Bayesian Optimization Meets Riemannian Manifolds in Robot Learning

Noémie Jaquier[1,2] **Leonel Rozo**[1] **Sylvain Calinon**[2] **Mathias Bürger**[1]
[1]Bosch Center for Artificial Intelligence
Renningen, Germany
name.surname@de.bosch.com
[2]Idiap Research Institute
Martigny, Switzerland
name.surname@idiap.ch

Abstract: Bayesian optimization (BO) recently became popular in robotics to optimize control parameters and parametric policies in direct reinforcement learning due to its data efficiency and gradient-free approach. However, its performance may be seriously compromised when the parameter space is high-dimensional. A way to tackle this problem is to introduce domain knowledge into the BO framework. We propose to exploit the geometry of non-Euclidean parameter spaces, which often arise in robotics (e.g. orientation, stiffness matrix). Our approach, built on Riemannian manifold theory, allows BO to properly measure similarities in the parameter space through geometry-aware kernel functions and to optimize the acquisition function on the manifold as an unconstrained problem. We test our approach in several benchmark artificial landscapes and using a 7-DOF simulated robot to learn orientation and impedance parameters for manipulation skills.

Keywords: Bayesian optimization, Riemannian manifolds, Robot learning

1 Introduction

When robots learn new skills or adapt their behavior to unseen conditions, their learning process needs to be safe, fast and data-efficient as the robot is a physical system interacting with the environment, making every single interaction costly. In reinforcement learning (RL) for robotics, Bayesian Optimization (BO) [1] has gained increasing interest due to its success on optimizing parametric policies in several challenging scenarios [2, 3, 4, 5]. Its popularity is due to its ability to model complex noisy cost functions in a data-efficient manner, contrasting to data-hungry methods used in deep RL [6]. However, BO performance degrades as the search space dimensionality increases, opening the door to different approaches dealing with the curse of dimensionality [7, 8, 9]. Its performance also depends on the generalization capabilities of Gaussian process (GP) models (the common surrogate model of BO), which is strongly impacted by the definition of both the mean and kernel functions.

A recent approach to improve BO performance is via domain knowledge, commonly introduced into the GP mean function [2] or through the design of task-specific kernels [10], as detailed in § 2. Nevertheless, several of these solutions are not task-agnostic, requiring new kernels as the task domain varies. A more scalable approach is to provide domain knowledge that generalizes over several tasks. In this line, we propose to provide BO with information about the geometry of the search space, a key feature often overlooked in BO applications. Geometry-awareness is particularly relevant when the parameter space is not Euclidean, which is common in robotic applications, where a variety of manifolds arise [11, 12, 13]. For example, forces and torques belong to the Euclidean manifold $\mathbb{R}^D$, stiffness, inertia and manipulability lie in the manifold of symmetric positive definite matrices $\mathcal{S}_{++}$, the special orthogonal group $SO(3)$ or the unit-sphere $\mathcal{S}^3$ are used to represent orientations, and the special Euclidean group $SE(3)$ describes robot poses.

We hypothesize that bringing geometry-awareness into BO may improve its performance and scalability. To do so, we bring Riemannian manifold theory to Bayesian optimization (see § 3 for a short background). We first propose to use geometry-aware kernels allowing GP to properly measure the similarity between parameters lying on a Riemannian manifold. Second, we exploit Riemannian manifold tools to consider the geometry of the search space when optimizing the acquisition func-

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

tion (see § 4). These two contributions lead to a fully geometry-aware BO framework (hereinafter called GaBO) which naturally handles the constraints of parameters lying on smooth differentiable manifolds. We test GaBO with different benchmark functions and, as a proof of concept, in a set of simulated scenarios aimed at refining simple control policies for robotic manipulation (see § 5). Our results show that GaBO outperforms the classical BO formulation in convergence, accuracy and scalability as the space dimensionality increases.

2 Related Work

BO has been widely applied in diverse robotic applications, such as behavior adaptation for damaged legged robots [2], controller tuning for balancing [14], biped locomotion [10, 5], whole body control [15], physical human-robot interaction [16, 17], and manipulation [18]. A key aspect of the success of BO is the use of domain knowledge, mainly introduced into the surrogate model or the acquisition function. This prior information aims at decreasing the problem complexity and improving the convergence and accuracy of BO. This section reviews how domain knowledge has been exploited to improve BO performance.

Cully et al. [2] used simulated walking behaviors as prior knowledge that allowed a robot to quickly adapt to drastic hardware changes. Pautrat et al. [19] extended this idea and exploited the GP mean function to introduce prior knowledge about the robot behavior, which was later used to learn new tasks using a pool of given priors. They also proposed a new acquisition function that harmonized the expected improvement and the prior model likelihood. Wilson et al. [20] used trajectory data extracted during policy execution to learn the initial state distribution, the transition and reward functions. These were used to generate Monte-Carlo estimates of policy performance that were used to define the GP mean function. The authors also introduced a Kullback-Leibler divergence kernel that used trajectory information to measure the relatedness between policies. In contrast to these works, we do not focus on imposing prior knowledge through the GP mean function but instead exploit the geometry of the parameter space to drive the BO exploration.

Marco et al. [14] tuned the parameters of a pole-balancing robot controller using BO with an acquisition function maximizing the information gain. The authors emphasized that a GP with common kernels may degrade the learning outcome [4], and thus kernels leveraging the controller structure may be preferred. Thus, they proposed two kernels exploiting the structure of linear quadratic regulators, which outperformed the common squared-exponential (SE) kernel. The shortcomings of the SE kernels were also analyzed by Martinez-Cantin [21], who proposed a set of adaptive kernels designed to model functions from nonstationary processes. This proved useful in direct policy search where failures result in large discontinuities or flat regions while fast variations are observed around the optimal policy. Rai et al. [5] used a gait feature transformation to generalize a range of locomotion controllers and robot morphologies. This transform reparameterized the original space of controller parameters to a 1D space, where a classical SE kernel was used. Antonova et al. [10] used a neural network to learn a kernel using simulated bipedal locomotion patterns. This neural-network kernel could be learned from the BO cost directly or from the trajectories in a cost-agnostic manner. Unlike these works, we do not design task-specific kernels but instead exploit geometry-aware kernel functions that can be used for different tasks whose search spaces share the same geometry.

Another way to introduce domain knowledge into BO is via the acquisition function. Englert and Toussaint [3] and Drie et al. [18] included domain knowledge into the tuning of robot control parameters through an acquisition function using success information embedded into a GP classifier. Yuan et al. [15] proposed a search space partitioning method to tackle the high dimensionality of a whole-body quadratic programming controller. The acquisition function was given only a subset of independent physically-meaningful partitions of the parameter space. In this line, our work includes domain knowledge into the acquisition function by exploiting geometry-aware optimization that handle parameters lying on Riemannian manifolds.

The work by Oh et al. [22] is one of the few in BO literature where geometry-awareness is considered. The authors applied a cylindrical transformation to the search space to overcome boundary issues when a sphere-like domain is given, and used a geometry-aware kernel. Their method was more accurate, efficient and scalable when compared to state-of-the art BO. However, Oh et al. [22] did not include geometry information into the optimization of the acquisition function. As mentioned previously, our approach is fully geometry-aware as both the kernel function and the optimization of the acquisition function consider the geometry of the search space.

3 Background

3.1 Bayesian Optimization

Bayesian optimization (BO) is a sequential search algorithm aiming at finding a global maximizer (or minimizer) of an unknown objective function f

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}}\, f(\boldsymbol{x}), \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^{D_\mathcal{X}}$ is some design space of interest, with $D_\mathcal{X}$ being the dimensionality of the parameter space. The black-box function f has no simple closed form, but can be observed point-wise by evaluating its value at any arbitrary query point $\boldsymbol{x}$ in the domain. This evaluation produces noise-corrupted (stochastic) outputs $y \in \mathbb{R}$ such that $\mathbb{E}[y|f(\boldsymbol{x})] = f(\boldsymbol{x})$, with observation noise σ.

In this setting, BO specifies a prior belief over the possible objective functions. Then, at each iteration n, this model is refined according to observed data $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ via Bayesian posterior update. An acquisition function $\gamma_n : \mathcal{X} \mapsto \mathbb{R}$ is constructed to guide the search for the optimum. This function evaluates the utility of candidate points for the next evaluation of f; therefore, the next query point $\boldsymbol{x}_{n+1}$ is selected by maximizing γ_n, i.e. $\boldsymbol{x}_{n+1} = \operatorname{argmax}_{\boldsymbol{x}} \gamma_n(\boldsymbol{x}; \mathcal{D}_n)$. After N queries, the algorithm makes a final recommendation $\boldsymbol{x}_N$, representing its best estimate of the optimizer.

The prior and posterior of f are commonly modeled using a Gaussian Process $\mathcal{GP}(\mu, k)$ with mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and positive-definite kernel (or covariance function) $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. Therefore, the function f follows a Gaussian prior $f(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$ where $\boldsymbol{\mu}_i = \mu(\boldsymbol{x}_i)$ and $\boldsymbol{K}$ is the covariance matrix with $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. With $\tilde{\boldsymbol{x}}$ representing an arbitrary test point, the random variable $f(\tilde{\boldsymbol{x}})$ conditioned on observations is also normally distributed with the following posterior mean and variance functions:

$$\mu_n(\tilde{\boldsymbol{x}}) = \mu(\tilde{\boldsymbol{x}}) + \boldsymbol{k}(\tilde{\boldsymbol{x}})^\mathsf{T}(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y}-\boldsymbol{\mu}) \text{ and } \sigma_n^2(\tilde{\boldsymbol{x}}) = k(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - \boldsymbol{k}(\tilde{\boldsymbol{x}})^\mathsf{T}(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}(\tilde{\boldsymbol{x}}), \tag{2}$$

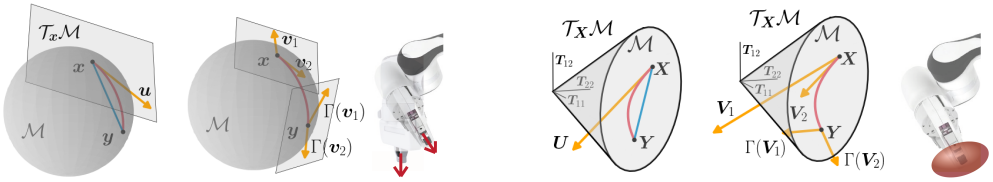
where $\boldsymbol{k}(\tilde{\boldsymbol{x}})$ is a vector of covariance terms between $\tilde{\boldsymbol{x}}$ and the observations $\boldsymbol{x}_i$ [23]. The posterior mean and variance evaluated at any point $\tilde{\boldsymbol{x}}$ respectively represent the model prediction and uncertainty of the objective function at $\tilde{\boldsymbol{x}}$. In BO, these functions are exploited to select the next query $\boldsymbol{x}_{n+1}$ by means of the acquisition function. The mean and kernel functions completely specify the GP and thus the model of the function f. The most common choice for the mean function is a constant value, while the kernel typically has the property that close points in the input space have stronger correlation than distant points. One popular kernel is the squared-exponential (SE) kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \exp(-\beta d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2)$, where $d(\cdot, \cdot)$ denotes the distance between two observations and the parameters β and θ control the horizontal and vertical scale of the function. The kernel parameters and the observation noise are usually inferred via maximum likelihood estimation (MLE).

The acquisition function balances exploitation (e.g. selecting the point with the highest posterior mean) and exploration (e.g. selecting the point with the highest posterior variance) using the information given by the posterior functions. In this paper, we use an improvement-based acquisition function, namely, expected improvement (EI) [24]. For EI, the next query intuitively corresponds to the point where the expected improvement over the previous best observation f_n^* is maximal.

3.2 Riemannian Manifolds

In robotics, diverse type of data do not belong to a vector space and thus the use of classical Euclidean space methods for treating and analyzing these variables is inadequate. A common example is the unit quaternion, widely used to represent orientations. The quaternion has unit norm and therefore can be represented as a point on the surface of a 3-sphere. Symmetric positive definite (SPD) matrices are also widely used in robotics in the form of stiffness and inertia matrices, or manipulability ellipsoids. Both the sphere and the space of SPD matrices can be endowed with a Riemannian metric to form Riemannian manifolds. Intuitively, a Riemannian manifold $\mathcal{M}$ is a mathematical space for which each point locally resembles an Euclidean space. For each point $\boldsymbol{x} \in \mathcal{M}$, there exists a tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ equipped with a smoothly-varying positive definite inner product called a Riemannian metric. This metric permits to define curve lengths on the manifold. These curves, called geodesics, are the generalization of straight lines on the Euclidean space to Riemannian manifolds, as they represent the minimum length curves between two points in $\mathcal{M}$ (see Fig. 1).

To utilize the Euclidean tangent spaces, we need mappings back and forth between $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ and $\mathcal{M}$, which are known as exponential and logarithmic maps. The exponential map $\operatorname{Exp}_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$



(a) Sphere manifold $\mathcal{S}^2$ (incl. e.g. orientations). (b) SPD manifold $\mathcal{S}^2_{++}$ (incl. e.g. stiffness ellipsoids)

Figure 1: (a) Points on the surface of the sphere, such as $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the manifold. (b) One point corresponds to a matrix $\left(\begin{smallmatrix} T_{11} & T_{12} \\ T_{12} & T_{22} \end{smallmatrix}\right) \in \text{Sym}^2$ in which the manifold is embedded. For all the graphs, the shortest path between $\boldsymbol{x}$ and $\boldsymbol{y}$ is the geodesic represented as a red curve, which differs from the Euclidean path depicted in blue. $\boldsymbol{u}$ lies on the tangent space of $\boldsymbol{x}$ such that $\boldsymbol{u} = \text{Log}_{\boldsymbol{x}}(\boldsymbol{y})$. $\Gamma(\boldsymbol{v}_1)$, $\Gamma(\boldsymbol{v}_2)$ are the parallel transported vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ from $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ to $\mathcal{T}_{\boldsymbol{y}}\mathcal{M}$. The inner product between vectors is conserved by this operation.

Table 1: Principal operations on the sphere $\mathcal{S}^d$ and SPD manifold $\mathcal{S}^D_{++}$ (see [25, 26, 27] for details).

Manifold	$d_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y})$	$\text{Exp}_{\boldsymbol{x}}(\boldsymbol{u})$	$\text{Log}_{\boldsymbol{x}}(\boldsymbol{y})$
S^d	$\arccos(\boldsymbol{x}^\mathsf{T}\boldsymbol{y})$	$\boldsymbol{x}\cos(\|\boldsymbol{u}\|) + \overline{\boldsymbol{u}}\sin(\boldsymbol{u})$	$d(\boldsymbol{x},\boldsymbol{y})\,\frac{\boldsymbol{y}-\boldsymbol{x}^\mathsf{T}\boldsymbol{y}\,\boldsymbol{x}}{\|\boldsymbol{y}-\boldsymbol{x}^\mathsf{T}\boldsymbol{y}\,\boldsymbol{x}\|}$
S^D_{++}	$\|\log(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{Y}\boldsymbol{X}^{-\frac{1}{2}})\|_\text{F}$	$\boldsymbol{X}^{\frac{1}{2}}\exp(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{U}\boldsymbol{X}^{-\frac{1}{2}})\boldsymbol{X}^{\frac{1}{2}}$	$\boldsymbol{X}^{\frac{1}{2}}\log(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{Y}\boldsymbol{X}^{-\frac{1}{2}})\boldsymbol{X}^{\frac{1}{2}}$
	Parallel transport $\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v})$		
S^d	$\Big(-\boldsymbol{x}\sin(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^\mathsf{T} + \overline{\boldsymbol{u}}\cos(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^\mathsf{T} + (\boldsymbol{I} - \overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}^\mathsf{T})\Big)\boldsymbol{v}$ with $\overline{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$		
S^D_{++}	$\boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}}\,\boldsymbol{V}\,\boldsymbol{A}^\mathsf{T}_{\boldsymbol{X}\to\boldsymbol{Y}}$ with $\boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}} = \boldsymbol{Y}^{\frac{1}{2}}\boldsymbol{X}^{-\frac{1}{2}}$		

maps a point $\boldsymbol{u}$ in the tangent space of $\boldsymbol{x}$ to a point $\boldsymbol{y}$ on the manifold, so that it lies on the geodesic starting at $\boldsymbol{x}$ in the direction $\boldsymbol{u}$ and such that the geodesic distance $d_{\mathcal{M}}$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ is equal to norm of the distance between $\boldsymbol{x}$ and $\boldsymbol{u}$. The inverse operation is called the logarithmic map $\text{Log}_{\boldsymbol{x}} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. Another useful operation over manifolds is the parallel transport $\Gamma_{\boldsymbol{x}\to\boldsymbol{y}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{T}_{\boldsymbol{y}}\mathcal{M}$, which moves elements between tangent spaces such that the inner product between two elements in the tangent space remains constant.

In this paper, we illustrate our approach in two manifolds widely used in robotics, namely the sphere and SPD manifolds. The unit sphere $\mathcal{S}^d$ is a d-dimensional manifold embedded in $\mathbb{R}^{d+1}$. The tangent space $\mathcal{T}_x\mathcal{S}^d$ is the hyperplane tangent to the sphere at $\boldsymbol{x}$. The manifold of $D \times D$ SPD matrices $\mathcal{S}^D_{++}$ can be represented as the interior of a convex cone embedded in its tangent space of symmetric matrices Sym^D. These manifolds and the foregoing operations are illustrated in Figure 1 and mathematically described in Table 1.

4 Bayesian Optimization on Riemannian Manifolds

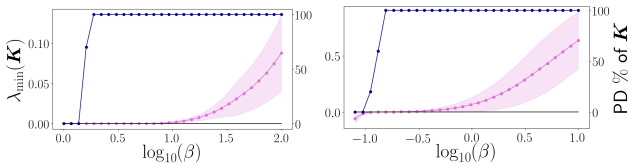
In this section, we present the geometry-aware BO (GaBO) framework that naturally handles the cases where the design space of parameters $\mathcal{X}$ is a Riemannian manifold or a subspace of a Riemannian manifold, i.e. $\mathcal{X} \subseteq \mathcal{M}$. To do so, we first need to model the unknown objective function f with a Gaussian process adapted to manifold-valued data. This is achieved by defining geometry-aware kernels measuring the similarity of the parameters on the manifold. Moreover, the selection of the next query point $\boldsymbol{x}_{n+1}$ is achieved by optimizing the acquisition function on the manifold $\mathcal{M}$.

4.1 Geometry-aware Kernels

The choice of the kernel function is crucial for the GP as it encodes our prior about the function f. As our parameters $\boldsymbol{x}$ belong to a Riemannian manifold, it is relevant to include this *a priori* knowledge in the choice of the kernel. A straightforward approach to adapt distance-based kernels to Riemannian manifolds is to replace the Euclidean distance d by the geodesic distance $d_{\mathcal{M}}$ in the definition of the kernel. Thus, the geodesic generalization of the SE kernel is given by (see [28])

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \exp(-\beta d_{\mathcal{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2). \tag{3}$$

Although it has been successfully used in some applications, Feragen et al. [29] showed that such a kernel is valid, i.e. positive definite (PD) for all the parameters values, only if the manifold is isometric to an Euclidean space. This implies that the geodesic SE kernel is not valid for curved manifolds such as $\mathcal{S}^d$ and $\mathcal{S}^D_{++}$. However, the same authors recently conjectured that there exists



$\mathcal{M}$	$\mathcal{S}^2$	$\mathcal{S}^3$	$\mathcal{S}^4$	$\mathcal{S}^2_{++}$	$\mathcal{S}^3_{++}$
$\beta_{\min}$	6.5	2.	1.2	0.6	0.2

Figure 2: Experimental selection of $\beta_{\min}$ in $\mathcal{S}^3$ (*left* plot) and $\mathcal{S}^3_{++}$ (*middle* plot). The percentage of PD geodesic SE kernel matrices $\boldsymbol{K}$ computed from 10 different sets of 500 samples on the manifold is depicted in blue (right axis). The corresponding distribution of minimum eigenvalue $\lambda_{\min}$ of $\boldsymbol{K}$ is depicted in purple (left axis). *Right* table: selected values of $\beta_{\min}$ for different Riemannian manifolds used in this paper.

intervals of the lengthscale parameter $\beta > \beta_{\min}$ resulting in PD kernels [30]. In this work, we follow this approach and determine experimentally the intervals of lengthscales β for which kernel matrices are PD for the manifolds of interest.

In order to compute $\beta_{\min}$, we sample 500 points from 10 Gaussian distributions on the manifold with random mean and covariance matrix $\boldsymbol{I}$. We then compute the corresponding kernel matrix $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for a range of β values with $\theta = 1$. We repeat this process 10 times for each value of β and compute the percentage of PD geodesic kernel matrices $\boldsymbol{K}$. As the minimum eigenvalue function is continuous and $\boldsymbol{K}_{\beta\to\infty} \to \boldsymbol{I}$, we fix $\beta_{\min}$ equal to the minimum value of β for which 100% of the matrices $\boldsymbol{K}$ are PD. Figure 2 (*left*) shows the percentage of PD geodesic kernel matrices and the distribution of their minimum eigenvalue $\lambda_{\min}$ as a function of β for $\mathcal{S}^3$ and $\mathcal{S}^3_{++}$. The values of $\beta_{\min}$ for the manifolds considered in this paper are provided in Fig. 2 (*right*).

Other types of kernels are available for specific manifolds and may also be used in BO [22]. For example, the geodesic Laplacian kernel is valid on spheres and hyperbolic spaces [29]. Moreover, kernels have been specifically designed for several manifolds (see e.g. [31] for the Grassmannian).

4.2 Optimization of Acquisition Functions

After refining the geometry-aware GP that models the unknown function f, the next query point $\boldsymbol{x}_{n+1}$ is selected by maximizing the acquisition function γ_n. In order to take into account the geometry of the domain $\mathcal{X} \subseteq \mathcal{M}$, we propose to optimize γ_n using optimization techniques on manifolds. Note that the acquisition functions are not altered but their search space is modified. In this context, optimization algorithms on Riemannian manifolds constitute a powerful alternative to constrained optimization. These geometry-aware algorithms reformulate constrained problems as an unconstrained optimization on manifolds and consider the intrinsic structure of the space of interest. Also, they tend to show lower computational complexity and better numerical properties [32].

In this paper, we use the conjugate gradient (CG) algorithm on Riemannian manifolds [33], described in Algorithm 1, to maximize the acquisition function γ_n (or minimize $\phi_n = -\gamma_n$), at each iteration n of GaBO. The recursive process of the method involves the same steps as the Euclidean CG, namely: (*i*) a line minimization along the search direction (step 3); (*ii*) the iterate update along the search direction (step 4); and (*iii*) the computation of the next search direction combining the gradient of the function at the new iterate and the previous search direction (steps 5-6). The differences with the Euclidean version are: (*1*) as the gradient $\nabla\phi(\boldsymbol{z}_k)$, and thus the search direction $\boldsymbol{\eta}_k$, belong to the tangent space of $\boldsymbol{z}_k$, the exponential map is needed to update the iterate along the search direction; (*2*) the step size is fixed by solving $\operatorname{argmin}_{\alpha_k} \phi_n\big(\operatorname{Exp}_{\boldsymbol{z}_k}(\alpha_k\boldsymbol{\eta}_k)\big)$ with a linesearch on the manifold (an example of linesearch algorithm on manifolds is provided in Appendix C); (*3*) the previous search direction $\boldsymbol{\eta}_k$ has to be parallel transported to the tangent space of $\boldsymbol{z}_{k+1}$ to be combined with the gradient of the new iterate $-\nabla\phi_n(\boldsymbol{z}_{k+1}) \in \mathcal{T}_{\boldsymbol{z}_{k+1}}\mathcal{M}$. Note that we presented the CG on manifold with the Hastenes-Stiefel update parameter β_k^{HS}, but other update techniques can also be extended to Riemannian manifolds [25].

Some problems may require to bound the search domain to a subspace, for example, to cope with physical limits or safety constraints in robotic systems when optimizing end-effector orientations or impedance parameters. In such cases, and particularly when the manifold is not a closed space, it is imperative to limit the domain of GaBO by defining boundary conditions inside the manifold. Therefore, the acquisition function is maximized over the domain $\mathcal{X} \subset \mathcal{M}$.

While most of the literature on manifold optimization focuses on problems where the only constraint is that the solution belongs to the manifold, only few works proposed to extend constrained optimization algorithms on Riemannian manifolds. In this context, we propose to extend the bound-constrained CG method [34] to Riemannian manifolds to cope with boundary conditions in the optimization. To do so, the steps 4-6 of Algorithm 1 are updated as described in Algorithm 2. At each iteration, if the updated iterate $\boldsymbol{z}_{k+1} \notin \mathcal{X}$, it is projected back onto the feasible domain and the search direction is reinitialized. In $\mathcal{S}^d$, we bound the design space by setting limits on the components of $\boldsymbol{x} \in \mathcal{S}^d$. If a component is out of the limits, we fix it as equal to the closest limit and reformatted the remaining components so that $\boldsymbol{x}$ still belongs to the manifold (see Appendix D for an example). For $\mathcal{S}^D_{++}$, we define limits on the eigenvalues λ of the SPD matrix. If an iterate $\boldsymbol{z}_{k+1} \notin \mathcal{X}$, we project it back to the domain by reducing/increasing the maximum/minimum eigenvalue of the iterate.

Notice that some problems may require to optimize several parameters belonging to different manifolds. In this case, the domain of GaBO is a product of manifolds. Consequently, the kernel function corresponds to the product of the kernels on the different manifolds and the optimization of the acquisition function operates directly on the product of manifolds.

Algorithm 1: Optimization of acquisition function with CG on Riemannian manifolds

Input: Acquisition function γ_n, initial iterate $\boldsymbol{z}_0 \in \mathcal{M}$
Output: Next parameter point $\boldsymbol{x}_{n+1}$

1 Set $\phi_n = -\gamma_n$ as the function to minimize and $\boldsymbol{\eta}_0 = -\nabla\phi_n(\boldsymbol{z}_0)$ as search direction ;
2 **for** $k = 0, 1 \ldots, K$ **do**
3 Compute the step size α_k using linesearch on Riemannian manifold ;
4 Set $\boldsymbol{z}_{k+1} = \mathrm{Exp}_{\boldsymbol{z}_k}(\alpha_k \boldsymbol{\eta}_k)$;
5 Compute $\beta_k^{\mathrm{HS}} = \frac{\langle \nabla\phi_n(\boldsymbol{z}_{k+1}), \nabla\phi_n(\boldsymbol{z}_{k+1}) - \nabla\phi_n(\boldsymbol{z}_k) \rangle_{\boldsymbol{z}_k}}{\langle \boldsymbol{\eta}_k, \nabla\phi_n(\boldsymbol{z}_{k+1}) - \nabla\phi_n(\boldsymbol{z}_k) \rangle_{\boldsymbol{z}_k}}$;
6 Set $\boldsymbol{\eta}_{k+1} = -\nabla\phi_n(\boldsymbol{z}_{k+1}) + \beta_k^{\mathrm{HS}}\, \Gamma_{\boldsymbol{z}_k \to \boldsymbol{z}_{k+1}}(\boldsymbol{\eta}_k)$;
7 **if** *the convergence criterion is reached* **then**
8 break
9 **end**
10 **end**
11 Set $\boldsymbol{x}_{n+1} = \boldsymbol{z}_{k+1}$

Algorithm 2: Update of steps 4-6 of Algorithm 1 for a domain $\mathcal{X} \subset \mathcal{M}$

4 Set $\boldsymbol{z}_{k+1} = \mathrm{Exp}_{\boldsymbol{z}_k}(\alpha_k \boldsymbol{\eta}_k)$;
if $\boldsymbol{z}_{k+1} \in \mathcal{X}$ **then**
  Execute steps 5-6 of Algorithm 1
else
  Project $\boldsymbol{z}_{k+1}$ to $\mathcal{X}$;
  Set $\boldsymbol{\eta}_{k+1} = -\nabla\phi(\boldsymbol{z}_{k+1})$;
end

5 Experiments

We test GaBO using some benchmark test functions and two simulated experiments with a 7-DOF Franka Emika Panda robot. All the implementations were built on the Python libraries GPflow [35], GPflowOpt [36] and Pymanopt [37]. The simulated experiments were performed using Pyrobolearn [38]. All the BO implementations use EI as acquisition function and were initialized with 5 random samples. Source codes and a video of the experiments are available at https://sites.google.com/view/geometry-aware-bo.

5.1 Benchmark functions

We use a couple of benchmark test functions to study the performance of GaBO in the Riemannian manifolds $\mathcal{S}^d$ and $\mathcal{S}^D_{++}$. To do so, we first project the test functions to these manifolds and then carry out the optimization by running 100 trials with random initialization. The selection of the kernel parameters is carried out by MLE. The search domain $\mathcal{X}$ of the test functions corresponds to the complete manifold for $\mathcal{S}^d$ and to SPD matrices with eigenvalues $\lambda \in [0.001, 5]$ for $\mathcal{S}^D_{++}$.

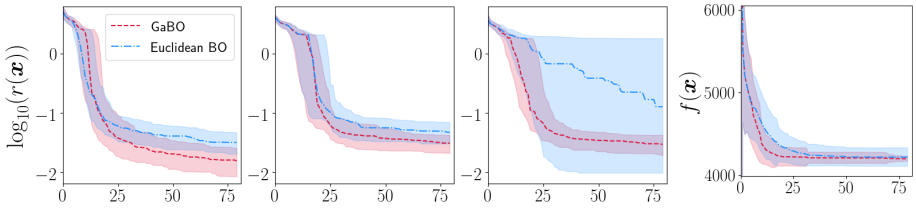


Figure 3: The three first plots show the logarithm of the simple regret in function of the BO iterations for the Ackley function in $\mathcal{S}^d$ with $d = \{2, 3, 4\}$ over 100 trials. The first and third quartiles are represented with a light tube around the median line. The last graph displays the total cost for end-effector orientation learning in $\mathcal{S}^3$ over 30 trials. The mean and one standard deviation are represented.

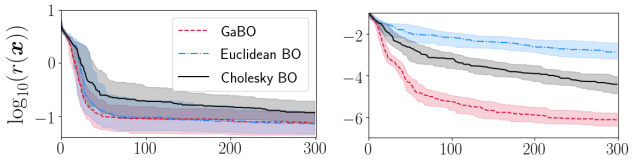


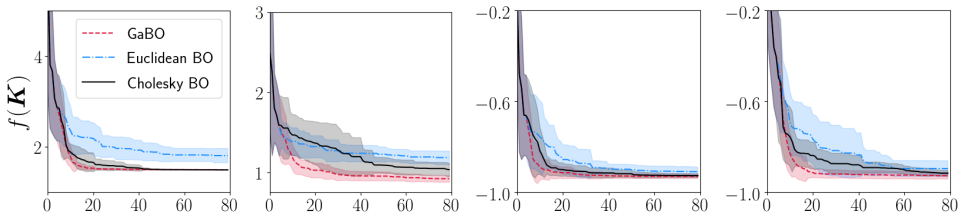
Figure 4: Logarithm of the simple regret for benchmark test functions in the Riemannian manifold of SPD matrices $\mathcal{S}^D_{++}$ over 100 trials. *Left*: Ackley function in $\mathcal{S}^2_{++}$. *Right*: bimodal distribution in $\mathcal{S}^2_{++}$. The first and third quartiles are represented with a light tube around the median line for the three BO methods.

In the case of $\mathcal{S}^d$, we compare GaBO against the classical BO which carries out all the operations in the Euclidean space (hereinafter called Euclidean BO). As the query points must belong to $\mathcal{S}^d$, i.e. $\|\boldsymbol{x}\| = 1$, the maximization of the acquisition is considered a constrained problem in Euclidean BO. We minimize the Ackley function for parameter spaces of dimensionality $d = \{2, 3, 4\}$ to analyze the methods performance as the parameter dimension increases. An example of the evolution of the surrogate model of GaBO for the Ackley function in $\mathcal{S}^2$ is shown in Appendix E. Figure 3 displays the median and the first and third quartiles of the logarithm of the simple regret along 80 BO iterations for the three aforementioned hypersphere manifolds. We observe that, although the performance of the two algorithms are comparable for low-dimensional hyperspheres, GaBO outperforms the Euclidean BO when the dimension increases: GaBO converges faster to a better optimizer with lower variance than Euclidean BO. This is particularly evident for the Ackley function on $\mathcal{S}^4$, where the median regret of Euclidean BO remains far from the minimum value and displays high variance, while GaBO converges to a value close to the optimum for all the trials after 70 iterations. The fact that GaBO can be slightly slower than the Euclidean BO to converge in $\mathcal{S}^2$ may be due to the relatively high value of $\beta_{\min}$ in the kernel (3) for this manifold. High $\beta_{\min}$ limits the spatial influence of each observation on the modeling of the function f. Thus, more observations are needed to model slowly-evolving regions of the function (see Appendix B). A solution to this is to bound the domain of the optimization to a subspace of the manifold, which is often necessary in most real applications, and to determine the value $\beta_{\min}$ of the kernel for this subspace, resulting in a lower $\beta_{\min}$.

Regarding the manifold $\mathcal{S}^D_{++}$, we compare our method against the Euclidean BO (augmented with the constraint $\lambda_{\min} > 0$) and an alternative implementation that takes advantage of the Cholesky decomposition of an SPD matrix $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\mathsf{T}$, so that the resulting parameter is the vectorization of the lower triangular matrix $\boldsymbol{L}$ (hereinafter called Cholesky BO). In this case, we test two functions, namely a bimodal Gaussian distribution and the Ackley function. Figure 4 shows the regret for 300 iterations of GaBO, Cholesky BO and Euclidean BO for the two test functions in $\mathcal{S}^2_{++}$. We observe that GaBO outperforms Cholesky and Euclidean BO in both cases. The computation time of GaBO is slightly higher than the Euclidean equivalent (see Appendix F). However, this is not a major drawback as physical robotic experiments to collect the next value of the function f will take significantly longer than the optimization process.

5.2 Simulated robotic experiments

We here evaluate GaBO performance when looking first for the optimal orientation for a simple regulation task, and second for the optimal stiffness matrix of an impedance control policy, which is of interest for variable impedance learning approaches. For both experiments, we use a simulated 7-DOF Franka Emika Panda robot. Costs parameters values are reported in Appendix G.
In the first experiment, we use BO as an orientation sampler aiming at satisfying the requirements defined by a cost function. This may be useful for tasks where the orientation reference of a con-



(a) $f_1(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^2_{++}$ (b) $f_1(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^3_{++}$ (c) $f_2(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^2_{++}$ (d) $f_2(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^3_{++}$

Figure 5: Total cost for stiffness learning of a Cartesian impedance controller using direct policy search over 15 trials. The mean and one standard deviation are represented.

troller needs to be refined to improve the task execution. In this experiment, the velocity-controlled robot samples an orientation reference $\boldsymbol{x} \equiv \hat{\boldsymbol{q}}_o$ around a prior orientation $\tilde{\boldsymbol{q}}_o$, fixed by the user, with the aim of minimizing the cost function $f(\boldsymbol{q}) = w_q \|\Delta(\tilde{\boldsymbol{q}}, \boldsymbol{q})\|^2 + w_{\boldsymbol{\tau}} \|\boldsymbol{\tau}\|^2 + w_m \text{cond}(\boldsymbol{M})$, where $\boldsymbol{q}_o$ is the current end-effector orientation, and $\text{cond}(\boldsymbol{M})$ is the condition number of the linear velocity manipulability ellipsoid. This cost function aims at minimizing the error between the prior and the current end-effector orientation with low joint torques and an isotropic manipulability ellipsoid. We run 30 trials with random initialization for each cost function. The last graph of Figure 3 shows the total cost for 80 iterations of GaBO and Euclidean BO. We observe that GaBO converges faster to a better optimizer with a close-to-zero variance over the trials. Note that the difference between the two methods is accentuated compared to the benchmark function in $\mathcal{S}^3$.

Since direct policy search has been a successful RL approach in robotics, for our second experiment, we seek to find the optimal Cartesian stiffness of a torque-controlled 7-DOF robotic arm implementing a Cartesian control policy $\boldsymbol{f} = \boldsymbol{K}^{\mathcal{P}}(\hat{\boldsymbol{p}} - \boldsymbol{p}) - \boldsymbol{K}^{\mathcal{D}}\dot{\boldsymbol{p}}$, where $\boldsymbol{p}$ and $\dot{\boldsymbol{p}}$ are the linear position and velocity of the robot end-effector, $\boldsymbol{K}^{\mathcal{P}}$ and $\boldsymbol{K}^{\mathcal{D}}$ are stiffness and damping matrices, and $\boldsymbol{f}$ is the control force (transformed to desired torques via $\boldsymbol{\tau} = \boldsymbol{J}^{\mathsf{T}}\boldsymbol{f}$). The robot task consists of tracking a desired Cartesian position $\hat{\boldsymbol{p}}$ while a constant external force $\boldsymbol{f}^e$ is applied to its end-effector. The policy parameter corresponds to the stiffness matrix, that is $\boldsymbol{x} \equiv \boldsymbol{K}^{\mathcal{P}}$. The stiffness-damping ratio is fixed as critically damped. We tested GaBO, Euclidean BO and Cholesky BO using two different cost functions: $f_1(\boldsymbol{K}^{\mathcal{P}}) = w_{\boldsymbol{p}} \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|^2 + w_d \det(\boldsymbol{K}^{\mathcal{P}}) + w_c \text{cond}(\boldsymbol{K}^{\mathcal{P}})$ and $f_2(\boldsymbol{K}^{\mathcal{P}}) = w_{\boldsymbol{p}} \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|^2 + w_{\boldsymbol{\tau}} \|\boldsymbol{\tau}\|^2$, for $\boldsymbol{K}^{\mathcal{P}} \in \{\mathcal{S}^2_{++}, \mathcal{S}^3_{++}\}$. The cost function f_1 aims at accurately tracking the desired position using a low-volume isotropic stiffness matrix, while f_2 aims at tracking the desired position accurately with low torques. For f_2, a -1 reward was added if the desired position was reached. In the case of $\mathcal{S}^2_{++}$, only a 2×2 submatrix of the full stiffness $\boldsymbol{K}^{\mathcal{P}}$ is optimized with BO, while the other components stay constant. Instead, all the components of the 3×3 matrix $\boldsymbol{K}^{\mathcal{P}}$ are optimized in $\mathcal{S}^3_{++}$. We run 15 randomly initialized trials for each cost function.

Figure 5 shows the total cost of the stiffness learning for the two cost functions in $\mathcal{S}^2_{++}$ and $\mathcal{S}^3_{++}$. We observe that Cholesky BO tends to outperform Euclidean BO as the complexity of the cost is increased, and that GaBO outperforms the other methods for all the test cases. Moreover, while the performance of Euclidean and Cholesky BO strongly degrades as the dimensionality increases, GaBO still provides accurate and low-variance solutions.

6 Conclusion

We proposed GaBO, a geometry-aware Bayesian optimization framework that exploited the geometry of the search space to properly seek optimal parameters that lie on Riemannian manifolds. To do so, we used geometry-aware kernels that allow GP to properly measure the similarity between parameters lying on a Riemannian manifold. Moreover, we exploited Riemannian manifold tools to consider the geometry of the search space when optimizing the acquisition function. GaBO provided faster convergence, better accuracy and lower solution variance when compared to geometry-unaware BO implementations. These differences were accentuated as the manifold dimensionality increased. In this line, we will evaluate GaBO in higher-dimensional manifolds. We also plan to investigate a more general projection approach to handle bound-constrained gradient methods on Riemannian manifolds. Finally, our proof-of-concept experiments open the door towards optimizing RL policies for complex robot learning scenarios where geometry-awareness may be relevant.

Acknowledgments

This work was carried out during a PhD Sabbatical of N. Jaquier at the Bosch Center for Artificial Intelligence (BCAI). The authors thank Andras Kupcsik and Lukas Fröhlich for their useful feedback on this manuscript.

References

[1] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[2] A. Cully, J. Clune, D. Tarapore, and J. B. Mouret. Robots that can adapt like animals. *Nature*, 521:503–507, 2015.

[3] P. Englert and M. Toussaint. Combined optimization and reinforcement learning for manipulations skills. In *Robotics: Science and Systems (R:SS)*, Michigan, USA, June 2016.

[4] A. Marco, P. Hennig, S. Schaal, and S. Trimpe. On the design of LQR kernels for efficient controller learning. In *IEEE Conference on Decision and Control (CDC)*, pages 5193–5200, Melbourne, Australia, December 2017.

[5] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1771–1778, Brisbane, Australia, May 2018.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Barath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34:26–38, 2017.

[7] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton. High dimensional Bayesian optimization using dropout. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2096–2102, Melbourne, Australia, August 2017.

[8] S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh. High dimensional Bayesian optimization with elastic Gaussian process. In *Intl. Conf. on Machine Learning (ICML)*, pages 2883–2891, Sydney, Australia, August 2017.

[9] J. Kirschner, M. Mutny, N. Hiller, R. Ischebeck, and A. Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In *Intl. Conf. on Machine Learning (ICML)*, pages 3429–3438, California, USA, June 2019.

[10] R. Antonova, A. Rai, and C. Atkeson. Deep kernels for optimizing locomotion controllers. In *Conference on Robot Learning (CoRL)*, pages 47–56, California, USA, November 2017.

[11] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon. Geometry-aware tracking of manipulability ellipsoids. In *Robotics: Science and Systems (R:SS)*, Pittsburgh, USA, June 2018.

[12] N. D. Ratliff, J. Issac, and D. Kappler. Riemannian motion policies. *arXiv preprint 1801.02854*, 2018. URL `https://arxiv.org/abs/1801.02854`.

[13] M. J. A. Zeestraten. *Programming by Demonstration on Riemannian Manifolds*. PhD thesis, Universitá degli Studi di Genova, 2018.

[14] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. Automatic LQR tuning based on Gaussian process global optimization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 270–277, Stockholm, Sweden, May 2016.

[15] K. Yuan, I. Chatzinikolaidis, and Z. Li. Bayesian optimization for whole-body control of high degrees of freedom robots through reduction of dimensionality. *IEEE Robotics and Automation Letters*, 4(3):2268–2275, 2019.

[16] A. Ghadirzadeh, J. Bütepage, A. Maki, D. Kragic, and M. Björkman. A sensorimotor reinforcement learning framework for physical human-robot interaction. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2682–2688, Daejeon, Korea, October 2016.

[17] A. Kupcsik, D. Hsu, and S. Lee. Learning dynamic robot-to-human object handover from human feedback. In *Intl. Symp. on Robotics Research (ISRR)*, Genoa, Italy, September 2015.

[18] D. Drie, P. Englert, and M. Toussaint. Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations. In *IEEE Intl. Conf. on Robotics and*

Automation (ICRA), pages 902–907, Singapore, May 2017.

[19] R. Pautrat, K. Chatzilygeroudis, and J.-B. Mouret. Bayesian optimization with automatic prior selection for data-efficient direct policy search. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 7571–7578, Brisbane, Australia, May 2018.

[20] A. Wilson, A. Fern, and P. Tadepalli. Using trajectory data to improve Bayesian optimization for reinforcement learning. *Journal of Machine Learning Research*, 15:253–282, 2014.

[21] R. Martinez-Cantin. Bayesian optimization with adaptive kernels for robot control. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3350–3356, Singapore, May 2017.

[22] C. Oh, E. Gavves, and M. Welling. BOCK: Bayesian optimization with cylindrical kernels. In *Intl. Conf. on Machine Learning (ICML)*, pages 3868–3877, Stockholm, Sweden, July 2018.

[23] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[24] J. Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404, 1975.

[25] P. A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007.

[26] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *Intl. Journal on Computer Vision*, 66(1):41–66, 2006.

[27] S. Sra and R. Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM Journal on Optimization*, 25(1):713–739, 2015.

[28] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi. Kernel methods on Riemannian manifolds with Gaussian RBF kernels. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 37(12):2464–2477, 2015.

[29] A. Feragen, F. Lauze, and S. Hauberg. Geodesic exponential kernels: When curvature and linearity conflict. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, June 2015.

[30] A. Feragen and S. Hauberg. Open problem: Kernel methods on manifolds and metric spaces. what is the probability of a positive definite geodesic exponential kernel? In *29th Annual Conference on Learning Theory*, pages 1647–1650, New York, USA, June 2016.

[31] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073, 2012.

[32] J. Hu, X. Liu, Z. Wen, and Y. Yuan. A brief introduction to manifold optimization. *arXiv preprint 1906.05450*, 2019. URL https://arxiv.org/abs/1906.05450.

[33] S. T. Smith. Optimization techniques on Riemannian manifold. *Fields Institute Communications*, 3:113–136, 1994.

[34] E. A. Vollebregt. The bound-constrained conjugate gradient method for non-negative matrices. *Journal of Optimization Theory and Applications*, 162(3):931–953, 2014.

[35] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

[36] N. Knudde, J. van der Herten, T. Dhaene, and I. Couckuyt. GPflowOpt: A Bayesian Optimization Library using TensorFlow. *arXiv preprint – arXiv:1711.03845*, 2017. URL https://arxiv.org/abs/1711.03845.

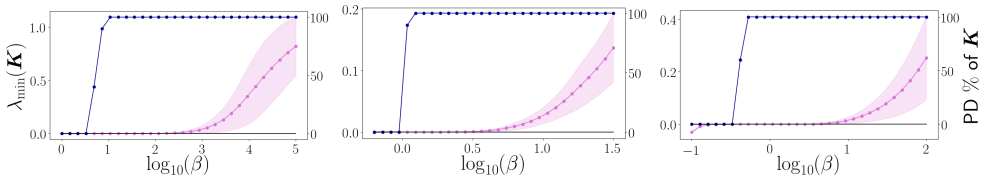
[37] J. Townsend, N. Koep, and S. Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137): 1–5, 2016.

[38] B. Delhaisse, L. Rozo, and D. G. Caldwell. Pyrobolearn: A Python framework for robot learning practitioners. In *Conference on Robot Learning (CoRL)*, Osaka, Japan, October 2019.

Appendices

A Geometry-aware kernel parameters

Figure 6 shows the results of the experimental selection process of $\beta_{\min}$ for the geodesic SE kernel, as introduced in Section 4.1, for the manifolds $\mathcal{S}^2$, $\mathcal{S}^4$ and $\mathcal{S}^2_{++}$.



(a) $\mathcal{S}^2$ (b) $\mathcal{S}^4$ (c) $\mathcal{S}^2_{++}$

Figure 6: Experimental selection of $\beta_{\min}$. The percentage of PD geodesic SE kernel matrices $\boldsymbol{K}$ computed from 10 different sets of 500 samples on the manifold is depicted in blue (right axis). The corresponding distribution of the minimum eigenvalue $\lambda_{\min}$ of $\boldsymbol{K}$ is depicted in purple (left axis).

B Influence of $\beta_{\min}$ on GaBO

In Section 5.1, we hypothesize that the fact that GaBO can be slightly slower than the Euclidean BO to converge in $\mathcal{S}^2$ may be due to the relatively high value of $\beta_{\min}$ in the kernel (3) for this manifold. In this appendix, we illustrate and analyze the effect of $\beta_{\min}$ on GaBO.

The value of β determines the spatial influence of the observations in the Gaussian process modeling the function f. In other words, the value of β controls the maximum size of the extrapolation region around one observation. Small β values permit to extrapolate in a large zone around one observation and therefore are well suited to model functions that evolve slowly with only few observations. For this type of functions, where the optimum β tends to be close to β_{min}, increasing β_{min} may result in a slight increase of the number of observations needed to properly model the function. Therefore, GaBO may need a few more query points to converge.

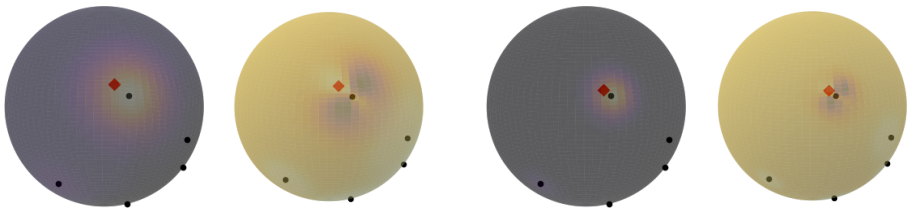
Figure 7 shows an example of the influence of the β value on the choice of the next query point. Two different cases with $\beta = 6.5$ and $\beta = 25$ are considered. The left graphs displays the mean of a GP based on 5 observation on the sphere with colors ranging from yellow (low values) to dark purple (high values). The corresponding acquisition function is displays on the right graphs with the same colors. The next query point, corresponding to the maximum of the acquisition function is depicted by a red square. We observe that the zone of influence of the observations is reduced for $\beta = 25$ compared to $\beta = 6.5$. This modifies the acquisition function, whose maximum is slightly closer to the observation for a higher value of β. Therefore, the number of query points needed to reach the optimum of the function may slightly increase if the value $\beta_{\min}$ is increased. Note that the difference between the query points obtained with $\beta = 6.5$ and $\beta = 25$ remains small for a consequent difference between the two tested β values. Therefore, a slight increase of $\beta_{\min}$ will have a limited impact on the number of iterations needed by GaBO to converge.

In the case of rapidly-varying functions that are therefore better modeled by β values significantly higher than β_{min}, a slight change on β_{min} will not impact GaBO.

C Adaptive line search on Riemaniann manifolds

Algorithm 3 presents the adaptive line search process on Riemannian manifolds [25]. The goal of this algorithm is to compute the stepsize α_k used in the conjugate gradient descent described in Section 4.2. The line search aims at finding the step size that maximizes the acquisition function, i.e. it minimizes the function $\phi_n = -\gamma_n$, along the search direction $\boldsymbol{\eta}_k$. As the search direction lies on the tangent space of the iterate $\boldsymbol{z}_k$, this corresponds to solving

$$\underset{\alpha_k}{\operatorname{argmin}}\, \phi_n\big(\operatorname{Exp}_{\boldsymbol{z}_k}(\alpha_k \boldsymbol{\eta}_k)\big). \tag{4}$$



(a) $\beta = 6.5$ (b) $\beta = 25$

Figure 7: Illustration of the effect of β on GaBO. The GP mean and corresponding acquisition function are depicted by colors ranging from yellow (low values) to dark purple (high values) on the left and right graphs, respectively. The observations and next query point are depicted by black dots and a red square.

Algorithm 3: Adaptive linesearch for CG on Riemannian manifolds

Input: Function $\phi_n = -\gamma_n$, iterate $\boldsymbol{z}_k$, search direction $\boldsymbol{\eta}_k$, initial stepsize α_k, contraction factor c
Output: Final stepsize α_k
for $j = 0, 1 \ldots, J$ **do**
  Set $\boldsymbol{z} = \mathrm{Exp}_{\boldsymbol{z}_k}(\alpha_k \eta_k)$;
  if $\phi_n(\boldsymbol{z}) > \phi_n(\boldsymbol{z}_k) + 0.5\alpha_k \langle \nabla\phi_n(\boldsymbol{z}_k), \boldsymbol{\eta}_k \rangle_{\boldsymbol{z}_k}$ **then**
    break
  end
  $\alpha_k = c\alpha_k$;
end
if $\phi_n(\boldsymbol{z}) > \phi_n(\boldsymbol{z}_k)$ **then**
  $\alpha_k = 0$
end

D Example of boundary conditions handling on $\mathcal{S}^1$

Figure 8 shows an example of the application of Algorithm 2 on the unit circle $\mathcal{S}^1$. In this example, an acquisition function is optimized over the domain $\mathcal{S}^1$ with the constraint $y \leq 0.6$. Assume that the point initially proposed by the algorithm in step 4 is $\boldsymbol{z} = (x, y)^\intercal = (0.49, 0.87)^\intercal$, depicted by a red dot. This point does not satisfy the constraint $y \leq 0.6$, and needs to be projected back onto the feasible domain. As explained in Section 4.2, we first fix the value of the y component to the closest limit and obtain the point $\tilde{\boldsymbol{z}} = (0.49, 0.6)^\intercal$, depicted by a yellow dot. Note that $\tilde{\boldsymbol{z}} \notin \mathcal{S}^1$ as its norm is not equal to 1, thus we need to project this point on the manifold. To guarantee that the bound constraints remain satisfied, we reformat only the components that were not affected by the constraints. In this example, y is fixed at 0.6 and only the component x varies to obtain a point on the manifold. Therefore, we obtain the final point $\boldsymbol{z}_{k+1} = (0.8, 0.6)^\intercal \in \mathcal{X}$, depicted by a blue dot.

E Evolution of the surrogate model of GaBO for the Ackley function in $\mathcal{S}^2$

Figure 9 shows an example of the evolution of the surrogate model of GaBO for the Ackley function in $\mathcal{S}^2$. The left column displays the mean of the GP model on the sphere with colors ranging from yellow (low values) to dark purple (high values). The observed points $\boldsymbol{x}_n$ are depicted by black dots.

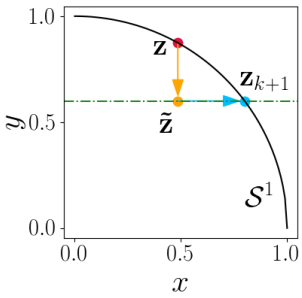


Figure 8: Application of Algorithm 2 on $\mathcal{S}^1$. The manifold $\mathcal{S}^1$ is depicted by a balck curve and the constraint $y \leq 0.6$ is shown by the green dotted line. The initial point $\boldsymbol{z}$, fixed point $\tilde{\boldsymbol{z}}$ and final point $\boldsymbol{z}_{k+1}$ are depicted by red, yellow and blue dots, respectively. Arrows show the order of the projections.

The global minimum $\boldsymbol{x}^* = (0, 0, 0)^\mathsf{T}$ is shown as a green star and the current best guess is depicted by a blue square. The middle and right columns show the GP model on the sphere projected on the dimensions x_1, x_2 and x_2, x_3, respectively. The mean value of the GP is represented on the vertical dimension with the same colors as the left column. The variance of the model, represented with two standard deviations, is depicted by a gray envelope around the mean.

We observe that the region around the optimum of the objective function is well modeled by GP model after 10 iterations. The current best guess is also close to the optimum. After 30 iterations, the GP results in a good approximation of the objective function with a low variance in the region around the optimum value.

F BO computation times

Table 2 provides the computation time per iteration of GaBO, Euclidean BO and Cholesky BO for the optimization of the Ackley function on the different manifolds used in the paper. Overall, the computation time of GaBO is slightly higher than the Euclidean equivalent. The increase observed for $\mathcal{S}^2_{++}$ when compared to $\mathcal{S}^d$ is due to the fact that the exponential and logarithmic maps are more complex in $\mathcal{S}^D_{++}$, involving matrix exponential and logarithm computations, than their equivalent on the hyperspheres. We do not consider the increase of the computation time as a major drawback as physical robotic experiments will take significantly longer than this time to collect the next value of the function f.

Table 2: Computation time for one iteration of the different BO frameworks when optimizing the Ackley function on several manifolds. All the time values are given in seconds [s].

	$\mathcal{S}^2$	$\mathcal{S}^3$	$\mathcal{S}^4$	$\mathcal{S}^2_{++}$
GaBO	1.55 ± 0.32	1.49 ± 0.42	1.94 ± 0.42	6.08 ± 5.36
Euclidean BO	0.48 ± 0.14	0.49 ± 0.17	0.53 ± 0.14	0.24 ± 0.17
Cholesky BO	-	-	-	0.54 ± 0.12

G Parameters of simulated robotic experiments

Tables 3, 4 and 5 provide the values of the parameters for the simulated robotic experiments presented in Section 5.2. The Franka-Emika robot was initialized from a joint position $(0., 0.3, 0., -1., 0., 1.5, 0.)^\mathsf{T}$ for all the experiments.

Table 3: Cost parameter values for the orientation sampling experiment.

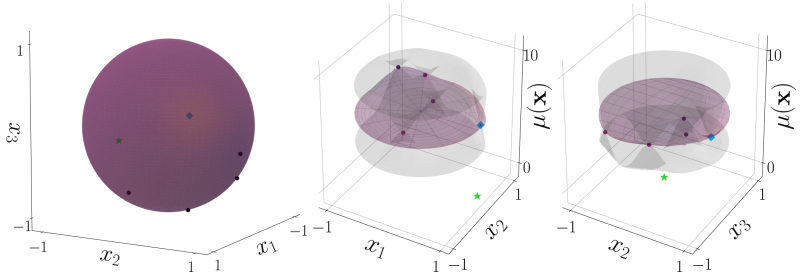
$\tilde{\boldsymbol{q}}$	w_q	$w_{\boldsymbol{\tau}}$	w_m
$(0.408, 0.408, 0, 0.816)^\mathsf{T}$	1	10^{-4}	0.1

Table 4: Cost parameter values of f_1 in the optimal stiffness learning experiment.

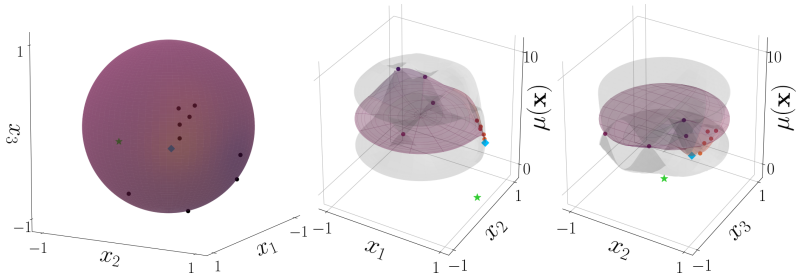
$\hat{\boldsymbol{p}}$ [m]	$\dot{\boldsymbol{p}}$	$w_{\boldsymbol{p}}$	w_d	w_c	$\boldsymbol{f}^e$ [N]
$(0.66, -0.01, 0.69)^\mathsf{T}$	$\mathbf{0}$	1	10^{-13}	10^{-4}	$(0, 20, -20)^\mathsf{T}$

Table 5: Cost parameter values of f_2 in the optimal stiffness learning experiment.

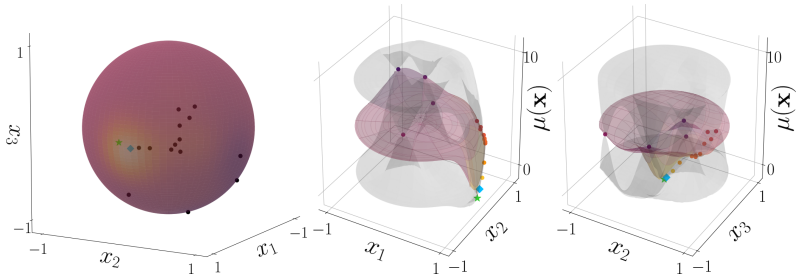
$\hat{\boldsymbol{p}}$ [m]	$\dot{\boldsymbol{p}}$	$w_{\boldsymbol{p}}$	$w_{\boldsymbol{\tau}}$	$\boldsymbol{f}^e$ [N]
$(0.5, -0.4, 0.75)^\mathsf{T}$	$\mathbf{0}$	1	10^{-5}	$(0, 20, -20)^\mathsf{T}$



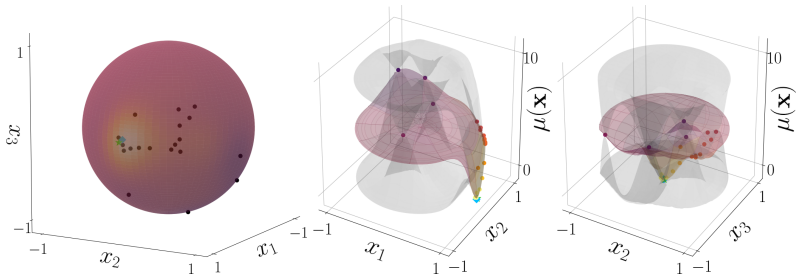
(a) Initial model



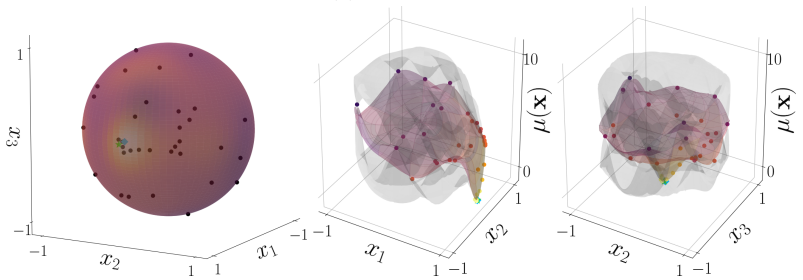
(b) Iteration 5



(c) Iteration 10



(d) Iteration 15



(e) Iteration 30

Figure 9: Surrogate GP of GaBO for the minimization of the Ackley function in $\mathcal{S}^2$. The GP mean is depicted by colors ranging from yellow (low values) to dark purple (high values) and $\pm$ two standard deviations in gray (projected graphs). The global minimum and the current best guess are shown as a green star and a blue square.