over the time interval, that is:

$$\mathrm{S}(X)_t^{i,i} = \frac{1}{2}\left(X_t^i - X_a^i\right)^2. \tag{6}$$

If, instead, we consider $c = 2$ for a two-dimensional path, the 1-fold iterated integral will have two components, namely:

$$\mathrm{S}(X)_t^i = X_t^i - X_a^i \tag{7}$$

$$\mathrm{S}(X)_t^j = X_t^j - X_a^j, \tag{8}$$

where each component accounts for the increment of the path on the respective axis. However, the 2-fold iterated integral would now have $c^2 = 4$ components,

$$\mathrm{S}(X)_t^{i,i} = \frac{1}{2!}\left(X_t^i - X_a^i\right)^2 \tag{9}$$

$$\mathrm{S}(X)_t^{j,j} = \frac{1}{2!}\left(X_t^j - X_a^j\right)^2 \tag{10}$$

$$\mathrm{S}(X)_t^{i,j} = \int_{a<t_2<t} \mathrm{S}(X)_{t_2}^i \mathrm{d}X_{t_2}^j = \int_{a<t_1<t_2<t} \mathrm{d}X_{t_1}^i \mathrm{d}X_{t_2}^j \tag{11}$$

$$\mathrm{S}(X)_t^{j,i} = \int_{a<t_2<t} \mathrm{S}(X)_{t_2}^j \mathrm{d}X_{t_2}^i = \int_{a<t_1<t_2<t} \mathrm{d}X_{t_1}^j \mathrm{d}X_{t_2}^i, \tag{12}$$

where the first two components are the same as equation (6). For the remaining components, their geometric intuition is given by resulting Lévy area, shown in Figure 3. The Lévy area is the signed area enclosed by the path and the chord connecting the end-points. The sign of the area depends on the sign of the winding number of the path moving around it (Chevyrev and Kormilitzin, 2016; Yang et al., 2017).

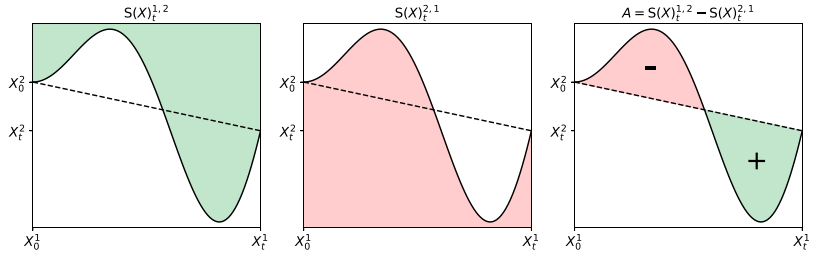

Figure 3. Geometric interpretation of the path signature. The path is moving from time 0 to time t. The dashed line is the chord connecting the two end-points of the path. *Left and centre*: These images depict two components of the 2-fold iterated integral of this path. *Right*: The image shows the Lévy area enclosed by the path and its chord.

The geometric interpretation of the *k-fold iterated* integral for $k > 2$ of a two-dimensional path is not trivial and therefore not presented here. Nonetheless, by analogy, for a three-dimensional path, the 1-fold, 2-fold, and 3-fold iterated integrals are units of displacement, area, and volume, respectively. Informally, we can proceed indefinitely, and we retrieve the path signature by collecting all iterated integrals of the path X.

Definition 1. Signature (Chevyrev and Kormilitzin, 2016). *The signature of a path* $X : t \in [a, b] \to \mathbb{R}^c$, *denoted by* $S(X)_t$, *is the infinite series of all iterated integrals of X. Formally,* $S(X)_t$ *is the sequence of real numbers*

$$S(X)_t = \left(1, S(X)_t^1, \ldots, S(X)_t^c, S(X)_t^{1,1}, S(X)_t^{1,2}, \ldots\right), \tag{13}$$

where the iterated integrals are defined as:

$$S(X)_t^{i_1,\ldots,i_k} = \int_{a<t_k<t} \cdots \int_{a<t_1<t_2} dX_{t_1}^{i_1} \ldots dX_{t_k}^{i_k}, \tag{14}$$

and the superscripts are drawn from the set $\mathcal{M}$ *of all multi-indexes,*

$$\mathcal{M} = \{(i_1, \ldots, i_k) |\ k \geq 1, i_1, \ldots, i_k \in \{1, \ldots, c\}\}. \tag{15}$$

In practice, we often apply a truncated signature up to a degree d, that is $S^d(X)_t$, defined as the finite collection of all terms of the signature up to multi-indices of length d.

The path signature was originally introduced by Chen (Chen, 1958) who applied it to piecewise smooth paths, and further developed by Lyons and others (Améndola et al., 2019; Boedihardjo et al., 2016; Hambly and Lyons, 2010; Lyons, 2014). The number of elements in the path signature depends on the dimension of the input c and the degree d and is given by c^d. Therefore, the time and space scalability of the signature is rather poor ($O(c^d)$), but this can be alleviated with the use of kernel methods as we will discuss in Section 4. The signature of a path has several interesting properties, which make it inherently interesting for applications in robotics.

3.2.1. Canonical feature map for paths. For all effects, the path signature can be thought of as a *linear* feature map (Fermanian, 2021) that transforms multivariate sequential data into an infinite length series of real numbers, which uniquely represents a trajectory through Euclidean space. This is valid even for paths with missing or irregularly sampled values (Boedihardjo et al., 2016; Hambly and Lyons, 2010).

3.2.2. Time-reversal. We informally define the time-reversed path $\overleftarrow{X}$ as the original path X moving backwards in time. It follows that the tensor product of the signatures $S(X)_{a,b} \otimes S(\overleftarrow{X})_{a,b} = 1$, which is the identity operation.

3.2.3. Uniqueness. The signature of every non-tree-like path is unique (Hambly and Lyons, 2010). A tree-like path is one in which a section exactly retraces itself. Tree-like paths are quite common in real data (e.g., in cyclic actions), and this could be a limiting factor of the signature's application. However, it has been proven (Hambly and Lyons, 2010) that if a path has at least one monotonous coordinate, then its signature is unique. The main significance of this result is that it provides a practical procedure to guarantee signature uniqueness by, for example, including a time dimension.

3.2.4. Invariance under reparametrisation. An important difficulty when vying for diversity in trajectory optimisation is the potential symmetry present in the data. This is particularly true when dealing with sequential data, such as, for instance, trajectories of an autonomous vehicle. In this case, the problem is compounded as there is an infinite group of symmetries given by the reparametrisation of a path (i.e., continuous surjections in the time domain to itself), each leading to distinct similarity metrics. In contrast, the path signature acts as a filter that is invariant to reparametrisation removing these troublesome symmetries and resulting in the same features as shown in Figure 4.

3.2.5. Dimension is independent of path length. The final property we will emphasise is how the dimension of the signature depends on its degree and the intrinsic dimension of the path but is independent of the path length. In other words, the signature dimension is invariant to the degree of discretisation of the path.

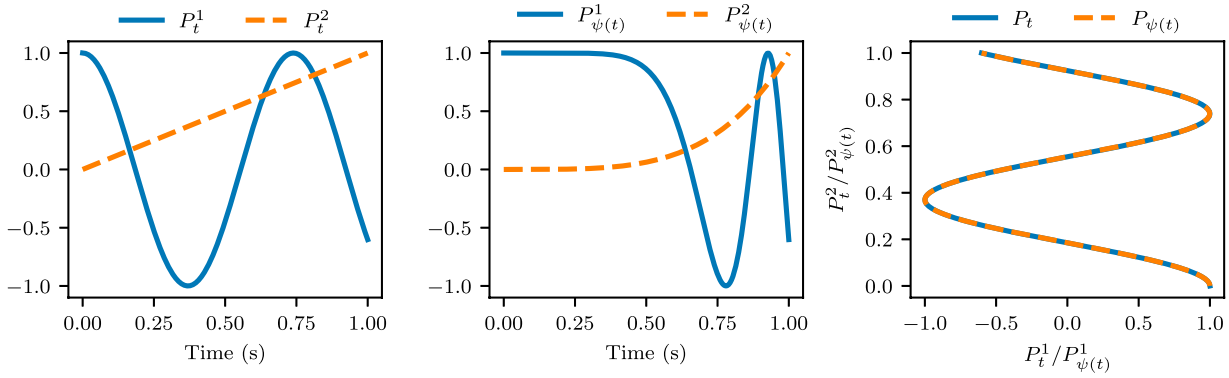

Figure 4. Signature invariance to reparametrisation. *Left*: Plot of the coordinates of a two-dimensional path P_t over time. Here, $P_t^1 = \cos(8.5t)$ and $P_t^2 = t$. *Centre*: Plot of the two coordinates of path P_t reparameterised by function ψ. Now, $P_{\psi(t)}^1 = \cos(8.5t^4)$ and $P_{\psi(t)}^2 = t^4$. *Right*: Plots of path P_t and its reparameterised version $P_{\psi(t)}$ are shown overlapping to illustrate how the change in time is irrelevant if the goal is achieving diverse paths. The signature of degree 2 for both paths is $\{1, -1.6, 1, 1.3, -0.9, -0.7, 0.5\}$.

3.3. Stein variational gradient descent

Variational inference (VI) (Blei et al., 2017) is an established and powerful method for approximating challenging posterior distributions in Bayesian Statistics. As opposed to Markov chain Monte Carlo (MCMC) (Haugh, 2021) approaches, in VI, the inference problem is cast as an optimisation problem in which a candidate distribution $q^*(\mathbf{x})$ within a distribution family $\mathcal{Q}$ is chosen to best approximate the target distribution $p(\mathbf{x})$. This is typically obtained by minimising the Kullback–Leibler (KL) divergence between distributions, defined as:

$$D_{\mathrm{KL}}(q(\mathbf{x})\|p(\mathbf{x})) := \int_{-\infty}^{\infty} -\log q(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} \mathrm{d}\mathbf{x}. \tag{16}$$

Such that the optimal candidate distribution is given by:

$$q^*(\mathbf{x}) = \arg\min_{q\in\mathcal{Q}} D_{\mathrm{KL}}(q(\mathbf{x})\|p(\mathbf{x})). \tag{17}$$

The solution also maximises the Evidence Lower Bound (ELBO), as expressed by the following objective:

$$q^* = \arg\max_{q\in\mathcal{Q}} \mathbb{E}_q[\log p(\mathbf{x})] - D_{\mathrm{KL}}(q(\mathbf{x})\|p(\mathbf{x})). \tag{18}$$

The main challenge that arises is defining an appropriate $\mathcal{Q}$. Stein variational gradient descent (SVGD) (Liu and Wang, 2016) addresses this issue while also solving for equation (16) by performing Bayesian inference in a non-parametric nature, removing the need for assumptions on restricted parametric families for $q(\mathbf{x})$. This approach approximates a posterior $p(\mathbf{x})$ with a set of particles $\{\mathbf{x}^i\}_{i=1}^{N_p}$, $\mathbf{x} \in \mathbb{R}^p$. These particles are iteratively updated in parallel according to:

$$\mathbf{x}^i \leftarrow \mathbf{x}^i + \epsilon \phi^*(\mathbf{x}^i), \tag{19}$$

given a step size ϵ. The function $\phi(\cdot)$ is known as the score function and defines the velocity field that maximally decreases the KL-divergence:

$$\phi^* = \arg\max_{\phi\in\mathcal{H}} \{-\nabla_\epsilon D_{\mathrm{KL}}(q_{[\epsilon\phi]}\|p), \text{s.t. } \|\phi\|_{\mathcal{H}} \le 1\}, \tag{20}$$

where $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) induced by a positive-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and $q[\epsilon\phi]$ indicates the particle distribution resulting from taking an update step as in equation (19). Recall that an RKHS $\mathcal{H}$ associated with a kernel k is a Hilbert space of functions endowed with an inner product $\langle \cdot, \cdot \rangle$ such that $f(\mathbf{x}) = \langle f, k(\cdot, \mathbf{x}) \rangle$ for any $f \in \mathcal{H}$ and any $\mathbf{x} \in \mathcal{X}$ (Schölkopf and Smola, 2002). In Liu and Wang (2016), the problem in (20) has been shown to yield a closed-form solution which can be interpreted as a functional gradient in $\mathcal{H}$ and approximated with the set of particles:

$$\phi^*(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim\widehat{q}}[k(\mathbf{y}, \mathbf{x})\nabla_{\mathbf{y}} \log p(\mathbf{y}) + \nabla_{\mathbf{y}} k(\mathbf{y}, \mathbf{x})], \tag{21}$$

with $\widehat{q} = \frac{1}{N_p}\sum_{i=1}^{N_p} \delta(\mathbf{x}^i)$ being an empirical distribution that approximates q with a set of Dirac delta functions $\delta(\mathbf{x}^i)$. For SVGD, k is typically a translation-invariant kernel, such as the squared-exponential or the Matérn kernels (Liu and Wang, 2016; Rasmussen and Williams, 2006).

4. Method

Our main goal is to find a diverse set of solutions to the problem presented in Section 3.1. To that end, we begin by reformulating equation (1) as a probabilistic inference problem. Next, we show that we can apply SVGD to approximate the posterior distribution of trajectories with a set of sampled paths. Finally, in Section 4.3, we present our main contribution discussing how we can promote diversity among the sample paths by leveraging the Path Signature Kernel.

4.1. Stein variational motion planning

To reframe the trajectory optimisation problem described in equation (1) as probabilistic inference, we introduce a

binary optimality criterion, $\mathcal{O}: \mathcal{P}_{\mathcal{X}} \rightarrow \{0,1\}$, analogously to (Barcelos et al., 2021; Levine, 2018). Simplifying the notation with $\mathcal{O}$ indicating $\mathcal{O}=1$, we can represent the posterior distribution of optimal trajectories as $p(X|\,\mathcal{O}) \propto p(\mathcal{O}|\,X)p(X)$, for a given optimality likelihood $p(\mathcal{O}|\,X)$ and trajectory prior $p(X)$. The *maximum a posteriori* (MAP) solution is given by finding the mode of the negative log posterior:

$$\begin{aligned} X^* &= \arg\min_X -\log p(\mathcal{O}|\,X) - \log p(X) \\ &= \arg\min_X \lambda\mathcal{C}(X) - \log p(X), \end{aligned} \tag{22}$$

where the last equality arises from the typical choice of the exponential distribution to represent the optimality likelihood (Theodorou, 2015; Williams et al., 2018), that is, $p(\mathcal{O}|\,X) = \exp(-\lambda\mathcal{C}(X))$ with λ being a temperature hyperparameter that controls the amount of exploration. In practice, the temperature is chosen empirically according to the desired variability of the sampled trajectories, since a lower temperature will penalise trajectories with higher costs less.

Rather than finding the MAP solution, we are interested in approximating the full posterior distribution, which may be multi-modal, and generating diverse solutions for the planning problem. As discussed in Section 3.3, we can apply SVGD to approximate the posterior distribution with a collection of particles. In the case at hand, each of such particles is a sampled path, such that equation (21) can be rewritten as:

$$\phi^*(X) = \mathbb{E}_{Y\sim q}[k(Y,X)\nabla_Y \log p(Y|\,\mathcal{O}) + \nabla_Y k(Y,X)]. \tag{23}$$

The score function presented in equation (23) is composed of two competing forces. On one hand, we have the kernel-smoothed gradient of the log-posterior pushing particles towards regions of higher probability, whereas the second term acts as a repulsive force, pushing particles away from one another.

It is worth emphasising that the kernel function is *static*, that is, it does not consider the sequential nature of the input paths. In effect, for a path of dimension c and s discrete time-steps, the inputs are projected onto a space $\mathcal{V} \subset \mathbb{R}^{c\times s}$ in which similarities are evaluated.

Finally, the posterior gradient can be computed by applying Bayes' rule, resulting in:

$$\nabla_Y \log p(Y|\,\mathcal{O}) = \nabla_X \log p(Y) - \nabla_Y \lambda\mathcal{C}(Y). \tag{24}$$

4.2. Stein variational motion planning with smooth paths

In previous work (Barfoot et al., 2014; Dong et al., 2016; Lambert and Boots, 2021; Mukadam et al., 2018), the prior distribution in equation (24) is defined in a way to promote smoothness on generated paths. This typically revolves around defining Gaussian Processes (Rasmussen and Williams, 2006) as priors and leveraging factor graphs for efficiency. Although effective, this approach still requires several latent variables to describe a desired trajectory, which implies on a higher dimensional inference problem.

Importantly, the problem dimensionality is directly related to the amount of repulsive force exerted by the kernel. In large dimensional problems, the repulsive force of translation-invariant kernels vanishes, allowing particles to concentrate around the posterior modes, which results in an underestimation of the posterior variance (Zhuo et al., 2018). This problem is further accentuated when considering the static nature of the kernel function, as discussed in the previous section.

In order to keep the inference problem low-dimensional while still enforcing smooth paths, we make use of *natural cubic splines* and aim to optimise the location of a small number of knots. These knots may be initialised in different ways, such as perturbations around a linear interpolation from the starting state $\mathbf{x}_s$ and goal state $\mathbf{x}_g$, sampled from an initial solution given by a shooting method (e.g., RRT (LaValle and Kuffner, 2001)), or drawn randomly from within the limits of $\mathcal{X}$. For simplicity, in this work we will opt for the latter.

Since path smoothness is induced by the splines, the choice of prior is more functionally related to the problem at hand. If one desires some degree of regularisation on the trajectory optimisation, a multivariate Gaussian prior centred at the placement of the initial knots may be used. Conversely, if we only wish to ensure the knots are within certain bounds, a less informative smoothed approximation of the uniform prior may be used. More concretely, for a box $B = x\colon a \le x \le b$, such prior would be defined as:

$$p(x) \propto \exp\left(-d(x,B)^2 \Big/ \sqrt{(2\sigma^2)}\right) \tag{25}$$

where the distance function $d(x,B)$ is given by $d(x,B) = \min|x - x'|, x' \in B$. Finally, we could define both a prior and hyper-prior if we wish to combine both effects (see Appendix C for details).

As discussed in Section 3.1, the cost functional $\mathcal{C}$ imposes penalties for collisions and defines the relevant performance criteria to be observed. Since only a small number of knots is used for each path, some of these criteria and, in particular, collision checking require that we discretise the resulting spline in a sufficiently dense amount of points. It is worth mentioning that $\mathcal{C}$ is typically non-differentiable and that the gradient in equation (24) is usually approximated with Monte Carlo samples (Barcelos et al., 2021). However, as this introduces an extra degree of stochasticity in the benchmark comparison, we will restrict our choice of $\mathcal{C}$ to be differentiable. We will discuss the performance criteria of each problem in the experimental section.

Algorithm 1: Kernel Signature Stein Variational Gradient Descent (SigSVGD)

Input: A cost function $\mathcal{C}(X)$ or target distribution $p(X)$, a prior distribution $q(X_{t_0})$, a signature kernel $k^{\oplus}$.
Output: A set of particles $\{X_t^i\}_{i=1}^{N_p}$ that approximates the posterior distribution over optimal paths.

Sample $\{X_{t_0}^i\}_{i=1}^{N_p} \sim q(X_{t_0})$;
while *task not complete* **do**
  if *using Monte Carlo samples* **then**
    Generate N_s samples for each path $X_t^{i,j} \leftarrow X_t^i + \eta_j$;
  if *using splines* **then**
    Generate decimated trajectories from knots X_t;
  Evaluate $\mathcal{C}(X_t)$ in parallel;
  if *target distribution* $p(X_t)$ *is available* **then**
    Update score $\phi^* \leftarrow \frac{1}{N_p}\sum_i^{N_p}[k^{\oplus}(X_t^i, X_t)\nabla_{X_t^i}\log p(X_t^i) + \nabla_{X_t^i}k^{\oplus}(X_t^i, X_t)]$;
  else
    Log-posterior gradient $\nabla_{X_t^i}\log p(X_t^i \mid \mathcal{O}) \approx \nabla_{X_t^i}\log q(X_{t-1}^i \mid \mathcal{O}) + \nabla_{X_t^i}\log\frac{1}{N_s}\sum_j^{N_s}\exp(-\alpha\mathcal{C}(X_t^{i,j}))$;
    Update score $\phi^* \leftarrow \frac{1}{N_p}\sum_i^{N_p}[k^{\oplus}(X_t^i, X_t)\nabla_{X_t^i}\log p(X_t^i \mid \mathcal{O}) + \nabla_{X_t^i}k^{\oplus}(X_t^i, X_t)]$;
  Update paths $X_t \leftarrow X_t + \epsilon\phi^*$;
  Update prior $q(X_t \mid \mathcal{O}) \leftarrow p(X_t \mid \mathcal{O})$; - For details, see (Barcelos, Lambert, Oliveira, Borges, Boots and Ramos 2021)
  $t \leftarrow t+1$;

4.3. Stein variational motion planning with path signature Kernel

In this section, we present our main contribution, which is a new formulation for motion planning in which Path Signature can be used to efficiently promote diversity in trajectory optimisation through the use of Signature Kernels. In Section 3, we discussed some desirable properties of the signature transform. The key insight is that the space of linear combination of signatures forms an algebra and makes the signature a faithful feature map for trajectories (Kiraly and Oberhauser, 2019).

With that in mind, perhaps the most straightforward use of the signature would be to redefine the kernel used in equations (20) and (21) as $\overline{k}(X,Y) = k(\mathrm{S}(X)_t, \mathrm{S}(Y)_t)$. However, as seen in Section 3, this approach would not be scalable given the exponential time and space complexity of the signature w.r.t. its degree. A single evaluation of the Gram kernel matrix for $\overline{k}$ would be an operation of order $O(n^2 \cdot c^d)$, where n is the number of concurrent paths being optimised, d is the degree of the signature, and c is the dimensionality of the space $\mathcal{P}_{\mathcal{X}} \ni X, Y$. Furthermore, kernel $\overline{k}$ is static in the sense that it does not take into account the sequential nature of its domain. Rather than a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, we want to define a kernel $k^+ : \mathcal{P}_{\mathcal{X}} \times \mathcal{P}_{\mathcal{X}} \to \mathbb{R}$, which takes into account the structure induced by paths.

Hence, we take a different approach and proceed by first projecting paths to an RKHS onto which we will then compute the signature. That is, given a kernel $k^+ : \mathcal{P}_{\mathcal{X}} \times \mathcal{P}_{\mathcal{X}} \to \mathbb{R}$, a path $X \in \mathcal{P}_{\mathcal{X}}$ can be lifted to a path in the RKHS $\mathcal{P}_{\mathcal{H}}$ through the map $k_X : t \mapsto k(X_t, \cdot)$, where $\mathcal{P}_{\mathcal{H}}$ is the set of $\mathcal{H}$-valued paths. Finally, we compute the signature of the lifted path $\mathrm{S}(k_X)_t$ and use it as our final feature map.

At first glance, this operation would further deteriorate scalability since the most useful $\mathcal{P}_{\mathcal{H}}$ are infinite dimensional, rendering the approach infeasible. However, results presented by Kiraly and Oberhauser (2019, Corollary 4.9) show that this approach can be completely kernelised. This allows them to define a *truncated signature kernel*, $k^+ : (X_t, Y_t) \mapsto \langle \mathrm{S}^d(k_X)_t, \mathrm{S}^d(k_Y)_t \rangle$, that can be efficiently computed using only evaluations of a static kernel $k(\mathbf{x}, \mathbf{y})$ at discretised timestamps. The number of evaluations depends on the truncation degree d and number of discretised steps l. Several algorithmic approaches are considered in Kiraly and Oberhauser (2019) with dynamic programming having complexity $O(n^2 \cdot l^2 \cdot d)$ to compute a $(n \times n)$-Gram matrix. Otherwise, approximations can be used to reduce the complexity to linear on l and n. However, even though the importance of the terms in the signature decay factorially (Lyons, 2014), the amount of coefficients grows exponentially, which means that for high values of d, the kernel k^+ would be restricted to low-dimensional applications.

Nonetheless, recent work (Salvi et al., 2021) proved that for two continuously differentiable input paths, the complete *signature kernel*,

$$k^{\oplus} : (X_t, Y_t) \mapsto \langle \mathrm{S}(k_X)_t, \mathrm{S}(k_Y)_t \rangle, \tag{26}$$

is the solution of a second-order, hyperbolic partial differential equation (PDE) known as Goursat PDE. Solving this PDE is a problem of complexity $O(l^2 \cdot c)$, so still restrictive on the discretisation of the path. However, by its intrinsic nature, the PDE can be parallelised, turning the complexity into $O(l \cdot c)$, as long as the GPU is able to accommodate the required number of threads. Therefore, the untruncated signature kernel can be efficiently and parallel computed using state-of-the-art hyperbolic PDE solvers and finite-difference evaluations of the static kernel k.

Hence, we can directly apply $k^{\oplus}$ in equation (23), and we now have a way to properly represent sequential data in feature space, resulting in the final gradient update function:

$$\phi^*(X) = \mathbb{E}[k^{\oplus}(Y, X)\nabla_Y \log p(\mathrm{Y} \mid \mathcal{O}) + \nabla_Y k^{\oplus}(Y, X)], \tag{27}$$

where the expectation is taken by sampling paths Y from $\widehat{q}$. For convenience, we will use the acronym SigSVGD, whether the algorithm is used for planning or control problems. A complete overview of the algorithm is presented in Algorithm 1.

5. Results

In this section, we present results to demonstrate the correctness and applicability of our method in a set of simulated experiments, ranging from simple 2D motion planning to a challenging benchmark for robotic manipulators. For another introductory example, a path following task is presented in Appendix B.

5.1. Motion planning on 2D terrain

Our first set of experiments consists of trajectory optimisation in a randomised 2D terrain illustrated in Figure 2. Regions of higher cost, or hills, are shown in a darker shade, whereas valleys are in a lighter colour. The terrain is

parameterised by a series of isotropic Multivariate Gaussian distributions placed randomly according to a Halton sequence and aggregated into a Gaussian Mixture Model denoted by p_{map}.

In this instance, paths are parameterised by natural cubic splines with $N_k = 2$ intermediary knots, apart from the start and goal state. Note that this is a design choice and the method is agnostic to the type of parameterisation chosen to encode the paths, like, for instance, GP or Random Fourier Features (RFF). Characteristics such as smoothness, computational cost, and differentiability should be taken into account when selecting a given parameterisation.

Our goal is to find the best placement for these knots to find paths from origin to goal that avoid regions of high cost but are not too long. We adopt the following cost function in order to balance trajectory length and navigability:

$$\mathcal{C}(\mathbf{x}_t) = \sum_{t\in[a,b]} \left(p_{\text{map}}(\mathbf{x}_t) + 75\,\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2\right), \tag{28}$$

where the ℓ^2-norm term is a piecewise linear approximation of the trajectory length. To ensure the approximation is valid, each trajectory is decimated into 100 waypoints before being evaluated by equation (28).

The initial knots are randomly placed, and the plots in Figure 2 show the final 20 trajectories found with three different optimisation methods. Furthermore, the colour of each path depicts its normalised final cost. On the left, we can see the solutions found with Batch Gradient Descent (BGD) and note how all trajectories converge to two modes of similar cost. The SVMP results are more diverse but failed to capture one of the BGD modes. Also note how, when multiple trajectories converge to a single trough, the spline knots are pushed away by the repulsive force resulting in suboptimal solutions. On the other hand, the trajectories found by SigSVGD are not only more diverse, finding more homotopic solutions, but are also able to coexist in the narrow valleys. This is possible since the repulsive force is being computed in the signature space and not based on the placement of the knots. Furthermore, notice how, for the same reason, the paths are more direct and coordinated when compared to SVMP.

5.2. Point-mass navigation on an obstacle grid

Here, our goal is to demonstrate the benefits of applying the signature kernel Model Predictive Control (MPC). To that end, we reproduce the point-mass planar navigation task presented in Barcelos et al. (2021) and (Lambert et al. (2020) and compare SVMPC against a modified implementation using SigSVGD. The objective is to navigate a holonomic point-mass robot from start to goal through an obstacle grid. Since the system dynamics is represented as a double integrator model with non-unitary mass m, the particle acceleration is given by $\ddot{\mathbf{x}} = m^{-1}\mathbf{u}$ and the control signal is the force applied to the point mass. We adopt the same cost function as in Barcelos et al. (2021), that is:

$$\mathcal{C}(\mathbf{x}_t, \mathbf{u}_t) = 0.5\ \ \mathbf{e}_t^{\mathrm{T}}\mathbf{e}_t + 0.25\ \ \dot{\mathbf{x}}_t^{\mathrm{T}}\dot{\mathbf{x}}_t + 0.2\ \ \mathbf{u}_t^{\mathrm{T}}\mathbf{u}_t + 1\{\text{col.}\}p$$
$$\mathcal{C}_{\text{term}}(\mathbf{x}_t, \mathbf{u}_t) = 1000\ \ \mathbf{e}_t^{\mathrm{T}}\mathbf{e}_t + 0.1\,\dot{\mathbf{x}}_t^{\mathrm{T}}\dot{\mathbf{x}}_t,$$

where $\mathbf{e}_t = \mathbf{x}_t - \mathbf{x}_g$ is the instantaneous position error and $p = 10^6$ is the penalty when a collision happens.

To create a controlled environment with several multimodal solutions, obstacles are placed equidistantly in a grid (see Figure 5). The simulator performs a simple collision check based on the particle's state and prevents any future movement in case a collision is detected, simulating a crash. Barriers are also placed at the environment boundaries to prevent the robot from easily circumventing the obstacle grid. As the indicator function makes the cost function non-differentiable, we need to compute approximate gradients using Monte Carlo sampling (Lambert et al., 2020). Furthermore, since we are using a stochastic controller, we also include CMA-ES and Model Predictive Path Integral (MPPI) (Williams et al., 2016) in the benchmark. A detailed account of the hyper-parameters used in the experiment is presented in Appendix A.

In this experiment, each of the particles in the optimisation is a path that represents the mean of a stochastic control policy. Gradients for the policy updates are generated by sampling the control policies and evaluating *rollouts* via an implicit model of the environment. As CMA-ES only entertains a single solution at any given time, to make the results comparable, we increase the amount of samples it evaluates at each step to be equivalent to the number of policies times the number of samples in SVMPC. One addition to the algorithm in Lambert et al. (2020) is the inclusion of particles with predefined primitive control policies which are not optimised. For example, policies which constantly apply the minimum, maximum, or no acceleration are all valid primitives. These primitive policies are also included in every candidate solution set of CMA-ES.

The inlay plot in Figure 5 illustrates how SigSVGD promotes policies that are more diverse, covering more of

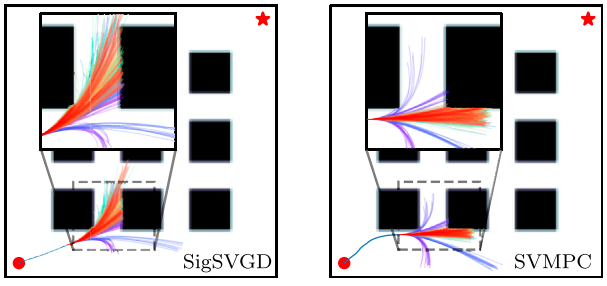

Figure 5. Point-mass navigation trajectories. The plot shows an intermediate time-step of the navigation task for SigSVGD, on the left, and SVMPC, on the right. An inset plot enlarges a patch of the map just ahead of the point mass. The rollout colour indicates from which of the policies, that is, paths in the optimisation, they originate, whereas fixed motion primitives are shown in purple. Note how rollouts generated by SigSVGD are more dispersed, providing a better gradient for policy updates.

the state space on forward rollouts. The outcome can be seen on Table 1. SigSVGD finds lower cost policies and is able to reach the goal in fewer steps than SVMPC. Due to the dynamical nature of the problem, we are unable to run the optimisation for many iterations during each time-step, as we need to get actions from the controller at a fast rate. This poses a challenge to CMA-ES, which crashed on all episodes despite having a much larger number of samples per step.

5.3. Benchmark comparison on robotic manipulator

To test our approach on a more complex planning problem, we compare batch gradient descent (i.e., parallel gradient descent on different initialisations), SVMP and SigSVGD in robotic manipulation problems generated using MotionBenchMaker (Chamzas et al., 2022). A problem consists of a scene with randomly placed obstacles and a consistent request to move the manipulator from its starting pose to a target configuration. For each scene in the benchmark, we generate four different requests and run the optimisation with five random seeds for a total of 20 episodes per scene. The numerical experiments are conducted on a system with Intel i7-10750H at 5.0 GHz with 12 cores CPU, 32 GB RAM (system memory) and an NVIDIA GeForce GTX 1650 Ti Mobile GPU.

The robot used is a Franka Emika Panda with 7 Degrees of Freedom (DOF). The cost function is designed to generate trajectories that are smooth, collision-free and with a short displacement of the robot's end-effector. We once again resort to a fully-differentiable function to reduce the extraneous influence of approximating gradients with Monte Carlo samples. As is typical in motion planning, the optimisation is performed directly in *configuration space* (C-space), which simplifies the search for feasible plans. To reduce the sampling space and promote smooth trajectories, we once again parameterise the path of each of the robot joints with natural cubic splines, adopting three intermediary knots besides those at the initial and target poses.

Table 1. Point-mass navigation results.

	Cost	Steps
SigSVGD	**1056.0 (58.4)**	**189.3 (12.6)**
SVMPC	1396.4 (73.0)	239.1 (49.4)
MPPI	1740.7 (192.3)	290.8 (23.7)
CMA-ES†	—	—

The table shows the mean and standard deviation for 20 episodes. Best results are presented in bold. *Cost* indicates the total accrued cost over the episode. CMA-ES cost is not shown as it could not complete the task on any episodes. *Steps* indicate the total number of time-steps the controller needs to reach the goal.
†CMA-ES could not complete any episodes, so results are omitted.

5.3.1. Regularising path length and dynamical motions. Finally, the use of splines to interpolate the trajectories ensures smoothness in generated trajectories, but that does not necessarily imply in smooth dynamics for the manipulator. To visualise this, consider, for example, a trajectory in $\mathcal{Q}$ parameterised by a natural cubic spline. The configurations $\mathbf{q}$ in between each knot can be interpolated, resulting in a smooth trajectory of the robot end-effector in Euclidean coordinates in SE(3). However, the same end-effector trajectory could be traversed in a constant linear speed or with a jerky acceleration and deceleration motion. More specifically, if we use a fixed number of interpolated configurations between knots without care to impose dynamical restrictions to the simulator, knots that are further apart will result in motions with greater speed and acceleration since a larger distance would be covered during the same interval. To avoid these abrupt motions on the robot joints, we introduce the term $\mathcal{C}_{\text{dyn}}$ to the cost function, which penalises the linear distance between consecutive configurations:

$$\mathcal{C}_{\text{dyn}} = \sum_{i=2}^{p} \mathbf{w}^{\mathrm{T}} \|\mathbf{q}_i - \mathbf{q}_{i-1}\|_2, \tag{29}$$

where p is the number of intermediary configurations chosen when discretising the path spline, and the weight $\mathbf{w}$ can be used to assign a higher importance to certain robot joints. We choose to adopt a vector $\mathbf{w}$, which is a linear interpolation from 1 to 0.7, where the higher value is assigned to the base joint of the manipulator and progressively reduced until the end-effector. A similar approach as the one presented in equation (29) can be used to penalise the length of the robot's trajectory in workspace.

Let $\gamma_{\text{ee}}(t) \in \mathbb{R}^3$ denote the parameterised continuous trajectory of the end-effector position of the robot, where $\gamma_{\text{ee}}(1)$ and $\gamma_{\text{ee}}(p)$ correspond to the end-effort positions of the start and end of the trajectory, respectively. We include a term to our cost function, $\mathcal{C}_{\text{len}}$, given by:

$$\mathcal{C}_{\text{len}} = \sum_{i=2}^{p} \|\gamma_{\text{ee}}(i) - \gamma_{\text{ee}}(i-1)\|_2, \tag{30}$$

that penalises exclusively the length of the end-effector path. In addition, we include a final term to improve the smoothness of the end-effector motion by penalising extreme curvature. The curvature cost can be calculated by:

$$\mathcal{C}_{\text{cur}} = \sum_{i=1}^{p} \frac{\left\|\gamma'_{\text{ee}}(i) \times \gamma''_{\text{ee}}(i)\right\|}{\left\|\gamma'_{\text{ee}}(i)\right\|^3}, \tag{31}$$

where γ'_{ee} and γ''_{ee} denote the first- and second-order derivatives of the continuous function, and $\times$ denotes the vector cross product. This brings us to our final cost function:

$$\mathcal{C} = 2.5\,\mathcal{C}_{\text{len}} + \mathcal{C}_{\text{cur}} + 2.5\,\mathcal{C}_{\text{dyn}} + \mathcal{C}_{\text{col}} + 10\,\mathcal{C}_{\text{s-col}}, \tag{32}$$

where each of the terms is, respectively, the cost for path length, path curvature, path dynamics, collision with the environment and self-collision. The optimisation is carried out for 500 iterations, and the kernel repulsive force is scheduled with cosine annealing (Loshchilov and Hutter, 2017). By reducing the repulsive force on the last portion of the optimisation, we allow trajectories at the same local minima to converge to the modes and are able to qualitatively measure the diversity of each approach.

The results shown on Figure 6 demonstrate how SigSVGD achieves better results in almost all metrics for every scenario. The proper representation of paths results in better exploration of the configuration space and leads to better global properties of the solutions found. This can be seen in Figure 7, which shows the end-effector paths for SigSVGD and SVMP. One of such paths is also illustrated in Figure 1. For completeness, we have also included results for trajectory feasibility and respective contact depth on collisions. Not only did SigSVGD not compromise on feasibility, but also showed a higher percentage of feasible trajectories and lower contact depths for rollouts in collision (see Table 2).

5.3.2. Robot collision as continuous cost. Typically, collision-checking is a binary check and non-differentiable. To generate differentiable collision checking with informative gradients, we resort to continuous occupancy grids. Occupancy grid maps are often generated from noisy and

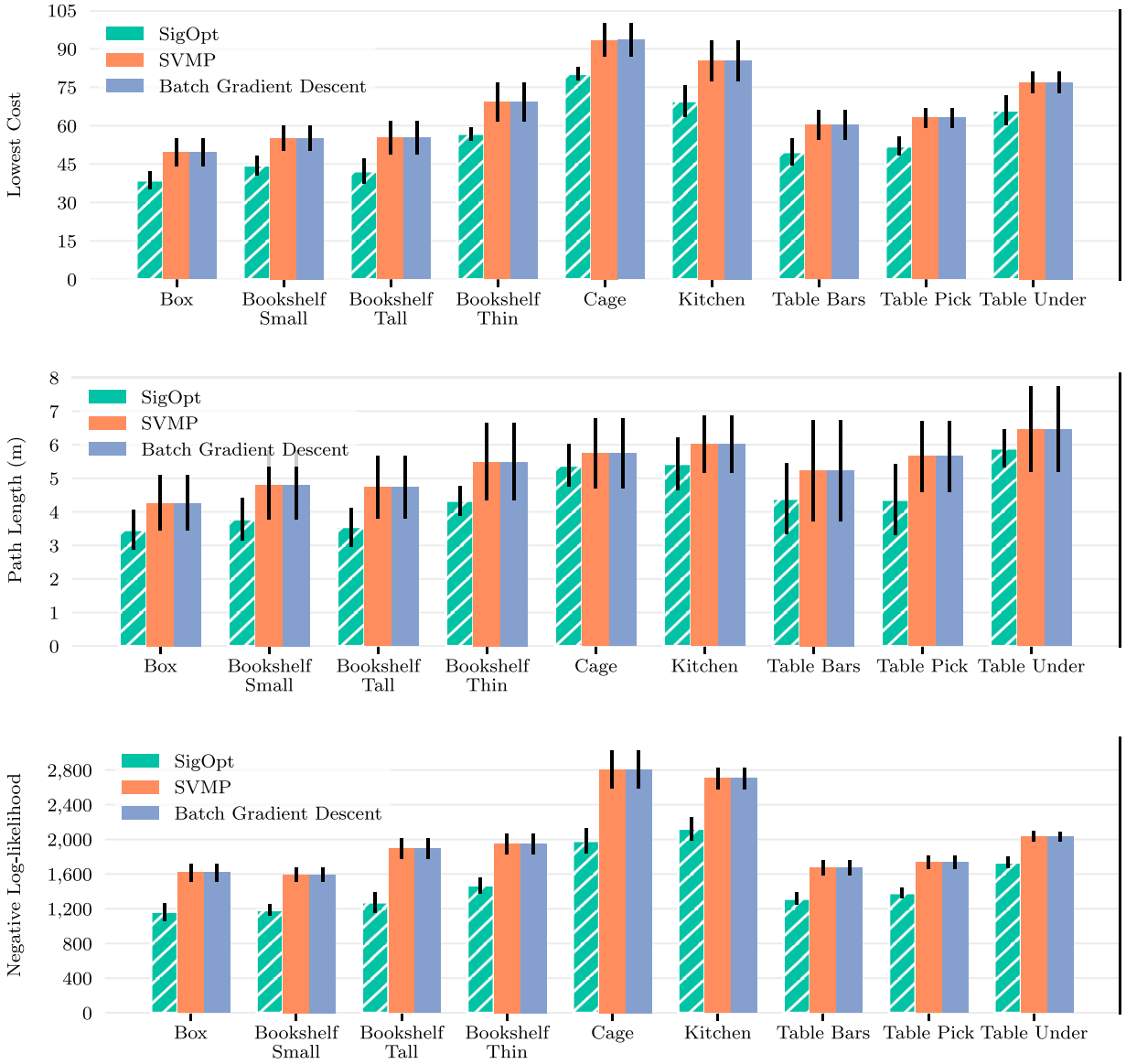

Figure 6. Motion planning benchmark. Results shown are the mean and standard deviation over five episodes for four distinct requests, totalling 20 iterations per scene. The best result is highlighted with a hatched bar. *Lowest cost* depicts the cost of the best trajectory found. *Path length* is the piecewise linear approximation of the end-effector trajectory length for the best trajectory. *NLL* indicates the negative log likelihood and, since we are using an exponential likelihood, represents the total cost of all sampled trajectories.

uncertain sensor measurement by discretising the space $\mathcal{W}$, where the robot operates (known as *workspace*) into grid cells, where each cell represents an evenly spaced field of binary random variables that corresponds to the presence of an obstacle at the given location. However, the discontinuity in-between each cell means these grid maps are non-differentiable and not suitable for optimisation-based planning. A continuous analogue of an occupancy map can be generalised by a kernelised projection to high-dimensional spaces (Ramos and Ott, 2016) or with distance-based methods (Jones et al., 2006).

In this work, we trade off the extra complexity of the methods previously mentioned for a coarser but simpler approach. Inspired by Danielczuk et al. (2021), we learn the

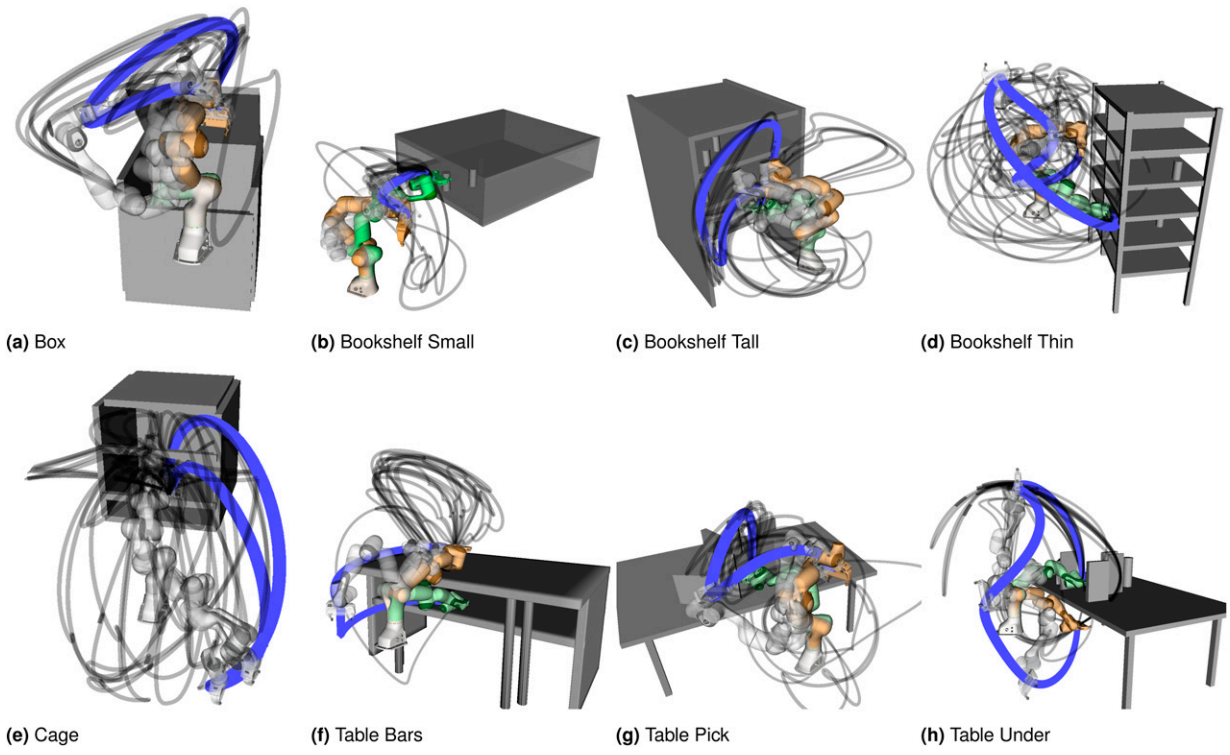

Figure 7. Visualisation of SigSVGD in the motion planning benchmark. The *Blue* and *Grey* lines denote the end-effector's trajectories with the former highlighting the trajectory with the lowest cost. The *Orange* and *Green* tinted robot poses denote the start and target configurations, respectively. The translucent robot poses denote in-between configurations of the lowest-cost solution. (a) Box. (b) Bookshelf Small. (c) Bookshelf Tall. (d) Bookshelf Thin. (e) Cage. (f) Table Bars. (g) Table Pick. (h) Table Under.

Table 2. Motion planning benchmark.

	SigSVGD		SVMP		Batch gradient descent	
Scene	Contact Depth	Feasible Pct.	Contact Depth	Feasible Pct.	Contact Depth	Feasible Pct.
Box	**3.64 (1.70)**	**94.84 (3.04)**	5.25 (3.84)	90.80 (6.72)	5.25 (3.84)	90.80 (6.73)
Bookshelf Small	**0.13 (0.31)**	**99.43 (1.04)**	0.38 (0.68)	98.71 (1.85)	0.38 (0.68)	98.72 (1.84)
Bookshelf Tall	**0.48 (0.40)**	**98.29 (1.14)**	0.68 (0.71)	97.83 (1.66)	0.68 (0.71)	97.83 (1.66)
Bookshelf Thin	**1.04 (1.30)**	**96.98 (2.42)**	1.88 (2.07)	95.25 (3.72)	1.88 (2.07)	95.25 (3.74)
Cage	**1.73 (1.58)**	**96.24 (3.02)**	3.62 (3.14)	91.84 (6.56)	3.62 (3.14)	91.84 (6.56)
Kitchen	**6.93 (4.34)**	**91.72 (4.50)**	10.16 (6.23)	88.26 (6.25)	10.16 (6.23)	88.27 (6.25)
Table Bars	**6.44 (5.81)**	**94.38 (5.24)**	7.71 (6.98)	93.60 (5.77)	7.71 (6.98)	93.59 (5.78)
Table Pick	**1.17 (1.01)**	**97.62 (1.77)**	1.52 (1.21)	96.81 (2.23)	1.52 (1.21)	96.81 (2.23)
Table Under	**2.32 (1.62)**	**95.77 (2.52)**	2.69 (1.86)	94.89 (3.13)	2.68 (1.85)	94.90 (3.10)

Results shown are the mean and standard deviation over five episodes for four distinct requests, totaling 20 iterations per scene. Best results are presented in bold. *Contact Depth* indicates the average collision depth of the trajectories found (in millimetres) if a collision happens. *Feasible Pct.* is the average percentage of the trajectory that is collision-free.

occupancy of each scene using a neural network as a universal function approximator. We train the network to approximate a continuous function that returns the likelihood of a robot configuration being occupied. The rationale for this choice is that, since all methods are optimised under the same conditions, the comparative results should not be substantially impacted by the overall quality of the map. Additionally, the trained network is fast to query and fast to obtain derivatives with respect to inputs, properties that are beneficial for querying of large batches of coordinates for motion planning.

Given a dataset of n pairs of coordinates and a binary value which indicates whether the coordinate is occupied, that is, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathcal{W} \subseteq \mathbb{R}^w$ and $y_i \in \{0, 1\}$ for $i = 1, \ldots, n$. The network then learns a mapping f_{col} between a coordinate of interest $\mathbf{x}$ and the probability of it being occupied, that is, $f_{\text{col}}(\mathbf{x}) = \mathbb{P}(y = 1 | \mathbf{x})$. A dataset of this format can be obtained, for instance, from depth sensors as point clouds. We model f_{col} as a fully-connected neural network, with tanh as the activation function between hidden layers and sigmoid as the output layer. The final network is akin to a binary classification problem, which can be learned via a binary cross-entropy loss with gradient descent optimisers. As such, we can construct a collision cost function $f_{\text{col}} : \mathcal{W} \rightarrow \mathbb{R}$ that maps workspace coordinates into cost values associated at the corresponding locations.

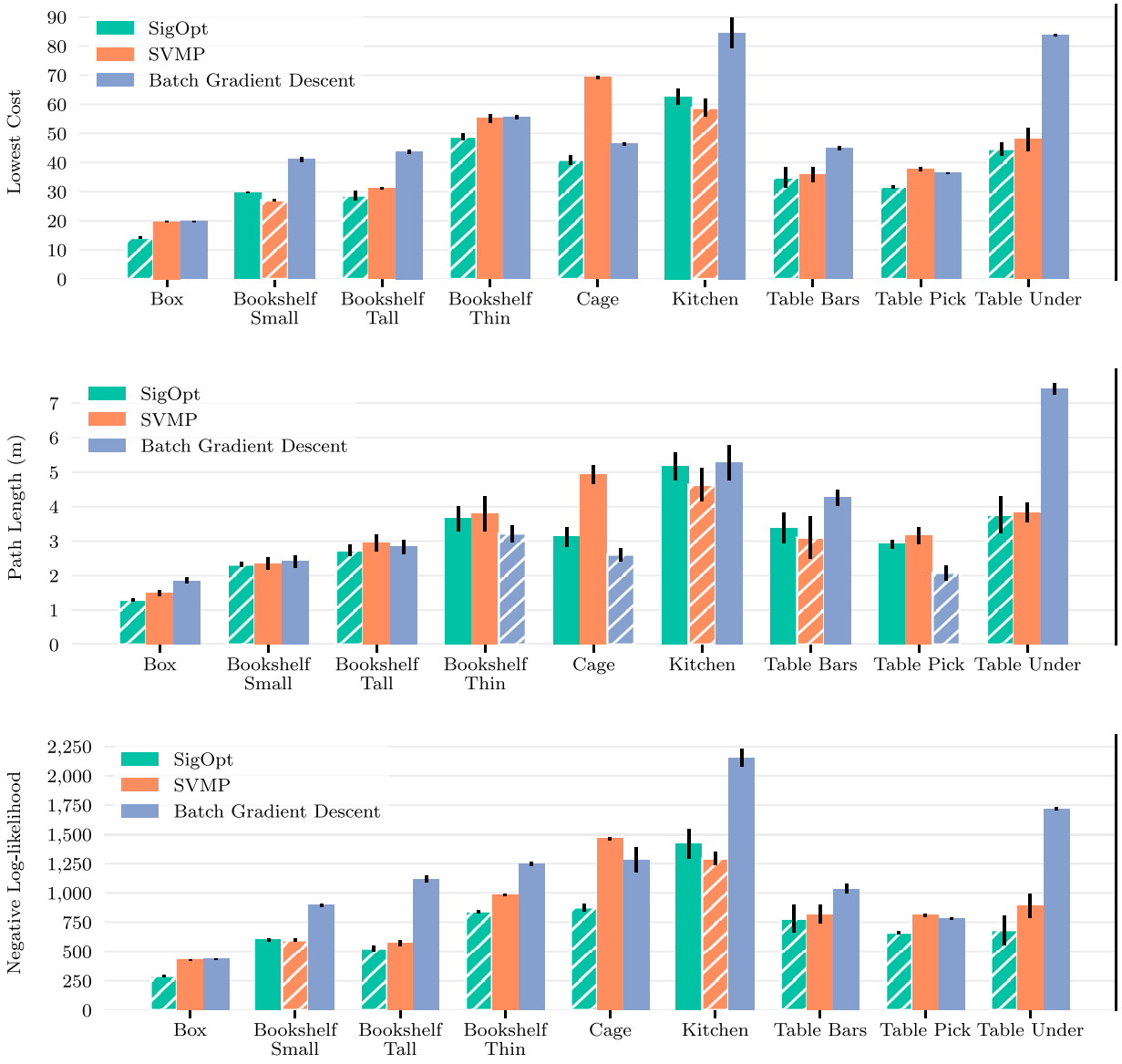

Figure 8. Motion planning benchmark with RRT*-Connect initialisation. Results shown are the mean and standard deviation over five episodes for four distinct requests, totalling 20 iterations per scene. The best result is highlighted with a hatched bar. *Lowest cost* depicts the cost of the best trajectory found. *Path length* is the piecewise linear approximation of the end-effector trajectory length for the best trajectory. *NLL* indicates the negative log likelihood and, since we are using an exponential likelihood, represents the total cost of all sampled trajectories.

A similar problem occurs when ascertaining whether a given configuration of the robot's joints is unfeasible, leading to a self-collision. We address this issue in a similar manner by training a separate neural network to approximate a continuous function $f_{\text{s-col}}$, which maps configurations of the robot to the likelihood of them being in self-collision. More precisely, $f_{\text{s-col}} : \mathcal{Q} \to \mathbb{R}$, where $f_{\text{s-col}}(\mathbf{q}) = \mathbb{P}(y = 1 | \mathbf{q})$ for $\mathbf{q}_i \in \mathcal{Q} \subseteq \mathbb{R}^d$ and $y_i \in \{0, 1\}$. The dataset used to train $f_{\text{s-col}}$ is generated by randomly choosing configurations within the joint limits of the robot and performing a binary self-collision check provided by the robot's API.

5.3.3. Bringing collision costs from workspace to configuration space. Collision checking requires information about the workspace geometry of the robot to determine whether it overlaps with objects in the environment. On the other hand, we assume that the robot movement is defined and optimised in C-space. The cost functions to shape robot behaviour are often defined in the Cartesian task space. We denote C-space as $\mathcal{Q} \subseteq \mathbb{R}^d$, where there are d joints in the case of a robotic manipulator. The joint configurations, $\mathbf{q} \in \mathcal{Q}$, are elements of the C-space, while Cartesian coordinates in task space are denoted as $\mathbf{x} \in \mathcal{W}$. We now outline the procedure of *pulling* a cost gradient defined in the workspace to the C-space.

We start by defining b body points on the robot, each with a forward kinematics function ψ_i mapping configurations to the Cartesian coordinates $\mathbf{x}_i$ at the body point, $\psi_i : \mathcal{Q} \to \mathcal{W}$, for each $i = 1, \ldots, b$. Let the Jacobian of the forward kinematics functions w.r.t. the joint configurations be denoted as:

$$\mathbf{J}(\cdot)^i_\psi = \frac{\mathrm{d}\psi_i}{\mathrm{d}\mathbf{q}}(\cdot). \tag{33}$$

The derivative of a cost potential $\mathcal{C}_{\text{col}}$, which operates on the body points, such as the occupancy cost potential, can then be *pulled* into the C-space with:

$$\nabla_{\mathbf{q}}\mathcal{C} = \sum_{i=1}^{b} \mathbf{J}(\mathbf{q})^i_\psi \nabla_{\mathbf{x}}\mathcal{C}, \tag{34}$$

which allows us to update trajectory in the C-space $\mathcal{Q}$ with cost in the Cartesian space $\mathcal{W}$.

5.3.4. Combining trajectory optimisation with RRT-initialised trajectories. To further explore the applicability of SigSVGD in scenarios where initial trajectories are already in proximity to a local minimum, we conducted a modified version of the experiment illustrated in Figures 6 and 7. Instead of randomly initialising the spline knots, we apply RRT*-Connect to initialise the k-many motion planning problem, where $k = 20$ is the number of particles used in our experiment. Subsequently, we proceed with the batch optimisation procedure just like in the previous experiment. It is important to note that motion planning with RRT*-Connect is inherently sequential, foregoing the advantages of parallel computation and introducing a notable overhead. Nevertheless, we adopt this approach in a controlled experiment to assess the contribution of SigSVGD under these specific conditions. The results are shown in Figure 8. The outcomes for each simulation are more variegated than those of Figure 6, but illustrate how even under these new conditions, SigSVGD helps in finding a better solution in most of the simulated scenarios.

6. Conclusion

This work, to the best of our knowledge, is the first to introduce the use of path signatures for trajectory optimisation in robotics. We discuss how this transformation can be used as a canonical *linear* feature map to represent trajectories and how it possesses many desirable properties, such as invariance under time reparametrisation. We use these ideas to construct SigSVGD, a *sequential* kernel method to solve control and motion planning problems in a variational inference setting. It approximates the posterior distribution over optimal paths with an empirical distribution comprised of a set of vector-valued particles that encode the sequential nature of paths and which are all optimised in parallel.

In previous work, it has been shown that approaching the optimisation from the variational perspective alleviates the problem of local optimality, providing a more diverse set of solutions. We argue that the use of signatures improves on previous work and can lead to even better global properties. Despite the signature poor scalability, we show how we can construct fast and paralellisable signature kernels by leveraging recent results in rough path theory. The RKHS induced by this kernel creates a structured space that captures the sequential nature of paths. This is demonstrated through an extensive set of experiments that the structure provided helps the functional optimisation, leading to better global solutions than equivalent methods without it. We hope the ideas herein presented will serve an inspiration for further research and stimulate a groundswell of new work capitalising on the benefits of signatures in many other fields within the robotics community.

Declaration of conflicting interests

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

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs

Lucas Barcelos https://orcid.org/0000-0003-4731-5723
Tin Lai https://orcid.org/0000-0003-0641-5250

References

Al-Bluwi I, Siméon T and Cortés J (2012) Motion planning algorithms for molecular simulations: a survey. *Computer Science Review* 6(4): 125–143. DOI: 10.1016/j.cosrev.2012.07.002, URL https://linkinghub.elsevier.com/retrieve/pii/S157401371200024X

Améndola C, Friz P and Sturmfels B (2019) Varieties of signature tensors. *Forum of Mathematics, Sigma* 7: e10. DOI: 10.1017/fms.2019.3, URL https://www.cambridge.org/core/product/identifier/S2050509419000033/type/journal_article

Barcelos L, Oliveira R, Possas R, et al. (2020) DISCO: double likelihood-free inference stochastic control. In: Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA). Paris, France: IEEE Robotics and Automation Society, Vol. 7.

Barcelos L, Lambert A, Oliveira R, et al. (2021) Dual online stein variational inference for control and dynamics. In: DA Shell, M Toussaint and MA Hsieh (eds) *Robotics: Science and Systems XVII (RSS). Virtual Event*, pp. 1–12. DOI:10.15607/RSS.2021.XVII.068. URL: https://dblp.org/rec/conf/rss/BarcelosL0BBR21.bibtex.timestamp Wed, 21 Jul 2021 17:07:40 +0200.

Barfoot T, Hay Tong C and Sarkka S (2014) Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression. In: *Robotics: Science and Systems X. Robotics*. Science and Systems Foundation, 1–9. DOI: 10.15607/RSS.2014.X.001. URL https://www.roboticsproceedings.org/rss10/p01.pdf

Berntorp K, Olofsson B, Lundahl K, et al. (2014) Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres. *Vehicle System Dynamics* 52(10): 1304–1332. DOI: 10.1080/00423114.2014.939094, URL https://www.tandfonline.com/doi/abs/10.1080/00423114.2014.939094

Blei DM, Kucukelbir A and McAuliffe JD (2017) Variational inference: a review for statisticians. *Journal of the American Statistical Association* 112(518): 859–877. DOI: 10.1080/01621459.2017.1285773, URL https://www.tandfonline.com/doi/full/10.1080/01621459.2017.1285773

Boedihardjo H, Geng X, Lyons T, et al. (2016) The signature of a rough path: uniqueness. *Advances in Mathematics* 293: 720–737. DOI: 10.1016/j.aim.2016.02.011, URL https://www.sciencedirect.com/science/article/pii/S0001870816301104

Byravan A, Boots B, Srinivasa SS, et al. (2014) Space-time functional gradient optimization for motion planning. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). Hong Kong, China: IEEE, 6499–6506. DOI: 10.1109/ICRA.2014.6907818, URL https://ieeexplore.ieee.org/document/6907818/

Camacho EF and Alba CB (2013) Model predictive control. In *Advanced Textbooks in Control and Signal Processing*. 2 edition. London: Springer-Verlag. URL https://link.springer.com/book/10.1007/978-0-85729-398-5

Chamzas C, Quintero-Pena C, Kingston Z, et al. (2022) MotionBenchMaker: a tool to generate and benchmark motion planning datasets. *IEEE Robotics and Automation Letters* 7(2): 882–889. DOI: 10.1109/LRA.2021.3133603, URL https://ieeexplore.ieee.org/document/9645379/

Chen KT (1954) Iterated integrals and exponential Homomorphisms. *Proceedings of the London Mathematical Society* S4(1): 502–512. DOI:10.1112/plms/s3-4.1.502. URL https://doi.wiley.com/10.1112/plms/s3-4.1.502

Chen KT (1958) Integration of paths - a faithful representation of paths by noncommutative formal power series. *Transactions of the American Mathematical Society*. American Mathematical Society 89(2): 395–407. DOI: 10.2307/1993193. URL https://www.jstor.org/stable/1993193. Publisher

Chen KT (1977) Iterated path integrals. *Bulletin of the American Mathematical Society* 83: 831–879.

Chevyrev I and Kormilitzin A (2016) *A Primer on the Signature Method in Machine Learning*. arXiv:1603.03788 [cs, stat] URL https://arxiv.org/abs/1603.03788

Danielczuk M, Mousavian A, Eppner C, et al. (2021) Object rearrangement using learned implicit collision functions. Piscataway, NJ, USA: IEEE Press. URL https://arxiv.org/abs/2011.10726 ArXiv:2011.10726 [cs].

Dong J, Mukadam M, Dellaert F, et al. (2016) Motion planning as probabilistic inference using Gaussian processes and factor graphs. In: *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 1–9. ISBN 978-0-9923747-2-3. DOI: 10.15607/RSS.2016.XII.001. URL https://www.roboticsproceedings.org/rss12/p01.pdf

Fermanian A (2021) Embedding and learning with signatures. *Computational Statistics & Data Analysis* 157: 107148, URL https://arxiv.org/abs/1911.13211

Gammell JD and Strub MP (2021) Asymptotically optimal sampling-based motion planning methods. *Annual Review of Control, Robotics, and Autonomous Systems* 4(1): 295–318. DOI: 10.1146/annurev-control-061920-093753, URL https://www.annualreviews.org/doi/10.1146/annurev-control-061920-093753

Gonzalez D, Perez J, Milanes V, et al. (2016) A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems* 17(4): 1135–1145. DOI: 10.1109/TITS.2015.2498841, URL https://ieeexplore.ieee.org/document/7339478/

Hämäläinen P, Babadi A, Ma X, et al. (2020) PPO-CMA: Proximal policy optimization with covariance matrix adaptation. In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. 1–6. DOI: 10.1109/MLSP49062.2020.9231618.

Hambly B and Lyons T (2010) Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics* 171(1): 109–167. DOI: 10.4007/annals.2010.171.109, URL https://annals.math.princeton.edu/2010/171-1/p02

Hansen N (2016) The CMA evolution strategy: a Tutorial. *Computing Research Repository*. DOI: 10.48550/arXiv.1604.00772, URL https://arxiv.org/abs/1604.00772v1

Hansen N, Müller SD and Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary*

Computation 11(1): 1–18. DOI: 10.1162/106365603321828970, URL https://direct.mit.edu/evco/article/11/1/1-18/1139

Haugh MB (2021) A tutorial on Markov chain Monte-Carlo and Bayesian modeling. *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3759243, URL https://www.ssrn.com/abstract=3759243

Heilmeier A, Wischnewski A, Hermansdorfer L, et al. (2020) Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics* 58(10): 1497–1527. DOI: 10.1080/00423114.2019.1631455, URL https://www.tandfonline.com/doi/full/10.1080/00423114.2019.1631455

Jaillet L and Simeon T (2008) Path deformation roadmaps: compact graphs with useful cycles for motion planning. *The International Journal of Robotics Research* 27(12): 1175–1188. Sage Publications Ltd STM. DOI: 10.1177/0278364908098411.

Jones M, Baerentzen J and Sramek M (2006) 3D distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics* 12(4): 581–599. IEEE Transactions on Visualization and Computer Graphics. DOI: 10.1109/TVCG.2006.56.

Kalakrishnan M, Chitta S, Theodorou E, et al. (2011) STOMP: stochastic trajectory optimization for motion planning. In: 2011 IEEE International Conference on Robotics and Automation, May 30 to June 5, 2021, Xi'An, China, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

Kavraki L, Svestka P, Latombe JC, et al. (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580. DOI: 10.1109/70.508439, URL https://ieeexplore.ieee.org/document/508439/

King J, Klingensmith M, Dellin C, et al. (2013) Pregrasp manipulation as trajectory optimization. In: *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, 1–8. DOI: 10.15607/RSS.2013.IX.015. https://www.roboticsproceedings.org/rss09/p15.pdf

Kiraly FJ and Oberhauser H (2019) Kernels for sequentially ordered data. *Journal of Machine Learning Research* 20(31): 31–45. https://jmlr.org/papers/v20/16-314.html

Lambert A and Boots B (2021) *Entropy Regularized Motion Planning via Stein Variational Inference*. URL https://arxiv.org/abs/2107.05146 ArXiv:2107.05146 [cs].

Lambert A, Fishman A, Fox D, et al. (2020) Stein variational model predictive control.*Proceedings of the 2020 Conference on Robot Learning, Proceedings of Machine Learning Research PMLR* 155: 1278–1297.

LaValle SM (2006) *Planning Algorithms*. Cambridge; New York: Cambridge University Press. ISBN 978-0-521-86205-9. OCLC: ocm65301992.

LaValle SM and Kuffner JJ (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research* 20(5): 378–400. DOI: 10.1177/02783640122067453, URL https://journals.sagepub.com/doi/10.1177/02783640122067453

Levine S (2018) *Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review*. arXiv: 1805.00909 [cs, stat] URL https://arxiv.org/abs/1805.00909. ArXiv:1805.00909

Liu Q and Wang D (2016) Stein variational gradient descent: a general purpose Bayesian inference algorithm. In: D Lee, M Sugiyama, U Luxburg, et al. (eds) *Advances in neural information processing systems*: Curran Associates, Inc., Vol. 29, 1–13. URL https://proceedings.neurips.cc/paper/2016/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf

Loshchilov I and Hutter F (2017) SGDR: stochastic gradient descent with warm restarts. In: 5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. Toulon, France. OpenReview.net, 1–16. URL https://openreview.net/forum?id=Skq89Scxx

Lyons T (2014) Rough paths, Signatures and the modelling of functions on streams. In: *Proceedings of the International Congress of Mathematicians*. Korea. https://arxiv.org/abs/1405.4537

Marinho Z, Boots B, Dragan A, et al. (2016) Functional gradient motion planning in reproducing kernel Hilbert spaces. In: *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation. DOI: 10.15607/RSS.2016.XII.046. https://www.roboticsproceedings.org/rss12/p46.pdf

Mukadam M, Dong J, Dellaert F, et al. (2017) Simultaneous trajectory estimation and planning via probabilistic inference. In: *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-3-0. DOI: 10.15607/RSS.2017.XIII.025. https://www.roboticsproceedings.org/rss13/p25.pdf

Mukadam M, Dong J, Yan X, et al. (2018) Continuous-time Gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research* 37(11): 1319–1340. DOI: 10.1177/0278364918790369, URL https://journals.sagepub.com/doi/10.1177/0278364918790369

Ramos F and Ott L (2016) Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research* 35(14): 1717–1730 London, England: Sage PublicationsSage UK. DOI: 10.1177/0278364916684382. URL https://journals.sagepub.com/doi/10.1177/0278364916684382.Publisher

Rasmussen CE and Williams CKI (2006) *Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning*. Cambridge, Mass: MIT Press. ISBN 978-0-262-18253-9. OCLC: ocm61285753.

Ratliff N, Zucker M, Bagnell JA, et al. (2009) *CHOMP: gradient optimization techniques for efficient motion planning*. In: 2009 IEEE International Conference on Robotics and Automation. Kobe, Japan, May 12-17: IEEE, pp. 489–494. ISBN 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152817. https://ieeexplore.ieee.org/document/5152817/

Salvi C, Cass T, Foster J, et al. (2021) The signature kernel is the solution of a Goursat PDE. *SIAM Journal on Mathematics of Data Science* 3(3): 873–899. DOI: 10.1137/20M1366794, URL https://epubs.siam.org/doi/10.1137/20M1366794

Schölkopf B and Smola AJ (2002) *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond. Adaptive Computation and Machine Learning Series,*

Reprint. edition. Cambridge, Mass: MIT Press. ISBN 978-0-262-19475-4 978-0-262-53657-8.

Schulman J, Ho J, Lee A, et al. (2013) Finding locally optimal, collision-free trajectories with sequential convex optimization. In: *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation. ISBN 978-981-07-3937-9. DOI: 10.15607/RSS.2013.IX.031. https://www.roboticsproceedings.org/rss09/p31.pdf

Silverman BW (1986) *Density Estimation for Statistics and Data Analysis*. Boston, MA: Springer US. ISBN 978-0-412-24620-3 978-1-4899-3324-9. DOI: 10.1007/978-1-4899-3324-9. URL https://link.springer.com/10.1007/978-1-4899-3324-9

Theodorou EA (2015) Nonlinear stochastic control and information theoretic dualities: connections, interdependencies and thermodynamic interpretations. *Entropy* 17(5): 3352–3375 Multidisciplinary Digital Publishing Institute. DOI: 10.3390/e17053352. URL https://www.mdpi.com/1099-4300/17/5/3352.Number:5Publisher

Tjanaka B, Fontaine MC, Kalkar A, et al. (2022) Training diverse high-dimensional controllers by scaling matrix adaptation MAP-annealing. *Computing Research Repository Abs/2210* 1: 02622. DOI: 10.48550/arXiv.2210.02622.

Williams G, Drews P, Goldfain B, et al. (2016) Aggressive driving with model predictive path integral control. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). Stockholm: IEEE, 1433–1440. ISBN 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487277. URL https://ieeexplore.ieee.org/document/7487277/

Williams G, Drews P, Goldfain B, et al. (2018) Information-theoretic model predictive control: theory and applications to autonomous driving. *IEEE Transactions on Robotics* 34(6): 1603–1622. DOI: 10.1109/TRO.2018.2865891.

Yang W, Lyons T, Ni H, et al. (2017) *Developing the path signature methodology and its application to landmark-based human action recognition*. https://arxiv.org/abs/1707.03993 ArXiv: 1707.03993 [cs] version: 1.

Yu H and Chen Y (2022) A Gaussian variational inference approach to motion planning. *Computing Research Repository Abs/2209* 3: 05655. URL. DOI: 10.48550/arXiv.2209.05655.

Zhuo J, Liu C, Shi J, et al. (2018) Message passing stein variational gradient descent. In: Proceedings of the 35th International Conference on Machine Learning 10.

Zucker M, Ratliff N, Dragan AD, et al. (2013) CHOMP: covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research* 32(10): 1164–1193. Sage Publications Ltd STM. DOI: 10.1177/0278364913488805.

Appendix A

A. Experiments hyper-parameters

In Table 3, we present the relevant hyper-parameters to reproduce the results in the paper. It is worth mentioning that the terrain in the 2D motion planning is randomly generated and will vary on each simulation. Another source of randomness arises when using Monte Carlo samples to approximate the gradient of the log posterior distribution. Furthermore, due to the stochastic nature of the initial placement of the spline knots, results will vary despite using analytic gradients.

Appendix B

B. Path following example

As a motivating example in Figure 9, we depict the results of a simple two-dimensional path following task. The goal is to reduce the error between the desired path and candidate paths. Since we want the error to be as small as possible, the optimal path is one centred at the

Table 3. Hyper-parameters used in the experiments.

Parameter	2D terrain	Point-mass navigation	Manipulator benchmark
Initial state, $\mathbf{x}_s$	[0.25, 0.75]	$[-1.8, -1.8]$	Problem dependent
Environment maximum velocity	—	5 m/sec	—
Environment maximum acceleration	—	—	—
Number of spline knots, N_k	4	—	5
Number of particles, N_p	20	30	20
Particle prior	Uniform	$\mathcal{N}[X, \mathbf{1}]$	Uniform
Number of action samples, N_a	—	10	—
Cost likelihood inverse temperature, λ	1.0	1.0	1.0
Control authority, Σ	—	5^2	—
Control horizon, H	—	30	—
Stationary kernel $k(\cdot, \cdot)$	Squared-exponential	Squared-exponential	Squared-exponential
Stationary kernel bandwidth, σ	1.5	Silverman's rule	1.5
Signature kernel bandwidth, σ	1.5	5.65	1.5
Signature kernel degree, d	4	3	6
Optimiser class	Adam	Adam	Adam
Learning rate, ϵ	5×10^{-2}	1	1×10^{-3}

origin across time. The objective function is defined as a correlated multivariate normal distribution across 10 consecutive discrete time-steps such that the optimality likelihood is computed for the entire discretised path. As the cost function is convex and we are optimising the paths directly—that is, not searching for an indirect policy that generates the candidate paths—the solution is trivial. Nonetheless, the example is useful to illustrate the differences between SigSVGD and SVMP.

The initial paths are sampled from a uniform distribution and optimised with SVMP and SigSVGD for 200 iterations. The length scale of the squared-exponential kernel is computed according to Silverman's rule (Silverman, 1986) based on the initial sample for SigSVGD and updated at each iteration using the same method for SVMP. The results in Figure 9 show how both methods are able to promote diversity on the resulting paths. However, close inspection of the SVMP solution illustrates how coordinates of the candidate paths at each time-step are optimised without coordination, resulting in many paths crisscrossing and non-optimal paths close to the origin. Conversely, SigSVGD is promoting diversity of complete paths, rather than coordinates at each cross-sectional time-step, resulting in more direct paths with higher optimality likelihood.

Appendix C

C. Including hyper-priors in SigSVGD

As mentioned in Section 4.2, if one wants to constraint the feasible set of the SVGD optimisation, a *hyper-prior* can be included in the algorithm. Let $h(\cdot)$ be a hyper-prior and $p[\cdot]$ the prior distribution over particles $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, and recall that the *score function* at each update is computed according to

$$\phi^*(\mathbf{x}) = \mathbb{E}_{\mathbf{y}\sim p}\left[k^{\oplus}(\mathbf{y},\mathbf{x})\nabla_{\mathbf{y}} \log p(\mathbf{y}|\ \mathcal{O}) + \nabla_{\mathbf{y}} k^{\oplus}(\mathbf{y},\mathbf{x})\right],$$

where the posterior distribution can be factored in $\log p(\mathbf{y}|\ \mathcal{O}) = \mathcal{L}(\mathcal{O}|\ \mathbf{y}) + \log p(\mathbf{y})$. We can include the hyper-prior in the formulation by variable substitution. Let $\log \widehat{p}(\cdot) = \log p(\cdot) + \log h(\cdot)$, then

$$\begin{aligned}
\phi^*(\mathbf{x}) &= \mathbb{E}_{\mathbf{y}\sim p}\left[k^{\oplus}(\mathbf{y},\mathbf{x})\nabla_{\mathbf{y}} \log p(\mathbf{y}|\ \mathcal{O}) + \nabla_{\mathbf{x}} k^{\oplus}(\mathbf{y},\mathbf{x})\right] \\
\phi^*(\mathbf{x}) &= \mathbb{E}_{\mathbf{y}\sim p}\left[k^{\oplus}(\mathbf{y},\mathbf{x})\nabla_{\mathbf{y}}[\mathcal{L}(\mathcal{O}|\ \mathbf{y}) + \log \widehat{p}(\mathbf{y})]\right. \\
&\quad \left. + \nabla_{\mathbf{y}} k^{\oplus}(\mathbf{y},\mathbf{x})\right] \\
\phi^*(\mathbf{x}) &= \mathbb{E}_{\mathbf{y}\sim p}\left[k^{\oplus}(\mathbf{y},\mathbf{x})\nabla_{\mathbf{y}}[\mathcal{L}(\mathcal{O}|\ \mathbf{y}) + \log p(\mathbf{y})\right. \\
&\quad \left. + \log h(\mathbf{y})] + \nabla_{\mathbf{y}} k^{\oplus}(\mathbf{y},\mathbf{x})\right],
\end{aligned}$$

where $h(\cdot)$ can be any differentiable probability density function. ■

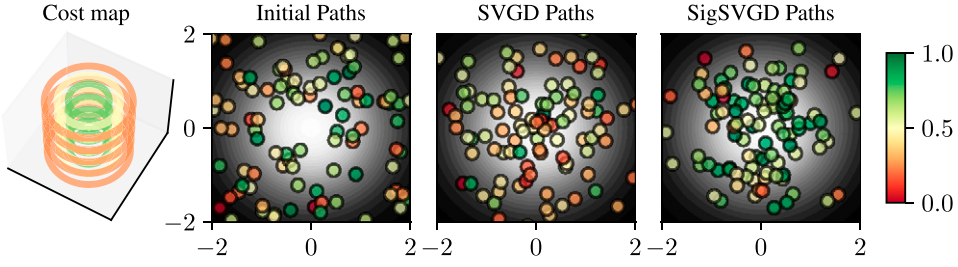

Figure 9. Qualitative analysis of trajectory-tracking task. *Left*: Contour plot of the optimality distribution over sequential time-steps (on the z-axis). *Centre-left*: Cross-section plot at a given time-step of initial path coordinates. The colour of each path indicates its normalised optimality probability. *Centre-right*: Cross-section plot of the paths after SVGD optimisation. The sampled paths are diverse and capture the variance of the target distribution. Note, however, that many non-optimal trajectories are close to the origin due to the lack of coordination between consecutive time-steps. *Right*: Cross-section plot of the paths after SigSVGD optimisation. Note how we achieve both diversity and a concentration of optimal paths near the origin.