

# PARALLEL TRANSPORT CONVOLUTION: DEFORMABLE CONVOLUTIONAL NETWORKS ON MANIFOLD-STRUCTURED DATA

STEFAN C. SCHONSHECK <sup>\*</sup>, BIN DONG <sup>†</sup>, AND RONGJIE LAI <sup>‡</sup>

**Abstract.** Convolution has played a prominent role in various applications in science and engineering for many years and has become a key operation in many neural networks. There has been a recent growth of interest in generalizing convolutions on 3D surfaces, often represented as compact manifolds. However, existing approaches cannot preserve all the desirable properties of Euclidean convolutions, namely: compactly supported filters, directionality, transferability across different manifolds. This paper develops a new generalization of the convolution operation, referred to as parallel transport convolution (PTC), on Riemannian manifolds and their discrete counterparts. PTC is designed based on parallel transportation that can translate information along a manifold and intrinsically preserve directionality. Furthermore, PTC allows for the construction of compactly supported filters and is also robust to manifold deformations. This enables us to perform wavelet-like operations and to define convolutional neural networks on curved domains.

**Key words.** Convolution Neural Networks, Parallel Transport, Shape Analysis, Manifold Learning

**AMS subject classifications.** 68U05, 65D18, 68T45

**1. Introduction.** Convolution is a fundamental mathematical operation that arises in many applications in science and engineering. Its ability to effectively extract local features, as well as its ease of use, has made it the cornerstone of many important techniques such as numerical partial differential equations and wavelets [11, 27, 31]. More recently, convolution plays a fundamentally important role in convolutional neural networks (CNN) [27] which have made remarkable progress and significantly advanced the state-of-the-art in image processing, analysis and recognition [27, 1, 24, 9, 18, 38, 29, 41].

In Euclidean space  $\mathbb{R}^n$ , the convolution of a function  $f$  with a kernel (or filter)  $K$  is defined as:

$$(1.1) \quad (f * K)(x) := \int_{\mathbb{R}^n} K(x - y)f(y)dy.$$

Unlike signals or images whose domain is shift invariant (such as images in the plane), functions defined on curved domains do not always have shift-invariance. To define robust convolutional operators on these curved domains, the key challenge is to properly define the translation operation. This is one of the main obstacles of generalizing CNN to manifolds.

There has been a recent surge of research in designing CNNs on manifolds or graphs. We refer the interested readers to [5] for a review of recent progress in this area. These approaches can be classified into three categories: spectral, patch-based and group action methods. Spectral methods are based on projecting a signal onto the eigen (Fourier) space and using the convolution theorem to define convolution.

---

<sup>\*</sup>Department of Mathematics, Rensselaer Polytechnic Institute, Troy, NY 12180, U.S.A. (schon@rpi.edu).

<sup>†</sup>Beijing International Center for Mathematical Research (BICMR), Peking University, Beijing China (dongbin@math.pku.edu.cn).

<sup>‡</sup>Corresponding author. Department of Mathematics, Rensselaer Polytechnic Institute, Troy, NY 12180, U.S.A. (lair@rpi.edu). R.Lai's research is supported in part by an NSF CAREER Award, DMS-1752934.

Patch-based methods use a patch operator to interpolate local geodesic discs on a certain given template. Group action-based methods are defined on homogeneous space with transitive group action. Here, we briefly review some of these approaches.

Spectral methods for manifold convolutions are based on the Fourier transform. The convolution theorem states that, for any two functions  $f$  and  $g$ :  $\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$  where  $\mathcal{F}$  is the Fourier transform, and  $\cdot$  denotes pointwise multiplication. This theorem can be naturally generalized to functions on manifolds using the Laplace-Beltrami (LB) eigensystem. This method has proven effective to handle functions on a fixed domain and can be applied to graphs as well [16, 6, 12, 17]. However, these methods have two fundamental limitations. First, the uncertainty principle states that a function can have compact support in either the time or frequency domain, but not both. These methods normally use only a finite number of eigenfunctions in the spectral domain. As a result, the kernels that arise from these methods are not localized (i.e. not compactly supported in the spatial domain). The second major drawback to these methods is that since they rely on the LB eigensystem of the domain, any deformation of the domain will change the LB eigensystem, which in turn changes the filters. The high-frequency LB eigenfunctions of a manifold are extremely sensitive to even small deformations. This means that anything designed for or learned on one manifold can only be applied to problems defined on the same domain. This limits the transferability of the spectral-based methods and makes them inefficient for working on large collections of shapes, although some recent work has been done to overcome some of these drawbacks [15, 4].

Patch-based methods were originally proposed in [32]. In this work, the authors propose using a local patch operator to interpolate local geodesic discs of the manifold to a fixed template and develop a Geodesic Convolutional Neural Network (GCNN). Then for each point on the manifold, the convolution is calculated as the multiplication between the values of the kernel and the extracted patch on the template. To do so, they create a local polar coordinate system at each point. One drawback to this approach is that there is no natural way to choose the origin of the polar coordinate. To overcome this, the authors consider an angular pooling operation that evaluates all rotations of their kernel at each point and selects the orientation which maximizes the convolution in a point-wise fashion. However, since the angular pooling operation is computed independently at each point, the selected orientation does not reflect the geometric structure of the base manifold and may not be consistent even for nearby points. More recently, [3] proposes an anisotropic convolutional neural network (ACNN) by replacing the aforementioned patch operator with an operator based on anisotropic heat kernels with the direction of anisotropy fixed on the principle curvature at each point. Although this introduces a new hyper-parameter (the level of anisotropy), it allows the kernels to be directionally aware. However, filters developed for applications on one manifold can only be applied to manifolds in which the local directions of principal curvature are the same. In [34], the authors proposed a mixture model network (MoNet) whereby they learn a patch operator to interpolate the functional value to a template. The convolution kernel is set to be a Gaussian function with learnable mean and covariance matrices. Note that MoNet requires a choice of local coordinates that may suffer from the same drawback as GCNN and ACNN.

Group action-based methods are recently discussed in [23, 10, 7]. A typical application of these methods is to extend CNN on the unit sphere [10], where convolutional operations can be defined by transferring kernels on the unit sphere through the rotation group. This idea can be generalized to a manifold  $\mathcal{M}$  with a transitive group

action  $G$ , where any two points  $p, q \in \mathcal{M}$  can be connected by some group element, i.e. there exists  $g \in G$  such that  $p = g \cdot q$ . In this setting, the manifold is called a homogeneous space which is essentially equivalent to a quotient group  $G/G_p$  where  $G_p$  is the stabilizer of the group action at  $p$ . However, the general manifolds considered in this paper often do not have an associated transitive group action. Therefore, it is still necessary to consider a new method to apply convolution on manifolds without group action structure.

Method	Filter Type	Support	Directional	Transferable	Deformable
Spectral [6]	Spectral	Global	✓	✗	✗
TFG [12]	Spectral	Global	✓	✗	✗
WFT [40]	Spectral	Local	✓	✗	✗
GCNN [32]	Patch	Local	✗	✓	✓
ACNN [3]	Patch	Local	✓	✓	✗
MDGEC [35]	Patch	Local	✓	✓	✗
<b>PTC</b>	Geodesic	Local	✓	✓	✓

TABLE 1

*Comparison on different generalizations of convolutional operator on general manifolds.*

In the Euclidean setting, convolution operators frequently used in practice have compactly supported filters that allow for fast and efficient computations on both CPUs and GPUs. Furthermore, they are directionally aware, deformable, and can be easily transferred from one signal domain to another. Previous attempts to generalize the convolution operator on manifolds have failed to preserve one or more of these key properties. In this paper, we propose a new way of defining the convolution operation on manifolds based on parallel transportation. We shall refer to the proposed convolution as the parallel transportation convolution (PTC). The proposed PTC is able to preserve all of the aforementioned key proprieties of Euclidean convolutions. This spatially defined convolution operation enjoys the flexibility of conducting isotropic or anisotropic diffusion, and it also enables us to perform wavelet-like operations and create convolutional neural networks on manifolds. Additionally, we show that PTC simplifies to the Euclidean convolution when the underlying domain is flat. Therefore, the PTC can be used to define natural generalizations of common Euclidean convolution-like operations on manifolds.

To be more precise, we seek a general convolution operator of the form:

$$(1.2) \quad (f *_{\mathcal{M}} K)(x) := \int_{\mathcal{M}} K(x, y) f(y) d_{\mathcal{M}} y.$$

where  $K(x, \cdot)$  is the parallel transport of a compact support kernel  $K(x_0, \cdot)$  to  $x$ . In the Euclidean case (1.1), the term  $x - y$  encapsulates the direction from  $x$  to  $y$ , while on a manifold such a vector can be understood as a tangent direction at  $x$  pointing to  $y$ . The crucial idea of PTC is to define a kernel function  $K(x, y)$  which is able to encode the direction  $x - y$  using parallel transportation in a way incorporating the manifold structure naturally.

Table 1 compares the proposed PTC with previous approaches. Since the group action methods are limited to homogenous spaces, which do not fit our objective of designing convolution on more general manifolds, we do not include these methods in the table. A method is called directional if the filters are able to characterize non-isotropic features of the data. A method is called transferable if the filters can be

learned on a manifold and then applied to another consistently. Spectral methods are not transferable as manifold deformations or re-ordering of Laplace-Beltrami eigensystems cause issues, and graph-based methods fail as they are sensitive to samples even from the same surface. Finally, a technique is called deformable if non-isometric deformations in the manifold (i.e. those which change properties such as curvature or local distances) do not drastically affect the convolution. Methods which rely on spectral convolutions are not deformable since the LB eigensystem is unstable and the ACNN [3] is not deformable as small deformations (such as those studied in 6.3) change the orientation of the filters.

The method of multi-directional geodesic equivariant convolution [35] is also based on parallel transportation but has a few drawbacks which can be overcome by our method. Their method requires the use of so-called angular max pooling. This may be overly sensitive to small deformations in the data, and it may also introduce discontinuities in the orientation of the filters. Thus, their network architectures need many more channels and pooling layers to achieve similar performance. The method in [43] also involves parallel frames via the locally flat connections to define convolution-like operations. This method attempts to overcome the problem of singularities in the field using some smoothness conditions to move the singularities to geometrically significant points. Alternatively, our method is based on solving the Eikonal equation rather than smoothing specific approximated solutions. We use multiple vector fields as discussed in Remark 4.1 to overcome the problem of singularities. Furthermore, we would also like to remark that the arXiv version of our work predates that of both aforementioned papers.

The rest of this paper is organized as follows. In section 2, we discuss the necessary mathematical background of differential manifolds and parallel transportation of vector fields on manifolds. Then, we introduce the proposed PTC for both smooth and discretized manifolds in section 3. In section 4, we introduce notions of strided, transposed convolutions and convolutional neural networks on manifolds and also discuss how to implement these tools in practice. In section 5, several numerical experiments illustrate the effectiveness of the proposed method. Finally, concluding remarks are made in section 6.

**2. Mathematical Background.** In this section, we discuss some background of differential manifolds and parallel transportation. This provides a motivation and theoretical preparation for the proposed convolutional operation.

**2.1. Manifolds, Tangent Spaces and the Exponential Map.** Let  $\mathcal{M}$  be a two dimensional differential manifold associated with a metric  $g_{\mathcal{M}}$ . For simplicity we assume that  $(\mathcal{M}, g_{\mathcal{M}})$  is embedded in  $\mathbb{R}^3$ . We write the set of all tangent vectors at any point  $x \in \mathcal{M}$  as  $\mathcal{T}_x\mathcal{M}$  which we refer to as the tangent plane of  $\mathcal{M}$  at  $x$ . The disjoint union of all tangent planes,  $\bigsqcup_x \{(x, v) \in \mathcal{M} \times \mathbb{R}^3 \mid x \in \mathcal{M}, v \in \mathcal{T}_x\mathcal{M}\}$ , forms a four dimensional differential manifold called the tangent bundle  $\mathcal{TM}$  of  $\mathcal{M}$ . A vector field  $X$  is a smooth assignment  $X : \mathcal{M} \rightarrow \mathcal{TM}$  such that  $X(x) \in \mathcal{T}_x\mathcal{M}$ ,  $\forall x \in \mathcal{M}$ . We denote the collection of all smooth vector fields on  $\mathcal{M}$  as  $C^\infty(\mathcal{M}, \mathcal{TM})$ .

Let  $\mathcal{T}_{x,\delta}\mathcal{M} = \{v \in \mathcal{T}_x\mathcal{M} \mid \langle v, v \rangle_{g_{\mathcal{M}}} \leq \delta\}$  be a  $\delta$ -neighborhood of the tangent space at a given point  $x$ . The *exponential map*,  $\exp : \mathcal{T}_{x,\delta}\mathcal{M} \rightarrow \mathcal{M}_{x,\delta}$ , maps vectors from the tangent space back onto a nearby region  $\mathcal{M}_{x,\delta} = \{y \in \mathcal{M} \mid d_{\mathcal{M}}(x, y) \leq \delta\}$  of  $x$  on the manifold. Formally, given  $v \in \mathcal{T}_{x,\delta}\mathcal{M}$  there exists a unique geodesic curve  $\gamma$  with  $\gamma(0) = x$  and  $\gamma'(0) = v$  such that  $\exp_x(v) = \gamma(1)$ . Note that this map is defined in the local neighborhood where the differential equation:  $\gamma'(0) = v$  with initial

condition  $\gamma(0) = x$  has a unique solution. The size of this neighborhood depends on the local geometry of the manifold. In fact, the exponential map defines a one-to-one correspondence between  $\mathcal{T}_{x,\delta}\mathcal{M}$  and  $\mathcal{M}_{x,\delta}$  if  $\delta$  is smaller than the injective radius of  $\mathcal{M}$  [22, 8]. Since this map is a bijection, there is a natural inverse (sometimes called the *logistic map*) which we denote as  $\exp_x^{-1} : \mathcal{M}_{x,\delta} \rightarrow \mathcal{T}_{x,\delta}\mathcal{M}$ .

**2.2. Parallel Transportation.** Parallel transportation is a method of translating a vector, based on an affine connection, along a smooth curve so the resulting vector is ‘parallel’. An affine connection translates the tangent spaces of points on a manifold in a way that allows us to differentiate vector fields along curves. Formally, an *affine connection* is a bilinear map  $\nabla : C^\infty(\mathcal{M}, \mathcal{T}\mathcal{M}) \times C^\infty(\mathcal{M}, \mathcal{T}\mathcal{M}) \rightarrow C^\infty(\mathcal{M}, \mathcal{T}\mathcal{M})$ , such that for all smooth functions  $f, g$  and all vector fields  $X, Y, Z$  on  $\mathcal{M}$  satisfy:

$$(2.1) \quad \begin{cases} \nabla_{fX+gY}Z = f\nabla_XZ + g\nabla_YZ \\ \nabla_X(aY+bZ) = a\nabla_XY + b\nabla_XZ \quad a, b \in \mathbb{R} \\ \nabla_X(fY) = df(X)Y + f\nabla_XY \end{cases}$$

In particular, an affine connection is called the Levi-Civita connection if it is torsion free ( $\nabla_XY - \nabla_YX = [X, Y]$ ) and compatible with the metric ( $X\langle Y, Z \rangle_{g_{\mathcal{M}}} = \langle \nabla_XY, Z \rangle_{g_{\mathcal{M}}} + \langle Y, \nabla_XZ \rangle_{g_{\mathcal{M}}}$ ). In this case, the transport induced by the connection preserves both the length of the transported vector and the angle with the path that it is transported along.

A curve  $\gamma : [0, \ell] \rightarrow \mathcal{M}$  on  $\mathcal{M}$  is called geodesic if  $\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) = 0$ . More precisely, using local coordinate system, we can write  $\dot{\gamma}(t) = \sum_{i=1}^2 \frac{dx^i}{dt} \partial x^i$ , then plugging in the covariant derivative leads to the following ordinary differential equation for a geodesic curve:

$$(2.2) \quad \frac{d^2x^k(t)}{dt^2} + \sum_{i,j=1}^2 \Gamma_{ij}^k \frac{dx^i(t)}{dt} \frac{dx^j(t)}{dt} = 0, \quad k = 1, 2$$

where  $\Gamma_{i,j}^k$  is the Christoffel symbols associated with the local coordinate system. For any two points  $x_0$  and  $x_1$  on a complete manifold  $\mathcal{M}$ , there will be a geodesic  $\gamma : [0, \ell] \rightarrow \mathcal{M}$  connecting  $x_0$  and  $x_1$ . A vector field  $X(t)$  on  $\gamma(t)$  is called *parallel* if  $\nabla_{\dot{\gamma}}X = 0$ . Therefore, given any vector  $v \in \mathcal{T}_{x_0}\mathcal{M}$ , we can transport  $v$  to a vector  $v'$  in  $\mathcal{T}_{x_1}\mathcal{M}$  by defining  $v' = X(\ell)$  from the solution of the initial value problem  $\nabla_{\dot{\gamma}(t)}X(t) = 0$  with  $X(0) = v$ . In other words, if we write  $X(t) = \sum_{i=1}^2 a^i(t) \partial x^i$ , the problem of solving  $X$  reduces to find the appropriate coefficients  $\{a^k(t)\}$  satisfying the parallel transport equation. This can be written as the following first order linear system:

$$(2.3) \quad \begin{cases} \frac{da^k(t)}{dt} + \sum_{i,j=1}^2 \frac{d\gamma^i}{dt} a^j(t) \Gamma_{ij}^k = 0, & k = 1, 2 \\ \sum_{i=1}^2 a^i(0) \partial x^i = v \end{cases}$$

Solving this equation finds a parallel vector field  $X$  along  $\gamma(t)$  which provides parallel transportation of  $v = X(0) \in \mathcal{T}_{x_0}\mathcal{M}$  to  $X(\ell) \in \mathcal{T}_{x_1}\mathcal{M}$ . We denote the parallel transportation of a vector from  $x_0$  to  $x_1$  along the geodesic as  $\mathcal{P}_{x_0}^{x_1} : \mathcal{T}_{x_0,\delta}\mathcal{M} \rightarrow \mathcal{T}_{x_1,\delta}\mathcal{M}$ .

**3. Parallel Transport Convolution (PTC).** In this section, we introduce parallel transport convolution on manifolds which provides a fundamental important building block of designing convolutional neural networks on manifolds. After that, we shall propose a numerical discretization of PTC.

**3.1. Mathematic definition of PTC.** Unlike one-dimensional signals or images whose base space is shift invariant, many interesting geometric objects modeled as curved manifolds do not have shift-invariance. This is an essential barrier to adopting CNNs to conduct learning on manifolds and graphs except for a few recent works where convolution is defined in the frequency space of the LB operator [6, 39, 36, 37]. However, these methods only manipulate the LB eigenvalues by splitting the high dimension information to LB eigenfunctions. Limitations include that it is always isotropic due to the LB operator and can only approximate the even-order differential operators [12]. In addition, there is another recent method discussed in [32], in which convolution is directly considered on the spatial domain using local integral on geodesic disc, although it does not involve manifold structure as transportation on manifolds is not considered. The lack of an appropriate method of defining convolution on manifolds motivates us to introduce the following way of defining convolution on manifolds through parallel transportation. This geometric way of defining convolution naturally integrates manifold structures and enables us to apply established learning techniques in Euclidean domains to their non-euclidean counterparts.

Let  $K(x_0, \cdot) : \mathcal{M}_{x_0, \delta} \rightarrow \mathbb{R}$  be a compactly supported kernel function centered at  $x_0$  with radius  $\delta$ . We assume  $K(x_0, y) = 0$  for  $y \notin \mathcal{M}_{x_0, \delta}$  and require the radius of the compact support parameter  $\delta$  be smaller than the injective radius of  $\mathcal{M}$  to guarantee the bijectivity of the exponential map. Note that this is a very mild assumption, since most modern CNN architectures use filters which are much smaller than the entire image. It is also important to remark that parameterization of  $K(x_0, \cdot)$  can be determined by user. It may be hand designed for specific applications, or be learned as a component of a neural network.

Our idea of defining convolution on manifolds relies on transporting this compactly supported kernel  $K(x_0, \cdot)$  to every other point on  $\mathcal{M}$  in a way of reflecting the manifold geometry. More specifically, given any point  $x \in \mathcal{M}$ , we first construct a vector field transportation  $\mathcal{P}_{x_0}^x : \mathcal{T}_{x_0, \delta} \mathcal{M} \rightarrow \mathcal{T}_{x, \delta} \mathcal{M}$  using the parallel transportation discussed in Section 2.2. Then  $K(x_0, \cdot)$  can be transported on  $\mathcal{M}$  as:

$$(3.1) \quad \begin{aligned} K(x, \cdot) : \mathcal{M}_{x, \delta} &\rightarrow \mathbb{R} \\ y &\mapsto K\left(x_0, \exp_{x_0} \circ (\mathcal{P}_{x_0}^x)^{-1} \circ \exp_x^{-1}(y)\right) \end{aligned}$$

Note that the above definition is analogous to convolution in the Euclidean space (1.1). Here, the exponential map  $\exp_x^{-1}(y)$  mimics the vector  $x - y$ , and  $\mathcal{P}_{x_0}^x$  is a generalizes the translation operation. In fact, it can be easily checked that the above definition is compatible with Euclidean case by setting the manifold  $\mathcal{M}$  to be  $\mathbb{R}^2$ .

By plugging (3.1) into (1.2), we can now formally define the *parallel transport convolution* operation of  $f$  which a filter  $k$ , centered at  $x_0$ :

$$(3.2) \quad \begin{aligned} (f *_{\mathcal{M}} K)(x) &:= \int_{\mathcal{M}} f(y) K(x, y) \, d_{\mathcal{M}} y \\ &= \int_{\mathcal{M}} f(y) K\left(x_0, \exp_{x_0} \circ (\mathcal{P}_{x_0}^x)^{-1} \circ \exp_x^{-1}(y)\right) \, d_{\mathcal{M}} y \end{aligned}$$

As natural extensions, this approach can also be used to define dilations, reflections and rotations of the kernel by simply manipulating the reference vector  $\exp_x^{-1}(y)$ .

More specifically, shrinking or expanding the kernel by a factor of  $s$  is defined by multiplying the lengths of the vectors in the tangent space by  $s$ . If  $s$  is chosen to be negative then the kernel is reflected through its center and dilated by a factor of  $|s|$ . Similarly, rotating the kernel can be achieved by multiplying a rotation matrix  $R_\theta$  to the reference vectors on the tangent plane. In summary, the scaling of  $K$  by  $s$  with a rotation of  $\theta$  is defined as:

$$(3.3) \quad K_{s,\theta}(x, y) := \frac{1}{C_x} K\left(x_0, \exp_{x_0} \circ (\mathcal{P}_{x_0}^x)^{-1}(s R_\theta \exp_x^{-1}(y))\right)$$

where  $R_\theta$  is a rotation matrix and  $\frac{1}{C_x}$  is a normalization constant that can be used to preserve volume of the kernel.

**THEOREM 3.1.** *Parallel transport convolution is invariant under isomorphism.*

*Proof.* By definition isomorphisms preserve the Riemannian metric and therefore distances and geodesic paths. Then both the paths  $\mathcal{P}_{x_0}^x$  and the metric  $d_{\mathcal{M}}y$  are invariant so is (3.2).  $\square$

**3.2. Numerical Discretization.** In stead of solving the system of ODEs (2.3) on manifolds, we novelly propose the following method to compute parallel transport by considering transition matrices among local frames generated by the vector field obtained from the distance function on manifolds. Our idea is motivated from the following fact. Given smooth vector fields  $\{\vec{b}^1, \vec{b}^2\}$ , one can define linear transformation among tangent planes  $\mathcal{L}(\gamma)_s^t : \mathcal{T}_{\gamma(s)}\mathcal{M} \rightarrow \mathcal{T}_{\gamma(t)}\mathcal{M}$ , then the corresponding parallel transport through the associated infinitesimal connection  $\nabla_{\dot{\gamma}}V = \lim_{h \rightarrow 0} \frac{1}{h}(\mathcal{L}(\gamma)_0^h(V_{\gamma(0)}) - V_{\gamma(0)})$  can be induced [21]. Therefore, construction of parallel transport is essentially equivalent to design vector fields on manifolds.

For convenience, we represent a two-dimensional manifold  $\mathcal{M}$  using triangle mesh  $\{V, E, T\}$ . Here  $V = \{v_i \in \mathbb{R}^3\}_{i=1}^n$  denotes vertices and  $T = \{\tau_s\}_{s=1}^l$  denotes faces. For each triangle  $\tau_s$ , we construct a local orthonormal frame  $\mathfrak{F}_s = \{\vec{b}_s^1, \vec{b}_s^2, \vec{n}_s\}$  where  $\vec{b}_s^1, \vec{b}_s^2$ , reflecting the intrinsic information, are tangent to  $\tau_s$ , and  $\vec{n}_s$ , reflecting the extrinsic information, is orthogonal to  $\tau_s$ . For an edge adjacent with  $\tau_s$  and  $\tau_t$ , we write  $R_{st}$  as an orthonormal transition matrix such that  $R_{st}\mathfrak{F}_t = \mathfrak{F}_s$ . Then any vector in  $\text{Span}\{\vec{b}_s^1, \vec{b}_s^2\}$  can be transported to  $\text{Span}\{\vec{b}_t^1, \vec{b}_t^2\}$  using the transition matrix  $R_{st}$ . This can be viewed as a discretization of connection and used to transport a vector on the tangent space of one given point to all other points. The compatibility condition of all  $R_{st}$  discussed in [42] can guarantee that no ambiguity will be introduced in this way. We remark this idea can be also be used for manifolds represented as point clouds by combining with the local mesh method developed in [26].

After the transportation is conducted, the convolution kernel can be transported to a new point by interpolating the transported vectors in the local tangent space at the target point. Computationally, we define a sparse matrix  $K$  where the  $i^{th}$  column is the transportation of the kernel to the  $i^{th}$  vertex. Thus, we have the following definition of discrete parallel transport convolution:

$$(3.4) \quad (f *_{\mathcal{M}} K)(x) := K^T \mathbf{M} F$$

where  $F$  is column vector representation the function  $f$  at each vertex and  $\mathbf{M}$  is the mass matrix. Note that once we have computed the vector field of the geodesic equation, the transportation of the kernel to each new center and multiplication with  $F$  is independent and can, therefore, be parallelized efficiently. Additionally, by discretizing the kernel function  $K$  as a fixed stencil, we can precompute the transportation



and interpolation of the stencil once before training. Then, PTC can be computed very efficiently using sparse matrices products. We provide detailed implementation about computing these sparse matrices in Appendix A.

Figure 1 illustrates the effect of the proposed method of transporting a kernel function on a manifold. This result shows that the proposed method produces an analogy of the behavior of a kernel function  $K(x - y)$  operating in the Euclidean domain. More importantly, we would like to emphasize that number of degrees of freedom in PTC is essentially the same as the classical convolution on the Euclidean domain.

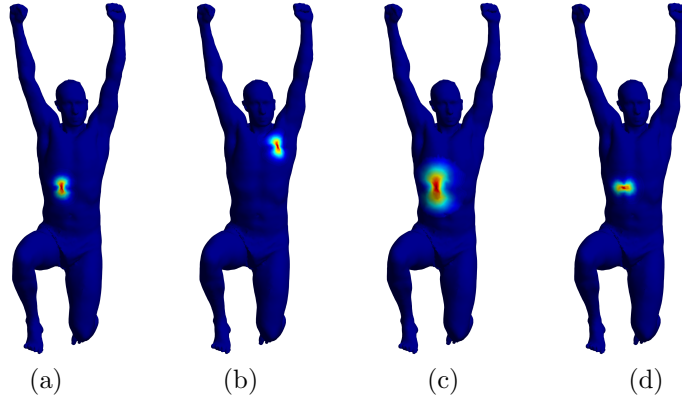


FIG. 1. A compactly supported kernel (a) is transported on a manifold from the FAUST data set [2] through translation (b), translation + dilation (c) and translation + rotation (d).

**4. Convolutional Neural Networks on Manifolds.** In this section, we discuss two more important ingredients in common CNN architectures: stride and transposed convolution and discuss. After that, we describe basic layers for CNNs on manifolds.

**4.1. Strided PTC.** In the discrete Euclidean setting, the *stride* of a convolution is the distance, usually measured in pixels, in which the kernel is translated on the image between each multiplication with the images [13]. The numerical discretization of PTC presented so far evaluates the transported kernel  $K$  at each point on the discretized point cloud (or vertex of the mesh). When the manifold is uniformly sampled, this results in a consistent distance between centers of the transported patch, and therefore a consistent stride. However, when the surface is discretized with inconsistent sampling, the distance between evaluation points will also be inconsistent. We overcome this inconsistency by including the mass matrix into the discrete PTC formulation (3.4), which normalizes the integral by the size of the local area elements.

Our proposed strided PTC formulation is based on the following observation: A Euclidean strided convolution is evaluated by transporting a kernel to an ‘evenly spaced’ subset of points from the euclidean domain. If conducted without padding, then this creates a contracted information and the resulting output of the convolution is both more compact (information from pixels that are far apart in the input become closer in the output) and smaller (in the number of total number of pixels) than the input. To mimic this effect, we compute a heretical sub-sampling of the mesh (sometimes called mesh coarsening) through the farthest point sampling (FPS) [33] method. Each level of subsampling corresponds to each level of strided convolution.



Let the original discrete manifold be represented as a set of points  $\mathcal{M}_0$ , and each hierarchical sub-sampling be computed such that  $\mathcal{M}_0 \supset \mathcal{M}_1 \supset \mathcal{M}_2 \dots \supset \mathcal{M}_k$ . Then the convolution from  $\mathcal{M}_i$  to  $\mathcal{M}_{i+1}$  can be defined as:

$$(4.1) \quad (f *_{\mathcal{M}_i \downarrow \mathcal{M}_{i+1}} K)(x) = \sum_{z \in \mathcal{M}_i} K(x, z) f(z) m_i(z) \quad \forall x \in \mathcal{M}_{i+1}$$

where  $m_i(z)$  is the local mass element at  $z$  from the  $\mathcal{M}_i$  level of sampling. These mass elements can be recomputed from the sub-sample point-cloud  $\mathcal{M}_i$  or can be aggregated by assigning each of the mass elements from the  $\mathcal{M}_{i-1}$ th sampling to its nearest neighbor in  $\mathcal{M}_i$  sampling.

**4.2. Transposed PTC.** Transposed convolution is often viewed as the opposite (or more formally as the adjoint) of strided convolution as it is a convolution that expands the size of the input. In the Euclidean setting, this is achieved by padding a signal (most often with zeros) then performing convolution with a fixed filter. This operation leads to a dilation of information. To mimic this operation, we reverse the subsampling scheme presented in 4.1 and define a convolution that takes signals from the  $i^{th}$  level to the  $(i-1)^{th}$  level. Given a signal  $f$  defined on  $\mathcal{M}_i$  and a kernel  $K$ , the transposed convolution of  $f$  from  $\mathcal{M}_i$  to  $\mathcal{M}_{i-1}$  is defined as:

$$(4.2) \quad (f *_{\mathcal{M}_i \uparrow \mathcal{M}_{i-1}} K)(x) = \sum_{z \in \mathcal{M}_{i-1}} K(x, z) f(z) m_{i-1}(z) \quad \forall x \in \mathcal{M}_{i-1}$$

To achieve this, we need to extend  $f$  to all of the points in  $\mathcal{M}_{i-1}$ . This can be done either through zero padding, analogous to the most common Euclidean operations or via harmonic extension. In either case, once the function  $f$  is well defined on the up-sampled mesh, the convolution is as simple as plugging in the correct mass matrix into equation (3.4).

**4.3. Convolutional neural networks on manifolds through PTC.** Using the proposed PTC, we can define convolutional neural networks on manifolds. We shall refer to these networks as PTCNets. Similar as CNNs on Euclidean domains, a PTCNet consists of an input and an output layer, as well as multiple hidden layers including fully connected layers, nonlinear layers, pooling layers and PTC layers listed as follows.

- **Fully Connected:**  $f_i^{out}(x) = \sum_{j=1}^N w_{ij} f_j^{in}(x)$ ,  $i = 1, \dots, L$ . This layer connects every neuron in one layer to every neuron in the previous layer. The coefficient matrix  $(w_{ij})$  parameterizes this layer and will be trained by a training data set.
- **Vector Connected (VC):**  $f_i^{out} = \sum_j +1^n w_j f_j^{in}$ . This layer linearly combines channels independent of the ordering of the discretization of points. This can also be thought of as a special case of the fully connected layer, in which each column of the weight matrix is a constant.
- **ReLU:**  $f_i^{out}(x) = \max\{0, f_i^{in}(x)\}$ ,  $i = 1, \dots, L$ . This is a fixed layer applying the nonlinear Rectified Linear Units function  $\max\{0, x\}$  to each input.
- **PTC:**  $f_{i,\alpha}^{out}(x) = \int K_\alpha(x, y) f_i^{in}(y) dy \approx K_\alpha \mathbf{M} F_i^{in}$ ,  $\alpha = 1, \dots, m$ . This layer applies the proposed PTC to the input, passes the result to the next layer. By choosing the correct mass matrix, these convolutions can be strided or transposed. Each  $K_\alpha$  is determined by the proposed PTC on manifolds with an initial convolution kernel  $K_\alpha(x_0, \cdot)$ , which parametrize the parallel transport convolution process and will be learned based on a training data set.

- **Vector Field Pooling:**  $f_i^{out}(x) = \max_{\alpha} f_{i,\alpha}^{in}(x)$ . The pooling layer can be implemented using several non-linear functions, among which the max-pooling is the most common way. By pooling over multiple vector fields, we can avoid troubles caused by singularities in the vector field. See section 5.4 for more details.

Using these layers, it is straightforward to adapt established network architectures in Euclidean domain cases to manifolds case as the only change is to replace conventional convolution by PTC. In addition, back-propagation can be achieved by taking derivation of  $K$ . The compact support of the convolution kernel is represented as a sparse matrix which makes computation efficient.

*Remark 4.1 (Vector Fields).* Thus far we have only considered transportation along the geodesic from some chosen seed point. Alternatively, an affine connection can be defined by an assignment  $\Xi$  as a family of linear transformations on tangent spaces along any smooth curve on  $\mathcal{M}$ . Consider  $\gamma_x^y$  a smooth arc joining two, not necessarily distinct, points from  $x$  to  $y$ , define  $\Xi(\gamma_x^y) : \mathcal{T}_x\mathcal{M} \rightarrow \mathcal{T}_y\mathcal{M}$ . If  $\Xi(\gamma_x^y)$  satisfies the following properties:

- 1)  $\Xi(\gamma_x^y)$  is non-singular,
- 2)  $\lim_{y \rightarrow x} \Xi(\gamma_x^y) = Id$ ,
- 3)  $\Xi(\gamma_x^z) = \Xi(\gamma_y^z)\Xi(\gamma_x^y)$ ,
- 4)  $\Xi$  is Frechét differentiable in terms of  $\gamma$ ,  $x$  and  $y$ .

Then, the vector  $\vec{v}_y$  is obtained by parallel displacement of  $\vec{v}_x$  along  $\gamma_x^y$  is provided as  $\vec{v}_y = \Xi(\gamma_x^y)\vec{v}_x$ . Consider  $V$  is a tangent vector field along  $\gamma$ , the associated infinitesimal connection  $\nabla_{\dot{\gamma}}V = \lim_{h \rightarrow 0} \frac{1}{h}(\Xi(\gamma_{\gamma(h)}^{\gamma(0)})V_{\gamma(h)} - V_{\gamma(0)})$  can be induced from  $\Xi$  [21]. In practice, we can compute the parallel transportation along any other vector field. For some applications, it may be more natural to use another vector field. To do so, we follow the same process except using this new vector field to form the first basis vector in  $V$ . This can be extremely beneficial in dealing with areas in which our geodesic vector field has a singularity. Around the singularity, the direction of the vector field is often highly variable. We can define another vector field that is more regular in this area (but may have singularities elsewhere) to analyze information near the singularity in the first field. The problem of designing and controlling the singularities of vector fields on surfaces is a well-studied problem for which many approaches already exist (see [14] for a review of such techniques). It is important to note that if we would like the results of our training to be generalizable (i.e. when working with multiple domains), then we need the vector fields to be generalizable as well. Another option is to use the average distance vector field. Finally, we could use the vector field given by principal directions of curvature. To do this, we use the geodesic vector field to fix the ambiguity in the first principal directions of curvature so that both vectors lie in the same half-plane as in [3]. Empirically, this works well in cases where the different manifolds are close to each other (see sections 5.2, 5.6), but fails in cases where the manifolds are less similar (see section 5.4). For these reasons using geodesic distances from canonically chosen points is a natural choice. This choice of paths is both highly non-trivial and problem-dependent. In the future, we will further explore options for making this choice by explicitly quantifying the stability of various choices of vector fields under different deformations.

**5. Numerical Experiments.** To illustrate the effectiveness of the proposed PTC, we conduct numerical experiments, including processing images on manifolds using PTC, classifying images on manifolds using PTCNets, demonstrating variational autoencoders on manifolds and learning features on manifolds for registration.

All numerical experiments on MNIST data are implemented in MATLAB on a PC with a 32GB RAM and 3.5GHz CPU, while the final experiment is implemented in Tensorflow with a NVIDIA GTX 1080 Ti graphics card. We remark that these experiments aim to demonstrate the capabilities of the proposed PTC for manipulating functions on curved domains by naturally extending existing wavelet and learning methods from Euclidean domains to curved domains. It is by no means to show that the experiments achieve state-of-the-art results.

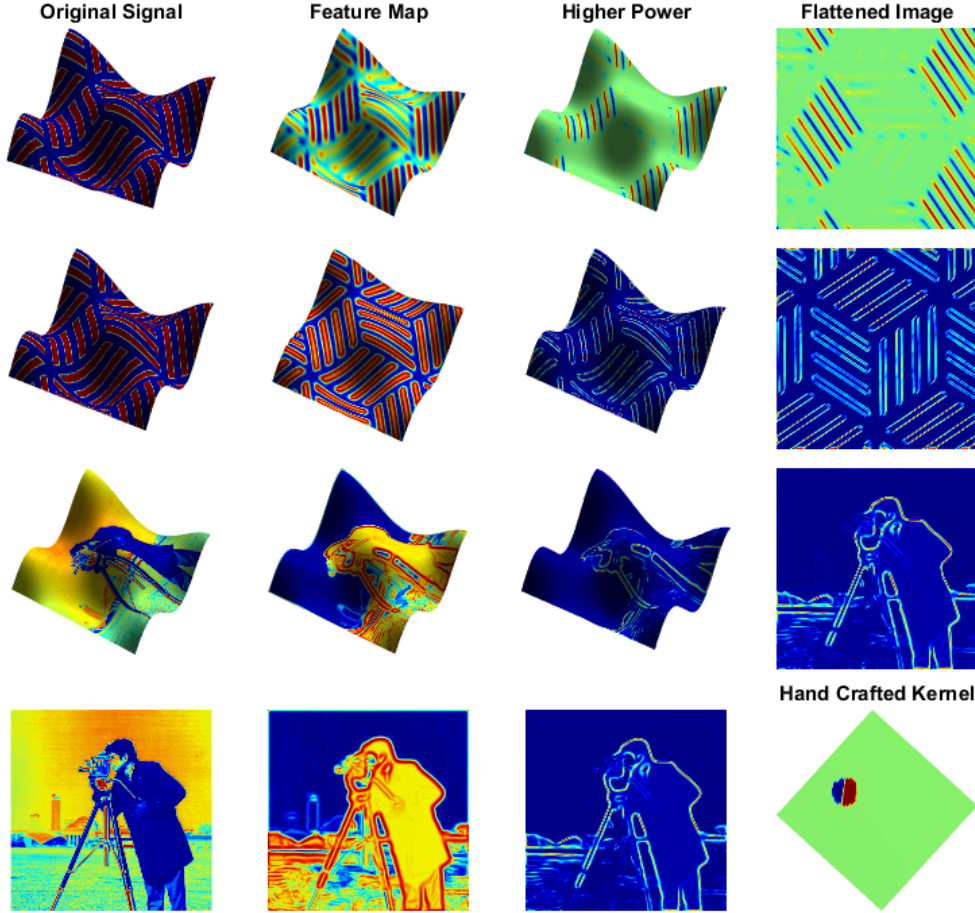


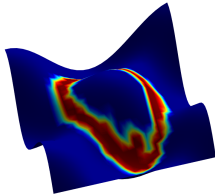
FIG. 2. *First Row: Convolutions without rotation on a test image. Second Row: Convolutions with rotation on test image. Third Row: Convolutions with rotation on a cameraman image. Fourth row: Conventional Euclidean convolution and the edge detector used in PTC.*

**5.1. Wavelet-like Operations.** In the first experiment, we demonstrate the effectiveness of our approach by performing simple signal processing tasks on manifolds using handcrafted filters. Then we compare the PTC results to those produced by traditional techniques applied to Euclidean domains. First, we apply PTC with a handcrafted edge detection filter to images on a manifold. By convolving this filter with the input image, we obtain an output feature function whose higher values indicate similarity to the predefined edge. In the first row of Figure 2, it is clear that the proposed convolution successfully highlights the edges with a similar orientation

to the input filter. In the second row of Figure 2, we allow additional rotations as we discussed in (3.3). We observe that the additional rotation flexibility can reliably capture all of the edges regardless of orientations. This illustrates the directional awareness of our method.

Furthermore, we apply this edge detector using PTC to a more realistic problem in the third row of Figure 2. It shows that the results are very close to those produced in an analogous Euclidean setting (fourth row). In the third column, we show the feature map raised to the fifth power for better contrast, and the last column shows a flattened version for better visualization.

**5.2. Single Manifold MNIST.** In this test, we conduct experiments to demonstrate the effectiveness of PTCNets to handle signals on manifolds. The most highly celebrated early application of CNNs was the recognition of handwritten digits [28]. We map all MNIST data to a curved manifold plotted in the left image of Figure 3. We use a simple network architecture consisting of a single convolution layer with 16 filters followed by a ReLU non-linear layer and then a fully connected layer that outputs a 10-dimensional vector of predictions. We apply this network architecture to four scenarios, including MNIST data on a Euclidean domain using traditional convolution, MNIST data on a Euclidean domain using GCNN, ACNN, spectral convolution and the proposed PTC.



Network	Domain	Accuracy
Traditional	Euclidean	<b>98.85</b>
Flat PTCNet	Euclidean	98.10
GCNN	Manifold	96.22
ACNN	Manifold	97.12
Spectral	Manifold	95.35
<b>PTCNet</b>	Manifold	<b>97.96</b>

FIG. 3. Comparison of our PTCNet to Euclidean case, GCNN, ACNN and a spectral based method on a single manifold.

Each network is implemented in MATLAB using only elementary functions and is trained using batch stochastic gradient descent with batch size 50 and a fixed learning rate  $\alpha = 10^{-3}$ . We also use the same random seed for the batch selection and the same initialization. We choose such a simple training regime in order to make the effects of different convolution operations as clear as possible. We measure the results by the overall network error after 5,000 iterations.

The table in Figure 3 shows the accuracy of the traditional CNN on a flat domain, a spectral net applied to a simple manifold, as well as our network applied to both a Euclidean domain (Flat PTCNet) and the manifold. The similar performance of Flat PTCNet to traditional CNN illustrates that our method is an appropriate generalization of convolution from flat domains to curved domains. To test the effectiveness of the proposed PTC, we compare our results with those obtained from GCNN and ACNN based on the same network structure. We observe that our method outperforms the spectral network, GCNN, and ACNN for this classification task on a single curved domain.

**5.3. Multi-Manifold MNIST.** One of the advantages of our method is that filters that are learned on one manifold can be applied to different domains. The spectral convolution-based methods do not have this transferability as different domains

are unlikely to share the same eigensystem. In this experiment, we first directly apply the network learned by the PTCNet, GCNN, ACNN and Spectral networks from Section 5.2 to a new manifold. As we illustrate in the first row of the table in Fig. 4, the accuracy of the spectral convolution-based method is dramatically reduced since the two manifolds have quite different eigensystems. Meanwhile, our PTCNet can still provide the most accurate results among all methods since the underlying geodesic vector fields of these manifolds are more stable to deformations than eigensystems are.

Training	Success Rate
Spectral	88.50
Single Manifold GCN	95.56
Multiple Manifold GCN	96.97
Single Manifold ACNN	92.52
Multiple Manifold ACNN	91.97
Single Manifold PTC	95.65
Multiple Manifold PTC	<b>97.32</b>

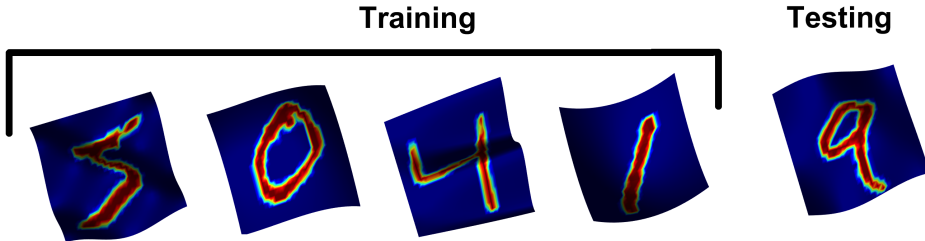


FIG. 4. Top table: Comparison of results from learning on single and multiple domains and then testing on a new manifold. Bottom: Manifolds used for multi-mainfold tests. The first four are used for training and the last is used for testing.

Furthermore, we conduct a new experiment in which we train our PTCNet on a variety of manifolds and test on different manifolds as shown in the bottom picture of Figure 4, where the first four manifolds are used as training domains, and the fifth one is used for testing. Notice that these deformation are non-isometric. From these experiments, it is clear that spectral methods and methods which require curvature to determine their alignment do not perform as well as our method. However, the geodesic vector fields of the manifolds are quite similar, and therefore filters learned through our technique should apply to the new problem. As we can see in the last row of the Table in Figure 4, the network achieves a 97.32% success rate since training on multiple manifolds allows PTC network to learn greater invariance to local deformation in the metric, which enables great transferability.

**5.4. Singularities of vector fields.** In each of the previous experiments, the vector field used to translate the convolutional kernels is chosen to be the gradient of the geodesic from one corner of the manifold. Although our convolution is well defined everywhere on these manifolds, the filters may be more variable near this singularity. To investigate the effects that these singularities may have on, we next test our network using different types of vector fields for the pervious experiments on the MINST data. PTC1 uses the vector field chosen as in the previous experiments. PTC2 uses a vector field with a singularity in the center of the domain. The next test (PTC3) has two separate vector fields, each with a singularity at a different point

on the interior of the domain. For this test, half the kernels are assigned to one vector field and half to the other. The last test uses four vector fields (PTC4), each with a singularity at a different point on the interior of the manifold. Table 2 shows the results of using these vector fields on the single and multiple manifold problems described previously. We observe that the presence of singularity can negatively affect the performance while using multiple vector fields can overcome these difficulties.

Implementation	VF	Sings per VF	Single: Accuracy	Multi: Accuracy
Spectral	-	-	92.10	88.50
PTC1	1	0	<b>97.96</b>	<b>97.32</b>
PTC2	1	1	94.92	94.51
PTC3	2	1	95.89	95.02
PTC4	4	1	96.01	95.28

TABLE 2

Success rate (SR) comparison of several of our networks on a single (the 4<sup>th</sup> column) and on multiple manifolds (the 5<sup>th</sup> column).

**5.5. MNSIT Convolutional Variational Auto-Encoder (CVAE).** Variational Auto-encoders (VAE) [20] are a generic tool used for data compression and generation. Given some input signal  $x$ , one wishes to compute an encoder function  $E : x \mapsto \hat{x}$  which greatly reduces the dimension of  $x$  and a decoder function  $D : \hat{x} \mapsto x$  which recovers  $x$ . Variational auto-encoders also require that the latent variable  $x$  follow some unit normal distribution:  $\hat{x} \sim N(0, I)$ . This requirement allows for the creation of new data by passing random samples from  $N(0, I)$  into the decoder as  $\hat{x}$ . An auto-encoder is also called convolutional, if the feature extraction in the encoder is done through strided convolution, and the upsampling in the decoder is done through transposed convolution.

In this test, we use the MNSIT handwritten digit database to validate our proposed definition by creating an CVAE on a manifold embedding of the MNSIT data set. We denote a PTC convolution layer as  $PTC_a B$  where  $a$  is the number of points in the discretized domain and  $B$  is the number of filters in this level. Then our architecture for the encoder is:

$$x \rightarrow PTC_{784}16 \rightarrow PTC_{196}16 \rightarrow PTC_{49}16 \rightarrow FC(10, 2) = (\mu, \Sigma)$$

Similarly the decoder is defined by:

$$\hat{x} \rightarrow N(\mu, \Sigma) \rightarrow FC(49, 1) \rightarrow PTC_{49}16 \rightarrow PTC_{196}16 \rightarrow PTC_{784}16 \rightarrow \sum_{axis=0} = x_{out}$$

The network is trained by simultaneously minimizing the KL divergence between  $N(\mu, \Sigma)$  and the  $L_2$  loss between  $x$  and  $x_{out}$ . Figure 5 shows several examples on pairs of input signals and their recover as well as several figures generated by randomly sampling latent variables from the unit normal distribution.

One additional advantage of this framework is that, since we only use fully connected layers at the coarsest level of sampling, we only need coarse correspondences to apply a trained model to a new manifold. Since the PTC layers are agnostic to re-indexing of the data points, and geodesics (excluding the cut locus) are stable under small deformations, we can use the filters learned on one domain to apply to another. The fully connected layer still requires a correspondence to be consistent.



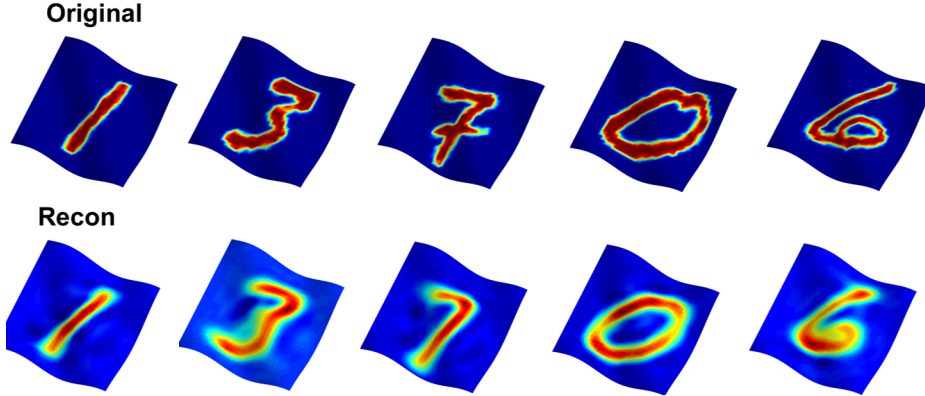


FIG. 5. PTC-VAE. Row 1: Input Image, Row 2: Recovered Image

Model	MSE
VAE-Flat	.0941
PTC-Single	.0956
PTC-Transfer	.0977

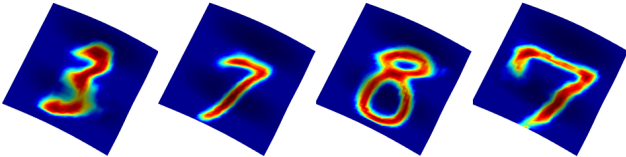


FIG. 6. Left: Reconstruction errors from test set for traditional flat VAE, PTC trained on a single manifold and transfer ed PTC, Right: Images Generated by randomly sampling latent variables as input to trained model and applying PTC to new surface not used during training.

Fortunately, the sparse correspondences required by this approach are much easier to compute than the dense correspondences, which would be required in a method without intrinsic down/upsampling. Figure 6 shows the reconstruction errors on the test set for a standard VAE, a PTC-VAE, and the PTC-transferred onto a new manifold. On the right side we show an example of several additional digits on a new manifold, given by a model trained on the previous surface.

**5.6. Feature Learning for Shape Registration.** One important application of convolution neural networks in shape processing is the creation of geometric features [5]. The goal of these networks is to output descriptor functions,  $F : \mathcal{M} \rightarrow \mathbb{R}$ , which accurately describe the local and global geometry of a manifold. In this section we implement a network based on the 'ShapeNet2' architecture originally presented in [32] for shape registration, substituting in our proposed definition of convolution. We remark that this architecture is not state-of-the-art, but provides a good framework for comparing geometric convolutions. In this network we input a 150-dimensional geometry vector into a vector connected layer which linearly combines these input features into a 16-dimensional signal. This signal is then passed through two layers of PTC (each followed by a Relu non-linearity) with 16 filters in each layer. The final features are the output of the second convolution layer. The network is trained by minimizing the following triplet loss:

$$\begin{aligned}
 L(\Theta) = & \sum_{(S_1, S_2)} \|F(S_1; \Theta) - F(S_2; \Theta)\|^2 \\
 (5.1) \quad & + \lambda \sum_{(S_1, S_3)} (\mu_1 - \|F(S_1; \Theta) - F(S_3; \Theta)\|)^2
 \end{aligned}$$



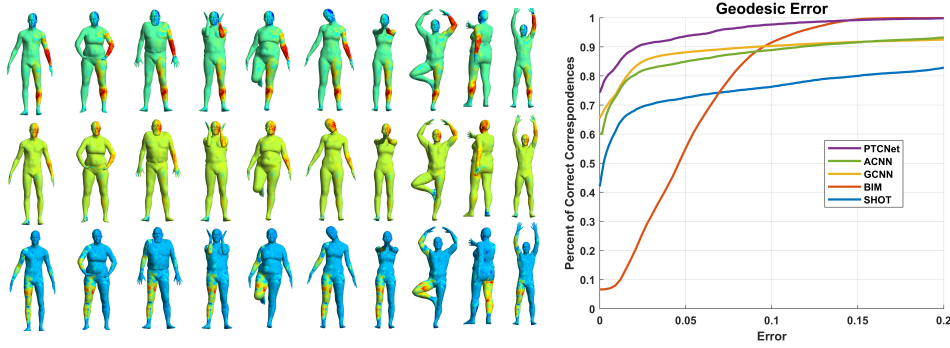


FIG. 7. Left: Example feature functions for shape correspondence on the Faust dataset. Right: geodesic errors in predicted correspondence of our method and several others.

where  $\{S_1, S_2\}$  are similar pairs of shapes,  $\{S_1, S_3\}$  are dissimilar and  $\mu$  is the user parameter representing the margin. We evaluated this model on the Faust dataset which contains 100 real-world scans (each with  $n = 6890$  points) of 10 individuals in 10 poses [2]. We use the first 80 figures for training, 10 for validation, and 10 for testing. Using the sparse matrix operations described in the appendix, each forward and backward propagation through a two-layer network, defined on a mesh containing 6890 points, can be calculated in less than half a second. The whole training process is completed in 8 hours using the ADAM algorithm [19]. Figure 7 shows three of the output feature functions across different individuals in the dataset, where the first 7 individuals (10 surfaces for each individual) are used for training, the 8<sup>th</sup> and 9<sup>th</sup> individuals are used for validation, and the last individual is used for testing. These consistent features lead to satisfactory registration results by simply conducting the nearest point search in the feature space. Figure 7 shows our registration performance, measured by the geodesic error between the predicted correspondence and the actual correspondence, compared to error from using the heat kernel signatures which were used as the input layer. We compare results with the GCNN and ACNN implementation of the ShapeNet2 [32, 3].

We note that the comparison here more focuses on illustrating the effectiveness of a new methodology but claiming to achieve state-of-the-art results since many SOTA methods rely on network architecture designs. For instance, some other advanced architectures for shape correspondence [30, 25] lead to better results as they involve solving some version (often relaxed) of the quadratic assignment problem based on some starting map. We intentionally try to use relatively simple network structures and demonstrate that our method of defining geometric convolutions outperforms other methods on relatively simple architectures. We thought this can more clearly indicate the effectiveness of our methods.

**6. Conclusion.** In this paper, we propose a generalization of the convolution operation on smooth manifolds using parallel transportation and discuss its numerical implementation. Using the proposed PTC, we have performed wavelet-like operations and built convolutional neural networks on curved domains. Our numerical experiments have shown that the PTC can perform as well as Euclidean methods on curved manifolds and is capable of including directional awareness, handling problems involving deformable manifolds, in particular, learning features for deformable manifolds registration. In our future works, we will apply our PTC to different applications

of comparing, classifying, and understanding manifold-structured data by combining it with recent advances of deep learning architectures.

**Appendix A: Computation of  $\mathbf{K}$  matrix.** The matrix  $\mathbf{K}$  contains the filter weights for the convolutional operation interpolated from the stencil onto the mesh/point cloud. Consider the case in which we have a single channel of  $n$  points and a single filter with radius  $d$ . Then  $\mathbf{K}$  will be a  $n$ -by- $n$  matrix where the  $i^{th}$  row represents the transportation of the filter to the  $i^{th}$  point. If there are  $m$  points on the discretized manifold within the  $d$ -radius around the  $i^{th}$  point, the row will have exactly  $m$  non-zero entries. These entries will be given by the linear interpolation of the filter weights to the manifold points computed in the tangent plane. Psuedocode for this operation is presented in Algorithm 7.1. Here we use subscripts to indicate local indexes in the frame denoted by the superscript :

---

**Algorithm 6.1** Compute  $\mathbf{K}$  matrix

---

- 1: Define stencil with  $m$  points at local coordinates  $\{c_j\}$  and maximum radius  $\delta$
  - 2: Compute vector field  $V_i$
  - 3: Initialize  $K$  as empty  $n \times n$  matrix
  - 4: **for** each point  $x_i$  on the manifold **do**
  - 5:   Find all points  $x_j$  withing a  $\delta$  radius of  $x_i$
  - 6:   Computing the local coordinates  $y_j^i$  of the points  $x_j$  in the frame  $V_i$
  - 7:   Compute the linear interpolation weights  $w_j^i$  from  $c_i$  to  $y_i$
  - 8:   Set  $\mathbf{K}_{ij} = w_j^i$
  - 9: **end for**
  - 10: **return**  $\mathbf{K}$
- 

**Appendix B: Efficient computation of PTC layers.** Since the limitation of spare matrix product implementation in TensorFlow and PyTorch, we use the following method to implement the proposed convolution. More specifically, we consider a mesh with  $n$  points, a signal with  $q$  channels  $F = (F_1, \dots, F_q) \in \mathbb{R}^{n \times q}$  and  $p$  filters each of which has  $q$  input channels denoted  $\mathbf{K} = \{K_{11}, \dots, K_{1p}, \dots, K_{q1}, \dots, K_{qp}\}$ . We would like to compute convolution  $F \star \mathbf{K} = \sum_{i=1}^q F_i \star K_{ij} \in \mathbb{R}^{n \times p}$ . Given a mesh with the mass matrix  $M$ , we write  $I_i$  as the index set of the neighborhood of the  $i$  point and denote  $W_i \in \mathbb{R}^{|I_i| \times k}$  the parallel transportation operation to the  $i$ -th point. The following method provides a fast, memory-efficient implementation of PTC convolution in TensorFlow and PyTorch.

We write  $Z_i = F_i^T M \in \mathbb{R}^{n \times 1}$ ,  $i = 1, \dots, q$  and let  $L = \sum_i |I_i|$ . We define  $\mathbf{Z}_i$  as a  $L \times L$  sparse matrix whose support at the  $k$ -th row is provided by  $I_k$  with value  $Z_i(I_k)$ , formally we write:

$$\mathbf{Z}_i = \begin{pmatrix} Z_i(I_1) \\ Z_i(I_2) \\ \vdots \\ Z_i(I_n) \end{pmatrix}, \quad \mathbf{Z} = \begin{pmatrix} \mathbf{Z}_1 & & & \\ & \mathbf{Z}_2 & & 0 \\ & & \ddots & \\ 0 & & & \ddots \\ & & & & \mathbf{Z}_q \end{pmatrix}$$

In addition, we define

$$\mathbf{W} = \begin{pmatrix} W_1 \\ W_2 \\ \vdots \\ \vdots \\ W_n \end{pmatrix}, \bar{\mathbf{W}} = \begin{pmatrix} \mathbf{W}K_{11} & \cdots & \mathbf{W}K_{1p} \\ \mathbf{W}K_{21} & \cdots & \mathbf{W}K_{2p} \\ \vdots & \ddots & \vdots \\ \mathbf{W}K_{q1} & \cdots & \mathbf{W}K_{qp} \end{pmatrix}$$

where  $\bar{\mathbf{W}} = \text{reshape}(\mathbf{W}\mathbf{K}, [Lq, p])$ . Finally, the PTC can be computed as

$$(F \star \mathbf{K}) = \left( \sum_{axis=3} \text{reshape}(\mathbf{Z}\bar{\mathbf{W}}, [p, n, q]) \right)^T$$

Using the above sparse matrix operations, the computation complexity of the proposed PTC is the same scale as the standard convolution in Euclidean domains.

#### REFERENCES

- [1] Y. BENGIO, I. J. GOODFELLOW, AND A. COURVILLE, *Deep learning*, Nature, 521 (2015), pp. 436–444.
- [2] F. BOGO, J. ROMERO, M. LOPER, AND M. J. BLACK, *Faust: Dataset and evaluation for 3d mesh registration*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3794–3801.
- [3] D. BOSCAINI, J. MASCI, E. RODOLÀ, AND M. BRONSTEIN, *Learning shape correspondence with anisotropic convolutional neural networks*, in Advances in Neural Information Processing Systems, 2016, pp. 3189–3197.
- [4] A. BRACHA, O. HALIM, AND R. KIMMEL, *Shape correspondence by aligning scale-invariant lbo eigenfunctions*, in Eurographics Workshop on 3D Object Retrieval, The Eurographics Association, 2020.
- [5] M. M. BRONSTEIN, J. BRUNA, Y. LECUN, A. SZLAM, AND P. VANDERGHEYNST, *Geometric deep learning: going beyond euclidean data*, IEEE Signal Processing Magazine, 34 (2017), pp. 18–42.
- [6] J. BRUNA, W. ZAREMBA, A. SZLAM, AND Y. LECUN, *Spectral networks and locally connected networks on graphs*, arXiv preprint arXiv:1312.6203, (2013).
- [7] R. CHAKRABORTY, M. BANERJEE, AND B. C. VEMURI, *H-cnns: Convolutional neural networks for riemannian homogeneous spaces*, arXiv preprint arXiv:1805.05487, (2018).
- [8] I. CHAVEL, *Riemannian geometry: a modern introduction*, vol. 98, Cambridge university press, 2006.
- [9] D. CIRESAN, A. GIUSTI, L. M. GAMBARDILLA, AND J. SCHMIDHUBER, *Deep neural networks segment neuronal membranes in electron microscopy images*, in Advances in neural information processing systems, 2012, pp. 2843–2851.
- [10] T. S. COHEN, M. GEIGER, J. KÖHLER, AND M. WELLING, *Spherical cnns*, arXiv preprint arXiv:1801.10130, (2018).
- [11] I. DAUBECHIES, *Ten lectures on wavelets*, vol. 61, SIAM, 1992.
- [12] B. DONG, *Sparse representation on graphs by tight wavelet frames and applications*, Applied and Computational Harmonic Analysis, 42 (3) (2017), pp. 452–479.
- [13] X. GLOROT AND Y. BENGIO, *Understanding the difficulty of training deep feedforward neural networks*, in Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.
- [14] F. D. GOES, M. DESBRUN, AND Y. TONG, *Vector fields*, in Course Notes AMC SIGGRAPH Asia, 2015.
- [15] O. HALIMI, O. LITANY, E. RODOLA, A. M. BRONSTEIN, AND R. KIMMEL, *Unsupervised learning of dense shape correspondence*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4370–4379.
- [16] D. K. HAMMOND, P. VANDERGHEYNST, AND R. GRIBONVAL, *Wavelets on graphs via spectral graph theory*, Applied and Computational Harmonic Analysis, 30 (2011), pp. 129–150.
- [17] M. HENAFF, J. BRUNA, AND Y. LECUN, *Deep convolutional networks on graph-structured data*, arXiv preprint arXiv:1506.05163, (2015).

- [18] G. HINTON, L. DENG, D. YU, G. E. DAHL, A.-R. MOHAMED, N. JAITLEY, A. SENIOR, V. VAN-  
HOUCHE, P. NGUYEN, AND T. N. SAINATH, *Deep neural networks for acoustic modeling  
in speech recognition: The shared views of four research groups*, IEEE Signal Processing  
Magazine, 29 (2012), pp. 82–97.
- [19] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint  
arXiv:1412.6980, (2014).
- [20] D. P. KINGMA AND M. WELLING, *Auto-encoding variational bayes*, arXiv preprint  
arXiv:1312.6114, (2013).
- [21] M. KNEBELMAN, *Spaces of relative parallelism*, Annals of Mathematics, (1951), pp. 387–399.
- [22] S. KOBAYASHI AND K. NOMIZU, *Foundations of differential geometry*, vol. 2, Interscience pub-  
lishers New York, 1969.
- [23] R. KONDOR AND S. TRIVEDI, *On the generalization of equivariance and convolution in neural  
networks to the action of compact groups*, arXiv preprint arXiv:1802.03690, (2018).
- [24] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep con-  
volutional neural networks*, in Advances in neural information processing systems, 2012,  
pp. 1097–1105.
- [25] Z. LÄHNER, M. VESTNER, A. BOYARSKI, O. LITANY, R. SLOSSBERG, T. REMEZ, E. RODOLÀ,  
A. BRONSTEIN, M. BRONSTEIN, R. KIMMEL, ET AL., *Efficient deformable shape correspon-  
dence via kernel matching*, arXiv preprint arXiv:1707.08991, (2017).
- [26] R. LAI, J. LIANG, AND H. ZHAO, *A local mesh method for solving pdes on point clouds*, Inverse  
Prob. and Imaging, 7 (2013), pp. 737–755.
- [27] Y. LECUN, B. E. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. E. HUBBARD, AND  
L. D. JACKEL, *Handwritten digit recognition with a back-propagation network*, in Advances  
in neural information processing systems, 1990, pp. 396–404.
- [28] Y. LECUN, L. BOTTOU, Y. BENGIO, AND P. HAFFNER, *Gradient-based learning applied to  
document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.
- [29] M. K. LEUNG, H. Y. XIONG, L. J. LEE, AND B. J. FREY, *Deep learning of the tissue-regulated  
splicing code*, Bioinformatics, 30 (2014), pp. i121–i129.
- [30] O. LITANY, T. REMEZ, E. RODOLÀ, A. BRONSTEIN, AND M. BRONSTEIN, *Deep functional maps:  
Structured prediction for dense shape correspondence*, in Proceedings of the IEEE Inter-  
national Conference on Computer Vision, 2017, pp. 5659–5667.
- [31] S. MALLAT, *A wavelet tour of signal processing: the sparse way*, Academic press, 2008.
- [32] J. MASCI, D. BOSCAINI, M. BRONSTEIN, AND P. VANDERGHEYNST, *Geodesic convolutional neural  
networks on riemannian manifolds*, in Proceedings of the IEEE international conference  
on computer vision workshops, 2015, pp. 37–45.
- [33] C. MOENNING AND N. A. DODGSON, *Fast marching farthest point sampling*, tech. report, Uni-  
versity of Cambridge, Computer Laboratory, 2003.
- [34] F. MONTI, D. BOSCAINI, J. MASCI, E. RODOLÀ, J. SVOBODA, AND M. M. BRONSTEIN, *Geo-  
metric deep learning on graphs and manifolds using mixture model cnns*, arXiv preprint  
arXiv:1611.08402, (2016).
- [35] A. POULENARD AND M. OVSJANIKOV, *Multi-directional geodesic neural networks via equivariant  
convolution*, ACM Transactions on Graphics (TOG), 37 (2018), pp. 1–14.
- [36] E. RODOLÀ, S. ROTA BULO, T. WINDHEUSER, M. VESTNER, AND D. CREMERS, *Dense non-rigid  
shape correspondence using random forests*, in Proceedings of the IEEE Conference on  
Computer Vision and Pattern Recognition, 2014, pp. 4177–4184.
- [37] R. RUSTAMOV AND L. J. GUIBAS, *Wavelets on graphs via deep learning*, in Advances in neural  
information processing systems, 2013, pp. 998–1006.
- [38] P. SERMANET, D. EIGEN, X. ZHANG, M. MATHIEU, R. FERGUS, AND Y. LECUN, *Overfeat: Inte-  
grated recognition, localization and detection using convolutional networks*, arXiv preprint  
arXiv:1312.6229, (2013).
- [39] J. SHOTTON, T. SHARP, A. KIPMAN, A. FITZGIBBON, M. FINOCCHIO, A. BLAKE, M. COOK,  
AND R. MOORE, *Real-time human pose recognition in parts from single depth images*,  
Communications of the ACM, 56 (2013), pp. 116–124.
- [40] D. I. SHUMAN, B. RICAUD, AND P. VANDERGHEYNST, *Vertex-frequency analysis on graphs*,  
Applied and Computational Harmonic Analysis, 40 (2016), pp. 260–291.
- [41] I. SUTSKEVER, O. VINYALS, AND Q. V. LE, *Sequence to sequence learning with neural networks*,  
in Advances in neural information processing systems, 2014, pp. 3104–3112.
- [42] Y. WANG, B. LIU, AND Y. TONG, *Linear surface reconstruction from discrete fundamental  
forms on triangle meshes*, in Computer Graphics Forum, vol. 31, Wiley Online Library,  
2012, pp. 2277–2287.
- [43] Y. YANG, S. LIU, H. PAN, Y. LIU, AND X. TONG, *Pfenn: Convolutional neural networks on 3d  
surfaces using parallel frames*, in Proceedings of the IEEE/CVF Conference on Computer

Vision and Pattern Recognition (CVPR), June 2020.