Manifold Random Features

Ananya Parashar[1 *], **Derek Long**[1 *], **Dwaipayan Saha**[1 *], **Krzysztof Choromanski**[1, 2 * †]
[1]Columbia University, [2]Google DeepMind

Abstract

We present a new paradigm for creating random features to approximate bi-variate functions (in particular, kernels) defined on general manifolds. This new mechanism of *Manifold Random Features* (MRFs) leverages discretization of the manifold and the recently introduced technique of *Graph Random Features* (GRFs; [8]) to learn continuous fields on manifolds. Those fields are used to find continuous approximation mechanisms that otherwise, in general scenarios, cannot be derived analytically. MRFs provide positive and bounded features, a key property for accurate, low-variance approximation. We show deep asymptotic connection between GRFs, defined on discrete graph objects, and continuous random features used for regular kernels. As a by-product of our method, we re-discover recently introduced mechanism of Gaussian kernel approximation applied in particular to improve linear-attention Transformers, considering simple random walks on graphs and by-passing original complex mathematical computations. We complement our algorithm with a rigorous theoretical analysis and verify in thorough experimental studies.

1 Introduction & Related Work

Random features (RFs) [26, 27, 17, 36, 25, 21, 11, 20, 9, 6, 30, 37] provide powerful techniques for translating potentially highly non-linear bi-variate functions into a simple dot-product (linear) kernel via randomized nonlinear transformations $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^m$ applied separately to two input vectors. Thus RFs lead to effective mappings between nonlinear methods in the original spaces and well-understood and computationally more efficient linear methods in the ϕ-transformed spaces. Examples range from classic algorithms such as SVM [15, 33] to feedforward fully-connected architectures [31, 16] and attention techniques in Transformers [10, 18, 23, 22, 34, 39].

Constructing efficient mappings $\phi_1, \phi_2 : \mathbb{R}^d \rightarrow \mathbb{R}^m$ (in many applications, a symmetric setting $\phi_1 = \phi_2 = \phi$ is considered, but a priori it is not a strict requirement) to approximate given bi-variate function $F : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ (with error $\epsilon \rightarrow 0$ as $m, d \rightarrow \infty$, often unbiasedly), as:

$$F(\mathbf{x}, \mathbf{y}) \approx \phi_1(\mathbf{x})^\top \phi_2(\mathbf{y}), \tag{1}$$

is a nontrivial task, even for the well-known kernel-functions, which are some of the most natural targets of the RF-methods. For example, a standard method applying trigonometric random features to approximate Gaussian and softmax kernels, proposed in [26], in practice does not work well in linear-attention Transformers, since it can produce negative features for the kernel that is strictly positive. The so-called *positive random features* from [10] address this challenge, but further refinements (e.g. the boundedness of the RFs) are needed to improve approximation quality (see: [19]). The problem becomes even more challenging when bi-variate functions F are defined on non-Euclidean spaces, e.g. on manifolds (see: Fig. 5).

*equal contribution
†Senior lead

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

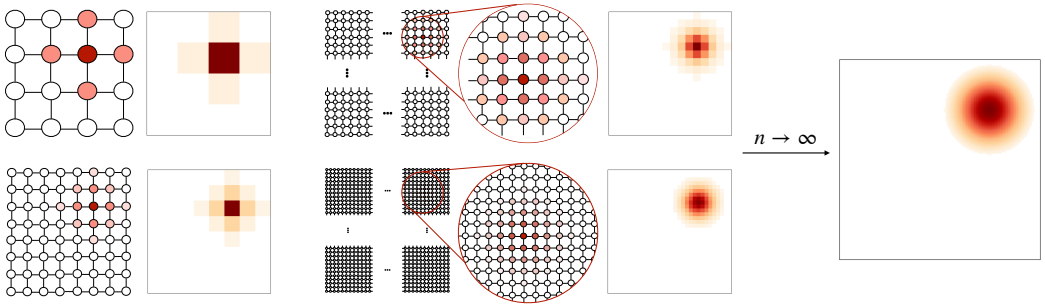

Figure 1: Four grid-graphs of sizes: 4×4, 8×8, 16×16, 32×32 with the distinguished vertex $\mathbf{z}$ and its corresponding *signature vectors* obtained by applying GRF algorithm [8]. The signature vectors are represented by color-coding different vertices (with more intense shades corresponding to large values and the most intense used to color vertex $\mathbf{z}$). Next to the graphs, those signature vectors are also represented by color-coding unit-squares. As the resolution of the grid goes to infinity, those representations converge to the continuous object, namely function g_i from Eq. 2.

In the Euclidean setting, a standard approach to constructing mappings ϕ_1, ϕ_2 is by re-writing function $F : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ as follows for some $g_1, g_2 : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$:

$$F(\mathbf{x}, \mathbf{y}) = \int_{\mathbb{R}^d} g_1(\mathbf{x}, \omega) g_2(\mathbf{y}, \omega) d\omega. \tag{2}$$

Given a probabilistic distribution $P \in \mathcal{P}(\mathbb{R}^d)$ with the corresponding density function $p : \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$, mappings $\phi_1, \phi_2 : \mathbb{R}^d \rightarrow \mathbb{R}^m$ can be then defined as follows for $i \in \{1, 2\}$:

$$\phi_i(\mathbf{z}) = \frac{1}{\sqrt{m}} \left(\frac{g_i(\mathbf{z}, \omega_1)}{\sqrt{p(\omega_1)}}, ..., \frac{g_i(\mathbf{z}, \omega_m)}{\sqrt{p(\omega_m)}} \right), \tag{3}$$

where $\omega_1, ..., \omega_m \sim P$. It is easy to see that the following holds: $F(\mathbf{x}, \mathbf{y}) = \mathbb{E}[\phi_1(\mathbf{x})^\top \phi_2(\mathbf{y})]$. It is assumed that $p(\omega) > 0$ for all $\omega \in \mathbb{R}^d$. If one wants to approximate F only on a set: $\mathcal{B}_F \times \mathcal{B}_F$ for some bounded $\mathcal{B}_F$, then in practice one can consider only "truncated" distributions $\mathcal{P}$ with p defined only on some (sufficiently large) bounded $\mathcal{B}_p$, such that $|\int_{\mathbb{R}^d \setminus \mathcal{B}_p} g_1(\mathbf{x}, \omega) g_2(\mathbf{y}, \omega) d\omega|$ is sufficiently small. Finding the representations given by Eq. 2 is usually a nontrivial task and generalizations to non-Euclidean spaces are even more challenging.

In this paper, we present a new paradigm for creating random features to approximate bi-variate functions (e.g. positive-definite kernels) defined on general manifolds. This new mechanism of *Manifold Random Features* (MRFs) leverages discretization of the manifold and the recently introduced technique of *Graph Random Features* (GRFs; [8, 29, 38, 7, 28]) to learn continuous fields on manifolds, corresponding to functions g_i from Eq. 2, but in the general (not necessarily Euclidean) setting. It turns out that these fields can be thought of as continuous extensions of the combinatorial objects, the so-called *signature vectors*, the building blocks of the GRF mechanism (see Figure 1). They are used to find continuous approximation mechanisms that otherwise, in general scenarios, cannot be derived analytically. MRFs provide positive and bounded features, a key property for accurate, low-variance approximation. We show deep asymptotic connection between GRFs, defined on discrete graph objects, and continuous random features used for regular kernels. As a by-product of our method, we rediscover recently introduced mechanism of Gaussian kernel approximation [19] applied in particular to improve linear-attention Transformers, considering simple random walks on graphs and by-passing original complex computations. We complement our algorithm with a rigorous theoretical analysis and verify it in thorough experimental studies.

2 Manifold Random Features

2.1 Preliminaries: Graph Random Features

MRFs are obtained by training continuous random feature mechanisms via discrete supervision given by *Graph Random Features* (GRFs). Our first step is to provide below a gentle introduction to GRFs.

We will consider weighted, undirected N-vertex graphs $\mathrm{G}(\mathrm{V}, \mathrm{E}, \mathbf{W} = [w(i,j)]_{i,j \in \mathrm{V}})$, where (1) V is a set of vertices, (2) $\mathrm{E} \subseteq \mathrm{V} \times \mathrm{V}$ is a set of undirected edges ($(i,j) \in \mathrm{E}$ indicates that there is an edge between i and j in G), and (3) $\mathbf{W} \in \mathbb{R}_{\geq 0}^{N \times N}$ is a weighted adjacency matrix (no-edges encoded by zeros).

We consider the following kernel matrix $\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W}) \in \mathbb{R}^{N \times N}$, where $\boldsymbol{\alpha} = (\alpha_k)_{k=0}^{\infty}$ and $\alpha_k \in \mathbb{R}$:

$$\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W}) = \sum_{k=0}^{\infty} \alpha_k \mathbf{W}^k. \tag{4}$$

For bounded $(\alpha_k)_{k=0}^{\infty}$ and $\|\mathbf{W}\|_{\infty}$ small enough, the above sum converges. The matrix $\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W})$ defines a kernel on the nodes of the graph. Importantly, Eq. 4 covers as special cases several classes of graph kernels, in particular *graph diffusion/heat kernels*, that will play an important role in our analysis later (see: Sec. 2.2.1).

GRFs provide a way to express $\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W})$ (in expectation) as $\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W}) \overset{\mathbb{E}}{=} \mathbf{K}_1 \mathbf{K}_2^{\top}$, for independently sampled $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{R}^{N \times d}$ and some $d \leq N$. This factorization gives an efficient (sub-quadratic) and unbiased approximation of the matrix-vector products $\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W})\mathbf{x}$ as $\mathbf{K}_1(\mathbf{K}_2^{\top}\mathbf{x})$, if $\mathbf{K}_1, \mathbf{K}_2$ are sparse or $d = o(N)$. This is often the case in practice. If this does not hold, explicitly materializing $\mathbf{K}_1 \mathbf{K}_2^{\top}$ enables one to approximate $\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W})$ in quadratic (versus cubic) time. Next, we describe the base GRF method for constructing sparse $\mathbf{K}_1, \mathbf{K}_2$ for $d = N$. Extensions for $d = o(N)$, using the Johnson-Lindenstrauss Transform [13], can be found in [8]. Each $\mathbf{K}_j$ for $j \in \{1, 2\}$ is obtained by row-wise stacking of the vectors $\phi_f(i) \in \mathbb{R}^N$ for $i \in \mathrm{V}$, where $f : \mathbb{R} \to \mathbb{R}$ is the graph-kernel specific *modulation function*.

The procedure to construct random vectors $\phi_f(i)$ is given in Algorithm 1. Intuitively, one samples an ensemble of random walks (RWs) from each node $i \in \mathrm{V}$. Each node maintains a scalar value, initially set to zero, which accumulates contributions from visiting walks. Every time a RW (which terminates with probability p_{halt}) visits a node, its scalar value is incremented by the walker's current *load*, weighted by the *modulation function* f evaluated at the walk length. The load is updated during each vertex-to-vertex transition, while f encodes the kernel coefficients $(\alpha_k)_{k=0}^{\infty}$.

After all walks terminate, the accumulated scalar values are concatenated and divided by the number of sampled walks to obtain the signature vector $\phi_f(i)$. For unbiased estimation, $f : \mathbb{N} \to \mathbb{C}$ needs to satisfy $\sum_{p=0}^{k} f(k-p)f(p) = \alpha_k$, for $k = 0, 1, ...$ (see Theorem 2.1 in [29]). We conclude that the modulation function f is obtained by de-convolving sequence $\boldsymbol{\alpha} = (\alpha_k)_{k=0}^{\infty}$ defining graph kernel.

2.2 Constructing MRFs

For a given Riemannian manifold $\mathcal{M}$, and a bi-variate function $F : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$, we are looking for the function $\phi_{\mathcal{M},F} : \mathcal{M} \to \mathbb{R}^m$, such that for $x, y \in \mathcal{M}$:

$$F(x,y) \approx \phi_{\mathcal{M},F}(x)^{\top} \phi_{\mathcal{M},F}(y). \tag{5}$$

We refer to $\phi_{\mathcal{M},F}(z)$ as a *manifold random features vector* for z. Since MRFs construct approximate kernels through inner products of feature vectors, the resulting kernels are positive semidefinite, even when the underlying series is truncated.

We are ready to explain how MRFs are constructed. We start with the pre-processing procedure, conducted only once for a given manifold $\mathcal{M}$, before inputs $z_1, ..., z_m$ (for which vectors $\phi_{\mathcal{M},F}(z_i)$ need to be computed) are even known. The procedure consists of the following two steps:

Discretizations of the input manifold $\mathcal{M}$: We choose a resolution parameter $N \in \mathbb{N}$. We then choose a finite set of points $\mathrm{V}_N = \{x_1, \ldots, x_N\} \subset \mathcal{M}$ that discretizes $\mathcal{M}$; for example, V_N can arise from a mesh, a quasi–uniform point cloud, or samples from the Riemannian volume measure. For a compact d-dimensional manifold, quasi-uniform sampling at spatial resolution ϵ typically requires $N = O(\epsilon^{-d})$ points, reflecting the dependence of discretization size on intrinsic dimension. We equip V_N with a weighted neighborhood graph $\mathrm{G}_N = (\mathrm{V}_N, \mathrm{E}_N, \mathbf{W}_N)$ that encodes the local geometry of $\mathcal{M}$, for instance by connecting each x_i to its k nearest neighbors with weights $\mathbf{W}_N(i,j) = \exp(-\|x_i - x_j\|^2/\sigma^2)$. We then consider a discretized version $F^{\mathrm{disc}} : \mathrm{V}_N \times \mathrm{V}_N \to \mathbb{R}$ of the original bi-variate function $F : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$, with the matrix $\mathbf{F} = [F(x_i, x_j)]_{i,j=1,\ldots,N}$

adhering to the form from Eq. 4. In the most general setting, F^{disc} can be approximated by a standard regression algorithm trained on the target kernel values $\mathcal{D} = \{(x_i, x_j, F(x_i, x_j))\}_{i,j=1}^{N}$ to learn a finite sequence of coefficients $(\alpha_0, \ldots, \alpha_K)$ for some $K > 0$, such that the resulting graph kernel from Eq. 4 approximates the discretized target kernel. However in many prominent cases of bi-variate functions defined on manifolds, this is not necessary. For instance, for the diffusion/heat kernels F defined on manifolds, the corresponding F^{disc} are the diffusion/heat kernels on the corresponding graphs G_N, with the coefficients α_k described by a closed-form formula. We describe this prominent special case in more depth in Sec. 2.2.1.

Training g-functions on manifolds: We start this phase by constructing a training dataset $\mathcal{T} = \{(x, \omega, \phi_f(x)[\omega])\}$, where $x, \omega \in \mathrm{V}_N$ and $\phi_f(x)[\omega]$ stands for the value of the signature vector $\phi_f(x)$ in node ω. In practice, $\mathcal{T}$ does not need to include the full N^2 set of pairs $(x, \omega) \in \mathrm{V}_n \times \mathrm{V}_n$ since diffusion-type kernels decay rapidly between far-apart nodes and signature vectors corresponding to neighboring nodes are similar. For example, one can subsample the starting nodes and prioritize pairs (x, ω) whose graph shortest-path distance is below a threshold R, reducing the training set size while retaining the most informative pairs. The signature vectors $\{\phi_f(x)\}_{x \in \mathrm{V}_N}$ are computed with the use of Algorithm 1 (the GRF method). We construct $\mathcal{T}$ to train a function $g_\theta^{\mathcal{M},f} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_{\geq 0}$, parameterized by θ and satisfying:

$$F(x, y) \approx \int_{\mathcal{M}} g_\theta^{\mathcal{M},f}(x, \omega) g_\theta^{\mathcal{M},f}(y, \omega) d\omega. \tag{6}$$

Note that $g_\theta^{\mathcal{M},f}$ is the generalization of the g-functions from Eq. 2. We model it as a neural network.

We train $g_\theta^{\mathcal{M},f}$ to regress the discrete signature vectors on $\mathrm{V}_N \times \mathrm{V}_N$, and clamp the final output of $g_\theta^{\mathcal{M},f}$ at zero. We define the following objective to minimize (where $\mathcal{L}$ is a $\mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ loss function, e.g. L_2-loss):

$$\min_\theta \frac{1}{|\mathcal{T}|} \sum_{(x, \omega, \phi_f(x)[\omega]) \in \mathcal{T}} \mathcal{L}\big(g_\theta^{\mathcal{M},f}(x, \omega),\, \phi_f(x)[\omega]\big).$$

After training, $g_\theta^{\mathcal{M},f}$ provides a smooth surrogate for discrete signatures: for $x, \omega \in \mathrm{V}_N$, $g_\theta^{\mathcal{M},f}(x, \omega) \approx \phi_f(x)(\omega)$. That completes pre-processing.

Inference: Trained $g_\theta^{\mathcal{M},f}$ provides a way to approximate $F(x, y)$ for any $x, y \in \mathcal{M}$ as follows:

$$F(x, y) \approx \sum_{\omega \in V_N} g_\theta^{\mathcal{M},f}(x, \omega) g_\theta^{\mathcal{M},f}(y, \omega). \tag{7}$$

This directly leads to the following definition of $\phi_{\mathcal{M},F}$:

$$\phi_{\mathcal{M},F}(z) = \frac{1}{\sqrt{m\kappa}} \left(\frac{g_\theta^{\mathcal{M},f}(z, \omega_1)}{\sqrt{p(\omega_1)}}, ..., \frac{g_\theta^{\mathcal{M},f}(z, \omega_m)}{\sqrt{p(\omega_m)}} \right), \tag{8}$$

where either: (1) $p \equiv 1$, $m = N$, $\kappa = \frac{1}{m}$, $\{\omega_1, ..., \omega_m\} = \mathrm{V}_N$ or: (2) p is the density function of the probability distribution $\mathcal{P}(\mathrm{V}_N)$ on V_N, $\kappa = 1$ and $\omega_1, ..., \omega_m \overset{\mathrm{iid}}{\sim} \mathcal{P}(\mathrm{V}_N)$. The latter variant leads to an unbiased estimation of the sum from Eq. 7 (which is exactly what the former variant is designed to compute). It provides m-dimensional RF-vectors, instead of N-dimensional and thus can be used for dimensionality reduction. Note that the full sum in Eq. 7 avoids additional sampling variance so evaluating all N feature coordinates is still preferred when tractable.

2.2.1 Diffusion/heat kernels

Let $(\mathcal{M}, g)$ be a compact d–dimensional Riemannian manifold and let $\Delta_{\mathcal{M}}$ be its Laplace–Beltrami operator [32]. The diffusion/heat kernel on $\mathcal{M}$ is defined as follows, for a given parameter $t > 0$:

$$\mathrm{K}_t^{\mathrm{heat}}(x, y) = \exp\big(t\Delta_{\mathcal{M}}\big)(x, y), \qquad x, y \in \mathcal{M}.$$

Let $\mathbf{L}_N$ denote the corresponding (rescaled) random–walk graph Laplacian on the discretized $\mathcal{M}$ (corresponding to the graph $\mathrm{G}_N = (\mathrm{V}_N = \{x_1, ..., x_N\}, \mathrm{E}_N, \mathbf{W}_N)$), chosen so that $\mathbf{L}_N$ converges

to $-\Delta_{\mathcal{M}}$ in the limit, as $N \to \infty$. The discrete graph diffusion/heat kernel at time $t > 0$ is defined as follows:

$$\mathrm{K}^{\mathrm{heat}}_{N,t}(x_i, x_j) := \big[\exp(-t\mathbf{L}_N)\big](i,j). \tag{9}$$

It converges pointwise to the manifold heat kernel $\mathrm{K}^{\mathrm{heat}}_t$ as the discretization is refined. It is easy to see that $\mathrm{K}^{\mathrm{heat}}_{N,t}$ adheres to the form from Eq. 4.

The class of manifold heat kernels provides a good insight into challenges related to using those constructs in machine learning that this paper addresses.

Let $(\mathcal{M}, g)$ be a compact d–dimensional Riemannian manifold. Since $\mathcal{M}$ is compact, the Laplace–Beltrami operator $-\Delta_{\mathcal{M}}$ has a discrete, nonnegative spectrum

$$0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \uparrow \infty,$$

with the corresponding orthonormal eigenfunctions $\{\phi_k\}_{k=0}^{\infty} \subset L^2(\mathcal{M})$ satisfying $-\Delta_{\mathcal{M}}\phi_k = \lambda_k \phi_k$ (the notation above means that the eigenvalues are nondecreasing and grow without bound as $k \to \infty$). The heat kernel then admits the following expansion:

$$\mathrm{K}^{\mathrm{heat}}_t(x,y) \;=\; \sum_{k=0}^{\infty} e^{-\lambda_k t}\, \phi_k(x)\, \phi_k(y), \quad t > 0. \tag{10}$$

This expansion converges absolutely and uniformly for any $t > 0$ on compact manifolds. For reference of this result, see Theorem 10.13 of [14].

A naive approach to computing $\mathrm{K}^{\mathrm{heat}}_t(x,y)$ is via the truncation of the infinite series from Eq. 10. For a discretization $\{x_1, \ldots, x_N\} \subset \mathcal{M}$, one can numerically approximate the first M eigenpairs of $-\Delta_{\mathcal{M}}$—or of a consistent graph Laplacian approximation $\mathbf{L}_N$—and build the truncated kernel:

$$\widehat{\mathrm{K}}^{\mathrm{heat}}_t(x,y) := \sum_{k=0}^{M-1} e^{-\lambda_k t} \phi_k(x) \phi_k(y), \tag{11}$$

which converges to (10) as $M \to \infty$. Thus even for this well-studied class of manifold bi-variate functions, the (approximate) computation of the kernel matrix $\mathbf{K}^{\mathrm{heat}}_t = [\mathrm{K}^{\mathrm{heat}}(x_i, x_j)]_{i,j=1,\ldots,N} \in \mathbb{R}^{N \times N}$ requires nontrivial spectral calculations of time complexity cubic in N. Furthermore, as it is the case in general for kernel methods not leveraging RFs, downstream applications (involving multiplications with matrices $\mathbf{K}^{\mathrm{heat}}_t$ or taking their inverses) require time $\Omega(N^2)$. In contrast, RF-based methods require sub-quadratic time (e.g. linear or log-linear for multiplications with kernel matrices, see: [5]). Note that for particularly symmetric manifolds $\mathcal{M}$, such as 2-dimensional spheres in $\mathbb{R}^3$, there exist more explicit formulae for $\mathrm{K}^{\mathrm{heat}}_t(x,y)$ (see: Appendix A.4.1), but they still require spectral computations.

Positivity & Boundedness: We reiterate that GRFs, by definition (see: Algorithm 1), provide positive random features. Furthermore, our NNs (using GRFs as teachers) are forced to produce such features (via inference-time clamping). Their boundedness follows from the fact that, in our applications (see: Sec. 3), the considered NNs are continuous functions defined on compact manifolds. As mentioned before, both properties (positivity & boundedness) play an important role in getting accurate estimators. More specifically positivity can reduce estimation variance and improve numerical stability, as demonstrated by positive random features for Gaussian/softmax kernels in linear-attention Transformer while boundedness further limits extreme feature values and supports stronger concentration guarantees.

As we already discussed in this section, general bi-variate functions F can be handled by MRFs via standard ML regression methods used to fit F to formula from Eq. 4. We also would like to emphasize that matrices $\mathbf{W}$ used there can be thought of as "generalized" weighted adjacency matrices, obtained from original weighted adjacency matrices via various transformations (e.g. degree-based normalization), as discussed in [29]. Thus in this paper we focused on the (already rich) family of functions defined by Eq. 4.

2.3 The curious case of the Gaussian kernel

MRFs are designed to efficiently apply bi-variate functions defined in non-Euclidean spaces. However, as a by-product of the presented methods, quite unexpectedly, we also obtain RF-methods for

approximating regular Gaussian kernels defined in the Euclidean spaces, that were only recently introduced. We do it by applying GRFs on grid-graphs. We do think that this result is of independent interest, since it shows how combinatorial tools can lead to purely continuous ML constructions. We provide more details here.

Consider the following setting in the d-dimensional space: the Euclidean hypercube $[0,1]^d$ equipped with the Gaussian (RBF) kernel of the following form:

$$\mathrm{K}_{\sigma}^{\mathrm{Gauss}}(\mathbf{x},\mathbf{y}) \;=\; \exp\Big(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\Big), \tag{12}$$

for a fixed $\sigma > 0$ and $x, y \in [0,1]^d$. We discretize $[0,1]^d$ with the grid of n rows/columns (so that $N = n^d$):

$$\mathrm{V}_n := \Big\{(k_1/n,\ldots,k_d/n) : k_j \in \{0,\ldots,n-1\}\Big\}, \tag{13}$$

and side length $h_n := 1/n$. We connect each grid point to its $2d$ nearest neighbors using wrap–around at the boundary, obtaining a d–dimensional discrete grid graph with vertex set V_n. Let $\mathbf{L}_n$ denote the rescaled random–walk graph Laplacian on this graph, constructed as in Theorem A.1, so that $\mathbf{L}_n$ converges to the continuous Laplacian $-\Delta$ on the discrete grid with wrap-around (the negated Laplace-Beltrami operator) in the limit $n \to \infty$. Take kernel $\mathrm{K}_{N,t}^{\mathrm{heat}}$ from Eq. 9 for $t = \frac{\sigma^2}{2}$. As discussed in Sec. 2.2.1, $\mathrm{K}_{N,t}^{\mathrm{heat}}$ converges pointwise to the heat kernel on $[0,1]^d$, which turns out to be the Gaussian kernel $\mathrm{K}_{\sigma}^{\mathrm{Gauss}}$ (to be more specific, $\mathrm{K}_{N,t}^{\mathrm{heat}}$ converges to the periodized Gaussian kernel; in the non–periodic Euclidean limit this reduces to the standard Gaussian kernel $\mathrm{K}_{\sigma}^{\mathrm{Gauss}}$, see: Theorem A.2).

For each grid point $x \in \mathrm{V}_n$ we now construct a signature vector $\phi_f(x) \in \mathbb{R}_{\geq 0}^{\mathrm{V}_n}$ corresponding to kernel $\mathrm{K}_{N,t}^{\mathrm{heat}}$, as in Algorithm 1. Those signature vectors are intrinsically related to the g-functions in the representations of Gaussian kernels applied in new RF-mechanisms to approximate them. By applying techniques from [19], we obtain the following positive and bounded features representation (for completeness, we also provide a proof, see: Theorem A.3):

$$\mathrm{K}_{\sigma}^{\mathrm{Gauss}}(\mathbf{x},\mathbf{y}) \;=\; \int_{\mathbb{R}^d} g_\sigma(\mathbf{x},\omega)\, g_\sigma(\mathbf{y},\omega)\, d\omega, \quad g_\sigma(\mathbf{x},\omega) := \Big(\frac{2}{\pi\sigma^2}\Big)^{d/4} \exp\Big(-\frac{\|\mathbf{x}-\omega\|^2}{\sigma^2}\Big). \tag{14}$$

To relate the discrete signature vectors $\phi_f(x)$ to the continuous features $g_\sigma(\mathbf{x},*)$, we interpret the sum over $\omega \in \mathrm{V}_n$ as a Riemann approximation of the integral in the expression above. We show in Appendix A.3.3 after multiplying by the grid–dependent constant $c_{d,\sigma,n} := (2\pi\sigma^2)^{d/4} n^{d/2}$, the rescaled vectors $\psi_f(x) := c_{d,\sigma,n} \cdot \phi_f(x)$ converge to $g_\sigma(\mathbf{x},\omega)$ evaluated on the grid points, as $n \to \infty$. Equivalently, for any $\mathbf{x}, \mathbf{y} \in [0,1]^d$ and the corresponding $x, y \in \mathrm{V}_n$ of the grid-graph, we have:

$$\lim_{n\to\infty} \big\langle \psi_f(x), \psi_f(y) \big\rangle \;=\; \mathrm{K}_{\sigma}^{\mathrm{Gauss}}(\mathbf{x},\mathbf{y}). \tag{15}$$

We conclude that increasingly refined grid-based GRFs recover the known continuous Gaussian random feature representation. While GRFs are not needed to evaluate these features once their explicit formula is known, this connection suggests a way to discover new continuous random feature mechanisms for other kernels on manifolds.

3 Experiments

We validate MRFs in several scenarios. As a warm-up and sanity check for the discrete-to-continuous construction, Appendix A.3.3 evaluates MRFs on regular Euclidean grids, where the limiting target is the Gaussian/RBF kernel (see: Sec. 2.3). As the grid is refined, the rescaled GRF signatures converge empirically to the analytic positive Gaussian feature map, and the induced inner products converge to the Gaussian kernel. This experiment verifies the mechanism predicted by the theory. We then transition to non-Euclidean spaces, the main target of MRFs, testing MRFs on various 2D surfaces in 3D space in Sec. 3.1, downstream interpolation tasks on meshes (representing rigid and deformable objects) in Sec. 3.2, and then on attention using manifold-valued tokens as well as higher-dimensional space and beyond in Sec. 3.3. Note that MRFs require an offline preprocessing stage: graph construction, random-walk supervision, and training of g_θ. This cost is worthwhile when

Table 1: Summary of MRF accuracy, intrinsic-kernel reconstruction, and inference speed on each manifold. The columns R^2, Mean RE, and MSE evaluate the learned surrogate $g_\theta(x, \cdot)$ against GRF signature values on validation nodes. The kernel columns report relative Frobenius error for reconstructing the intrinsic manifold heat kernel: "Best Ambient RF" is the best tuned ambient Euclidean RF baseline among RFF/ORF/PRF/PORF, allowing up to $4\times$ more features. Timing is measured on kernel evaluation for 512 out-of-sample manifold points; "Spectral" denotes the baseline spectral decomposition method. Full ambient RF results are in Appendix A.4.5, Table 3.

Manifold	**Surrogate validation**			**Kernel rel. Frob. error ↓**		**Kernel evaluation time**		
	R^2 ↑	Mean RE ↓	MSE ↓	MRF ↓	Best Ambient RF ↓	MRFs Time ↓	Spectral Time ↓	Speed-up ↑
Sphere	0.997	0.23	26.9	**0.067**	0.101	$0.066 \pm 3.3 \times 10^{-5}$	$3.41 \pm 3.1 \times 10^{-5}$	**55.2**
Ellipsoid	0.995	0.37	46.7	**0.047**	0.291	$0.058 \pm 2.0 \times 10^{-5}$	$3.59 \pm 1.7 \times 10^{-5}$	**61.2**
Möbius strip	0.997	0.11	17.9	**0.035**	0.489	$0.059 \pm 2.3 \times 10^{-5}$	$2.14 \pm 1.7 \times 10^{-5}$	**37.2**
Torus	0.983	0.44	115.7	**0.062**	0.417	$0.061 \pm 2.9 \times 10^{-5}$	$3.62 \pm 3.1 \times 10^{-5}$	**58.1**

the learned features are reused for many out-of-sample evaluations or downstream inference steps, but may not be justified for one-off small problems. Experiments were run using a TPU v6e and an A100 GPU (for neural network training only). Appendix A.9 reports an additional FAUST dense correspondence experiment, sensitivity analysis for various design choices such as graph weights, neighborhood size, network architecture, and sampling density, as well as preprocessing timings and empirical scaling fits for Thingi10K and `flag_simple`.

3.1 2D Surfaces in 3D Spaces

In Table 1, we evaluate the approximation of g-functions and diffusion/heat kernels with MRFs on four embedded non-linear geometries: a sphere, an ellipsoid, a Möbius strip, and a torus (or donut; detailed results for this shape are in the Appendix: Sec. A.4.3). More details of this procedure are in Appendix A.4.

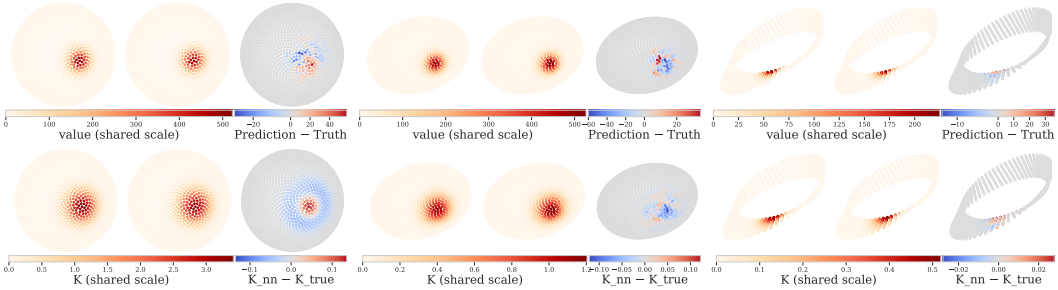

Figure 2: Qualitative field and heat kernel comparisons for a sphere (left), an ellipsoid (middle), and a Möbius strip (right). Example validation start (top row): predicted $g_\theta(\mathbf{x}, \cdot)$, ground truth $\phi_f(x)[\cdot]$, and residual $g_\theta(\mathbf{x}, \cdot) - \phi_f(x)[\cdot]$ on the geometry. Reference graph heat kernel (bottom row) $\mathrm{K}^{\mathrm{heat}}(\mathbf{x}, \cdot)$ compared to Frobenius-aligned induced kernel $\widetilde{\mathrm{K}}_\theta(\mathbf{x}, \cdot) = \widetilde{\mathrm{K}}_{\mathrm{NN}}(\mathbf{x}, \cdot)$, and their difference, for a representative start point.

For all considered manifolds, the 3D field plots confirm that the learned predictor reproduces the localized, structured heat profile (for a sphere, radially symmetric) and that residuals are concentrated near the peak (Fig. 2). After Frobenius scale alignment (Appendix A.2, Eq. 16), the induced kernel matrix $\widetilde{\mathbf{K}}_\theta$ matches the analytic kernel matrix, with remaining error again localized near the center. Validation scatter plots show strong monotonic agreement with the ground truth values, with the largest deviations at the highest-intensity region near the source point where the field is most sharply peaked. Importantly, MRFs offer significant speedups over the brute-force approaches, from $\mathbf{37x}$ to $\mathbf{61x}+$ in inference.

A natural question is whether manifold-aware features are necessary, or whether one can simply ignore the intrinsic geometry and apply standard Euclidean random features to the ambient coordinates. We therefore compare MRFs against Random Fourier Features (RFF; [26]), Orthogonal Random Features (ORF; [37]), Positive Random Features (PRF; [10]), and Positive Orthogonal Random Features (PORF) (with the best ambient RF baseline after tuning appearing in Table 1). Each baseline

Table 2: Random-feature comparison for vertex normals and velocity prediction. Values are averaged across mesh size $|V|$. Growth exponents α are fitted from interpolation time $t_{\text{interp}}(N) \propto N^{\alpha}$. For each task and each fixed feature budget m, RFF/ORF/PRF/PORF use the best tuned σ.

			$m = 256$				$m = 1024$			
Application	**Metric**	**MRF**	**RFF**	**ORF**	**PRF**	**PORF**	**RFF**	**ORF**	**PRF**	**PORF**
Vertex norm. prediction	cosine similarity ↑	**0.85**	0.71	0.72	0.40	0.41	0.78	0.78	0.47	0.43
	growth α ↓	**0.50**	1.06	1.01	1.09	0.97	1.10	1.07	1.13	1.14
Velocity prediction	cosine similarity ↑	**0.93**	0.65	0.66	0.68	0.69	0.66	0.79	0.70	0.70
	growth α ↓	**0.71**	1.18	1.20	1.19	1.19	1.03	1.02	1.04	1.03

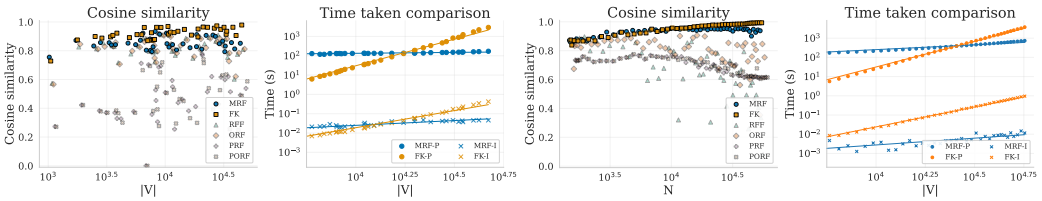

Figure 3: Comparison across methods for vertex normal prediction on `Thingi10K` (left) and velocity prediction on `flag_simple` (right); preprocessing and interpolation times denoted by '-P' and '-I'.

approximates the ambient Gaussian kernel $\exp(-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2)$ in $\mathbb{R}^3$, with σ tuned by grid search, while the target remains the intrinsic manifold heat kernel. The full comparison, including all RF methods and feature budgets $m \in \{4000, 8000, 16000\}$, is reported in Appendix A.4.5, Table 3. MRFs consistently outperform these other methods, showing that the gains come from respecting the intrinsic geometry of the manifold.

3.2 Interpolation on Meshes

As a downstream application for MRFs, we choose the task of interpolating various fields defined on meshes representing rigid and deformable objects.

Vertex normal prediction: Here we focus on predicting normal vectors in the vertices of the mesh representing rigid body. Given a mesh G with vertex set V and vertex positions $\{\mathbf{x}_i \in \mathbb{R}^3\}_{i\in \mathrm{V}}$, let $\{\mathbf{n}_i \in \mathbb{R}^3\}_{i\in \mathrm{V}}$ denote the unit vertex normals computed from incident faces. We randomly mask a subset $M \subseteq \mathrm{V}$ of size $|M| = 0.8|\mathrm{V}|$ (80% missing normals), set masked normals to zero, and predict normals at masked vertices using kernel-based field integration on G. Using meshes from the Thingi10k dataset [41], we compare our MRF approach against the baseline method of explicitly forming the full heat kernel matrix via spectral computations (see Appendix A.5.1 for more details).

Fig. 3 shows that the full-kernel construction cost grows rapidly with $|\mathrm{V}|$, whereas MRF preprocessing remains comparatively stable because the walk and training budgets are fixed. In our largest completed run ($|\mathrm{V}| = 47{,}024$), full-kernel construction took 2936 seconds, while MRF data generation and training took 171 seconds; full-kernel interpolation took 0.42 seconds, while MRF interpolation took 0.45 seconds. This preprocessing gap widens for larger meshes because explicitly storing $K \in \mathbb{R}^{|\mathrm{V}|\times|\mathrm{V}|}$ requires $O(|\mathrm{V}|^2)$ memory, making full kernel formation impractical beyond 10^5 vertices. At the same time, Table 2 shows that MRFs achieve the highest average cosine similarity and the best interpolation-time scaling among RF methods. Thus, MRFs retain the accuracy of intrinsic heat-kernel interpolation while scaling more favorably than explicit full-kernel construction and outperforming ambient Euclidean RF features.

Velocity prediction: We also evaluate interpolation on the `flag_simple` dataset [24] where the signal of interest is the per-node velocity field. For a given mesh with node set V and node positions $\{\mathbf{x}_i \in \mathbb{R}^3\}_{i\in \mathrm{V}}$, let $\{\mathbf{u}_i \in \mathbb{R}^3\}_{i\in \mathrm{V}}$ denote the ground truth velocity vectors at a fixed time step. We randomly mask a small subset $M \subseteq \mathrm{V}$ with $|M| = 0.05|\mathrm{V}|$ (5% missing velocities), set $u_i = 0$ for $i \in M$, and predict velocities at masked nodes using heat-kernel field integration (setup as in [12]). The `flag_simple` meshes have only ≈1.5k vertices, which can be too small to expose inference-time scaling trends. To probe larger problem sizes, we construct a denser node set $\widetilde{\mathrm{V}}$ by sampling additional points on triangle faces, using area-weighted barycentric sampling. The mesh surface is represented by a set of triangular faces. Concretely, for a target size $|\widetilde{\mathrm{V}}| = n_{\text{dense}}$, we draw

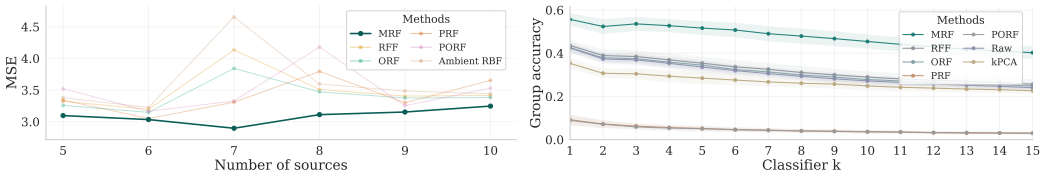

Figure 4: **Left**: Masked reconstruction on Möbius-strip tokens. **Right**: Performance on Brodatz dataset across k and over multiple seeds.

$n_{\text{dense}} - |\mathrm{V}|$ faces i.i.d. (with probability proportional to triangle area). For each sampled face, we draw barycentric weights $(\beta_0, \beta_1, \beta_2)$ (uniform on the 2-simplex) and define the sampled point's position as a convex combination of the triangle's three incident vertex positions. At each time step, we assign a velocity to each sampled point by applying the same barycentric weights to the velocities of the corresponding face vertices, yielding a dense velocity field $\mathbf{U} \in \mathbb{R}^{|\widetilde{V}| \times 3}$.

We then build a sparse graph on $\widetilde{\mathrm{V}}$ (using the reference geometry from the first evaluated frame) via a kNN construction in $\mathbb{R}^3$ and define a symmetric, normalized affinity matrix $\mathbf{W}_f := \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ where $\mathbf{W}$ has Gaussian weights $W_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\sigma^2)$ and $\mathbf{D}$ is the degree matrix of $\mathbf{W}$. The corresponding heat kernel is $\mathbf{K} = \exp(\tau \mathbf{W}_f)$. Table 2 shows that MRFs again achieve the best average cosine similarity and the best scaling exponent among RF baselines. Figure 3 further shows that the dense spectral baseline exhibits the expected steep growth in preprocessing time, while MRF preprocessing remains comparatively stable across the tested range since training/feature budgets are fixed per discretization. At inference time, MRFs consistently outperform the baseline, with speedups that exceed $\mathbf{10x}$ as n_{dense} increases.

3.3 Attention on Manifold-Valued Tokens and Manifolds Beyond 2D Surfaces

Linear-attention: A key motivation for random features is linearizing attention, replacing a nonlinear query-key similarity with an inner product of feature maps. Standard RF-based linear attention assumes Euclidean geometry. To test whether MRFs provide a useful geometry-aware alternative, we isolate the geometric kernel component in a masked reconstruction task on manifold-valued tokens sampled from a Möbius strip. The reference operator is an intrinsic graph-geodesic heat kernel, and each method replaces this operator by either an exact kernel approximation or a low-rank feature approximation. Figure 4 shows that MRFs obtain the best MSE for every source count and the best relative ℓ_2 error on five of six source counts over other RF methods and the exact Gaussian kernel in ambient Euclidean space (ambient RBF); full results with additional baselines are in Appendix A.6, Table 4. These results suggest MRFs can serve as geometry-aware random features for linearized attention when token embeddings lie on, or are constrained by, a non-Euclidean geometry.

Higher-dimensional descriptor manifolds: To test whether MRFs remain useful beyond embedded two-dimensional surfaces, we evaluate them on a Brodatz transformed-view classification task using 15-dimensional log-SPD covariance descriptors [3, 35]. Following the procedure in the paper, the graph is built on an unlabeled descriptor cloud generated from transformed texture windows, GRF signatures are computed on this graph, and the learned MRF features are used for kNN classification with group-vote evaluation. We compare against using raw log-SPD descriptors (Raw), a Gaussian-kernel PCA embedding (kPCA), and other RF methods. We report group accuracy, where predictions from 50 windows belonging to the same texture-angle block are aggregated by majority vote. As shown in Fig. 4 (and Table 5, Appendix A.7, where there are further experiment details), MRFs achieve the best-tuned accuracy, with mean accuracy (over multiple seeds) of 55.7%, compared with the second highest accuracy of 43.5% achieved by RFF. Beyond these experiments, MRFs could also support geometric computations in robotics and 3D vision, including applications involving Gromov–Wasserstein distances.

Non-compact manifolds: Compactness is natural in many practical use cases—in applied settings, the manifolds of interest (molecular surfaces, mesh surfaces, etc.) are often compact—and the present theory and main experiments focus on compact manifolds. This assumption is used to obtain finite discretizations, bounded learned features, and clean heat-kernel convergence statements. While out of the scope of this work, we provide some insight on how MRFs can be applied to non-compact

manifolds (see Appendix A.8 for details). In particular, we perform a preliminary experiment on hyperbolic 3-space $\mathbb{H}^3$, a canonical non-compact manifold with exponential volume growth, and show that MRFs extend to non-compact manifolds in a *local-on-compacts* sense: when evaluation is restricted to a fixed bounded region, the MRF approximation improves as the finite support used to construct the features is enlarged. This does not constitute a global uniform guarantee on $\mathbb{H}^3$, but suggests a natural exhaustion-based route for extending MRFs beyond the compact setting. Developing the corresponding theory, and testing it across broader classes of non-compact manifolds, is an interesting direction for future work.

4 Conclusion

We have presented a new paradigm of *Manifold Random Features* (MRFs), designed for scalable computations involving bi-variate functions operating on manifolds. MRFs are built with the use of Graph Random Features (GRFs), operating on the discretized manifolds. GRFs are used as teachers for the supervised learning of the continuous fields encoding the MRF-mechanisms. We provide strong empirical validation of MRFs, complemented with rigorous mathematical analysis. As a byproduct of our analysis, we show deep connection between combinatorial concepts of random walks on graphs representing discretized manifolds and continuous kernel functions on these manifolds, in particular by re-discovering recently proposed new class of estimators approximating Gaussian kernels with positive and bounded random features.

Impact Statement & Limitations

This paper presents work whose goal is to advance the field of Machine Learning. There are no immediate societal impacts of our work, as we focus on a methodological framework for approximating functions on manifolds. We do believe that this work might lead to the discovery of new mechanisms of random features operating in continuous spaces and encoded as neural networks that are trained with supervision provided by GRFs. Accuracy depends on sampling density (Table 9), and achieving adequate coverage becomes more costly in higher dimensions. The method also requires the discretized target to be well approximated by the graph-kernel family in Eq. 4. Extending MRFs beyond compact manifolds is left to future work as noted above at the end of Sec. 3.3.

Funding Transparency Statement

We did not receive any third party funding or third party support to help us complete this work. Additionally, to the best of our knowledge, there are no competing interests (financial activities outside the submitted work).

References

[1] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, June 2014. IEEE.

[2] Luca Brandolini, Christine Choirat, Leonardo Colzani, Giacomo Gigante, Raffaello Seri, and Giancarlo Travaglini. Quadrature rules and distribution of points on manifolds, 2010. URL `https://arxiv.org/abs/1012.5409`.

[3] Phil Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover Publications, New York, 1966.

[4] Carlos A. Cadavid, Paulina Hoyos, Jay Jorgenson, Lejla Smajlović, and Juan D. Vélez. On an approach for evaluating certain trigonometric character sums using the discrete time heat kernel, 2022. URL `https://arxiv.org/abs/2201.07878`.

[5] Krzysztof Choromanski and Vikas Sindhwani. Recycling randomness with structure for sublinear time kernel expansions. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New*

York City, NY, USA, June 19-24, 2016, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2502–2510. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/choromanski16.html.

[6] Krzysztof Choromanski, Mark Rowland, Wenyu Chen, and Adrian Weller. Unifying orthogonal monte carlo methods. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1203–1212. PMLR, 2019. URL http://proceedings.mlr.press/v97/choromanski19a.html.

[7] Krzysztof Choromanski, Avinava Dubey, Arijit Sehanobish, and Isaac Reid. Computationally-efficient graph modeling with refined graph random features. *CoRR*, abs/2510.07716, 2025. doi: 10.48550/ARXIV.2510.07716. URL https://doi.org/10.48550/arXiv.2510.07716.

[8] Krzysztof Marcin Choromanski. Taming graph kernels with random features. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 5964–5977. PMLR, 2023. URL https://proceedings.mlr.press/v202/choromanski23a.html.

[9] Krzysztof Marcin Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 219–228, 2017.

[10] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

[11] Krzysztof Marcin Choromanski, Han Lin, Haoxian Chen, Arijit Sehanobish, Yuanzhe Ma, Deepali Jain, Jake Varley, Andy Zeng, Michael S. Ryoo, Valerii Likhosherstov, Dmitry Kalashnikov, Vikas Sindhwani, and Adrian Weller. Hybrid random features. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=EMigfE6ZeS.

[12] Krzysztof Marcin Choromanski, Arijit Sehanobish, Han Lin, Yunfan Zhao, Eli Berger, Tetiana Parshakova, Alvin Pan, David Watkins, Tianyi Zhang, Valerii Likhosherstov, Somnath Basu Roy Chowdhury, Kumar Avinava Dubey, Deepali Jain, Tamas Sarlos, Snigdha Chaturvedi, and Adrian Weller. Efficient graph field integrators meet point clouds. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5978–6004. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/choromanski23b.html.

[13] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003. doi: 10.1002/RSA.10073. URL https://doi.org/10.1002/rsa.10073.

[14] Alexander Grigor'yan. *Heat Kernel and Analysis on Manifolds*, volume 47 of *AMS/IP Studies in Advanced Mathematics*. American Mathematical Society, 2009.

[15] Po-Sen Huang, Haim Avron, Tara N. Sainath, Vikas Sindhwani, and Bhuvana Ramabhadran. Kernel methods match deep neural networks on TIMIT. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 205–209. IEEE, 2014. doi: 10.1109/ICASSP.2014.6853587. URL https://doi.org/10.1109/ICASSP.2014.6853587.

[16] Sang Min Kim, Byeongchan Kim, Arijit Sehanobish, Krzysztof Choromanski, Dongseok Shim, Avinava Dubey, and Min hwan Oh. Magnituder layers for implicit neural representations in 3d, 2024. URL https://arxiv.org/abs/2410.09771.

[17] Quoc V. Le, Tamás Sarlós, and Alexander J. Smola. Fastfood - computing hilbert space expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 244–252. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/le13.html.

[18] Valerii Likhosherstov, Krzysztof Marcin Choromanski, Jared Quincy Davis, Xingyou Song, and Adrian Weller. Sub-linear memory: How to make performers slim. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6707–6719, 2021.

[19] Valerii Likhosherstov, Krzysztof Marcin Choromanski, Kumar Avinava Dubey, Frederick Liu, Tamás Sarlós, and Adrian Weller. Chefs' random tables: Non-trigonometric random features. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[20] Valerii Likhosherstov, Krzysztof Marcin Choromanski, Kumar Avinava Dubey, Frederick Liu, Tamás Sarlós, and Adrian Weller. Dense-exponential random features: Sharp positive estimators of the gaussian kernel. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[21] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan A. K. Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):7128–7148, 2022. doi: 10.1109/TPAMI.2021.3097011. URL https://doi.org/10.1109/TPAMI.2021.3097011.

[22] Duke Nguyen, Aditya Joshi, and Flora D. Salim. Spectraformer: A unified random feature framework for transformer. *CoRR*, abs/2405.15310, 2024. doi: 10.48550/ARXIV.2405.15310. URL https://doi.org/10.48550/arXiv.2405.15310.

[23] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

[24] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. *CoRR*, abs/2010.03409, 2020. URL https://arxiv.org/abs/2010.03409.

[25] Ninh Pham and Rasmus Pagh. Tensor sketch: Fast and scalable polynomial kernel approximation. *CoRR*, abs/2505.08146, 2025. doi: 10.48550/ARXIV.2505.08146. URL https://doi.org/10.48550/arXiv.2505.08146.

[26] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1177–1184. Curran Associates, Inc., 2007.

[27] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21, Proceedings of*

the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008, pages 1313–1320. Curran Associates, Inc., 2008.

[28] Isaac Reid, Eli Berger, Krzysztof Marcin Choromanski, and Adrian Weller. Repelling random walks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=31IOmrnoP4.

[29] Isaac Reid, Krzysztof Marcin Choromanski, Eli Berger, and Adrian Weller. General graph random features. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=viftsX50Rt.

[30] Mark Rowland, Krzysztof Choromanski, François Chalus, Aldo Pacchiano, Tamás Sarlós, Richard E. Turner, and Adrian Weller. Geometrically coupled monte carlo sampling. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 195–205, 2018.

[31] Arijit Sehanobish, Krzysztof Marcin Choromanski, Yunfan Zhao, Kumar Avinava Dubey, and Valerii Likhosherstov. Scalable neural network kernels. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=4iPw1klFWa.

[32] Justin Solomon, Keenan Crane, and Etienne Vouga. Laplace-beltrami: The swiss army knife of geometry processing. Symposium on Geometry Processing, 2014. URL http://ddg.cs.columbia.edu/SGP2014/LaplaceBeltrami.pdf.

[33] Yitong Sun, Anna C. Gilbert, and Ambuj Tewari. But how does it work in theory? linear SVM with random features. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3383–3392, 2018.

[34] Yutao Sun, Zhenyu Li, Yike Zhang, Tengyu Pan, Bowen Dong, Yuyi Guo, and Jianyong Wang. Efficient attention mechanisms for large language models: A survey. *CoRR*, abs/2507.19595, 2025. doi: 10.48550/ARXIV.2507.19595. URL https://doi.org/10.48550/arXiv.2507.19595.

[35] Manik Varma and Andrew Zisserman. Texture classification: Are filter banks necessary? In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 691–698, 2003.

[36] Jiyan Yang, Vikas Sindhwani, Haim Avron, and Michael W. Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 485–493. JMLR.org, 2014. URL http://proceedings.mlr.press/v32/yangb14.html.

[37] Felix X. Yu, Ananda Theertha Suresh, Krzysztof Marcin Choromanski, Daniel N. Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1975–1983, 2016.

[38] Matthew Zhang, Jihao Andreas Lin, Krzysztof Choromanski, Adrian Weller, Richard E. Turner, and Isaac Reid. Graph random features for scalable gaussian processes. *CoRR*, abs/2509.03691, 2025. doi: 10.48550/ARXIV.2509.03691. URL https://doi.org/10.48550/arXiv.2509.03691.

[39] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=4g02l2N2Nx.

[40] Chenchao Zhao and Jun S. Song. Exact heat kernel on a hypersphere and its applications in kernel svm. *Frontiers in Applied Mathematics and Statistics*, 4, January 2018. ISSN 2297-4687. doi: 10.3389/fams.2018.00001. URL http://dx.doi.org/10.3389/fams.2018.00001.

[41] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *CoRR*, abs/1605.04797, 2016. URL http://arxiv.org/abs/1605.04797.

A Appendix

This appendix first reports auxiliary GRF/MRF implementation details and the Gaussian-grid theory supporting the Euclidean limiting case. Then we report supplementary experimental protocols and full results in the same order as the main experimental section for consistency.

A.1 Additional GRF/MRF preliminaries

We collect two auxiliary items used throughout the paper: an illustration of manifold heat-kernel fields and the regular GRF random-walk estimator.

A.1.1 Heat-kernel illustration

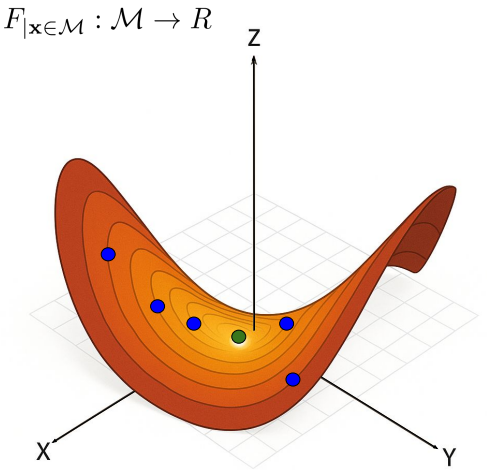

Figure 5: An illustration of the heat kernel $F : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ defined on the saddle-like surface $\mathcal{M}$ with a distinguished point $\mathbf{x} \in \mathcal{M}$ marked green. The values $F(\mathbf{x}, \mathbf{y})$ for different $\mathbf{y} \in \mathcal{M}$ are color-coded (different shades of red), with a couple of samples $\mathbf{y}$ highlighted as blue dots. For a general manifold, F is given by a system of partial differential equations and neither it nor the corresponding RF-mechanism are given by the closed-form expressions.

A.1.2 Regular GRF random-walk construction

Algorithm 1 **Regular GRFs:** Construct vectors $\phi_f(i) \in \mathbb{R}^N$ to approximate $\mathbf{K}_{\boldsymbol{\alpha}}(\mathbf{W})$

Input: weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$, vector of unweighted node degrees (number of out-neighbours) $\deg \in \mathbb{R}^N$, modulation function $f : (\mathbb{N} \cup \{0\}) \to \mathbb{R}$, termination probability $p_{\text{halt}} \in (0, 1)$, node $i \in \mathcal{N}$, number of random walks to sample $n_{\text{walk}} \in \mathbb{N}$.
Output: signature vector $\phi_f(i) \in \mathbb{R}^N$.

initialize: $\phi_f(i) \leftarrow \mathbf{0}$
for $w = 1, ..., n_{\text{walk}}$ **do**
  initialize: `load` $\leftarrow 1$, `current_node` $\leftarrow i$, `terminated` $\leftarrow$ `False`, `walk_length` $\leftarrow 0$
  while `terminated` $=$ `False` **do**
    $\phi_f(i)$[`current_node`] $\leftarrow \phi_f(i)$[`current_node`] + `load` $\times f($ `walk_length` $)$
    `walk_length` $\leftarrow$ `walk_length` $+ 1$
    `new_node` $\leftarrow$ `Unif` $[\mathcal{N}($ `current_node` $)]$ ▷ assign to one of neighbours
    `load` $\leftarrow$ `load` $\times \frac{\deg[\texttt{current_node}]}{1-p_{\text{halt}}} \times \mathbf{W}[$ `current_node`, `new_node` $]$ ▷ update load
    `current_node` $\leftarrow$ `new_node`
    `terminated` $\leftarrow (t \sim \text{Unif}(0, 1) < p_{\text{halt}})$ ▷ draw RV t to decide on termination
  end while
end for
normalize: $\phi_f(i) \leftarrow \phi_f(i)/n_{\text{walk}}$

A.2 Kernel reconstruction and scale alignment

In order to address the scaling challenge, as we go from the original continuous space to the discretized space, we conduct scale alignment that can be thought of as extra normalization. In matrix form, letting $\mathbf{G}_\theta \in \mathbb{R}^{N\times N}$ with $(\mathbf{G}_\theta)_{ij} = g_\theta(\mathbf{x}_i, \boldsymbol{\omega}_j)$, the induced kernel is $\mathbf{K}_\theta = \mathbf{G}_\theta\,\mathbf{G}_\theta^\top$. To compare against a ground-truth kernel matrix $\mathbf{K}_{\mathrm{GT}}$ (see Sec. 2.2.1 for how these values are computed), we apply a single global scaling that matches Frobenius norms:

$$\widetilde{\mathbf{K}}_\theta \;=\; \alpha\mathbf{K}_\theta, \qquad \alpha \;=\; \frac{\|\mathbf{K}_{\mathrm{GT}}\|_F}{\|\mathbf{K}_\theta\|_F}. \tag{16}$$

All kernel errors and visualizations use $\widetilde{\mathbf{K}}_\theta$.

A.3 Gaussian-grid limit: from graph diffusion to continuous features

This section consolidates the discrete-to-continuous argument behind the Gaussian-kernel sanity check in Sec. 2.3. We first show convergence of the rescaled grid Laplacian and its diffusion kernel, then recall the continuous positive Gaussian feature factorization, and finally give empirical validation on finite grids.

A.3.1 Diffusion kernel on the discrete Euclidean grid

Let $d \in \mathbb{N}$ and, for each $n \in \mathbb{N}$, let $h_n := \frac{1}{n}$. Let $\mathbb{T}^d$ be the d–dimensional torus and let $\mathrm{V}_n \subset \mathbb{T}^d$ be the regular d–dimensional grid

$$\mathrm{V}_n := \Big\{x = (k_1 h_n, \ldots, k_d h_n) : k_j \in \{0, \ldots, n-1\}\Big\}.$$

Consider the graph whose vertices are V_n and whose neighbors are the $2d$ wrap–around nearest neighbors.

Theorem A.1 (Convergence of Discrete to Continuous Laplacian)**.** *Define the averaging operator* $\mathbf{T}_n : \mathbb{R}^{\mathrm{V}_n} \to \mathbb{R}^{\mathrm{V}_n}$ *by*

$$(\mathbf{T}_n f)(x) := \frac{1}{2d}\sum_{y\sim x} f(y),$$

where the sum is over the $2d$ *neighbors* y *of* x *in the wrap-around grid. Furthermore, denote the (random–walk) graph and rescaled Laplacian as*

$$\mathbf{L}_{\mathrm{curr}}^{(n)} := \mathbf{I} - \mathbf{T}_n \qquad \mathbf{L}_n := \frac{2d}{h_n^2}\,\mathbf{L}_{\mathrm{curr}}^{(n)} = \frac{2d}{h_n^2}\,(\mathbf{I} - \mathbf{T}_n).$$

Let Δ *be the usual Laplacian on* $\mathbb{T}^d$*. Then we have*

$$\mathbf{L}_n \xrightarrow[n\to\infty]{} -\Delta. \tag{17}$$

Proof. For $f \in C^4(\mathbb{T}^d)$ define its restriction to the grid V_n by $f_n(x) := f(x)$, $x \in \mathrm{V}_n$. Fix $x \in \mathbb{T}^d$ and $f \in C^4(\mathbb{T}^d)$. For each coordinate direction $\mathbf{e}_k$ ($k = 1, \ldots, d$), we have the Taylor expansions

$$f(x \pm h_n\mathbf{e}_k) = f(x) \pm h_n\partial_k f(x) + \frac{h_n^2}{2}\,\partial_{kk} f(x) \pm \frac{h_n^3}{6}\,\partial_{kkk} f(\xi_\pm) + \frac{h_n^4}{24}\,\partial_{kkkk} f(\eta_\pm),$$

for some points $\xi_\pm, \eta_\pm$ on the segment between x and $x \pm h_n\mathbf{e}_k$. Adding these two expansions gives

$$f(x + h_n\mathbf{e}_k) + f(x - h_n\mathbf{e}_k) = 2f(x) + h_n^2\partial_{kk} f(x) + O(h_n^4).$$

Summing over $k = 1, \ldots, d$,

$$\begin{aligned}
\sum_{y\sim x} f(y) &= \sum_{k=1}^{d}\big(f(x + h_n\mathbf{e}_k) + f(x - h_n\mathbf{e}_k)\big) \\
&= \sum_{k=1}^{d}\big(2f(x) + h_n^2\partial_{kk} f(x) + O(h_n^4)\big) \\
&= 2d\,f(x) + h_n^2\sum_{k=1}^{d}\partial_{kk} f(x) + O(dh_n^4).
\end{aligned}$$

Furthermore, we know

$$\begin{aligned}(\mathbf{L}_{\mathrm{curr}}^{(n)} f_n)(x) &= f(x) - \frac{1}{2d}\sum_{y\sim x} f(y) \\ &= -\frac{h_n^2}{2d}\sum_{k=1}^{d} \partial_{kk} f(x) + O(h_n^4) \\ &= -\frac{h_n^2}{2d}\,\Delta f(x) + O(h_n^4),\end{aligned}$$

where we plugged in our earlier result in the second line. Multiplying by $\frac{2d}{h_n^2}$ gives

$$(\mathbf{L}_n f_n)(x) = \frac{2d}{h_n^2}(\mathbf{L}_{\mathrm{curr}}^{(n)} f_n)(x) = -\,\Delta f(x) + O(h_n^2),$$

for any $x \in \mathrm{V}_n$. Thus, for each smooth f,

$$\mathbf{L}_n f_n \;\xrightarrow[n\to\infty]{}\; -\Delta f.$$

□

Theorem A.2 (Convergence of Discrete Diffusion Kernel to Periodized Gaussian)**.** *Fix $\sigma > 0$ and let $f \in C^4(\mathbb{T}^d)$. Then we have*

$$\exp\Big(-\tfrac{\sigma^2}{2}\mathbf{L}_n\Big) f\big|_{\mathrm{V}_n} \;\xrightarrow[n\to\infty]{}\; \Big(\exp\big(\tfrac{\sigma^2}{2}\Delta\big) f\Big)\big|_{\mathrm{V}_n}.$$

Moreover, the limiting operator $\exp(\frac{\sigma^2}{2}\Delta)$ acts by convolution with a periodized Gaussian kernel

$$k_\sigma(x,y) = \sum_{k\in\mathbb{Z}^d} \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\Big(-\frac{\|x-y+k\|^2}{2\sigma^2}\Big), \qquad x, y \in \mathbb{T}^d.$$

Proof. Using the result of Theorem A.1 and exponentiating the operators implies that for any fixed $t \geq 0$,

$$e^{-t\mathbf{L}_n} f\big|_{\mathrm{V}_n} \;\longrightarrow\; \big(e^{t\Delta} f\big)\big|_{\mathrm{V}_n}.$$

Taking $t = \sigma^2/2$ yields the stated convergence.

The operator $e^{t\Delta}$ on the torus $\mathbb{T}^d$ is the heat semigroup and admits an integral representation

$$(e^{t\Delta} f)(x) = \int_{\mathbb{T}^d} p_t(x,y)\, f(y)\, dy,$$

where p_t is the heat kernel on the torus. This kernel is obtained by periodizing the Euclidean heat kernel [2]

$$p_t(x,y) = \sum_{k\in\mathbb{Z}^d} \frac{1}{(4\pi t)^{d/2}} \exp\Big(-\frac{\|x-y+k\|^2}{4t}\Big).$$

Setting $t = \sigma^2/2$ gives

$$p_{\sigma^2/2}(x,y) = \sum_{k\in\mathbb{Z}^d} \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\Big(-\frac{\|x-y+k\|^2}{2\sigma^2}\Big),$$

which is a Gaussian kernel with variance σ^2 in each coordinate, periodized to respect the torus geometry. □

Remark: If instead of the discrete grid, we worked on $\mathbb{R}^d$ (no wrap–around), the corresponding kernel would be the standard Gaussian

$$\mathrm{K}_\sigma(\mathbf{x},\mathbf{y}) := \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\Big(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\Big),$$

which is the usual Gaussian kernel with bandwidth σ. On the torus we obtain its periodic version by summing over $k \in \mathbb{Z}^d$.

A.3.2 Euclidean Gaussian feature representation

Theorem A.3 (RBF Kernel Factorization)**.** *Let $d \in \mathbb{N}$ and $\sigma > 0$, and define*

$$g_\sigma(\mathbf{x}, \boldsymbol{\omega}) := \left(\frac{2}{\pi\sigma^2}\right)^{d/4} \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\omega}\|^2}{\sigma^2}\right), \qquad \mathbf{x}, \boldsymbol{\omega} \in \mathbb{R}^d.$$

Then for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$\int_{\mathbb{R}^d} g_\sigma(\mathbf{x}, \boldsymbol{\omega})\, g_\sigma(\mathbf{y}, \boldsymbol{\omega})\, d\boldsymbol{\omega} = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right).$$

In particular, g_σ is a continuous feature map for the Gaussian kernel with bandwidth σ.

Proof. Set $C_\sigma := \left(\frac{2}{\pi\sigma^2}\right)^{d/4}$ and $\mathbf{m} := \frac{\mathbf{x}+\mathbf{y}}{2}$ for fixed $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. By Lemma A.4 we have

$$\|\mathbf{x}-\boldsymbol{\omega}\|^2 + \|\mathbf{y}-\boldsymbol{\omega}\|^2 = 2\|\boldsymbol{\omega}-\mathbf{m}\|^2 + \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2.$$

Therefore,

$$\begin{aligned}
\int_{\mathbb{R}^d} g_\sigma(\mathbf{x}, \boldsymbol{\omega})\, g_\sigma(\mathbf{y}, \boldsymbol{\omega})\, d\boldsymbol{\omega} &= C_\sigma^2 \int_{\mathbb{R}^d} \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\omega}\|^2}{\sigma^2} - \frac{\|\mathbf{y}-\boldsymbol{\omega}\|^2}{\sigma^2}\right) d\boldsymbol{\omega} \\
&= C_\sigma^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right) \int_{\mathbb{R}^d} \exp\left(-\frac{2\|\boldsymbol{\omega}-\mathbf{m}\|^2}{\sigma^2}\right) d\boldsymbol{\omega} \\
&= C_\sigma^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right) \left(\frac{\pi\sigma^2}{2}\right)^{d/2} \\
&= \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right),
\end{aligned}$$

noting that for $a > 0$, $\int_{\mathbb{R}} e^{-au^2}\, du = \sqrt{\frac{\pi}{a}}$. □

Lemma A.4. *For any $\mathbf{x}, \mathbf{y}$ in $\mathbb{R}^d$ let $\mathbf{m} := \frac{\mathbf{x}+\mathbf{y}}{2}$, then we have*

$$\|\mathbf{x}-\boldsymbol{\omega}\|^2 + \|\mathbf{y}-\boldsymbol{\omega}\|^2 = 2\|\boldsymbol{\omega}-\mathbf{m}\|^2 + \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2, \qquad \boldsymbol{\omega} \in \mathbb{R}^d. \tag{18}$$

Proof. We have

$$\begin{aligned}
\|\mathbf{x}-\boldsymbol{\omega}\|^2 + \|\mathbf{y}-\boldsymbol{\omega}\|^2 &= (\|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \boldsymbol{\omega}\rangle + \|\boldsymbol{\omega}\|^2) + (\|\mathbf{y}\|^2 - 2\langle \mathbf{y}, \boldsymbol{\omega}\rangle + \|\boldsymbol{\omega}\|^2) \\
&= 2\|\boldsymbol{\omega}\|^2 - 2\langle \mathbf{x}+\mathbf{y}, \boldsymbol{\omega}\rangle + \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 \\
&= 2\left(\|\boldsymbol{\omega}\|^2 - \left\langle \mathbf{x}+\mathbf{y}, \boldsymbol{\omega}\right\rangle + \left\|\frac{\mathbf{x}+\mathbf{y}}{2}\right\|^2\right) + \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \tfrac{1}{2}\|\mathbf{x}+\mathbf{y}\|^2 \\
&= 2\|\boldsymbol{\omega}-\mathbf{m}\|^2 + \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2.
\end{aligned}$$

□

A.3.3 Empirical Gaussian-grid validation

Following the discussion in Sec. 2.3, for each dimension $d \in \{2, 4, ..., 32\}$, we discretize the hypercube $[0,1]^d$ via the regular grid V_n with spacing $h_n = 1/n$ and n^d nodes, connecting each node to its $2d$ wrap–around nearest neighbors. For each choice of n and d, we construct signature vectors $\phi_f(x)$ at every grid node $x \in \mathrm{V}_n$ using the random walk procedure of Algorithm 1 with a fixed diffusion scale $\sigma = 0.2$, termination probability $p_{\text{halt}} = 0.005$, $n_{\text{walk}} = 100{,}000$ walks per node, and vary n from 5 to 105 in steps of 10. We then form rescaled vectors $\psi_f(x) = c_{d,\sigma,n}\, \phi_f(x)$.

Note: On the (d-dimensional grid) V_n, the random-walk Laplacian decomposes as a Kronecker sum of 1D operators. Hence, the corresponding terms $e^{-tL/2}$ and the induced kernel e^{-tL} factorize across coordinates; see Prop. 20 [4] for further details. We exploit this to compute the 1D quantities and tensorize to obtain the d-dimensional results.

We compute the relative mean squared error (MSE) of approximating $g_\sigma(\mathbf{c}, *)$ with $\psi_f(c)[*]$, as well as that of approximating true kernel values with those induced by MRFs (in the grid points). The latter are given as:

$$\widehat{\mathrm{K}}^{\mathrm{Gauss}}_{\sigma,n}(x, y) := \big\langle \psi_f(x), \psi_f(y) \big\rangle, \qquad x, y \in \mathrm{V}_n. \tag{19}$$

Across all tested dimensions, we observe that errors decrease rapidly as the grid is refined (see: Fig. 6).

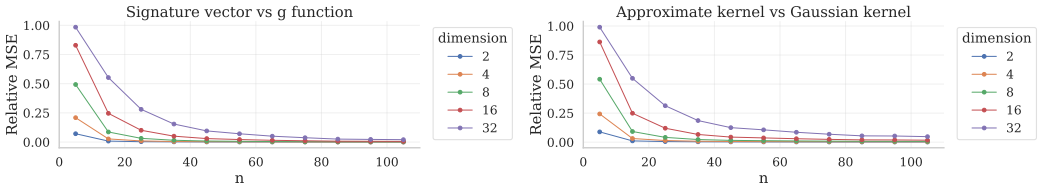

Figure 6: Empirical relative mean squared errors (MSEs) of the approximations of g-functions and true Gaussian kernel values with signature vectors and kernels induced by them for $d \in \{2, 4, ..., 32\}$ and $n = 5, 15, ..., 105$. Relative MSE of the approximation of vector $\mathbf{y}$ with random vector $\mathbf{x}$ is defined as $\mathbb{E}[||\mathbf{x} - \mathbf{y}||_2^2 / ||\mathbf{y}||_2^2]$. We use $s = 30$ repetitions for each n value (standard deviations not visible; negligible relative to means).

A.3.4 Convergence visualization for MRF-induced Gaussian kernels

We use the Gaussian-grid setting above. For a representative grid point c (we choose the center $c = (\frac{1}{2}, \ldots, \frac{1}{2})$), we visualize the field $\omega \mapsto \psi_n(c)[\omega]$ over V_n and compare it to the analytic feature $g_\sigma(c, \omega)$ evaluated on the same grid. As n increases, the discrete field becomes increasingly smooth and radially symmetric, approaching the shape of the Gaussian bump predicted by the theory (see Fig. 7 below for the two–dimensional case).

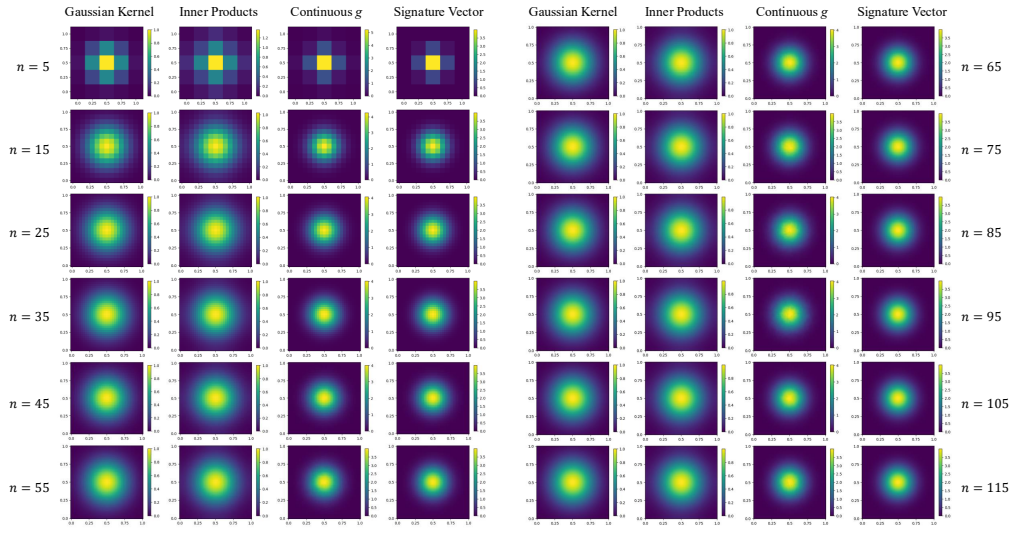

Figure 7: **First column:** Exact values of the Gaussian kernel, computed in the grid points and broadcast to all the points in the continuous rectangle corresponding to that grid point. **Second column:** Inner(dot)-products of the signature vectors, leading to the approximate Gaussian kernel values. **Third column:** The values of the exact function $g_\sigma(c, \omega)$ (for a fixed c), computed in the grid points and broadcast to all the points in the continuous rectangle corresponding to that grid point. **Fourth column:** Values of the renormalized signature vector $\psi(c)[\omega]$ for a fixed c. We use the following setting: $d = 2$, $\sigma = 0.2$, $p_{\mathrm{halt}} = 0.005$, and $n_{\mathrm{walk}} = 100{,}000$.

A.4 Surface heat-kernel reconstruction experiments

This section gives the mathematical ground truth and full experimental details for the 2D embedded-surface experiments, followed by the ambient Euclidean RF baseline sweep and validation diagnostics.

A.4.1 Sphere ground-truth heat kernel

For the unit hypersphere $S^d \subset \mathbb{R}^{d+1}$, the heat kernel admits an explicit spectral expansion in terms of Gegenbauer polynomials. Following Theorem 1 of Zhao and Song [40], the exact hyperspherical heat kernel $G^{\text{ext}}(\mathbf{x}, \mathbf{y}; t)$ on S^{n-1} (with $n = d + 1$) can be written as the uniformly and absolutely convergent series

$$G^{\text{ext}}(\mathbf{x}, \mathbf{y}; t) \;=\; \sum_{\ell=0}^{\infty} e^{-\ell(\ell+n-2)t}\, \frac{2\ell + n - 2}{n-2}\, \frac{1}{A_{S^{n-1}}}\, C_\ell^{\frac{n-2}{2}}(\mathbf{x} \cdot \mathbf{y}), \tag{20}$$

where C_ℓ^α denotes the Gegenbauer polynomial of order ℓ and index α, $\mathbf{x} \cdot \mathbf{y} \in [-1, 1]$, $t > 0$, and

$$A_{S^{n-1}} \;=\; \frac{2\pi^{\frac{n}{2}}}{\Gamma\big(\frac{n}{2}\big)}$$

is the surface area of S^{n-1}.

In the case $d = 2$ (so $n = 3$ and we are on S^2), we have $\alpha = \frac{n-2}{2} = \frac{1}{2}$, and the Gegenbauer polynomials reduce to Legendre polynomials via $C_\ell^{1/2}(z) = P_\ell(z)$. Therefore (20) specializes to the classical spherical expansion

$$G^{\text{ext}}(\mathbf{x}, \mathbf{y}; t) \;=\; \sum_{\ell=0}^{\infty} \frac{2\ell + 1}{4\pi} e^{-\ell(\ell+1)t}\, P_\ell(\langle \mathbf{x}, \mathbf{y} \rangle). \tag{21}$$

For numerical ground truth, we truncate (21) at $\ell \le L_{\max}$:

$$\mathrm{K}_t^{\text{true}}(\mathbf{x}_i, \mathbf{x}_j) \;\approx\; \sum_{\ell=0}^{L_{\max}} \frac{2\ell + 1}{4\pi}\, e^{-\ell(\ell+1)t}\, P_\ell(\langle \mathbf{x}_i, \mathbf{x}_j \rangle),$$

where $L_{\max}$ is chosen so that the tail mass is negligible. In our experiments, we choose $L_{\max} = 50$.

A.4.2 Common setup and surface-specific details

For each geometry, we discretize the surface into $N = 4000$ points, build a kNN graph (with $k = 24$ for the Möbius strip and $k = 8$ for the other three), generate signature feature vectors from 1000 (sampled) start points with parameters: $n_{\text{walk}} = 100,000$ (number of random walks), $p_{\text{halt}} = 0.01$, and $\sigma^2 = 20$. We then train a neural network predictor $g_\theta(\mathbf{x}, \omega)$ to learn $\phi_f(x)[\omega]$. Our network architecture is a small multilayer perceptron that takes as input the coordinates of the pair $(\mathbf{x}, \omega)$ along with an approximation of the geodesic distance $d_{\mathcal{M}}(\mathbf{x}, \omega)$, (computed via a shortest path calculation between x and ω), and outputs the signature-vector values. Training was run for 1000 epochs with an absolute relative error loss (clamped at $\varepsilon = 0.1$), defined as follows:

$$\mathcal{L}(g_\theta(\mathbf{x}, \omega), \phi_f(x)(\omega)) = \frac{|g_\theta(\mathbf{x}, \omega) - \phi_f(x)(\omega)|}{\max\{\phi_f(x)(\omega), \varepsilon\}}, \tag{22}$$

and optimized with Adam. We report root mean square error (RMSE), validation diagnostics (prediction vs. actual and error vs. truth; those in the Appendix: Sec. A.4.4), qualitative 3D visualizations of $g_\theta(\mathbf{x}, \cdot)$, $\phi_f(x)[\cdot]$, and their difference for representative validation starts, and heat-kernel comparisons using the kernel induced by the learned features. From the learned feature matrix $\mathbf{G}_\theta$ we form $\mathbf{K}_\theta = \mathbf{G}_\theta \mathbf{G}_\theta^\top$, and compare it against the ground-truth kernel matrix $\mathbf{K}_{\text{GT}}$. Before reporting kernel errors, we apply the Frobenius-norm alignment (see: Appendix, Sec. A.2) and evaluate the renormalized version $\widetilde{\mathbf{K}}_\theta$.

Sphere: We embed a sphere $x^2 + y^2 + z^2 = 1$ in $\mathbb{R}^3$. For the ground truth, we use the analytic spherical heat kernel (see: Sec. A.4.1) truncated at $L_{\max} = 50$ spherical harmonics. The diffusion-time used in the analytic kernel is set to $t_{\text{analytical}} \approx 0.25$ to match the diffusion length observed on the discretized graph (and account for the number of nodes N).

Ellipsoid: We embed an ellipsoid $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$ in $\mathbb{R}^3$ (with $a = 1.0, b = 1.3, c = 0.7$) and discretize it via scaled Fibonacci construction. As a reference kernel, we use the graph heat kernel $\mathrm{K}_{\mathrm{true}} = \exp(-t\mathbf{L})$ computed from a kNN graph Laplacian on the ellipsoid.

Möbius strip: We embed a Möbius strip in $\mathbb{R}^3$ and discretize it using a uniform grid in the intrinsic parameters (u, v). The ground truth is constructed similarly as for the ellipsoid. The Möbius strip is in principle more sensitive to discretization and graph construction (e.g., near-duplicate points, irregular sampling density, or disconnected kNN graphs can destabilize training).

A.4.3 Additional Results: Torus

We embed a torus in $\mathbb{R}^3$ and discretize it using a uniform grid. We then proceed as for the sphere and ellipsoid case. The conclusions are the same as for the sphere and the ellipsoid (see: Fig. 8).

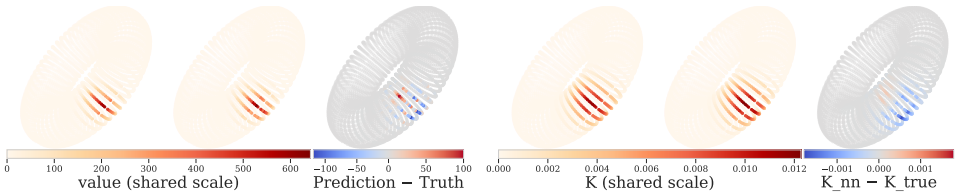

Figure 8: Qualitative field and heat kernel comparison for a torus. Example validation start (left): predicted $g_\theta(\mathbf{x}, \cdot)$, ground truth $\phi_f(\mathbf{x})[\cdot]$, and residual on the torus. Reference graph heat kernel (right) $\mathrm{K}^{\mathrm{heat}}(\mathbf{x}, \cdot)$ compared to Frobenius-aligned induced kernel $\widetilde{\mathrm{K}}_\theta(\mathbf{x}, \cdot) = \widetilde{\mathrm{K}}_{\mathrm{NN}}(\mathbf{x}, \cdot)$, and their difference, for a representative start point.

A.4.4 Validation Diagnostics for All Shapes

Validation diagnostics results for all the shapes, confirming strong performance of the MRF method, are presented in Fig. 9, Fig. 10, Fig. 11 and Fig. 12.

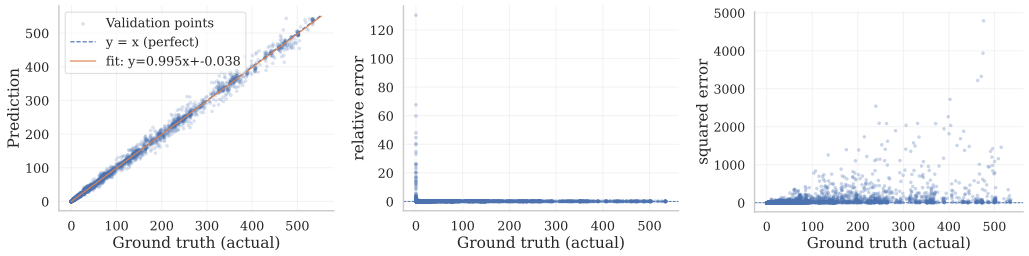

Figure 9: **Sphere validation diagnostics.** Left: predicted vs. ground-truth φ_t values. Middle/right: relative error and squared error as a function of ground-truth magnitude, highlighting where the model under/over-estimates.

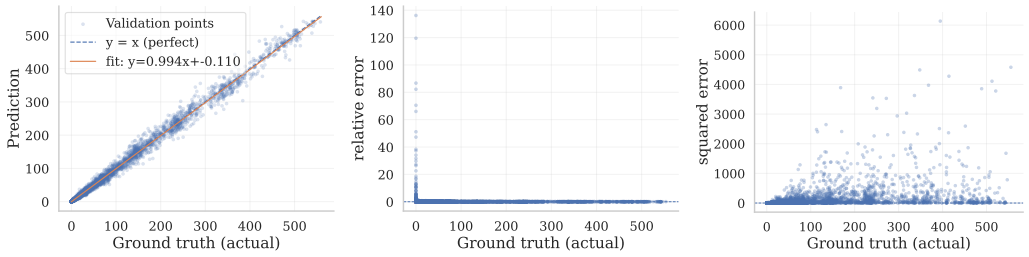

Figure 10: **Ellipsoid validation diagnostics.** Left: predicted vs. ground-truth φ_t values. Middle/right: relative and squared errors as a function of ground-truth magnitude.

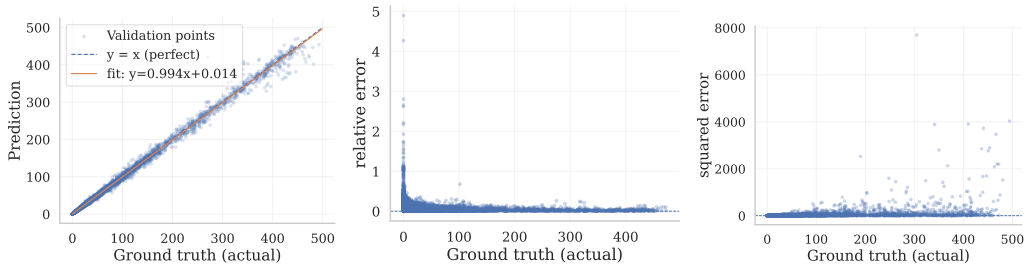

Figure 11: **Möbius validation diagnostics.** Predicted vs. ground-truth φ_t values (left) and error vs. truth (middle/right), illustrating calibration and where errors concentrate.

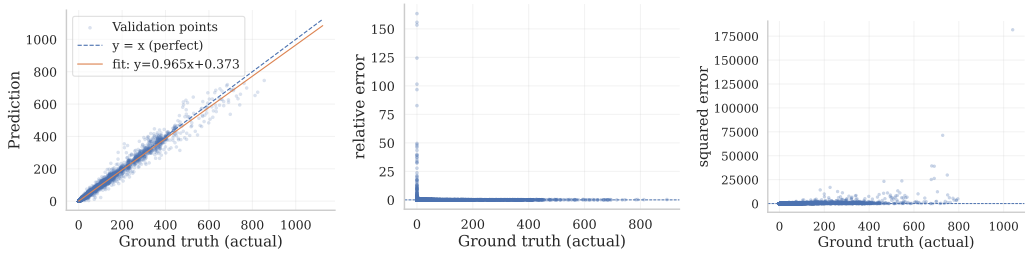

Figure 12: **Torus validation diagnostics.** Predicted vs. ground-truth φ_t values (left) and error vs. truth (middle/right), illustrating calibration and where errors concentrate.

A.4.5 Ambient-space Euclidean random-feature baselines

We evaluate all methods on kernel reconstruction against the ground-truth manifold heat kernel, reporting R^2, RMSE, MAE, relative error (mean and median), and relative Frobenius norm.

Table 3 expands the compact main-text comparison in Table 1 by reporting all ambient RF methods and all tested feature budgets. The downstream RF comparison is reported separately in Table 2. Clearly, MRFs outperform all Euclidean baselines across every surface and every metric, even when the baselines are allowed up to $4\times$ more features.

A.5 Mesh interpolation experimental details

A.5.1 Vertex normal prediction setup

We consider vertex normal interpolation on a triangular mesh with vertex set V, faces F, and vertex positions $\{\mathbf{x}_i \in \mathbb{R}^3\}_{i\in\mathrm{V}}$. Ground-truth vertex normals $\{\mathbf{n}_i \in \mathbb{R}^3\}_{i\in\mathrm{V}}$ are computed by accumulating incident (unnormalized) face normals and normalizing each vertex vector. We randomly mask a subset $M \subseteq \mathrm{V}$ with $|M| = 0.8|\mathrm{V}|$ (80% missing), set $\widetilde{\mathbf{n}}_i = \mathbf{0}$ for $i \in M$ and $\widetilde{\mathbf{n}}_i = \mathbf{n}_i$ otherwise, and predict normals at the masked vertices via kernel-based field integration. Performance is reported as the mean cosine similarity on the masked set, $\frac{1}{|M|}\sum_{i\in M}(\mathbf{n}_i^{\mathrm{pred}})^\top \mathbf{n}_i$, where $\mathbf{n}_i^{\mathrm{pred}}$ is the unit-normalized prediction. The mesh graph $G = (\mathrm{V}, E)$ uses the mesh edges induced by faces. For each edge $(i, j) \in E$, we define an affinity

$$(\mathbf{W})_{ij} = \exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\Big), \qquad \sigma^2 = \mathrm{median}_{(i,j)\in E}\|\mathbf{x}_i - \mathbf{x}_j\|^2,$$

compute degrees $(\mathbf{D})_{ii} = \sum_j (\mathbf{W})_{ij}$, and form the symmetrically-normalized matrix $\mathbf{W}_f = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. The full-kernel (FK) baseline explicitly constructs the dense kernel $\mathbf{K} = \exp(\tau\mathbf{W}_f)$, with $\tau = 20$, by eigen-decomposition of the dense $\mathbf{W}_f$ (double precision) and then applies $\mathbf{N}^{\mathrm{pred}} = \mathbf{K}\widetilde{\mathbf{N}}$ with $\widetilde{\mathbf{N}} \in \mathbb{R}^{|\mathrm{V}|\times 3}$ the masked normal field (masked rows set to $\mathbf{0}$), followed by row-wise unit normalization. We record FK preprocessing time as dense $\mathbf{W}_f$ construction plus eigendecomposition and kernel formation, and FK interpolation time as the matrix–field product $\mathbf{K}\widetilde{\mathbf{N}}$ plus normalization.

Table 3: Kernel reconstruction accuracy: MRFs vs. ambient-space Euclidean random features with tuned σ. MRF vectors have dimension 4000; baselines are evaluated at 4000, 8000, and 16000 features. RMSE and MAE in units of 10^{-4}. Best results **bolded**.

Surface	Method	# features	$R^2 \uparrow$	RMSE $\downarrow$	MAE $\downarrow$	Mean RE $\downarrow$	Median RE $\downarrow$	Rel. Frob. $\downarrow$
Sphere	**MRF**	–	**0.995**	**242.71**	**59.11**	**0.039**	**0.005**	**0.067**
	RFF	4000	0.975	556.09	433.66	33.418	26.056	0.153
	ORF		0.976	553.63	419.85	29.771	23.221	0.153
	PRF		0.147	3266.96	1647.41	89.222	78.716	0.901
	PORF		0.257	3048.50	935.74	20.537	4.146	0.841
	RFF	8000	0.985	432.81	323.86	23.246	18.064	0.119
	ORF		0.983	463.96	331.72	21.359	16.734	0.128
	PRF		0.235	3094.36	890.47	16.485	3.055	0.853
	PORF		0.351	2850.61	1019.82	28.066	6.216	0.786
	RFF	16000	0.989	367.93	253.48	15.913	12.253	0.101
	ORF		0.987	403.97	270.26	15.857	12.390	0.111
	PRF		0.391	2759.94	1027.42	28.759	6.300	0.761
	PORF		0.269	3023.77	931.91	20.888	4.484	0.834
Ellipsoid	**MRF**	–	**0.998**	**0.56**	**0.10**	**0.004**	**0.000**	**0.047**
	RFF	4000	0.904	3.63	1.93	0.159	0.109	0.304
	ORF		0.898	3.74	1.94	0.158	0.102	0.313
	PRF		0.144	10.81	3.05	0.161	0.040	0.905
	PORF		0.247	10.14	3.50	0.224	0.068	0.849
	RFF	8000	0.909	3.52	1.64	0.130	0.076	0.295
	ORF		0.902	3.66	1.70	0.135	0.074	0.306
	PRF		0.305	9.74	3.70	0.253	0.090	0.815
	PORF		0.304	9.75	3.80	0.265	0.092	0.816
	RFF	16000	0.911	3.48	1.46	0.112	0.054	0.291
	ORF		0.906	3.58	1.53	0.118	0.056	0.300
	PRF		0.326	9.59	3.88	0.276	0.099	0.803
	PORF		0.327	9.58	3.94	0.283	0.107	0.802
Möbius strip	**MRF**	–	**0.999**	**0.39**	**0.06**	**0.002**	**0.000**	**0.035**
	RFF	4000	0.738	5.67	2.08	0.140	0.093	0.499
	ORF		0.738	5.67	2.08	0.141	0.094	0.499
	PRF		0.120	10.39	3.06	0.172	0.026	0.915
	PORF		0.154	10.19	3.21	0.192	0.031	0.898
	RFF	8000	0.745	5.60	1.83	0.116	0.068	0.493
	ORF		0.744	5.60	1.80	0.113	0.065	0.493
	PRF		0.203	9.89	3.51	0.233	0.051	0.871
	PORF		0.192	9.96	3.43	0.221	0.043	0.877
	RFF	16000	0.748	5.56	1.65	0.098	0.049	0.489
	ORF		0.748	5.56	1.63	0.096	0.047	0.490
	PRF		0.216	9.81	3.55	0.239	0.051	0.864
	PORF		0.212	9.83	3.58	0.241	0.053	0.866
Torus	**MRF**	–	**0.996**	**0.73**	**0.12**	**0.005**	**0.000**	**0.062**
	RFF	4000	0.817	4.96	2.25	0.180	0.106	0.418
	ORF		0.803	5.16	2.35	0.188	0.103	0.434
	PRF		0.138	10.77	3.30	0.190	0.057	0.908
	PORF		0.189	10.45	3.66	0.237	0.076	0.881
	RFF	8000	0.821	4.91	2.00	0.155	0.072	0.414
	ORF		0.806	5.11	2.14	0.167	0.073	0.430
	PRF		0.221	10.24	3.97	0.276	0.107	0.863
	PORF		0.220	10.24	3.92	0.269	0.098	0.863
	RFF	16000	0.818	4.95	1.91	0.146	0.052	0.417
	ORF		0.812	5.02	1.98	0.152	0.054	0.423
	PRF		0.232	10.17	4.02	0.281	0.108	0.857
	PORF		0.222	10.23	4.01	0.279	0.109	0.862

MRFs target the same kernel $\mathbf{K} = \exp(\tau \mathbf{W}_f)$ but avoid explicit construction by learning a low-rank factorization. We first estimate signature values corresponding to $\exp(\frac{\tau}{2}\mathbf{W}_f)$ using the g-GRF random-walk estimator with modulation coefficients $\alpha_k = (\frac{\tau}{2})^k/k!$ (truncated when $\alpha_k < 10^{-300}$), halting probability $p_{\text{halt}} = 0.01$, and $n_{\text{walk}} = 10{,}000$ walks per start node. We sample 1000 start nodes uniformly without replacement (restricted to vertices when a denser discretization is used). For each start, we sample candidate anchors ω uniformly, keep all samples with target value at least 0.1, and retain smaller targets with probability 0.025. We also train a continuous surrogate $g_\theta(\mathbf{x}_{\text{start}}, \mathbf{x}_\omega)$ using a 3-layer MLP with two 128-wide ReLU hidden layers. Optimization uses Adam with learning rate 10^{-3}, batch size $32{,}768$, 1000 epochs, a relative-error loss with $\varepsilon = 0.1$, and a 20% validation split.

For training g_θ we use a denser point set on the surface: by default we set $N_{\text{dense}} = \max(|\mathrm{V}|, 5000)$ by adding area-weighted samples on faces and build a kNN graph on these points with $k = 16$. Inference remains on the original vertices: we sample $n_{\text{rf}} = 256$ anchors $\{\omega_\ell\}$ uniformly from the dense set and approximate $\mathbf{K}$ via $\mathbf{K} \approx \mathbf{\Phi}\mathbf{\Phi}^\top$ with $(\mathbf{\Phi})_{i\ell} = g_\theta(\mathbf{x}_i, \mathbf{x}_{\omega_\ell}, d(i, \omega_\ell))/\sqrt{n_{\text{rf}}}$, applying $\mathbf{N}^{\text{pred}} \approx \mathbf{\Phi}(\mathbf{\Phi}^\top \widetilde{\mathbf{N}})$ and normalizing row-wise. MRF preprocessing time includes supervision generation and training, and MRF interpolation time includes anchor sampling, distance computation, network evaluation, and accumulation. We evaluate on meshes from Thingi10k with the following mesh IDs:

```
[368622, 42435, 65282, 116878, 409624, 101902, 73410, 87602, 255172, 98480,
57140, 285606, 96123, 203289, 87601, 409629, 37384, 57084, 136024, 202267,
101619, 72896, 90064, 127243, 78671, 285610, 75667, 80597, 75651, 75654,
75657, 75665, 75652, 123472, 88855, 444375, 208741, 73877]
```

A.5.2 More details regarding velocity prediction setup

Since velocity magnitudes are meaningful (unlike unit normals), we use a normalized kernel interpolant. Let $\mathbf{m} \in \{0,1\}^{|\widetilde{\mathrm{V}}|}$ be the indicator of observed nodes ($m_i = 1$ if $i \notin M$, else 0), and let $\mathbf{m} \odot \mathbf{U}$ denote row-wise masking. We predict the full field via normalization:

$$\mathbf{U}^{\text{pred}} \;=\; \frac{\mathbf{K}(\mathbf{m} \odot \mathbf{U})}{\mathbf{K}\mathbf{m}}\,, \tag{23}$$

where the division is row-wise and applied to each of the 3 velocity channels.

To match the full-kernel baseline used in vertex-normal prediction, we compute $\mathbf{K}$ via a full spectral decomposition. Specifically, we form $\mathbf{W}_f$ and compute $\mathbf{W}_f = \mathbf{V}\,\mathrm{diag}(\boldsymbol{\lambda})\mathbf{V}^\top$, so that $\mathbf{K} = \mathbf{V}\,\mathrm{diag}(\exp(\tau\boldsymbol{\lambda}))\mathbf{V}^\top$. Baseline preprocessing time therefore includes graph construction and the dense eigendecomposition (dominated by $O(|\widetilde{\mathrm{V}}|^3)$ time and $O(|\widetilde{\mathrm{V}}|^2)$ memory to store $\mathbf{V}$), and each per-frame interpolation requires two dense applications of $\mathbf{K}$ (we apply $\mathbf{K}$ once to the stacked right-hand side $[\mathbf{m},\ \mathbf{m} \odot \mathbf{U}]$ to obtain both numerator and denominator).

For MRFs, we train the regressor g_θ once per discretization on the same dense graph, and precompute a random-feature matrix $\mathbf{Z} \in \mathbb{R}^{|\widetilde{\mathrm{V}}| \times M}$ for a fixed number of features M. This yields the approximation $\mathbf{K} \approx \mathbf{Z}\mathbf{Z}^\top$, so that per-frame interpolation reduces to two thin matrix multiplications: $\mathbf{Z}\mathbf{Z}^\top(\mathbf{m} \odot \mathbf{U}) = \mathbf{Z}(\mathbf{Z}^\top(\mathbf{m} \odot \mathbf{U}))$ and similarly for $\mathbf{K}\mathbf{m}$. We report MRF preprocessing as supervision generation, training, and feature precomputation, and MRF interpolation time as the rest.

A.6 Manifold-valued attention: full masked-reconstruction results

Table 4 reports all Euclidean RF, graph/geodesic low-rank, exact-kernel, and linear-kernel baselines for every source count. Specifically, we compared MRFs against the full graph heat kernel (Exact heat), a low-rank Nyström approximation of that same manifold heat kernel (Geodesic Nystrom), a low-rank spectral approximation of that same manifold heat kernel (Graph diffusion map), the exact Gaussian kernel in ambient Euclidean space (Exact RBF), a low-rank anchor-feature approximation of that ambient Gaussian kernel (Linear RBF), and other RF methods.

Table 4: Masked reconstruction results for source counts 5–10. Lower is better for both metrics.

	Sources 5		Sources 6		Sources 7		Sources 8		Sources 9		Sources 10	
Method	MSE	Rel. ℓ_2	MSE	Rel. ℓ_2	MSE	Rel. ℓ_2	MSE	Rel. ℓ_2	MSE	Rel. ℓ_2	MSE	Rel. ℓ_2
MRF	**3.10**	**1.97**	**3.03**	2.90	**2.90**	**4.87**	**3.11**	**2.15**	**3.15**	**3.46**	**3.25**	**2.46**
Geodesic Nyström	3.77	2.05	4.76	4.49	6.38	7.54	5.74	4.40	4.52	6.87	4.59	5.22
Graph diffusion map	3.61	2.67	3.67	3.37	3.98	7.27	3.96	4.11	3.72	5.37	3.70	4.52
RFF	3.32	2.36	3.20	3.40	4.14	6.76	3.51	3.30	3.40	5.03	3.42	3.52
ORF	3.26	2.24	3.15	3.16	3.84	6.69	3.47	3.20	3.37	4.92	3.38	3.38
PRF	3.34	2.52	3.05	**2.23**	3.31	5.64	3.79	4.63	3.30	4.51	3.65	4.20
PORF	3.52	2.75	3.16	3.18	3.33	6.47	4.18	5.26	3.25	3.97	3.53	4.01
Exact RBF	3.38	2.47	3.22	3.53	4.66	7.36	3.60	3.48	3.48	5.41	3.44	3.65
Linear RBF	3.29	2.42	3.11	3.20	3.92	6.72	3.40	3.07	3.35	4.90	3.38	3.42
Exact heat	3.87	2.00	5.00	4.36	6.91	8.08	5.93	4.47	4.47	5.82	4.81	5.67

A.7 Higher-dimensional descriptor manifolds: Brodatz

We evaluate MRFs on a harder Brodatz protocol designed to test whether unlabeled transformed samples help recover a useful descriptor manifold. Each grayscale Brodatz image defines one class, and each sampled window is mapped to a 15-dimensional log-SPD covariance descriptor computed from intensity, first-order gradients, and second-order derivatives (i.e., each window becomes a 5×5 SPD covariance matrix). Supervision is intentionally sparse: for each class we use only 6 labeled windows drawn from 2 canonical rotations (0° and 90°). The manifold is built by sampling a large unlabeled pool from intermediate rotations ($0^\circ, 15^\circ, 45^\circ, 75^\circ, 90^\circ, 105^\circ, 135^\circ, 165^\circ$). Evaluation is performed on held-out rotations ($30^\circ, 60^\circ, 120^\circ, 150^\circ$). This yields 111 classes, 666 labeled training descriptors, 66,600 unlabeled graph descriptors, and 22,200 test descriptors. As in the paper, we build a graph on the unlabeled transformed descriptor cloud, compute GRF-style random-walk signatures, train the continuous surrogate, then use the resulting MRF features for kNN classification. MRFs noticeably outperform other methods. The metric is group accuracy, where predictions from 50 windows belonging to the same texture-angle block are aggregated by majority vote. Tuned $k \in 1, ..., 15$ for kNN; $k = 1$ was best in every case.

Table 5: Brodatz transformed-view classification with group-vote evaluation. Entries are group accuracy in percent, reported as mean $\pm$ standard deviation over 30 seeds. For RF and kernel methods, the feature dimension or embedding dimension is tuned within each method and k.

Method	$\mathbf{k = 1}$	$\mathbf{k = 3}$	$\mathbf{k = 5}$	$\mathbf{k = 10}$	$\mathbf{k = 15}$
MRF	**55.7 ± 2.9**	**53.6 ± 3.3**	**51.6 ± 3.2**	**45.4 ± 3.3**	**40.3 ± 3.0**
Raw	42.4 ± 2.2	36.9 ± 2.4	33.6 ± 2.7	27.2 ± 2.2	24.1 ± 1.6
ORF	42.6 ± 2.3	37.5 ± 2.4	34.5 ± 2.5	27.8 ± 1.9	24.9 ± 1.7
RFF	43.5 ± 2.5	38.5 ± 2.8	35.4 ± 3.1	28.9 ± 2.6	25.6 ± 2.4
kPCA	35.3 ± 2.4	30.5 ± 2.5	28.5 ± 2.8	24.8 ± 2.1	22.6 ± 1.6
PRF	9.2 ± 2.4	6.3 ± 1.2	5.2 ± 1.1	3.7 ± 0.8	3.0 ± 0.4
PORF	8.9 ± 2.5	5.9 ± 1.4	5.0 ± 1.4	3.6 ± 0.8	3.0 ± 0.8

A.8 Non-Compact Manifolds

Non-compact manifolds require additional care because a finite graph can only approximate bounded regions. Rather than attempting a global discretization, we adopt an exhaustion-based (local-on-compacts) approach: we approximate the heat kernel using finite constructions on balls $B_R(o)$, and evaluate accuracy on a fixed inner region $B_r(o)$. Empirically, we observe that approximation error on $B_r(o)$ decreases as outer radii R increases (see Tab. 6). We evaluate the approximation on a fixed inner ball with $r = 1.5$, using 128 query nodes sampled from that inner region and comparing against the analytic heat kernel at time $t = 0.5$. GRF and MRF walk budgets are increased with R.

A.9 Additional comparisons and sensitivity analyses

A.9.1 Dense correspondence on FAUST

We evaluate dense correspondence on FAUST [1] by predicting where every point on a human body mesh belongs on a fixed 6,890-vertex reference template. We use 80 human meshes for training and

Table 6: Local-on-compacts exhaustion results for the $\mathbb{H}^3$ MRF experiment. Errors are evaluated on a fixed inner test ball while increasing the outer truncation radius R. RMSE values are reported in units of 10^{-3}.

	$\mathbf{R = 2.0}$	$\mathbf{R = 2.5}$	$\mathbf{R = 3.0}$	$\mathbf{R = 3.5}$	$\mathbf{R = 4.0}$	$\mathbf{R = 4.5}$	$\mathbf{R = 5.0}$
Rel. ℓ_2	0.31	0.29	0.25	0.26	0.19	0.21	0.18
RMSE (10^{-3})	3.3	3.1	2.7	2.8	2.1	2.3	1.9

20 held-out shapes for testing, with each test shape evaluated in six forms: the original mesh, four remeshed versions (with different vertex arrangements and connectivity), and a point-cloud. This gives 120 test representations on a rank-64 eigensystem. We compare MRFs with ambient ORF, truncated spectral diffusion, Nyström, and Chebyshev approximations. Table 7 reports normalized geodesic error, the percentage of correct keypoints (PCK), and exact-label accuracy. A correspondence is correct at threshold $q\%$ when its geodesic error is below $q\%$ of the template's geodesic diameter. MRF-64 achieves the lowest mean, median, and 95th-percentile errors and the highest accuracy at every reported PCK threshold.

Table 7: Dense correspondence on FAUST. Mean, median, and P95 are normalized geodesic errors, with P95 denoting the 95th percentile; lower is better. PCK and exact-label accuracy are percentages.

Method	Mean	Median	P95	PCK@1%	PCK@2%	PCK@5%	PCK@10%	Exact-label acc.
MRF-64	0.01055	0.00366	0.02750	75.21	91.37	98.02	99.22	38.68
Ambient ORF-64	0.06725	0.01874	0.50620	29.47	52.33	78.92	90.46	9.66
Spectral-64	0.12074	0.02019	0.73796	34.23	49.77	65.90	74.85	11.21
Nyström-64	0.11926	0.02012	0.73665	34.39	49.85	65.87	74.97	11.28
Chebyshev-24	0.18967	0.06475	0.75583	28.72	34.74	45.89	57.06	17.20

A.9.2 Sensitivity analysis across design parameters

We vary graph weights, neighborhood size, and network architecture one at a time on the flag interpolation task. Recall the baseline configuration uses a symmetric 16-nearest-neighbor graph, a global constant Gaussian bandwidth, and two ReLU hidden layers of width 128. For graph weights, we may ablate by using locally-tuned weights instead, where we use a different Gaussian bandwidth at each node, determined by the distances to its nearby neighbors. This allows edge weights to adapt to local sampling density, with smaller bandwidths in densely sampled regions and larger bandwidths in sparsely sampled regions. Specifically, the weight between nodes i and j is given by $W_{ij} = \exp(-\|x_i - x_j\|^2/(\sigma_i\sigma_j))$, where σ_i and σ_j are their respective local bandwidths (i.e., the distance to the kth-farthest neighbor). In contrast, uniform weights assign the same weight (of one) to every connected pair of nodes, regardless of their distance, before normalization.

For the $k = 8$ and $k = 32$ variants, we vary the number of nearest neighbors while keeping the global Gaussian weighting scheme, with the bandwidth recomputed for each choice of k. Architecture variants change only the hidden-layer width or depth. Table 8 shows that the tested architecture changes have little effect on downstream performance. Graph construction has a larger effect, and improvements in graph-kernel approximation do not always translate into better interpolation. In particular, $k = 8$ improves downstream cosine similarity while increasing exact-heat error; $k = 32$ produces the opposite tradeoff.

A.9.3 Sampling density on the flag surface

We vary the number of sampled points on the flag surface. Specifically, the $N = 1579$ discretization uses the original mesh vertices and the $N = 4000$ and $N = 8000$ discretizations augment these vertices with barycentric samples from the mesh faces. Furthermore, graph-construction settings and training and evaluation budgets are held fixed. Table 9 shows that denser sampling improves both downstream interpolation metrics, with a small increase in preprocessing time over the tested range. Note that these measurements were obtained separately from the main-body experiments and on different hardware, so the reported preprocessing times are not directly comparable.

Table 8: Sensitivity to graph construction and network architecture on flag interpolation. Lower is better for all error metrics; higher is better for cosine similarity. Architecture labels give the number of hidden layers followed

Variant	Held-out pair relative error	Exact-heat relative Frobenius error	Downstream cosine similarity	Downstream relative ℓ_2 error
Baseline W, $k = 16$, 2×128	0.0461 ± 0.0014	0.6093 ± 0.0615	0.9356 ± 0.0087	0.2584 ± 0.0084
Locally-tuned W	0.0526 ± 0.0007	0.5734 ± 0.0533	0.9302 ± 0.0099	0.2737 ± 0.0104
Uniform W	0.0497 ± 0.0006	0.5177 ± 0.0442	0.9237 ± 0.0103	0.2891 ± 0.0095
$k = 8$	0.0230 ± 0.0015	0.8319 ± 0.0487	0.9453 ± 0.0079	0.2287 ± 0.0062
$k = 32$	0.0566 ± 0.0022	0.4632 ± 0.0458	0.9123 ± 0.0109	0.3122 ± 0.0108
2×64	0.0472 ± 0.0008	0.6144 ± 0.0469	0.9353 ± 0.0086	0.2588 ± 0.0083
2×256	0.0474 ± 0.0002	0.6122 ± 0.0716	0.9359 ± 0.0088	0.2579 ± 0.0081
3×128	0.0493 ± 0.0007	0.6062 ± 0.0449	0.9352 ± 0.0095	0.2591 ± 0.0091

Table 9: Effect of sampling density on flag interpolation. Lower is better for relative errors, and higher is better for cosine similarity.

N	Validation relative error	Downstream cosine similarity	Downstream relative ℓ_2 error	MRF preprocessing (s)
1579	0.3419 ± 0.0023	0.8479 ± 0.0230	0.6194 ± 0.1748	66.26 ± 0.28
4000	0.3692 ± 0.0033	0.9167 ± 0.0010	0.3456 ± 0.0750	67.60 ± 0.92
8000	0.3340 ± 0.0033	0.9424 ± 0.0029	0.2706 ± 0.0456	68.88 ± 0.66

A.9.4 Graph resolution and diffusion-time matching

We evaluate graph-heat approximation on the four synthetic geometries against an analytic or high-resolution reference, using the same five sets of evaluation points across graph resolutions (for consistent comparison). As the graph resolution increases, we adjust the diffusion time using the graph spectral gap (i.e. the smallest positive eigenvalue for a connected graph Laplacian) to maintain a comparable diffusion scale across discretizations. For comparison, we also evaluate the finest discretization $N = 16384$ with a fixed diffusion time $\tau = 20$.

Table 10 shows that approximation error decreases with increasing resolution when diffusion time is matched. In contrast, keeping $\tau = 20$ at the finest resolution leads to substantially larger errors across all four geometries, highlighting the importance of calibrating the diffusion scale.

Table 10: Relative Frobenius error of the graph heat operator against an analytic or high-resolution reference. The first three columns use spectral-gap time matching; the final column retains $\tau = 20$ at the finest resolution. Lower is better.

	Matched diffusion time			Fixed $\tau = 20$
Geometry	$N = 256$	$N = 4000$	$N = 16384$	$N = 16384$
Sphere	0.418 ± 0.013	0.058 ± 0.004	0.021 ± 0.002	0.655 ± 0.036
Ellipsoid	0.391 ± 0.022	0.151 ± 0.009	0.093 ± 0.009	0.574 ± 0.024
Torus	0.488 ± 0.018	0.295 ± 0.014	0.258 ± 0.015	0.736 ± 0.028
Möbius strip	0.905 ± 0.024	0.520 ± 0.022	0.391 ± 0.026	0.841 ± 0.098

A.9.5 Measured preprocessing scaling

We report preprocessing measurements for 38 Thingi10K meshes and 49 discretizations of the `flag_simple` surface. Table 11 reports the timing endpoints and empirical power-law fits $t_{\mathrm{pre}}(N) \propto N^{\beta}$. We see that MRF preprocessing grows more slowly than dense full-kernel construction, which quickly becomes intractable, over the tested ranges.

A.10 Code availability

Code for all our experiments can be found here.

Table 11: Preprocessing measurements and empirical scaling fits. N is the number of discretization points, β is the fitted exponent in $t_{\mathrm{pre}}(N) \propto N^{\beta}$, and R^2 measures goodness of fit. Time ranges give the measurements at the smallest and largest tested discretizations.

Dataset	Method	N range	Time range (s)	β	R^2
Thingi10K	MRF	1094–47024	120–171	0.09	0.81
Thingi10K	Dense full kernel	1094–47024	0.13–2936	2.75	0.995
`flag_simple`	MRF	1579–56804	126–746	0.50	0.954
`flag_simple`	Dense full kernel	1579–56804	0.34–3754	2.67	0.997

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction outline and state the claims made in the paper, and give a clear summary of the contributions that we make. Our theoretical and empirical results reflect these claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Although we don't have a separate limitations section we point out the limitations and assumptions of MRFs throughout the paper. For example, at the beginning of Section 3 we state that MRFs require an offline preprocessing cost that makes it worthwhile for larger problems (not one-off small problems), and at the end of Section 3.3 we mention that our focus is on compact manifolds and future work will be needed to extend our analysis to non-compact settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All theoretical results have detailed assumptions and proofs in the appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All information needed to reproduce the experimental results are detailed in Section 3 and the appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is available here (we have put this at the end of the appendices). Relevant synthetic data can be generated and/or found online (all open access).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details are all specified in the experiments section (3) and/or the appendices. These can be used to understand results and how to recreate results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For experiments with randomness/uncertainty, error bands or error bars have been added (note that some are too small to see on the graph due to the scaling of the y-axis, but they are indeed there).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments were run using a TPU v6e and an A100 GPU (for neural network training only). This is mentioned at the beginning of Section 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, our paper conforms with the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: To the best of our knowledge, there are no immediate societal impacts. We focus on a methodological framework for kernels on manifolds, without direct applications that could harm society.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper poses no such risks. We focus on a methodological framework for kernels on manifolds—this does not have high risk for misuse and thus no safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: To the best of our knowledge, we have properly credited anything used that was not created by ourselves (i.e. external datasets and methods used, including Thingi10K, flag_simple/MeshGraphNets, Brodatz, and RF baselines). The code repository was fully developed and maintained by us, the owners/authors.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset introduced by the paper is the experimental implementation. Code is available anonymously here.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.