Eurographics2019

The 40th Annual Conference of the European Association for Computer Graphics

May 6-10 Genova, Italy

3D Domains (intrinsic)

Niloy Mitra UCL

Iasonas Kokkinos UCL

Federico Monti USI Lugano

Emanuele Rodolà La Sapienza

Michael Bronstein Imperial College, USI Lugano

Or Litany Stanford University, Facebook

Leonidas Guibas Stanford University

http://geometry.cs.ucl.ac.uk/dl_for_CG/

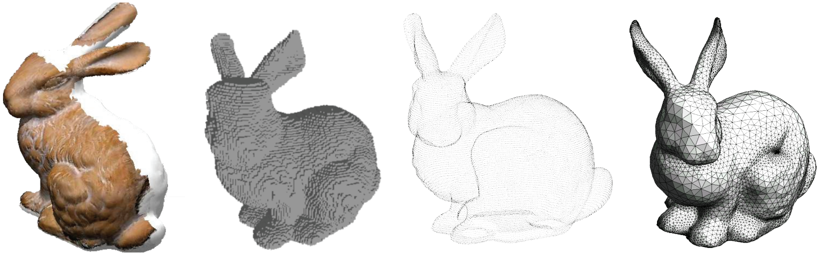
Timetable

			Niloy	Federico	Iasonas	Emanuele
Theory/Basics	Introduction	9:00	X	X	X	X
	Machine Learning Basics	~ 9:05	X			
	Neural Network Basics	~ 9:35		X		
	Alternatives to Direct Supervision (GANs)	~11:00			X	
State of the Art	Image Domain	~11:45			X	
	3D Domains (extrinsic)	~13:30	X			
	3D Domains (intrinsic)	~ 14:15				X
	Physics and Animation	~ 16:00	X			
	Discussion	~ 16:45	X	X	X	X

Sessions: A. 9:00-10:30 (**coffee**) B. 11:00-12:30 [**LUNCH**] C. 13:30-15:00 (**coffee**) D. 15:30-17:00

Deep Learning on Manifolds

Shape representation



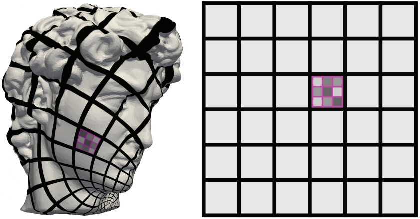
Extrinsic vs Intrinsic CNNs

Intrinsic

Different formulations of non-Euclidean CNNs

Spectral domain

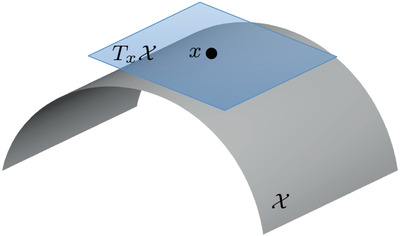
Spatial domain



Parametric domain

Riemannian geometry in one minute

- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x

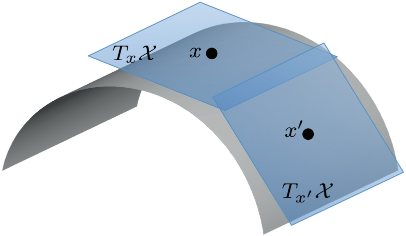


Riemannian geometry in one minute

- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

depending smoothly on x



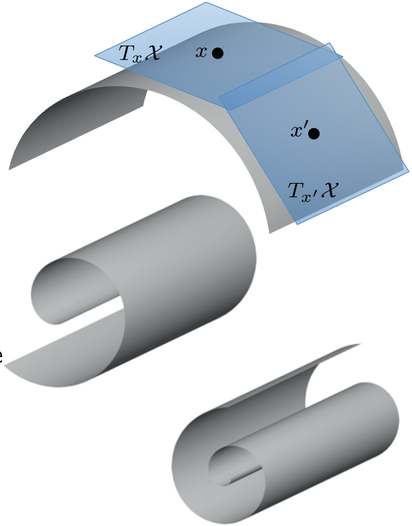
Riemannian geometry in one minute

- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \rightarrow \mathbb{R}$$

depending smoothly on x

Isometry = metric-preserving shape deformation



Riemannian geometry in one minute

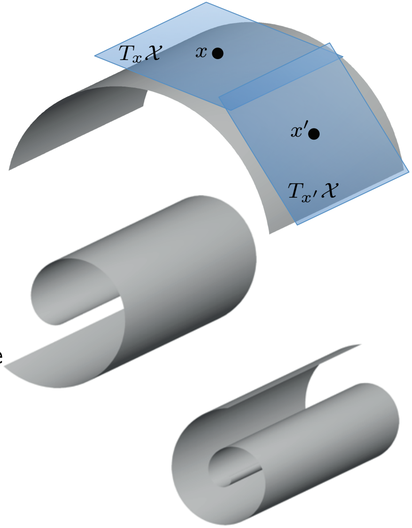
- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

depending smoothly on x

Isometry = metric-preserving shape deformation

Intrinsic = expressed solely in terms of the Riemannian metric



Riemannian geometry in one minute

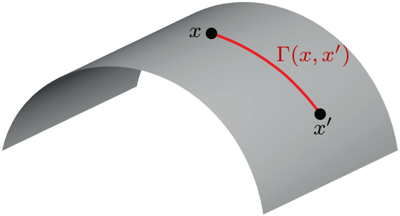
- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

  depending smoothly on x

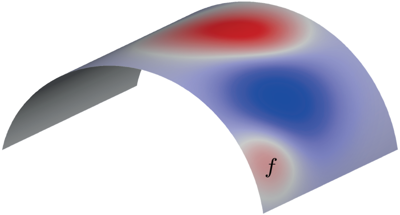
  Isometry = metric-preserving shape deformation

  Intrinsic = expressed solely in terms of the Riemannian metric
- Geodesic = shortest path on $\mathcal{X}$ between x and x'



Calculus on manifolds

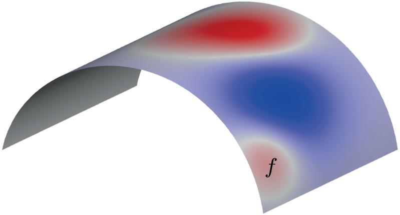
- Scalar field $f : \mathcal{X} \to \mathbb{R}$



Calculus on manifolds

- Scalar field $f : \mathcal{X} \to \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$



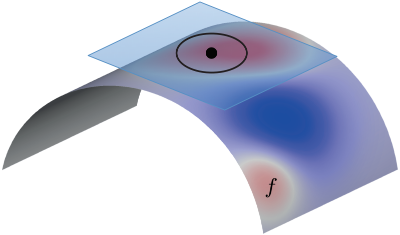
Calculus on manifolds

- Scalar field $f : \mathcal{X} \to \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$

- Laplacian operator $\Delta f = -\mathrm{div}(\nabla f)$

  "difference between $f(x)$ and average value of f around x"



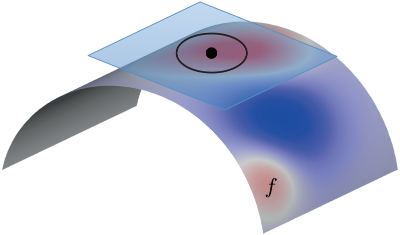
Calculus on manifolds

- Scalar field $f : \mathcal{X} \to \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$

- Laplacian operator $\Delta f = -\mathrm{div}(\nabla f)$

  "difference between $f(x)$ and average value of f around x"



  - Intrinsic (expressed solely in terms of the Riemannian metric)

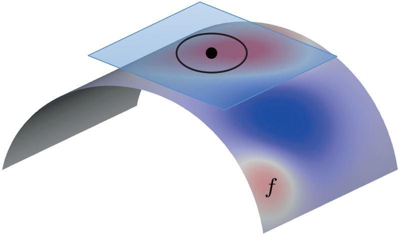
Calculus on manifolds

- Scalar field $f : \mathcal{X} \to \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x)g(x)dx$$

- Laplacian operator $\Delta f = -\mathrm{div}(\nabla f)$

  "difference between $f(x)$ and average value of f around x"



  - Intrinsic (expressed solely in terms of the Riemannian metric)
  - Isometry-invariant

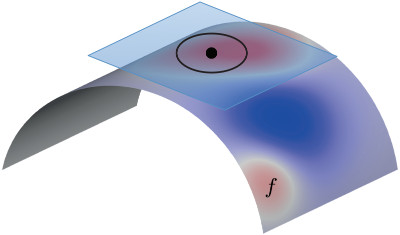
Calculus on manifolds

- Scalar field $f : \mathcal{X} \to \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$

- Laplacian operator $\Delta f = -\mathrm{div}(\nabla f)$

  "difference between $f(x)$ and average value of f around x"



- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})}$

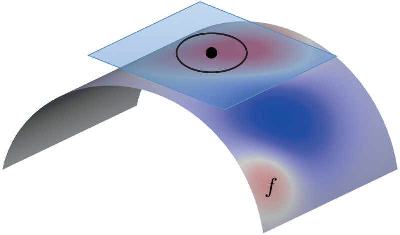
Calculus on manifolds

- Scalar field $f : \mathcal{X} \to \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x)g(x)dx$$

- Laplacian operator $\Delta f = -\mathrm{div}(\nabla f)$

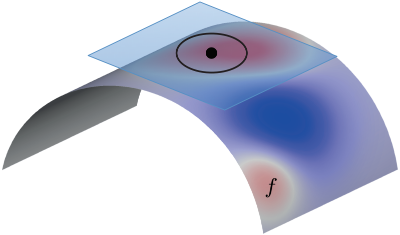
  "difference between $f(x)$ and average value of f around x"



- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow$ orthogonal eigenfunctions

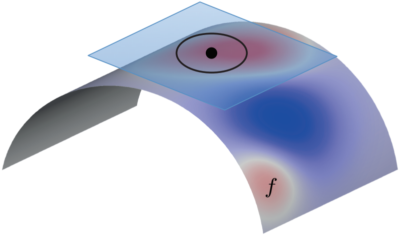
Calculus on manifolds

- Scalar field $f : \mathcal{X} \rightarrow \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product
$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$
- Laplacian operator $\Delta f = -\mathrm{div}(\nabla f)$
"difference between $f(x)$ and average value of f around x"



- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow$ orthogonal eigenfunctions
- Positive semidefinite

Calculus on manifolds



- Scalar field $f : \mathcal{X} \rightarrow \mathbb{R}$
- Hilbert space $L^2(\mathcal{X})$ with inner product

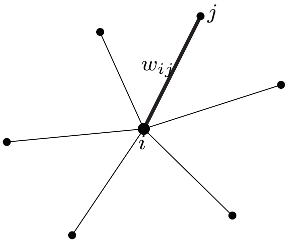
$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$

- Laplacian operator $\Delta f = -\mathrm{div}(\nabla f)$

  "difference between $f(x)$ and average value of f around x"

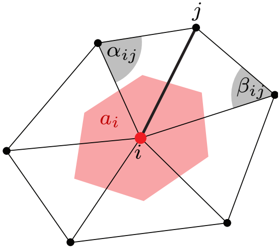
  - Intrinsic (expressed solely in terms of the Riemannian metric)
  - Isometry-invariant
  - Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow$ orthogonal eigenfunctions
  - Positive semidefinite $\Rightarrow$ non-negative eigenvalues

Discrete Laplacian



Undirected graph $(\mathcal{V}, \mathcal{E})$

$$(\Delta f)_i \approx \sum_{(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$



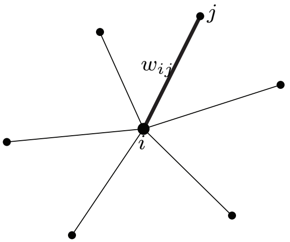
Triangular mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$

$$(\Delta f)_i \approx \frac{1}{a_i} \sum_{(i,j)\in\mathcal{E}} \frac{\cot\alpha_{ij} + \cot\beta_{ij}}{2}(f_i - f_j)$$

$a_i =$ local area element

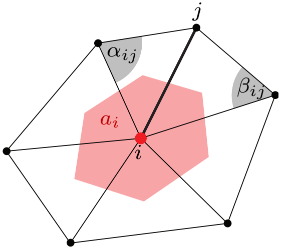
Tutte 1963; MacNeal 1949; Duffin 1959; Pinkall, Polthier 1993

Discrete Laplacian



Undirected graph $(\mathcal{V}, \mathcal{E})$

$$(\Delta f)_i \approx \sum_{(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$



Triangular mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$

$$(\Delta f)_i \approx \frac{1}{a_i} \sum_{(i,j)\in\mathcal{E}} \frac{\cot\alpha_{ij} + \cot\beta_{ij}}{2}(f_i - f_j)$$

a_i = local area element

In matrix-vector notation

$$\mathbf{\Delta f} = \mathbf{A}^{-1}(\mathbf{D} - \mathbf{W})\mathbf{f}$$

where $\mathbf{f} = (f_1, \ldots, f_n)^\top$, $\mathbf{W}$ is the stiffness matrix, $\mathbf{A} = \mathrm{diag}(a_1, \ldots, a_n)$ is the mass matrix, and $\mathbf{D} = \mathrm{diag}(\sum_{j\neq 1} w_{1j}, \ldots, \sum_{j\neq n} w_{nj})$

Tutte 1963; MacNeal 1949; Duffin 1959; Pinkall, Polthier 1993

Laplacian eigenfunctions and eigenvalues

$$\mathbf{\Delta\Phi} = \mathbf{\Phi\Lambda}$$

- $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal matrix of non-negative eigenvalues
- $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ a matrix of eigenvectors

Laplacian eigenfunctions and eigenvalues

$$\mathbf{A}^{-1}(\mathbf{D} - \mathbf{W})\mathbf{\Phi} = \mathbf{\Phi}\mathbf{\Lambda}$$

- $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal matrix of non-negative eigenvalues
- $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ a matrix of eigenvectors

Laplacian eigenfunctions and eigenvalues

$$(\mathbf{D} - \mathbf{W})\mathbf{\Phi} = \mathbf{A}\mathbf{\Phi}\mathbf{\Lambda}$$

- $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal matrix of non-negative eigenvalues
- $\mathbf{\Phi} = (\phi_1, \ldots, \phi_n)$ an $\mathbf{A}$-orthonormal matrix of eigenvectors ($\mathbf{\Phi}^\top \mathbf{A}\mathbf{\Phi} = \mathbf{I}$)

Laplacian eigenfunctions and eigenvalues

$$\mathbf{A}^{-1/2}(\mathbf{D}-\mathbf{W})\mathbf{A}^{-1/2}\mathbf{A}^{1/2}\mathbf{\Phi} = \mathbf{A}^{1/2}\mathbf{\Phi}\mathbf{\Lambda}$$

- $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal matrix of non-negative eigenvalues
- $\mathbf{\Phi} = (\phi_1, \ldots, \phi_n)$ an $\mathbf{A}$-orthonormal matrix of eigenvectors ($\mathbf{\Phi}^\top \mathbf{A}\mathbf{\Phi} = \mathbf{I}$)

Laplacian eigenfunctions and eigenvalues

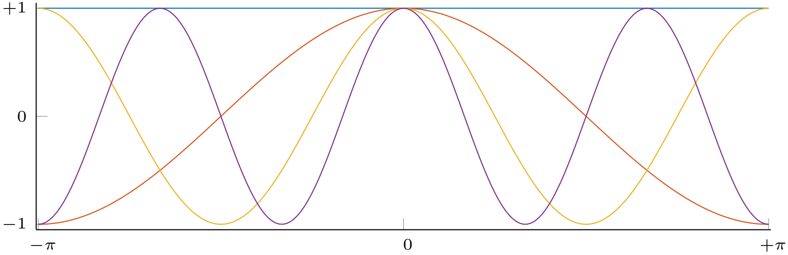
$$\mathbf{A}^{-1/2}(\mathbf{D}-\mathbf{W})\mathbf{A}^{-1/2}\boldsymbol{\Psi} = \boldsymbol{\Psi}\boldsymbol{\Lambda}$$

- $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal matrix of non-negative eigenvalues
- $\boldsymbol{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n)$ an orthonormal matrix of eigenvectors ($\boldsymbol{\Psi}^\top \boldsymbol{\Psi} = \mathbf{I}$)

Laplacian eigenfunctions and eigenvalues

$$\mathbf{A}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{A}^{-1/2}\mathbf{\Psi} = \mathbf{\Psi}\mathbf{\Lambda}$$

- $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal matrix of non-negative eigenvalues
- $\mathbf{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n)$ an orthonormal matrix of eigenvectors ($\mathbf{\Psi}^\top \mathbf{\Psi} = \mathbf{I}$)

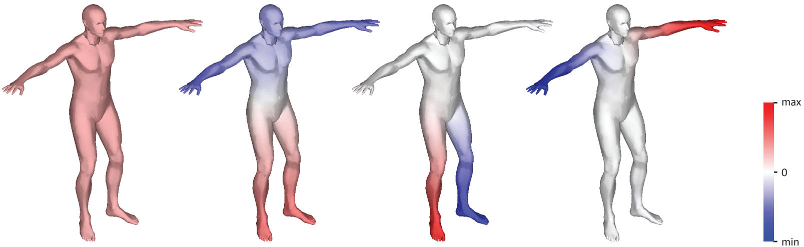


First eigenfunctions of 1D Euclidean Laplacian = standard Fourier basis

Laplacian eigenfunctions and eigenvalues

$$\mathbf{A}^{-1/2}(\mathbf{D}-\mathbf{W})\mathbf{A}^{-1/2}\boldsymbol{\Psi} = \boldsymbol{\Psi}\boldsymbol{\Lambda}$$

- $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal matrix of non-negative eigenvalues
- $\boldsymbol{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n)$ an orthonormal matrix of eigenvectors ($\boldsymbol{\Psi}^\top\boldsymbol{\Psi} = \mathbf{I}$)



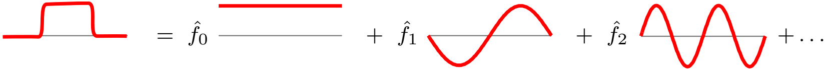
ϕ_1 ϕ_2 ϕ_3 ϕ_4

First eigenfunctions of a manifold Laplacian

Fourier analysis: Euclidean

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

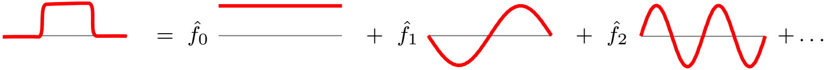
$$f(x) = \sum_{k \geq 0} \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx' \, e^{ikx}$$



Fourier analysis: Euclidean

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

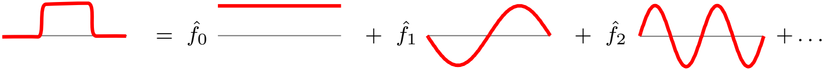
$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx'}_{\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}} e^{ikx}$$



Fourier analysis: Euclidean

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

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

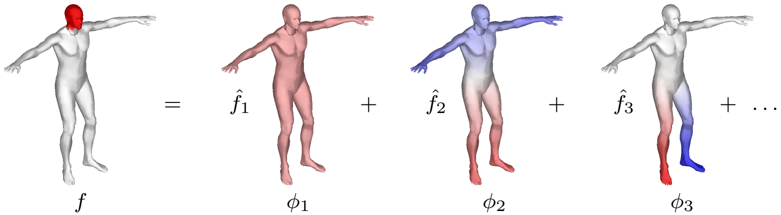


Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

Fourier analysis: non-Euclidean

A function $f : \mathcal{X} \rightarrow \mathbb{R}$ 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

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

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \rightarrow \mathbb{R}$ 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: 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: 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

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

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

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

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} \quad = \quad \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \dots, f_n)^\top$ and $\mathbf{g} = (g_1, \dots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{circulant matrix}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \dots, f_n)^\top$ and $\mathbf{g} = (g_1, \dots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{diagonalized by Fourier basis}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \boldsymbol{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \boldsymbol{\Phi}^\top \mathbf{f}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \mathbf{\Phi} \begin{bmatrix} \hat{f}_1 \cdot \hat{g}_1 \\ \vdots \\ \hat{f}_n \cdot \hat{g}_n \end{bmatrix}$$

Spectral convolution

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

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

Spectral convolution

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

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

Spectral convolution

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

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

Spectral convolution

Spectral 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

$$\mathbf{f} \star \mathbf{g} = \mathbf{\Phi} \, (\mathbf{\Phi}^\top \mathbf{g}) \circ (\mathbf{\Phi}^\top \mathbf{f})$$

Spectral convolution

Spectral 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

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

Spectral convolution

Spectral 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

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

Spectral convolution

Spectral 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

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

- Not shift-invariant! ($\mathbf{G}$ has no circulant structure)

Spectral convolution

Spectral 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

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

- Not shift-invariant! ($\mathbf{G}$ has no circulant structure)
- Filter coefficients depend on basis $\phi_1, \ldots, \phi_n$

Spectral CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \boldsymbol{\Phi}\mathbf{W}\boldsymbol{\Phi}^\top\mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

Bruna et al. 2014 (first applied to graphs)

Spectral CNN

Convolution expressed in the spectral domain

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

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

☹ Filters are basis-dependent ⇒ do not generalize across domains

Bruna et al. 2014 (first applied to graphs)

Spectral CNN

Convolution expressed in the spectral domain

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

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

☹ Filters are basis-dependent $\Rightarrow$ do not generalize across domains
☹ $\mathcal{O}(n)$ parameters per layer

Bruna et al. 2014 (first applied to graphs)

Spectral CNN

Convolution expressed in the spectral domain

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

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

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

Bruna et al. 2014 (first applied to graphs)

Spectral CNN

Convolution expressed in the spectral domain

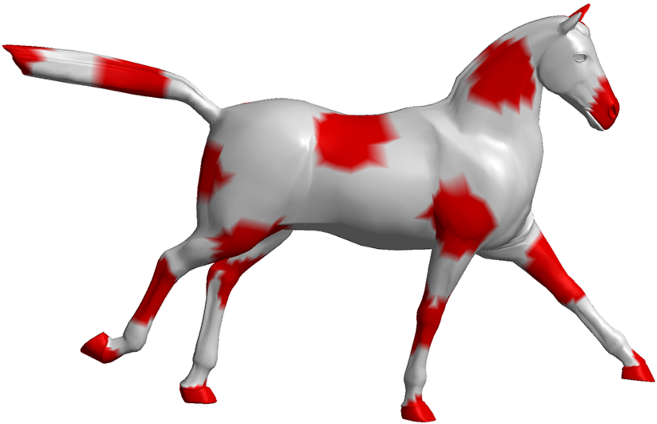
$$\mathbf{g} = \mathbf{\Phi} \mathbf{W} \mathbf{\Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ do not generalize across domains
- ☹ $\mathcal{O}(n)$ 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 et al. 2014 (first applied to graphs)

Basis dependence



Function $\mathbf{f}$

Basis dependence

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

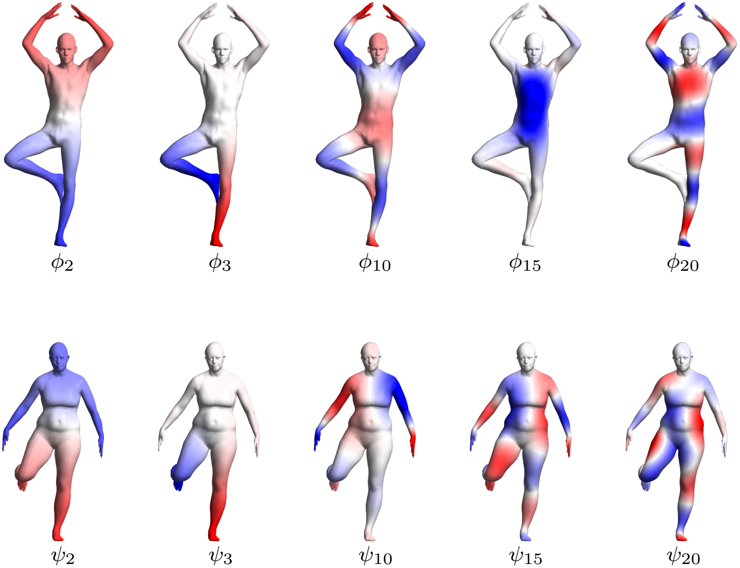
Basis dependence

filter, different basis $\mathbf{\Psi W \Psi^\top f}$

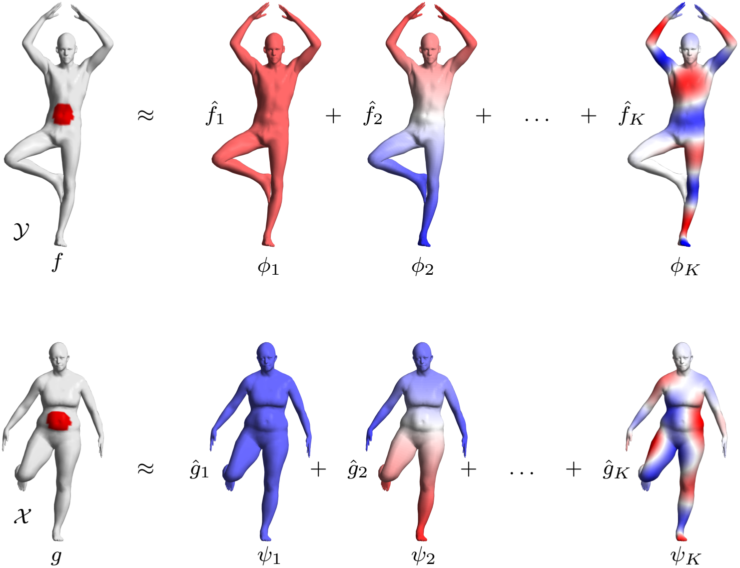
Basis dependence

ncy Laplacian eigenvector ϕ_{50}

Laplacian eigenbases on non-isometric domains

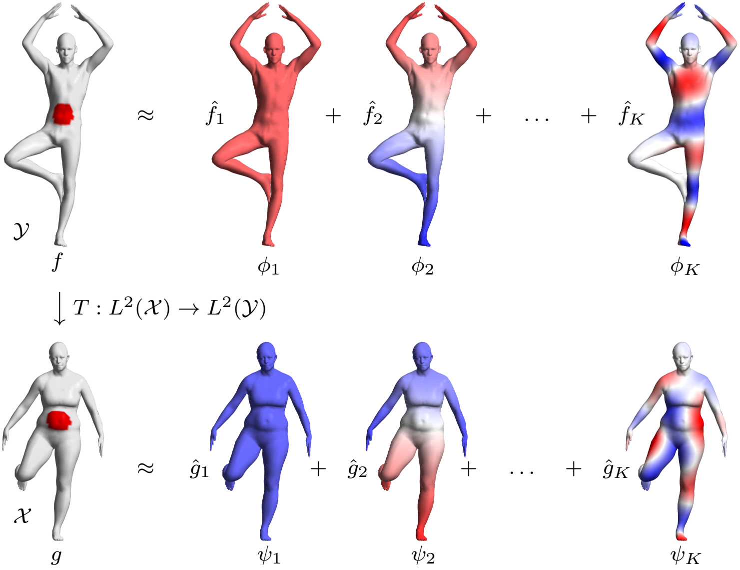


Functional maps



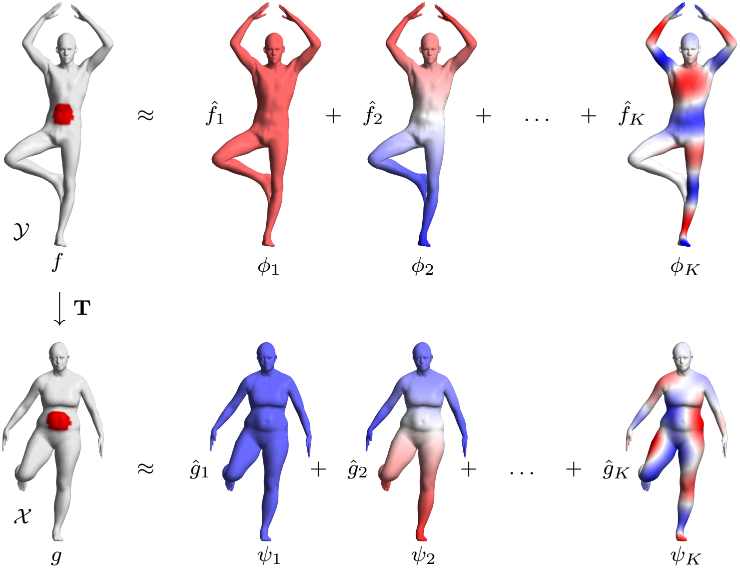
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



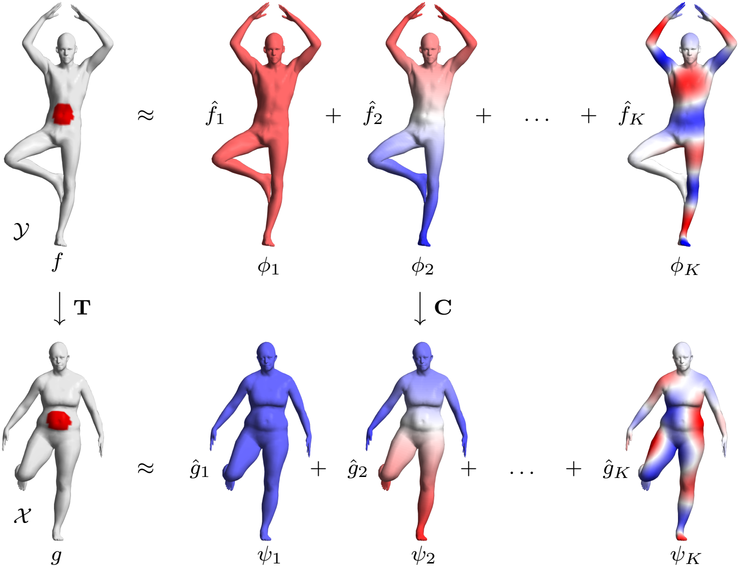
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



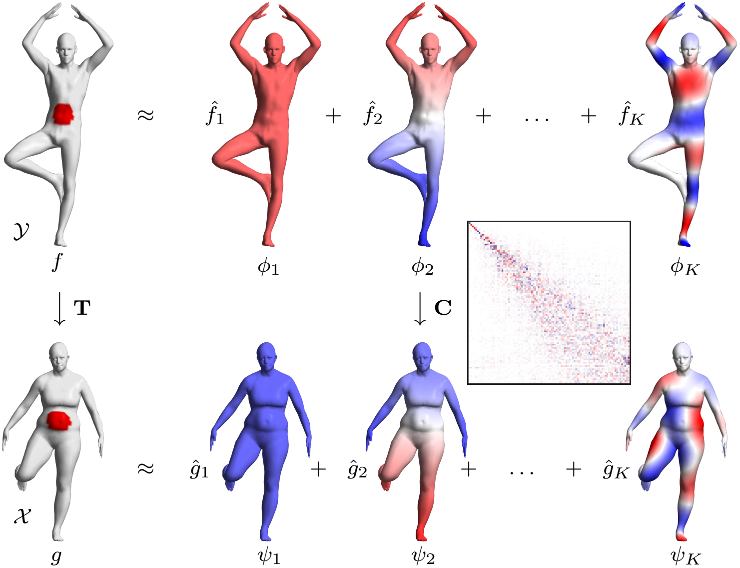
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



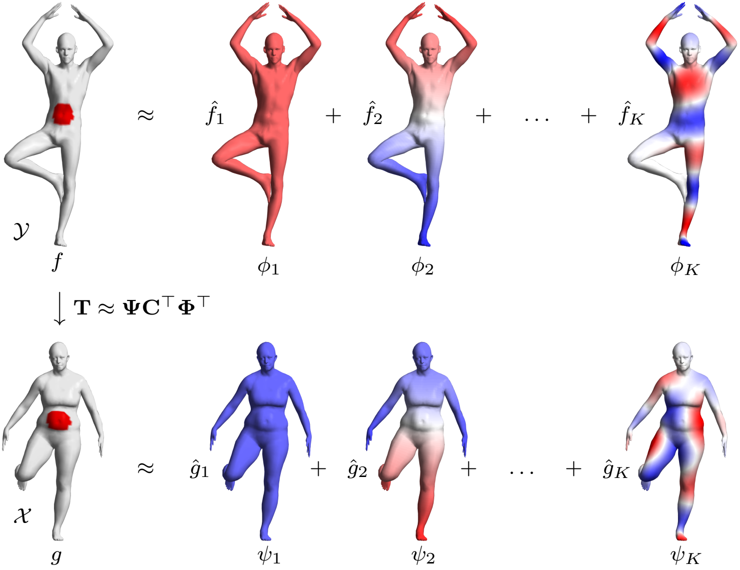
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



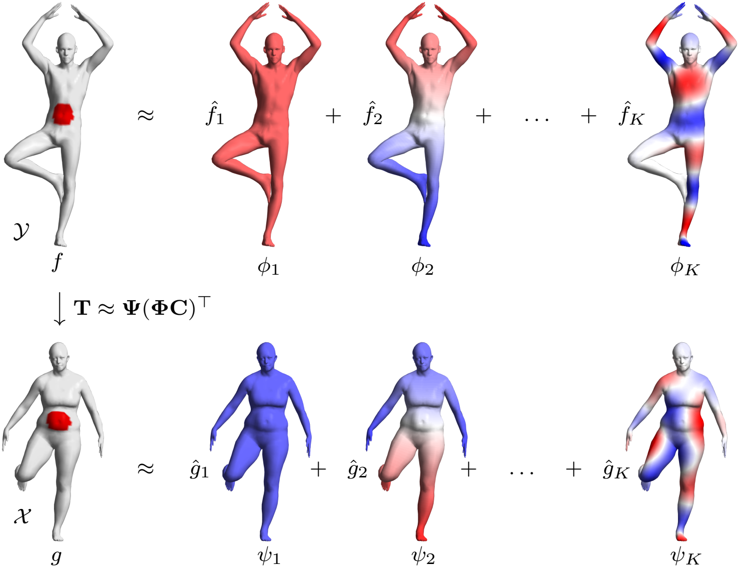
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Basis synchronization with functional maps



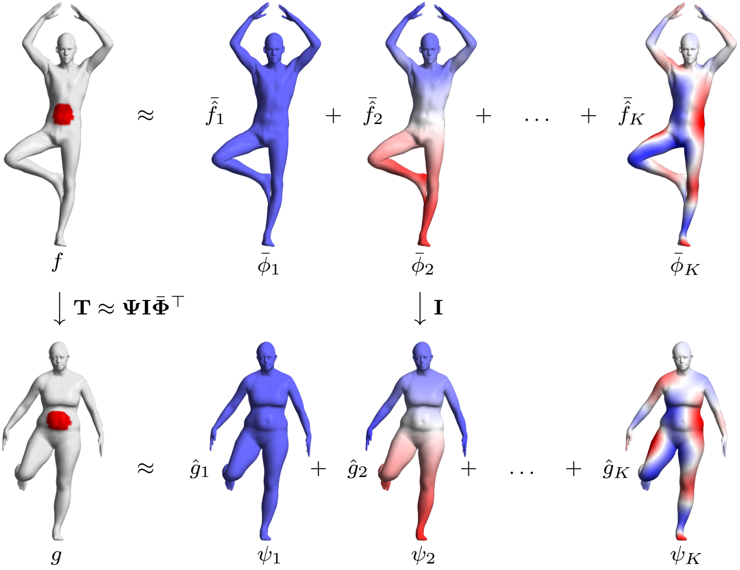
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Basis synchronization with functional maps



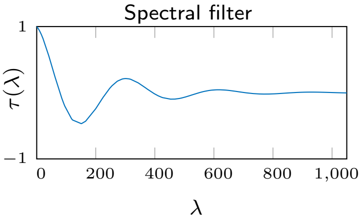
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

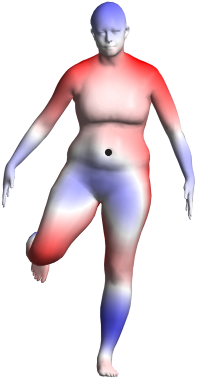
Basis synchronization with functional maps



Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Filtering in different bases





$\mathbf{\Phi}\tau(\mathbf{\Lambda}_{\mathbf{\Phi}})\mathbf{\Phi}^\top\boldsymbol{\delta}_0$

$\mathbf{\Psi}\tau(\mathbf{\Lambda}_{\mathbf{\Psi}})\mathbf{\Psi}^\top\boldsymbol{\delta}_0$

Apply spectral filter $\tau(\lambda)$ in different bases $\mathbf{\Phi}$ and $\mathbf{\Psi}$

$\Rightarrow$ **different results!**

Filtering in different bases

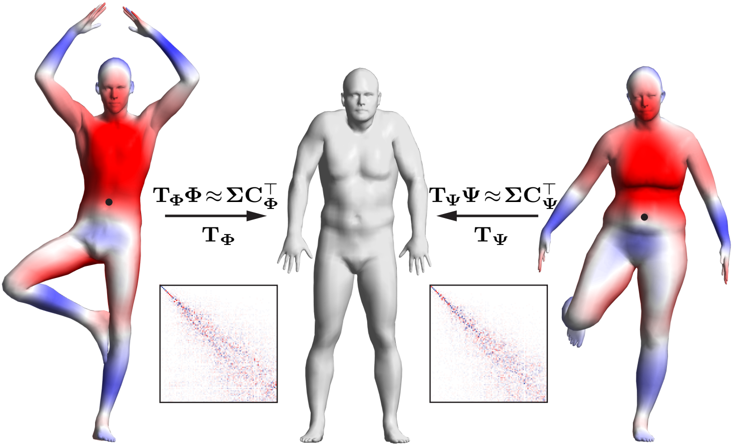
$\boldsymbol{\Phi}\tau(\boldsymbol{\Lambda}_{\boldsymbol{\Phi}})\boldsymbol{\Phi}^\top\boldsymbol{\delta}_0$

Canonical shape
with basis $\boldsymbol{\Sigma}, \boldsymbol{\Lambda}$

$\boldsymbol{\Psi}\tau(\boldsymbol{\Lambda}_{\boldsymbol{\Psi}})\boldsymbol{\Psi}^\top\boldsymbol{\delta}_0$

Apply spectral filter $\tau(\lambda)$ in different bases $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$
$\Rightarrow$ **different results!**

Filtering in synchronized bases



$\Phi \mathbf{C}_\Phi \tau(\Lambda) \mathbf{C}_\Phi^\top \Phi^\top \delta_0$

Canonical shape with basis Σ, Λ

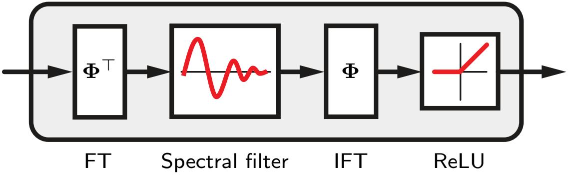
$\Psi \mathbf{C}_\Psi \tau(\Lambda) \mathbf{C}_\Psi^\top \Psi^\top \delta_0$

Apply spectral filter $\tau(\lambda)$ in synchronized bases $\Phi \mathbf{C}_\Phi$ and $\Psi \mathbf{C}_\Psi$

$\Rightarrow$ **similar results!**

Yi et al. 2017

Spectral CNN



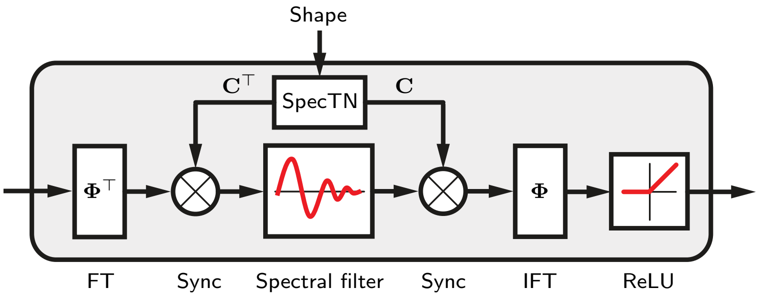
Convolutional filter of a Spectral CNN

☹ Fixed basis $\Rightarrow$ Does not generalize across domains

☺ Possible $\mathcal{O}(n)$ complexity avoiding explicit FT and IFT

Bruna et al. 2014

Spectral Transformer Network



Convolutional filter of a Spectral Transformer Network

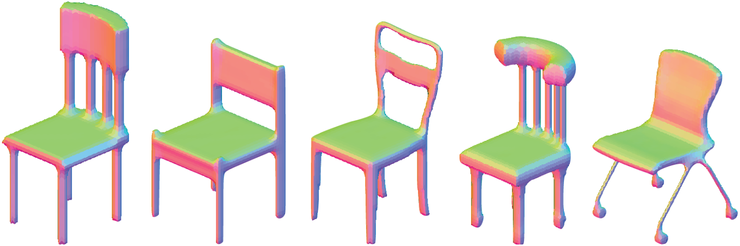
☺ Basis synchronization allows generalization across domains

☹ Explicit FT and IFT

(Jaderberg et al. 2015); Yi et al. 2017

Example: normal prediction with SpecTN

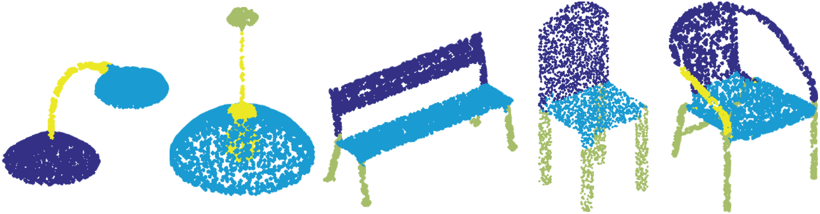
Predicted



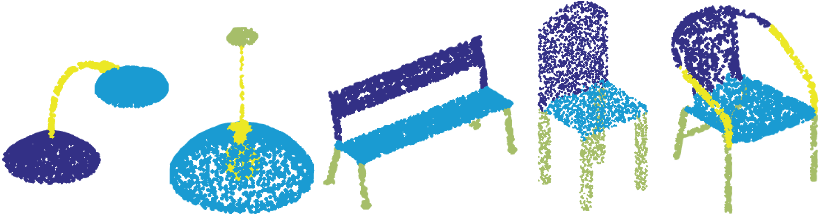
Groundtruth

Yi et al. 2017

Example: shape segmentation with SpecTN



Predicted



Groundtruth

Yi et al. 2017

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the filter

$$\tau(\mathbf{\Delta})\mathbf{f} = \mathbf{\Phi}\tau(\mathbf{\Lambda})\mathbf{\Phi}^\top \mathbf{f}$$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the filter

$$\tau(\boldsymbol{\Delta})\mathbf{f} = \boldsymbol{\Phi} \begin{pmatrix} \tau(\lambda_1) & & \\ & \ddots & \\ & & \tau(\lambda_n) \end{pmatrix} \boldsymbol{\Phi}^\top \mathbf{f}$$

Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

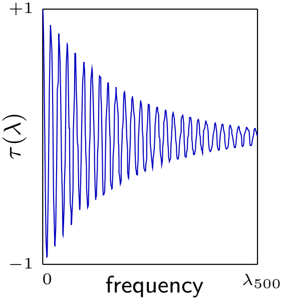
Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the parametric filter with learnable parameters $\boldsymbol{\alpha}$

$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \boldsymbol{\Phi} \begin{pmatrix} \tau_{\boldsymbol{\alpha}}(\lambda_1) & & \\ & \ddots & \\ & & \tau_{\boldsymbol{\alpha}}(\lambda_n) \end{pmatrix} \boldsymbol{\Phi}^\top \mathbf{f}$$

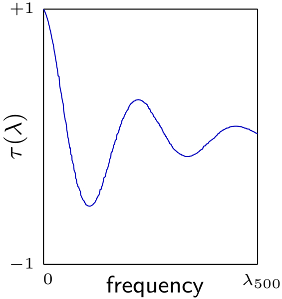
Bruna et al. 2014; Henaff, Bruna, LeCun 2015

Localization and smoothness



smooth spectral filter (delocalized in space)

Localization and smoothness



mooth spectral filter (localized in space)

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{\ell=0}^{r} \alpha_\ell \lambda^\ell$$

where $\boldsymbol{\alpha} = (\alpha_0, \dots, \alpha_r)^\top$ is the vector of filter parameters

Defferrard, Bresson, Vandergheynst 2016

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{\ell=0}^{r} \alpha_\ell \lambda^\ell$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

☺ $\mathcal{O}(1)$ parameters per layer

Defferrard, Bresson, Vandergheynst 2016

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{\ell=0}^{r} \alpha_\ell \lambda^\ell$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

☺ $\mathcal{O}(1)$ parameters per layer

☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(nr)$ complexity

Defferrard, Bresson, Vandergheynst 2016

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) = \sum_{\ell=0}^{r} \alpha_\ell \boldsymbol{\Delta}^\ell$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(nr)$ complexity
- ☺ Filters have guaranteed r-ring support

Defferrard, Bresson, Vandergheynst 2016

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) = \sum_{\ell=0}^{r} \alpha_\ell \boldsymbol{\Delta}^\ell$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

☺ $\mathcal{O}(1)$ parameters per layer

☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(nr)$ complexity

☺ Filters have guaranteed r-ring support

☹ Mesh dependent !

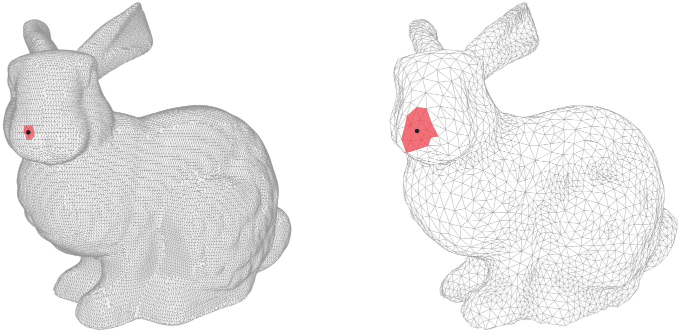
Defferrard, Bresson, Vandergheynst 2016

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) = \sum_{\ell=0}^{r} \alpha_\ell \boldsymbol{\Delta}^\ell$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters



Defferrard, Bresson, Vandergheynst 2016

Spectral CNN rational filters (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{\ell=1}^{r} c_\ell (h\lambda - i)^\ell (h\lambda + i)^{-\ell} \Big\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

Levie et al. 2017

Spectral CNN rational filters (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\left\{ \sum_{\ell=1}^{r} c_\ell (h\lambda - i)^\ell (h\lambda + i)^{-\ell} \right\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☹ $\mathcal{O}(n^3)$ computational complexity with direct matrix inversion

Levie et al. 2017

Spectral CNN rational filters (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{\ell=1}^{r} c_\ell (h\lambda - i)^\ell (h\lambda + i)^{-\ell} \Big\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☺ $\mathcal{O}(n)$ computational complexity with **Jacobi approximate matrix inversion**

Levie et al. 2017

Spectral CNN rational filters (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\left\{ \sum_{\ell=1}^{r} c_\ell (h\lambda - i)^\ell (h\lambda + i)^{-\ell} \right\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☺ $\mathcal{O}(n)$ computational complexity with **Jacobi approximate matrix inversion**
- ☺ Spectral zoom property allowing to better localize in frequency

Levie et al. 2017

Spectral CNN rational filters (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\left\{ \sum_{\ell=1}^{r} c_\ell (h\lambda - i)^\ell (h\lambda + i)^{-\ell} \right\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☺ $\mathcal{O}(n)$ computational complexity with **Jacobi approximate matrix inversion**
- ☺ Spectral zoom property allowing to better localize in frequency
- ☺ Richer class of filters than polynomials for the same order

Levie et al. 2017

Spectral CNN rational filters (CayleyNet)

Represent spectral transfer function as a Cayley polynomial or order r

$$\tau_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\left\{ \sum_{\ell=1}^{r} c_\ell (h\lambda - i)^\ell (h\lambda + i)^{-\ell} \right\}$$

where the filter parameters are the vector of real/complex coefficients $\mathbf{c} = (c_0, \ldots, c_r)^\top$ and the spectral zoom h

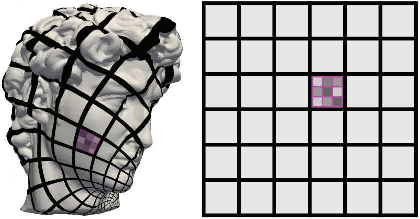
- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed exponential spatial decay
- ☺ $\mathcal{O}(n)$ computational complexity with **Jacobi approximate matrix inversion**
- ☺ Spectral zoom property allowing to better localize in frequency
- ☺ Richer class of filters than polynomials for the same order
- ☺ Scale invariance

Levie et al. 2017

Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain



Parametric domain

Convolution

Euclidean

Spatial domain

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

Spectral domain

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

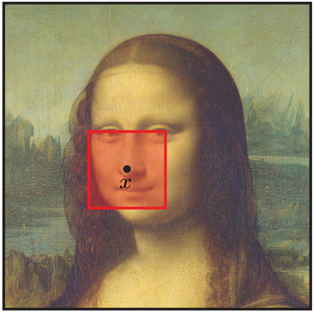
'Convolution Theorem'

Non-Euclidean

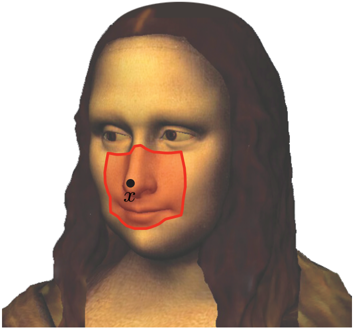
?

$$\widehat{(f \star g)}_k = \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})}$$

Spatial convolution



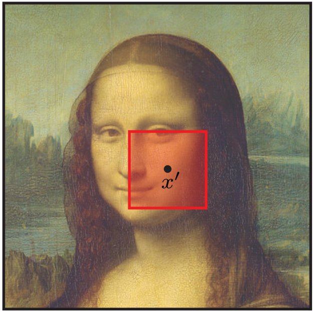
Euclidean



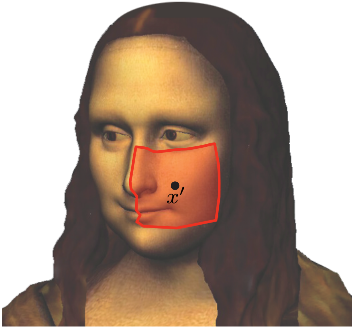
Non-Euclidean

Masci et al. 2015

Spatial convolution



Euclidean

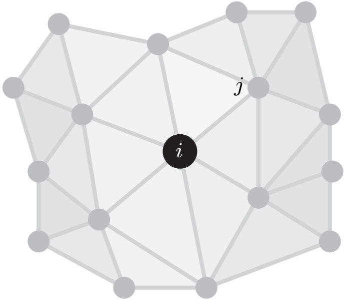


Non-Euclidean

Masci et al. 2015

Convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)

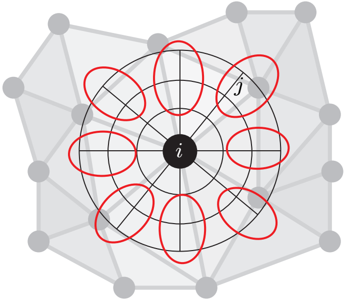


Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)

- Local weights $w_1(\mathbf{u}), \ldots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$



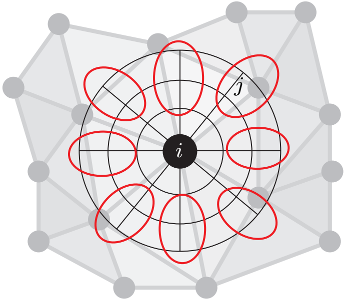
Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)

- Local weights $w_1(\mathbf{u}), \ldots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$, e.g. Gaussians

$$w_\ell = \exp\left(-(\mathbf{u} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1}(\mathbf{u} - \boldsymbol{\mu}_\ell)\right)$$

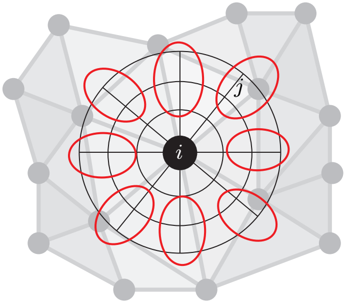


Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)
- Local weights $w_1(\mathbf{u}), \ldots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$, e.g. Gaussians

$$w_\ell = \exp\left(-(\mathbf{u}-\boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1}(\mathbf{u}-\boldsymbol{\mu}_\ell)\right)$$



- Spatial convolution with filter g

$$\mathbf{x}_i' \propto \sum_{\ell=1}^{L} g_\ell \sum_{j=1}^{n} w_\ell(\mathbf{u}_{ij})\mathbf{x}_j$$

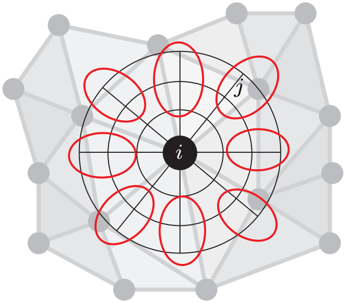
where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Convolution on meshes

- Local system of coordinates $\mathbf{u}_{ij}$ around i (e.g. geodesic polar)
- Local weights $w_1(\mathbf{u}), \ldots, w_L(\mathbf{u})$ w.r.t. $\mathbf{u}$, e.g. Gaussians

  $$w_\ell = \exp\left(-(\mathbf{u} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1}(\mathbf{u} - \boldsymbol{\mu}_\ell)\right)$$



- Spatial convolution with filter g

$$\mathbf{x}'_i \propto \sum_{\ell=1}^{L} g_\ell \underbrace{\sum_{j=1}^{n} w_\ell(\mathbf{u}_{ij})\mathbf{x}_j}_{\text{patch operator}}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

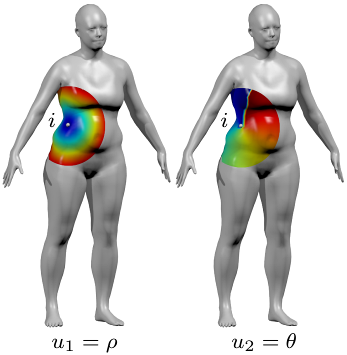
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

ρ_{ij} = geodesic distance from i to j

θ_{ij} = direction of geodesic from i to j



Masci et al. 2015; Monti et al. 2017

Geodesic polar coordinates

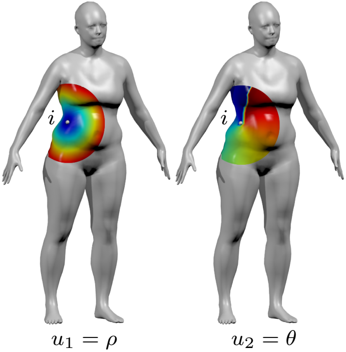
- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

ρ_{ij} = geodesic distance from i to j

θ_{ij} = direction of geodesic from i to j

- Orientation ambiguity!



Masci et al. 2015; Monti et al. 2017

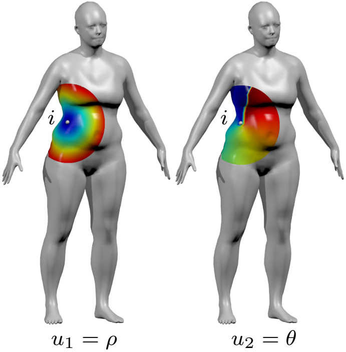
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

ρ_{ij} = geodesic distance from i to j

θ_{ij} = direction of geodesic from i to j



- Orientation ambiguity!
  - Canionical direction (e.g. intrinsic vector field, max curvature direction)

Masci et al. 2015; Monti et al. 2017

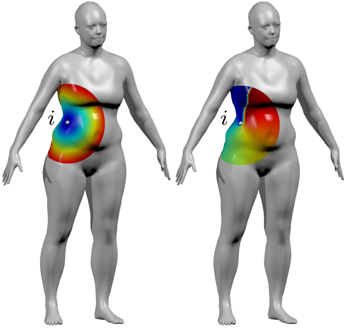
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

ρ_{ij} = geodesic distance from i to j

θ_{ij} = direction of geodesic from i to j



$u_1 = \rho$ $u_2 = \theta$

- Orientation ambiguity!
  - Canionical direction (e.g. intrinsic vector field, max curvature direction)
  - Angular max pooling: apply a rotating filter

$$\mathbf{x}'_i \propto \max_{\Delta\theta \in [0,2\pi)} \sum_{\ell=1}^{L} g_\ell \sum_{j=1}^{n} w_\ell(\rho_{ij}, \theta_{ij} + \Delta\theta)\mathbf{x}_j$$

Masci et al. 2015; Monti et al. 2017

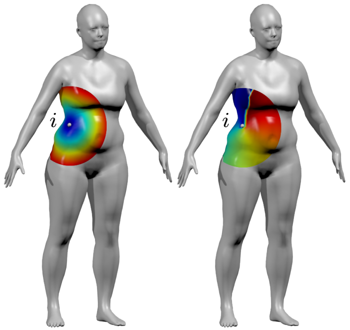
Geodesic polar coordinates

- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

ρ_{ij} = geodesic distance from i to j

θ_{ij} = direction of geodesic from i to j



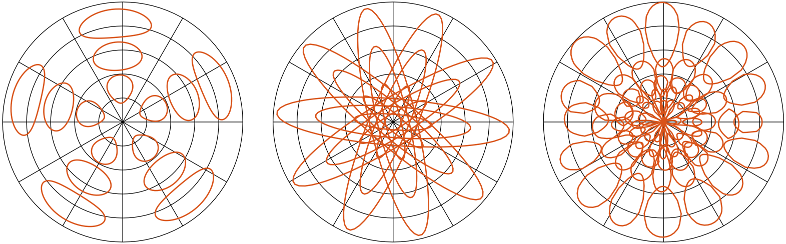
- Orientation ambiguity!
  - Canionical direction (e.g. intrinsic vector field, max curvature direction)
  - Angular max pooling: apply a rotating filter

$$\mathbf{x}'_i \propto \max_{\Delta\theta \in [0, 2\pi)} \sum_{\ell=1}^{L} g_\ell \sum_{j=1}^{n} w_\ell(\rho_{ij}, \theta_{ij} + \Delta\theta)\mathbf{x}_j$$

  - Fourier transform magnitute w.r.t. θ

Masci et al. 2015; Monti et al. 2017

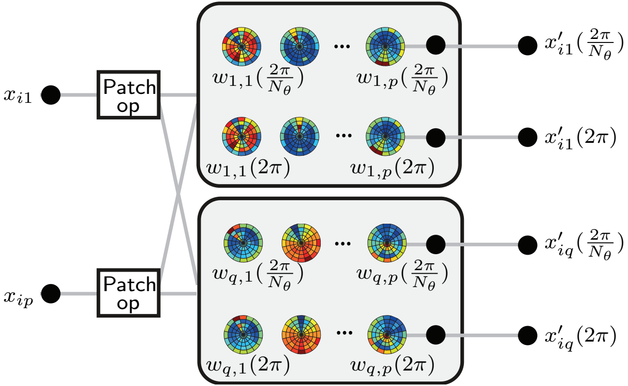
Patch operator weight functions



GCNN ACNN MoNet

Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2017 (MoNet)

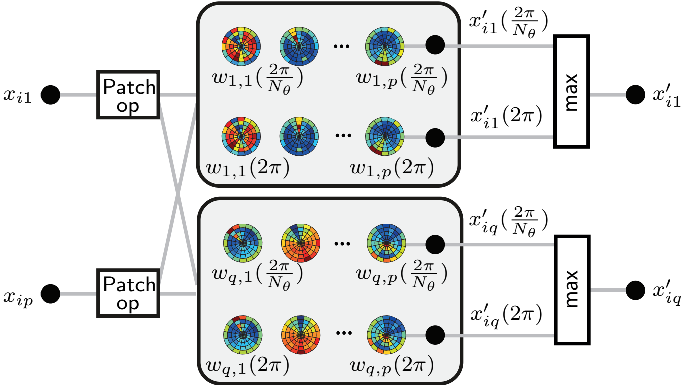
Geodesic convolution layer



Conv. layer
$$x'_{il}(\Delta\theta) \propto \sum_{l'=1}^{p} \underbrace{\sum_{\ell=1}^{L} g_\ell \sum_{j=1}^{n} w_{l,l',\ell}(\rho_{ij}, \theta_{ij} + \Delta\theta) x_{jl'}}_{\text{Conv. with filter rotated by } \Delta\theta}$$

Masci et al. 2015

Geodesic convolution layer

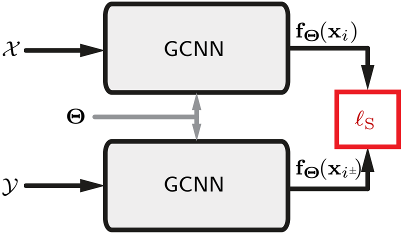


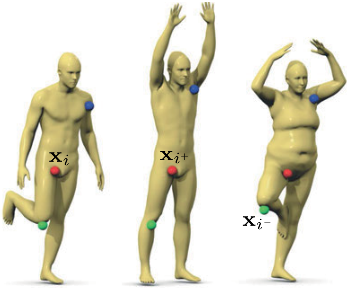
Conv. layer $$x'_{il}(\Delta\theta) \propto \sum_{l'=1}^{p} \underbrace{\sum_{\ell=1}^{L} g_\ell \sum_{j=1}^{n} w_{l,l',\ell}(\rho_{ij}, \theta_{ij} + \Delta\theta) x_{jl'}}_{\text{Conv. with filter rotated by } \Delta\theta}$$

Angular max pooling $$x'_{il} = \max_{\Delta\theta} x'_{il}(\Delta\theta)$$

Masci et al. 2015

Learning local descriptors with GCNN





Training set	positive (i, i^+) and negative (i, i^-) pairs of points
Siamese net	two net instances with shared parameters $\mathbf{\Theta}$
Poitwise feature cost	$\ell_{\mathrm{S}}(\mathbf{\Theta}) = \gamma \sum_{i,i^+} \lVert \mathbf{f}_{\mathbf{\Theta}}(\mathbf{x}_i) - \mathbf{f}_{\mathbf{\Theta}}(\mathbf{x}_{i^+}) \rVert_2^2 + (1-\gamma) \sum_{i,i^-} \left[\mu - \lVert \mathbf{f}_{\mathbf{\Theta}}(\mathbf{x}_i) - \mathbf{f}_{\mathbf{\Theta}}(\mathbf{x}_{i^-}) \rVert_2^2\right]_+$

Masci et al. 2015

HKS descriptor

Distance in the space of local Heat Kernel Signature (HKS) features
(shown is distance from a point on the shoulder marked in white)

Descriptor: Sun, Ovsjanikov, Guibas 2009 (HKS); data: B et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

WKS descriptor

Distance in the space of local Wave Kernel Signature (WKS) features (shown is distance from a point on the shoulder marked in white)

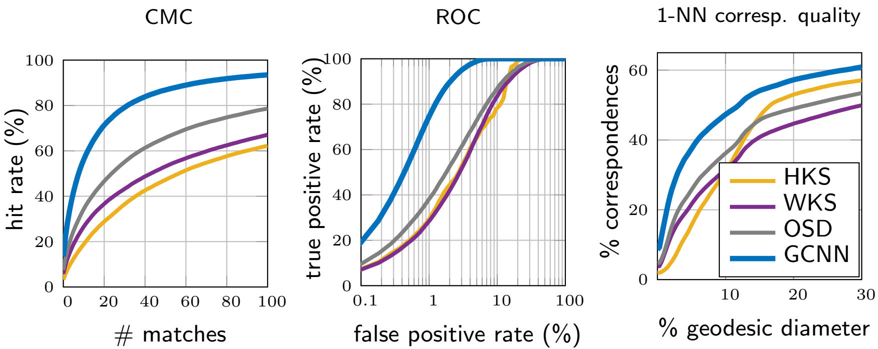
Descriptor: Aubry, Schlickewei, Cremers 2011 (WKS); data: B et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

Descriptor learned with GCNN

Distance in the space of local GCNN features
(shown is distance from a point on the shoulder marked in white)

Descriptor: Masci et al. 2015 (GCNN); data: B et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

Descriptor quality comparison



Descriptor performance using symmetric Princeton benchmark
(training and testing: disjoint subsets of FAUST)

Methods: Sun et al. 2009 (HKS); Aubry et al. 2011 (WKS); Litman, B 2014 (OSD); Masci et al. 2015 (GCNN); data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011

Homogeneous diffusion

$$f_t(x) = -\mathrm{div}(c\nabla f(x))$$

$c =$ thermal diffusivity constant describing heat conduction properties of the material (diffusion speed is equal everywhere)

Anisotropic diffusion

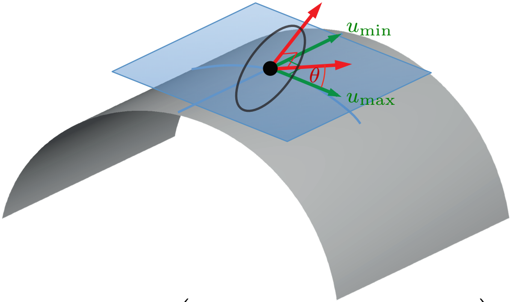
$$f_t(x) = -\mathrm{div}(\mathbf{A}(x)\nabla f(x))$$

$\mathbf{A}(x) =$ heat conductivity tensor describing heat conduction properties of the material (diffusion speed is position $+$ direction dependent)

Anisotropic diffusion

Anisotropic

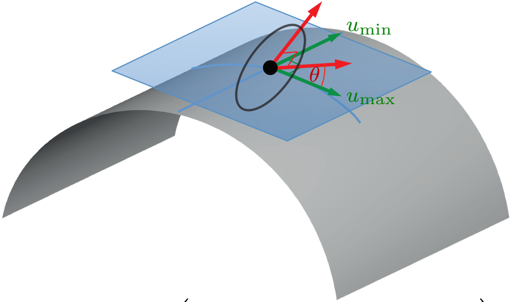
Anisotropic diffusion on manifolds



$$f_t(x) = -\mathrm{div}\left(\mathbf{R}_\theta \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\theta^\top \nabla f(x) \right)$$

Andreux et al. 2014; Boscaini et al. 2016

Anisotropic diffusion on manifolds



$$f_t(x) = -\mathrm{div}\left(\underbrace{\mathbf{R}_\theta \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\theta^\top}_{\mathbf{D}_{\alpha\theta}(x)} \nabla f(x) \right)$$

- Anisotropic Laplacian $\Delta_{\alpha\theta} f(x) = \mathrm{div}\left(D_{\alpha\theta}(x) \nabla f(x)\right)$
- $\theta =$ orientation w.r.t. max curvature direction
- $\alpha =$ 'elongation'

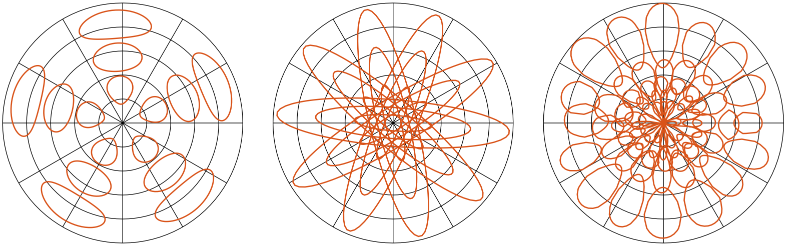
Andreux et al. 2014; Boscaini et al. 2016

Anisotropic heat kernels

$$h_{\alpha\theta t}(x, x') = \sum_{k \geq 0} e^{-t\lambda_{\alpha\theta k}} \phi_{\alpha\theta k}(x) \phi_{\alpha\theta k}(x')$$

Orientation θ Elongation α

Patch operator weight functions



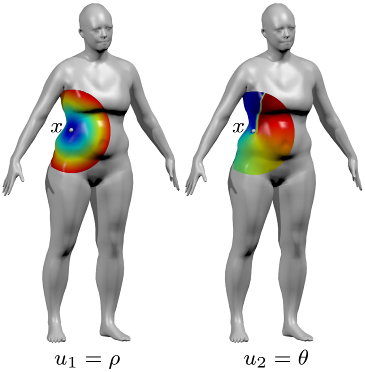
GCNN ACNN **MoNet**

Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2016 (MoNet)

Learnable patches on manifolds

- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$



Monti et al. 2016

Learnable patches on manifolds

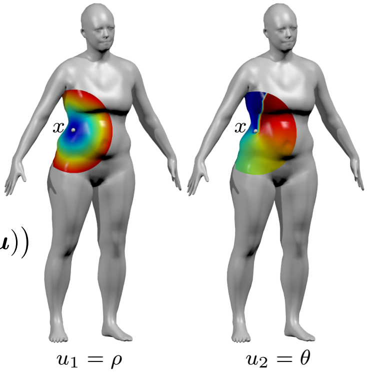
- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\left(-\tfrac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\right)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



Monti et al. 2016

Learnable patches on manifolds

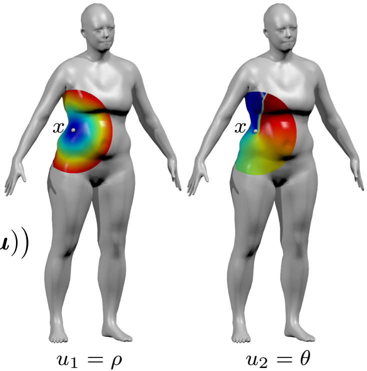
- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\left(-\tfrac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\right)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



- Spatial convolution

$$\mathbf{x}'_i \propto \sum_{\ell=1}^{L} g_\ell \sum_{j=1}^{n} w_{\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell}(\mathbf{u}_{ij})\,\mathbf{x}_j$$

Monti et al. 2016

Learnable patches on manifolds

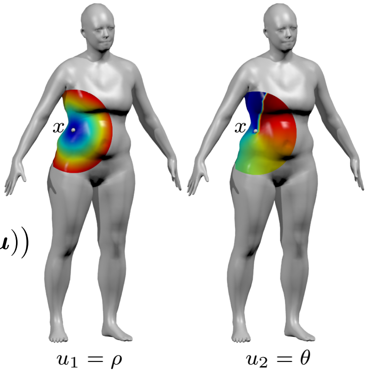
- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\left(-\tfrac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\right)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



- Spatial convolution

$$\mathbf{x}'_i \propto \sum_{j=1}^{n} \sum_{\ell=1}^{L} g_\ell w_{\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell}(\mathbf{u}_{ij})\, \mathbf{x}_j$$

Monti et al. 2016

Learnable patches on manifolds (MoNet)

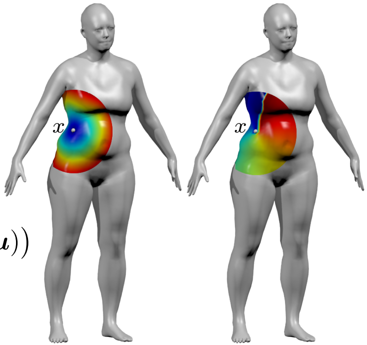
- Geodesic polar coordinates

$$\mathbf{u}_{ij} = (\rho_{ij}, \theta_{ij})$$

- Gaussian weighting functions

$$w_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = \exp\left(-\tfrac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\right)$$

with learnable covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$



$u_1 = \rho$ $\quad$ $u_2 = \theta$

- Spatial convolution

$$\mathbf{x}_i' \propto \sum_{j=1}^{n} \underbrace{\sum_{\ell=1}^{L} g_\ell w_{\boldsymbol{\mu}_\ell,\boldsymbol{\Sigma}_\ell}(\mathbf{u}_{ij})}_{\text{Gaussian mixture}} \mathbf{x}_j$$

Monti et al. 2016

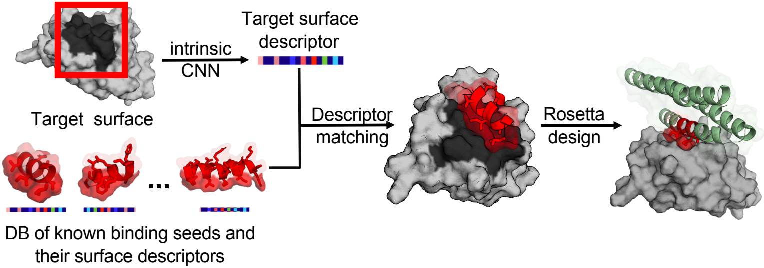
MoNet as generalization of previous methods

Method	Coordinates $\mathbf{u}_{ij}$	Weight function $w_{\mathbf{\Theta}}(\mathbf{u})$
CNN[1]	$\mathbf{u}_j - \mathbf{u}_i$	$\delta(\mathbf{u}-\mathbf{v})$
GCNN[2]	ρ_{ij}, θ_{ij}	$\exp\left(-\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top \begin{pmatrix} \sigma_\rho^2 & \\ & \sigma_\theta^2 \end{pmatrix}^{-1} (\mathbf{u}-\mathbf{v})\right)$
ACNN[3]	ρ_{ij}, θ_{ij}	$\exp\left(-t\mathbf{u}^\top \mathbf{R}_\varphi \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\varphi^\top \mathbf{u}\right)$
MoNet[4]	ρ_{ij}, θ_{ij}	$\exp\left(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\right)$
		learnable parameters $\mathbf{\Theta} = (\boldsymbol{\mu}, \mathbf{\Sigma})$

Some CNN models can be considered as particular settings of MoNet with weighting functions of different form

Methods: [1]LeCun et al. 1998; [2]Masci et al. 2015; [3]Boscaini et al. 2016; [4]Monti et al. 2016

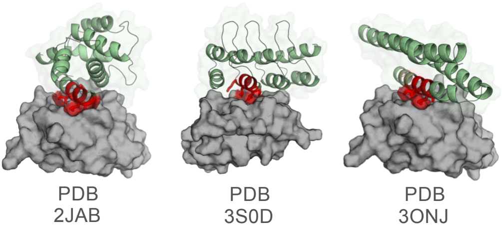
Application of MoNet: Protein-Protein Interaction

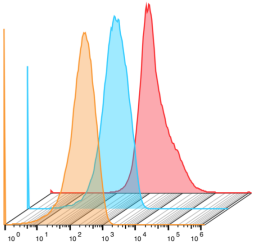


Designing protein binder for the PD-L1 protein target

Collaboration with B. Correia and P. Gainza-Cirauqui (EPFL)

Application of MoNet: Protein-Protein Interaction



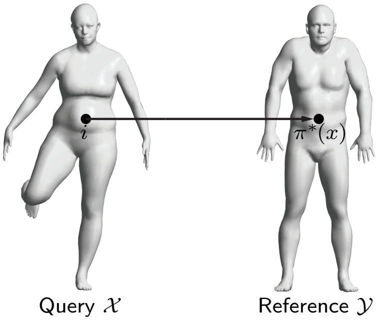


Experimentally confirmed computational design of PD-L1 binders

Collaboration with B. Correia and P. Gainza-Cirauqui (EPFL)

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$
- Correspondence = label each query vertex $i \in \{1, \ldots, n\}$ as reference vertex $\pi_i \in \{1, \ldots, m\}$



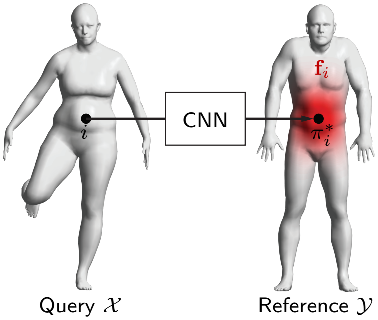
Rodolà et al. 2014; Masci et al. 2015

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$
- Correspondence = label each query vertex $i \in \{1, \ldots, n\}$ as reference vertex $\pi_i \in \{1, \ldots, m\}$
- Net output at i after softmax layer

$$\mathbf{f}_{\Theta}(\mathbf{x}_i) = (f_{i1}, \ldots, f_{im})$$

$=$ probability distribution on $\mathcal{Y}$



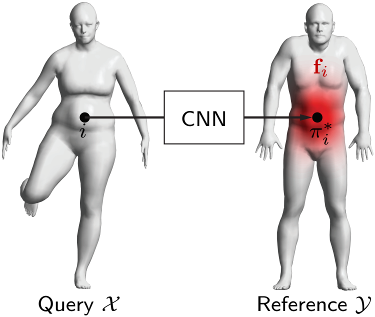
Rodolà et al. 2014; Masci et al. 2015

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$
- Correspondence = label each query vertex $i \in \{1, \ldots, n\}$ as reference vertex $\pi_i \in \{1, \ldots, m\}$
- Net output at i after softmax layer

$$\mathbf{f}_{\mathbf{\Theta}}(\mathbf{x}_i) = (f_{i1}, \ldots, f_{im})$$

$=$ probability distribution on $\mathcal{Y}$

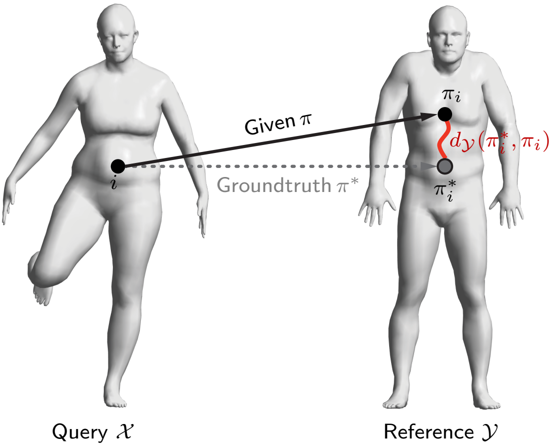


Minimize on training set the cross entropy between groundtruth correspondence and output probability distribution w.r.t. net parameters $\mathbf{\Theta}$

$$\min_{\mathbf{\Theta}} \sum_{i=1}^{n} H(\boldsymbol{\delta}_{\pi_i^*}, \mathbf{f}_{\mathbf{\Theta}}(\mathbf{x}_i))$$

Rodolà et al. 2014; Masci et al. 2015

Correspondence evaluation: Princeton benchmark

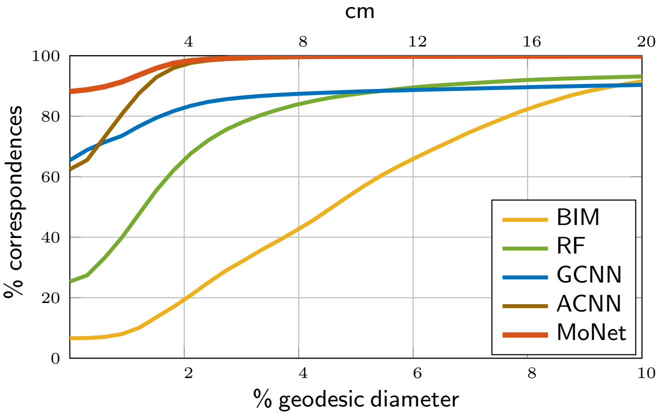


Pointwise correspondence error = geodesic distance from the groundtruth

$$\epsilon_i = d_{\mathcal{Y}}(\pi_i^*, \pi_i)$$

Kim et al. 2011

Correspondence quality comparison



Correspondence evaluated using asymmetric Princeton benchmark
(training and testing: disjoint subsets of FAUST)

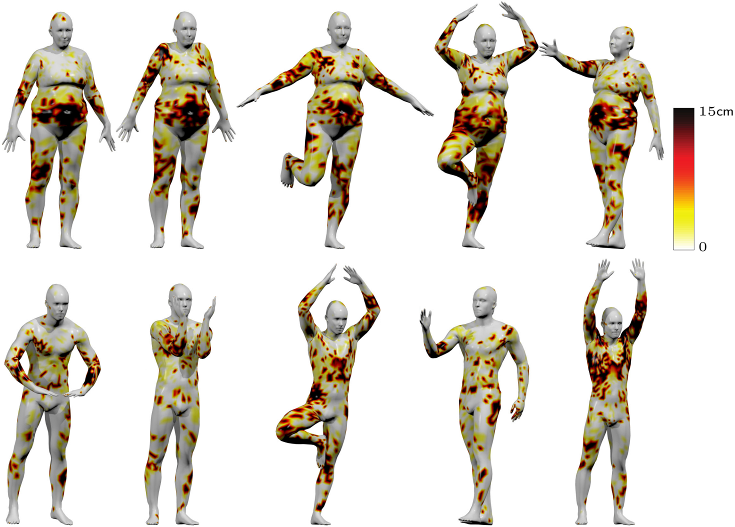
Methods: Kim et al. 2011 (BIM); Rodolà et al. 2014 (RF); Boscaini et al. 2015 (ADD); Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2016 (MoNet); data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011

Shape correspondence error: Blended Intrinsic Map

Pointwise correspondence error (geodesic distance from groundtruth)

Kim, Lipman, Funkhouser 2011

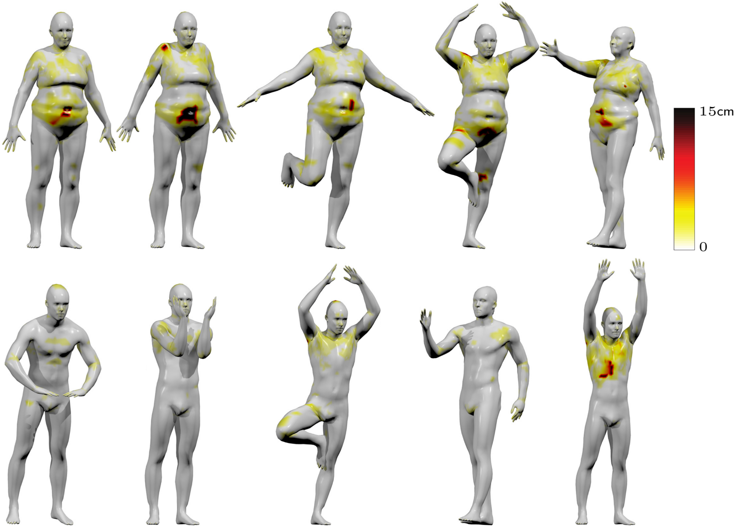
Shape correspondence error: Geodesic CNN



Pointwise correspondence error (geodesic distance from groundtruth)

Masci et al. 2015

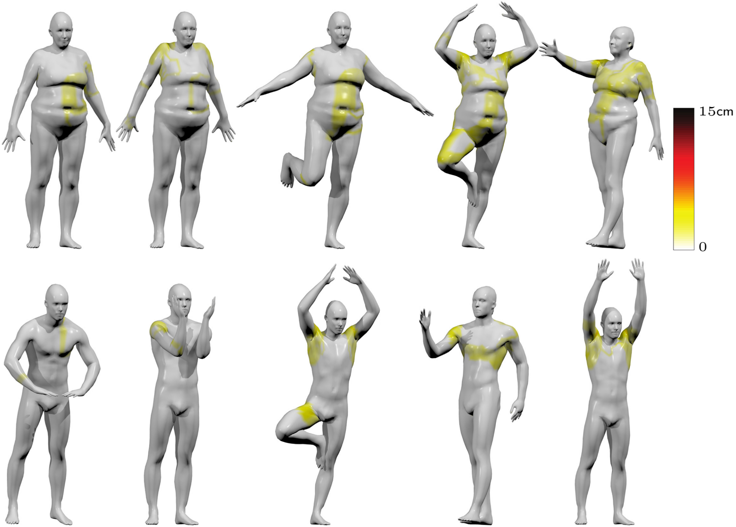
Shape correspondence error: Anisotropic CNN



Pointwise correspondence error (geodesic distance from groundtruth)

Boscaini et al. 2016

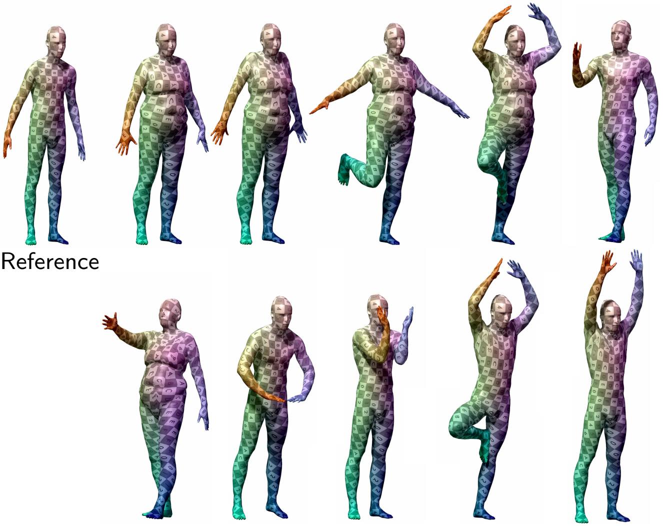
Shape correspondence error: MoNet



Pointwise correspondence error (geodesic distance from groundtruth)

Monti et al. 2016

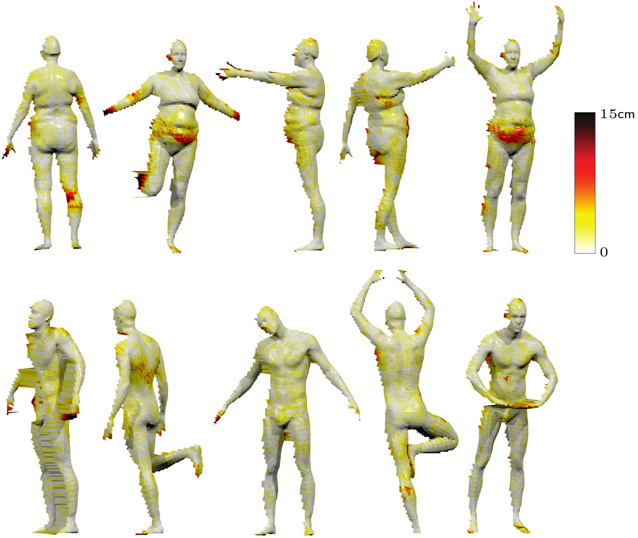
Shape correspondence visualization: MoNet



Texture transferred from reference to query shapes

Monti et al. 2016

Correspondence on range images: MoNet



Pointwise correspondence error (geodesic distance from groundtruth)

Monti et al. 2016

Correspondence with MoNet: Range images

Reference

Correspondence visualization (similar colors encode corresponding points)

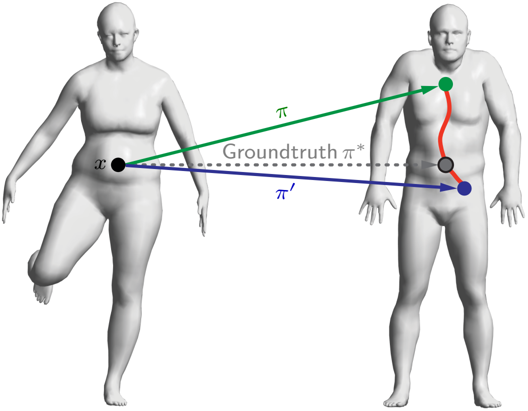
Monti et al. 2016

Correspondence with MoNet: Range images

Correspondence visualization (similar colors encode corresponding points)

Monti et al. 2016

Correspondence as classification problem, revisited

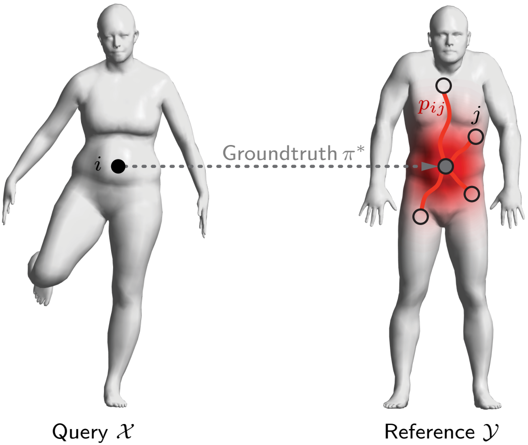


Query $\mathcal{X}$ Reference $\mathcal{Y}$

Classification cost considers equally correspondences that deviate from the groundtruth (no matter how far)

Kim et al. 2011

Soft correspondence error

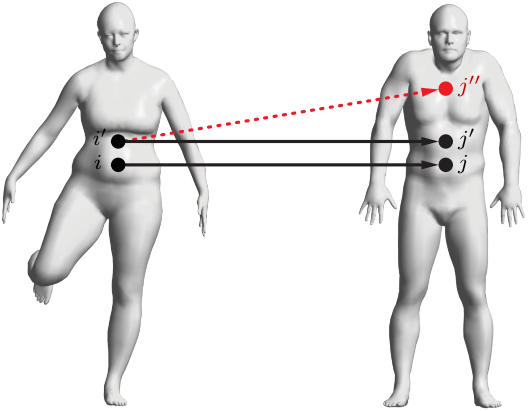


Soft correspondence error = probability-weighted geodesic distance from the groundtruth

$$\bar{\epsilon}_i = \sum_{j=1}^{m} p_{ij} d_{\mathcal{Y}}(\pi_i^*, j)$$

Kovnatsky et al. 2015; Litany et al. 2017

Pointwise vs Structured learning

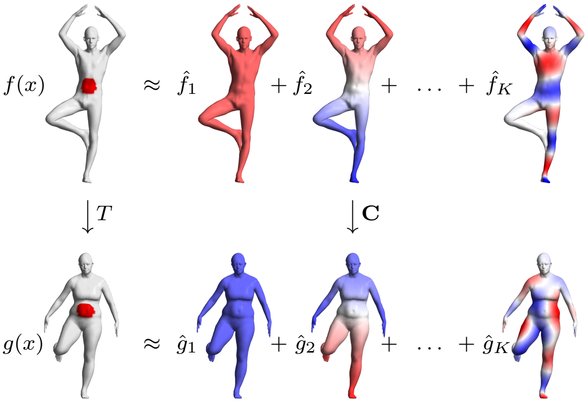


Query $\mathcal{X}$ Reference $\mathcal{Y}$

Nearby points i, i' on query shape are **not guaranteed** to map to nearby points j, j' on reference shape at **test time**

Litany et al. 2017

Functional maps: spectral domain

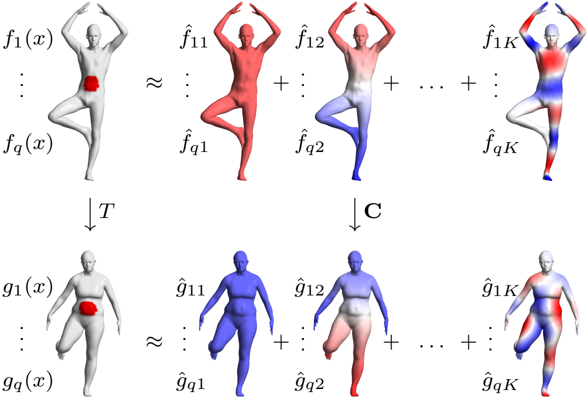


Functional correspondence T = linear map $\mathbf{C}$ between Fourier coefficients

$$\hat{\mathbf{g}}^\top = \hat{\mathbf{f}}^\top \mathbf{C}$$

Ovsjanikov et al. 2012

Functional maps: spectral domain

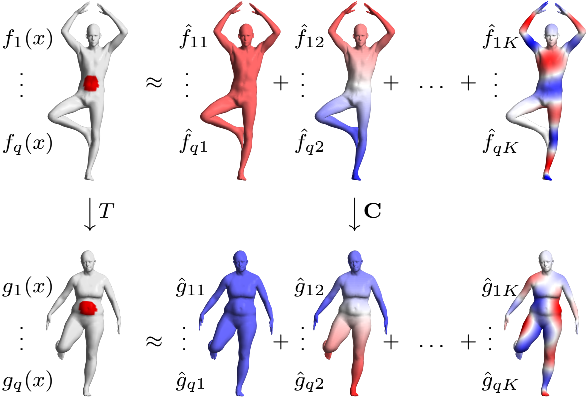


Recover correspondence from $q \geq k$ dimensional pointwise features

$$\begin{pmatrix} \hat{g}_{11} & \hat{g}_{12} & \dots & \hat{g}_{1K} \\ \vdots & \vdots & & \vdots \\ \hat{g}_{q1} & \hat{g}_{q2} & \dots & \hat{g}_{qK} \end{pmatrix} = \begin{pmatrix} \hat{f}_{11} & \hat{f}_{12} & \dots & \hat{f}_{1K} \\ \vdots & \vdots & & \vdots \\ \hat{f}_{q1} & \hat{f}_{q2} & \dots & \hat{f}_{qK} \end{pmatrix} \mathbf{C}$$

Ovsjanikov et al. 2012

Functional maps: spectral domain

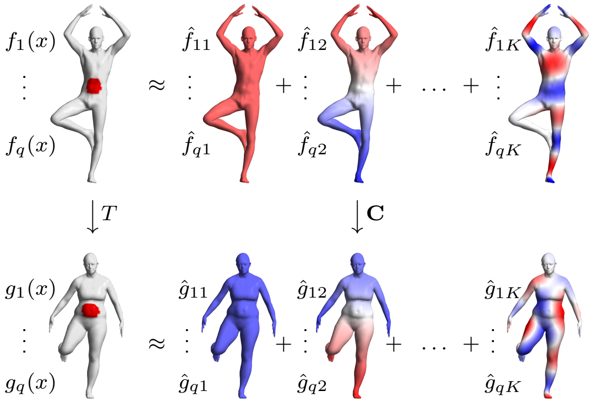


Recover correspondence from $q \geq k$ dimensional pointwise features

$$\hat{\mathbf{G}} = \hat{\mathbf{F}}\mathbf{C}$$

Ovsjanikov et al. 2012

Functional maps: spectral domain

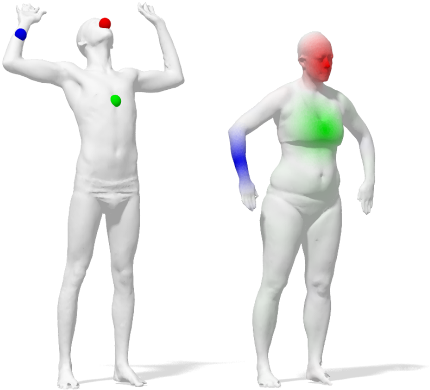


Recover correspondence from $q \geq k$ dimensional pointwise features

$$\mathbf{C}^* = \underset{\mathbf{C}}{\operatorname{argmin}} \|\hat{\mathbf{F}}\mathbf{C} - \hat{\mathbf{G}}\|_{\mathrm{F}}^2$$

Ovsjanikov et al. 2012

Functional maps: spatial domain

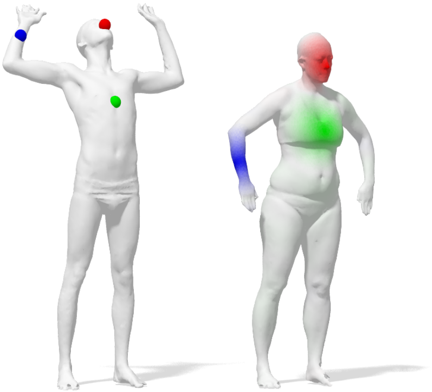


Rank-K approximation of spatial correspondence

$$\mathbf{T} \approx \mathbf{\Psi}\mathbf{C}\mathbf{\Phi}^\top$$

Ovsjanikov et al. 2012

Functional maps: spatial domain

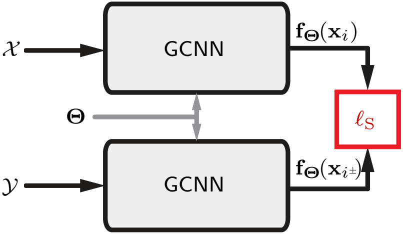


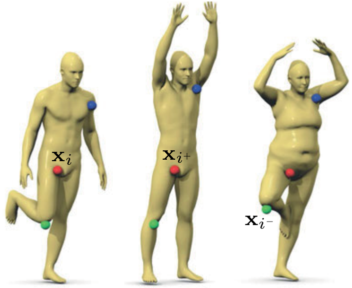
Probability p_{ij} of point j mapping to i

$$\mathbf{P} \approx |\mathbf{\Psi C \Phi}^\top|_{\|\cdot\|}$$

Ovsjanikov et al. 2012

Siamese metric learning





Training set — positive (i, i^+) and negative (i, i^-) pairs of points

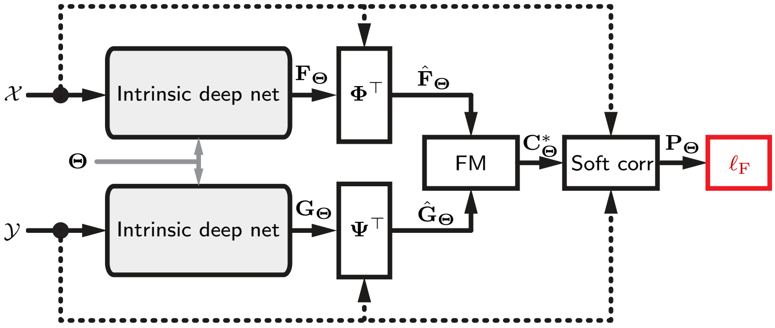
Siamese net — two net instances with shared parameters $\boldsymbol{\Theta}$

Poitwise feature cost

$$\ell_{\mathrm{S}}(\boldsymbol{\Theta}) = \gamma \sum_{i,i^+} \|\mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{x}_i) - \mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{x}_{i^+})\|_2^2 + (1-\gamma) \sum_{i,i^-} \left[\mu - \|\mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{x}_i) - \mathbf{f}_{\boldsymbol{\Theta}}(\mathbf{x}_{i^-})\|_2^2\right]_+$$

Masci et al. 2015

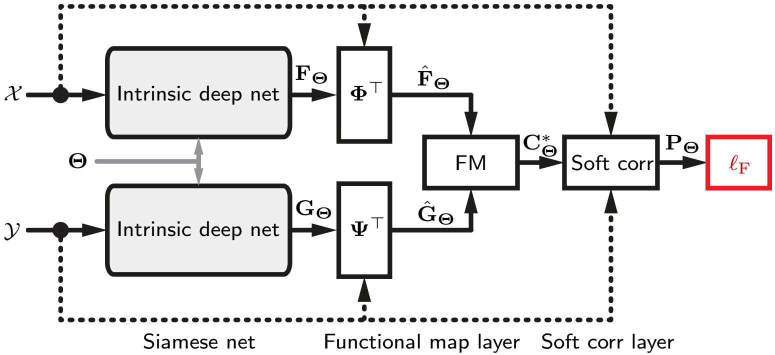
Structured correspondence with FMNet



Siamese net	two net instances with shared parameters $\boldsymbol{\Theta}$
Functional map layer	$\mathbf{C}^*_{\boldsymbol{\Theta}} = \underset{\mathbf{C}}{\operatorname{argmin}} \|\hat{\mathbf{F}}_{\boldsymbol{\Theta}}\mathbf{C} - \hat{\mathbf{G}}_{\boldsymbol{\Theta}}\|^2_{\mathrm{F}}$
Soft correspondence layer	$\mathbf{P}_{\boldsymbol{\Theta}} = \lvert \boldsymbol{\Psi}\mathbf{C}_{\boldsymbol{\Theta}}\boldsymbol{\Phi}^\top \rvert_{\|\cdot\|}$
Soft error cost	$\ell_{\mathrm{F}}(\boldsymbol{\Theta}) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{\boldsymbol{\Theta},ij}\, d_{\mathcal{Y}}(\pi^*_i, j)$

Litany et al. 2017

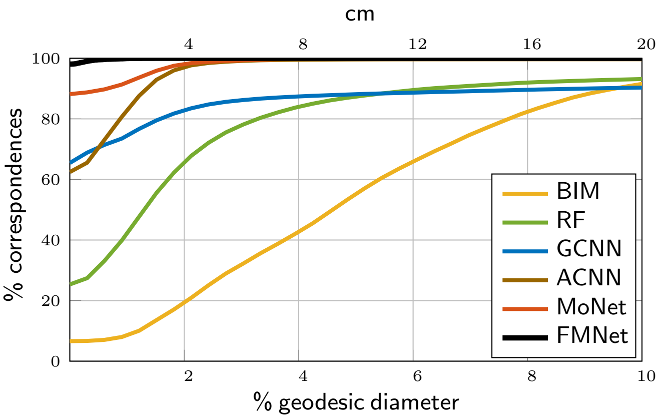
Structured correspondence with FMNet



Siamese net	two net instances with shared parameters $\mathbf{\Theta}$
Functional map layer	$\mathbf{C}^*_\Theta = \hat{\mathbf{F}}^\dagger_\Theta \hat{\mathbf{G}}_\Theta$
Soft correspondence layer	$\mathbf{P}_\Theta = \lvert \mathbf{\Psi}\mathbf{C}_\Theta\mathbf{\Phi}^\top \rvert_{\lVert\cdot\rVert}$
Soft error cost	$\ell_{\mathrm{F}}(\mathbf{\Theta}) = \lVert \mathbf{P}_\Theta \circ \mathbf{D}_\mathcal{Y} \rVert$

Litany et al. 2017

Correspondence quality comparison

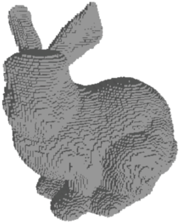


Correspondence evaluated using asymmetric Princeton benchmark
(training and testing: disjoint subsets of FAUST)

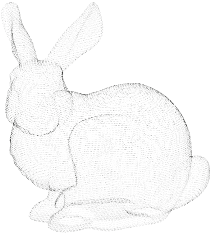
Methods: Kim et al. 2011 (BIM); Rodolà et al. 2014 (RF); Boscaini et al. 2015 (ADD); Masci et al. 2015 (GCNN); Boscaini et al. 2016 (ACNN); Monti et al. 2016 (MoNet); Litany et al. 2017 (FMNet); data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011

Shape representation

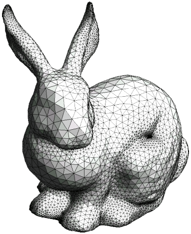
Image-based



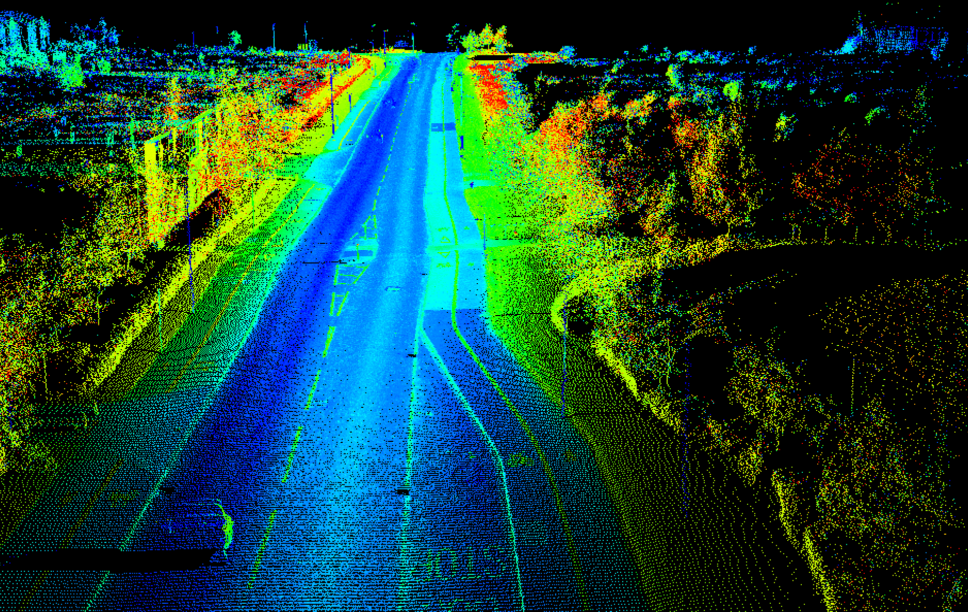
Volumetric



Point-based



Surface-based



PointNet: learning on sets

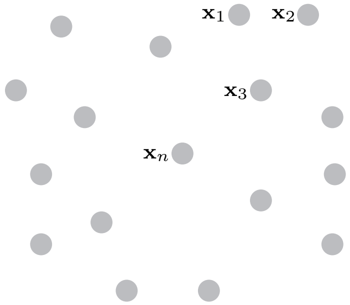
(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets

- Permutation-invariant function

  $$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i



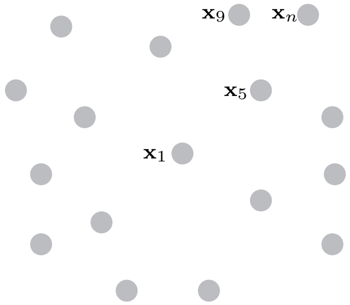
(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets

- Permutation-invariant function

  $$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i



(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

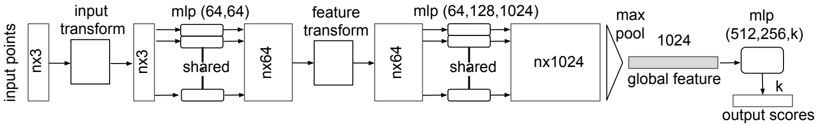
PointNet: learning on sets

- Permutation-invariant function

  $$f(\mathbf{x}_1,\ldots,\mathbf{x}_n) = f(\mathbf{x}_{\pi_1},\ldots,\mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

- Shared function $h_{\boldsymbol{\Theta}}(\cdot)$ applied to each point + permutation-invariant aggregation ($\max$ or $\sum$)



(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets

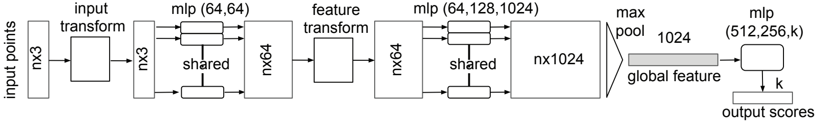
- Permutation-invariant function

  $$f(\mathbf{x}_1,\ldots,\mathbf{x}_n) = f(\mathbf{x}_{\pi_1},\ldots,\mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

- Shared function $h_{\mathbf{\Theta}}(\cdot)$ applied to each point + permutation-invariant aggregation ($\max$ or $\sum$)

- Spatial transformer units



(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017

PointNet: learning on sets

- Permutation-invariant function

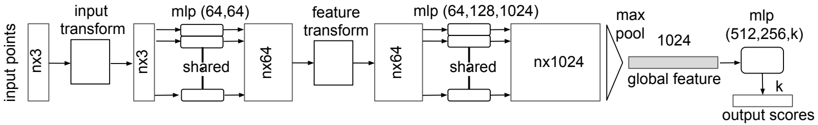
  $$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_n})$$

  where $\mathbf{x}_i \in \mathbb{R}^d$ is feature at vertex i

- Shared function $h_{\boldsymbol{\Theta}}(\cdot)$ applied to each point + permutation-invariant aggregation ($\max$ or $\sum$)

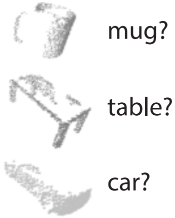
- Spatial transformer units

- Local grouping (PointNet++, PCPNet)



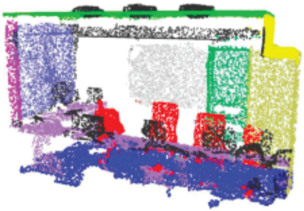
(Vinyals, Bengio, Kudlur 2015); Qi et al. 2017; Qi, Yi et al. 2017; Guerrero et al. 2018

PointNet applications

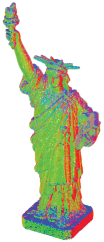


Object recognition

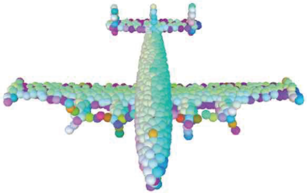
Part segmentation



Semantic segmentation



Curvature

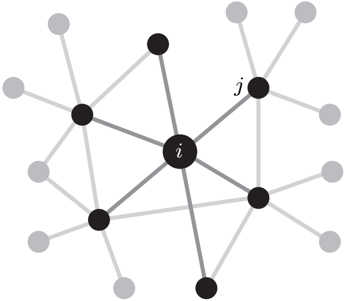


Normals

Point cloud generation

Graph-based edge convolution

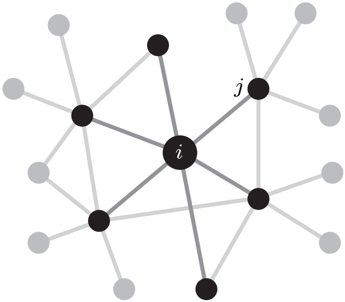
- Local neighborhood structure modeled as a graph



Wang et al. 2018

Graph-based edge convolution

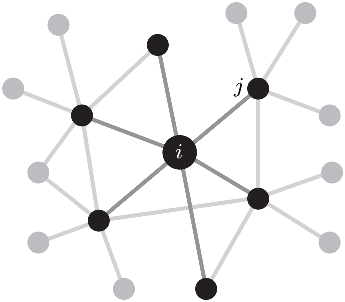
- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\mathbf{\Theta}}(\cdot, \cdot)$ parametrized by $\mathbf{\Theta}$



Wang et al. 2018

Graph-based edge convolution

- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\boldsymbol{\Theta}}(\cdot,\cdot)$ parametrized by $\boldsymbol{\Theta}$
- Permutation-invariant aggregation operator $\square$ (e.g. $\sum$ or $\max$) on the neighborhood of i

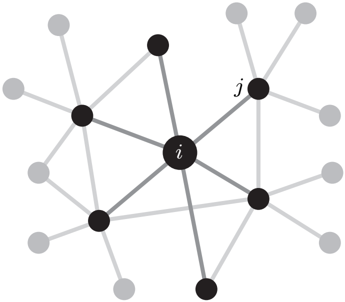


Wang et al. 2018

Graph-based edge convolution

- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\mathbf{\Theta}}(\cdot, \cdot)$ parametrized by $\mathbf{\Theta}$
- Permutation-invariant aggregation operator $\square$ (e.g. $\sum$ or $\max$) on the neighborhood of i
- Edge convolution (EdgeConv)

$$\mathbf{x}_i' = \mathop{\square}_{j} h_{\mathbf{\Theta}}(\mathbf{x}_i, \mathbf{x}_j)$$

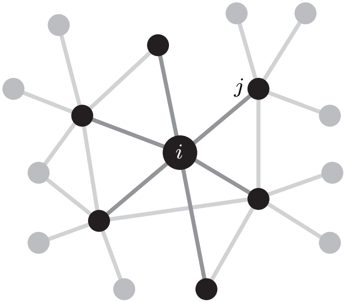


Wang et al. 2018

Graph-based edge convolution

- Local neighborhood structure modeled as a graph
- Edge feature function $h_{\mathbf{\Theta}}(\cdot, \cdot)$ parametrized by $\mathbf{\Theta}$
- Permutation-invariant aggregation operator $\square$ (e.g. $\sum$ or $\max$) on the neighborhood of i
- Edge convolution (EdgeConv)

$$\mathbf{x}'_i = \underset{j}{\square} h_{\mathbf{\Theta}}(\mathbf{x}_i, \mathbf{x}_j)$$



Learnable local (nonlinear) operator

Wang et al. 2018

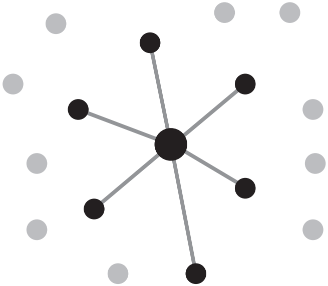
Particular cases

Method	Aggregation $\square$	Edge feature $h(\mathbf{x}_i, \mathbf{x}_j)$
Laplacian	$\sum$	$w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$
PointNet[1]	–	$h(\mathbf{x}_i)$
PointNet++[2]	$\max$	$h(\mathbf{x}_i)$
MoNet[3]	$\sum$	$\sum_\ell g_\ell w_\ell(\mathbf{u}_{ij})\mathbf{x}_j$
PCNN[4]	$\sum$	$\sum_{\ell m} c(\mathbf{x}_i \cdot \mathbf{k}_{\ell m}) w_i q_{\mathbf{\Theta}_\ell}(\mathbf{x}_i, \mathbf{x}_j)$

Wang et al. 2018; [1]Qi et al. 2017; [2]Qi, Su et al. 2017; [3]Monti et al. 2017; [4]Atzmon et al. 2018

Dynamic Graph CNN (DynGCNN)

Construct k-NN graph in feature space and update it after each layer

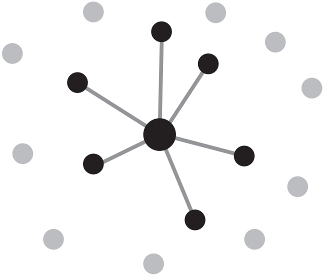


Layer l

Features $\mathbf{x}_1^{(l)}, \ldots, \mathbf{x}_n^{(l)} \in \mathbb{R}^{d_l}$

k-NN graph $\mathcal{G}^{(l)}$

$h^{(l)} : \mathbb{R}^{d_l} \times \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$



Layer $l+1$

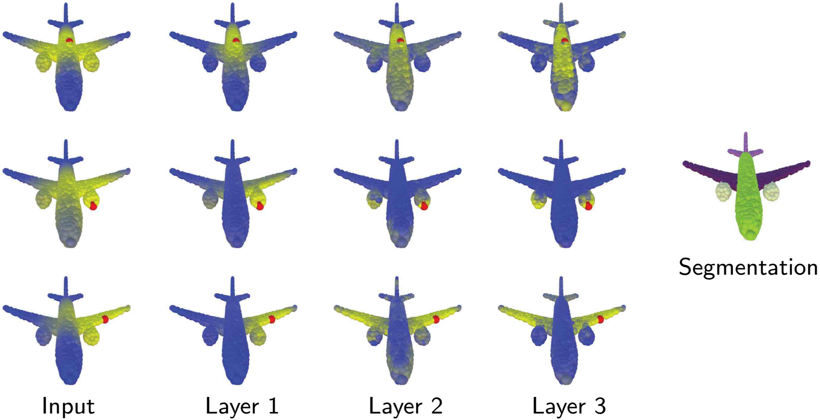
Features $\mathbf{x}_1^{(l+1)}, \ldots, \mathbf{x}_n^{(l+1)} \in \mathbb{R}^{d_{l+1}}$

k-NN graph $\mathcal{G}^{(l+1)}$

$h^{(l+1)} : \mathbb{R}^{d_{l+1}} \times \mathbb{R}^{d_{l+1}} \to \mathbb{R}^{d_{l+2}}$

Wang et al. 2018

Learning semantic features



Left: Distance from red point in the feature space of different DynGCNN layers
Right: semantic segmentation results

Wang et al. 2018

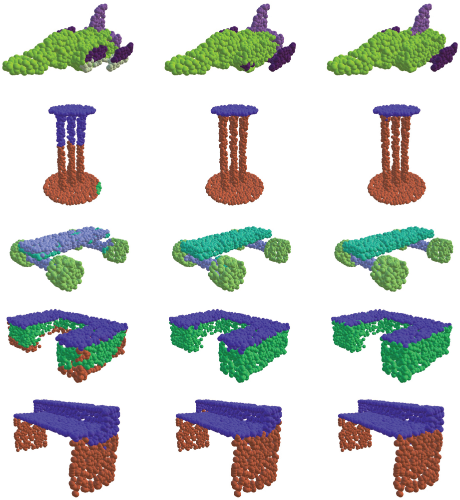
Shape classification (ModelNet40)

Method	Mean class accuracy	Overall accuracy
3DShapeNet[1]	77.3%	84.7%
VoxNet[2]	83.0%	85.9%
Subvolume[3]	86.0%	89.2%
ECC[4]	83.2%	87.4%
PointNet[5]	86.0%	89.2%
PointNet++[6]	–	90.7%
Kd-Net[7]	–	91.8%
DynGCNN (baseline) [8]	88.8%	91.2%
DynGCNN[8]	**90.2%**	**92.2%**

Classification accuracy of different methods on ModelNet40

Methods: [1]Wu et al. 2015; [2]Maturana et al. 2015; Qi et al. 2016; [4]Simonovsky, Komodakis 2017; [5]Qi et al. 2017; [6]Qi, Su et al. 2017; [7]Klokov, Lempitsky 2017; [8]Wang et al. 2018; data: Wu et al. 2015 (ModelNet)

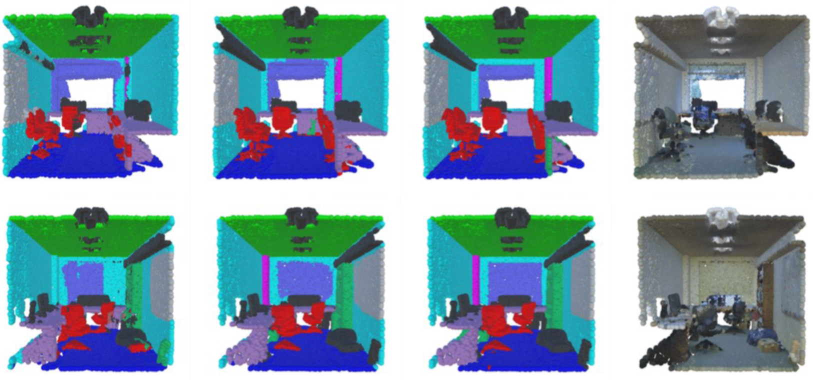
Semantic segmentation: synthetic (ShapeNet)



PointNet **DynGCNN** Groundtruth

Methods: Qi et al. 2017 (PointNet); Wang et al. 2018 (DynGCNN); data: Yi et al. 2016 (ShapeNet)

Semantic segmentation: indoor scans (S3DIS)



PointNet | **DynGCNN** | Groundtruth | Scene

Results of semantic segmentation of point cloud+RGB data
using different architectures

Methods: Qi et al. 2017 (PointNet); Wang et al. 2018 (DynGCNN); data: Armeni et al. 2016 (S3DIS)

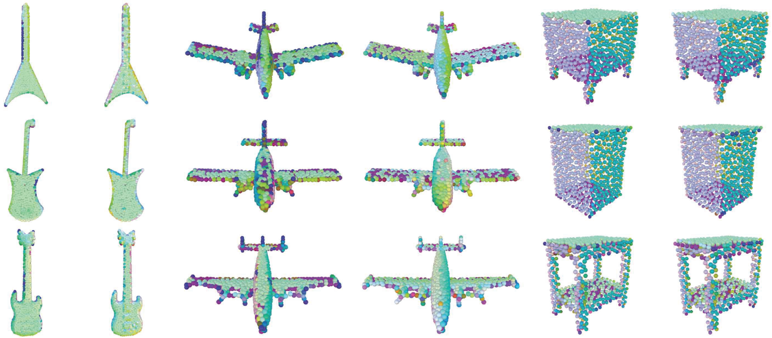
Shape segmentation: indoor scans (S3DIS)

Method	Mean IoU	Overall accuracy
PointNet (Baseline)[1]	20.1%	53.2%
PointNet[1]	47.6%	78.5%
MS + CU(2)[2]	47.8%	79.2%
G + RCU[2]	49.7%	81.1%
DynGCNN[3]	**56.1%**	**84.1%**

S3DIS indoor scene semantic segmentation accuracy

Methods: [1]Qi et al. 2017; [2]Engelmann et al. 2017 [3]Wang et al. 2018; data: Armeni et al. 2016 (S3DIS)

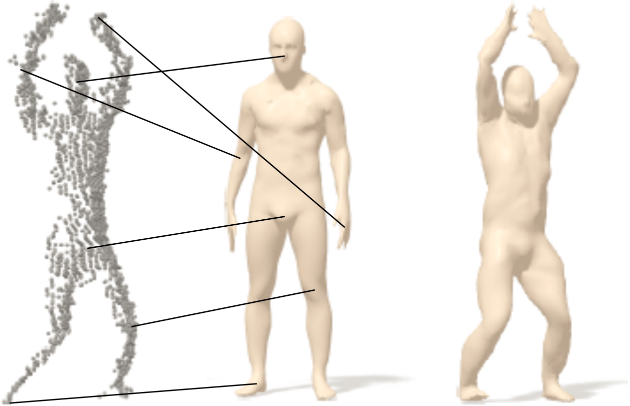
Surface normal prediction



Surface normal predicted using DynGCNN (odd columns) and groundtruth (even columns). Normal direction is color-coded

Wang et al. 2018; data: Wu et al. 2015 (ModelNet)

3D shape analysis and synthesis



Input
3D scan

Correspondence to
Reference shape

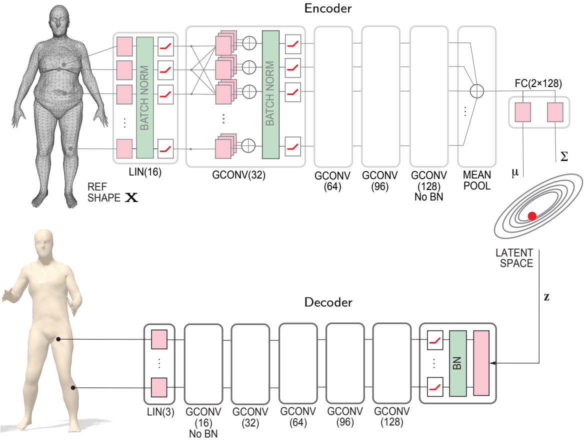
Deformation of
reference shape

Intrinsic Variational Autoencoder (VAE)

Minimize

$$||\mathrm{Dec}(\mathrm{Enc}(\mathbf{X})) - \mathbf{X}||_{\mathrm{F}}$$
$$+\lambda D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{X})||p(\mathbf{z}))$$

w.r.t. net parameters



$\mathrm{Dec}(\mathrm{Enc}(\mathbf{X}))$

Litany et al. 2017; training on Dynamic FAUST (Bogo et al. 2017)

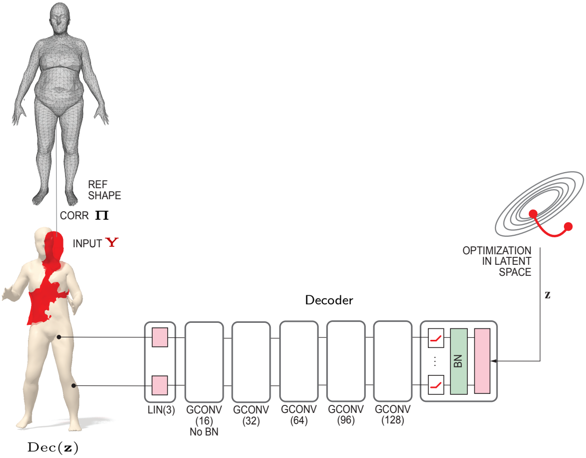
Shape completion

Minimize

$$\|\mathrm{Dec}(\mathbf{z})\mathbf{\Pi} - \mathbf{T}\mathbf{Y}\|_F$$

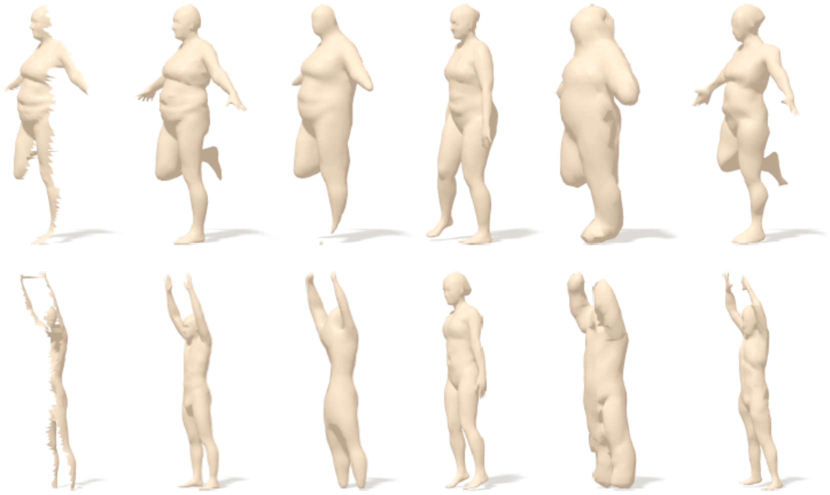
in alternating manner

w.r.t. $\mathbf{z}$ and $\mathbf{T} \in SO(3)$



Litany et al. 2017

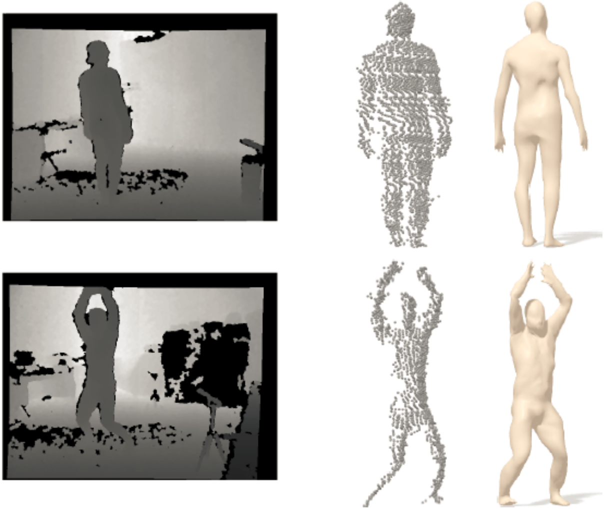
Shape completion comparison



Input Groundtruth Poisson Nearest 3D-EPN **IVAE**

Methods: Litany et al. 2017; Dai et al. 2016 (3D-EPN); Kazhdan et al. 2013 (Poisson)

Shape completion examples

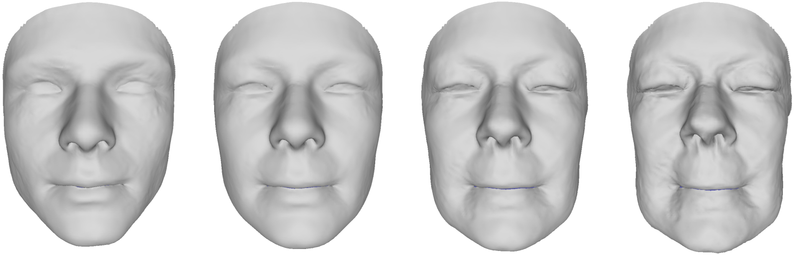


Litany et al. 2017; data: Ofli et al. 2014 (MHAD)

Shape completion examples

Litany et al. 2017; data: Bogo et al. 2014 (FAUST)

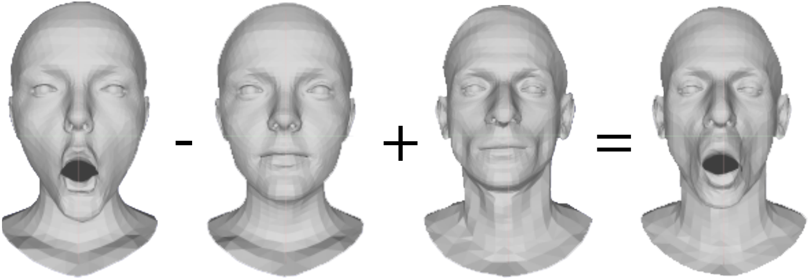
Generative models of faces



X $\quad$ $0.5X + 0.5Y$ $\quad$ Y $\quad$ $-0.5X + 1.5Y$

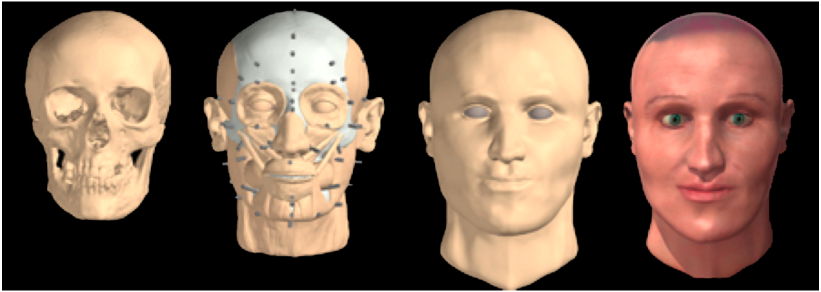
Bouritsas, Bokhnyak, Zafeiriou, B 2019

Generative models of faces



Bouritsas, Bokhnyak, Zafeiriou, B 2019

Face from DNA



Collaboration with P. Claes (KU Leuven)

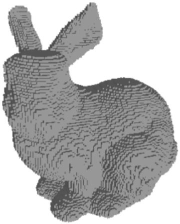
Summary: intrinsic deep learning

- Intrinsic deep learning allows architectures that are deformation invariant by construction
- Vastly less parameters / training data
- Part of a bigger trend of Geometric deep learning on non-Euclidean domains such as graphs
- Several ways of defining intrinsic convolution - each has its own advantages / disadvantages
- Intrinsic shape synthesis (especially with different topology) is a big open question - part of a broader problem of graph generating networks

Summary: deep learning on 3D data

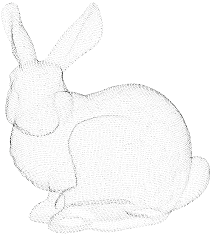
Image-based

- ☺ Simple
- ☺ Efficient*
- ☹ Not geometric
- ☹ No invariance***



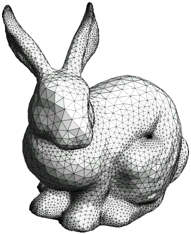
Volumetric

- ☺ Simple
- ☹ High complexity**
- ☹ Coarse
- ☹ No invariance***



Point-based

- ☺ Efficient
- ☺ Simple
- ☺ Accurate
- ☹ No invariance***

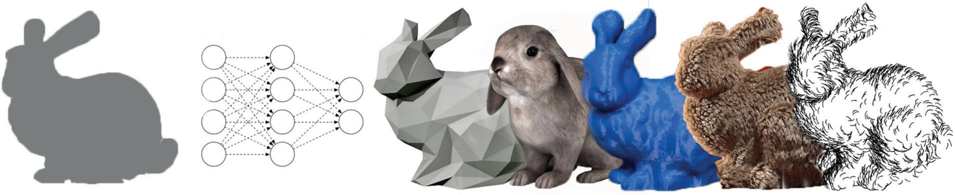


Surface-based

- ☺ Efficient
- ☺ Accurate
- ☺ Deform. invariant
- ☹ Non-standard operations

*Rendering can be slow and memory-heavy. **Can be remedied to some extend by hierarchical data structures. ***No invariance to deformations. Rigid transformations can be remedied to some extent by transformer units.

Course Information (slides/code/comments)



http://geometry.cs.ucl.ac.uk/dl_for_CG/

Scan me