2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Generating Continuous Paths On Learned Constraint Manifolds Using Policy Search

Ethan Canzini[1,2,*], Simon Pope[1] and Ashutosh Tiwari[1]

***Abstract*— Many robotic manipulation tasks are constrained due to kinematic limitations placed on the object being manipulated. This increases the complexity of manipulation tasks that operate in high dimensions, leading to increased risk that sampling based planners are unable to find optimal solutions. Whilst trajectory optimisation methods provide guaranteed optimal solutions when implementing constraints, they only provide locally optimal solutions in sequential decision-making and struggle to provide globally optimal paths. These constraints can be incorporated into the probabilistic latent spaces by using demonstrations that satisfy the constraint function. However whenever constraints change or a manipulator must perform different tasks the network must be retrained to accommodate the new constraints. In this paper, we provide an approach that allows the training of a single learned manifold that can be augmented to determine the constraint manifold for the manipulation task. Using this manifold, the geodesic between two points can be computed using policy search to solve the cost function associated with the geodesic curve length $\mathcal{L}_\gamma$. We provide comparisons in terms of path length against popular path planning algorithms with different kinematic constraints, demonstrating our method's ability to find optimal shortest paths on constraint manifolds.**

I. INTRODUCTION

Global path planning remains a ubiquitous part of robotic systems, underpinning the majority of control methods that exist in autonomous systems [1]. Formally, global path planning involves the decision-making system of the autonomous robot to find collision-free paths through its specified environment which can be either dynamic or static [2]. Historically, graph-search methods such as A^* and sampling-based planners (SBPs) such as RRT* have seen the most prominence due to their guarantees of finding global paths fast and their asymptotic stability for convergence. However, these methods rely on sampling the configuration space of the robot, which for certain systems such as robotic manipulators and Stewart-Gough platforms can be embedded in higher dimensions. This curse of dimensionality can lead to SBPs either exceeding time constraints for generating a global plan or failing to converge on an optimal solution, making them unsuitable for path planning high dimensions. Additionally, robots operating under movement constraints require extra care when developing planning algorithms, leading to an increase in complexity.

Optimisation methods are becoming common-place in many robotic applications. Of particular note is trajectory optimisation, which seeks to find optimal inputs for autonomous systems over a trajectory [3]. Recent work in path planning has found that trajectory optimisation methods can be useful for both legged robots and manipulators in stochastic environments with random obstacles to find locally optimal global paths [4]. Despite these advancements, trajectory optimisation methods can fail to find globally optimal solutions as they operate on a sequential decision-making process that couples together the path generation with the control aspect of the robot platform. Furthermore, much like the SBPs mentioned previously, these methods require that the configuration space of the robot be discretised, leading to disjointed paths throughout the environment.

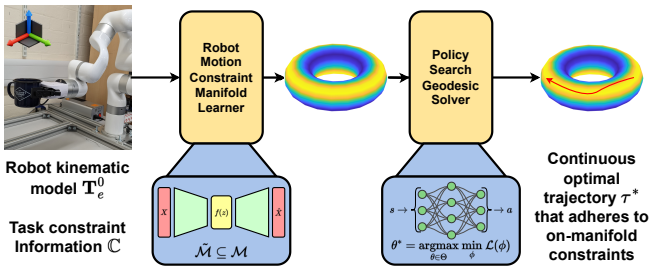


Fig. 1: The framework outlined in this paper to generate optimal paths on constraint manifolds. An approximation of the Riemannian motion manifold of a manipulator is determined through the use of a Deep VAE, which then has a sub-manifold identified by the specified constraint information. The geodesic corresponding to the shortest path is determined by policy search to obtain the optimal path.

For many manipulation tasks, autonomous robots are constrained to specific positions and orientations they can take. These constraints can be imposed by the geometry of the task being performed [5], the kinematic constraints of the robot [6] or due to shared manipulation between different robots [7]. Motion policies in these areas can be shown to act as Riemannian manifolds [8], although computing these manifolds is historically done in an online manner during the sampling process of path planners [5]. This increases the computational requirements of the planning algorithms, which is increased further when considering that manipulation tasks can be considered in either the task space or the joint space of the robot.

In this paper, we will present a novel combination of constraint manifold learning that uses policy search to find a continuous global plan across a known environment. Our contributions are listed as follows:

- We implement a manifold learning approach that, when combined with a Riemannian sub-manifold metric, can

[1]All authors are with the Department of Automatic Control & Systems Engineering, University of Sheffield, UK
[2]Ethan Canzini is a Research Scientist at Airbus Robotics, Airbus UK
[*]Corresponding Author: ecanzini1@sheffield.ac.uk

allow for the reconfiguration of the learned manifold based on the nature of the constrained manipulation task, removing the need to retrain the learned manifold when new constraints are applied
- We present a continuous path planner that uses policy search reinforcement learning to compute the shortest path on the learned manifold by utilising the properties of geodesics within Riemannian geometry.

Our results provide an approach through which better optimal solutions for manipulator trajectories can be computed, leading to further developments within autonomous decision-making. We will evaluate our approach on simple manifolds and constrained manipulation tasks, with path length comparisons against popular planning algorithms. Our code is available on GitHub here.

II. RELATED WORK

A. Learning Manifolds from Data

For a variety of data-driven applications, visualising certain variables can be difficult due to their high dimensionality. By sampling the variables, this data can be shown to exist on a manifold [8], which allows for the exploitation of functionality associated with Riemannian geometry. Riemannian manifolds can act as dimensional reducers, where a manifold is defined as a tuple $\langle \mathcal{M}, \mathbf{M} \rangle$ of manifold $\mathcal{M}$ and Riemannian metric $\mathbf{M}$ with a mapping function [9]:

$$\mathcal{M} = f(\mathcal{Z}) \text{ with } f : \mathcal{Z} \to \mathcal{X} \tag{1}$$

Both $\mathcal{Z}$ and $\mathcal{X}$ are open sub-sets of Euclidean space, with the former representing the lower dimensional manifold. Approximating this function f constitutes the process of manifold learning from data, where a stochastic variation of this function is applied to provide interpolation between points on the original data set [8].

Some early work in manifold learning applied probabilistic regression to estimate the manifold function in equation 1. Gaussian processes (GPs) [10] are a class of non-parametric stochastic functions that infer the function $f(\cdot)$ based on the distribution of the input data $\mathbf{x}$. For learning manifolds, GPs can be used to learn the mapping function $\mathbf{f}$ where the collected on-manifold data is noisy [11]. Alternatively, Calandra et al. [12] obtain a regressive model by transforming the input data into a feature space, where regression allows the computation of on-manifold data that retains its topological structure. However, these methods with vanilla GPs rely on labelled data to compute the regression process. Another flavour of GPs, known as Gaussian process latent variable models (GPLVMs) encode the input data into a latent space using GP regression. This can be used to model data that is not visibly on-manifold, such as human gait modelling to analyse the toroidal manifold that humans exhibit during motion [13]. The latent space within the GPLVM can be used to exploit differential geometry to calculate geodesics across the manifold, however Adams [14] noted that geodesics are not unique such that solving the corresponding ODE will provide multiple solutions for smooth curves.

Recent advances in generative machine learning methods have emboldened the manifold learning paradigm. This is particularly evident in the realm of constraint manifolds, where the dataset constitutes to joint positions that satisfy a constraint function $\mathbf{f}(\theta)$. In their work [15], Fernández et al. encode a data set of manipulator joint positions θ, using a variational autoencoder (VAE) deep neural network, which can then be used to sample positions through the VAE's decoder network that satisfy the constraint for sequential motion planning. A similar approach is taken by Acar and Tee [16] where they embed positions that satisfy the constraint function into both a VAE and a generative adversarial network (GAN), then compare the ability of both networks to generate a new dataset of points that satisfy the constraint. This new set of points can be then be used to generate paths using SBPs rather than planning in the latent space. Another method that allows for the generating of new paths from manifold-based data is by learning from demonstration (LfD), whereby robots learn optimal actions from human demonstrations [17]. These demonstrations can be encoded into the latent space of a VAE, either as end-effector position and orientation or via joint level commands [8]. Using these demonstrations and the learned manifold metric that is computed from the Jacobian of the VAE's encoder network, the curve length of the geodesic γ is computed as:

$$\mathcal{L}_\gamma = \int_{\mathbf{z}_0}^{\mathbf{z}_1} \sqrt{\dot{\gamma}(z)^\intercal \mathbf{M}(\gamma(z))\dot{\gamma}(z)}\ \mathrm{d}t \tag{2}$$

As there is no closed-form solution to finding geodesics on manifolds, the curve length $\mathcal{L}_\gamma$ can be interpreted as a cost function to be minimised. Beik-Mohammedi et al. [8] discuss methods of minimising this Riemannian length on the manifold by approximating it as a spline, but choose instead to discretise the manifold into a grid then execute a search algorithm across the manifold.

B. Continuous Path Planning

For all planning algorithms, the configuration space is split between the free space for the robot $\mathcal{C}_{\text{free}}$ and the space occupied by obstacles $\mathcal{C}_{\text{obs}}$. As mentioned previously, SBPs repeatedly sample $\mathcal{C}_{\text{free}}$, then construct a graph across the nodes to find the shortest path [2]. Search algorithms such as A^* and RRT* aim to find paths that minimise the distance across the graph:

$$\tau^* = \arg\min_{\tau} \sum_{i=1}^{n} J(\tau_i) \tag{3}$$

SBPs struggle when the robots begin to plan in higher dimensions, particularly with serial or parallel robotic architectures. Spurred by developments in mobile robotics, methods to solve continuous path planning problems have gained prevalence by blending together grid-search methods with optimal control to find continuous plans throughout $\mathcal{C}_{\text{free}}$ by assigning costs to nodes in the graph then interpolating across the graph [18]. Borrowing methods from trajectory optimisation techniques, Kogan and Murray use a receding

horizon optimisation approach to generate plans up to 70 metres ahead of an autonomous vehicle [19]. This method shares many similarities with trajectory optimisation, which focuses more on sequential optimisation after each new action is taken. This can lead to the coupling of path and action states, which becomes the sub-discipline of motion planning for autonomous systems [20]. Another approach is to use Bayesian optimisation to find continuous paths that pass through specific way points [21].

Similar to [8], generative machine learning techniques can be used to generate continuous paths. Diffusion models can learn to generate continuous paths by learning noisy representations of given trajectories, then denoising them at runtime. This work has shown promise with both 2D planning for legged robots [22] and with manipulation through trajectory optimisation learning [23].

III. METHODOLOGY

A. Learned Constraint Manifolds

For manipulators, there are traditional methods of representing the joint positions as an n-torus [24]. However, when moving beyond simple 2 degrees of freedom (DOF) planar robots, this representation becomes computationally expensive to evaluate. As mentioned in section II-A, manipulation profiles can be approximated using the latent space of the VAE encoder architecture based on the evidence lower bound (ELBO) loss function:

$$\mathcal{L}_{\text{ELBO}} = -D_{\text{KL}}\left[q_\zeta(z|\omega)\|p(z)\right] + \mathbb{E}_{q_\zeta}\left[\log(p_\Omega(\omega|z))\right] \quad (4)$$

As shown in [8], this latent space can be formulated as a Riemannian manifold where the pullback metric from the decoder networks is used to illustrate the confidence the network has in its estimation of the manifold. Whilst the projection process can be done online during manipulation, learning a sparse representation of the manifold from data can reduce computational complexity at run-time, whilst allowing new metrics such as obstacle avoidance and constraints to be integrated into the manifold for further utility. The metric for this manifold can be determined as the Jacobian of the mapping function, which corresponds to the mixture of the decoder's mean and variance networks [8]:

$$\mathbf{M}^\omega(z) = J_\mu^\omega(z)^\intercal J_\mu^\omega(z) + J_\sigma^\omega(z)^\intercal J_\sigma^\omega(z) \quad (5)$$

This metric alone determines the curvature of the lower dimensional manifold corresponding to the joint profile of the manipulator. However, when applying constraints to the manipulation profile, we want to determine the sub-manifold of this profile that satisfies the constraints, be they zero movement constraints or range of motion constraints.

Constraints are typically represented as task end-effector limitations, where the task may be to keep an object level or to move only in a certain rotation [5]. These constraints can be applied at the joint level by examining the end-effector position and orientation, represented in RPY Euler angles format for a 6DOF end-effector. The permitted movements by the end-effector in the task frame $\mathcal{F}_e$ is represented by the constraint vector:

$$\mathbb{C} = \begin{bmatrix} c_x & c_y & c_z & c_\beta & c_\alpha & c_\gamma \end{bmatrix} \quad (6)$$

where the constraint variable can either be a binary value $c \in \{0,1\}$ that dictate which axes are being constrained (i.e. for horizontal motion control, c_β and c_α equal 1), or a integer representing the tolerance in the movement (i.e for relaxed horizontal motion control, $c_\beta, c_\alpha \in \mathbb{R}$). This constraint vector directly affects the end-effector positions through a element-wise product against the state of the robot end effector computed using the transformation matrix $\mathbf{T}_e^0(\omega)$:

$$\begin{aligned} \mathbf{f}(\theta) &= \mathbb{C} \odot \mathbf{x}_e^0 \\ &= \mathbb{C} \odot \mathbf{T}_e^0(\omega) \end{aligned} \quad (7)$$

Given a joint vector ω, we can compute a constraint vector based on the kinematics of the robot manipulator, which can be extrapolated into a diagonal matrix diag$[\mathbf{f}(\omega)]$. Using this matrix, we can construct a sub-manifold embedded within the joint space.

Theorem 1 (Constraint Sub-Manifold): Let $\langle \mathcal{M}, \mathbf{M} \rangle$ be a manifold encompassing the free movement space of the robot $\mathcal{C}_{\text{free}}$. A constraint manifold $\langle \tilde{\mathcal{M}}, \tilde{\mathbf{M}} \rangle$ is considered embedded $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ if every point $\tilde{p} \in \tilde{\mathcal{M}}$ exists in $\mathcal{M}$ and the constraint function $\tilde{\mathbf{F}} : \mathbf{f}(\theta) \to \mathbb{R}$ exists.

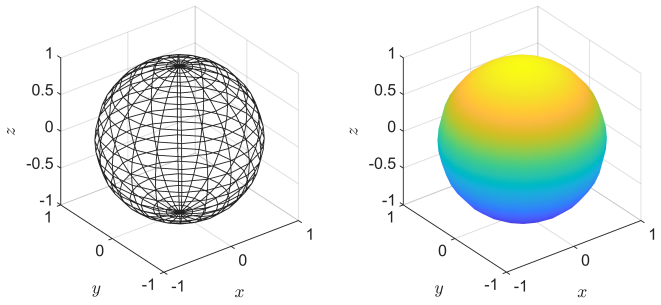
Provided that theorem 1 holds, the embedded manifold can be seen as an inclusion map $\imath_\theta : \mathbb{R}^k \to \mathbb{R}^n$ that isolates constraint manifolds from the learned manifolds using a joint space constraint function $\mathbf{f}(\theta)$. In practise, the map $\imath_\theta$ can result in either a $\mathbb{R}^{n \times n}$ matrix on the manifold tangent space $T\mathcal{M}(z)$ or a scalar value [25]. Furthermore, the use of $\imath_\theta$ as opposed to the metric allows the preservation of the metric created within the VAE manifold, allowing the curvature of the original manifold to be maintained regardless of the constraint. When evaluating this map, a value of zero indicates the constraint has been satisfied for the joint position θ and a non-zero value indicates deviation from the constraint. This indicates that the constraint function $\tilde{\mathbf{F}}$ creates a set of manifolds that satisfy the constraint:

$$\begin{aligned} \tilde{\mathcal{M}}_\theta &= \left\{ \tilde{\mathcal{M}}_i \subseteq \mathcal{M} \,|\, \imath_\theta \right\} \\ \text{where } \tilde{\mathcal{M}}_i &= \{ z \in \mathcal{Z} \,|\, f(z) \}, \text{ for } i = 1 \ldots N \end{aligned} \quad (8)$$

where N is the number of sub-manifolds arising from the inclusion map $\imath_\theta$. An example of this mapping shown in figure 2 for an $\mathcal{S}^2$ sphere and $\mathbb{T}^2$ torus embedded in Euclidean space.

B. Finding Geodesics Via Policy Search

As shown in equation 2, the curve length of a geodesic is a function of the Riemannian metric on the manifold. In two dimensions, this geodesic can be approximated as a spline with $\mathcal{C}_1(t)$ and $\mathcal{C}_2(t)$:



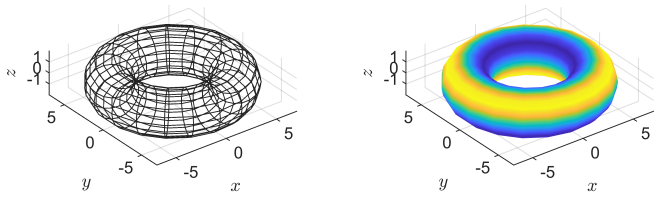


Fig. 2: Manifolds with induced metrics: **(a)** (LEFT) A sphere $\mathcal{S}^2$ embedded into $\mathbb{R}^3$ with a metric (RIGHT) corresponding to point elevation, creating a sub-manifold equivalent to an $\mathcal{S}^1$ sphere; **(b)** (LEFT) A torus $\mathbb{T}^2 = \mathcal{S}^1 \times \mathcal{S}^1$ embedded into $\mathbb{R}^3$ with a metric (RIGHT) corresponding to the angle around the torus, creating a set of $\mathcal{S}^1$ manifolds that satisfy the constraint.

$$\begin{aligned} \mathcal{C}_1(t) &= \phi_{1,0} + \phi_{1,1}t + \ldots + \phi_{1,k}t^k \\ \mathcal{C}_2(u) &= \phi_{2,0} + \phi_{2,1}t + \ldots + \phi_{2,k}t^k \end{aligned} \tag{9}$$

where k is the number of the exponents in the spline approximation, equal to 3 when using cubic splines [21]. For higher N-dimensional manifolds, a generalised form can be taken as:

$$\gamma(z) \approx \sum_{n=1}^{N} \mathcal{C}_n(t) = \sum_{n=1}^{N} \sum_{k=0}^{K} \phi_{n,k}t^k \tag{10}$$

where N is the dimensionality of the latent space manifold and K is the number of exponents to approximate the spline to. The dimensionality of the latent space is normally visualised as $N \in \{2,3\}$, which in this work we will work with $N = 2$. As noted in [21], the values of $\phi_{n,0}$ are the starting coordinates of the path, meaning that the approximation in equation 10 becomes:

$$\gamma(z) \approx \left[\sum_{n=1}^{N} \left(\sum_{k=1}^{K} \phi_{n,k}t^k \right) + \phi_{n,0} \right] \tag{11}$$

In the case of the cubic spline in two dimensions where $\mathcal{C}_1(t) = \mathbf{x}(t)$ and $\mathcal{C}_2(t) = \mathbf{y}(t)$, this reduces the action space to $K = 6$ and this formulates our integral from equation 2 which is in terms of our latent space variable z into the parameterised curve length:

$$\mathcal{L}_\gamma(t) = \int_{t=0}^{t=1} \sqrt{\dot{\mathbf{x}}(t)^\intercal \mathbf{M} \dot{\mathbf{x}}(t) + \dot{\mathbf{y}}(t)^\intercal \mathbf{M} \dot{\mathbf{y}}(t)}\ \mathrm{d}t \tag{12}$$

As we are concerned with minimising the curve length of the geodesic on the manifold, we can rephrase this as an optimisation problem:

$$\mathcal{J}(\gamma) = \underset{\gamma}{\text{minimise}}\ \mathcal{L}_\gamma(t) \tag{13}$$

The dimensionality of γ means that traditional optimisation methods will struggle to find globally optimal solutions. Reinforcement learning (RL) has been used as a suitable alternative for high-dimensional systems, and has demonstrated the ability to find geodesics on manifolds [26]. Due to the synonymous nature of the geodesic formula in equation 13, policy search represents a fast optimiser that can discover global solutions in a data efficient manner. Policy search uses a parameterised policy $\pi_\theta : \Theta \times \mathcal{S} \to \mathcal{A}$ to map a state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. The optimal parameters θ^* for the policy π_θ^* in an episodic RL problem as:

$$\theta^* = \underset{\theta \in \Theta}{\arg\max}\ \mathbb{E}\left[\sum_{i=0}^{I} r_i \right] \tag{14}$$

To model the geodesic optimisation process as a Markov Decision process (MDP), we can model the state as the current values of $\phi_{n,k}$ from equation 9 and the action space as value changes to these parameters. The function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ provides a reward corresponding to the change in the curve length due to the new state. As we are ultimately concerned with minimising the curve length, we define our reward function in terms of the curve length and the change in the curve length, with the difference corresponding to $\Delta_i = \mathcal{L}_\gamma^i - \mathcal{L}_\gamma^{i+1}$. The Gompertz function, from the family of logistic functions, can be used to provide a sliding reward function in the range $r_i \in [-1,1]$, whereby larger values of reward are provided to the agent when the curve length reduction is greater. The two parameters in the function, b and c, can be seen as hyper-parameters that can be tuned. When rewarding the agent, we want to ensure that the agent transitions to a shorter curve length in the shortest amount of time-steps per episode. Therefore we compose our overall reward per step r_i based on the curve length $\mathcal{L}_i$ and the change in curve lengths Δ_i, with constants $K_\mathcal{L}$ and K_Δ allowing the tuning of the specific rewards based on the application:

$$\begin{aligned} r_i &= K_\mathcal{L} R_\mathcal{L} + K_\Delta R_\Delta \\ \text{where } R(x) &= e^{-be^{-cx}} \end{aligned} \tag{15}$$

and we further define $R_\mathcal{L} := R(\mathcal{L}_i)$ and $R_\Delta = R(\Delta_i)\mathbb{1}_{\{\Delta_i > 0\}}$ where $\mathbb{1}$ is the indicator function. This enables the agent to receive a higher reward when transitioning to a curve of lower length with a greater change in the curve length.

To find the optimal policy parameters θ^*, we use proximal policy optimisation (PPO) [27] to optimise the parameters for the episodic return shown in equation 14. A simple example in $\mathbb{R}^2$, where the agent finds optimal values $\phi_{n,k}^* \in \Phi$ for a curve is shown in figure 3.

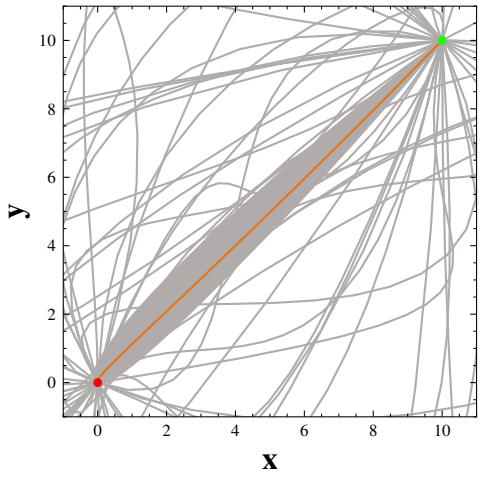


Fig. 3: Using policy search to find the shortest path in $\mathbb{R}^2$. The lines in grey indicate sampled paths, with the learned shortest path shown in orange. In this space, as the manifold is Euclidean, $\mathbf{M} = 1$ and the curve length is reduced to $\int_0^1 \sqrt{\dot{\mathbf{x}}^2 + \dot{\mathbf{y}}^2}\ \mathrm{d}t$

IV. RESULTS

For our experiments, we consider the task of manipulating a cup of water by a 6 degree of freedom (DOF) xArm 6 robot manipulator, shown in figure 1. This imposes a hard constraint regarding the orientation in the roll and pitch axes to avoid spillage of the fluid. The resulting learned constraint manifold is used for the planning stage, where we will compare its performance in terms of path length against popular path planning algorithms.

A. Constraint Manifold Learning

To learn the constraint manifold, we first learn an approximation of the joint position manifold. This is is done by creating a dataset of joint configurations, then training a deep VAE model as done in [8]. The latent space manifold, denoted as $\mathbf{z} \in \mathcal{Z}$, can be visualised in figure 4a. The Riemannian metric from equation 5 represents the curvature of the manifold, with the color bar representing this curvature.

Using this result, we can apply the map ι_θ to extract the resulting constraint manifold. As we have maintained the curvature of the manifold, the resulting manifold is in the same dimension but is a reduced sub-manifold that satisfy the constraint map. This result, shown in figure 4b, shows how the resulting sub-manifold maintains the curvature and dimensions of the original, yet only certain regions now remain due to the desired constraint.

B. Continuous Path Planning

To compare our learned geodesic against current SBPs, we use three different planners as comparisons: Probabilisic roadmap (PRM), RRT* and bi-directional RRT (BiRRT) [2]. These planners generate paths in the joint space of the robot where they project their sampled positions by evaulating the constraint at each time step, which is then embedded into the latent space of our manifold to compute the path length. As noted in [8], the shortest geodesic in the manifold constitutes to the shortest path in Euclidean space, allowing us to compare our learned geodesic against the SBP paths. We compare our method in terms of shortest path generated over a set of policies and the average path length. An example of the learned geodesic generated by the policy is shown in figure 4c.

TABLE I: Comparison of path lengths and planning times of our method for geodesic learning against SBPs. Path lengths are unit-less and dimensionless and correspond to the shortest path across the manifold

Method	Shortest Path	Average	Planning Time (s)
PRM	499.42	530.89 ± 24.43	9.75 ± 1.53
RRT*	134.80	155.28 ± 17.84	5.21 ± 1.38
BiRRT	53.00	84.65 ± 25.52	2.903 ± 0.16
Learned Geodesic	70.56	79.81 ± 7.64	3.8 ± 1.81

As shown in table I, the learned geodesic is able to generate far shorter paths when compared against other methods, with the only comparison being to the BiRRT method. This method bears the most similarity to ours, as it seeks to connect the start and end nodes through sampling the free configuration space. However, BiRRT is limited by it's sampling method and that it may not generate the same path again or may use more nodes than necessary when computing a trajectory. In contrast, our learned geodesic represents the shortest path between two points on the manifold and does not require sampling the free space. This curve can then be decoded through the VAE's decoder to generate a continuous path in the joint space of the manipulator.

V. CONCLUSIONS

This work presents a novel approach to continuous global path planning by exploiting the curvature of the resulting constraint manifold. Our work used a learned constraint manifold that combines a deep VAE representation manifold of a manipulator's joint space with an inclusion map that finds the sub-manifold satisfying the movement constraint. Exploiting this sub-manifold and its curvature allows the computation of a geodesic using a cubic spline approximation with reinforcement learning to find the shortest continuous path on this manifold. Our method is able to out-perform classic SBPs in terms of path length, but would benefit from future work to analysis the computational cost of computing these paths against SBPs due to the path integration calculation. Additionally, methods such as PPO require many data points before converging on an optimal solution. Further work would implement the use of parallel processing to improve the numerical integration for the curve-length and reduce the runtime inline with SBPs shown in table I. Additionally, a comparison in compute time and resources against other optimisation methods such as that in [3] would solidify our method in its performance. An additional implementation task would be to generalise beyond MDPs into partially observable MDPs to allow for real-time obstacle avoidance in unknown environments.

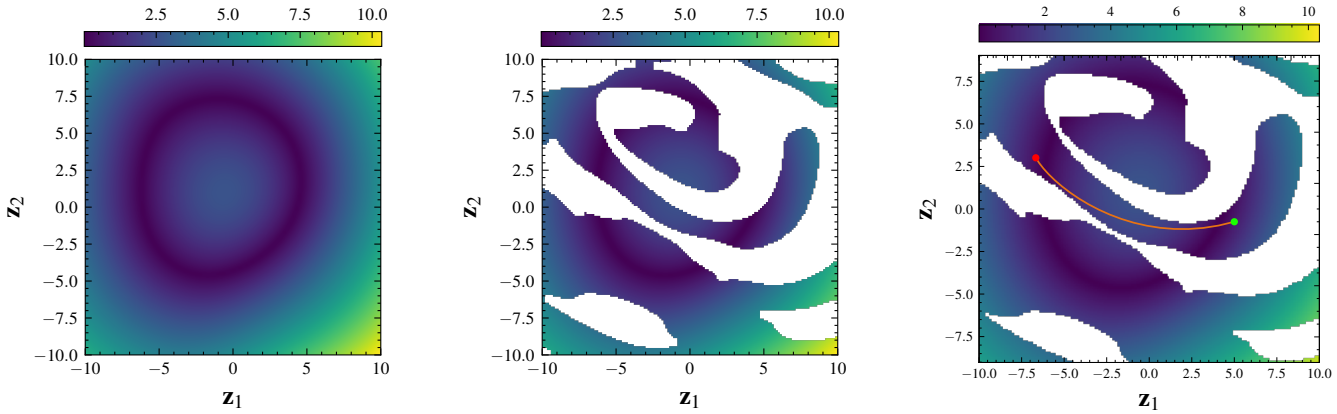


Fig. 4: Results of training the learned constraint manifold and computing the geodesic: **(a)** Plot of the latent space of the VAE. This resulting manifold is without any constraints, with the equation 5 constituting the curvature of the manifold; **(b)** Plot of the resulting constraint manifold. The curvature from the prior plot is maintained, but this sub-manifold constitutes all the latent space positions that satisfy the impose horizontal constraint; **(c)** A learned shortest-path geodesic using policy search embedded into the manifold.

ACKNOWLEDGMENTS

The work of Ethan Canzini was supported by the EPSRC ICASE Award with Airbus UK and was co-sponsored by the University of Sheffield and Airbus UK Assembly Technologies. This work received funding from the UKRI EPSRC Made Smarter Innovation-Research Centre for Connected Factories (Grant EP/V062123/1) and by the RAEng/Airbus Research Chairs & Senior Research Fellowships Scheme (Grant RCSRF1718/5/41).

REFERENCES

[1] C. Zhou *et al.*, "A review of motion planning algorithms for intelligent robots," en, *Journal of Intelligent Manufacturing*, vol. 33, no. 2, pp. 387–424, Feb. 2022.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[3] B. Sundaralingam *et al.*, "Curobo: Parallelized collision-free robot motion generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 8112–8119.

[4] M. Kalakrishnan *et al.*, "STOMP: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574.

[5] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, Jun. 2010.

[6] L. Jaillet and J. M. Porta, "Path planning under kinematic constraints by rapidly exploring manifolds," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 105–117, Feb. 2013.

[7] P. Englert *et al.*, "Sampling-based motion planning on sequenced manifolds," vol. 17, Jul. 2021.

[8] H. Beik-Mohammadi *et al.*, "Reactive motion generation on learned riemannian manifolds," en, *The International Journal of Robotics Research*, vol. 42, no. 10, pp. 729–754, Sep. 2023.

[9] J. M. Lee, *Introduction to riemannian manifolds* (Graduate Texts in Mathematics). Springer, 2018.

[10] K. P. Murphy, *Probabilistic Machine Learning: Advanced Topics*. The MIT Press, 2023.

[11] D. B. Dunson and N. Wu, "Inferring manifolds from noisy data using gaussian processes," no. arXiv:2110.07478, Oct. 2022.

[12] R. Calandra *et al.*, "Manifold gaussian processes for regression," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 3338–3345.

[13] M. Ding and G. Fan, "Multilayer joint gait-pose manifolds for human gait motion modeling," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2413–2424, Nov. 2015.

[14] L. C. Adams, "Gaussian process manifold learning," en, Doctoral Thesis, Vanderbilt University, Nashville, Tenessee, Jan. 2021.

[15] I. M. R. Fernández *et al.*, "Learning manifolds for sequential motion planning," in *RSS 2020 Workshop on Learning Task and Motion Planning*, Virtual, Jul. 2020.

[16] C. Acar and K. P. Tee, "Approximating constraint manifolds using generative models for sampling-based constrained motion planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 8451–8457.

[17] T. Osa *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1–2, pp. 1–179, 2018.

[18] J. R. Sánchez-Ibáñez *et al.*, "Path planning for autonomous mobile robots: A review," en, *Sensors*, vol. 21, no. 2323, p. 7898, Jan. 2021.

[19] D. Kogan and R. M. Murray, "Optimization-based navigation for the darpa grand challenge," en, in *Proc. 45th Conference on Decision & Control*, IEEE, 2006.

[20] A. H. Qureshi *et al.*, "Constrained motion planning networks x," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 868–886, Apr. 2022.

[21] R. Marchant and F. Ramos, "Bayesian optimisation for informative continuous path planning," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6136–6143.

[22] J. Liu *et al.*, "DiPPeR: Diffusion-Based 2D Path Planner Applied on Legged Robots," en, in *2024 IEEE International Conference on Robotics and Automation (ICRA 2024)*, Yokohama, Japan: IEEE, May 2024.

[23] M. Janner *et al.*, "Planning with diffusion for flexible behavior synthesis," en, in *Proceedings of the 39th International Conference on Machine Learning*, PMLR, Jun. 2022, pp. 9902–9915.

[24] J. W. Burdick, "Global kinematics for manipulator planning and control," vol. 1196, SPIE, Feb. 1990, pp. 57–69.

[25] K. Jang *et al.*, "Motion planning for closed-chain constraints based on probabilistic roadmap with improved connectivity," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 4, pp. 2035–2043, Aug. 2022.

[26] D. Kious *et al.*, "Finding geodesics on graphs using reinforcement learning," *The Annals of Applied Probability*, vol. 32, no. 5, pp. 3889–3929, Oct. 2022.

[27] J. Schulman *et al.*, "Proximal policy optimization algorithms," Jul. 2017, arXiv: 1707.06347 Citation Key: Schulman2017.