Geometry Processing and Geometric Deep Learning

MVA Course, Lecture 3, part 2

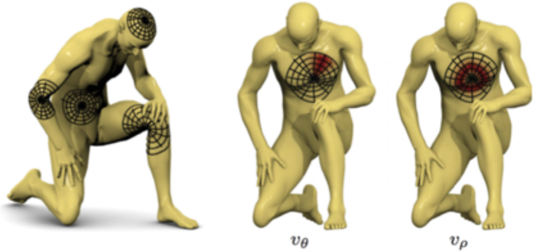
16/ 10 / 2024

Maks Ovsjanikov

Approaches for 3D Deep-Learning

View-based

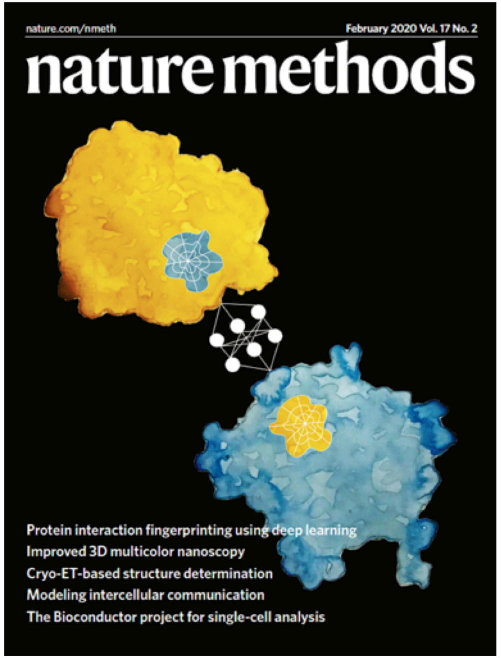
Volumetric

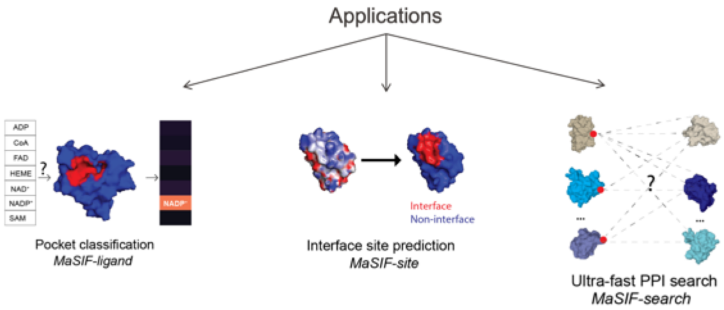


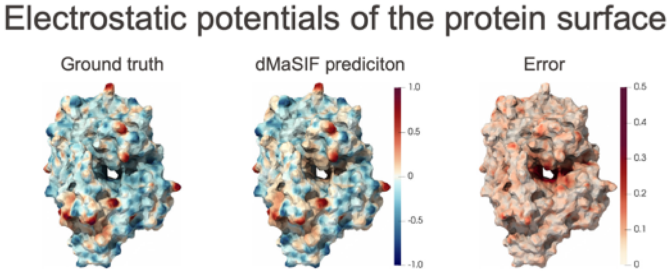
Intrinsic (surface-based)

Point-based

Tasks for intrinsic deep learning



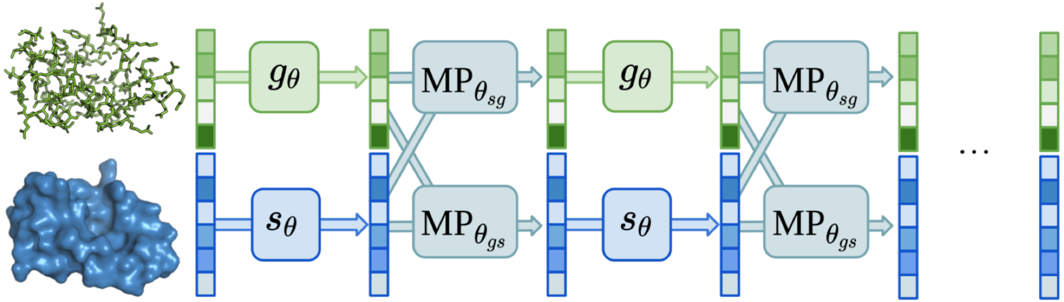


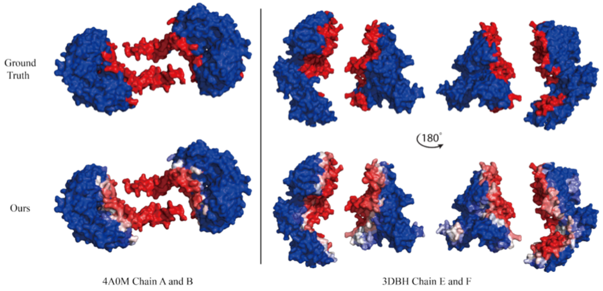


"Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning", P. Gainza et al. Nature Methods 2020

Some Applications

Simultaneously using graphs and surfaces to learn protein-related tasks.

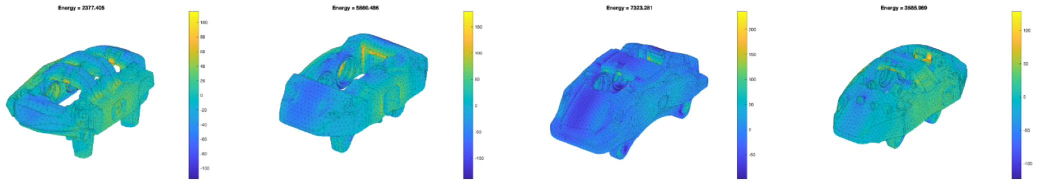


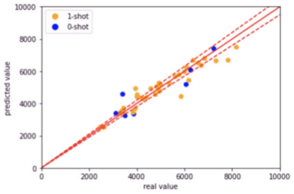


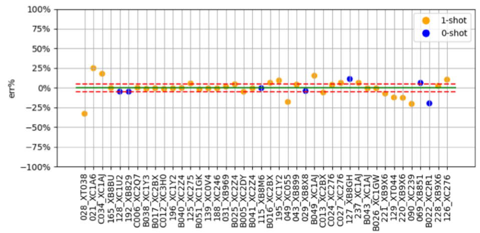
V. Mallet, S. Attaiki, Y. Miao, B. Correia and M Ovsjanikov. "AtomSurf: Surface Representation for Learning on Protein Structures." arXiv preprint arXiv:2309.16519 (2023).

Some Applications

Surface-based learning for predicting physical properties of complex shapes.







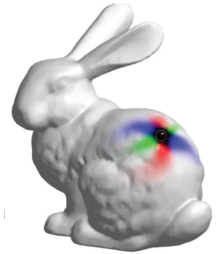
Data-driven energy prediction technique with DiffusionNet

Today: Deep Learning on 3D shapes

- Recap of CNNs and their properties
- Multi-view, extrinsic, projection-based approaches
- Spectral methods, pros and cons
- Intrinsic approaches
- Learning via diffusion

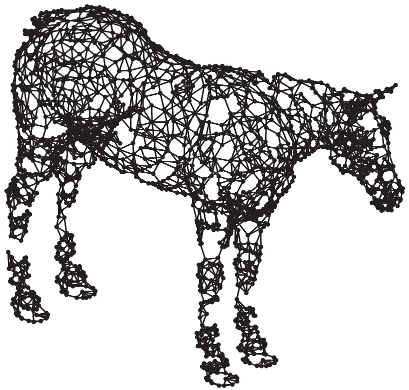
Main question (for the rest of the lecture):

How to enable neural networks to operate *directly on 3D surfaces*?



Spectral domain Deep Learning methods

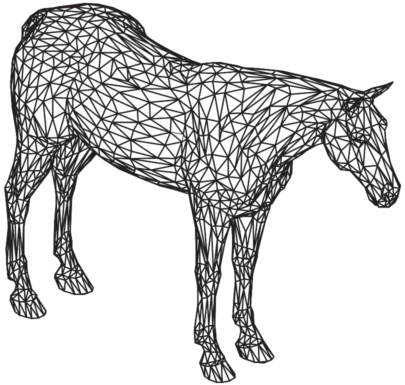
Discrete surfaces



Nearest neighbor graph

Vertices $\mathcal{V} = \{1, \dots, n\}$

Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$



Triangular mesh

Vertices $\mathcal{V} = \{1, \dots, n\}$

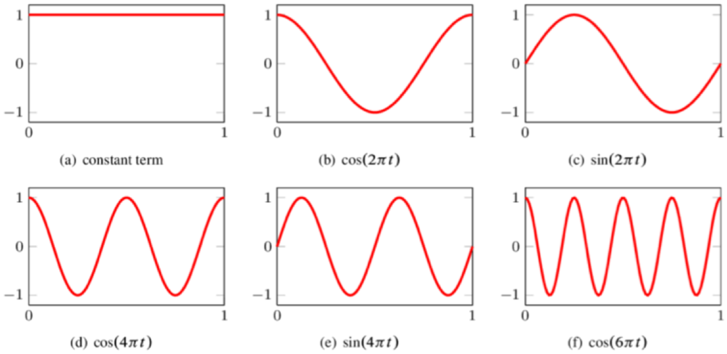
Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

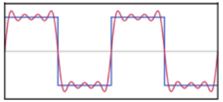
Faces $\mathcal{F} = \{(i, j, k) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} : (i, j), (j, k), (k, i) \in \mathcal{E}\}$

Manifold mesh = each edge is shared by 2 faces + each vertex has 1 loop

Fourier Bases

Fourier basis on a periodic domain [0, 1]:



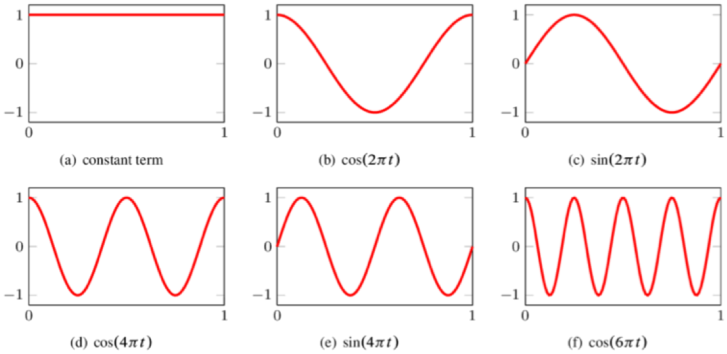


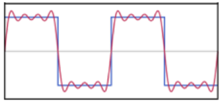
Classical result: any (sufficiently "nice") function can be written as a linear combination of Fourier basis functions:

$$f(t) = a_0 + \sum_{k=1}^{\infty} \left(a_k \sin(2\pi kt) + b_k \cos(2\pi kt) \right)$$

Fourier Bases

Fourier basis on a periodic domain [0, 1]:





Classical result: any (sufficiently "nice") function can be written as a linear combination of Fourier basis functions:

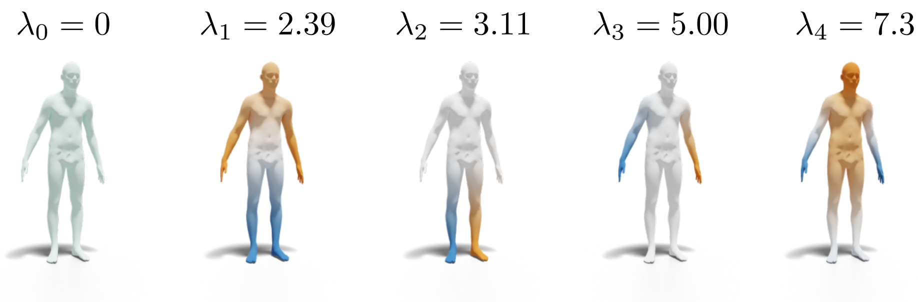
$$f(t) = \sum_k a_k e^{ikt}$$

Laplace-beltrami eigenfunctions

Eigenfunctions of the *Laplacian* matrix.

$$L\phi_i = \lambda_i \phi_i$$

- Generalization of Fourier to graphs and surfaces.
- Ordered by eigenvalues and provide a natural notion of *frequency* (scale).



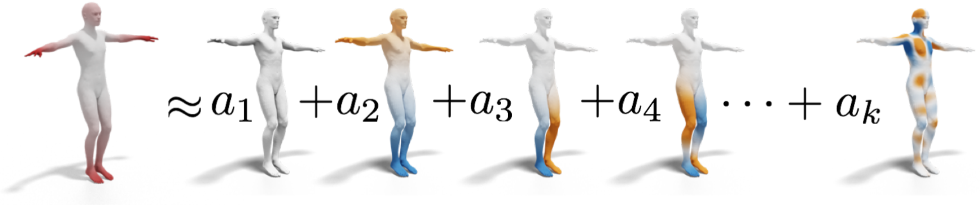
Bruno Levy "Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry." *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06).*

Signal Processing on Surfaces

Eigenfunctions of the *Laplacian* matrix.

$$L\phi_i = \lambda_i \phi_i$$

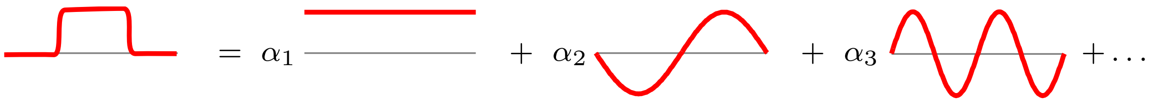
- Generalization of Fourier to graphs and surfaces.
- Ordered by eigenvalues and provide a natural notion of *frequency* (scale).
- Can be used to express functions on a surface:



Fourier analysis: Euclidean Space

A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as Fourier series

$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx'}_{\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}} e^{ikx}$$

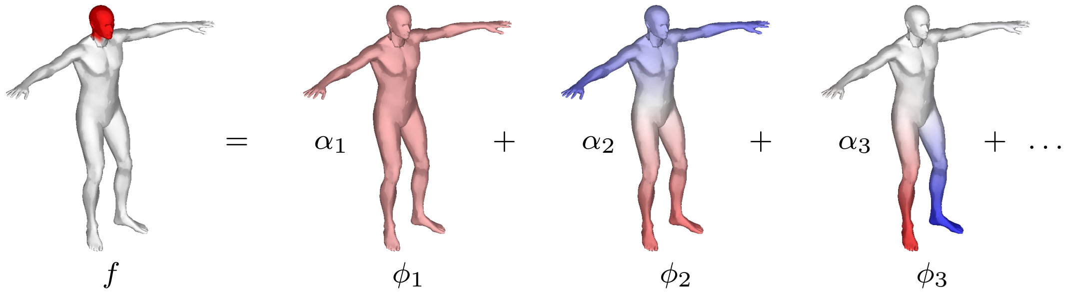


Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

Fourier analysis: Non-Euclidean Space

A function $f : \mathcal{X} \to \mathbb{R}$ can be written as Fourier series

$$f(x) = \sum_{k \geq 1} \underbrace{\int_{\mathcal{X}} f(x')\phi_k(x')dx'}_{\hat{f}_k = \langle f, \phi_k \rangle_{L^2(\mathcal{X})}} \phi_k(x)$$



Fourier basis = Laplacian eigenfunctions: $\Delta\phi_k(x) = \lambda_k \phi_k(x)$

Convolution: Euclidean Space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

- Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
- Convolution operator commutes with Laplacian: $(\Delta f) \star g \;= \Delta(f \star g)$
- Convolution theorem: Fourier transform diagonalizes the convolution operator $\Rightarrow$ convolution can be computed in the Fourier domain as

$$\widehat{(f \star g)} = \hat{f} \cdot \hat{g}$$

- Efficient computation using FFT

d'Alembert 1754; Borel 1899

Generalized Convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g = \underbrace{\sum_{k \geq 1} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})}}_{\text{product in the Fourier domain}} \phi_k}_{\text{inverse Fourier transform}}$$

Generalized Convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g = \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \phi_k$$

In matrix-vector notation

$$f \star g = \Phi \left(\Phi^+ g \circ \Phi^+ f \right)$$

$$\Phi^+ = \Phi^T \text{ if } \Phi^T \Phi = Id$$

$$\Phi^+ = \Phi^T A \text{ if } \Phi^T A \Phi = Id$$

- Elementwise multiplication. I.e., $(a \circ b)_i = a_i b_i$

Generalized Convolution

Generalized convolution of $f, g \in L^2(\mathcal{X})$ can be defined by analogy

$$f \star g \quad = \quad \sum_{k \geq 1} \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})} \, \phi_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \boldsymbol{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \boldsymbol{\Phi}^\top \, \mathbf{f}$$

Spectral Convolution

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi \left(\sum_{l'=1}^{p} \mathbf{\Phi} \mathbf{W}_{l,l'} \mathbf{\Phi}^\top \mathbf{f}_{l'} \right) \qquad \begin{array}{l} l = 1, \ldots, q \\ l' = 1, \ldots, p \end{array}$$

where $\mathbf{W}_{l,l'} = n \times n$ diagonal matrix of filter coefficients

Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." ICLR 2013

Spectral Convolution

Convolutional layer expressed in the spectral domain

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\mathbf{W}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \qquad \begin{array}{l} l = 1,\dots,q \\ l' = 1,\dots,p \end{array}$$

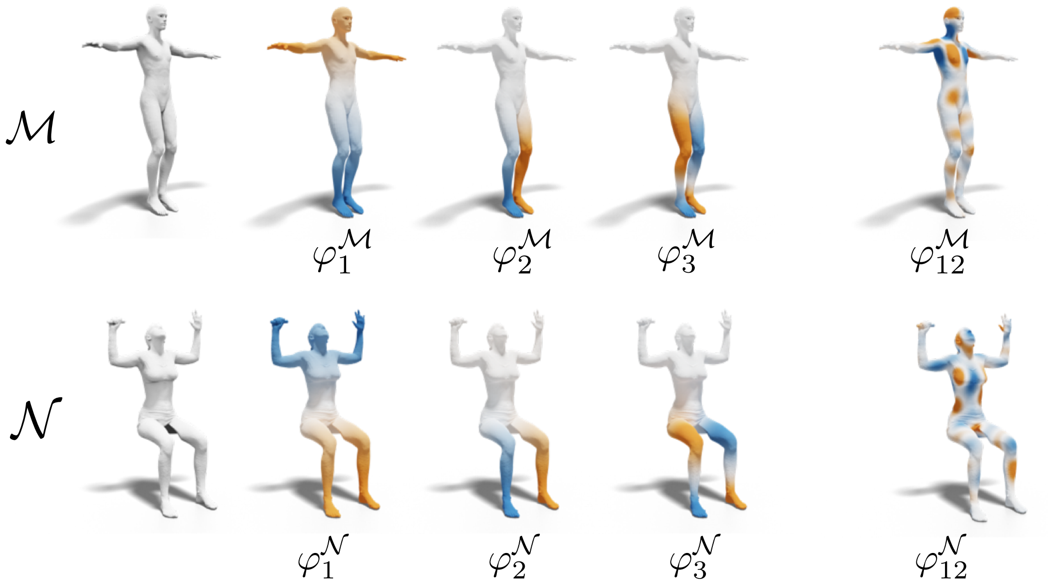
where $\mathbf{W}_{l,l'} = n \times n$ diagonal matrix of filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs!
- ☹ $\mathcal{O}(n)$ parameters per layer
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on graphs)
- ☹ No guarantee of spatial localization of filters

Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." ICLR 2013

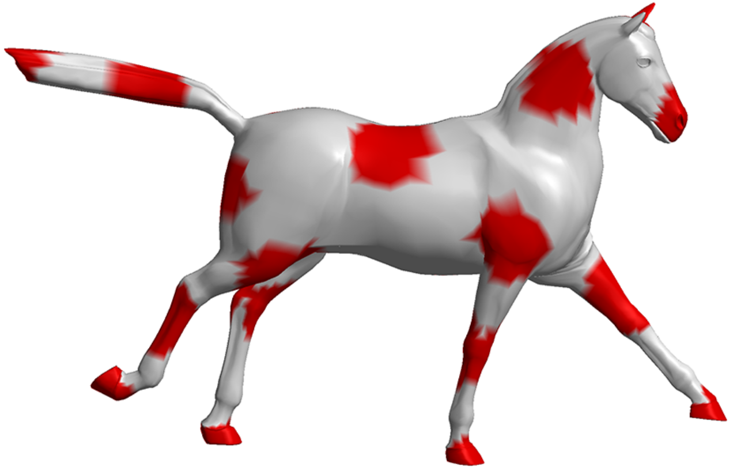
Domain-dependence of the basis

The Fourier (Laplacian) basis on a surface *depends on the surface*



The filters applied on one shape *do not* translate to a new shape

Domain-dependence of the basis



Function $\mathbf{f}$

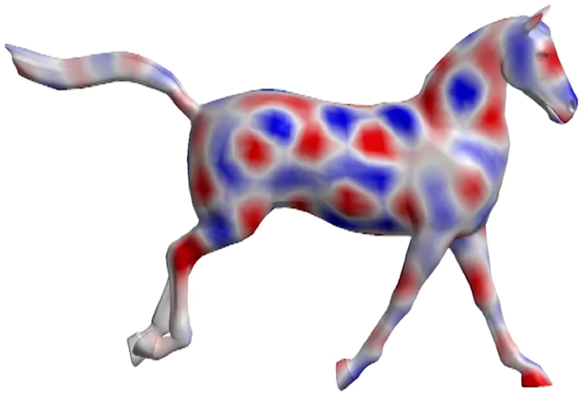
Image credit: E. Rodolà

Domain-dependence of the basis

'Edge detecting' spectral filter $\mathbf{\Phi W \Phi^\top f}$

Image credit: E. Rodolà

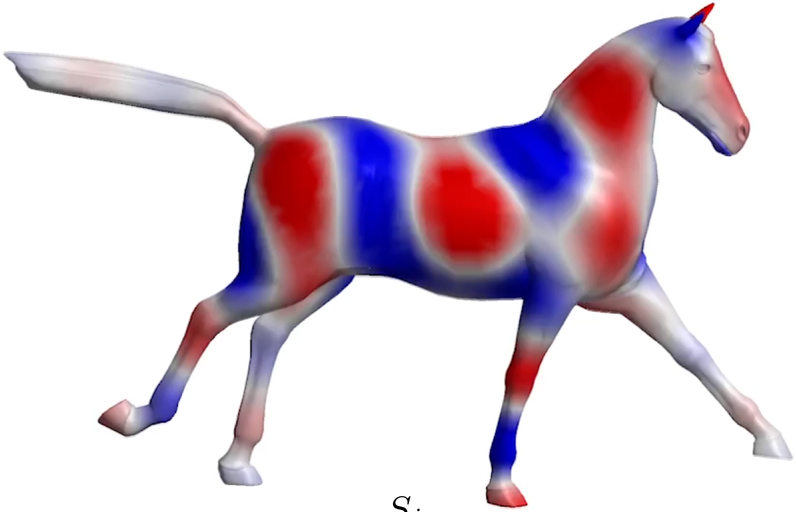
Domain-dependence of the filter



Applying the same filter on different shapes

$$\mathbf{\Phi W \Phi^\top f}$$

Domain-dependence of the basis



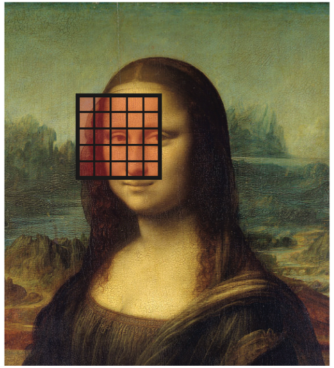
$\varphi_{10}^{S_i}$

Laplacian eigenfunction on different shapes

Spatial domain Deep Learning methods

Non-Euclidean learning

Idea: apply kernels directly on the surface!



Euclidean

Non-Euclidean

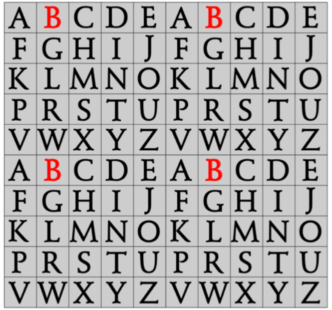
image credit M.Bronstein

Geometric Deep Learning: Going Beyond Euclidean Data Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P.. IEEE Signal Processing Magazine. 2017

A Comprehensive Survey on Geometric Deep Learning. Cao W, Yan Z, He Z, He Z. 2020

Non-Euclidean learning

Standard approach: texture mapping on 3D shapes.



Euclidean

Non-Euclidean

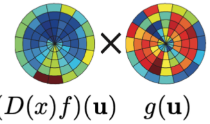
Challenges:

- No canonical *global system* of coordinates
  No grid structure for convolution
- No shift-invariance

Non-Euclidean learning

Key idea: parameterize the shape *locally*. Define a **patch operator:**

$$(f \star g)(x) = \int \underbrace{(D(x)f)(\mathbf{u})}_{} \times \underbrace{g(\mathbf{u})}_{} \, d\mathbf{u}$$



The kernel g(u) is defined in the Euclidean plane. It is *applied* on the surface.

J. Masci et al. "Geodesic convolutional neural networks on riemannian manifolds." CVPRW, 2015

Geodesic convolutional neural networks

Key idea: parameterize the shape *locally.*

Using local polar coordinates on the surface can multiply the signal f with a trainable kernel g:

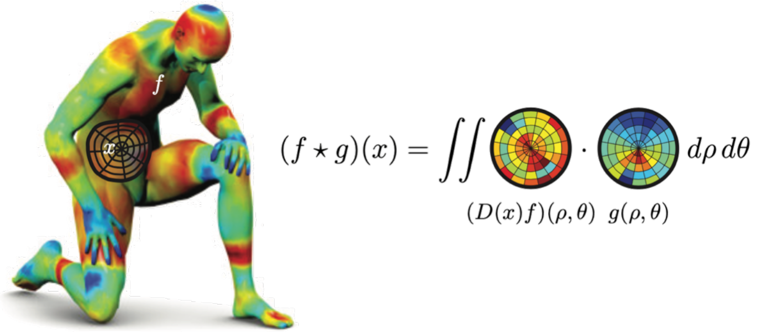


image credit M.Bronstein

Product is a scalar per point ⇒ a real-valued function on the surface. Can stack in layers.

- Geodesic convolutional neural networks on Riemannian manifolds, Masci et al., 2015
- Geometric deep learning on graphs and manifolds using mixture model CNNs, Monti et al., 2017
- CNNs on Surfaces using Rotation-Equivariant Features, Wiersma et al. 2020. …

Geodesic convolutional neural networks

Key idea: parameterize the shape *locally.*

Using local polar coordinates on the surface can multiply the signal f with a trainable kernel g:

$$(f \star g)(x) = \iint (D(x)f)(\rho,\theta) \cdot g(\rho,\theta) \, d\rho \, d\theta$$

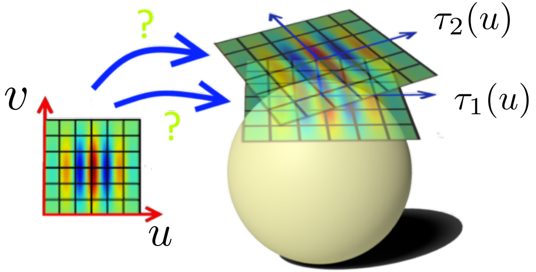
image credit M.Bronstein

Product is a scalar per point ⇒ a real-valued function on the surface.
Can stack in layers. **Intrinsic = pose invariant!**

- Geodesic convolutional neural networks on Riemannian manifolds, Masci et al., 2015
- Geometric deep learning on graphs and manifolds using mixture model CNNs, Monti et al., 2017
- CNNs on Surfaces using Rotation-Equivariant Features, Wiersma et al. 2020. …

Intrinsic Learning on Surfaces

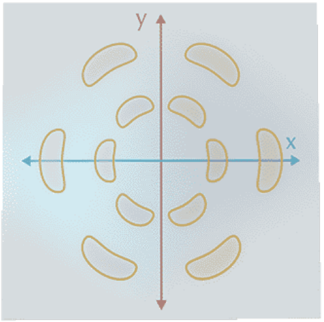
Challenge: parameterization only defined up to rotation, even locally:

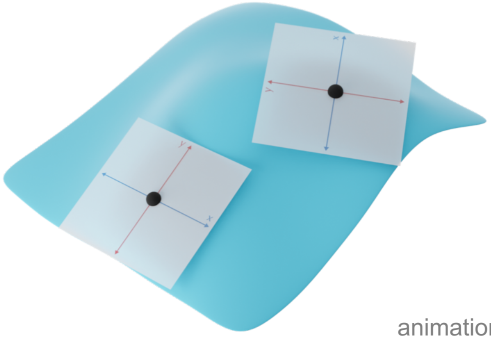


Masci, Boscaini, Bronstein, Vandergheynst 2015

Intrinsic Learning on Surfaces

Challenge: parameterization only defined up to rotation, even locally:



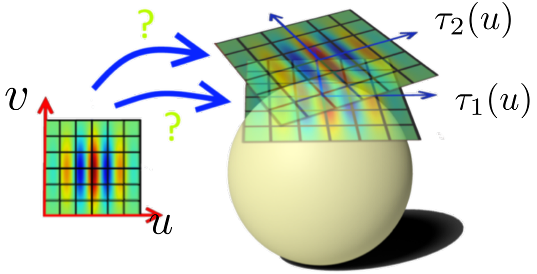


animation by R Wiersma et al.

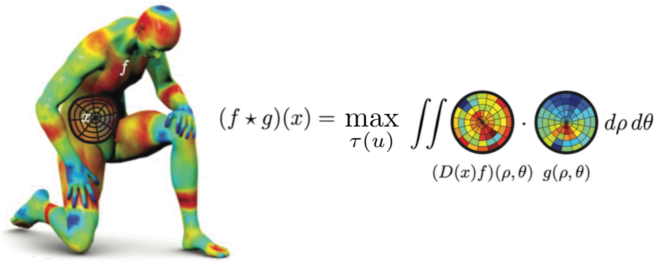
Masci, Boscaini, Bronstein, Vandergheynst 2015

Non-Euclidean learning

Challenge: parameterization only defined up to rotation, even locally:



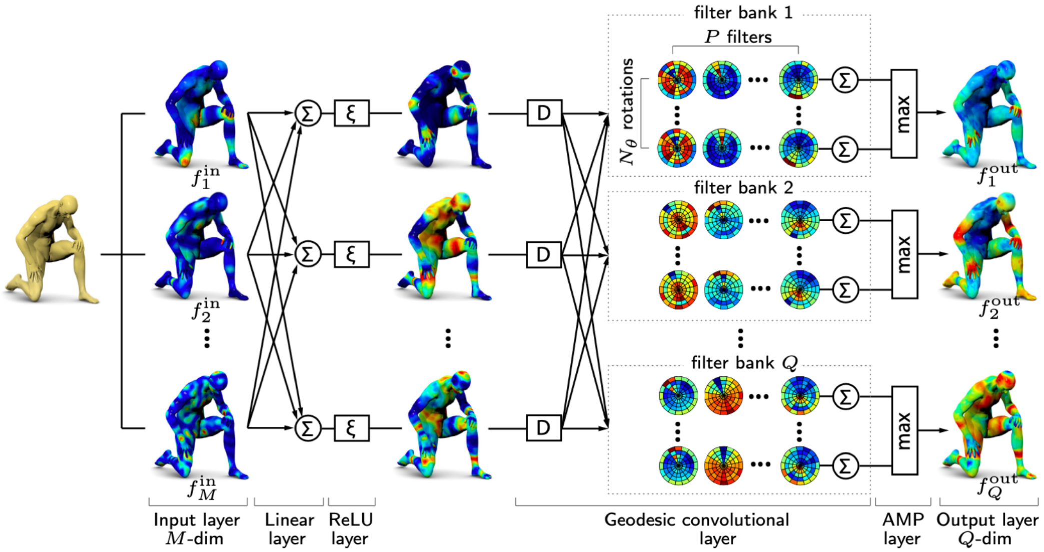
Early solution: take the maximal response *across all rotations.*



J. Masci et al. "Geodesic convolutional neural networks on riemannian manifolds." CVPRW, 2015

Intrinsic Learning on Surfaces

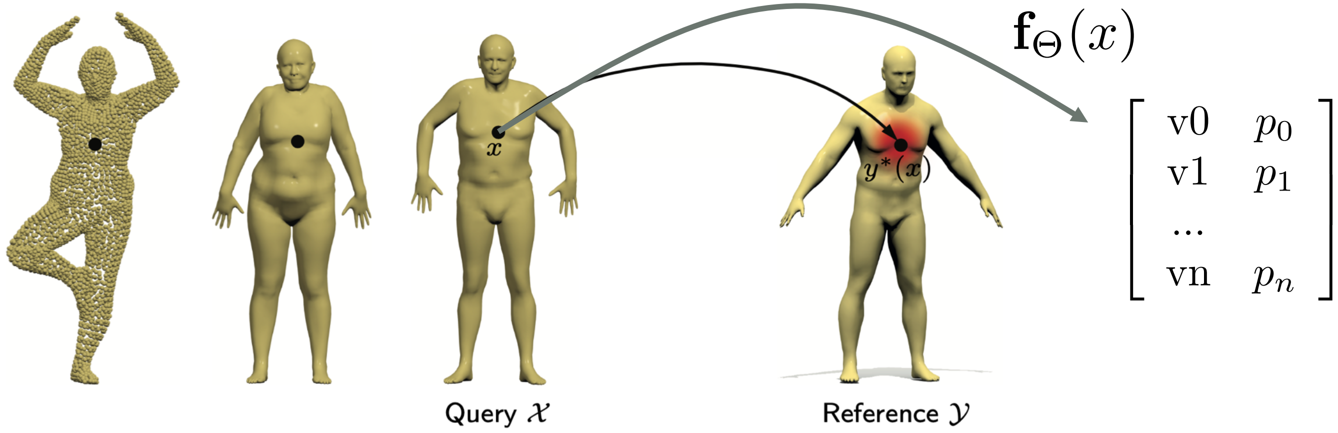
Basic GCNN (geodesic CNN) architecture:



slide credit E. Rodolà

Geodesic convolutional neural networks on Riemannian manifolds, Masci, Boscaini, Bronstein, Vandergheynst 2015

Learning Correspondences



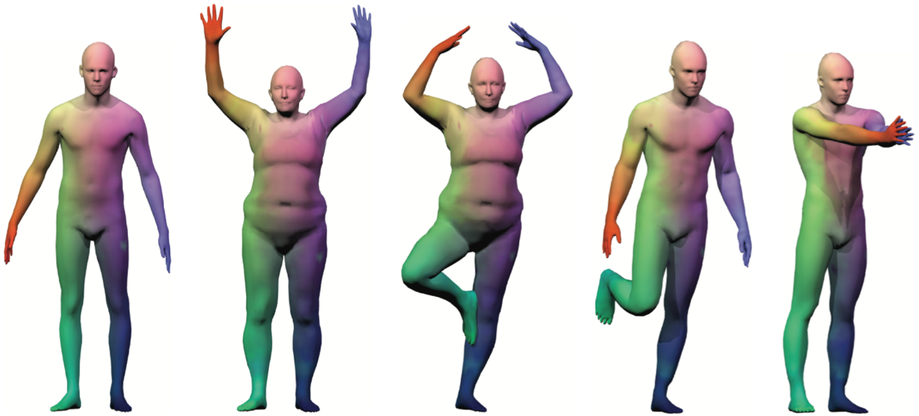
- Correspondence = labeling problem
- GCNN output $\mathbf{f}_\Theta(x)$ = probability distribution on reference $\mathcal{Y}$
- Minimize logistic regression cost w.r.t. GCNN parameters $\mathbf{\Theta}$

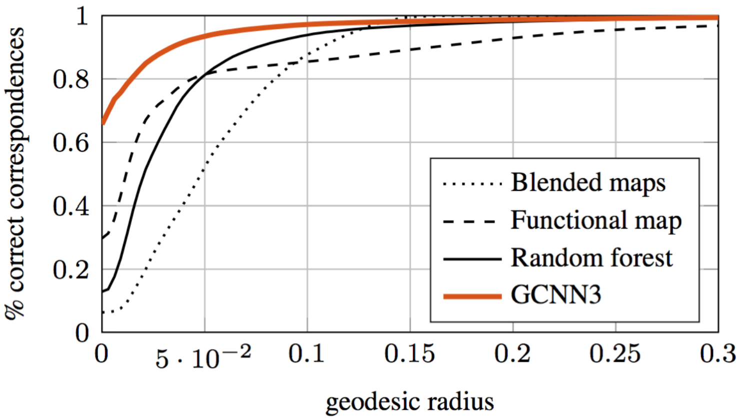
$$\ell(\mathbf{\Theta}) = -\sum_{(x, y^*(x)) \in \mathcal{T}} \langle \delta_{y^*(x)}, \log \mathbf{f}_\Theta(x) \rangle_{L^2(\mathcal{Y})}$$

image credit E. Rodolà

Geodesic convolutional neural networks on Riemannian manifolds, Masci, Boscaini, Bronstein, Vandergheynst 2015

Learning Correspondences with GCNN





Correspondence found using GCNN (similar colors encode corresponding points)

slide credit E. Rodolà

Geodesic convolutional neural networks on Riemannian manifolds, Masci, Boscaini, Bronstein, Vandergheynst 2015

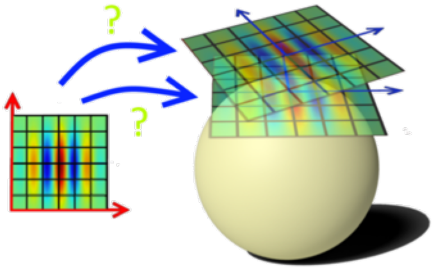
GCNN: Key challenges

Problem 1:

Local parametrization only defined up to rotation

Problem 2:

Angular max pooling *loses directional information*.

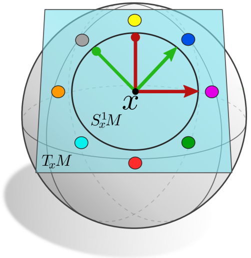


Parametrization Ambiguity

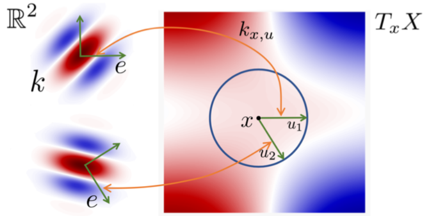
More recent solutions to kernel rotation ambiguity :

Designing *equivariant* networks

- **Multi-directional Geodesic Neural Networks (MDGCNN)**



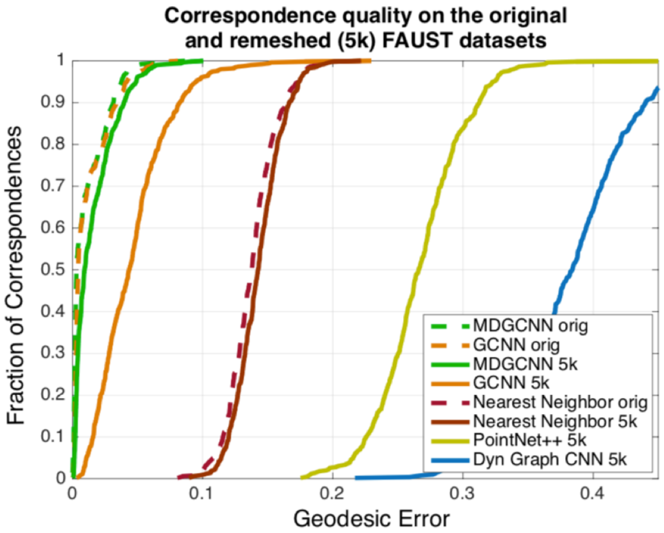
1. Work with *directional functions* $\varphi(x, v) \to \mathbb{R}, v \in T_x$
2. Keep responses *of all kernel rotations*: $(\varphi \star k)(x, v) = \varphi \star k_{x,v}$
3. Apply angular max pooling once *at the end of the network*

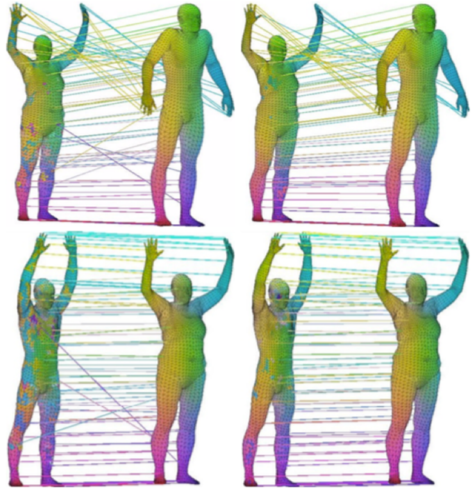


Multi-directional Geodesic Neural Networks via Equivariant Convolution, Poulenard, O. , SIGGRAPH Asia 2018

CNNs on Surfaces using Rotation-Equivariant Features, R. Wiersma, E. Eisemann, K. Hildebrandt, SIGGRAPH 2020

MDGCNN Results





GCNN **MDGCNN**

* remeshed datasets released as part of: *Continuous and Orientation-preserving Correspondences via Functional Maps*, J. Ren, A. Poulenard, P. Wonka, M. O, SIGGRAPH Asia 2018

Multi-directional Geodesic Neural Networks via Equivariant Convolution, Poulenard, O. , SIGGRAPH Asia 2018

Issues

Technical Problem:

Local parametrization only defined up to rotation

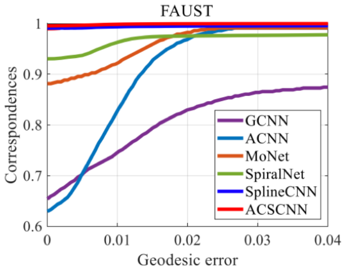
Even Bigger Problem:

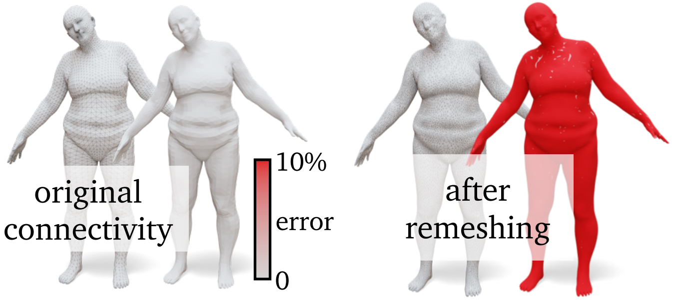
Local patch operations – slow and not robust!

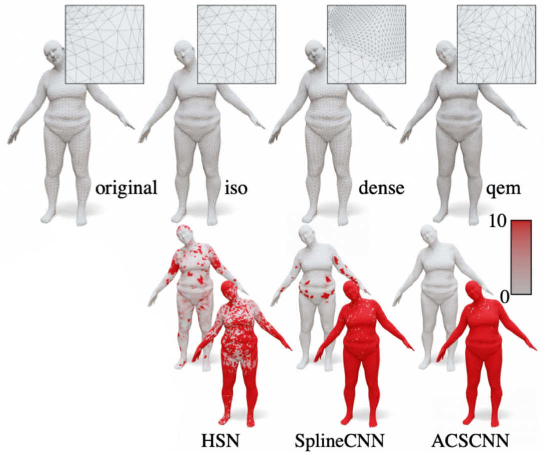
Main Question:

Can we avoid local patch parametrization and yet enable local *communication* on the surface?

Learning Correspondences – Results



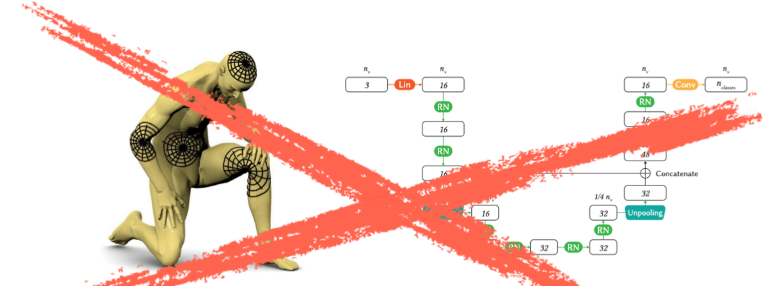




Many existing Geometric Deep Learning methods break under even mild remeshing.

Li, Qinsong, et al. *Shape correspondence using anisotropic Chebyshev spectral CNNs*, CVPR 2020.

Common Intrinsic Surface Learning



geodesic convolutions

pooling hierarchies

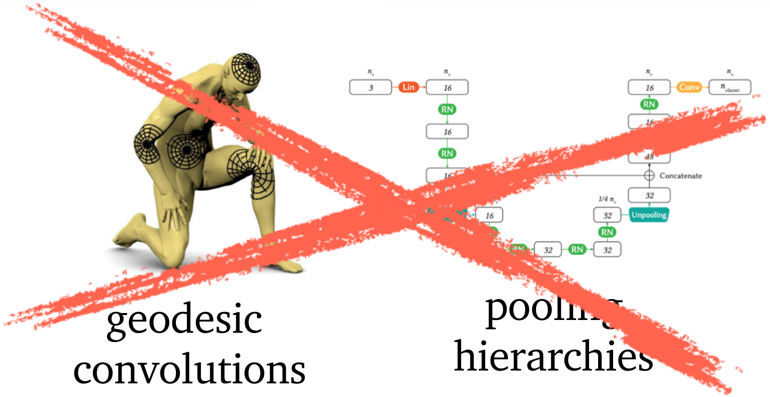
difficult on surfaces

source of non-robustness

use diffusion instead!

Alternative: Simple diffusion-based network

pointwise MLP + **learned diffusion** + gradient features



geodesic convolutions

pooling hierarchies

difficult on surfaces

source of non-robustness

use diffusion instead!

DiffusionNet: Discretization Agnostic Learning on Surfaces, N. Sharp, S. Attaiki, K. Crane, M.O., Trans. On Graph., 2022

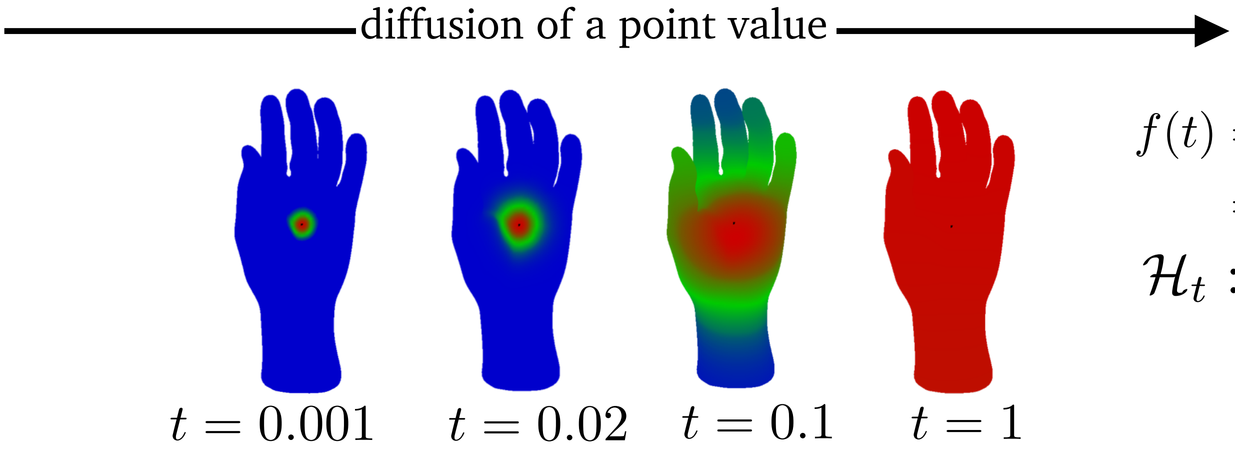
Basics: Laplacians and Diffusion

$$\frac{d}{dt} f(t) = \Delta f(t)$$

↳ Basic linear PDE

↳ defined on surfaces via the Laplace-Beltrami operator Δ

↳ implemented & well-studied on many domains



$$f(t) = \mathcal{H}_t f_0 = \exp(\Delta t) f_0$$

$\mathcal{H}_t$: Heat operator

Learned diffusion

~~convolution~~
~~pooling~~

✓ diffusion!

learned diffusion layer

$$h_t : \mathbb{R}^{\Omega \times k} \to \mathbb{R}^{\Omega \times k}$$

↳ parameterized by $t \in \mathbb{R}^k_{\geq 0}$

Key idea: the *diffusion time* is a learned parameter

↳ variable per-channel spatial support

↳ ranges from purely local to totally global

↳ automatically optimized during training

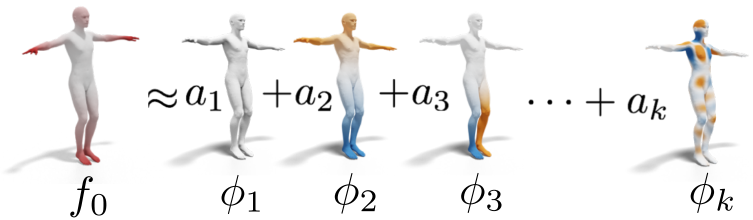
> **Lemma**: diffusion + pointwise MLPs can represent all (radially symmetric) convolutions.

DiffusionNet: Discretization Agnostic Learning on Surfaces, N. Sharp, S. Attaiki, K. Crane, M.O., Trans. On Graph., 2022

Evaluating diffusion

There are many ways to evaluate diffusion…

In Laplacian eigenbasis, diffusion is just multiplying by $e^{-\lambda t}$



I.e., if a function at time 0 is: $f_0 = \sum_i a_i \phi_i$

Then at time t (after diffusion):

$$f_t = \sum_i a_i e^{-t\lambda_i} \phi_i$$

Differentiable with respect to t.

Key idea: the *diffusion time* t is a learnable parameter.

Note: this is not spectral learning (we don't learn any *frequency-dependent* filters)!

Spatial gradient features

Typical architectures process input in blocks (of channels).

For DiffusionNet each channel *is a function on the shape.*

k functions Learned transformation *m functions*

E.g. for diffusion $f_j^{\text{out}} = \mathcal{H}_{t_j} f_j^{\text{in}}$

$$f_j^{\text{out}} = \exp(\Delta t_j) f_j^{\text{in}}$$

t_j learnable diffusion for channel *j*.

Spatial gradient features

Typical architectures process input in blocks (of channels).

For DiffusionNet each channel *is a function on the shape.*

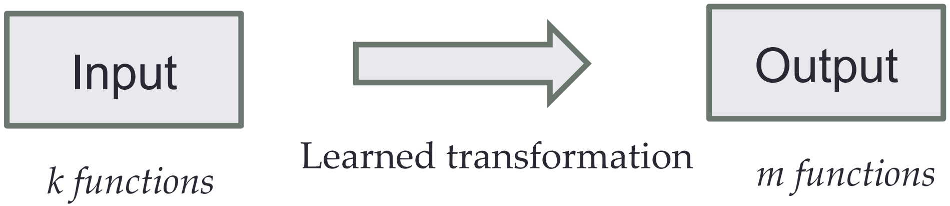
k functions Learned transformation *m functions*

E.g. for a *linear transformation* $f_j^{\text{out}}(x) = \sum_i A_{ij} f_i^{\text{in}}(x)$ A_{ij} are learned parameters (independent of *x*).

Spatial gradient features

Typical architectures process input in blocks (of channels).

For DiffusionNet each channel *is a function on the shape.*



E.g. for a *linear transformation* $f_j^{\text{out}}(x) = \sum_i A_{ij} f_i^{\text{in}}(x)$ A_{ij} are learned parameters (independent of *x*).

DiffusionNet gradient features are similar:

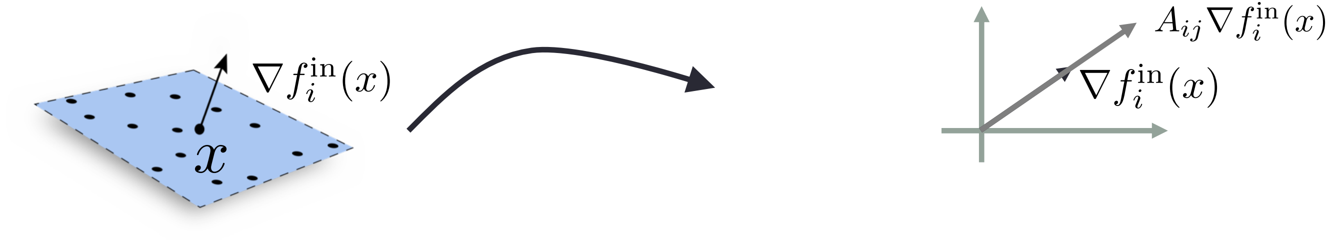
$$f_j^{\text{out}}(x) = \sum_i A_{ij} < \nabla f_i^{\text{in}}(x), \nabla f_j^{\text{in}}(x) >$$

Spatial gradient features

One more trick:

$$f_j^{\text{out}}(x) = \sum_i A_{ij} < \nabla f_i^{\text{in}}(x), \nabla f_j^{\text{in}}(x) >$$

We can represent $\nabla f_i^{\text{in}}(x)$ as a 2D vector in the tangent plane of x:



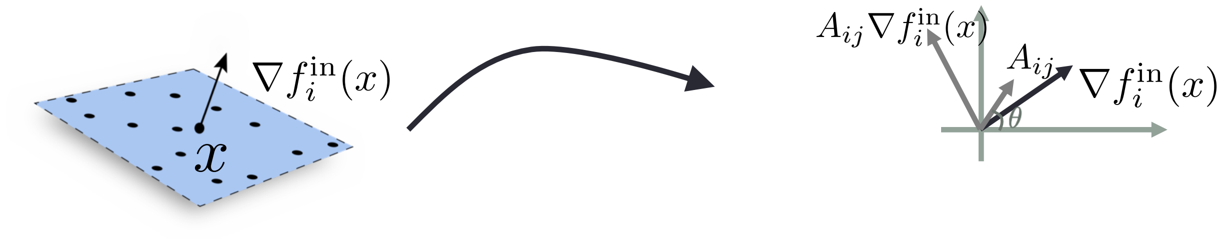
If A_{ij} is a *real number,* then $A_{ij} \nabla f_i^{\text{in}}(x)$ is just a *scaled* version of $\nabla f_i^{\text{in}}(x)$.

Spatial gradient features

One more trick:

$$f_j^{\text{out}}(x) = \sum_i A_{ij} < \nabla f_i^{\text{in}}(x), \nabla f_j^{\text{in}}(x) >$$

We can represent $\nabla f_i^{\text{in}}(x)$ as a 2D vector in the tangent plane of *x*:



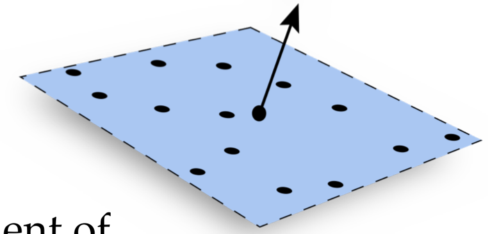
Instead of simply *scaling* $\nabla f_i^{\text{in}}(x)$, we can *scale and rotate* it in the 2D plane. Useful for injecting directional information into the pipeline. Can be conveniently done via *complex multiplication*.

Spatial gradient features

Challenge: we want to go beyond radially-symmetric filters

Solution: append extra features, dot products of spatial gradients

radially symmetric filters only | beyond radially symmetric filters



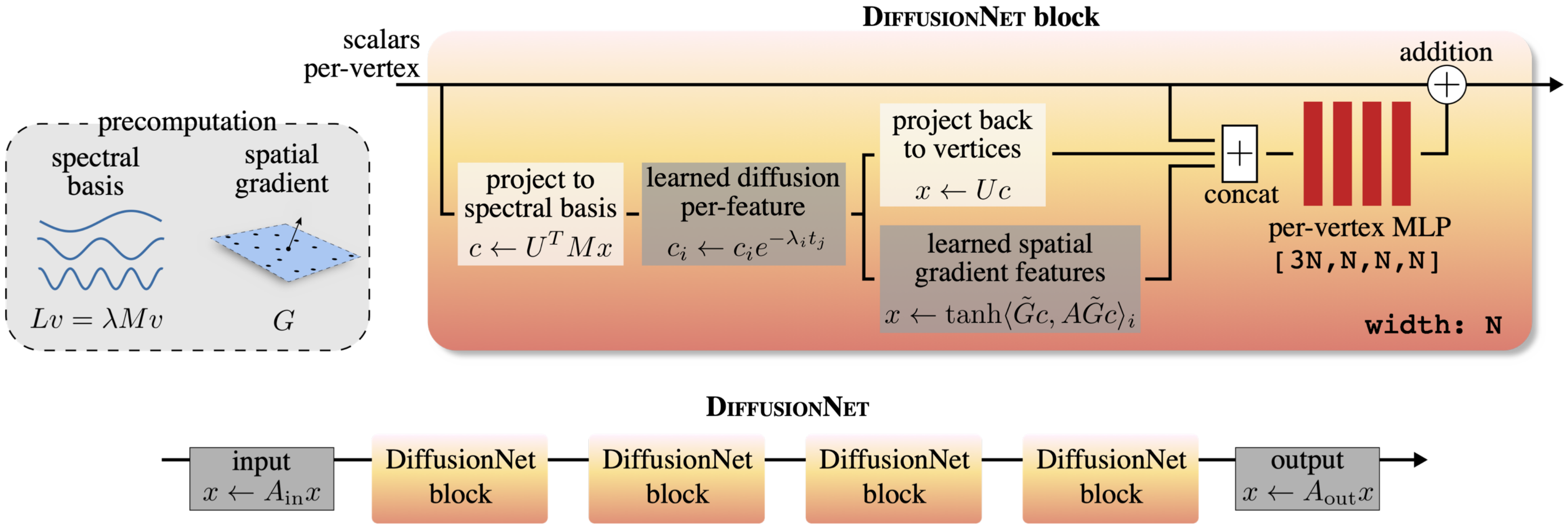
spatial gradient of scalar features $z = \nabla u$

learned scaling A

Our gradient features: $\tanh(\langle z, Az \rangle)$

$$output(i) = \tanh(\textstyle\sum_j \langle z(i), A_{ij} z(j) \rangle)$$

DiffusionNet Architecture

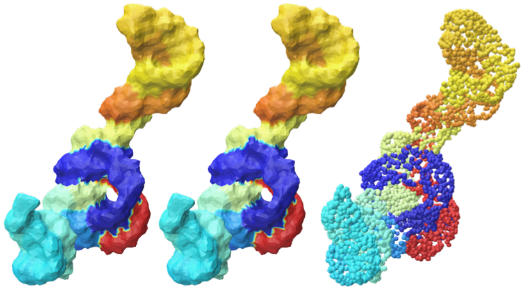


DiffusionNet: Discretization Agnostic Learning on Surfaces, N. Sharp, S. Attaiki, K. Crane, M.O., Trans. On Graph., 2022

DiffusionNet Results

State-of-the-art on several benchmarks!

RNA segmentation

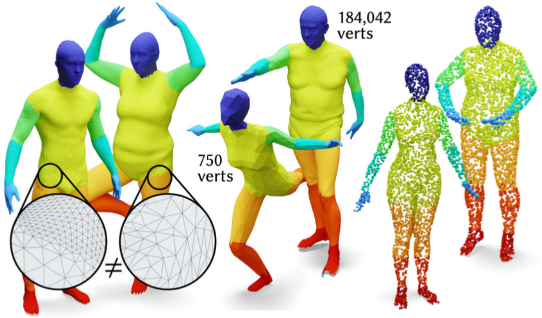


ground truth | mesh prediction | point cloud prediction

	Method	Accuracy
cloud	PointNet++ [Qi et al. 2017a]	74.4%
	PCNN [Atzmon et al. 2018]	78.0%
	SPHNet [Poulenard et al. 2019]	80.1%
	DiffusionNet - hks	84.0%
	DiffusionNet - xyz	**85.0%**
mesh	SplineCNN [Fey et al. 2018]	53.6%
	SurfaceNetworks [Kostrikov et al. 2018]	88.5%
	DiffusionNet - hks	91.0%
	DiffusionNet - xyz	**91.5%**

*

Human part segmentation



	Method	Accuracy
	GCNN [Masci et al. 2015]	86.4%
	ACNN [Boscaini et al. 2016]	83.7%
	Toric Cover [Maron et al. 2017]	88.0%
	PointNet++ [Qi et al. 2017a]	90.8%
	MDGCNN [Poulenard et al. 2018]	88.6%
	DGCNN [Wang et al. 2019]	89.7%
	SNGC [Haim et al. 2019]	91.0%
	HSN [Wiersma et al. 2020]	91.1%
	MeshWalker [Lahav and Tal 2020]	**92.7%**
	CGConv [Yang et al. 2021]	89.9%
	FC [Mitchel et al. 2021]	92.5%
	DiffusionNet - xyz	90.6%
	DiffusionNet - hks	91.7%
†see note	MeshCNN [Hanocka et al. 2019]	92.3%
	MeshWalker [Lahav and Tal 2020]	94.8%
	DiffusionNet - xyz	**95.5%**
	DiffusionNet - hks	**95.5%**
‡see note	PD-MeshNet [Milano et al. 2020]	85.6%
	HodgeNet [Smirnov and Solomon 2021]	85.0%
	DiffusionNet - xyz	**90.3%**
	DiffusionNet - hks	**90.8%**

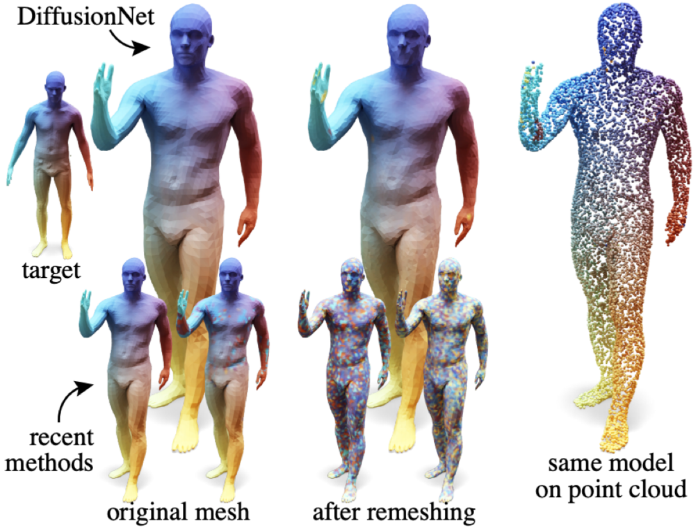
DiffusionNet: Discretization Agnostic Learning on Surfaces, N. Sharp, S. Attaiki, K. Crane, M.O., Trans. On Graph., 2022

*data released as part of Poulenard et al. *Effective rotation-invariant point cnn with spherical harmonics kernels*, 3DV 2019

DiffusionNet Results

Shape Correspondence with *DiffusionNet*:

Method / Dataset	**FAUST**	**SCAPE**	**FonS**	**SonF**
KPConv [Thomas et al. 2019]	3.1	4.4	11.0	6.0
KPConv - hks	2.9	3.3	10.6	5.5
HSN [Wiersma et al. 2020]	3.3	3.5	25.4	16.7
ACSCNN [Li et al. 2020c]	**2.7**	3.2	8.4	6.0
DiffusionNet - hks	**2.7**	**3.0**	3.8	**3.0**
DiffusionNet - xyz	**2.7**	**3.0**	**3.3**	**3.0**
+ ZoomOut	1.9	2.4	2.4	1.9



DiffusionNet: Discretization Agnostic Learning on Surfaces, N. Sharp, S. Attaiki, K. Crane, M.O., Trans. On Graph., 2022

Performance

(all on single RTX2070, in PyTorch)

↳ training at 42ms/input on 15k vertex/point inputs

↳ < 2.5 GB memory usage for training

↳ ~3 sec precompute per input on CPU (scipy)

Key property in practice:
runs easily on full-size meshes/clouds!
(no remeshing/downsampling)

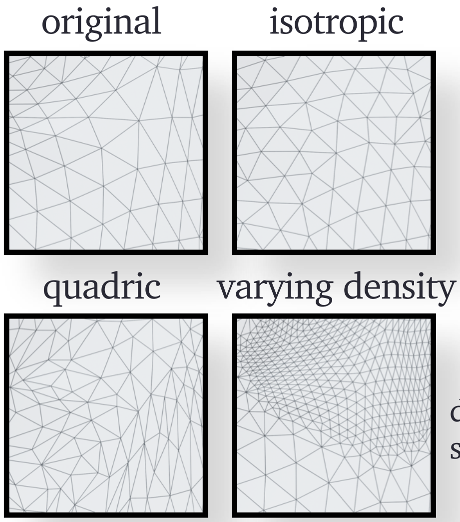
148k vertices

DiffusionNet: Discretization Agnostic Learning on Surfaces, N. Sharp, S. Attaiki, K. Crane, M.O., Trans. On Graph., 2022

Benefits

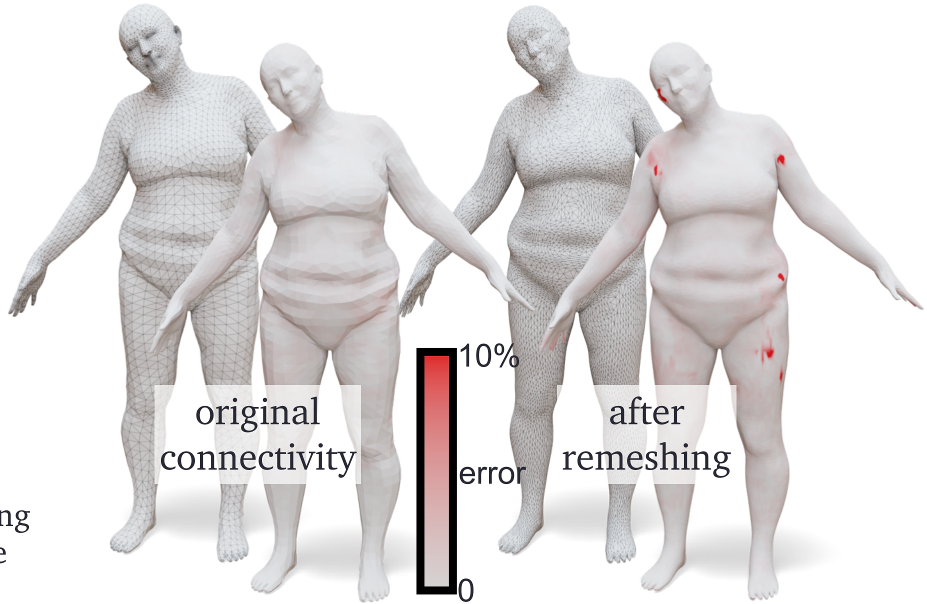
Sampling

DiffusionNet is quite robust to resampling by default



dataset for evaluating sampling invariance

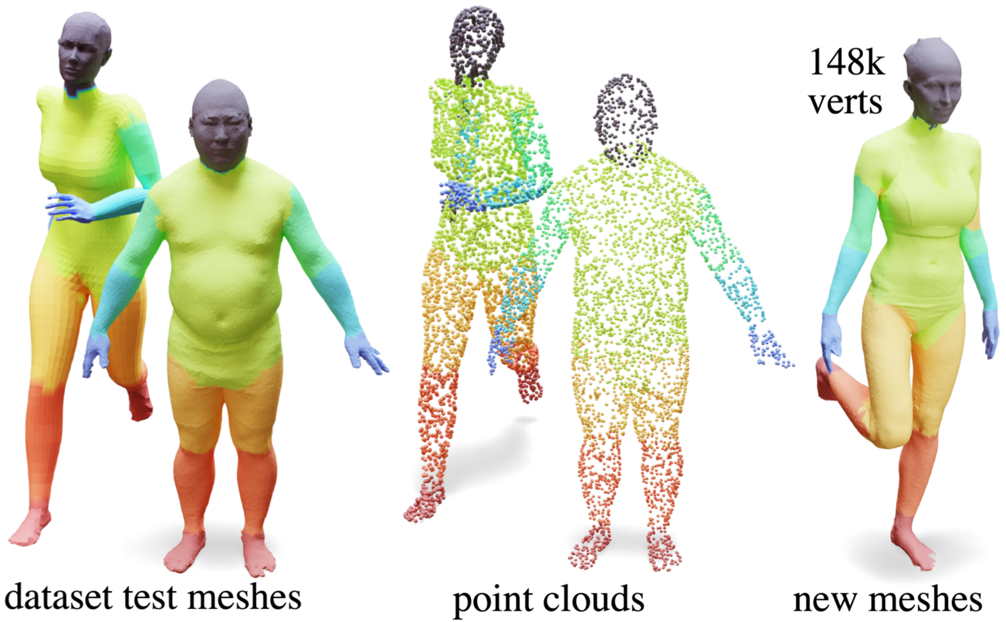
& point cloud sampling



Benefits

Transferability

train on meshes,
infer on a point cloud!



Conclusion

- Spectral filtering is efficient but not easy to generalize across domains

- Geodesic CNNs generalize well, but can be unstable and not robust.

- DiffusionNet a robust method for learning on surfaces.