Sampling Over Riemannian Manifolds Using Kernel Herding

Sandesh Adhikary[1] and Byron Boots[1]

Abstract— **Kernel herding is a deterministic sampling algorithm designed to draw 'super samples' from probability distributions when provided with their kernel mean embeddings in a reproducing kernel Hilbert space (RKHS). Empirical expectations of functions in the RKHS formed using these super samples tend to converge even faster than random sampling from the true distribution itself. Standard implementations of kernel herding have been restricted to sampling over flat Euclidean spaces, which is not ideal for applications such as robotics where more general Riemannian manifolds may be appropriate. We propose to adapt kernel herding to Riemannian manifolds by (1) using geometry-aware kernels that incorporate the appropriate distance metric for the manifold and (2) using Riemannian optimization to constrain herded samples to lie on the manifold. We evaluate our approach on problems involving various manifolds commonly used in robotics including the SO(3) manifold of rotation matrices, the spherical manifold used to encode unit quaternions, and the manifold of symmetric positive definite matrices. We demonstrate that our approach outperforms existing alternatives on the task of resampling from empirical distributions of weighted particles, a problem encountered in applications such as particle filtering. We also demonstrate how Riemannian kernel herding can be used as part of the kernel recursive approximate Bayesian computation algorithm to estimate parameters of black-box simulators, including inertia matrices of an Adroit robot hand simulator. Our results confirm that exploiting geometric information through our approach to kernel herding yields better results than alternatives including standard kernel herding with heuristic projections.**

I. Introduction

Kernel mean embeddings are powerful statistical tools that enable learning and inference in potentially infinite-dimensional feature spaces. By representing probability distributions as elements of reproducing kernel Hilbert spaces, kernel mean embeddings allow us to apply various kernel-based algorithms directly on distributions [1], [2]. While there are many tools available to sample from probability density functions, they are not directly applicable to kernel mean embeddings. *Kernel herding* is a deterministic algorithm that allows us to draw samples from distributions when they are represented as kernel mean embeddings [3]. This is particularly useful for methods that operate in reproducing kernel Hilbert spaces (RKHS), including many algorithms designed for robotics problems such as filtering [4], [5], simulator parameter estimation [6], [7], and likelihood-free Bayesian inference [8], [6]. Adapting kernel herding to Riemannian manifolds also extends the applicability of these algorithms to problems defined over Riemannian manifolds.

Current applications of kernel herding have been restricted to sampling over flat Euclidean spaces [3], [4], [6], [9], and the standard kernel herding algorithm does not directly translate to general Riemannian manifolds encountered in many applications. For example, in robotics, quantities like stiffness and inertia are represented by symmetric positive definite (SPD) matrices, which form Riemannian manifolds when equipped with a Riemannian metric. The special orthogonal group SO(3) of rotation matrices is used to represent robot orientations and rotations. The spherical manifold of unit-vectors can also be used to encode orientations and rotations as unit quaternions. The spherical manifold is also foundational to directional statistics [10], [11] and has a wide range of applications including topic modeling in natural language processing [12], protein structure modeling [13], speaker clustering from audio data [14], etc. Exploiting the Riemannian structure of data spaces has proven to be useful in numerous applications in robotics including model predictive control [15], imitation learning [16], Bayesian optimization [17], and so on. Our aim is to similarly incorporate the Riemannian structure of data spaces for the task of sampling from empirical distributions and kernel mean embeddings.

We propose to adapt kernel herding to Riemannian manifolds by (1) using geometry-aware kernels that incorporate the appropriate distance metric for the manifold, and (2) using Riemannian optimization to ensure that we only generate feasible samples that lie on the manifold. Geometry-aware kernels have been successfully applied to many problems on Riemannian manifolds including Bayesian optimization [17], [18], spectral image classification [19] and object recognition [20]. Our approach is similar in spirit to that of [17], which applies a combination of geometry-aware kernels and Riemannian optimization to Bayesian optimization. We focus on a different problem of sampling over Riemannian manifolds. While sampling from distributions over Riemannian manifolds has been tackled in various contexts, the problem of sampling from kernel mean embeddings over Riemannian manifolds is yet to be addressed. There is significant literature dedicated to extending Markov chain Monte Carlo (MCMC) methods to Riemannian manifolds [21], [22], [23], [24]. Utilizing the Riemannian kernelized Stein discrepancy [25], [26], the Riemannian Stein variational gradient descent algorithm provides a particle-based alternative that mitigates unwanted positive sample correlations observed in MCMC methods [25]. The same discrepancy measure may also allow a similar extension of other sampling algorithms such as Stein points [9]. However, all of these methods rely on having access to (potentially unnormalized) probability density functions. In contrast, we focus on a different access model where samples are to be drawn from distributions embedded in an RKHS or empirical distributions defined via sets of weighted particles.

We provide an overview of kernel herding and kernel mean embeddings in Section II. In Section III, we describe our approach to adapt it to domains where samples must lie on Riemannian manifolds. Finally, in Section IV, we present experimental results in resampling from a set of weighted particles, and estimating the

[1] University of Washington, Seattle WA 98105, USA. {adhikary, bboots}@cs.washington.edu

parameters of a robotic simulator. Our experiments cover various manifolds pertinent to robotics including SO(3), the spherical manifold, and the manifold of SPD matrices.

II. Kernel Herding

We begin with a summary of kernel herding, along with background on how probability distributions are embedded in an RKHS as kernel mean embeddings. We refer the reader to [1] and [2] for thorough surveys of these concepts.

Reproducing Kernel Hilbert Spaces Let $\mathcal{X}$ be the domain of our data, and ϕ a non-linear mapping to a higher-dimensional feature space. Most kernel-based learning algorithms access these feature vectors only through their inner products, which we can associate with a symmetric kernel function $k(\mathbf{x},\mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$. Instead of inferring the kernel from a feature map, it is generally more convenient to define the kernel directly. If a kernel is symmetric and positive definite (SPD) such that $\sum_{ij} c_i c_j k(\mathbf{x}_i,\mathbf{x}_j) \geq 0$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ and $c_i, c_j \in \mathbb{R}$, the function $k(\mathbf{x},\mathbf{x}')$ is the inner product of a feature map from $\mathcal{X}$ to $\mathcal{H}$, a potentially infinite-dimensional Hilbert space [27], [2]. The Hilbert space $\mathcal{H}$ associated with SPD kernels has some additional structure; it is actually a *reproducing kernel* Hilbert space (RKHS) [27] that satisfies the following reproducing property

$$\mathbf{f}(\mathbf{x}) = \langle k(\mathbf{x},\cdot), \mathbf{f} \rangle_{\mathcal{H}} \quad \forall\, \mathbf{f} \in \mathcal{H} \;\text{ and }\; \mathbf{x} \in \mathcal{X}$$

Kernel based methods tend to offer the advantage of high representational power that is decoupled from the dimensionality of $\mathcal{X}$. Instead, the complexity of these methods is tied to the number of data samples, the memory costs due to which may need to be offset by pruning the number of samples retained [28], [3].

Kernel Mean Embeddings Besides embedding individual samples from $\mathcal{X}$ to an RKHS, we can also embed entire distributions defined over $\mathcal{X}$. A probability distribution P is embedded in an RKHS as a *kernel mean embedding* $\boldsymbol{\mu}_P$, corresponding to the expected value of the kernel function $\mathbb{E}_{\mathbf{x}\sim P} k(\mathbf{x},\cdot)$. We define $\boldsymbol{\mu}_P$ and its empirical estimate $\hat{\boldsymbol{\mu}}_P$ as follows

$$\boldsymbol{\mu}_P = \int k(\mathbf{x},\cdot) dP(\mathbf{x}) \qquad \hat{\boldsymbol{\mu}}_P = \frac{1}{n} \sum_{i=1}^{n} k(\mathbf{x}_i,\cdot) \;\text{ for } \mathbf{x}_i \sim P$$

The empirical estimate $\hat{\boldsymbol{\mu}}_P$ converges to $\boldsymbol{\mu}_P$ at a rate of $O(n^{-1/2})$ [29]. We can also construct the empirical mean embedding as a *weighted* average $\hat{\boldsymbol{\mu}}_P = \sum_{i=1}^{n} w_i k(\mathbf{x}_i,\cdot)$ from points $\{\mathbf{x}_i\}$ not necessarily sampled from P. Note that the weights w_i can, in general, be negative; preventing us from using various off-the-shelf sampling techniques. Given a kernel mean embedding of a distribution, we can calculate expected values of functions $\mathbf{f} \in \mathcal{H}$ as follows

$$\mathbb{E}_{\mathbf{x}\sim P}(f(x)) = \langle \mathbf{f}, \boldsymbol{\mu}_P \rangle_{\mathcal{H}} \approx \langle \mathbf{f}, \hat{\boldsymbol{\mu}}_P \rangle$$

If the kernel is additionally *characteristic*, the mapping from P to $\boldsymbol{\mu}_P$ is injective, and $\boldsymbol{\mu}_P$ is the complete, unique description of P [30]. Popular examples of characteristic kernels over Euclidean space include the Gaussian and Laplacian kernels [30].

Kernel Herding Kernel herding is an approach for drawing samples from probability distributions via their kernel mean embeddings [3]. Given a kernel function k and a target kernel mean embedding $\hat{\boldsymbol{\mu}}_P$, or a set of weighted samples $\{w_i, \tilde{\mathbf{x}}_i\}$ representing $\hat{\boldsymbol{\mu}}_P$, kernel herding constructs a new set of samples $\{\mathbf{x}_t\}_{t=1}^{n}$ by solving the following optimization problem n times

$$\begin{aligned}
\mathbf{x}_t &= \operatorname*{argmin}_{\mathbf{x}} \mathcal{E}_k(\, \mathbf{x},\; \hat{\boldsymbol{\mu}}_P,\, \{\mathbf{x}_i\}_{i=1}^{t-1}\,) \\
&= \operatorname*{argmin}_{\mathbf{x}} -\langle k(\mathbf{x},\cdot), \hat{\boldsymbol{\mu}}_P \rangle + I(t>1) \sum_{j=1}^{t-1} \frac{1}{t} k(\mathbf{x},\mathbf{x}_j) \\
&= \operatorname*{argmin}_{\mathbf{x}} -\sum_{i=1}^{n} w_i k(\mathbf{x},\tilde{\mathbf{x}}_i) + I(t>1) \sum_{j=1}^{t-1} \frac{1}{t} k(\mathbf{x},\mathbf{x}_j) \qquad (1)
\end{aligned}$$

As implied by the indicator function $I(t>1)$, the second term in the objective is not applied to the first herded sample.

As shown in [3], if $k(\mathbf{x},\mathbf{x}) = R > 0$ for all $\mathbf{x} \in \mathcal{X}$, kernel herding generates samples that minimize the maximum mean discrepaency (MMD) [31] defined as follows

$$\mathcal{E}_{\text{MMD}} = \left\| \hat{\boldsymbol{\mu}}_P - \frac{1}{n} \sum_{t=1}^{n} k(\mathbf{x}_t,\cdot) \right\|_{\mathcal{H}}^2$$

For bounded kernels, kernel herding decreases $\mathcal{E}_{\text{MMD}}$ at a rate of $O(n^{-1/2})$, and under additional assumptions (only guaranteed for finite-dimensional $\mathcal{H}$), this convergence rate improves to $O(n^{-1})$. Since this is faster than random sampling from P itself [32], [3], herded samples are also called 'super samples'. While $\mathcal{H}$ is generally not finite dimensional, we still tend to get fast convergence in practice, even when Equation 1 is only solved approximately [3].

In the first iteration of kernel herding, minimizing $\mathcal{E}_k$ in Equation 1 corresponds to finding a new sample $\mathbf{x}$ such that $k(\mathbf{x},\cdot)$ is close to the target mean embedding $\hat{\boldsymbol{\mu}}_P$. For $t>1$, we append this attractive force with an additional repulsive force that penalizes $\mathbf{x}$ that are similar (as determined by the kernel) to those already herded in prior iterations. The fast convergence of kernel herding is often attributed to this repulsive force that can induce negative auto-correlation between the herded samples [3]. Positive auto-correlation between generated samples can cause them to be tightly clustered for small n, as observed in various MCMC methods [33]. Other kernel-based sampling methods such as Stein variational gradient descent [34] and Stein points [9] also employ a similar repulsive force. The kernel herding updates in Equation 1 can also be interpreted as a special case of the Frank Wolfe algorithm [32].

III. Kernel Herding on Riemannian Manifolds

We identify two attributes of kernel herding that make it particularly amenable to be adapted to general Riemannian manifolds. First, since the algorithm accesses data exclusively through kernel evaluations, geometric structure of the data space can be injected by picking an appropriate kernel. Such geometry-aware kernels should take into account the curvature of the underlying manifold when assigning similarities between data points. Second, the herding algorithm does not prescribe a particular method to solve Equation 1; it has been been solved using approaches including gradient descent [3], black-box optimization [9], and exhaustive search over a restricted search-space [4]. Over the years, Riemannian counterparts have been developed for many such optimization routines that optimize directly on the manifold of interest [35], [36]. Thus, our approach to kernel herding over Riemannian manifolds is to (1) use kernels that incorporate the

correct notion of distance for the manifold, and (2) use Riemannian optimization techniques to restrict herded samples on desired manifolds. Since the theoretical properties of kernel herding are agnostic to these two adjustments, convergence guarantees of the original algorithm carry over to our Riemannian adaptation. We now summarize the two components of our proposal to adapt kernel herding to non-Euclidean Riemannian manifolds.

Riemannian Manifolds We provide a brief summary of Riemannian manifolds, and describe some of its properties relevant to designing geometry-aware kernels and optimization algorithms. An n-dimensional manifold $\mathcal{M}$ is a topological space that is locally homeomorphic to $\mathbb{R}^n$. Intuitively, manifolds are similar to flat Euclidean space within a neighborhood around any point, even though they may exhibit significant curvature outside the neighborhood. For a point $\mathbf{x}$ on a *differentiable* manifold, the vector space spanned by the tangent vectors of all possible curves passing through $\mathbf{x}$ is called the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ at $\mathbf{x}$. Note that the tangent space is different for different points on the manifold; the Euclidean manifold is a special case where all tangent spaces are isomorphic to the manifold itself. A *Riemannian manifold* is a differentiable manifold with a family of smoothly varying inner products defined on its tangent spaces. The length of a curve γ on a Riemmanian manifold is computed by integrating the square root of these smoothly varying inner products of tangent vectors along it. A *geodesic curve* on $\mathcal{M}$ is one that locally minimizes distances along it, and the *geodesic distance* $d_{\mathcal{M}}(\mathbf{x},\mathbf{y})$ between two points $\mathbf{x},\mathbf{y} \in \mathcal{M}$ is the length of the shortest geodesic curve connecting them. We can derive closed form formulae for geodesic distances for many commonly used Riemannian manifolds.

Geodesic Exponential Kernels The first component in our Riemannian kernel herding scheme is to use geometry-aware kernels that assign similarity based on the curvature of the data manifold. Various geometry-aware kernels have been specifcally designed for manifolds such as the Grassmann manifold [37], [38], symmetric positive definite matrices [39], cylinders [18], etc. As a simpler alternative to designing problem specific kernels, we explore the use of *generic* kernels that are applicable across a wide range of domains. One such kernel is the exponential kernel defined as follows

$$k(\mathbf{x},\mathbf{y}) = \exp(-\lambda\, d(\mathbf{x},\mathbf{y})^q) \quad \lambda,\, q > 0 \tag{2}$$

where λ is the bandwidth, and $d(\mathbf{x},\mathbf{y})$ is a distance function raised to the q-th power. When $d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2$, we obtain the well-known Euclidean Laplacian ($q = 1$) and Gaussian ($q=2$) kernels. A natural way to adapt these kernels to a general Riemannian manifold $\mathcal{M}$ is to simply set the distance function $d(\mathbf{x},\mathbf{y})$ to be its geodesic distance $d_{\mathcal{M}}(\mathbf{x},\mathbf{y})$ [40], [39], [17]. However, the positivity of the kernel evaluation in Equation 2 does not guarantee a positive definite kernel; we require that the kernel always results in positive semi-definite kernel matrices [2]. Indeed, *geodesic exponential kernels* are, in general, not SPD [41] and thus may not define a valid RKHS. The geodesic Gaussian kernel is SPD for all bandwidths only if the manifold is isomorphic to Euclidean space [41]. The geodesic Laplacian kernel is slightly more flexible and is SPD for all bandwidths if the geodesic distance is a conditionally negative definite metric, which is the case for manifolds including spheres and hyperbolic spaces [41].

While geodesic exponential kernels may not be SPD for *all* bandwidths, they tend to be SPD with very high probability for bandwidths above some threshold when evaluated on finite datasets [42]. As the kernel bandwidth goes to infinity, the kernel matrix approaches the identity, which is SPD with non-negative eigenvalues. Since the minimum eigenvalue function is continuous, there exists some threshold bandwidth beyond which the kernel is always SPD. Prior works [42], [17] have estimated this bandwidth threshold by checking how often the kernel matrix is positive definite when evaluated on random sub-samples of data for different bandwidths. If this threshold is too large, the kernel matrix could degenerate into the identity, assigning zero similarity to non-identical samples. However, empirical results [42], [17], including our own, suggest that threshold bandwidths for geodesic exponential kernels tend to be small enough for the kernel to remain sufficiently discriminative.

From a practical standpoint, geodesic exponential kernels are very flexible. All we need to apply them on a manifold is its geodesic distance, and the empirically calculated bandwidth threshold. In our experiments, we opt for geodesic Laplacian kernels ($q=1$ in Equation 2). We found this kernel to outperform the Gaussian kernel on resampling experiments over various manifolds. Moreover, the Laplacian kernel is provably SPD for the spherical manifold, an important manifold routinely encountered in many practical applications. Recently, kernels with q greater than but still close to 1 have been shown to work well for a different problem setting over the spherical manifold of unit quaternions [43]; the authors chose to avoid the exact Laplacian kernel due to gradient instabilities in their optimization problem.

Optimization over Riemannian Manifolds The second component in our Riemannian kernel herding scheme is to swap Euclidean optimization methods used to solve Equation 1 with their Riemannian generalizations. The key insight in these generalizations is to translate motion along a straight line in flat spaces to motion along curves. For instance, consider the case of Riemannian gradient descent, which we use in our experiments. Gradient descent follows a simple recipe: step along the direction opposite to the gradient $g(\mathbf{x})$ of the objective until we arrive at an optima. In Euclidean space, this is done simply by adding $-\alpha g(\mathbf{x})$ to our current estimate $\mathbf{x}$, where α is the step-size. However, in non-Euclidean spaces, we cannot add the gradient $g(\mathbf{x}) \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ to points on the manifold, which is generally not isomorphic to $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Thus, we must first map motion along the gradient direction onto a smooth curve on the manifold – obtaining a *curve* of steepest descent. This is achieved using a *retraction map* $\mathcal{R}_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \rightarrow \mathcal{M}$, which corresponds to moving along a smooth curve $\gamma(t)$ on the manifold such that we start at our current estimate ($\gamma(0) = \mathbf{x}$) pointed towards the negative gradient ($\gamma'(0) = -g(\mathbf{x})$). The gradient update step is then simply computed as $\mathbf{x}_t = \mathcal{R}_{\mathbf{x}_{t-1}}(-\alpha g(\mathbf{x}_{t-1}))$. Retraction maps can be derived in closed-form for many commonly used Riemannian manifolds.

Since gradient descent can be slow to converge, it is generally supplemented with additional optimization techniques such as the conjugate gradient method [44], [45], momentum [46], [47], RMSprop, Adam [48], etc. These adaptive strategies combine the current gradient of the objective with gradients computed at prior iterations to form averaged descent directions. In Euclidean

space, this is done simply as a weighted linear combination since all gradients are elements of the same tangent space. For general Riemannian manifolds, we must first map all gradients to be combined into a common tangent space. This is done using the manifold's *parallel transport map* $\Gamma_{\mathbf{x}\to\mathbf{y}}:\mathcal{T}_{\mathbf{x}}\mathcal{M}\to\mathcal{T}_{\mathbf{y}}\mathcal{M}$, which maps elements from one tangent space to another while preserving inner products. Recently, [49] have developed Riemannian counterparts of various adaptive gradient descent schemes including the Riemannian Adam, which is the optimization algorithm used in our experiments unless noted otherwise. This optimization scheme provides computational benefits over various second-order methods [50], [51], and provides improvements over vanilla gradient descent through adaptive gradient updates. Moreover, the traditional Adam [48] is widely used in Euclidean optimization, which is used in our baseline comparisons.

IV. Experiments

We evaluate our proposal for Riemannian kernel herding on two tasks: resampling from empirical distributions and estimating parameters of black-box simulators. The first task of resampling is useful in robotics problems such as recovering from sample degeneracy in particle filters [52]. In the second set of experiments, we demonstrate how adapting kernel herding to general Riemannian manifolds allows us to do the same for useful algorithms that use herding as an intermediate step. As an example, we apply the kernel recursive approximate Bayesian computation (KR-ABC) algorithm to estimate SPD matrix parameters of simulators, including the covariance matrix of a Gaussian distribution and the inertia matrix of the palm of an Adroit robot hand simulator.

For all experiments, we use the geodesic Laplacian kernel, defined as $k(\mathbf{x},\mathbf{y})=\exp(-\lambda\, d_{\mathcal{M}}(\mathbf{x},\mathbf{y}))$ for the relevant manifold. As these kernels tend to be SPD only above some threshold bandwidth for non-Euclidean manifolds, we empirically calculate this threshold via the protocol used in [42] and [17]. Namely, we draw 100 uniformly distributed random samples for the manifold, and compute the geodesic Laplacian kernel matrix over a range of 50 bandwidths and empirically estimate the bandwidth above which kernel matrices are SPD with probability $\sim$1. For each bandwidth, we repeat this process 10 times and calculate the proportion of the resulting kernel matrices that are positive semi-definite, i.e., have non-negative eigenvalues. When tuning kernel parameters, we only search over bandwidths above this empirical threshold. Note that this step is not required for the spherical manifold, for which the geodesic Laplacian kernel is always SPD.

Manifold	Geodesic Distance $d_{\mathcal{M}}(\mathbf{x},\mathbf{y})$
S^3 Hypersphere	$\arccos\left(\mathbf{x}^T\mathbf{y}\right)$
SO(3) Rotation matrices	$\left(\sum_{k=1}^{n}\theta_k^2\right)^{1/2}$ $\{e^{i\theta_k}\}$ are eigenvalues of $\mathbf{x}^T\mathbf{y}$
SPD matrices	$\left\|\texttt{logm}\left(\mathbf{x}^{-\frac{1}{2}}\mathbf{y}\mathbf{x}^{-\frac{1}{2}}\right)\right\|_F$

TABLE I: Geodesic distances. `logm` is the matrix log operation.

We perform experiments over various commonly used manifolds in robotics: the SO(3) manifold of rotation matrices, the S^3 spherical manifold that can encode rotations as unit quaternions, and the manifold of SPD matrices, which are used to encode quantities like inertia, stiffness, etc. For the S^3 and SPD manifolds we use the arc-cosine distance and the affine invariant metric as geodesic distances respectively. For the SO(3) manifold, we use the geodesic distance defined for connected components (including SO(3)) on the orthogonal group [53]. We provide the closed-form expressions for these distances in Table I. Except for SO(3), we use the Riemannian Adam algorithm from the Geoopt package [54], [55] for optimization. Since SO(3) is not supported in Geoopt, we use the conjugate gradient method with adaptive line-search from the Pymanopt package [56]. All our experiments were performed on a desktop with 16 Intel Core i9-9900K 3.60GHz CPUs, and 34.4 GB RAM.

A. Resampling

As our first experiment, we evaluate Riemannian kernel herding on the task of generating samples from an empirical distribution of weighted particles. We compare our approach against an adaptation of a recently proposed technique [57] based on optimal transport (OT) theory to draw resamples from a weighted set of particles from Lie groups (a differentiable manifold with additional group structure) including SO(3). While the original resampling approach in [57] was designed for Lie groups, we apply the general strategy to Riemannian manifolds with known geodesic distance functions given in Table I.

The OT approach proceeds in two steps. First, we draw samples with standard sampling-importance resampling (SIR), i.e., we pick out samples (with replacement) from the original set of particles according to their weights. If the particle weights are not fairly uniform, a few highly weighted particles can dominate the set of SIR resamples, leading to reduced particle diversity or sample impoverishment [52]. To combat this, the second step of the OT approach computes a transportation map from the empirical distribution of the original particles to that of the SIR resamples using the manifold's geodesic distance as the ground metric. The final output is a fresh set of samples matching the distribution of the SIR resamples but with higher particle diversity. To compute the OT map for a particle, we first solve the entropy-regularized OT problem (using the Python Optimal Transport package [58]), which results in a set of weights for the SIR resamples for each particle. The transport map for a particle is then calculated as the Riemannian weighted centroid of the SIR resamples using these new weights. For Lie groups such as SO(3), the authors solve this centroid problem using an iterative scheme relying on the Cayley lifting map [57].

In Figure 1, we compare the OT approach against Riemannian kernel herding on the task of resampling from a set of 750 SO(3) matrices with normally distributed positive weights. We created 6 such datasets, using the first one to tune model parameters, and the remaining 5 to evaluate them. For kernel herding, we performed a random hyperparameter search over kernel bandwidths and learning rates for Adam. For the OT approach, we similarly tuned its entropy regularization. We evaluate the resamples via their average *sampling errors* on test sets, i.e., the first Wasserstein distance to the weighted targets using the manifold's geodesic distance as the ground metric. As shown in Figure 1, we experiment with computing the Riemannian centroid required for the OT approach using the iterative Cayley maps proposed in [57] (labeled *OT + Cayley Centroid*) as well as using gradient descent

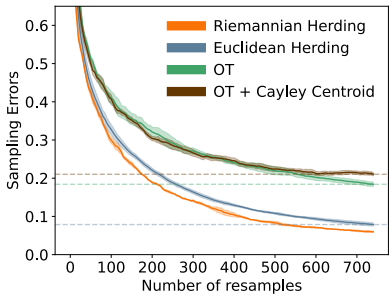

Fig. 1: Resampling on SO(3) (rotation matrices). Error bars represent standard deviation of the mean error over 5 test sets.

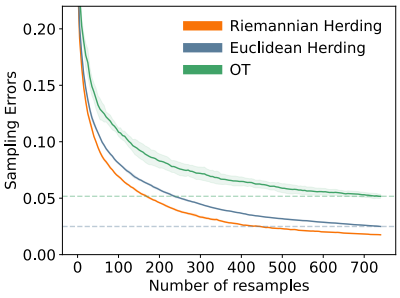

Fig. 2: Resampling on S^3 hyperspheres (quaternions). Error bars represent one standard deviation of the mean error over 5 test sets.

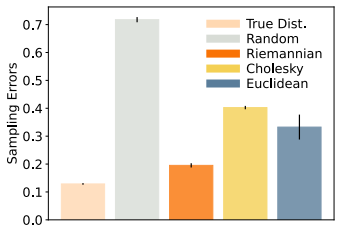

Fig. 3: Covariance matrix estimation for a Gaussian. Error bars denote one standard deviation around the mean error across 9 sets of test observations.

(labeled *OT*). We find that Riemannian kernel herding is able to match the final sampling error from the OT approaches with about 500 fewer samples. We also present results for Euclidean kernel herding, where we use the Euclidean Laplacian kernel and Euclidean gradient descent (with Adam) but heuristically project the generated samples onto the SO(3) manifold. Riemannian kernel herding is able to match the final sampling error of this Euclidean approach with around 200 fewer resamples.

Besides matrices on SO(3), rotations can also be parameterized by unit-quaternions, which are elements of the spherical manifold S^3. In fact, quaternions are often preferred over rotation matrices as they use fewer parameters, and are more computationally efficient. We repeat the earlier experiment for the S^3 manifold and report results in Figure 2. As before, we find that Riemannian kernel herding outperforms both the OT approach as well as Euclidean kernel herding.

The faster convergence of Riemannian kernel herding over the OT approach is likely due to negative sample auto-correlations induced by Equation 1 [3]. Furthermore, while the OT approach transports particles towards the distribution of SIR resamples, kernel herding draws samples directly from the original weighted empirical distribution. However, the OT approach does have the advantage of being computationally faster and highly parallelizable across samples, while kernel herding tends to be slower and is inherently sequential. Thus, in practice, Riemannian kernel herding is likely to provide an advantage over the OT approach when it is worth spending additional computation to produce high quality resamples. This may be the case, for instance, in particle filters with computationally expensive transition or observation models. As a final comparison between the two approaches, we note that Riemannian kernel herding provides greater flexibility by allowing the weights on target samples to be negative. Such negative weights are routinely encountered in methods operating directly with kernel mean embedddings. We now focus on one such method for simulator parameter estimation where negative weights prevent us from using the OT approach.

B. Simulator Parameter Estimation

In our second set of experiments, we demonstrate how adapting kernel herding to Riemannian manifolds allows us to do the same for other algorithms reliant on it. Specifically, we focus on the kernel recursive approximate Bayesian computation (KR-ABC) algorithm applied to the problem of estimating the parameters of simulators [6]. Consider a black-box simulator that takes as input a parameter θ, and produces observations y conditioned on it. Given some target observation y^*, our goal is to find parameters that are likely to have generated it. In KR-ABC, we begin by initializing a set of n parameters $\{\theta_i\}_{i=1}^n$, and query the simulator to obtain corresponding observations $\{y_i\}_{i=1}^n$. Given kernels k_θ and k_y for the parameters and observations respectively, we estimate the kernel mean embedding of the distribution $P_{y^*}(\theta)$ of parameters likely to have produced y^* as follows

$$\hat{\mu}_{P_{y^*}(\theta)} = \sum_{i=1}^{n} w_i k_\theta(\cdot, \theta_i)$$
$$\text{where } [w_1, \ldots, w_n]^T = (G + n\delta\mathbb{I})^{-1}\mathbf{k}_y(y^*)$$

Here, $\mathbf{k}(y^*) = [k_y(y_1, y^*), \cdots k_y(y_n, y^*)]^T$, $G = [k_y(y_i, y_j)]_{i,j=1}^n$, $\mathbb{I}$ is the identity matrix and $\delta > 0$ is a regularization constant. The calculation of the (potentially negative) weights above corresponds to a kernel-ridge regression [6]. At each iteration, kernel herding is used to draw samples from $\hat{\mu}_{P_{y^*}(\theta)}$, which are then applied to the simulator to generate observations for the next iteration. After the final iteration, we return the first herded sample as our estimated parameter.

For our experiments, we focus on problems where θ is an element of a Riemannian manifold. Namely, we apply Riemannian KR-ABC, using our proposed Riemannian kernel herding scheme, to two problems where the parameters of interest are SPD matrices. In our first problem, the simulator is a 3D zero-mean Gaussian distribution, and we estimate its SPD covariance matrix using samples drawn from it. We created 10 datasets, each consisting of 1000 samples from the target distribution as observations; we use the first dataset to tune hyperparameters, and the rest to evaluate models. For comparison, we also implemented

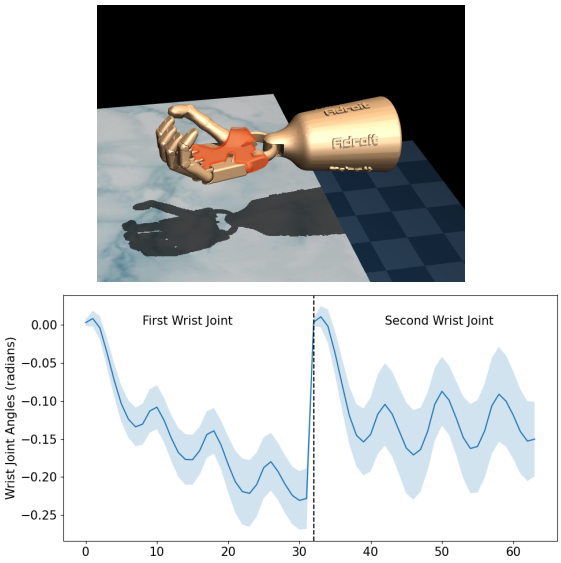

Fig. 4: (top) The Adroit robot hand simulator. We estimate the inertia matrix of its palm, highlighted in red. (bottom) The trajectory of angular positions of the two wrist joints. Our observations are the concatenation of these two 32-step trajectories. The shaded region denotes the standard deviation within the different trajectories.

Euclidean KR-ABC, where k_θ is the Euclidean Laplacian kernel and we solve the herding problem using Euclidean optimization followed by a projection onto SPD matrices. We also test another Euclidean variant that parameterizes SPD matrices via a Cholesky decomposition, avoiding the need for projections. For all methods, we set the number of herded samples $n = 100$, and tune the regularizer δ, the learning rate for Adam, and the bandwidths for k_θ and k_y (a Euclidean Laplacian kernel) through a randomized hyperparameter search. We fix the number of epochs for Adam at 100 and run KR-ABC for 20 iterations.

In Figure 3, we compare the different approaches in terms of their sampling errors, i.e., the first Wasserstein distance between the target samples and those generated from the estimated covariance matrices. For comparison, we also plot the sampling errors of iid samples drawn from the true distribution (labeled *True Dist*) and samples drawn from a random uniform distribution (labeled *Random*). Recall that kernel herding is known to result in super samples that converge faster than iid sampling. We find that Riemannian KR-ABC outperforms all other methods, yielding sampling errors much lower than that for random uniform samples and only slightly higher than that for iid samples from the true distribution.

Finally, we apply Riemannian KR-ABC with Riemannian kernel herding to the task of estimating the 3D inertia matrix of the palm of an Adroit robotic hand simulator[1] (Figure 4). We use as observations the concatenated trajectories of angular positions of its two wrist joints under a fixed control sequence for 32 steps, i.e., a 64 dimensional observation. Our goal is to learn an inertia matrix that produces trajectories similar to the target observations. This problem setting can arise in practice when calibrating a robotic simulator based on observations collected from a real robot [7].

We generated 6 distinct trajectories of observations by applying

[1]Environments obtained from `https://github.com/vikashplus/mj_envs` [59]

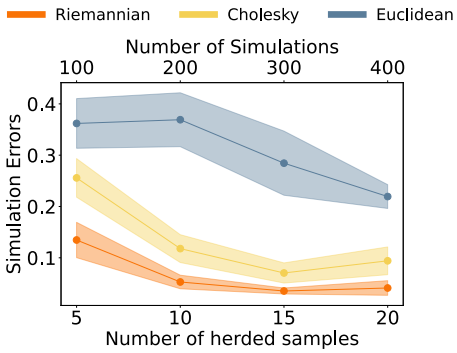

Fig. 5: Inertia matrix estimation for the robot hand simulator. Error bars denote standard error in the mean over 5 sets of observations.

different sequences of controls; we use the first one to tune hyperparameters and the remaining to evaluate models. When evaluating a model's performance, we compute its *simulation error*, i.e., the average norm of the difference between the true observations and those simulated from the estimated parameters using the same controls. In Figure 5, we show that Riemannian KR-ABC with Riemannian kernel herding outperforms both Euclidean KR-ABC with heuristic projections and the Cholesky parameterized variant without projections. The gains from the Riemannian approach appear to be larger for smaller number of parameter samples, which directly translates to the number of simulator queries at each iteration of the KR-ABC algorithm. Thus, Riemannian KR-ABC was able to obtain better estimates of the inertia matrix with fewer simulator queries than the other methods.

V. Conclusion

We presented an approach to adapting kernel herding to Riemannian manifolds by using (1) geometry-aware kernels that incorporate the correct distance metric for the manifold, and (2) Riemannian optimization to constrain generated samples to lie on the manifold. We demonstrated that our approach outperforms various alternatives on experiments involving resampling from empirical distributions, and estimating parameters of black-box simulators. Our approach provides an intuitive and effective way of incorporating geometric information into the task of sampling from kernel mean embeddings. Possible directions for future work may be to apply a similar approach to related sampling techniques such as the Stein points algorithm [9] which also uses herding. Another direction could be to identify other algorithms that could benefit from using herding as an intermediate step. Additionally, in this paper, we chose to focus on geodesic exponential kernels due to their generality. For specific applications, it would be pertinent to explore kernels customized or learned for the task at hand, as well as kernels that are provably characteristic. Applications that benefit from incorporating kernel herding are likely to be those that require computing expectations of functions with minimal sample evaluations. One such application may be problems requiring evaluations of implicit stochastic policies in robotic control and reinforcement learning.

VI. Acknowledgements

We thank Josie Thompson for assistance with experiments.

REFERENCES

[1] L. Song, K. Fukumizu, and A. Gretton, "Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 98–111, 2013.

[2] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, "Kernel mean embedding of distributions: A review and beyond," *Found. Trends Mach. Learn.*, vol. 10, no. 1-2, 2017.

[3] Y. Chen, M. Welling, and A. Smola, "Super-samples from kernel herding," in *UAI*, 2010.

[4] M. Kanagawa, Y. Nishiyama, A. Gretton, and K. Fukumizu, "Filtering with state-observation examples via kernel monte carlo filter," *Neural Comput.*, vol. 28, no. 2, 2016.

[5] S. Lacoste-Julien, F. Lindsten, and F. Bach, "Sequential kernel herding: Frank-wolfe optimization for particle filtering," in *Artificial Intelligence and Statistics*, 2015.

[6] T. Kajihara, M. Kanagawa, K. Yamazaki, and K. Fukumizu, "Kernel recursive abc: Point estimation with intractable likelihood," in *ICML*, 2018.

[7] K. Kisamori, M. Kanagawa, and K. Yamazaki, "Simulator calibration under covariate shift with kernels," in *AISTATS*, 2020.

[8] K. Hsu and F. Ramos, "Bayesian learning of conditional kernel mean embeddings for automatic likelihood-free inference," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

[9] W. Y. Chen, L. Mackey, J. Gorham, F.-X. Briol, and C. Oates, "Stein points," in *International Conference on Machine Learning*, 2018.

[10] K. V. Mardia and P. E. Jupp, *Directional statistics*. John Wiley & Sons, 2009, vol. 494.

[11] S. Sra, "Directional statistics in machine learning: a brief review," *Applied Directional Statistics: Modern Methods and Case Studies*, vol. 225, 2018.

[12] J. Reisinger, A. Waters, B. Silverthorn, and R. J. Mooney, "Spherical topic models," in *ICML*, 2010.

[13] K. V. Mardia, C. C. Taylor, and G. K. Subramaniam, "Protein bioinformatics and mixtures of bivariate von mises distributions for angular data," *Biometrics*, vol. 63, no. 2, pp. 505–512, 2007.

[14] H. Tang, S. M. Chu, and T. S. Huang, "Generative model-based speaker clustering via mixture of von mises-fisher distributions," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009.

[15] S. Calinon, "Gaussians on riemannian manifolds: Applications for robot learning and adaptive control," *IEEE Robotics & Automation Magazine*, vol. 27, no. 2, pp. 33–45, 2020.

[16] M. J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on riemannian manifolds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1240–1247, 2017.

[17] N. Jaquier, L. Rozo, S. Calinon, and M. Bürger, "Bayesian optimization meets riemannian manifolds in robot learning," in *CoRL*, 2020.

[18] C. Oh, E. Gavves, and M. Welling, "Bock: Bayesian optimization with cylindrical kernels," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3868–3877.

[19] P. Honeine and C. Richard, "The angular kernel in machine learning for hyperspectral data classification," in *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*. IEEE, 2010, pp. 1–4.

[20] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Combining multiple manifold-valued descriptors for improved object recognition," in *2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2013, pp. 1–6.

[21] M. Girolami and B. Calderhead, "Riemann manifold langevin and hamiltonian monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 2, pp. 123–214, 2011.

[22] M. Brubaker, M. Salzmann, and R. Urtasun, "A family of mcmc methods on implicitly defined manifolds," in *Artificial intelligence and statistics*. PMLR, 2012, pp. 161–172.

[23] S. Byrne and M. Girolami, "Geodesic monte carlo on embedded manifolds," *Scandinavian Journal of Statistics*, vol. 40, no. 4, pp. 825–845, 2013.

[24] P. Diaconis, S. Holmes, M. Shahshahani, *et al.*, "Sampling from a manifold," in *Advances in modern statistical theory and applications: a Festschrift in honor of Morris L. Eaton*. Institute of Mathematical Statistics, 2013, pp. 102–125.

[25] C. Liu and J. Zhu, "Riemannian stein variational gradient descent for bayesian inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[26] A. Barp, C. Oates, E. Porcu, M. Girolami, *et al.*, "A riemann-stein kernel method," *arXiv preprint arXiv:1810.04946*, 2018.

[27] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.

[28] A. Koppel, A. S. Bedi, V. Elvira, and B. M. Sadler, "Nearly consistent finite particle estimates in streaming importance sampling," *IEEE Transactions on Signal Processing*, vol. 69, pp. 6401–6415, 2021.

[29] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A hilbert space embedding for distributions," in *International Conference on Algorithmic Learning Theory*. Springer, 2007, pp. 13–31.

[30] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf, "Kernel measures of conditional dependence." in *NeurIPS*, vol. 20, 2007.

[31] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[32] F. Bach, S. Lacoste-Julien, and G. Obozinski, "On the equivalence between herding and conditional gradient algorithms," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012.

[33] C. Geyer, "Introduction to markov chain monte carlo," *Handbook of markov chain monte carlo*, vol. 20116022, p. 45, 2011.

[34] Q. Liu and D. Wang, "Stein variational gradient descent: a general purpose bayesian inference algorithm," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 2378–2386.

[35] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[36] N. Boumal, "An introduction to optimization on smooth manifolds," *Available online, Aug*, 2020.

[37] J. Hamm and D. Lee, "Extended grassmann kernels for subspace-based learning," *Advances in neural information processing systems*, vol. 21, pp. 601–608, 2008.

[38] J. Hamm and D. D. Lee, "Grassmann discriminant analysis: a unifying view on subspace-based learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 376–383.

[39] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Kernel methods on the riemannian manifold of symmetric positive definite matrices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 73–80.

[40] ——, "Kernel methods on riemannian manifolds with gaussian rbf kernels," in *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, 2015, pp. 2464–2477.

[41] A. Feragen, F. Lauze, and S. Hauberg, "Geodesic exponential kernels: When curvature and linearity conflict," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[42] 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 *COLT*, 2016.

[43] T. Birdal, M. Arbel, U. Simsekli, and L. J. Guibas, "Synchronizing probability measures on rotations via optimal transport," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1566–1576, 2020.

[44] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*. NBS Washington, DC, 1952, vol. 49, no. 1.

[45] Y.-H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property," *SIAM Journal on optimization*, vol. 10, no. 1, pp. 177–182, 1999.

[46] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr Computational Mathematics and Mathematical Physics*, vol. 4, pp. 1–17, 1964.

[47] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks : the official journal of the International Neural Network Society*, vol. 12 1, pp. 145–151, 1999.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[49] G. Bécigneul and O.-E. Ganea, "Riemannian adaptive optimization methods," in *ICML*, 2019.

[50] P.-A. Absil, C. G. Baker, and K. A. Gallivan, "Trust-region methods on riemannian manifolds," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 303–330, 2007.

[51] S. T. Smith, "Optimization techniques on riemannian manifolds," *Fields institute communications*, vol. 3, no. 3, pp. 113–135, 1994.

[52] A. Doucet, A. M. Johansen, *et al.*, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009.

[53] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, 1998.

[54] M. Kochurov, R. Karimov, and S. Kozlukov, "Geoopt: Riemannian optimization in pytorch," *arXiv preprint arXiv:2005.02819*, 2020.

[55] G. Bécigneul and O.-E. Ganea, "Riemannian adaptive optimization methods," *ArXiv*, vol. abs/1810.00760, 2019.
[56] J. Townsend, N. Koep, and S. Weichwald, "Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation," *J. Mach. Learn. Res.*, vol. 17, no. 137, 2016.
[57] Z. Wang and V. Solo, "Lie group state estimation via optimal transport," in *ICASSP*, 2020.
[58] R. Flamary and N. Courty, "Pot python optimal transport library," 2017. [Online]. Available: https://pythonot.github.io/
[59] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.