Eurographics Symposium on Geometry Processing 2018
T. Ju and A. Vaxman
(Guest Editors)

Volume 37 (2018), Number 5

Kernel Functional Maps

L. Wang[1] A. Gehre[1,2] M. M. Bronstein[3,4,5,6] J. Solomon[1]

[1]MIT [2]RWTH Aachen University [3]Imperial College London [4]USI Lugano [5]Intel [6]IAS TU Munich

Abstract
Functional maps provide a means of extracting correspondences between surfaces using linear-algebraic machinery. While the functional framework suggests efficient algorithms for map computation, the basic technique does not incorporate the intuition that pointwise modifications of a descriptor function (e.g. composition of a descriptor and a nonlinearity) should be preserved under the mapping; the end result is that the basic functional maps problem can be underdetermined without regularization or additional assumptions on the map. In this paper, we show how this problem can be addressed through kernelization*, in which descriptors are lifted to higher-dimensional vectors or even infinite-length sequences of values. The key observation is that optimization problems for functional maps only depend on inner products between descriptors rather than descriptor values themselves. These inner products can be evaluated efficiently through use of kernel functions. In addition to deriving a kernelized version of functional maps including a recent extension in terms of pointwise multiplication operators, we provide an efficient conjugate gradient algorithm for optimizing our generalized problem as well as a strategy for low-rank estimation of kernel matrices through the Nyström approximation.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computer graphics—Computational Geometry and Object Modeling

1. Introduction

Shape correspondence is a critical problem in geometry processing, with applications in texture transfer, semantic segmentation, shape interpolation, and other tasks. The basic goal is to identify pairs of matching points between two surfaces, which may differ in their local geometric features or meshing but exhibit shared structure that can be leveraged to extract a smooth map.

A key idea introduced to correspondence in [OBCS*12] involves the extraction of *functional maps* rather than point-to-point maps. While the basic output of correspondence tools used to be a mapping from points on one surface to points on another, the functional perspective views a map as a correspondence between functions *over* one surface to functions *over* another. The end result is a model that poses correspondence using the languages of functional analysis and linear algebra rather than differential geometry: Feature preservation is expressed via preservation of descriptor functions, and distortion is measured through commutativity between the map and differential operators like the Laplacian. Beyond leveraging efficient linear algebra tools to extract correspondences, functional map algorithms enjoy multiscale properties inherited from writing the map in a basis such as the Laplace–Beltrami eigenmodes.

A small puzzle, however, persists in the basic functional maps formulation. Suppose a functional map takes a function $f(x)$ over surface M to its image $f \circ \phi(y)$ over surface N; here, $\phi : M \to N$ is an underlying point-to-point map. Even if f is mapped exactly, there is no guarantee that the functional map will preserve pointwise modifications of f, such as $g(x) := f^2(x)$ or $h(x) = \cos f(x)$. One way to understand this challenge is to count degrees of freedom. A functional map from one triangle mesh of n vertices into another has n^2 degrees of freedom, but $3n$ values are sufficient to specify a point-to-point map precisely: The positions of the vertices of one mesh on the other.

In this paper, we formalize and explore the limits of perhaps the most straightforward means of resolving the aforementioned issue, *kernelization*. Long recognized as a powerful technique in machine learning, kernelization involves lifting descriptors (or features) into higher-dimensional spaces in the context of least-squares and other problems over Hilbert spaces. In our example above, kernelization might replace a single descriptor $f(x) : M \to \mathbb{R}$ with a triplet of descriptors $(f(x), f^2(x), \cos f(x)) : M \to \mathbb{R}^3$. Since this deterministic lifting process simply manipulates the input descriptors in a pointwise fashion, it may not appear to add more information; in reality, however, we can show that this change addresses underdetermined problems that otherwise appear in the basic functional maps framework.

Two recent papers [NO17, NMR*18] consider a related problem, namely *pointwise product* preservation under functional maps, in two different ways. The first uses a different data term enforcing commutativity of the functional map with the pointwise product operation. The second represents functional maps in an extended basis containing pointwise products of the basis functions. Importantly, kernelization can be applied to these and other formulations and flavors of functional maps, complementing their work by explicitly promoting preservation of not only pointwise products but also nonlinearities applied to descriptor values [NO17, NMR*18].

We begin by showing how the most popular objective functions proposed for functional maps can be kernelized using a straightforward sequence of algebraic simplifications. Beyond this mathematical contribution, we provide an efficient means of optimizing for functional maps after introducing a kernel, including application of the Nyström approximation to avoid using any matrices that scale quadratically in the size of the mesh.

2. Related Work

2.1. Functional Maps

Geometric tools in graphics, medical imaging, vision, and other disciplines require correspondence algorithms. Many techniques have been applied to this problem, surveyed in [VKZHCO11].

Our paper involves the functional map framework introduced in [OBCS*12]. The idea is that a smooth map between surfaces $\phi : M \to N$ admits a dual $F_\phi : L_2(N) \to L_2(M)$ defined through composition: For $f \in L_2(N)$, we define $F_\phi[f] := f \circ \phi$. Functional map algorithms estimate F_ϕ directly rather than ϕ, using tools from optimization and linear algebra. Correspondence is posed as a variational problem for a matrix representing F_ϕ in a basis.

Multiple follow-up works aimed mainly at improving the stability and applicability of the framework; see [COC*17] for a survey. Notable extensions of functional maps include incorporation of sparsity-based regularization [PBB*13], optimization on the manifold of orthogonal matrices [KBB*13, KGB16], design [ADK16] and tuning [COC14, LRR*17] of descriptors, correspondence between shape collections [HWG14, KGB16], partial correspondence [LRB*16, RCB*17], incorporation of descriptors through operators rather than least-squares [NO17], use of adjoint operators to incorporate information about the reverse map [ERGB16, HO17], and regularization based on conserving products in addition to linear combinations [NMR*18]. Functional maps also have been combined with deep learning to learn how to extract dense correspondences [LRR*17].

Our paper observes that descriptor-based objective terms common to many of the techniques cited above admit kernelization. Although we test our generalization through the pipelines proposed in [OBCS*12, NO17], our technique is applicable to any method with a least-squares descriptor preservation term.

2.2. Kernelization and Nyström Approximation

Kernelization is a key technique in machine learning and was a central focus of research during the early years of that discipline [CST00]; see §4 for mathematical discussion. The basic observation that learning tools such as the perceptron algorithm [Aiz64], principal component analysis [SSM97], canonical correlation analysis [LF00], and support vector machines [CST00] can be lifted to potentially infinite-dimensional inner product spaces while maintaining computability. Our kernelized extension of the simplest descriptor term in functional maps [OBCS*12] can be viewed as an application of kernel ridge regression [SGV98]; to our knowledge kernelization of the more recent functional maps technique [NO17] does not correspond to any standard kernel method in machine learning.

Kernelization can reduce from an infinite-dimensional feature space to a finite-dimensional kernel matrix, but the kernel matrix is still a dense matrix whose size scales quadratically in the size of the input data. The Nyström approximation [Nys30] we will use in §6 replaces the full kernel with a low-rank matrix, an effective approximation when the spectrum of the kernel decays quickly. The Nyström method has been applied effectively to large-scale machine learning [WS01], reducing computational load while accounting for relationships between *all* members of a dataset (at least through their proximity to a subsampled set of key points) rather than downsampling.

3. Mathematical Preliminaries

3.1. Optimization for Functional Maps

To begin, suppose M and N are two triangulated surfaces, with m and n vertices respectively. We assume these surfaces have diagonal mass matrices $A_M \in \mathbb{R}_+^{m\times m}$ and $A_N \in \mathbb{R}_+^{n\times n}$, respectively. We furthermore assume that the surfaces are equipped with bases $\Phi_M \in \mathbb{R}^{m\times k_M}$ and $\Phi_N \in \mathbb{R}^{n\times k_N}$ approximately spanning the set of functions over the two surfaces; for simplicity we will assume that the bases are orthonormal:

$$\Phi_M^\top A_M \Phi_M = I_{k_M\times k_M}$$
$$\Phi_N^\top A_N \Phi_N = I_{k_N\times k_N}.$$

A key example involves computing the bases Φ_M and Φ_N as Laplace–Beltrami eigenvectors, for which $\Delta\Phi = A\Phi\Lambda$, where Δ is the cotangent Laplacian matrix [PP93] and Λ is a diagonal matrix of nonnegative eigenvalues.

A *functional map* is a linear mapping from functions over one surface to functions over another [OBCS*12]. After truncating to the Φ bases, we can think of a functional map as a matrix $C \in \mathbb{R}^{k_N\times k_M}$ transforming coefficients of a function $p_M \in \mathbb{R}^m$ to those of a mapped function $p_N \in \mathbb{R}^n$:

$$C\Phi_M^\top A_M p_M = \Phi_N^\top A_N p_N.$$

The goal of functional mapping algorithms is to recover the matrix C. Most approaches are *variational* in nature, optimizing for the best matrix C minimizing a designed objective function. For instance, if the map is an isometry then we expect it to commute with the Laplacian Δ, in which case we may wish to minimize a commutator norm w.r.t. the matrix C [OBCS*12]

$$F_\Delta(C) := \|C\Lambda_M - \Lambda_N C\|_{\text{Fro}}^2 \tag{1}$$

Low values of this objective function indicate that applying the source surface's Laplacian and then transporting a function by C is nearly identical to transporting the function and then applying the Laplacian of the target surface.

More pertinent to the discussion in this paper, suppose we are given d descriptor functions that match between the two surfaces arranged into matrices $P_N \in \mathbb{R}^{n\times d}$ and $P_M \in \mathbb{R}^{m\times d}$, respectively; for instance, the heat kernel signature [SOG09] is preserved under isometry and could be precomputed for the inputs. After writing them in the Φ bases, we can store the descriptor coefficients in matrices $\hat{P}_M := \Phi_M^\top A_M P_M \in \mathbb{R}^{k_m\times d}$ and $\hat{P}_N := \Phi_N^\top A_N P_N \in \mathbb{R}^{k_n\times d}$. The original work on functional maps proposed measuring descriptor preservation through the objective function

$$F_P(C) := \|C\hat{P}_M - \hat{P}_N\|_{\mathrm{Fro}}^2. \tag{2}$$

This objective simply checks that the images of the descriptors under the functional map match. More recent work [NO17] proposes an operator-based technique for measuring descriptor preservation:

$$F_D(C) := \sum_i \|CX_M^i - X_N^i C\|_{\mathrm{Fro}}^2, \tag{3}$$

where

$$\begin{aligned} X_M^i &:= \Phi_M^\top A_M \operatorname{diag}(p_M^i)\Phi_M \\ X_N^i &:= \Phi_N^\top A_N \operatorname{diag}(p_N^i)\Phi_N. \end{aligned}$$

Here, p^i denotes the i-th descriptor function in the per-vertex basis. The idea of this objective function is that the functional map C should commute with the operator that multiplies a function pointwise by the descriptor.

Additional constraints can further improve the quality of a functional map. For instance, if the underlying point-to-point map is area-preserving, then C should be an orthogonal matrix: $C^\top C = I_{k_M\times k_M}$ [OBCS*12]; in this setting, minimization of (2) can be posed as manifold optimization [KGB16]. Other changes of basis may imply that C should be sparse or low-rank [PBB*13, GCR*17]. Partial correspondence is encoded by a matrix C with approximately slanted-diagonal structure [RCB*17]. Finally, a pointwise map corresponds to a pointwise product preserving functional map [NO17]; under this assumption, the matrix C can be extended with extra coefficients in the pointwise products of the bases Φ [NMR*18].

3.2. Kernels and Mercer's Theorem

Our main contribution is to show that (2) and (3) are amenable to *kernelization*. We work with the following definition:

Definition (Positive definite kernel on $\mathbb{R}^d$)**.** *A* positive definite kernel *on $\mathbb{R}^d$ is a symmetric continuous function $K:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ such that*

$$\begin{pmatrix} K(x_1,x_1) & K(x_1,x_2) & \dots & K(x_1,x_m) \\ K(x_2,x_1) & K(x_2,x_2) & \dots & K(x_2,x_m) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_m,x_1) & K(x_m,x_2) & \dots & K(x_m,x_m) \end{pmatrix} \succeq 0$$

for any finite set of points $\{x_1,\dots,x_m\}\subseteq\mathbb{R}^d$ with $m\geq 1$.

In this paper, we will use the term "kernel" to refer only to positive definite kernels.

Mercer's theorem is a theoretical result about positive definite kernels indicating that they can be regarded as dot products in a high (potentially infinite) dimensional space:

Theorem (Mercer's theorem, simplified)**.** *Suppose $K(x,y)\in L^2(\mathbb{R}^d\times\mathbb{R}^d)$ is a positive definite kernel. Then, there exists a sequence of functions $\{\varphi_i(x)\}_{i=1}^\infty$ such that*

$$K(x,y) = \sum_{i=1}^\infty \varphi_i(x)\varphi_i(y), \tag{4}$$

with convergence in the L^2 norm.

Mercer's theorem is essentially a corollary of the spectral theorem; we can write $\phi_i = \sqrt{\lambda_i}\psi_i$ where ψ_i is a unit eigenvector of the operator T_K defined via $\langle T_K f, g\rangle := \int_{\mathbb{R}^d\times\mathbb{R}^d} K(x,y)f(x)g(y)\,dx\,dy$.

Mercer's theorem allows linear techniques to be lifted to other spaces. It has been applied to countless basic problems in learning, from ridge regression to SVMs [CST00]. The basic observation is that many problems only require inner products *between* data points rather than the data points themselves. For instance, consider the normal equations $A^\top Ax = A^\top b$ resulting from the least-squares problem $\min_x \|Ax-b\|_2^2$. Notice

$$A^\top A = \begin{pmatrix} a_1\cdot a_1 & a_1\cdot a_2 & \cdots & a_1\cdot a_n \\ a_2\cdot a_1 & a_2\cdot a_2 & \cdots & a_2\cdot a_n \\ \vdots & \vdots & \ddots & \vdots \\ a_n\cdot a_1 & a_n\cdot a_2 & \cdots & a_n\cdot a_n \end{pmatrix} \quad A^\top b = \begin{pmatrix} a_1\cdot b \\ a_2\cdot b \\ \vdots \\ a_n\cdot b \end{pmatrix},$$

where a_i is the i-th column of A. The dot products $a_i\cdot a_j$ can be replaced with kernelized values $K(a_i,a_j)$, which—by Mercer's theorem—effectively lifts the data to a higher-dimensional space before performing least-squares. For example, in a scalar variable the kernel $K(s,t) := (1,s,s^2)\cdot(1,t,t^2)$ lifts linear least-squares to quadratic least-squares. Other kernels, e.g. the Gaussian kernel $K(x,y) := \exp(-\gamma\|x-y\|_2^2)$, correspond to infinite sequences of functions φ_i in (4).

From a practical perspective, Mercer's theorem allows for a much broader class of liftings φ than would be achievable by applying the desired modifications directly to the descriptor functions. In particular, while it may be easy algorithmically to take inner products of low-dimensional liftings like $t\mapsto(1,t,t^2)$, Gaussian kernels and many others belong to a class for which evaluating the inner product $K(x,y)=\varphi(x)\cdot\varphi(y)$ is *easier* than evaluating φ directly; indeed, the φ corresponding to a Gaussian kernel is an infinitely-long sequence that cannot be stored on a computer (see the proof of the proposition in §4.4).

4. Kernelizing Functional Maps

Our basic motivation for kernelizing functional maps comes from an easily-understood limitation of the current functional mapping tools. In particular, suppose a descriptor $f(x)$ is included in a term like (2). Even if $f(x)$ is preserved exactly under the functional map—and hence the corresponding piece of $F_P(C)$ vanishes—it can be the case that a derived descriptor like $g(x) := f(x)^2$ is not. Hence, we might augment our list of descriptors to also include $f(x)^2$ and other pointwise variations of $f(x)$, or even the product of descriptors $f(x)h(x)$; kernelization attempts to capture some set of derived descriptors in this fashion, implicitly or explicitly. Surprisingly, this change is not just superficial but rather introduces a significant improvement to functional maps algorithms.

Our goal in this section is to show that the descriptor preservation objectives (2) and (3) can be kernelized. We additionally prove under weak assumptions that kernelization restores the rank of the functional mapping problem, showing that fewer than n descriptors are needed to recover an $n \times n$ functional map.

4.1. Kernel Matrices

As in §3.1, suppose we are given d matching descriptors on meshes M and N. We will assume these are *not* in a reduced basis, and hence our descriptor matrices contain one value per vertex: $P_M \in \mathbb{R}^{m\times d}$ and $P_N \in \mathbb{R}^{n\times d}$.

Our basic observation is that function preservation constraints (2) and (3) can be written completely in terms of *kernel (Gram) matrices* whose entries are given by

$$K_{\text{linear}}(P_M,P_N)_{ij} := (P_{M,\text{row } i})\cdot(P_{N,\text{row } j}) = (P_M P_N^\top)_{ij}.$$

Suppose we are given a feature map $\varphi : \mathbb{R}^d \to \mathbb{R}^{d'}$, which lifts the d-dimensional features to d'-dimensional features; typically we think of $d' > d$ although this will not be necessary for our discussion below. As a preprocess, we could run the rows of P_M and P_N through φ before computing the functional map. In this case, we evidently will be able to write the resulting problem in terms of

$$K_\varphi(P_M,P_N)_{ij} := \varphi(P_{M,\text{row } i})\cdot\varphi(P_{N,\text{row } j}). \tag{5}$$

Mercer's Theorem, introduced in §3.2, indicates that any positive definite kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ implicitly can be written in form (5). Hence, we will eventually bypass the use of φ and work directly with kernel functions K, via the kernel matrix definition

$$K(P_M,P_N)_{ij} := K(P_{M,\text{row } i}, P_{N,\text{row } j}). \tag{6}$$

Our basic outline is to show that (2) and (3) can be written in terms of K_{linear}; then, kernelizing these terms is as simple as replacing K_{linear} with a Mercer kernel matrix of the form (6).

4.2. Linear Function Preservation

As a warm up, consider the objective term defined in (2). Since in this section we will assume descriptors are known per-vertex, we will use a slightly modified version of F_P:

$$F_P(C) = \|C\Phi_M^\top A_M P_M - \Phi_N^\top A_N P_N\|_{\text{Fro}}^2. \tag{7}$$

This expression assumes P_M and P_N are known per-vertex; the "$\Phi^\top A$" matrices project into the bases whose columns are the Φ's.

Expanding the square shows:

$$\begin{aligned}F_P(C) =& \operatorname{Tr}(C^\top C\Phi_M^\top A_M K_{\text{linear}}(P_M,P_M)A_M\Phi_M)\\ &- 2\operatorname{Tr}(C\Phi_M^\top A_M K_{\text{linear}}(P_M,P_N)A_N\Phi_N) + \text{const.}\end{aligned}$$

Hence, kernelizing (2) involves substituting in a general kernel K for K_{linear}, leading to the energy term

$$\begin{aligned}F_{P,K}(C) =& \operatorname{Tr}(C^\top C\Phi_M^\top A_M K(P_M,P_M)A_M\Phi_M)\\ &- 2\operatorname{Tr}(C\Phi_M^\top A_M K(P_M,P_N)A_N\Phi_N) + \text{const.},\end{aligned} \tag{8}$$

with gradient

$$\nabla_C F_{P,K} = 2C\Phi_M^\top A_M K(P_M,P_M)A_M\Phi_M - 2\Phi_N^\top A_N K(P_N,P_M)A_M\Phi_M.$$

This expression is somewhat complicated, but the key observation is that $F_{P,K}$ only involves evaluating K rather than explicitly applying a potentially high-dimensional feature map φ.

4.3. Commutative Function Preservation

Now we consider the more complicated expression (3). Once again we expand the square:

$$F_D(C) = \sum_i \left[\operatorname{Tr}(C^\top C X_M^i X_M^{i\top}) - 2\operatorname{Tr}(CX_M^i C^\top X_N^{i\top}) + \operatorname{Tr}(CC^\top X_N^{i\top} X_N^i)\right]. \tag{9}$$

Algebraic rearrangements lead to the following expressions:

$$\sum_i X_M^i X_M^{i\top} = \Phi_M^\top A_M \left[(\Phi_M\Phi_M^\top)\odot K_{\text{linear}}(P_M,P_M)\right] A_M\Phi_M \tag{10}$$

$$\sum_i X_N^{i\top} X_N^i = \Phi_N^\top A_N \left[(\Phi_N\Phi_N^\top)\odot K_{\text{linear}}(P_N,P_N)\right] A_N\Phi_N \tag{11}$$

$$\sum_i C X_M^i C^\top X_N^{i\top} = C\Phi_M^\top A_M \left[(\Phi_M C^\top \Phi_N^\top)\odot K_{\text{linear}}(P_M,P_N)\right] A_N\Phi_N, \tag{12}$$

where $\odot$ denotes the elementwise (Hadamard) product.

Replacing K_{linear} with a general kernel K in (10), (11), and (12) and subsequently substituting into (9) yields a kernelized version of commutative function preservation, which we will denote $F_{D,K}$. The gradient of this objective term is given by

$$\begin{aligned}\nabla_C F_{D,K} =& 2C\Phi_M^\top A_M \left[(\Phi_M\Phi_M^\top)\odot K(P_M,P_M)\right] A_M\Phi_M\\ &- 4\Phi_N^\top A_N[(\Phi_N C\Phi_M^\top)\odot K(P_N,P_M)]A_M\Phi_M\\ &+ 2\Phi_N^\top A_N \left[(\Phi_N\Phi_N^\top)\odot K(P_N,P_N)\right] A_N\Phi_N C.\end{aligned}$$

Once again, while the notation is somewhat cumbersome, the end result here is that the commutative function preservation term (3) readily admits kernelization. As we will see in the next section, using a kernel does not incur significant computational cost relative to solving the non-kernelized problem: In the end, the matrix products above lead to matrices whose size is on the order of that of C rather than the full mesh.

4.4. Analysis

The simplest formulation for functional maps, in which the descriptor term (7) is minimized without regularization, can fail not just in the presence of noisy data but also if the descriptor matrices P_M and P_N are not full-rank relative to the size of C. This issue makes it difficult to recover functional maps that are pullbacks of point-to-point maps, since a quadratic amount of data is needed to have a full-rank descriptor matrix.

We can address this issue by using nearly any positive definite kernel. As an example, here we verify the invertibility of (7) when using a Gaussian kernel $K_\alpha(P_M,P_N)_{ij} := e^{-\alpha\|P_{M,\text{row } i} - P_{N,\text{row } j}\|_2^2}$:

Proposition. *$K_\alpha(P,P)$ is invertible whenever P has distinct rows.*

Proof. We recall a standard proof. Without loss of generality, shift the problem so that no row p_i of P equals zero. Expanding the square, $K_{ij} = e^{-\alpha\|p_i\|_2} e^{2\alpha p_i\cdot p_j} e^{-\alpha\|p_j\|_2^2}$. Hence, $K = D^\top GD$ where

$D = \mathrm{diag}(e^{-\alpha\|p_i\|_2})$ and $G_{ij} = e^{2\alpha p_i \cdot p_j}$. D is trivially invertible, so it suffices to argue about G. Substituting the Taylor expansion,

$$G_{ij} = \sum_{m=0}^{\infty} \frac{(2\alpha p_i \cdot p_j)^m}{m!} := \sum_{m=0}^{\infty} \frac{(2\alpha\|p_i\|_2\|p_j\|_2)^m}{m!} G_{ij}^{(m)}, \qquad (13)$$

where $G_{ij}^{(m)} := (p_i \cdot p_j / \|p_i\|_2 \|p_j\|)^m$. We have $G^{(0)} = 11^\top \succeq 0$ and $G^{(1)} = \hat{P}\hat{P}^\top \succeq 0$, where $\hat{P}$ contains the rows of P normalized to unit length; these are semidefinite since they are the outer product of a matrix and itself. Furthermore, $G^{(m)} = G^{(1)} \odot \cdots \odot G^{(1)} \succeq 0$ since the Hadamard product of semidefinite matrices is semidefinite.

Note $\mathrm{diag}\, G^{(m)} = 1$for all m. When $i \neq j$, by Cauchy–Schwarz $G_{ij}^{(m)} = (p_i \cdot p_j / \|p_i\|_2 \|p_j\|)^m \xrightarrow{m\to\infty} 0$. Hence, $G^{(m)} \xrightarrow{m\to\infty} I$, for large m each term in the sum is diagonally dominant and hence invertible. So, G is a sum of semidefinite matrices with at least one invertible term, showing G is positive definite, as needed. □

Other (non-kernel) extensions of functional maps may have similar nondegeneracy properties, demonstrated by the experiments in [NO17] with relatively few descriptors. The notable property here is that invertibility is a consequence of standard arguments about positive definite kernels and does not require modifying the original functional maps problem.

It is also worth noting a corollary, that kernels can recover point-to-point maps from accurate but low-dimensional descriptors, a property that does not hold for the original functional maps formulation thanks to loss of rank. This is true not just for Gaussian kernels but rather for *any* full-rank kernel; roughly this occurs whenever the corresponding Taylor expansion (13) has infinite length.

This proposition also highlights a curious property of methods like kernel functional maps and [NO17], which only need a few descriptors. Even in the presence of *one* descriptor we recover a unique functional map, although we know generically at least two coordinates are needed to specify a point on a surface. This is an interface between discrete and smooth geometry: The kernel K_α evaluated on a discrete set of sampled points generically is unlikely to hit the same level set of an underlying smooth function twice. Hence, K_α in this case will be invertible but likely ill-conditioned. In our experiments, however, we find that the kernelized problem becomes well-conditioned with just a few descriptors, whereas the non-kernelized problem may remain ill-conditioned or non-invertible.

5. Algorithm

Our implementation extends the technique described in [NO17]. We follow their protocol for computing functional maps (which in turn follows [OBCS*12]), with the one exception being a modified least-squares optimization problem that incorporates kernelization:

$$C_{\mathrm{opt}}^{K} := \arg\min_{C} F_{P,K}(C) + F_{D,K}(C) + \alpha F_\Delta(C). \qquad (14)$$

When $K(x,y) = x \cdot y$, (14) reduces to the formulation in [NO17] exactly, although we choose to use an alternative scalable optimization scheme.

```
function ApplyOperator($C$)
    $G_{P,K} \leftarrow C\Phi_M^\top A_M K(P_M,P_M) A_M \Phi_M$
    $G_{D,K} \leftarrow \left\{\begin{array}{l} C\Phi_M^\top A_M \left[(\Phi_M\Phi_M^\top)\odot K(P_M,P_M)\right] A_M\Phi_M \\ -2\Phi_N^\top A_N[(\Phi_N C\Phi_M^\top)\odot K(P_N,P_M)]A_M\Phi_M \\ +\Phi_N^\top A_N \left[(\Phi_N\Phi_N^\top)\odot K(P_N,P_N)\right] A_N\Phi_N C \end{array}\right.$
    $G_\Delta \leftarrow C\Delta_M^2 - 2\Delta_N C\Delta_M + \Delta_N^2 C$
    return $G_{P,K} + G_{D,K} + \alpha G_\Delta$
end function

function CGSolve($\Phi_M, A_M, P_M, \Delta_M, \Phi_N, A_N, P_N, \Delta_N, \alpha$)
    $B, P, R \leftarrow \Phi_N^\top A_N K(P_N,P_M) A_M \Phi_M$
    $r \leftarrow \langle R,R \rangle$
    $C \leftarrow$Zeros$(k_N, k_M)$
    for $k = 1,2,\ldots$ do
        $Q \leftarrow$ ApplyOperator$(P)$
        $a \leftarrow \frac{r}{\langle P,Q\rangle}$
        $r_0 \leftarrow r$
        $C \leftarrow C + aP$
        $R \leftarrow R - aQ$
        $r \leftarrow \langle R,R \rangle$
        if $\sqrt{r} < \varepsilon$ then return $C$
        $P \leftarrow R + \frac{rP}{r_0}$
    end for
end function
```

Figure 1: *Although the variable in (14) is a matrix C, the objective is still a least-squares problem to which we can apply the conjugate gradient algorithm. Note our actual implementation precomputes matrices repeated above; when the kernel matrices $K(\cdot,\cdot)$ are too large we apply the Nyström approximation described in §6.2.*

Conjugate gradients. While previous work largely uses direct linear solvers, for scalability we minimize (14) using the iterative conjugate gradient algorithm. This technique is guaranteed to find a global optimum since (14) is a sum of squared norms applied to linear expressions in the unknown matrix. Our iterative algorithm, whose pseudocode is given in Figure 1, avoids computing Kronecker products with repeated dense blocks, which often appear in direct solver implementations of these optimization problems.

The standard conjugate gradients algorithm solves a minimization problem of the form $\|Ax-b\|_2^2$ by iteratively multiplying by $A^\top A$; notice $\nabla_x \|Ax-b\|_2^2 = 2A^\top Ax - 2A^\top b$, so the linear operator $A^\top A$ is the linear term in the gradient of the least-squares objective. To make conjugate gradients applicable to our method, we introduce a function ApplyOperator, which returns the linear piece of the gradient of (14). As a side benefit to our work, we find this approach to be an equally efficient and more easily-implemented alternative to direct solver machinery used for past functional maps work.

6. Nyström Approximation

Our basic algorithm is guaranteed to converge but suffers from a challenging efficiency drawback: The $K(\cdot,\cdot)$ matrices are too large to store for typical meshes. Here we show how this challenge can be overcome with a standard low-rank approximation of K known as the Nyström approximation [WS01].

6.1. Basic Construction

Suppose $K \in \mathbb{R}^{n\times n}$ is a symmetric, positive definite kernel matrix. Following [WS01], suppose we subsample n' rows from K, dividing it into a block matrix:

$$K = \begin{pmatrix} K_{11} & K_{12} \\ K_{12}^\top & K_{22} \end{pmatrix}. \tag{15}$$

Here, $K_{11} \in \mathbb{R}^{n'\times n'}$, $K_{12} \in \mathbb{R}^{n'\times(n-n')}$, and $K_{22} \in \mathbb{R}^{(n-n')\times(n-n')}$.

Assuming $n' \ll n$, typically we have the storage capacity computationally to store all blocks of this matrix *except* K_{22}. Suppose, however, that K is rank-n' and that the first n' rows are linearly independent. Then, one can show that $K_{22} = K_{12}^\top K_{11}^{-1} K_{12}$, a product of low-rank factors.

The Nyström approximation for *any* kernel matrix writes

$$K_{22} \approx K_{12}^\top K_{11}^{-1} K_{12}. \tag{16}$$

This approximation holds exactly in the low rank case. More broadly, if the eigenvalues of K decay at a reasonable rate, this approximation can be extremely effective even if n' rows are subsampled at random.

6.2. Nyström for Functional Maps

It is tempting to apply the Nyström approximation—or a suitable extension for asymmetric matrices [NAS16]—to each $K(\cdot,\cdot)$ term in Figure 1 independently to reduce computational burden. This strategy, however, turns out to be ineffective: Approximating the terms independently disconnects the algorithm from the least-squares problem (14), and as a result the conjugate gradient algorithm no longer is guaranteed to converge thanks to a loss of symmetry and/or positive definiteness. Instead, we derive a strategy that maintains positive definiteness of the functional maps linear system while applying the Nyström formula to reduce computational burden.

Recall that $P_M \in \mathbb{R}^{m\times d}$ and $P_N \in \mathbb{R}^{n\times d}$ contain per-vertex descriptors on M and N. The basic issue we wish to tackle is that matrices like $K(P_N, P_M)$ are too large to compute and store.

Take a subset of m' vertices from M and n' vertices from N; choices of subsampling strategies are evaluated in §7.1. Concatenate their descriptors into a matrix $P_S \in \mathbb{R}^{(m'+n')\times d}$ and define

$$\begin{aligned} R &:= \begin{pmatrix} K(P_S,P_M) & K(P_S,P_N) \end{pmatrix} \\ R_0 &:= K(P_S,P_S). \end{aligned}$$

Note R_0 is a submatrix of R. We apply the Nyström approximation to the "extended kernel" matrix

$$\overline{K} := \begin{pmatrix} R_0 & R \\ R^\top & K\left(\begin{pmatrix} P_M \\ P_N \end{pmatrix}, \begin{pmatrix} P_M \\ P_N \end{pmatrix}\right) \end{pmatrix}$$

to derive the estimate formula

$$K\left(\begin{pmatrix} P_M \\ P_N \end{pmatrix}, \begin{pmatrix} P_M \\ P_N \end{pmatrix}\right) \approx R^\top R_0^{-1} R. \tag{17}$$

Contrasting somewhat with the notation in §6.1, here we obtained a kernel approximation for the entire set of inner products between pairs of descriptors on M, pairs of descriptors on N, *and* pairs of descriptors from the two surfaces, including those in the sample set.

Since kernel matrices are symmetric and positive definite, we use eigenvalue decomposition to write $R_0 = V\Lambda V^\top$ for an orthogonal matrix V and a diagonal matrix Λ whose diagonal is positive. Define

$$W = \begin{pmatrix} W_M \\ W_N \end{pmatrix} = R^\top V \Lambda^{-1/2},$$

where $W_M \in \mathbb{R}^{m\times(m'+n')}$ and $W_N \in \mathbb{R}^{n\times(m'+n')}$ are blocks corresponding to M and N. This leads us to the approximations:

$$\begin{aligned} K(P_M,P_M) &\approx W_M W_M^\top \\ K(P_N,P_M) &\approx W_N W_M^\top \\ K(P_N,P_N) &\approx W_N W_N^\top. \end{aligned} \tag{18}$$

These are consistent with an approximated joint kernel matrix, maintaining positive definiteness of our approximation to the algorithm in Figure 1 if they are substituted into the relevant formulas.

It is worth noting an alternative interpretation of our Nyström approximations in (18). In effect, these approximations have led to *new* $(m'+n')$-dimensional per-vertex descriptors in the rows of W_M and W_N, designed to approximate the potentially infinite-dimensional kernel lifting. Even if $m'+n'$ is larger than the dimensionality of the original descriptors, these matrices are typically full rank, as suggested in §4.4.

In the end, our conjugate gradients algorithm in Figure 1 is easily modified to include the Nyström approximation above, and as long as expressions in the resulting APPLYOPERATOR function are parenthesized properly we only require memory scaling with $(m'+n')(m+n)$ instead of $(m+n)^2$, a considerable savings as long as we choose $m', n' \sim 100-500$ relative to the full number of vertices m on M and n on N. The only implementation challenge is that the expression for $G_{D,K}$ must be implemented manually rather than using standard linear algebra libraries to avoid incurring larger space/time requirements induced by computing the Hadamard product explicitly.

7. Experiments

7.1. Kernel Approximation

We begin by evaluating the effect of the Nyström approximation in §6, to choose parameters for our experiments. Our task is to decide how many descriptors m' and n' to sample on M and N; larger m' and n' improve quality of the kernel approximation at additional computational expense. We also choose *which* descriptors to subsample in constructing (R_0, R) in (17).

As an illustrative experiment, we consider a Gaussian kernel $K_\gamma(x,y) := e^{-\gamma\|x-y\|_2^2} \in \mathbb{R}^{m\times m}$ applied to the wave kernel signature descriptor [ASC11] on the shape in Figure 2. While approximation quality depends on the informativeness of the descriptor and complexity of the shape, this example is intended to illustrate a pattern we observe across our datasets and justifies a rough conservative choice of m', n'. In this experiment, the descriptors are normalized to have a maximum value of 1.

We consider $\gamma \in \{0.01, 0.1, 1, 10\}$; small γ decreases the distance

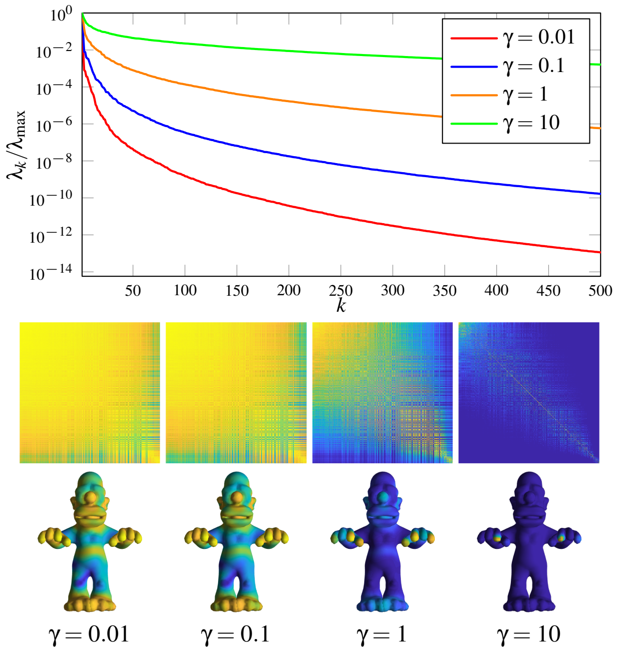

Figure 2: *(top) Spectra of the kernels for the experiment in §7.1; (bottom) visualization of the kernels as matrices over all pairs of vectors, as well as a single row as a function on the surface.*

between the different descriptors and hence makes them appear more similar. Figure 2 plots the spectra of the full kernel matrices relative to the largest eigenvalue. As expected, λ_k drops fastest when γ is small since the corresponding kernel matrix is closer to uniform.[†] The spectra lower-bound performance of *any* rank-k approximation, in the sense that the best approximation of a symmetric matrix in the Frobenius norm is given by its projection onto the space spanned by the top k eigenvectors. Even in the $\gamma = 10$ case, rank-k approximation error drops under 1% for $k \geq 300$.

The previous plot lower-bounds the best possible performance, but we do not have access to the best-possible rank-k approximation of the kernel, since computing the full kernel would be inefficient. Instead, the Nyström rank-k approximation depends on a subsampled set of rows. Figure 3 evaluates three possible strategies against the spectral lower-bound on approximation quality:

- Uniform: Choose m rows uniformly at random
- Geodesic FPS: Choose a farthest-point sample of points with respect to geodesic distance along the surface, starting with a random initializer
- Descriptor FPS: Choose a farthest-point sample in descriptor space starting with a random initializer

Each γ follows similar pattern: The "basic" Nyström uniform approach is outperformed by FPS. FPS in descriptor space slightly outperforms geodesic FPS. Based on our experiments, our remaining tests use descriptor FPS with $k = 400$; this achieves relative error of 5.8×10^{-11}, 5.1×10^{-8}, 6.9×10^{-5}, and 0.051 for our four γ values.

7.2. Kernel Parameters and Map Quality

Figure 4 shows a qualitative comparison of our kernel functional maps approach to [OBCS*12] with an increasing number of corresponding descriptor pairs (left to right). In the top row, we show the source shape and the point-to-point correspondences for [OBCS*12] using a texture map. The bottom row shows our results. We use a Gaussian kernel and set $\gamma = 500$. In both cases, we use the wave kernel signature (2, 4, 10, and 100 functions); we set $k_N = k_M = 100$ and $\alpha = 1$ in all experiments. Next to the point correspondences, we depict the geodesic distance to the ground truth correspondences for ranging from 0 (white) to 0.2 (red) in log-scale. The results show that our method requires far fewer descriptors (in our examples, the map with 2 descriptor pairs already shows good correspondence quality) than the baseline functional maps [OBCS*12] to obtain low-error point correspondences.

Figure 5 shows performance of our kernelized method pairs of shapes sampled from the FAUST dataset.[‡] Here, we show performance for Gaussian kernels of ranging parameter γ as well as a hyperbolic tangent kernel resembling the nonlinearity used in neural networks. We compute 100×100 functional maps with a rank-200 Nyström approximation and wave kernel signature/map descriptors [ASC11]. Our implementation uses the test framework code from [NO17], including their implementations of basis functions and descriptors.

We find that the most effective choice of kernel parameter is the largest value of γ. This demonstrates our intuition, that a sharp kernel can be more informative. A Gaussian kernel with large γ can increase the effective rank of the kernel matrix, more easily distinguishing points from one another than dot products between descriptors in non-kernelized space.

Figure 6 compares our method's performance on the the same dataset to that of [NO17] and [OBCS*12] for both 100 and 10 descriptors. We find that for both 100 and 10 descriptors, our method performs slightly better than [NO17] and significantly better than [OBCS*12], with the kernel parameters specified in Figure 6. As we have discussed, recall that [NO17] can be understood as a special case of our method with a trivial dot product kernel. We find that the ideal γ for our method depends on the number of descriptors ($\gamma = 500$ for 100 descriptors and $\gamma = 100$ for 10 descriptors). Parameters were chosen via grid search on a large set of parameters and narrowing down the best parameter settings for a single shape pair in FAUST. The plots are then generated for a random sample of 100 shape pairs in FAUST.

7.3. Optimization

Our tests are written in MATLAB, using a single thread. The main bottleneck of our implementation is application of the kernel matrix. While the Nyström approximation brings this operation down to linear complexity, it still takes a significant amount of CPU time;

[†] The uniform kernel $K(x,y) \equiv 1$ occurs when $\gamma = 0$ and leads to a rank-1 factorization $K = 11^{\top}$.

[‡] http://faust.is.tue.mpg.de/

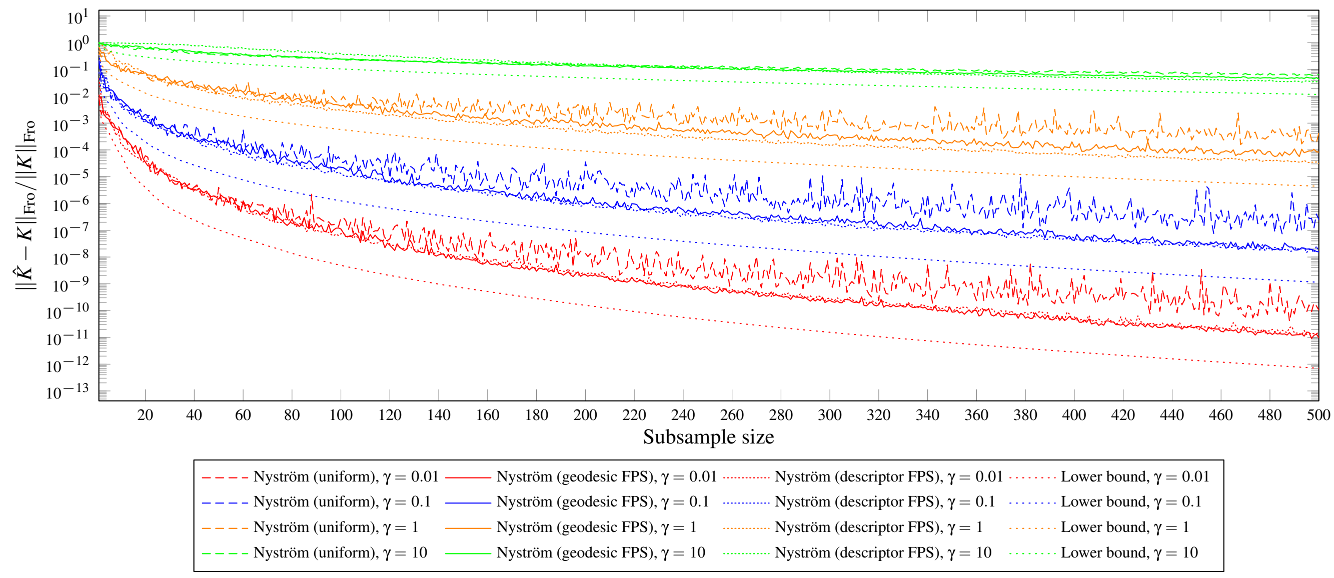

Figure 3: *Comparison of Nyström sampling strategies for example in Figure 2. The four kernel parameters are colored identically to Figure 2; line style corresponds to a sampling strategy.*

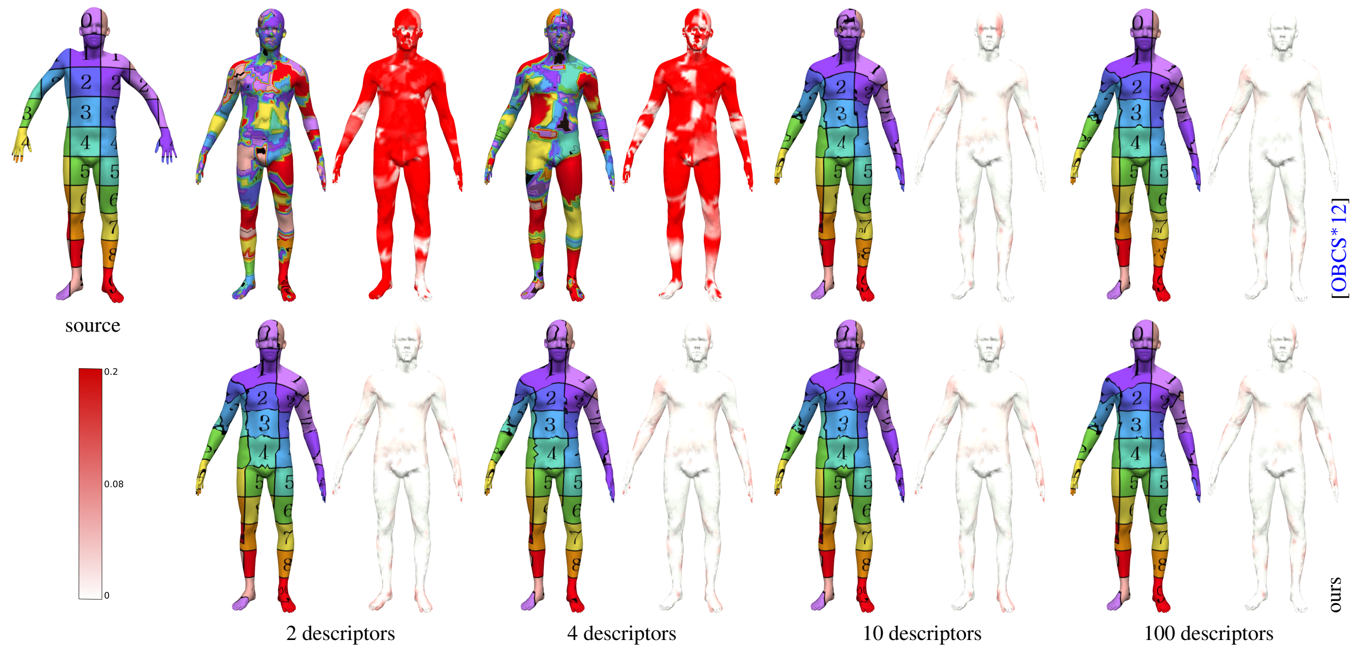

Figure 4: *Map quality and geodesic error. We compare the map quality of our method using a Gaussian kernel (bottom) to [OBCS*12] (top). The number of corresponding descriptor pairs is increased from left to right. For each target shape we show the point correspondences (left) and the geodesic error to the ground-truth (right) in a range from* 0 *(white) to* 0.2 *(red) in log-scale. With our method only few descriptor-pairs are required to obtain mappings with low geodesic error.*

we anticipate that a GPU-based implementation could be extremely successful for accelerating CG iterations, since all steps can be written in terms of simple matrix operations.

On a 2.6 GHz Intel i7 CPU with 16 GB memory, our implementation takes an average of 1.47 sec to apply the positive definite operator in CG for a mesh with 2529 vertices and a rank-400 Nyström approximation of the Gaussian kernel. Figure 7 shows convergence of the residual of the linear system for this test; note that it does not have to be monotonic for the conjugate gradient algorithm.

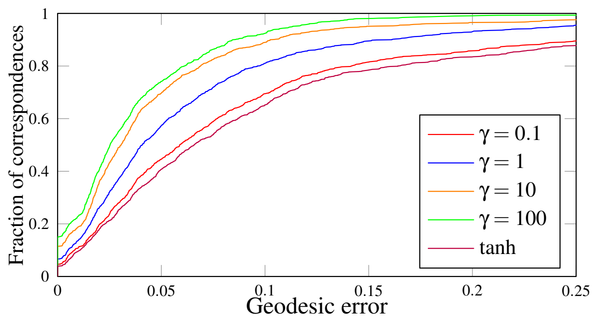

Figure 5: *Performance on FAUST dataset for different kernel parameters.*

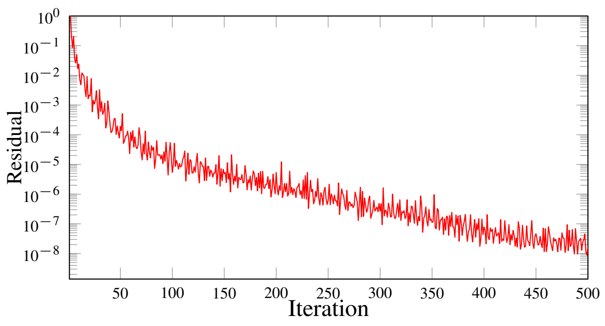

Figure 7: *Residual convergence for test in §7.3.*

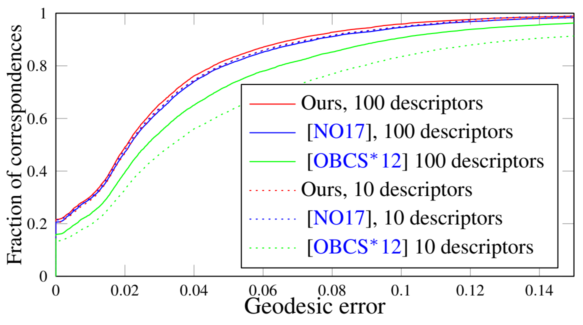

Figure 6: *Comparison to [NO17] and [OBCS* 12] on FAUST. Our methods in this experiment use 10 iterations of the CG algorithm and the Gaussian kernel. For the γ parameter of the Gaussian kernel, $\gamma = 500$ and $\gamma = 100$ for the 100 and 10 descriptor cases respectively.*

8. Discussion and Conclusion

Functional maps hold promise for efficient correspondence, but their quality strongly depends on how descriptors are computed and used. Kernelization squeezes the most value out of a fixed set of descriptors through a generalization of the basic technique. While—similarly to kernel ridge regression in machine learning—introducing a kernel incurs quadratic storage/computation cost in the simplest formulation, the Nyström approximation and iterative optimization make this generalized method scalable in practice. Our formulation and accompanying algorithm are usable "out-of-the-box" for improving functional maps; here we demonstrate how to kernelize two functional maps techniques [OBCS*12, NO17] but anticipate that the basic technique is relevant to any formulation using least-squares problems to recover functional maps.

From a theoretical perspective, our work indicates that the geometry of descriptor space is a relevant consideration in optimizing functional maps tools for quality. In effect, kernelization changes notions of proximity between descriptors in a fashion that is independent from assumptions like isometry on the geometry of the surfaces being mapped to one another. Of course, descriptors themselves are derived from a shape, and a careful characterization of the relationship between the embedding of the shape and its embedding in descriptor space—with and without kernelization—may yield insight into the best way to choose or adjust possible shape descriptors. An additional theoretical task is to evaluate the likelihood that a given kernel will be effective in recovering point-to-point maps, e.g. by verifying (approximate) preservation of pointwise products of function values [NMR*18].

Beyond its immediate relevance to existing functional maps algorithms, our work suggests several avenues for future research. While the experiments in §7 could be understood as cross-validation tests to choose the best kernel parameters for a given descriptor and class of shapes, one could also imagine an end-to-end formulation that includes the choice of kernels as a variable; our low-rank kernel approximation and optimization strategy will have to be revisited in this nonlinear regime. Perhaps the most necessary future work, however, lies in the computation of per-vertex descriptors, whose quality has strong bearing on the quality of a functional map, perhaps facilitated by machine learning tools to uncover the relevant features for a given correspondence problem. Kernelization complements this effort by enabling these tools to (1) learn just a few extremely effective descriptors and (2) generalizing the treatment of the computed descriptors to any Hilbert space.

Acknowledgments. J. Solomon thanks Polina Golland for suggesting the use of the Nyström method in this context. L. Wang was supported by the MIT EECS Research and Innovation Scholars Program. A. Gehre received funding from the German Academic Exchange Service with the scholarship *FIT Weltweit*. M. Bronstein is supported in part by the ERC Consolidator Grant No. 724228 (LEMAN), Google Faculty Research Awards, an Amazon AWS Machine Learning Research grant, an Nvidia equipment grant, a Radcliffe Fellowship at the Institute for Advanced Study (Harvard University), and a Rudolf Diesel Industrial Fellowship at the Institute for Advanced Study (TU Munich). J. Solomon acknowledges the generous support of Army Research Office grant W911NF-12-R0011 ("Smooth Modeling of Flows on Graphs"), from the MIT Research Support Committee, from the MIT–IBM Watson AI Lab, from the Skoltech–MIT Next Generation Program, and from an Amazon Research Award.

References

[ADK16] AFLALO Y., DUBROVINA A., KIMMEL R.: Spectral generalized multi-dimensional scaling. *IJCV 118*, 3 (2016), 380–392. 2

[Aiz64] AIZERMAN M. A.: Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control 25* (1964), 821–837. 2

[ASC11] AUBRY M., SCHLICKEWEI U., CREMERS D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In *Proc. ICCV Workshops* (2011), IEEE, pp. 1626–1633. 6, 7

[COC14] CORMAN É., OVSJANIKOV M., CHAMBOLLE A.: Supervised descriptor learning for non-rigid shape matching. In *Prof. NORDIA* (2014), Springer. 2

[COC*17] CHAZAL F., OVSJANIKOV M., CORMAN E., BRONSTEIN M., RODOLA E., BEN-CHEN M., GUIBAS L., BRONSTEIN A.: Computing and processing correspondences with functional maps. In *ACM SIGGRAPH 2017 Courses* (2017). 2

[CST00] CRISTIANINI N., SHAWE-TAYLOR J.: *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000. 2, 3

[ERGB16] EYNARD D., RODOLA E., GLASHOFF K., BRONSTEIN M. M.: Coupled functional maps. In *Proc. 3DV* (2016). 2

[GCR*17] GASPARETTO A., COSMO L., RODOLA E., BRONSTEIN M. M., TORSELLO A.: Spatial maps: From low rank spectral to sparse spatial functional representations. In *Proc. 3DV* (2017). 3

[HO17] HUANG R., OVSJANIKOV M.: Adjoint map representation for shape analysis and matching. *Computer Graphics Forum 36*, 5 (2017), 151–163. 2

[HWG14] HUANG Q., WANG F., GUIBAS L.: Functional map networks for analyzing and exploring large shape collections. *TOG 33*, 4 (2014), 36. 2

[KBB*13] KOVNATSKY A., BRONSTEIN M. M., BRONSTEIN A. M., GLASHOFF K., KIMMEL R.: Coupled quasi-harmonic bases. *Computer Graphics Forum 32*, 2 (2013), 439–448. 2

[KGB16] KOVNATSKY A., GLASHOFF K., BRONSTEIN M. M.: MADMM: a generic algorithm for non-smooth optimization on manifolds. In *Proc. ECCV* (2016). 2, 3

[LF00] LAI P. L., FYFE C.: Kernel and nonlinear canonical correlation analysis. *Int. J. Neural Systems 10*, 05 (2000), 365–377. 2

[LRB*16] LITANY O., RODOLÀ E., BRONSTEIN A. M., BRONSTEIN M. M., CREMERS D.: Non-rigid puzzles. *Computer Graphics Forum 35*, 5 (2016), 135–143. 2

[LRR*17] LITANY O., REMEZ T., RODOLA E., BRONSTEIN A. M., BRONSTEIN M. M.: Deep functional maps: Structured prediction for dense shape correspondence. In *Proc. ICCV* (2017), vol. 2, p. 8. 2

[NAS16] NEMTSOV A., AVERBUCH A., SCHCLAR A.: Matrix compression using the Nyström method. *Intelligent Data Analysis 20*, 5 (2016), 997–1019. 6

[NMR*18] NOGNENG D., MELZI S., RODOLÀ E., CASTELLANI U., BRONSTEIN M., OVSJANIKOV M.: Improved functional mappings via product preservation. *Computer Graphics Forum* (2018). 1, 2, 3, 9

[NO17] NOGNENG D., OVSJANIKOV M.: Informative descriptor preservation via commutativity for shape matching. *Computer Graphics Forum 36*, 2 (2017), 259–267. 1, 2, 3, 5, 7, 9

[Nys30] NYSTRÖM E. J.: Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica 54*, 1 (1930), 185–204. 2

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional maps: a flexible representation of maps between shapes. *TOG 31*, 4 (2012), 30. 1, 2, 3, 5, 7, 8, 9

[PBB*13] POKRASS J., BRONSTEIN A. M., BRONSTEIN M. M., SPRECHMANN P., SAPIRO G.: Sparse modeling of intrinsic correspondences. *Computer Graphics Forum 32*, 2 (2013), 459–468. 2, 3

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental mathematics 2*, 1 (1993), 15–36. 2

[RCB*17] RODOLÀ E., COSMO L., BRONSTEIN M. M., TORSELLO A., CREMERS D.: Partial functional correspondence. *Computer Graphics Forum 36*, 1 (2017), 222–236. 2, 3

[SGV98] SAUNDERS C., GAMMERMAN A., VOVK V.: Ridge regression learning algorithm in dual variables. *Proc. ICML 15* (1998). 2

[SOG09] SUN J., OVSJANIKOV M., GUIBAS L.: A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum 28*, 5 (2009), 1383–1392. 3

[SSM97] SCHÖLKOPF B., SMOLA A., MÜLLER K.-R.: Kernel principal component analysis. In *Proc. ICANN* (1997), Springer, pp. 583–588. 2

[VKZHCO11] VAN KAICK O., ZHANG H., HAMARNEH G., COHEN-OR D.: A survey on shape correspondence. *Computer Graphics Forum 30*, 6 (2011), 1681–1707. 2

[WS01] WILLIAMS C. K., SEEGER M.: Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems* (2001), pp. 682–688. 2, 6