Agenda

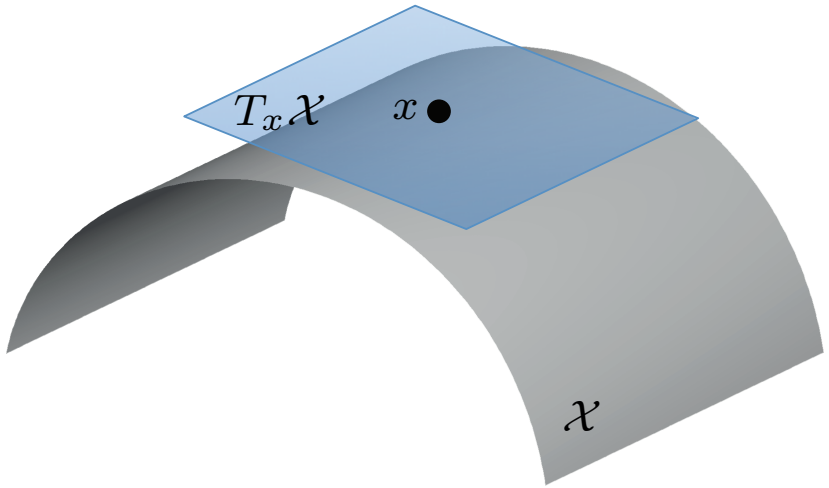
- Deep learning on regular structures
  - Multi-view representation
  - Volumetric Representation
- **Intrinsic deep learning on manifolds**
  - Spectral methods
  - Spatial methods
  - Embedding-based methods
- Deep learning on point cloud and parametric models

Extrinsic vs Intrinsic CNNs

Intrinsic

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

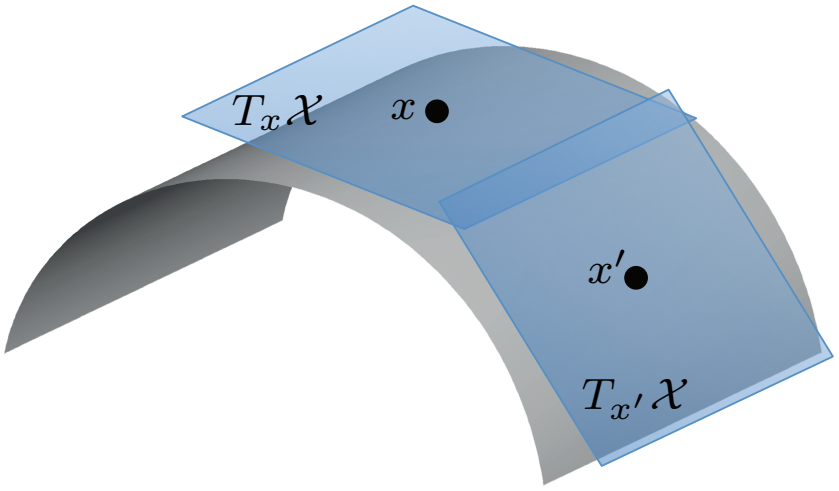


Riemannian geometry in one minute

- 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



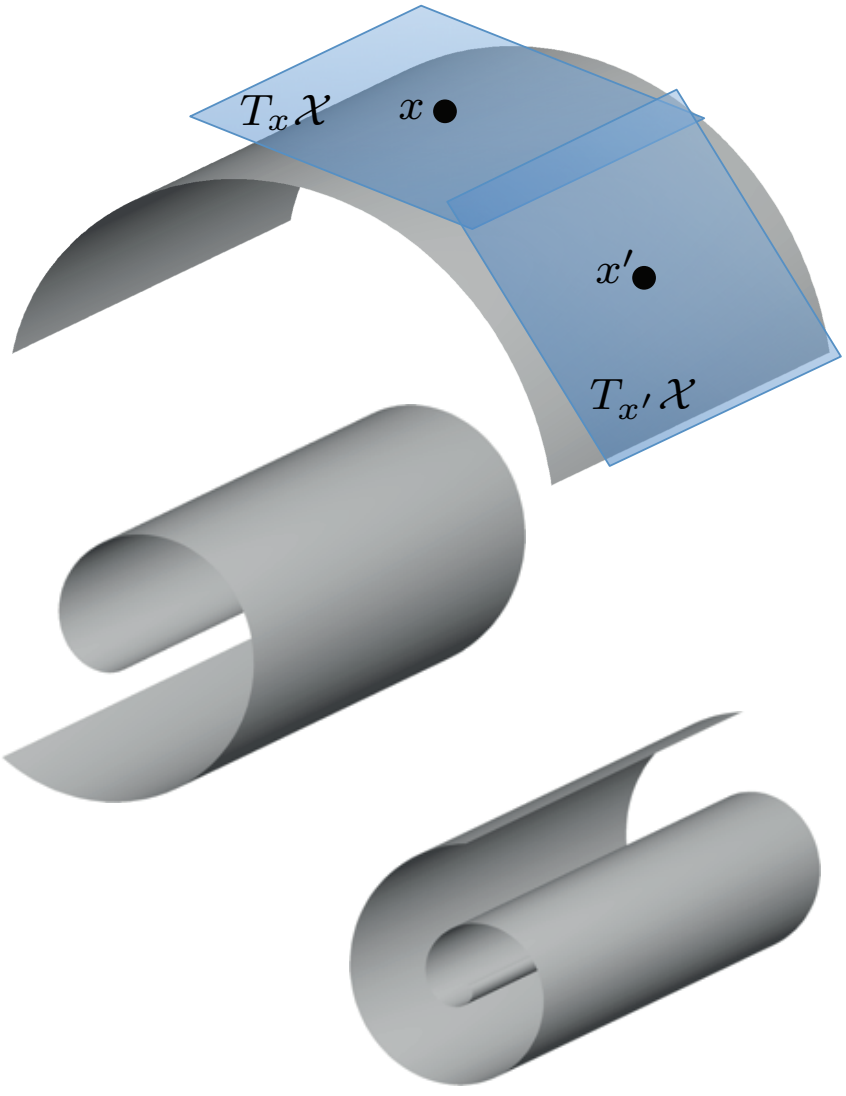
Riemannian geometry in one minute

- 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

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

depending smoothly on x

Isometry = metric-preserving shape deformation



Riemannian geometry in one minute

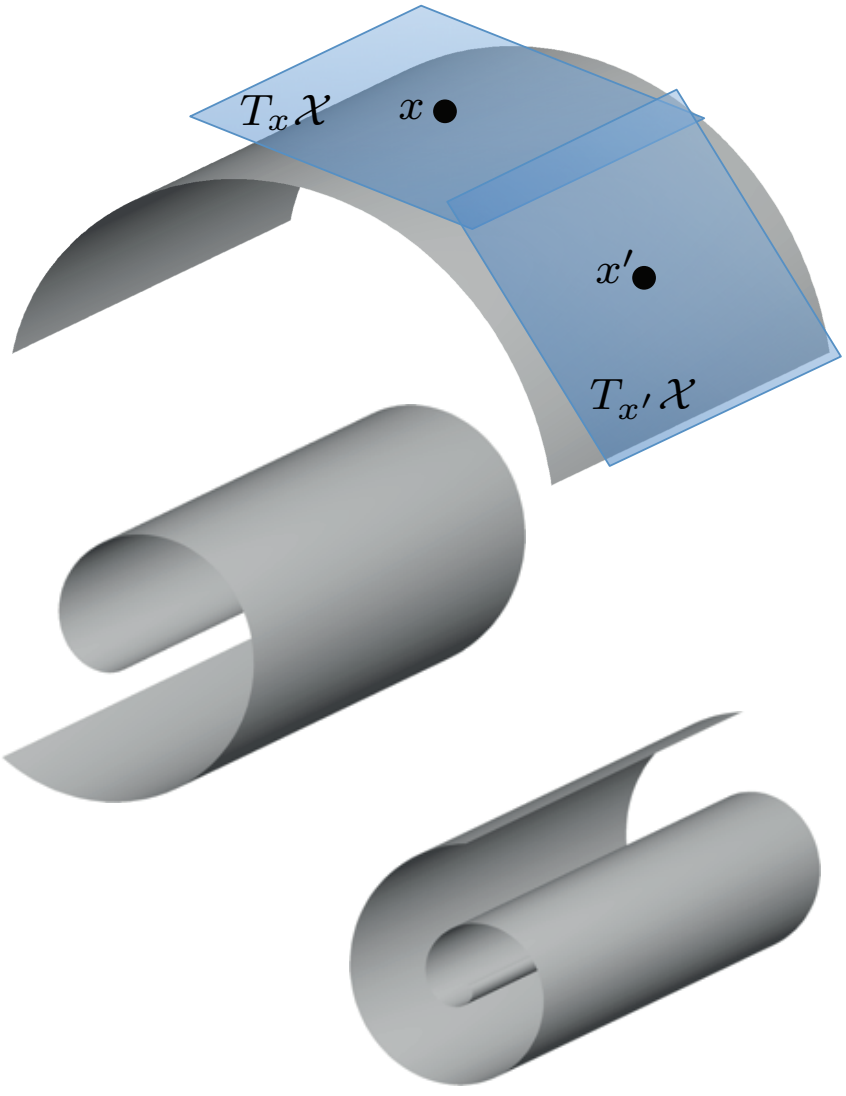
- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric

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

depending smoothly on x

Isometry = metric-preserving shape deformation

Intrinsic = expressed solely in terms of the Riemannian metric



Riemannian geometry in one minute

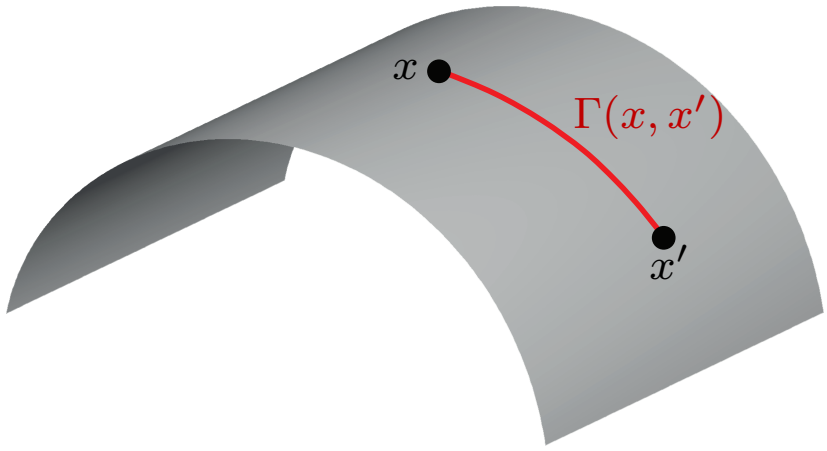
- Manifold $\mathcal{X}$ = topological space
- No global Euclidean structure
- Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x
- Riemannian metric

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

  depending smoothly on x

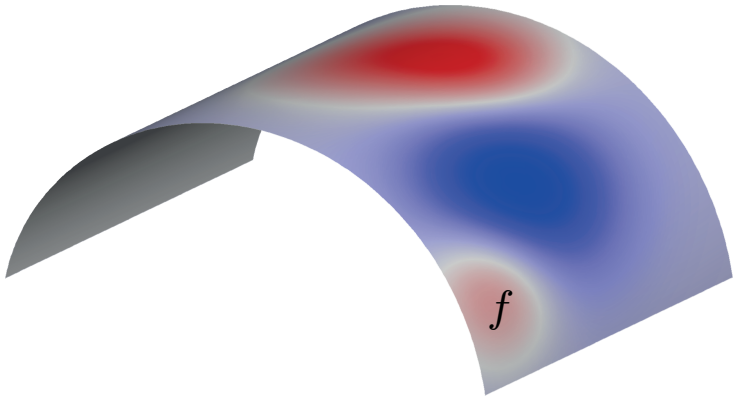
  Isometry = metric-preserving shape deformation

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



Calculus on manifolds: scalar fields

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



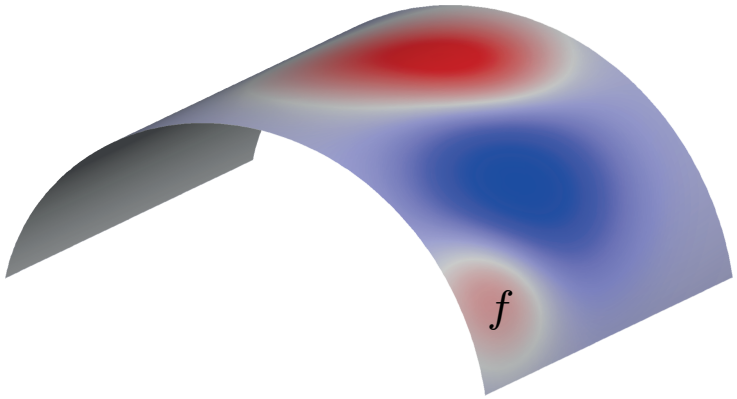
Calculus on manifolds: scalar fields

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

Hilbert space $L^2(\mathcal{X})$ with inner product

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

where $dx =$ area element induced by the Riemannian metric

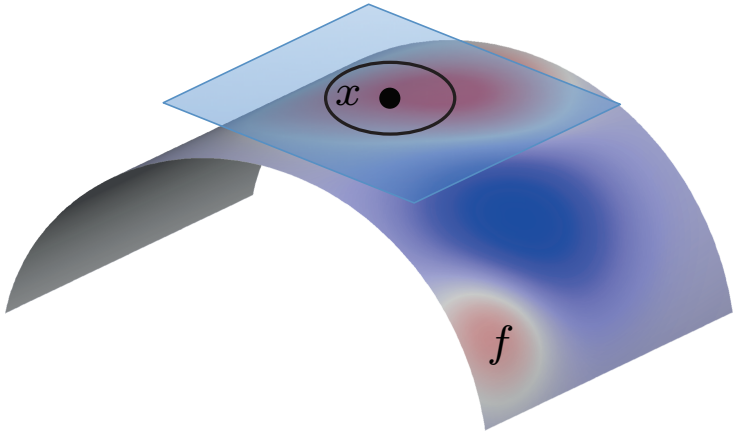


Calculus on manifolds: Laplacian operator

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

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

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

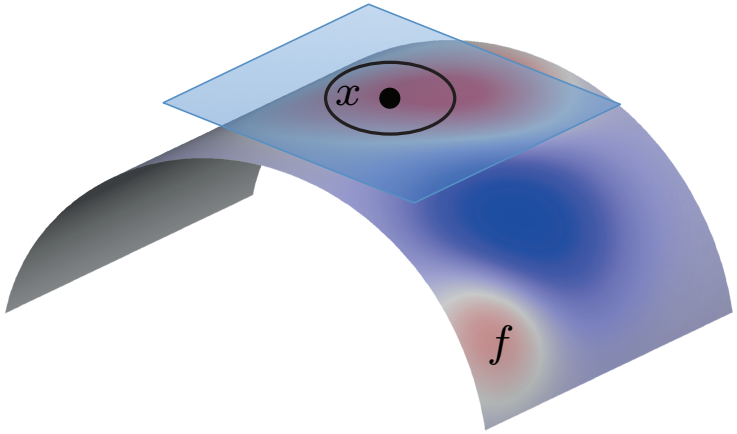


Calculus on manifolds: Laplacian operator

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

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

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



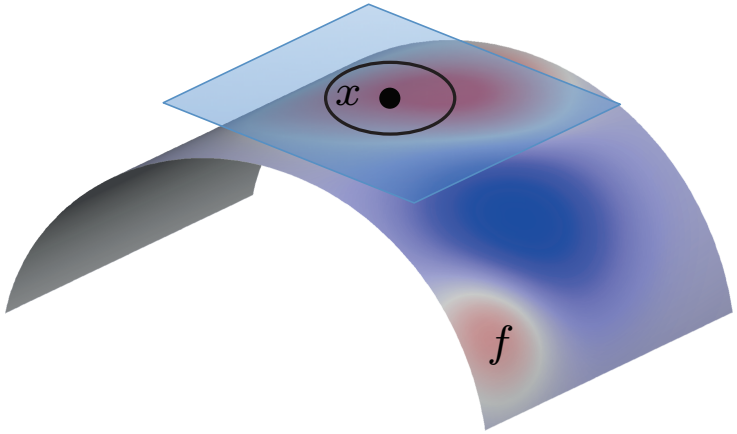
- Intrinsic (expressed solely in terms of the Riemannian metric)

Calculus on manifolds: Laplacian operator

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

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

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



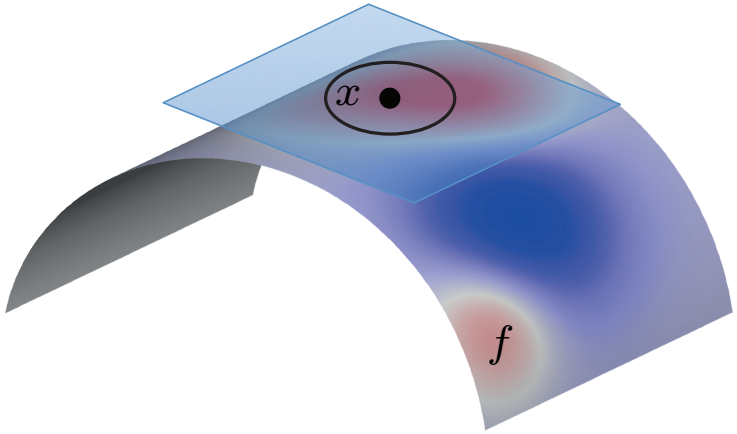
- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant

Calculus on manifolds: Laplacian operator

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

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

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



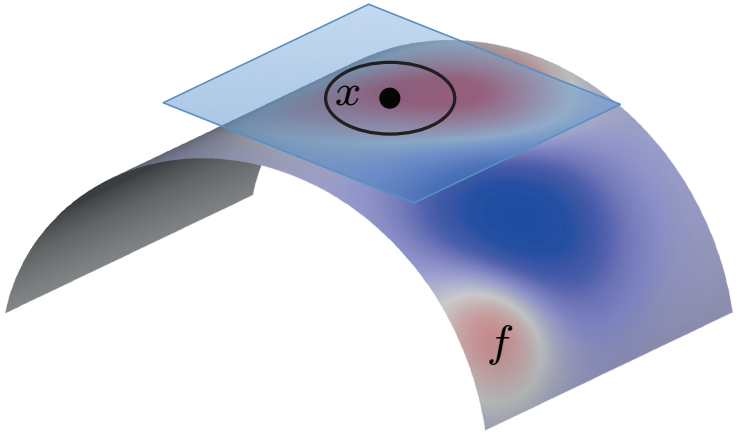
- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})}$

Calculus on manifolds: Laplacian operator

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

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

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



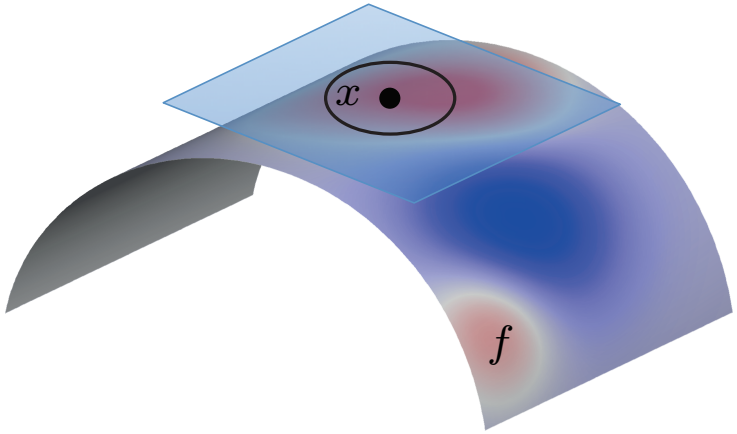
- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow$ orthogonal eigenfunctions

Calculus on manifolds: Laplacian operator

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

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

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



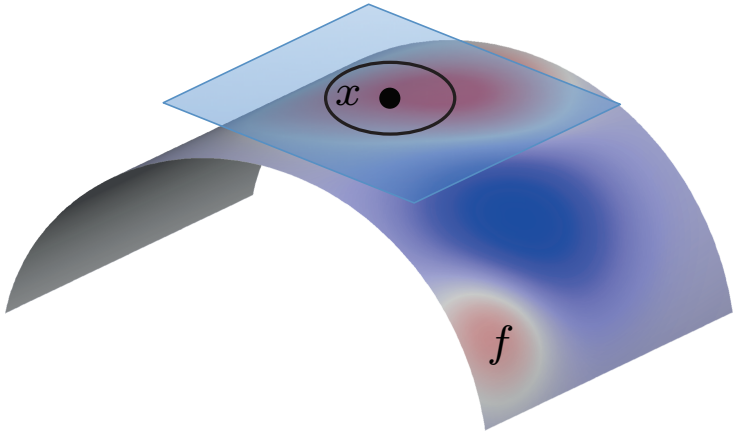
- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow$ orthogonal eigenfunctions
- Positive semidefinite

Calculus on manifolds: Laplacian operator

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

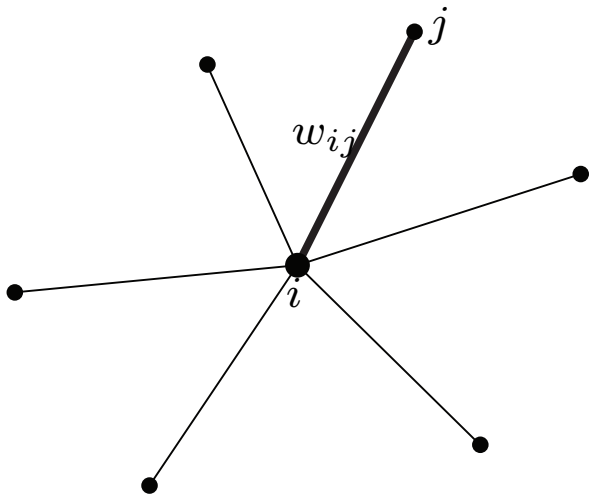
$$\Delta f = -\mathrm{div}(\nabla f)$$

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



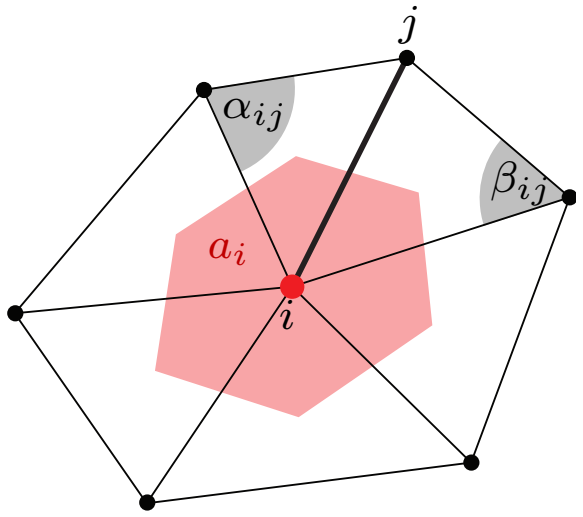
- Intrinsic (expressed solely in terms of the Riemannian metric)
- Isometry-invariant
- Self-adjoint $\langle \Delta f, g \rangle_{L^2(\mathcal{X})} = \langle f, \Delta g \rangle_{L^2(\mathcal{X})} \Rightarrow$ orthogonal eigenfunctions
- Positive semidefinite $\Rightarrow$ non-negative eigenvalues

Discrete Laplacian



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

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



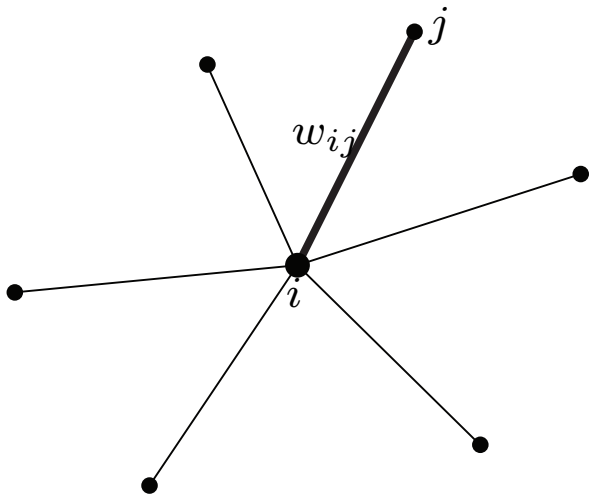
Triangular mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$

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

$a_i =$ local area element

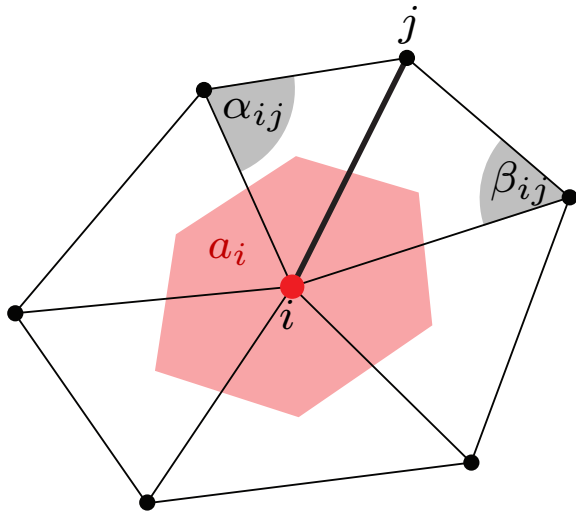
Tutte 1963; MacNeal 1949; Duffin 1959; Pinkall, Polthier 1993

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

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

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

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\mathbf{\Delta}$

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

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

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\boldsymbol{\Delta}$

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

where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues and $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ is a matrix of eigenvectors

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\boldsymbol{\Delta}$

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

where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues and $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ is an $\mathbf{A}$-orthonormal matrix of eigenvectors ($\boldsymbol{\Phi}^\top\mathbf{A}\boldsymbol{\Phi} = \mathbf{I}$)

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\boldsymbol{\Delta}$

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

where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix of eigenvalues and $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ is an $\mathbf{A}$-orthonormal matrix of eigenvectors ($\boldsymbol{\Phi}^\top\mathbf{A}\boldsymbol{\Phi} = \mathbf{I}$)

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

Eigendecomposition of a discrete Laplacian matrix $\boldsymbol{\Delta}$

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

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

Laplacian eigenfunctions and eigenvalues

Laplacian Δ of a compact manifold $\mathcal{X}$ has countably many eigenfunctions

$$\Delta\phi_i(x) = \lambda_i\phi_i(x), \qquad i = 1, 2, \ldots$$

- Eigenfunctions are real and orthonormal $\langle\phi_i, \phi_j\rangle_{L^2(\mathcal{X})} = \delta_{ij}$
- Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots$

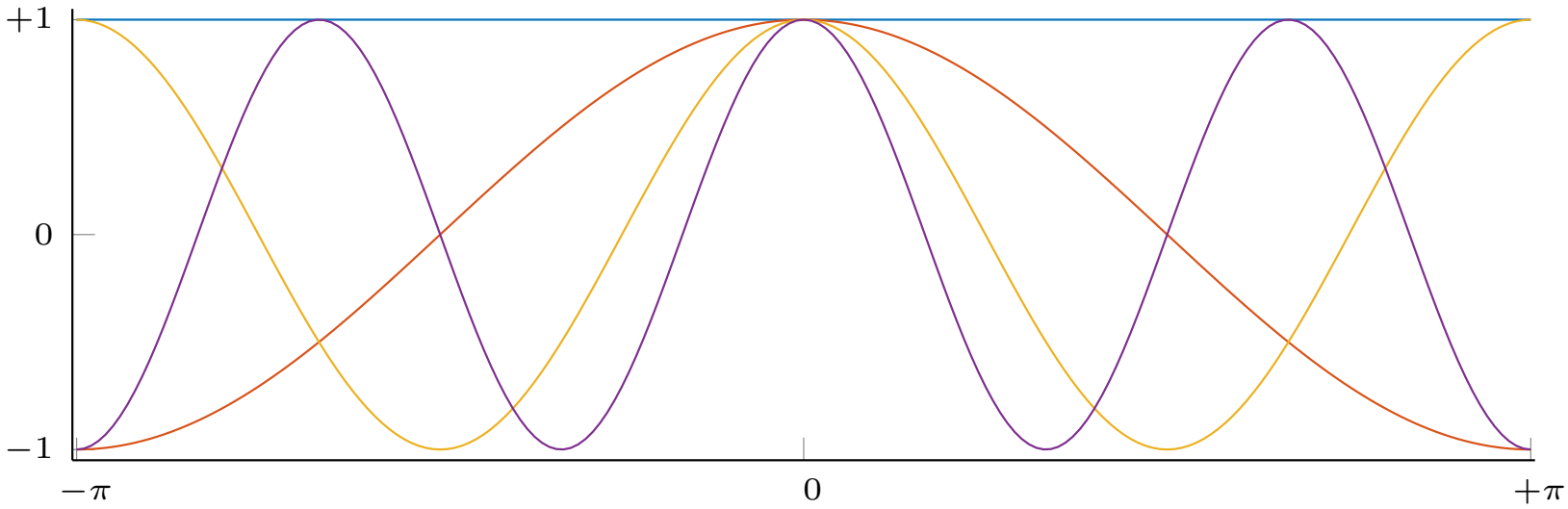
Eigendecomposition of a discrete Laplacian matrix $\boldsymbol{\Delta}$

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

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

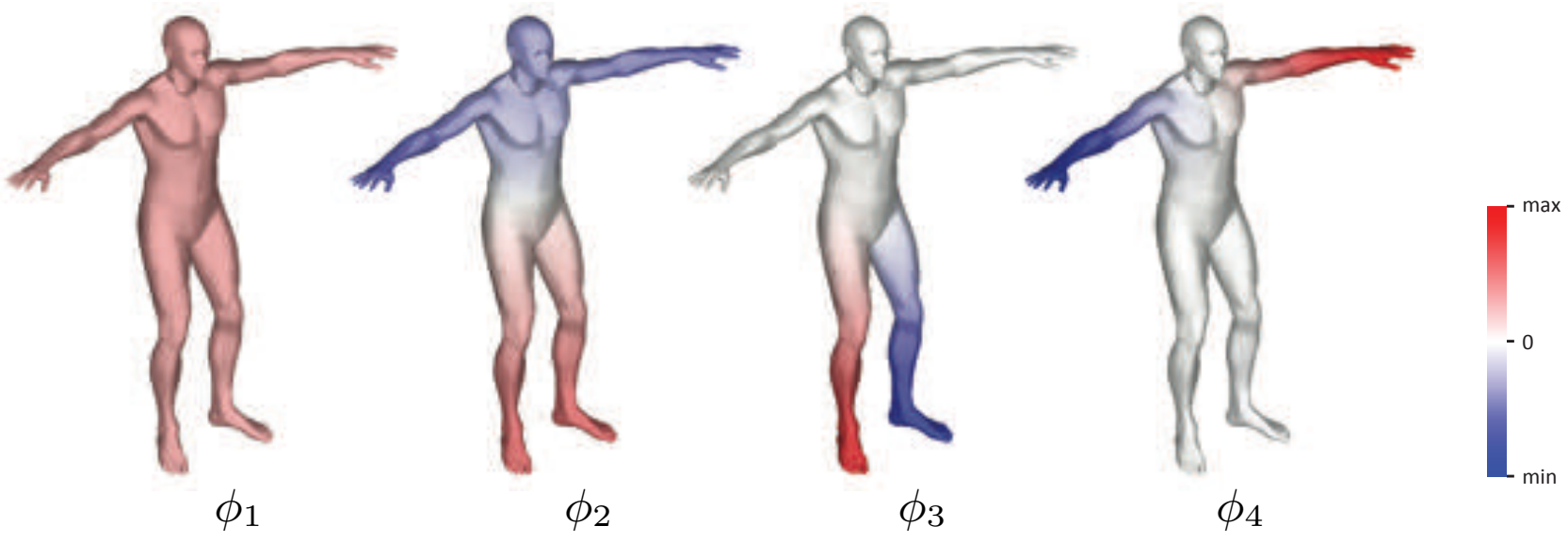
Laplacian eigenvectors = smoothest orthonormal basis

Laplacian eigenfunctions: Euclidean



First eigenfunctions of 1D Euclidean Laplacian = standard Fourier basis

Laplacian eigenfunctions: non-Euclidean



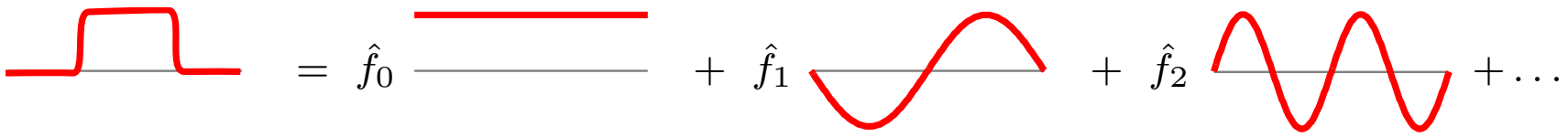
ϕ_1 ϕ_2 ϕ_3 ϕ_4

First eigenfunctions of a manifold Laplacian

Fourier analysis: Euclidean

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

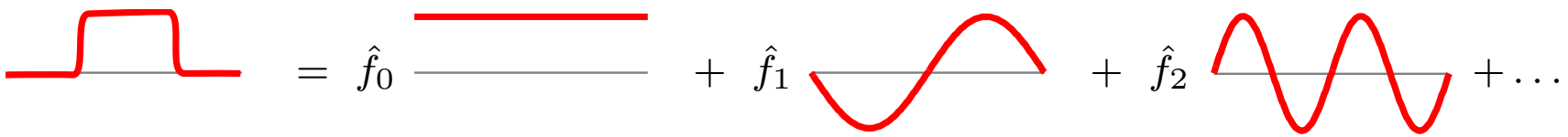
$$f(x) = \sum_{k \geq 0} \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx' \, e^{ikx}$$



Fourier analysis: Euclidean

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

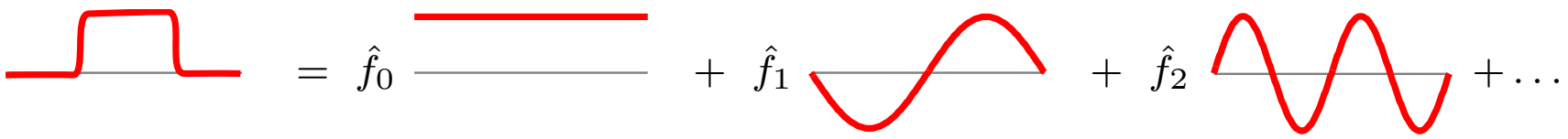
$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} \, dx'}_{\hat{f}_k = \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}} e^{ikx}$$



Fourier analysis: Euclidean

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

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

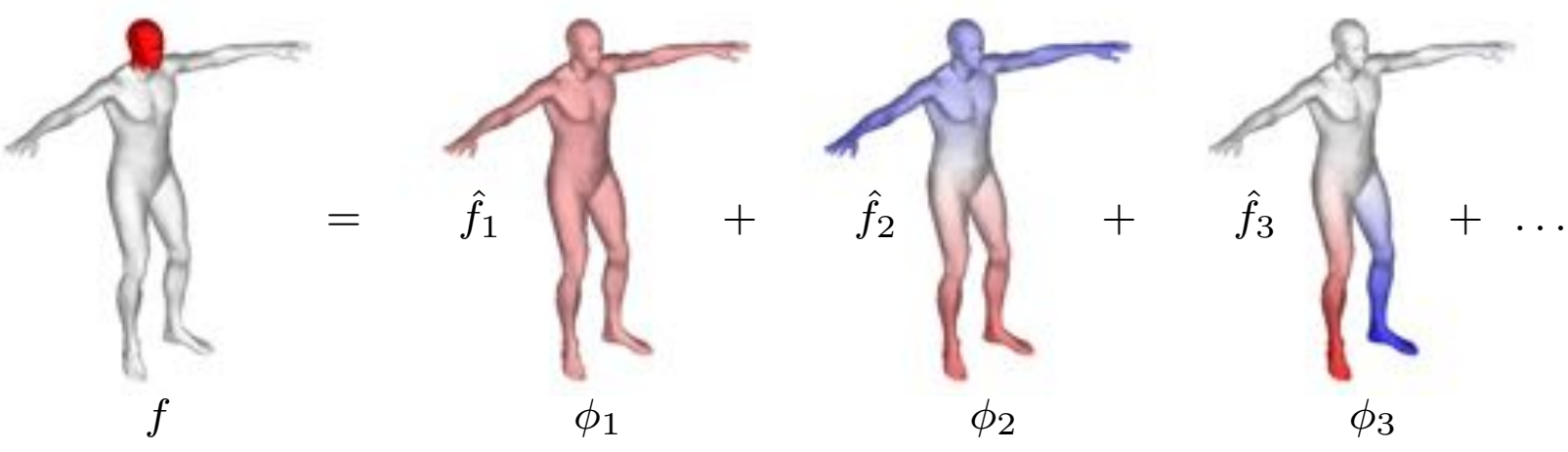


Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

Fourier analysis: non-Euclidean

A function $f : \mathcal{X} \to \mathbb{R}$ 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: $\Delta\phi_k(x) = \lambda_k\phi_k(x)$

Physical application: heat equation

$$f_t = -c\Delta f$$

Newton's law of cooling: rate of change of the temperature of an object is proportional to the difference between its own temperature and the temperature of the surrounding

c [m^2/sec] = thermal diffusivity constant

Newton 1701

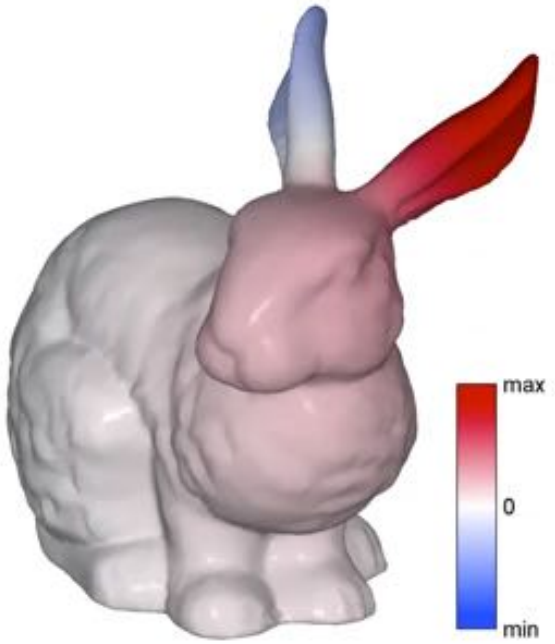
Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$

- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$



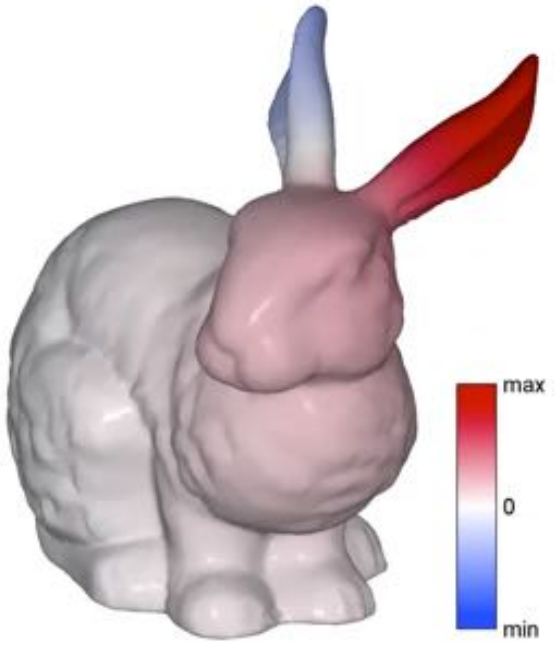
- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Solution of the heat equation expressed through the heat operator

$$f(x,t) \quad = \quad e^{-t\Delta} f_0(x)$$

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$



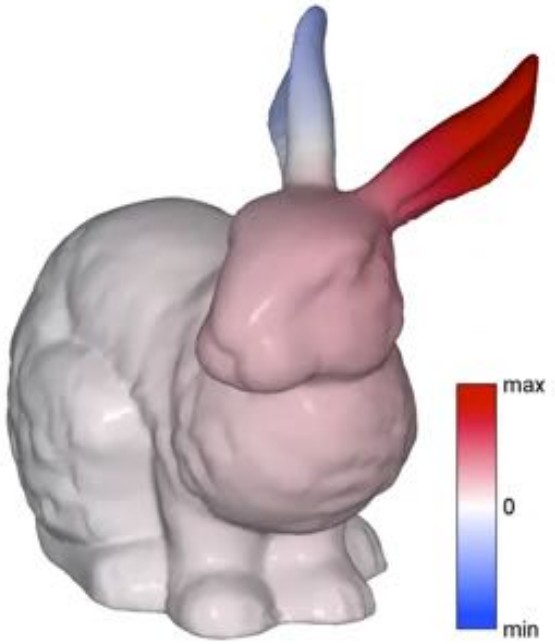
- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Solution of the heat equation expressed through the heat operator

$$f(x,t) \quad = \quad e^{-t\Delta} f_0(x) = \sum_{k \geq 1} \langle f_0, \phi_k \rangle_{L^2(\mathcal{X})} e^{-t\lambda_k} \phi_k(x)$$

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$



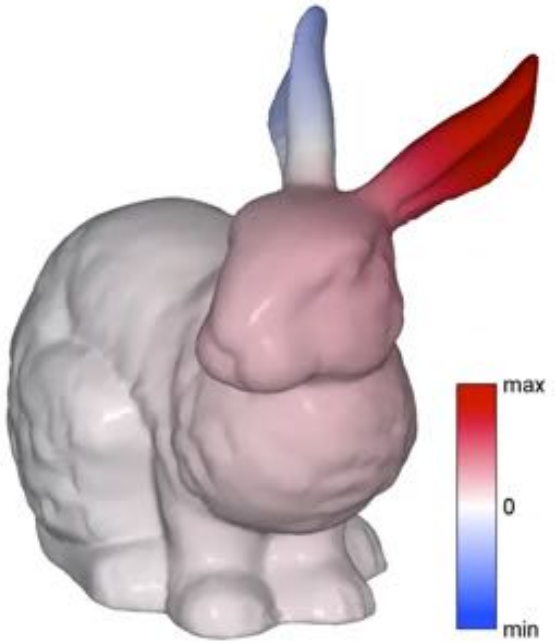
- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

Solution of the heat equation expressed through the heat operator

$$\begin{aligned} f(x,t) &= e^{-t\Delta} f_0(x) = \sum_{k\geq 1} \langle f_0, \phi_k \rangle_{L^2(\mathcal{X})} e^{-t\lambda_k} \phi_k(x) \\ &= \int_{\mathcal{X}} f_0(x') \sum_{k\geq 1} e^{-t\lambda_k} \phi_k(x) \phi_k(x')\, dx' \end{aligned}$$

Heat diffusion on manifolds

$$\begin{cases} f_t(x,t) = -\Delta f(x,t) \\ f(x,0) = f_0(x) \end{cases}$$

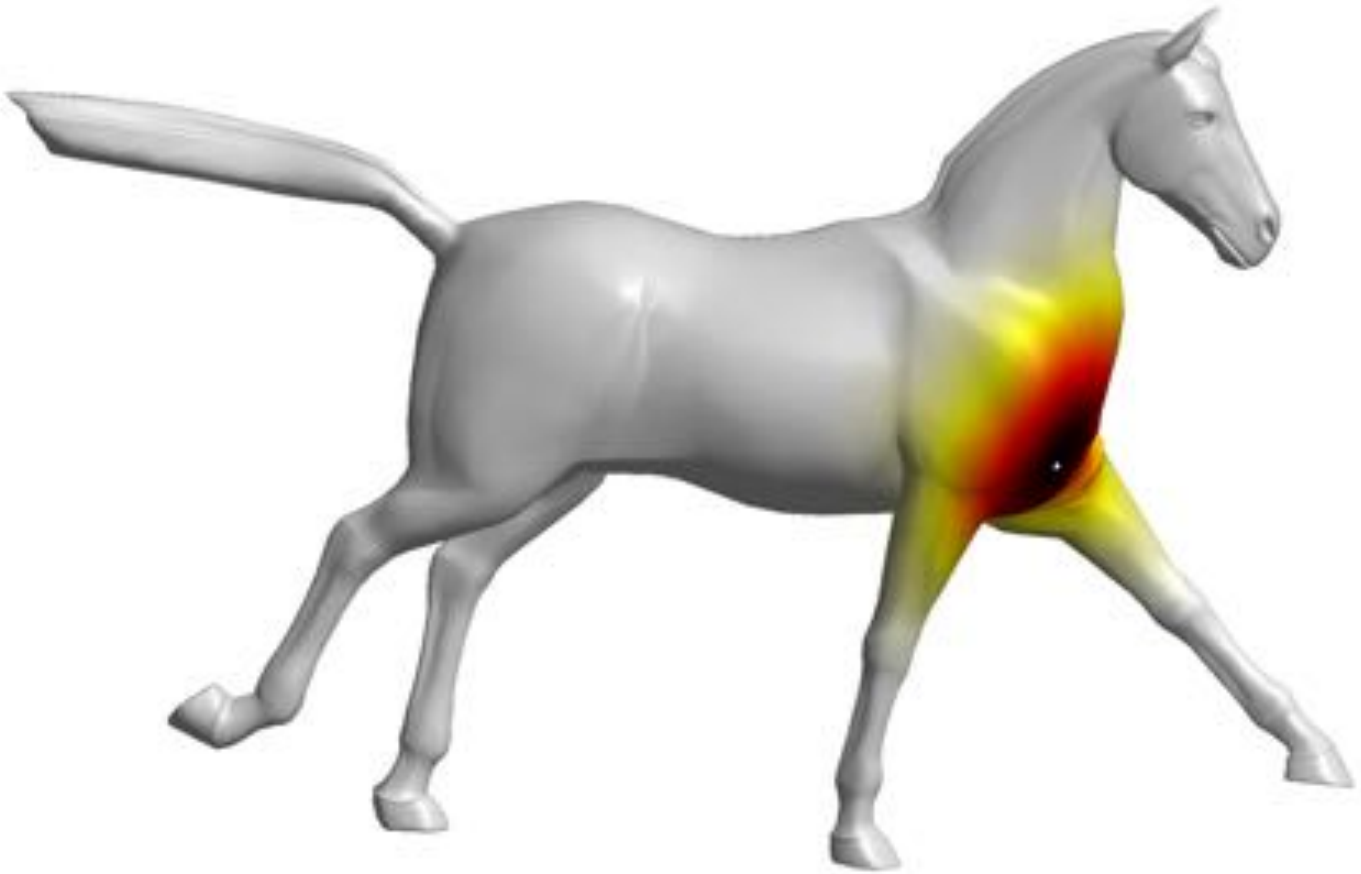


- $f(x,t) =$ amount of heat at point x at time t
- $f_0(x) =$ initial heat distribution

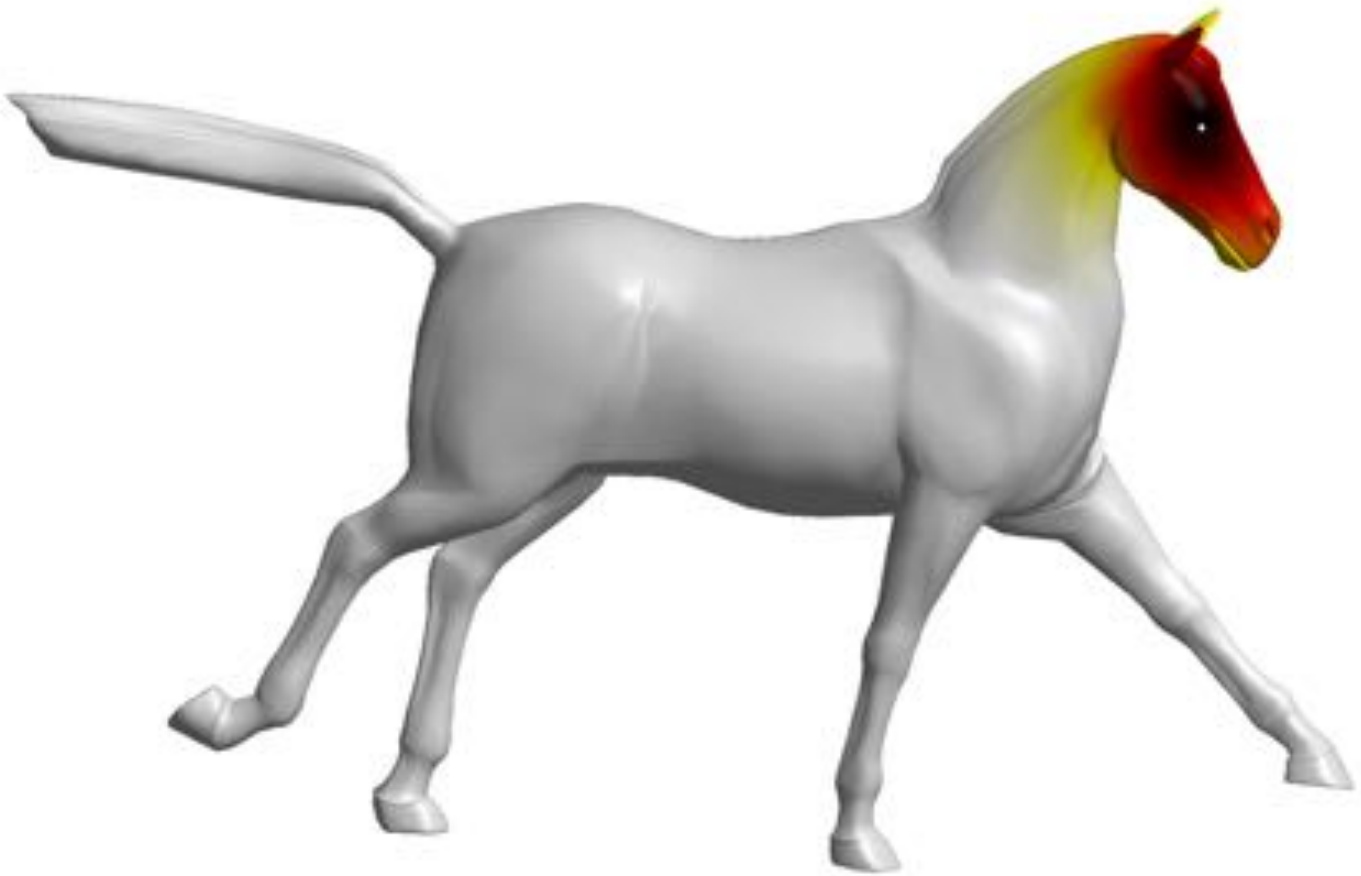
Solution of the heat equation expressed through the heat operator

$$\begin{aligned} f(x,t) &= e^{-t\Delta} f_0(x) = \sum_{k\geq 1} \langle f_0, \phi_k \rangle_{L^2(\mathcal{X})} e^{-t\lambda_k} \phi_k(x) \\ &= \int_{\mathcal{X}} f_0(x') \underbrace{\sum_{k\geq 1} e^{-t\lambda_k} \phi_k(x)\phi_k(x')}_{\text{heat kernel } h_t(x,x')} dx' \end{aligned}$$

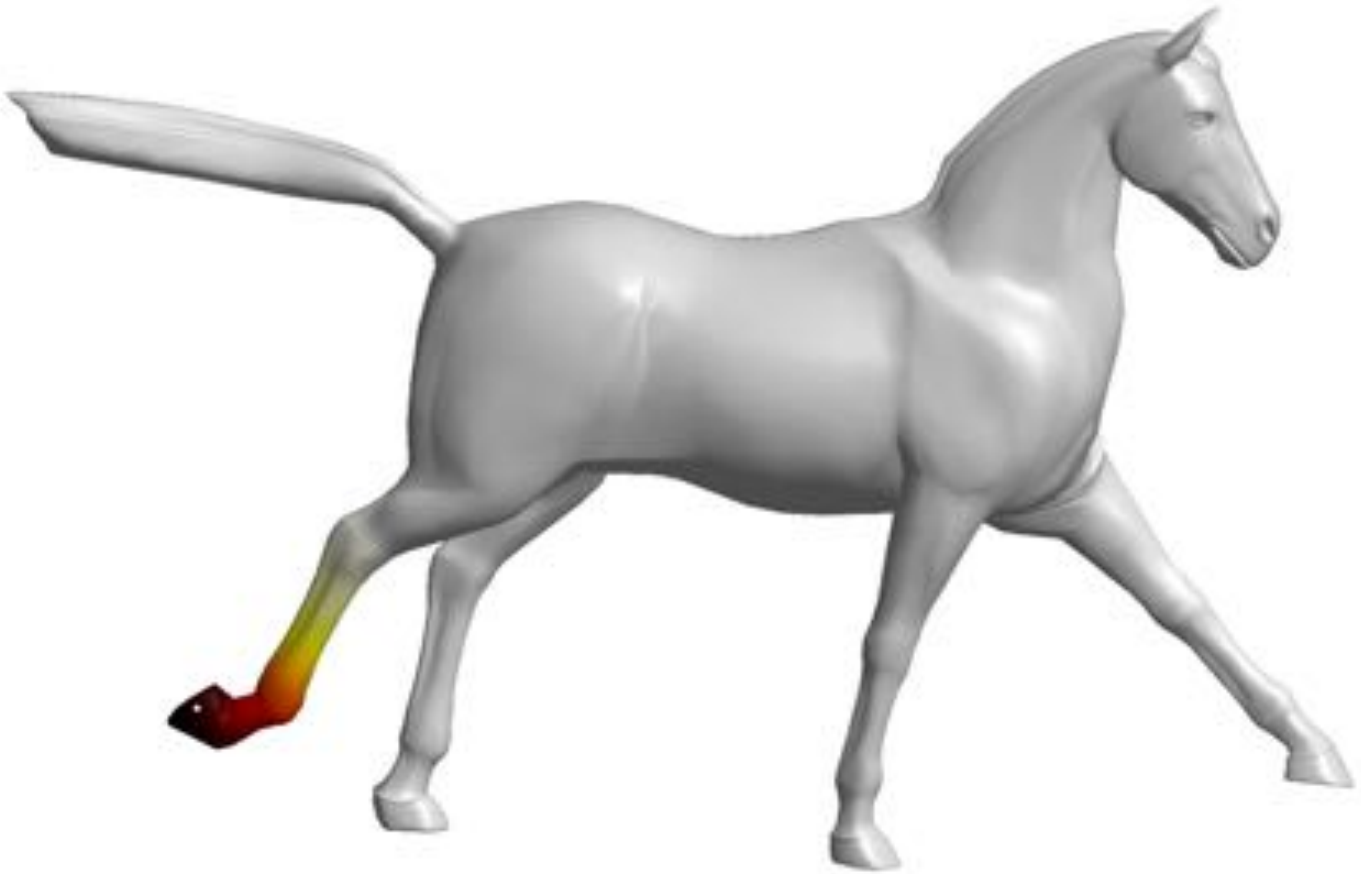
Heat kernels



Heat kernels



Heat kernels



Heat kernels

Interpretation of the heat kernel

Solution of the heat equation on a manifold $\mathcal{X}$ expressed in the Laplacian eigenbasis $\Delta\phi_k(x) = \lambda_k \phi_k(x)$

$$f(x,t) = \int_{\mathcal{X}} f_0(x') \underbrace{\sum_{k\geq 1} e^{-t\lambda_k} \phi_k(x)\phi_k(x')}_{\text{heat kernel } h_t(x,x')} dx'$$

Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space $[-\pi, \pi]$ expressed in the Laplacian eigenbasis $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

$$f(x,t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \underbrace{\sum_{k \geq 0} e^{-tk^2} e^{ikx} e^{-ikx'}}_{\text{heat kernel } h_t(x,x')} dx'$$

Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space $[-\pi, \pi]$ expressed in the Laplacian eigenbasis $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

$$f(x,t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \underbrace{\sum_{k \geq 0} e^{-tk^2} e^{ik(x-x')}}_{\text{heat kernel } h_t(x-x')} dx'$$

Interpretation of the heat kernel

Solution of the heat equation on a Euclidean space $[-\pi, \pi]$ expressed in the Laplacian eigenbasis $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

$$
\begin{aligned}
f(x,t) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f_0(x') \underbrace{\sum_{k \geq 0} e^{-tk^2} e^{ik(x-x')}}_{\text{heat kernel } h_t(x-x')} dx' \\
&= (f_0 \star h_t)(x)
\end{aligned}
$$

Heat kernel = impulse response

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 $f, g : [-\pi, \pi] \to \mathbb{R}$ 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: 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 $\mathbf{g} = (g_1, \ldots, g_n)^\top$

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

Convolution Theorem

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

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

Convolution Theorem

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

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

Convolution Theorem

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

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

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

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

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

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

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

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

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

Spectral convolution

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

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

In matrix-vector notation

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

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

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

In matrix-vector notation

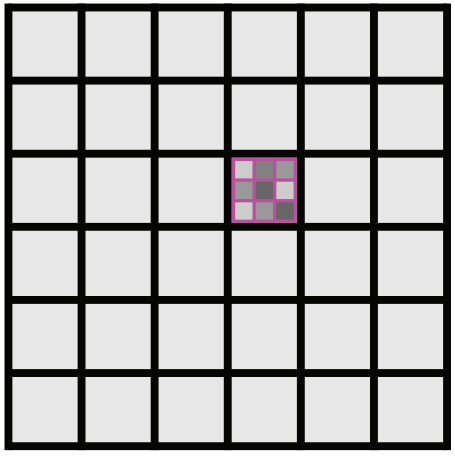
$$\mathbf{f} \star \mathbf{g} = \underbrace{\mathbf{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \mathbf{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

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

Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain



Embedding domain

Spectral CNN

Convolutional layer expressed in the spectral domain

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

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

Bruna et al. 2014

Spectral CNN

Convolutional layer expressed in the spectral domain

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

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

☹ $\mathcal{O}(n)$ parameters per layer

Bruna et al. 2014

Spectral CNN

Convolutional layer expressed in the spectral domain

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

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

- ☹ $\mathcal{O}(n)$ parameters per layer
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on manifolds or graphs)

Bruna et al. 2014

Spectral CNN

Convolutional layer expressed in the spectral domain

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

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

- ☹ $\mathcal{O}(n)$ parameters per layer
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on manifolds or graphs)
- ☹ No guarantee of spatial localization of filters

Bruna et al. 2014

Spectral CNN

Convolutional layer expressed in the spectral domain

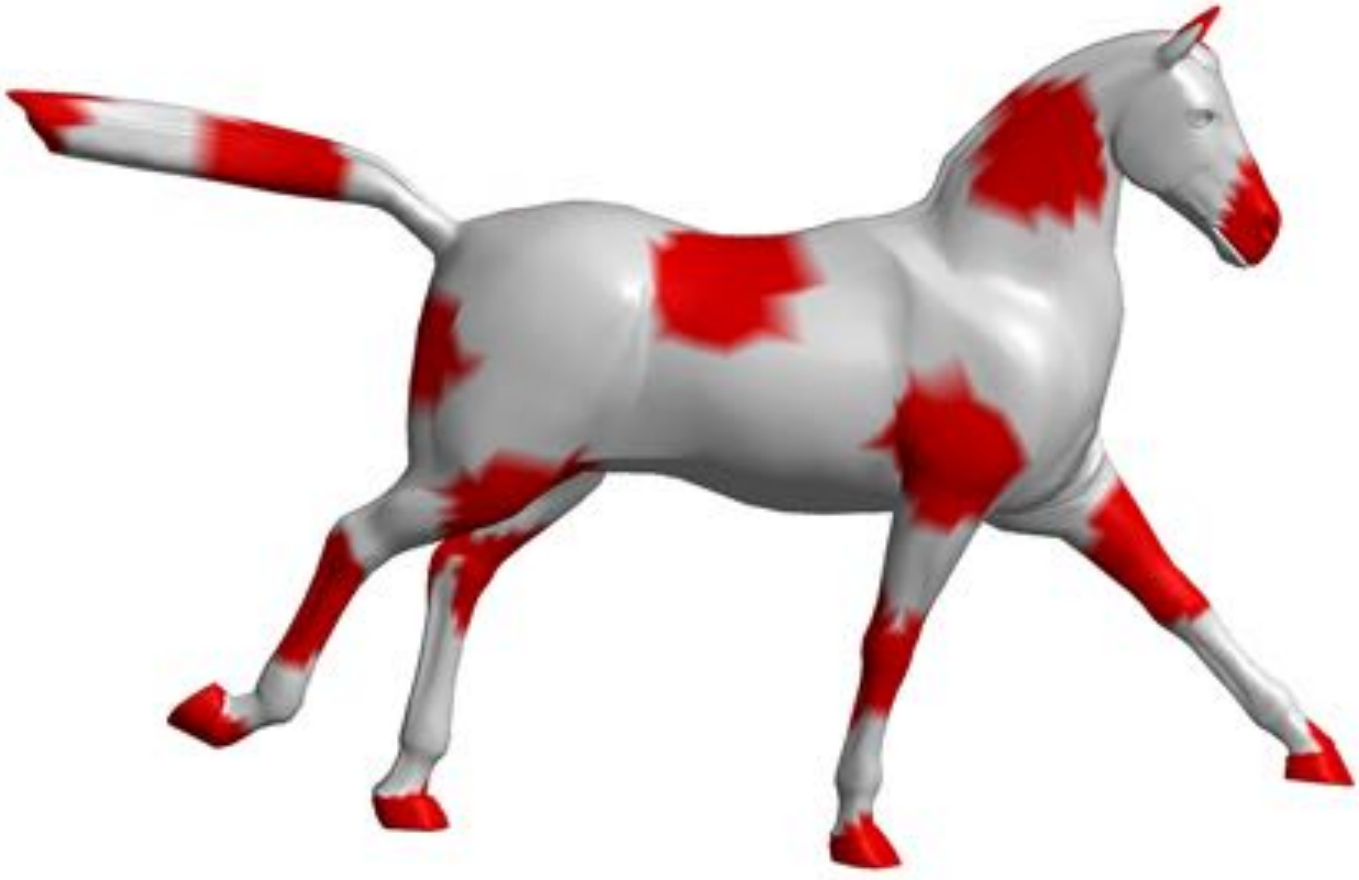
$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\hat{\mathbf{W}}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right) \quad \begin{matrix} l = 1,\dots,q \\ l' = 1,\dots,p \end{matrix}$$

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

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

Bruna et al. 2014

Basis dependence



Function $\mathbf{f}$

Basis dependence

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

Basis dependence

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

Basis dependence

cy Laplacian eigenvector $\phi_{7,2}$

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

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

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

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

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

$\Rightarrow$ Localization in space $=$ smoothness in frequency domain

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

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

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

$\Rightarrow$ 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

In the Euclidean setting (by Parseval's identity)

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

$\Rightarrow$ 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

In the Euclidean setting (by Parseval's identity)

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

$\Rightarrow$ Localization in space $=$ smoothness in frequency domain

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

Application of the filter

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

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

Localization and Smoothness

In the Euclidean setting (by Parseval's identity)

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

$\Rightarrow$ 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

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ 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

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

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

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

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

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support

Defferrard, Bresson, Vandergheynst 2016

Spectral CNN with polynomial filters

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j$$

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

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support
- ☺ 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

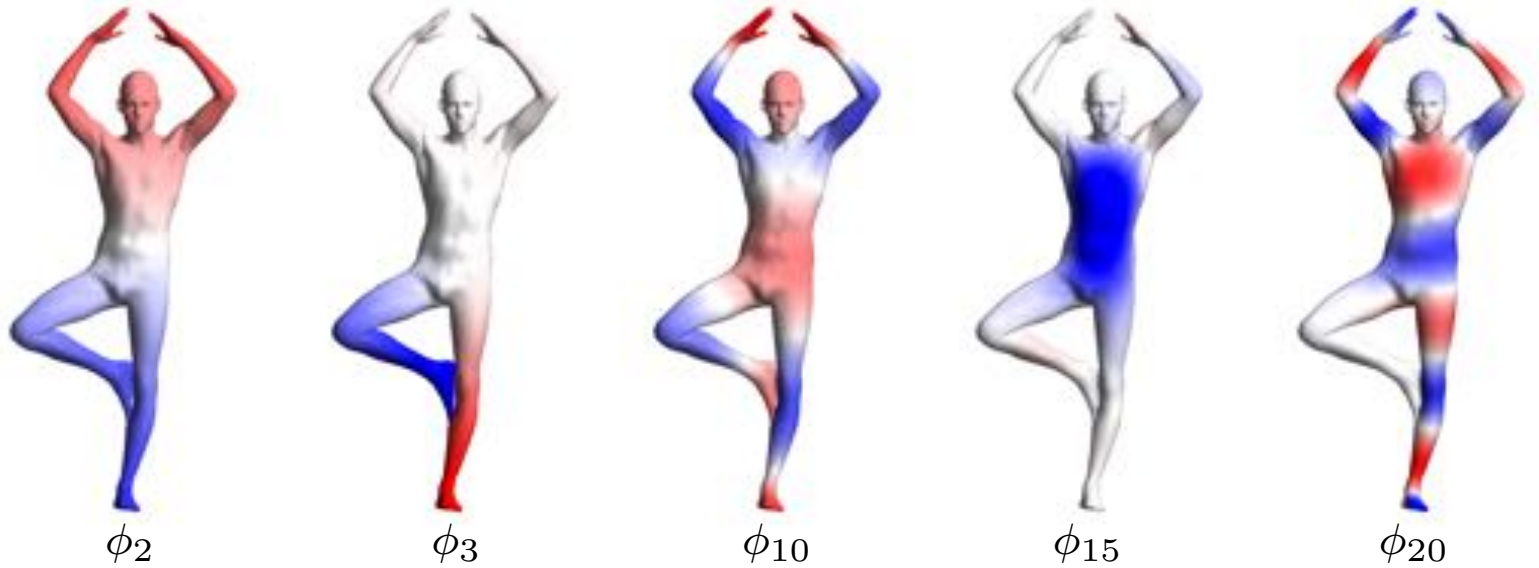
$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j$$

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

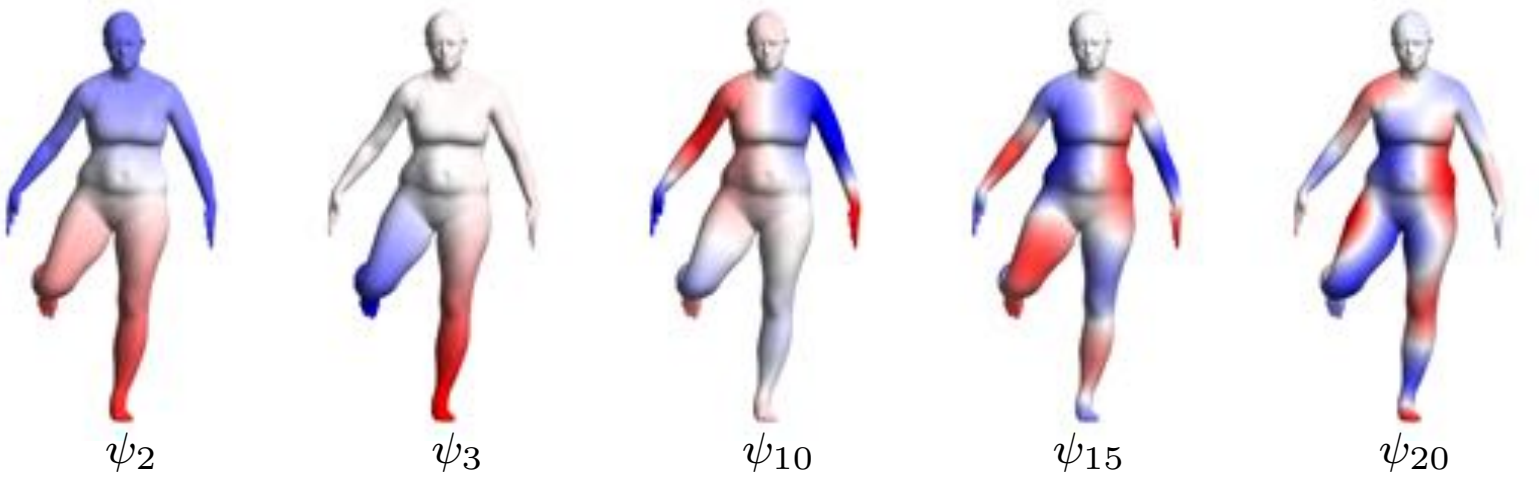
- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support
- ☺ No explicit computation of $\boldsymbol{\Phi}^\top, \boldsymbol{\Phi} \Rightarrow \mathcal{O}(nr)$ complexity
- ☹ Does not generalize across domains

Defferrard, Bresson, Vandergheynst 2016

Laplacian eigenbases on non-isometric domains

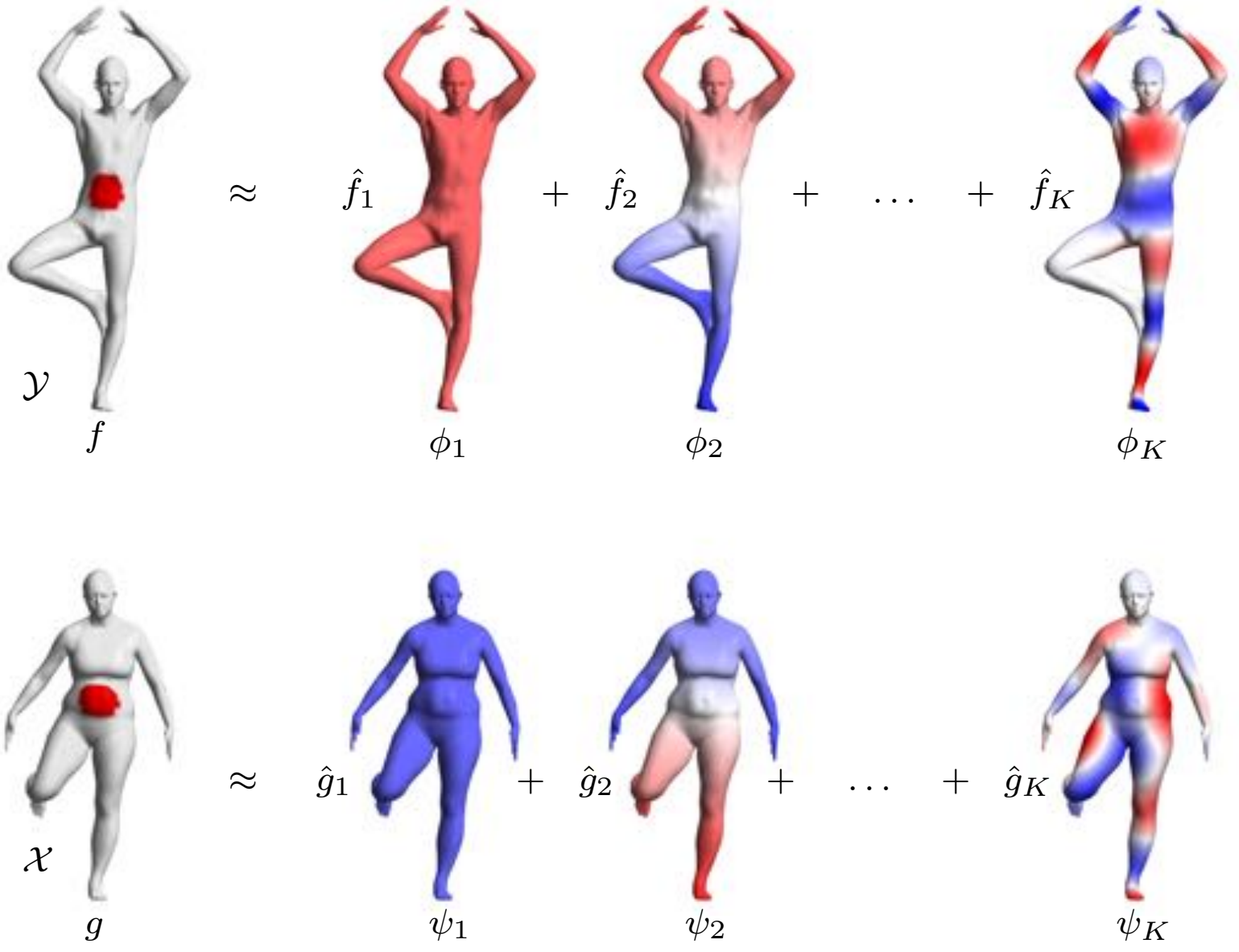


ϕ_2 ϕ_3 ϕ_{10} ϕ_{15} ϕ_{20}



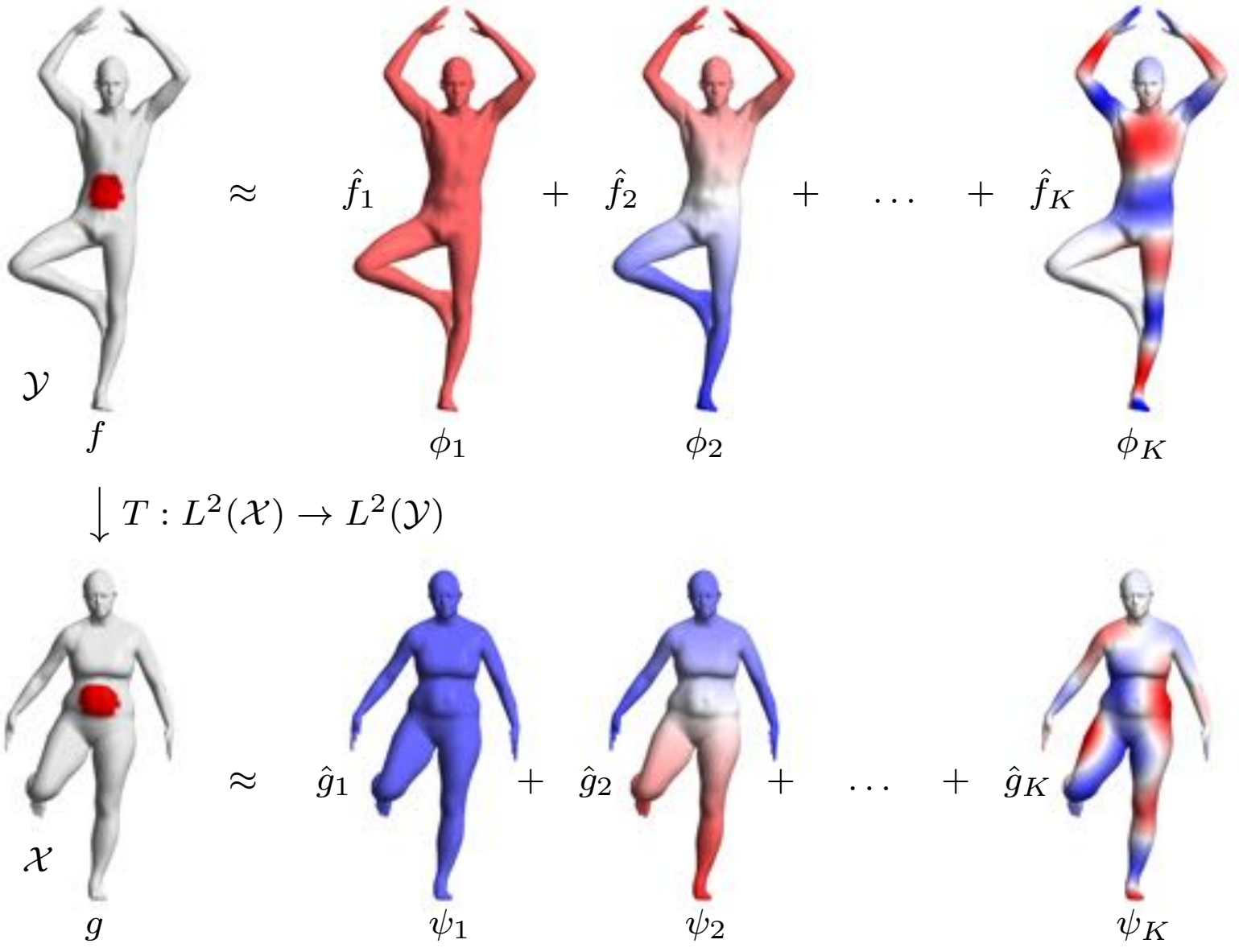
ψ_2 ψ_3 ψ_{10} ψ_{15} ψ_{20}

Functional maps



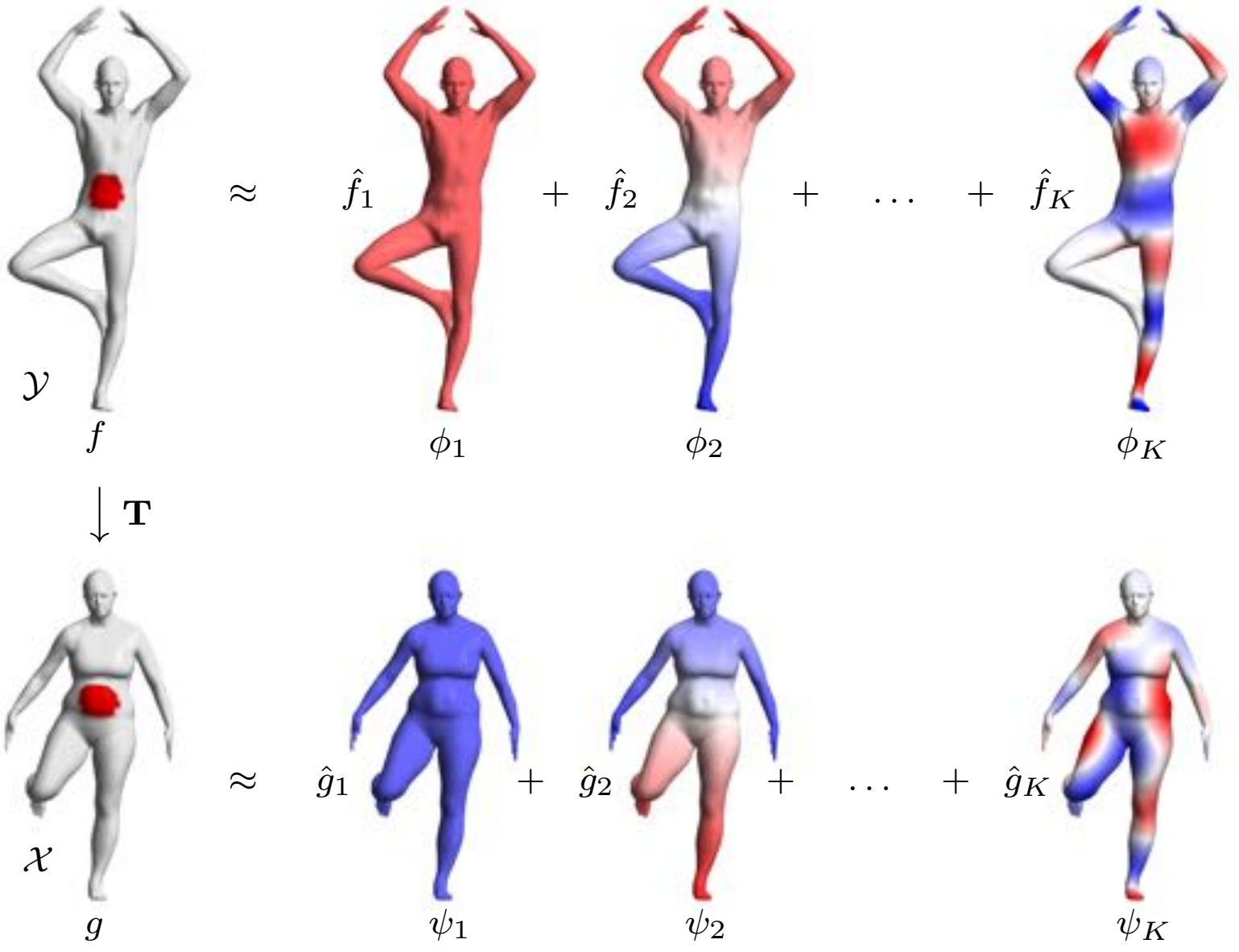
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



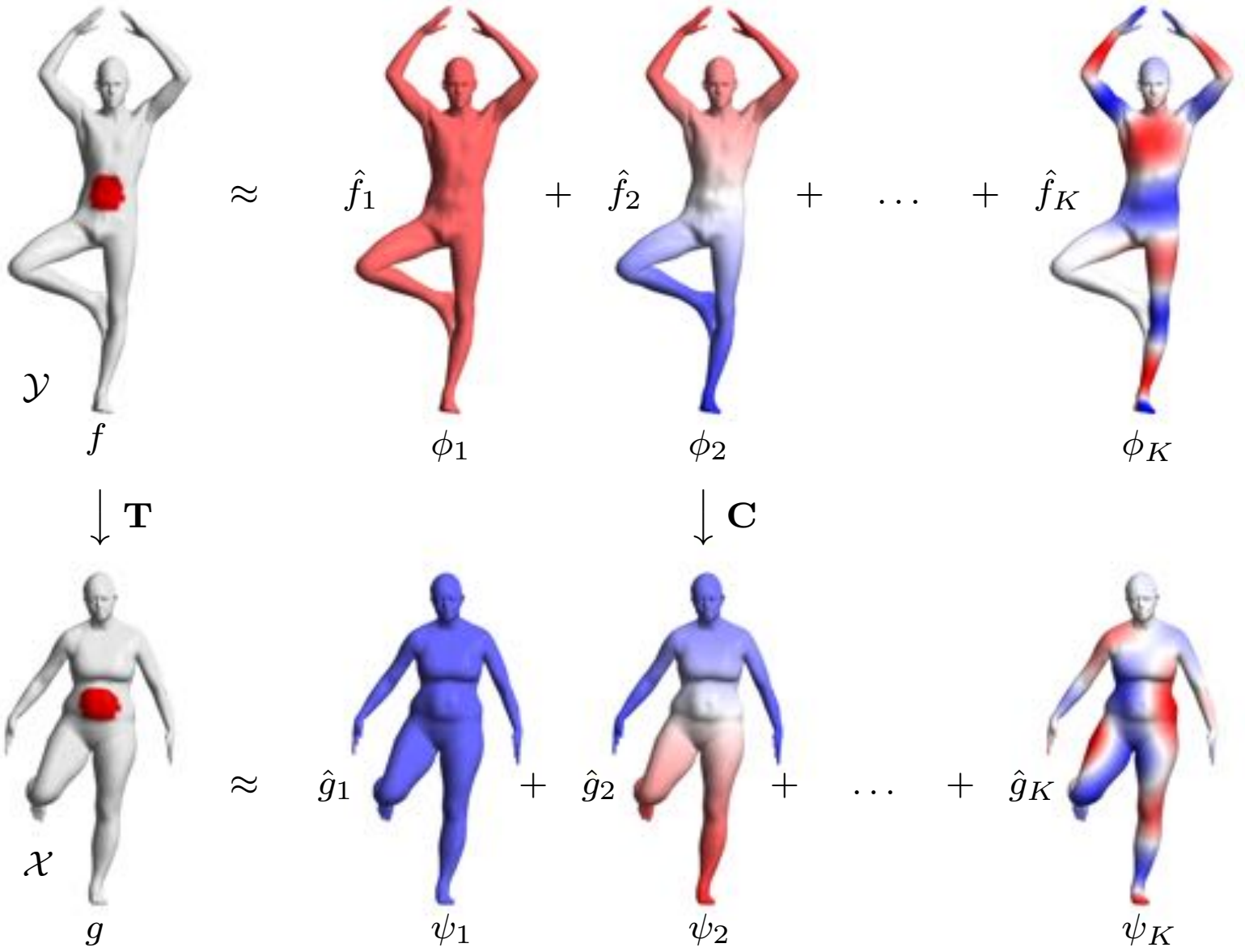
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



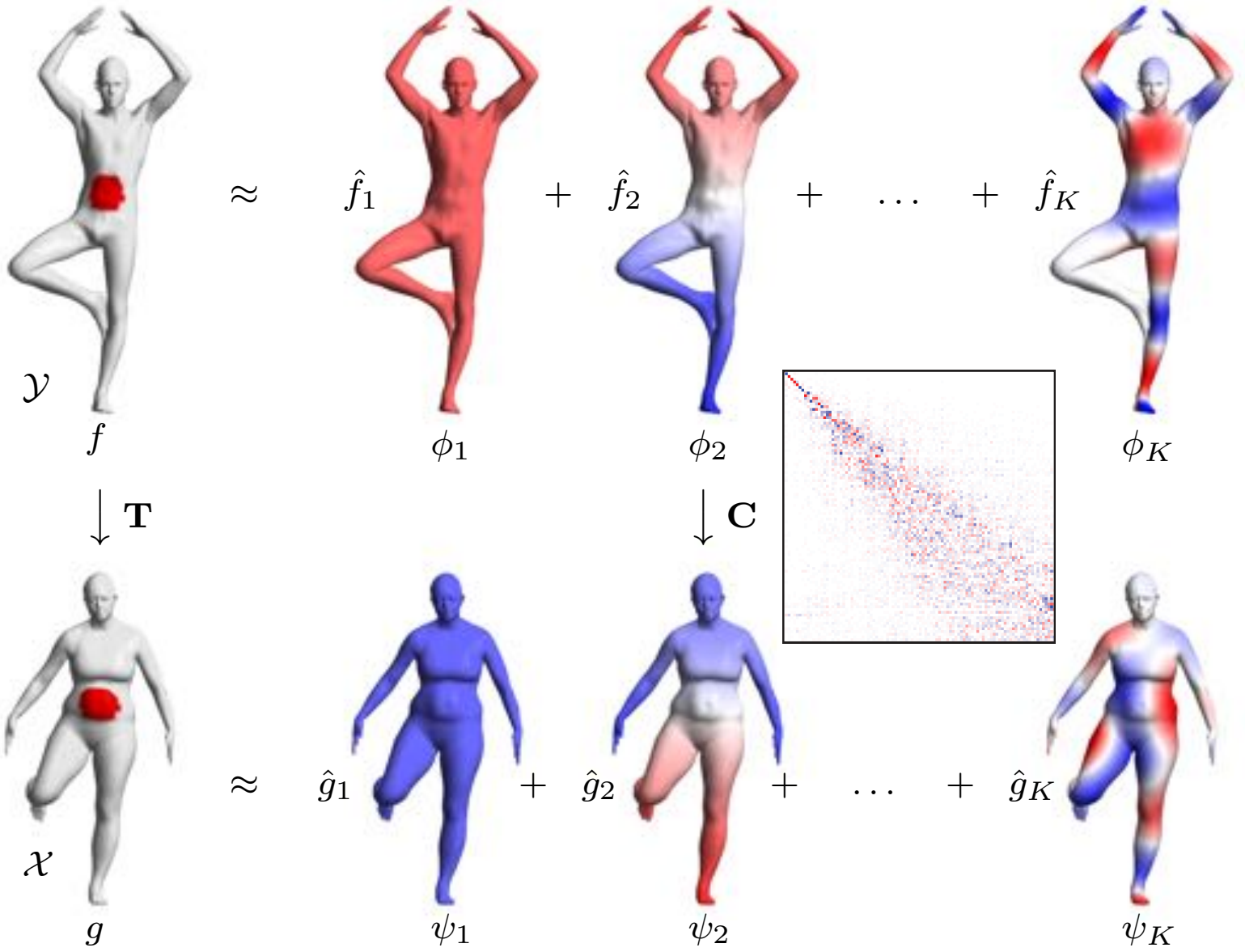
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps



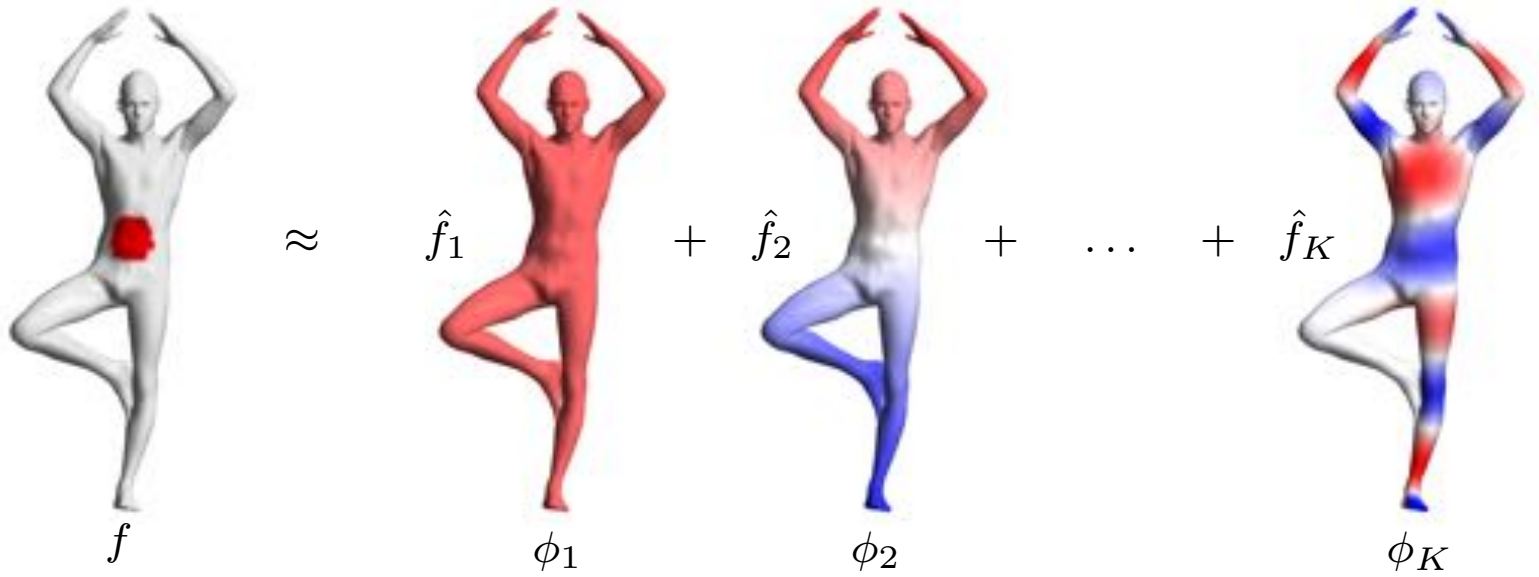
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Functional maps

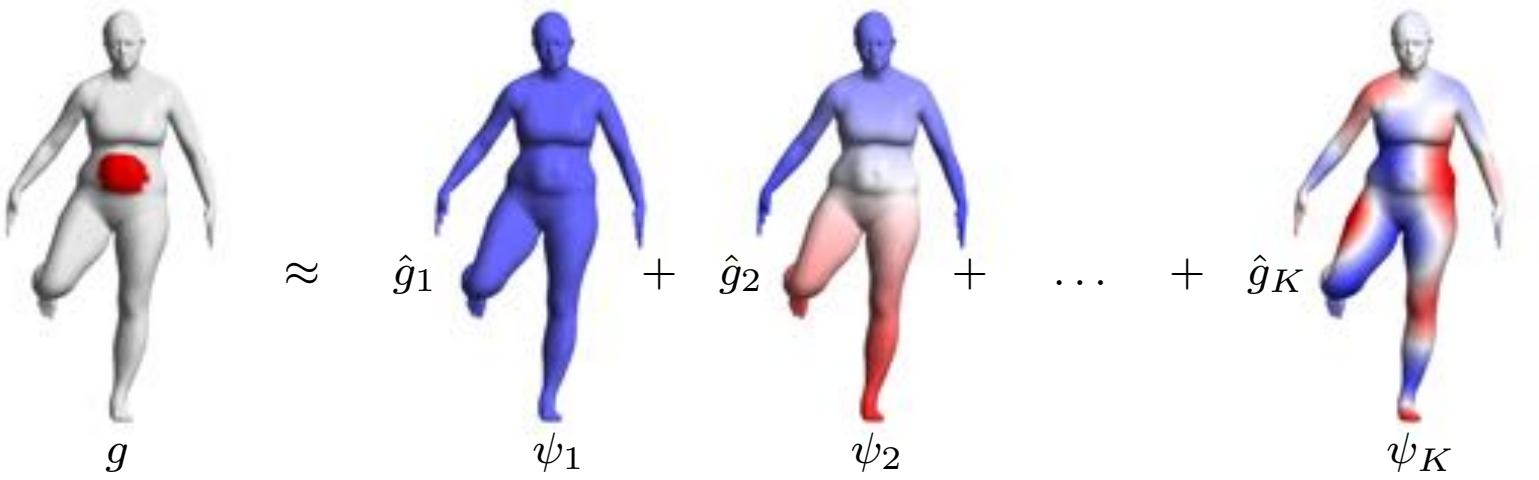


Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Basis synchronization with functional maps

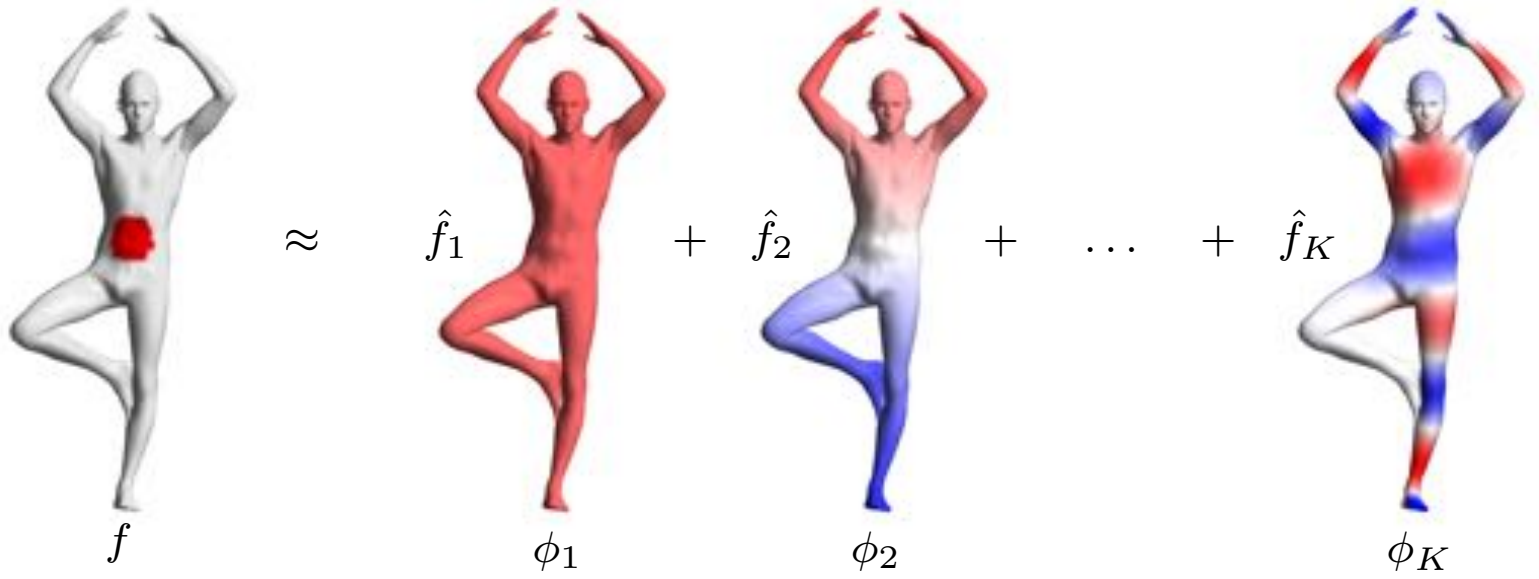


$$\downarrow \mathbf{T} \approx \mathbf{\Psi C}^\top \mathbf{\Phi}^\top$$

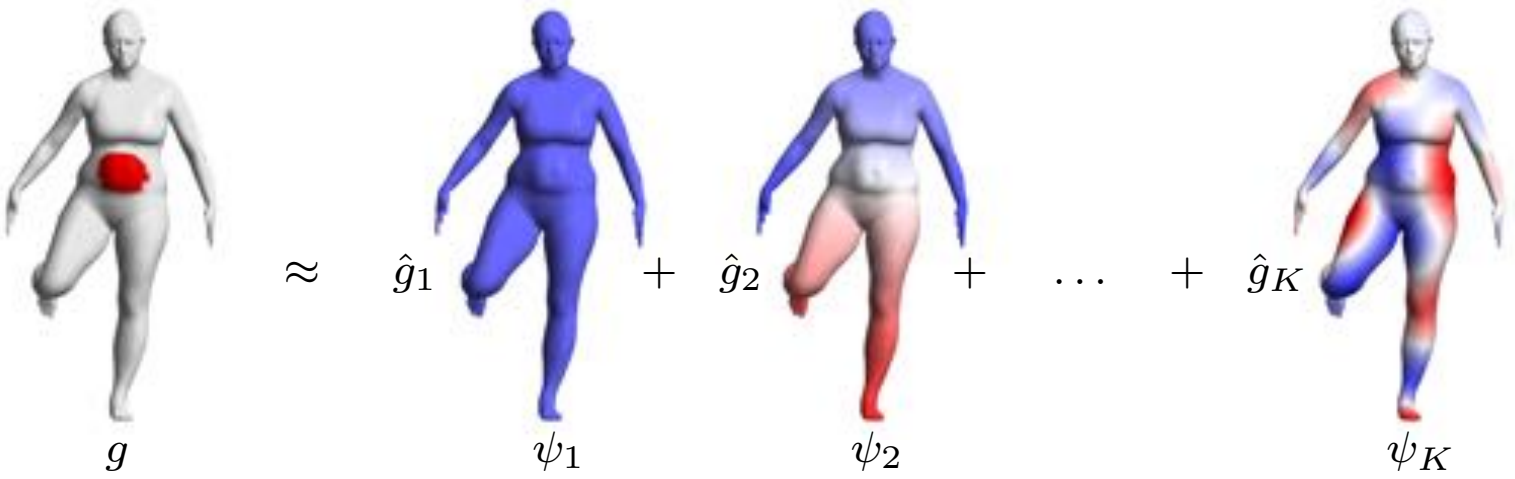


Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

Basis synchronization with functional maps

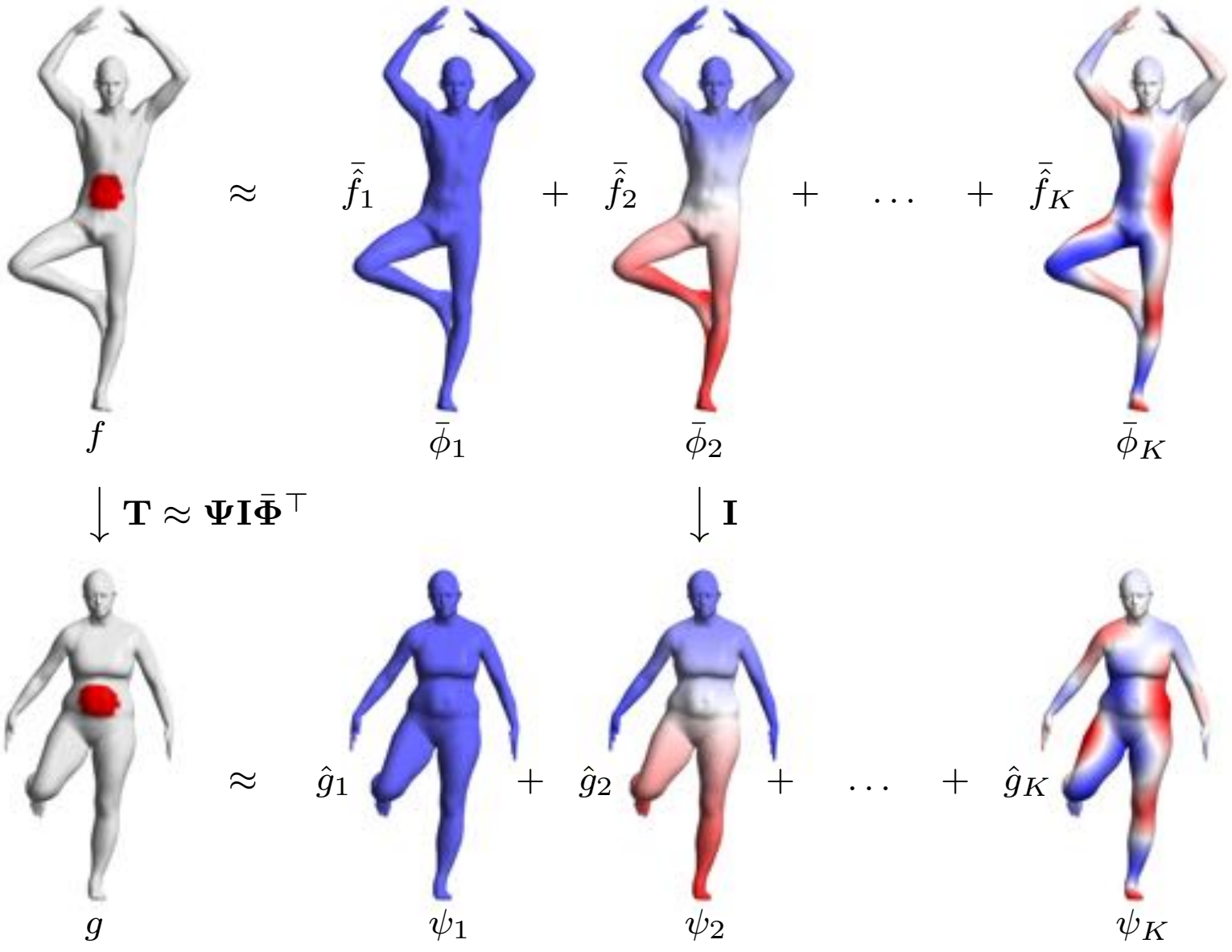


$\downarrow \mathbf{T} \approx \mathbf{\Psi}(\mathbf{\Phi}\mathbf{C})^\top$



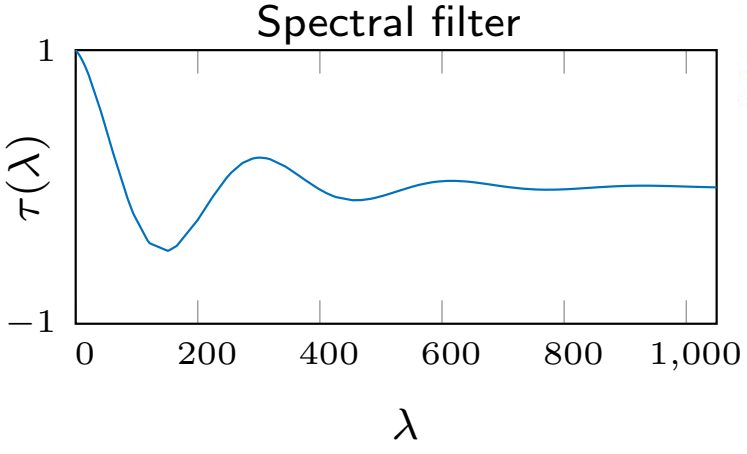
Ovsjanikov et al. 2012; Eynard et al. 2012; Kovnatsky et al. 2013

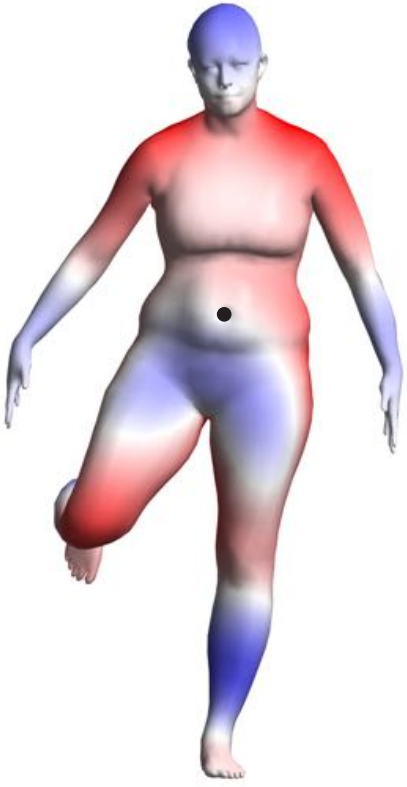
Basis synchronization with functional maps



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

Filtering in different bases





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

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

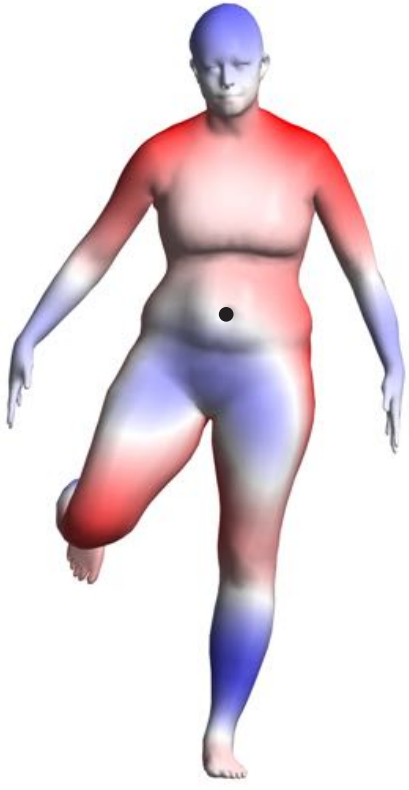
Apply spectral filter $\tau(\lambda)$ in different bases $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$

$\Rightarrow$ **different results!**

Filtering in different bases

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

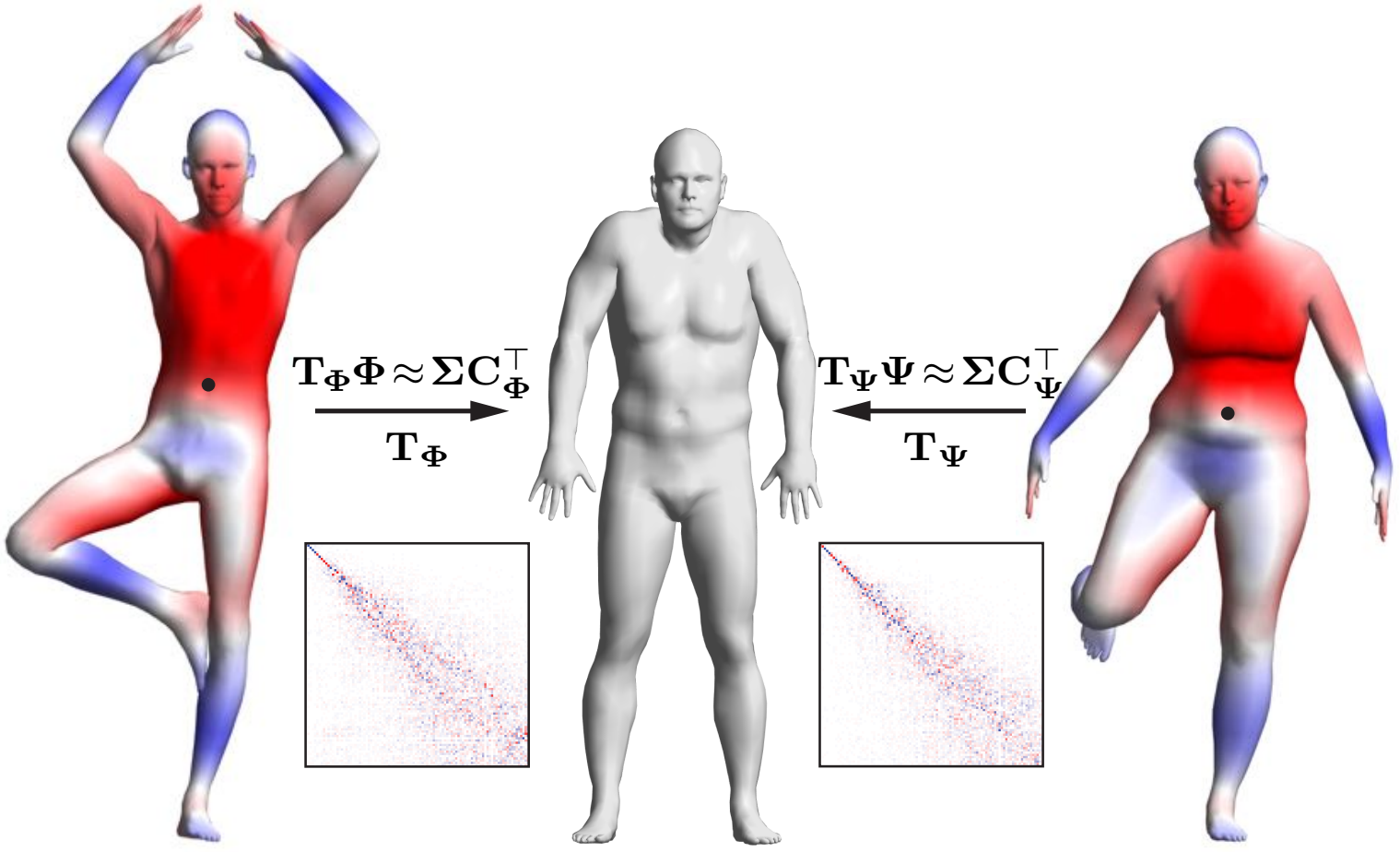
Canonical shape
with basis $\boldsymbol{\Sigma}, \boldsymbol{\Lambda}$



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

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

Filtering in synchronized bases



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

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

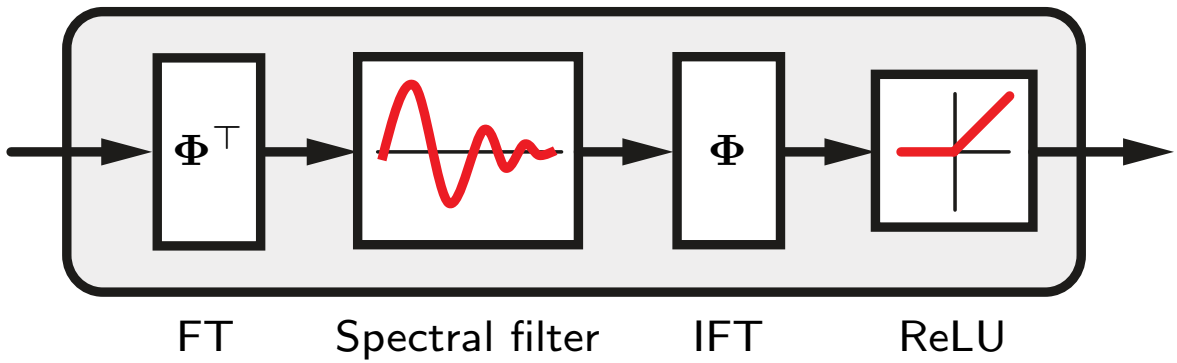
$\mathbf{\Psi}\mathbf{C}_{\mathbf{\Psi}}\tau(\mathbf{\Lambda})\mathbf{C}_{\mathbf{\Psi}}^{\top}\mathbf{\Psi}^{\top}\boldsymbol{\delta}_0$

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

$\Rightarrow$ **similar results!**

Yi et al. 2017

Spectral CNN



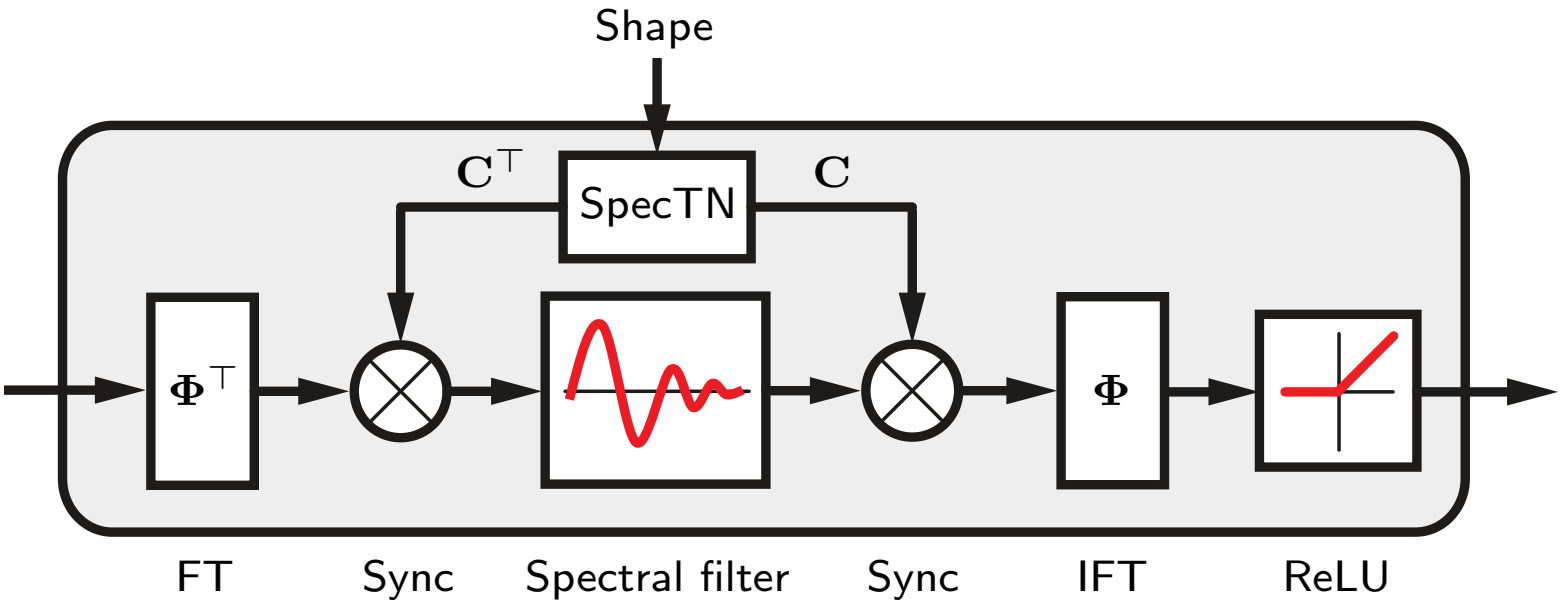
Convolutional filter of a Spectral CNN

☹ 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

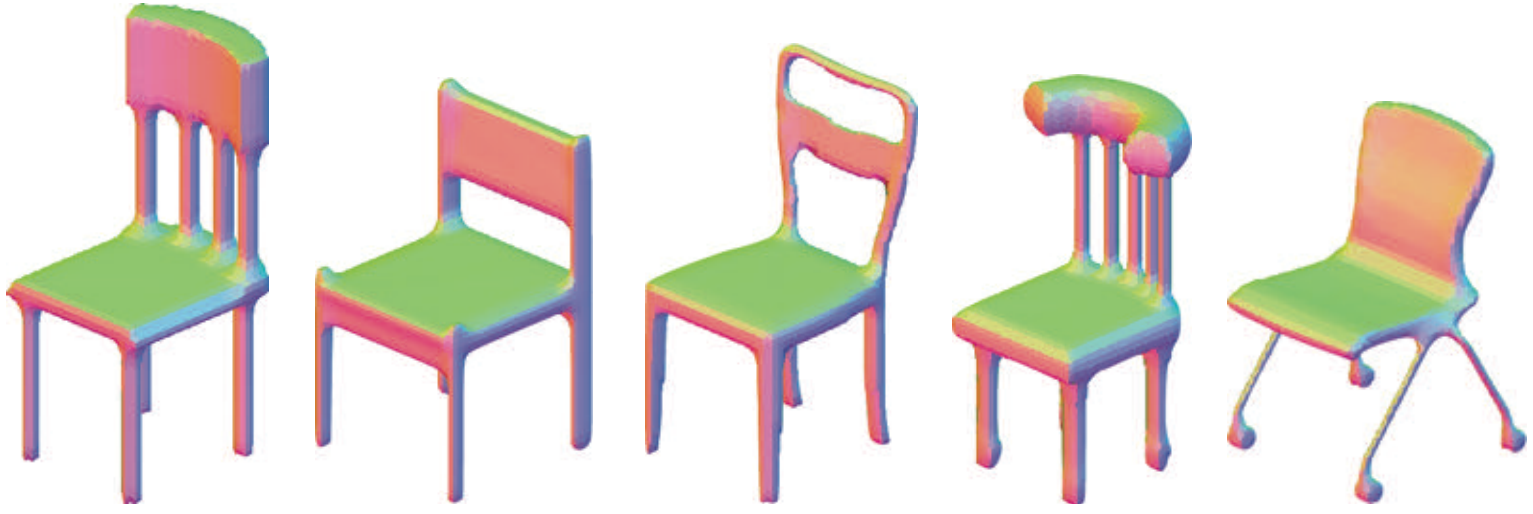
☺ Basis synchronization allows generalization across domains

☹ Explicit FT and IFT

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

Example: normal prediction with SpecTN

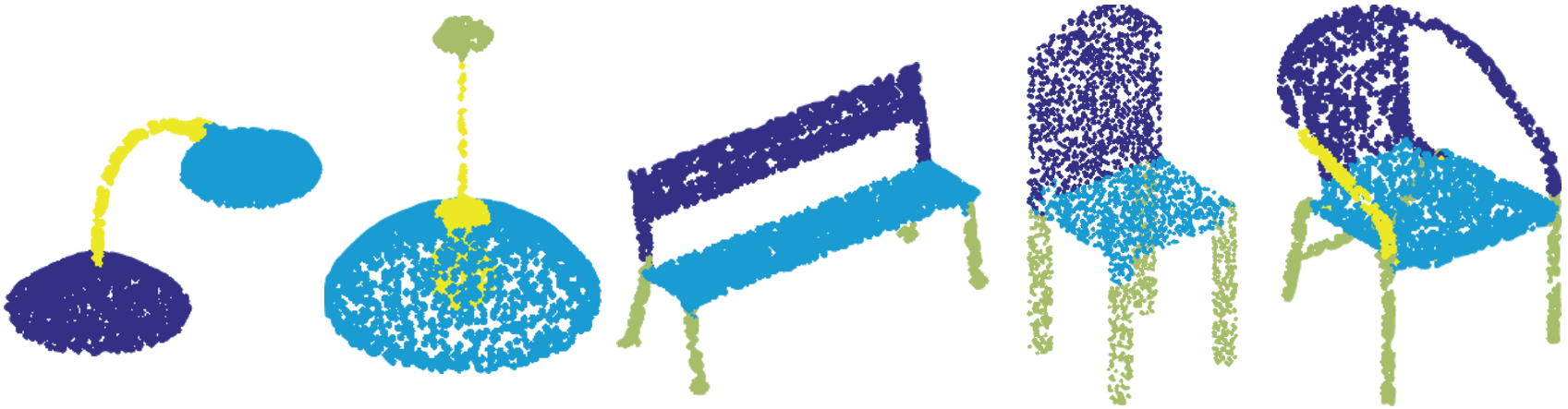
Predicted



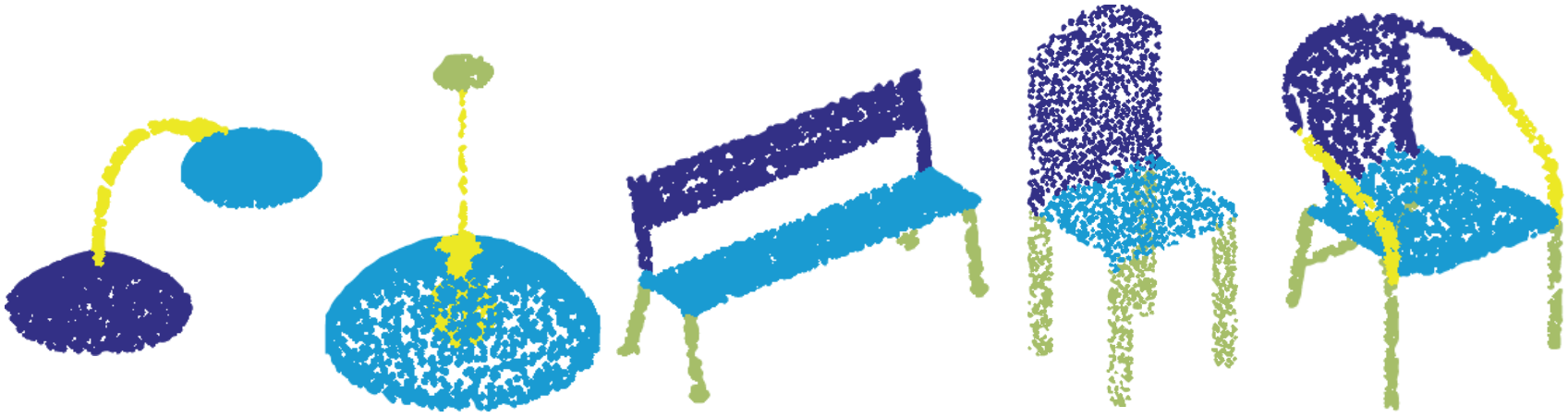
Groundtruth

Yi et al. 2017

Example: shape segmentation with SpecTN



Predicted



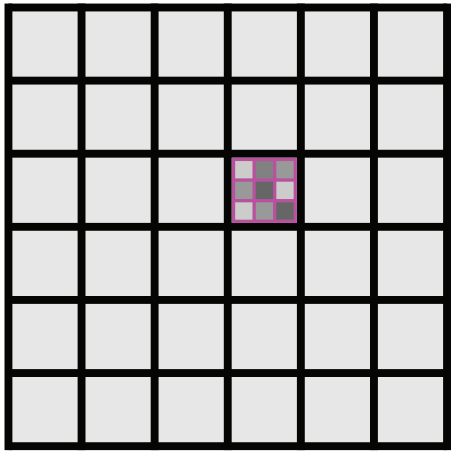
Groundtruth

Yi et al. 2017

Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain

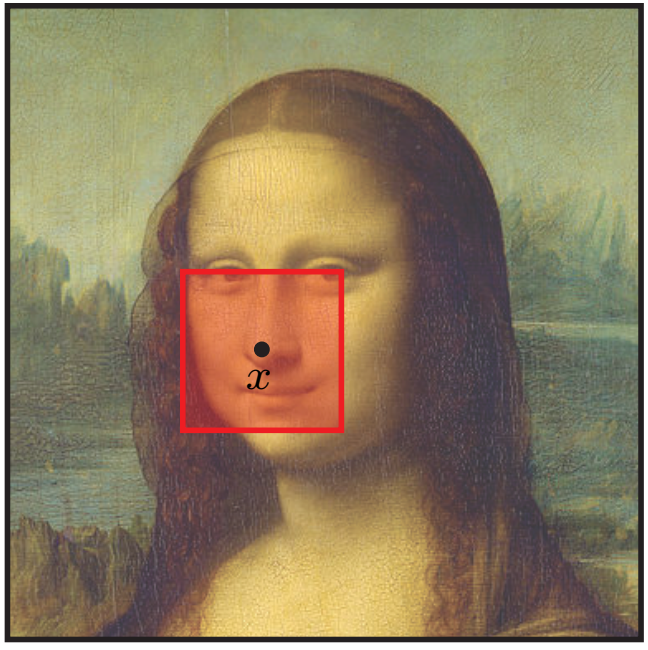


Embedding domain

Convolution

Euclidean	**Non-Euclidean**
Spatial domain	
$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x-x')dx'$	?
Spectral domain	
$\widehat{(f \star g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$	$\widehat{(f \star g)}_k = \langle f, \phi_k \rangle_{L^2(\mathcal{X})} \langle g, \phi_k \rangle_{L^2(\mathcal{X})}$
'Convolution Theorem'	

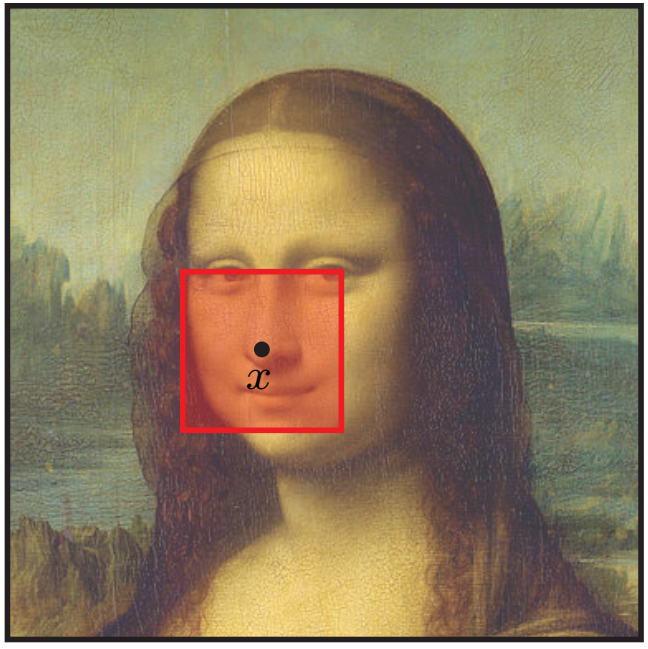
Patch operator



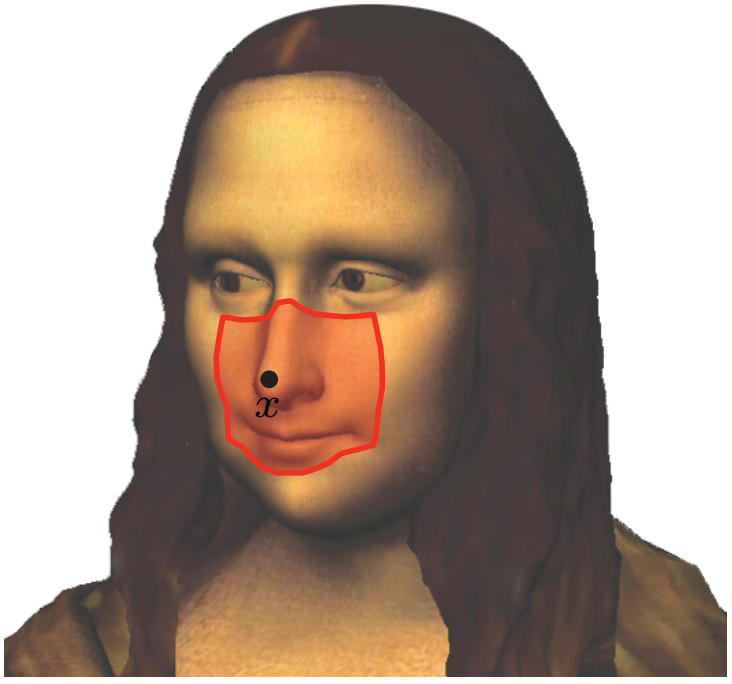
Image

Boscaini et al. 2015

Patch operator



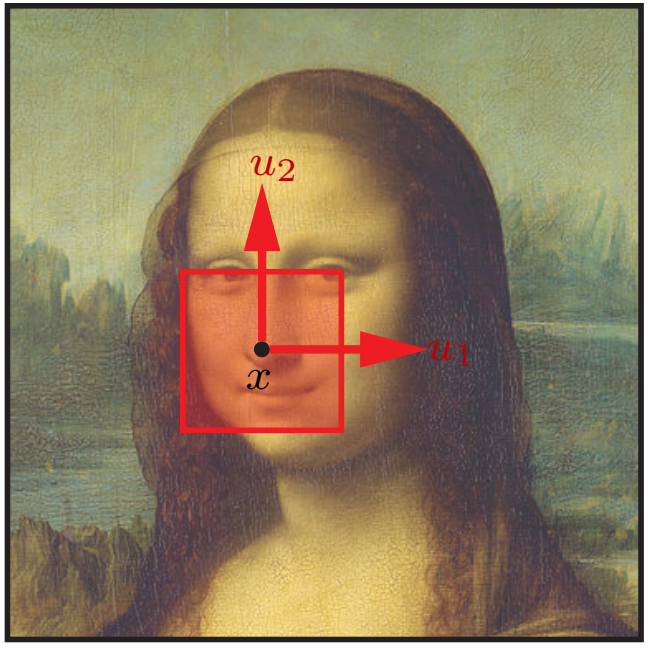
Image



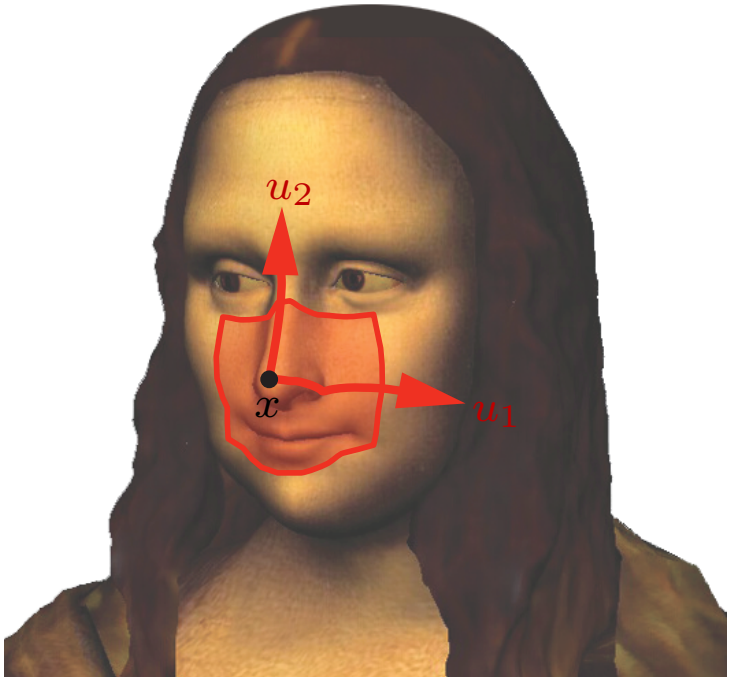
Manifold

Boscaini et al. 2015

Patch operator



Image

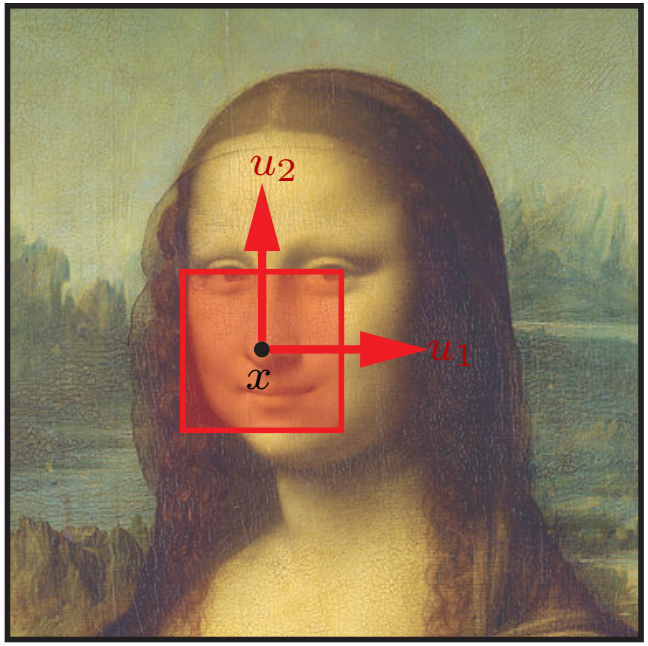


Manifold

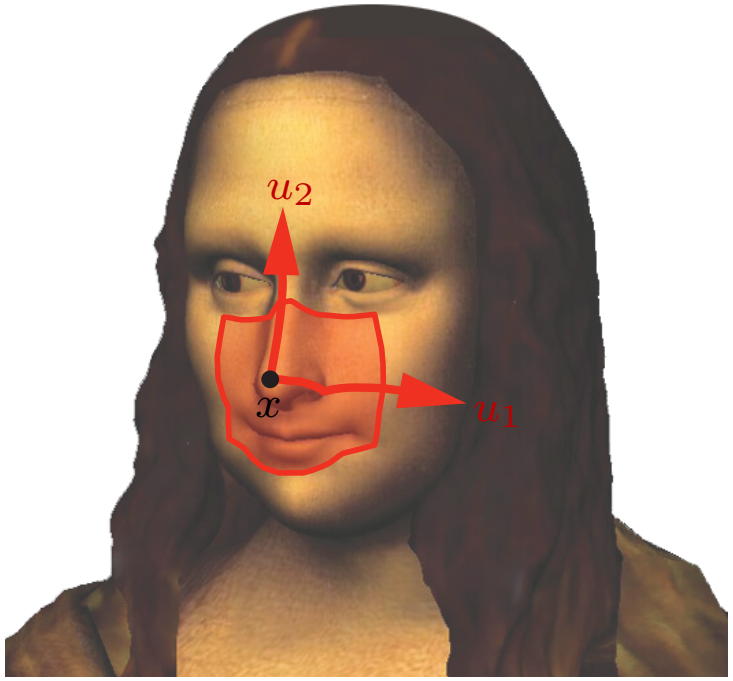
- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \to [0,1]^2$

Boscaini et al. 2015

Patch operator



Image



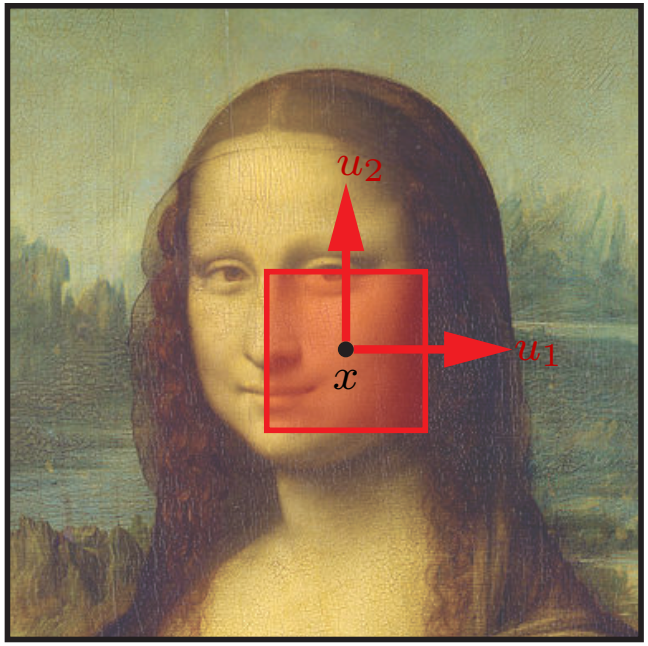
Manifold

- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \to [0,1]^2$
- Patch operator $\mathcal{D} : L^2(\mathcal{X}) \to L^2([0,1]^2)$ mapping f around x

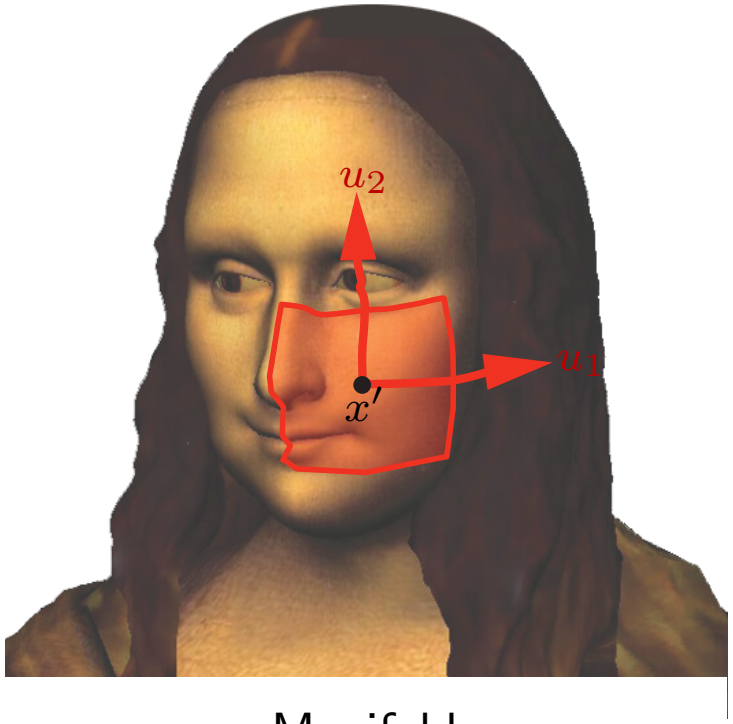
$$(\mathcal{D}(x)f)(\mathbf{u}) = (f \circ \varsigma_x^{-1})(\mathbf{u})$$

Boscaini et al. 2015

Patch operator



Image



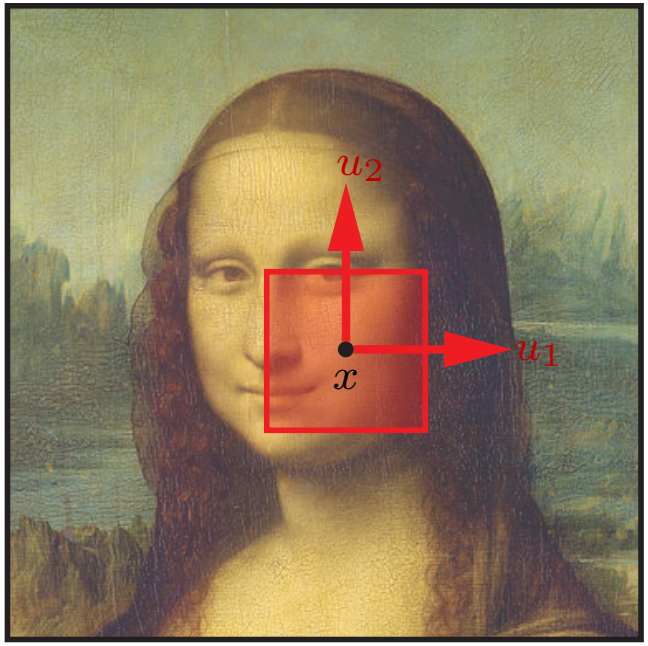
Manifold

- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \to [0,1]^2$
- Patch operator $\mathcal{D} : L^2(\mathcal{X}) \to L^2([0,1]^2)$ mapping f around x

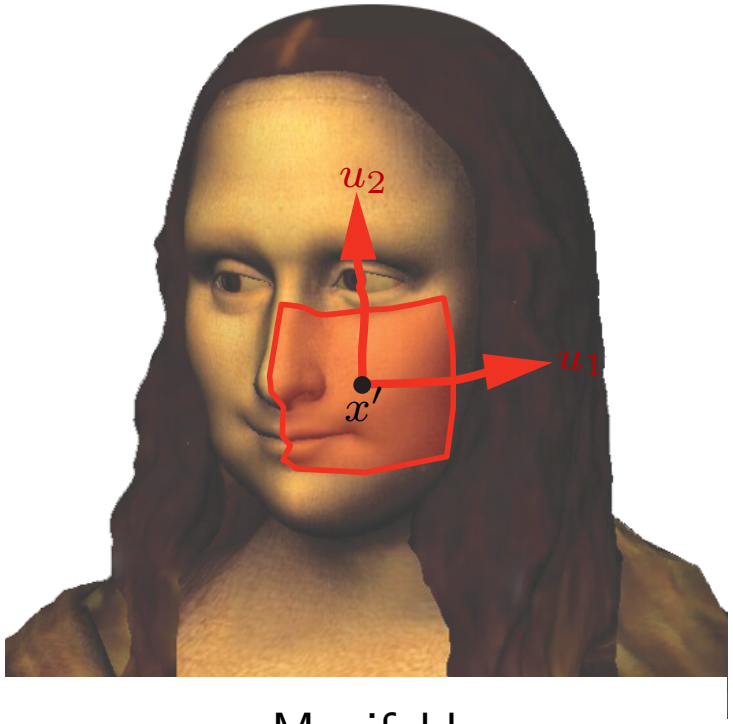
$$(\mathcal{D}(x)f)(\mathbf{u}) = (f \circ \varsigma_x^{-1})(\mathbf{u})$$

Boscaini et al. 2015

Patch operator



Image



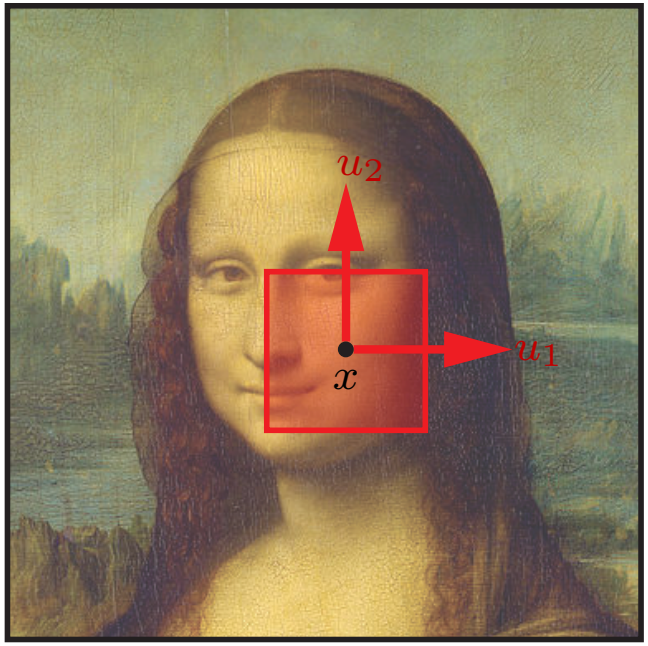
Manifold

- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \to [0,1]^2$
- Patch operator applying weighting function $w_{\mathbf{u}}(x,x') = \delta_{\varsigma_x^{-1}(\mathbf{u})}(x')$

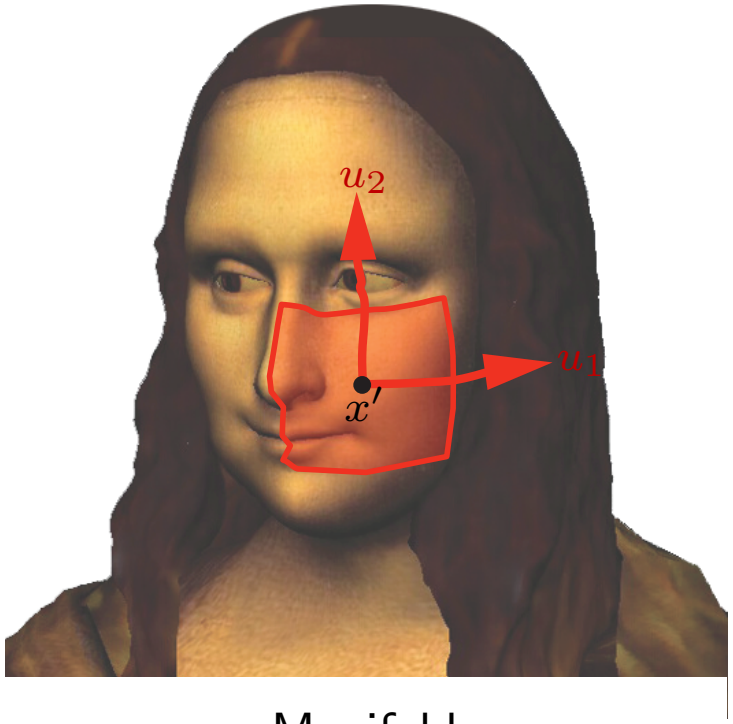
$$(\mathcal{D}(x)f)(\mathbf{u}) = \langle f, w_{\mathbf{u}}(x,x') \rangle_{L^2(\mathcal{X})}$$

Boscaini et al. 2015

Patch operator



Image



Manifold

- Local system of coordinates: bijection $\varsigma_x : B_{\rho_0}(x) \to [0,1]^2$
- Patch operator applying weighting function $w_{\mathbf{u}}(x,x') = \delta_{\varsigma_x^{-1}(\mathbf{u})}(x')$

$$(\mathcal{D}(x)f)(\mathbf{u}) = \langle f, w_{\mathbf{u}}(x,x') \rangle_{L^2(\mathcal{X})}$$

In Euclidean case, $w(x,x')$ is shift-invariant

Boscaini et al. 2015

Spatial convolution

Spatial convolution of $f \in L^2(\mathcal{X})$ with continuous filter $g \in L^2([0,1]^2)$

$$(f \star g)(x) \quad = \quad \int_{[0,1]^2} g(\mathbf{u})(\mathcal{D}(x)f)(\mathbf{u})\, d\mathbf{u}$$

Spatial convolution

Spatial convolution of $f \in L^2(\mathcal{X})$ with discrete filter $g = (g_1, \ldots, g_J)$

$$(f \star g)(x) \quad = \quad \sum_{j=1}^{J} g_j (\mathcal{D}(x) f)_j$$

Geodesic polar patch operator

Patch expressed in local geodesic polar coordinates

$$(\mathcal{D}(x)f)(\rho,\theta) \quad = \quad \int_{\mathcal{X}} \underbrace{w_\rho(x,x')w_\theta(x,x')}_{w_{\rho\theta}(x,x')} f(x')\,dx'$$

Radial weight

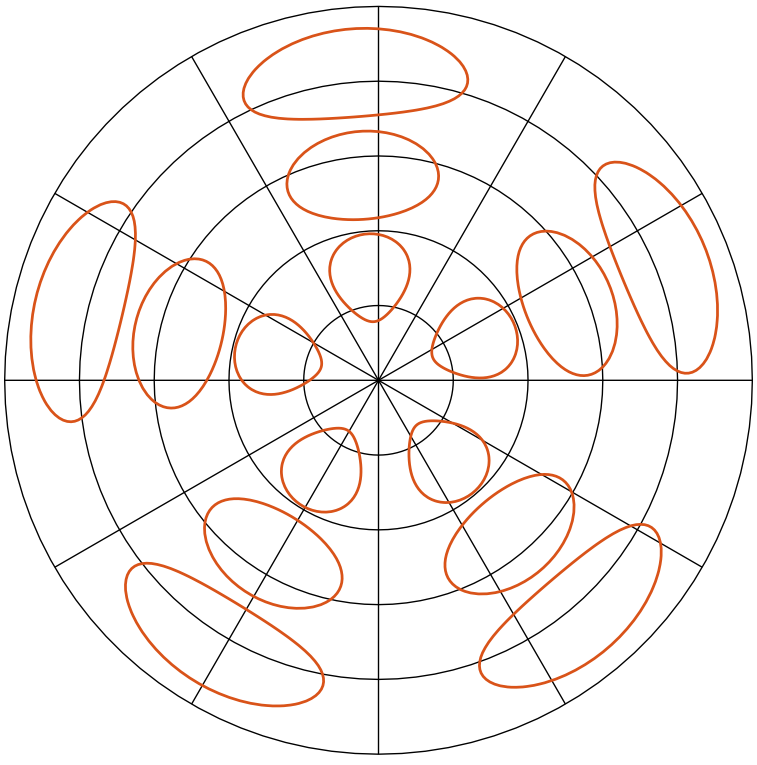
$$w_\rho(x,x') \propto e^{-(d_{\mathcal{X}}(x,x')-\rho)^2/\sigma_\rho^2}$$

Angular weight

$$w_\theta(x,x') \propto e^{-d^2_{\mathcal{X}}(\Gamma_\theta(x),x')/\sigma_\theta^2}$$

Kokkinos et al. 2012; Boscaini et al. 2015

Geodesic polar patch operator construction



Weighting functions of the geodesic polar patch operator shown in (ρ, θ) coordinates (contours mark the $\frac{1}{2}$-level set)

Masci et al. 2015

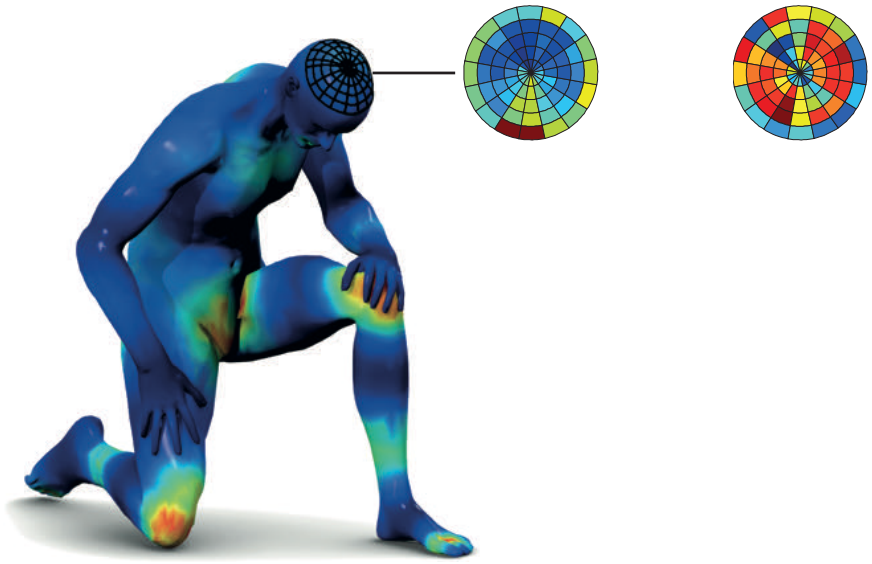
Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} (\mathcal{D}(x) f)(\rho, \theta)\ g(\theta, \rho)\ d\rho d\theta$$

Masci et al. 2015

Geodesic convolution

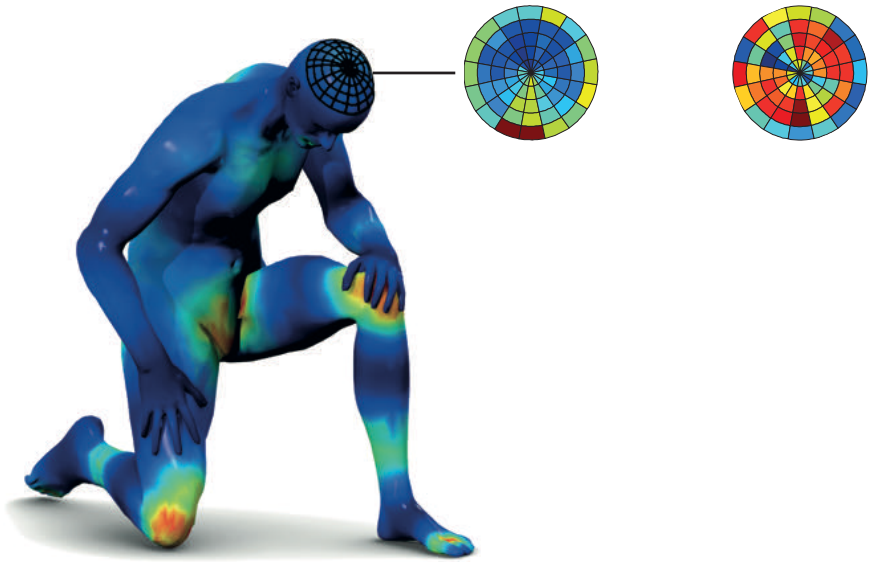
$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta,\rho)}\ d\rho d\theta$$



Masci et al. 2015

Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta+\Delta\theta,\rho)}\ d\rho d\theta$$

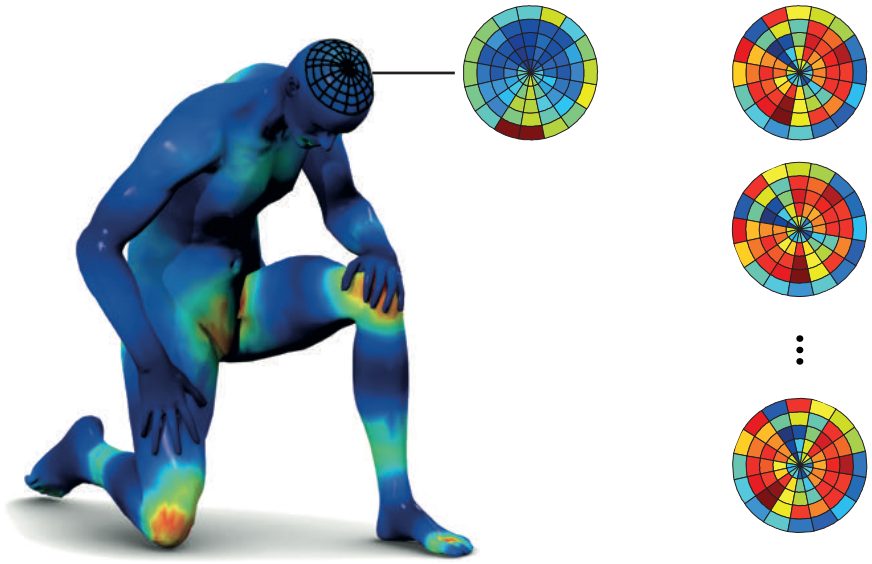


Angular coordinate origin is arbitrary = rotation ambiguity!

Masci et al. 2015

Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta+\Delta\theta,\rho)}\ d\rho d\theta$$



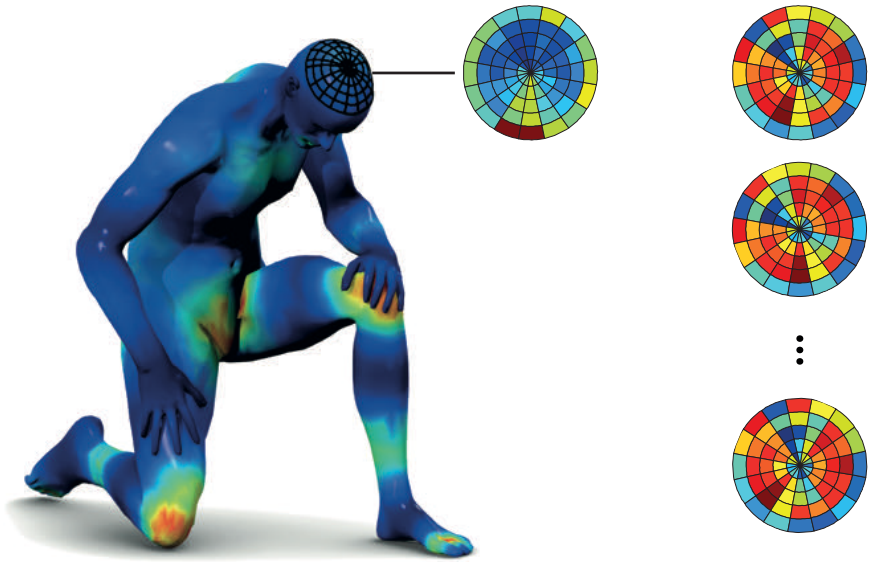
Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector

Masci et al. 2015

Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta+\Delta\theta,\rho)}\ d\rho d\theta$$



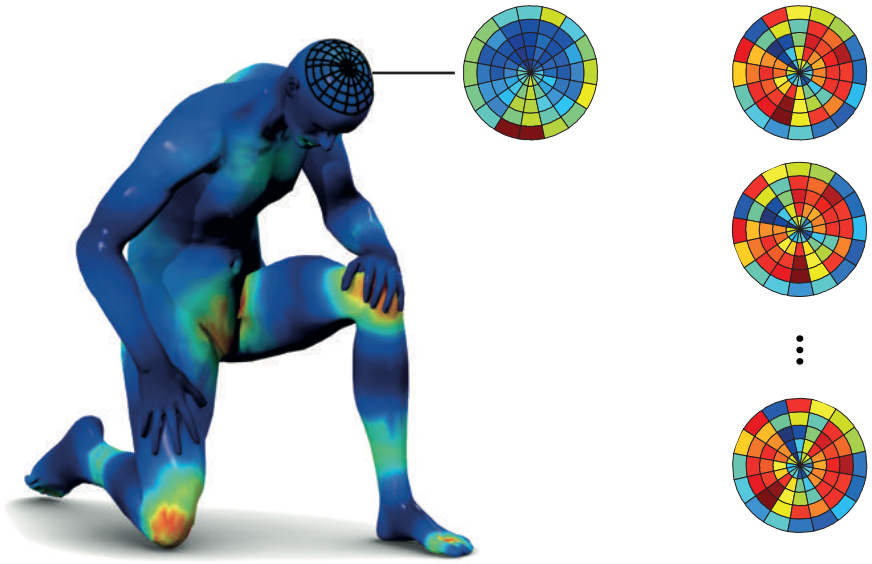
Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform w.r.t. θ

Masci et al. 2015

Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta+\Delta\theta,\rho)}\ d\rho d\theta$$



Angular coordinate origin is arbitrary = rotation ambiguity!

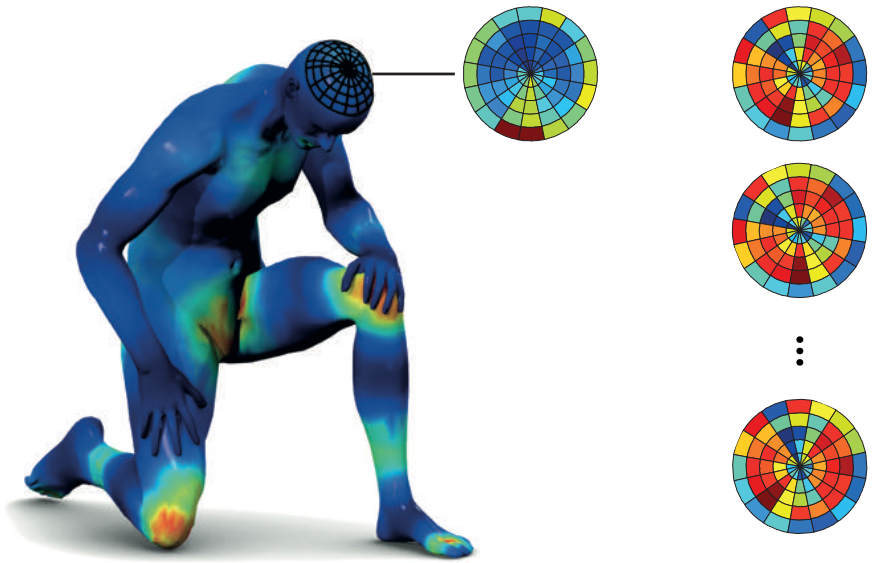
- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform w.r.t. θ

$$(\mathcal{D}(x)f)(\rho,\theta) \quad \xleftrightarrow{\mathcal{F}_\theta} \quad \widehat{(\mathcal{D}(x)f)}(\rho,\omega)$$

Masci et al. 2015

Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta+\Delta\theta,\rho)}\ d\rho d\theta$$



Angular coordinate origin is arbitrary = rotation ambiguity!

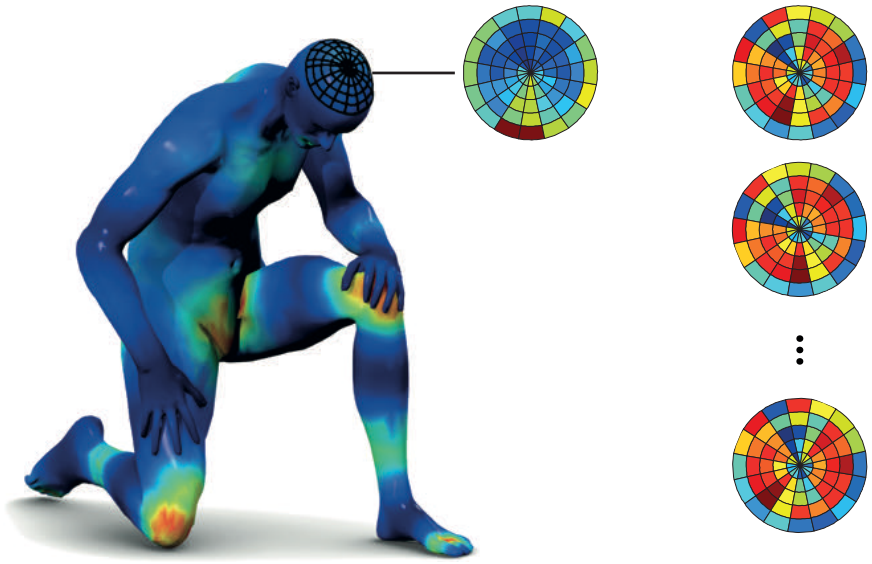
- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform w.r.t. θ

$$(\mathcal{D}(x)f)(\rho,\theta+\Delta\theta) \quad \xleftrightarrow{\mathcal{F}_\theta} \quad e^{-i\omega\theta}\widehat{(\mathcal{D}(x)f)}(\rho,\omega)$$

Masci et al. 2015

Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta+\Delta\theta,\rho)}\ d\rho d\theta$$



Angular coordinate origin is arbitrary = rotation ambiguity!

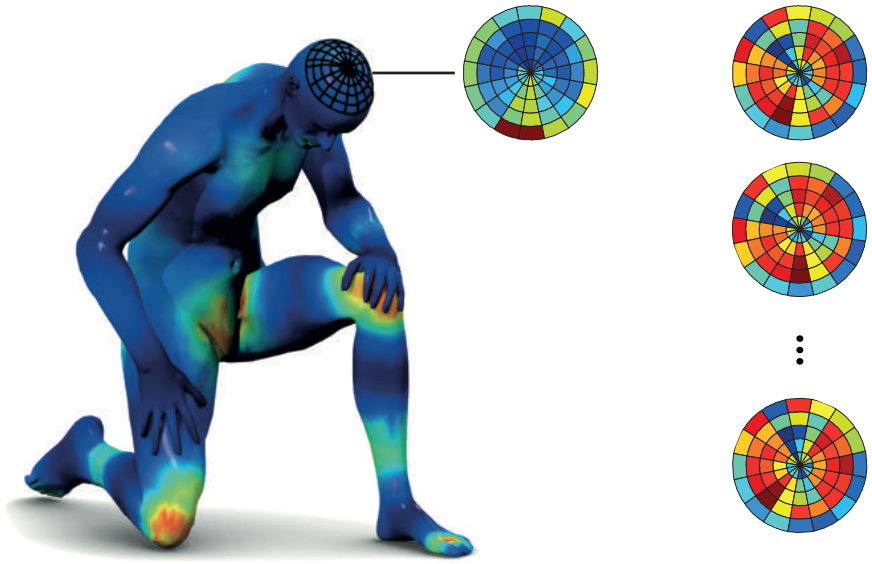
- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform magnitude w.r.t. θ

$$(\mathcal{D}(x)f)(\rho,\theta+\Delta\theta) \quad \xleftrightarrow{\mathcal{F}_\theta} \quad \left|e^{-i\omega\theta}\widehat{(\mathcal{D}(x)f)}(\rho,\omega)\right|$$

Masci et al. 2015

Geodesic convolution

$$(f \star g)(x) \quad = \quad \int_0^{\rho_0} \int_0^{2\pi} \underbrace{(\mathcal{D}(x)f)(\rho,\theta)}\ \underbrace{g(\theta+\Delta\theta,\rho)}\ d\rho d\theta$$

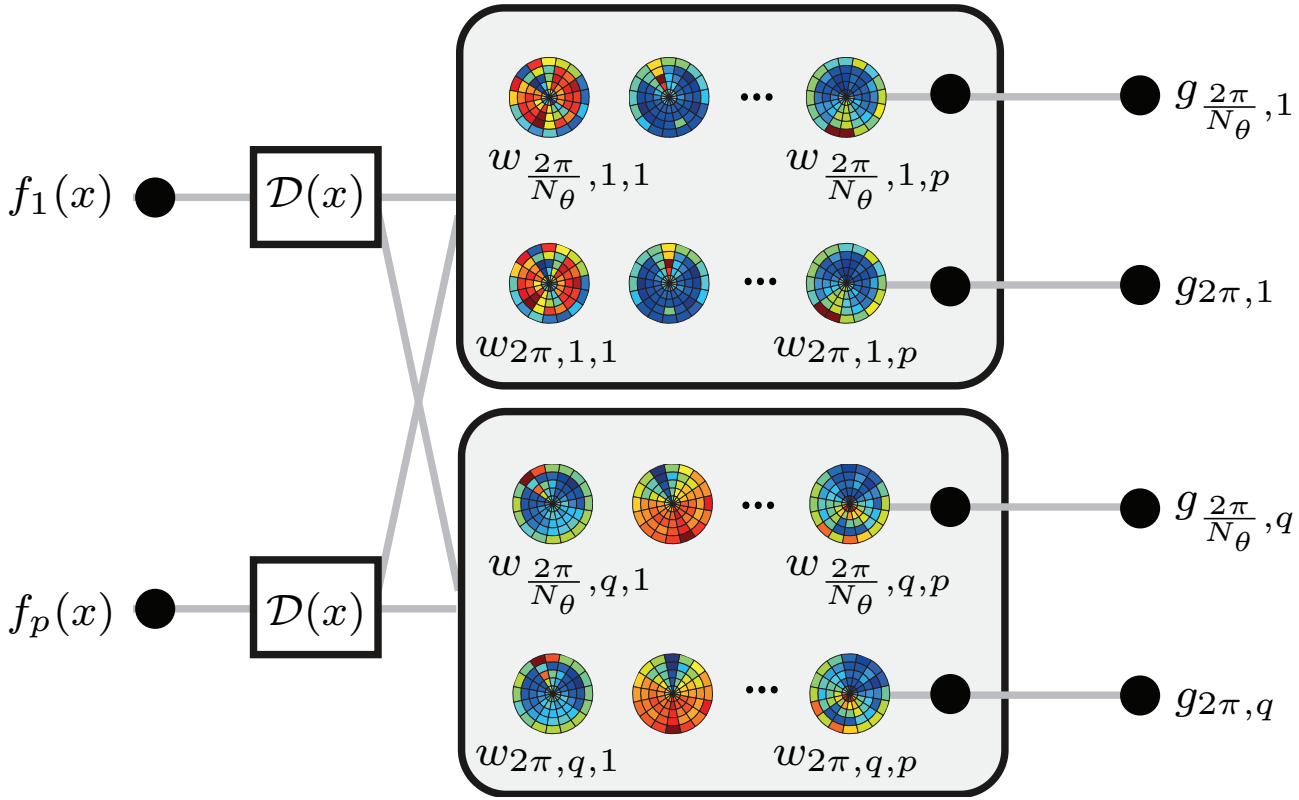


Angular coordinate origin is arbitrary = rotation ambiguity!

- Select reference direction, e.g. maximum curvature vector
- Take Fourier transform magnitude w.r.t. θ
- Keep all possible rotations

Masci et al. 2015

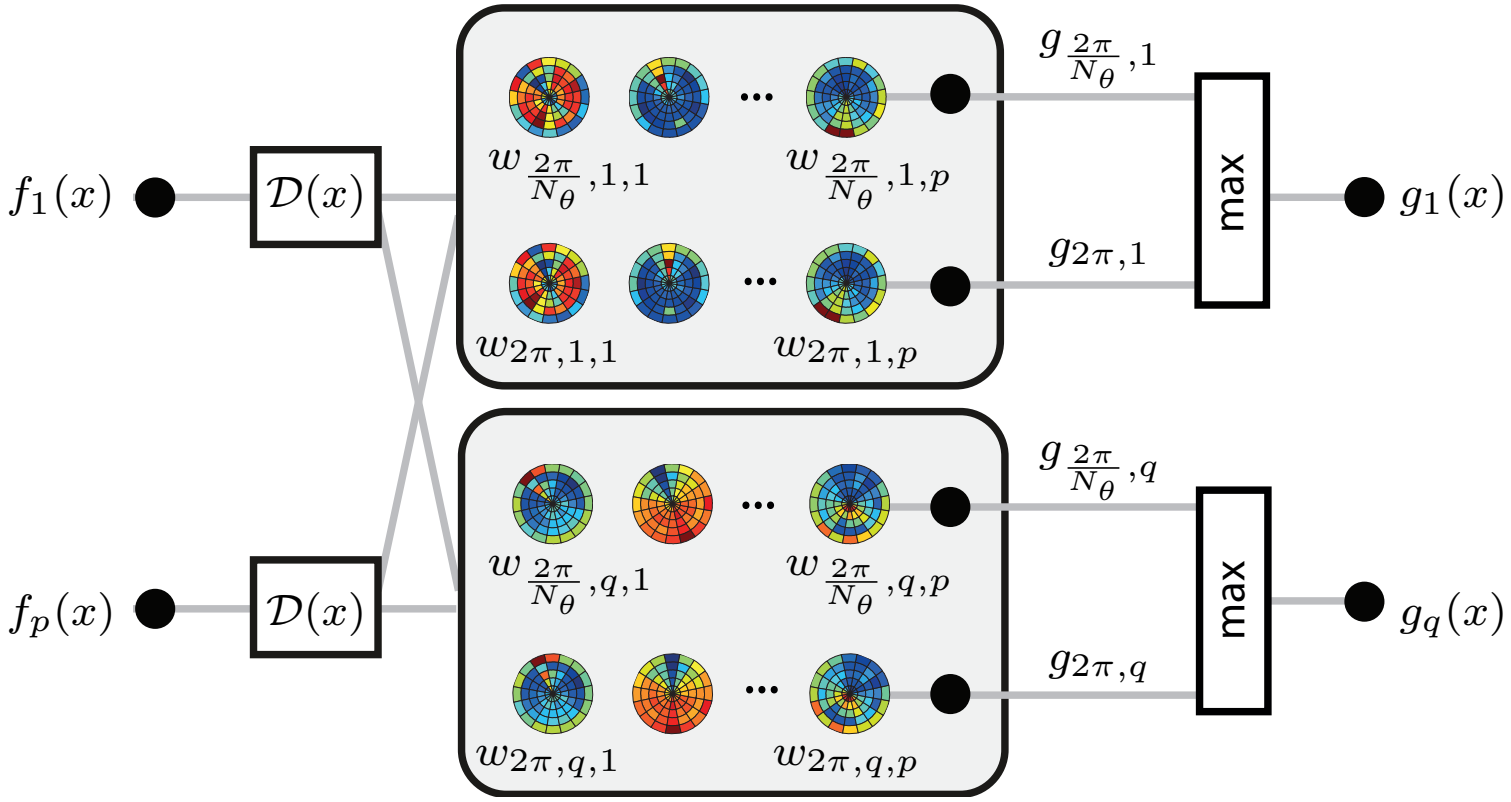
Geodesic convolution layer



Conv. layer $$g_{\Delta\theta,l}(x) = \xi\left(\sum_{l'=1}^{p}(f_{l'} \star w_{\Delta\theta,l,l'})(x)\right) \quad \begin{array}{l} l = 1,\dots,q \\ l' = 1,\dots,p \\ \Delta\theta = \frac{2\pi}{N_\theta},\dots,2\pi \end{array}$$

Masci et al. 2015

Geodesic convolution layer



Conv. layer

$$g_{\Delta\theta,l}(x) = \xi\left(\sum_{l'=1}^{p}(f_{l'} \star w_{\Delta\theta,l,l'})(x)\right) \quad \begin{array}{l} l = 1,\dots,q \\ l' = 1,\dots,p \\ \Delta\theta = \frac{2\pi}{N_\theta},\dots,2\pi \end{array}$$

Angular max pooling

$$g_l(x) = \max_{\Delta\theta} g_{\Delta\theta,l}(x)$$

Masci et al. 2015

Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

$$g_l(x) = \max_{\Delta\theta} \xi\Bigg(\sum_{l'=1}^{p} \underbrace{\int_0^{\rho_0} \int_0^{2\pi} w_{l,l'}(\rho, \theta + \Delta\theta)(\mathcal{D}(x) f_{l'})(\rho, \theta)\, d\rho d\theta}_{(f_{l'} \star w_{\Delta\theta, l, l'})(x)} \Bigg)$$

$$l = 1, \ldots, q$$
$$l' = 1, \ldots, p$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J})$ are spatial filter coefficients

Boscaini et al. 2015

Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

$$g_l(x) = \max_{\Delta\theta} \xi\Bigg(\sum_{l'=1}^{p} \underbrace{\int_0^{\rho_0} \int_0^{2\pi} w_{l,l'}(\rho, \theta + \Delta\theta)(\mathcal{D}(x) f_{l'})(\rho, \theta)\, d\rho d\theta}_{(f_{l'} \star w_{\Delta\theta, l, l'})(x)} \Bigg)$$

$$l = 1, \ldots, q$$
$$l' = 1, \ldots, p$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J})$ are spatial filter coefficients

☺ Directional filters

Boscaini et al. 2015

Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

$$g_l(x) = \max_{\Delta\theta} \xi\Bigg(\sum_{l'=1}^{p} \underbrace{\int_0^{\rho_0} \int_0^{2\pi} w_{l,l'}(\rho, \theta + \Delta\theta)(\mathcal{D}(x) f_{l'})(\rho, \theta)\, d\rho d\theta}_{(f_{l'} \star w_{\Delta\theta, l, l'})(x)} \Bigg)$$

$$l = 1, \ldots, q$$
$$l' = 1, \ldots, p$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J})$ are spatial filter coefficients

☺ Directional filters

☺ Spatially-localized filters

Boscaini et al. 2015

Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

$$g_l(x) = \max_{\Delta\theta} \xi\Bigg(\sum_{l'=1}^{p} \underbrace{\int_0^{\rho_0} \int_0^{2\pi} w_{l,l'}(\rho, \theta + \Delta\theta)(\mathcal{D}(x) f_{l'})(\rho, \theta)\, d\rho d\theta}_{(f_{l'} \star w_{\Delta\theta,l,l'})(x)} \Bigg)$$

$$l = 1, \ldots, q$$
$$l' = 1, \ldots, p$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J})$ are spatial filter coefficients

- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ $\mathcal{O}(1)$ parameters per layer

Boscaini et al. 2015

Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

$$g_l(x) = \max_{\Delta\theta} \xi\Bigg(\sum_{l'=1}^{p} \underbrace{\int_0^{\rho_0} \int_0^{2\pi} w_{l,l'}(\rho, \theta + \Delta\theta)(\mathcal{D}(x) f_{l'})(\rho, \theta)\, d\rho d\theta}_{(f_{l'} \star w_{\Delta\theta,l,l'})(x)} \Bigg)$$

$$l = 1, \ldots, q$$
$$l' = 1, \ldots, p$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J})$ are spatial filter coefficients

- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ All operations are local $\Rightarrow$ $\mathcal{O}(n)$ computational complexity

Boscaini et al. 2015

Geodesic CNN (GCNN)

Convolutional layer expressed in the spatial domain using geodesic polar patch operator + angular max pooling to solve rotational ambiguity

$$g_l(x) = \max_{\Delta\theta} \xi\Bigg(\sum_{l'=1}^{p} \underbrace{\int_0^{\rho_0} \int_0^{2\pi} w_{l,l'}(\rho, \theta + \Delta\theta)(\mathcal{D}(x) f_{l'})(\rho, \theta)\, d\rho d\theta}_{(f_{l'} \star w_{\Delta\theta,l,l'})(x)} \Bigg)$$

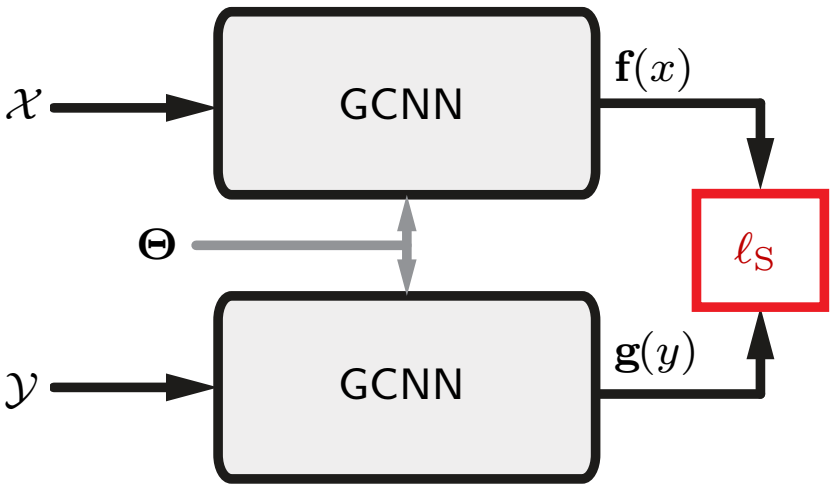
$$l = 1, \ldots, q$$
$$l' = 1, \ldots, p$$

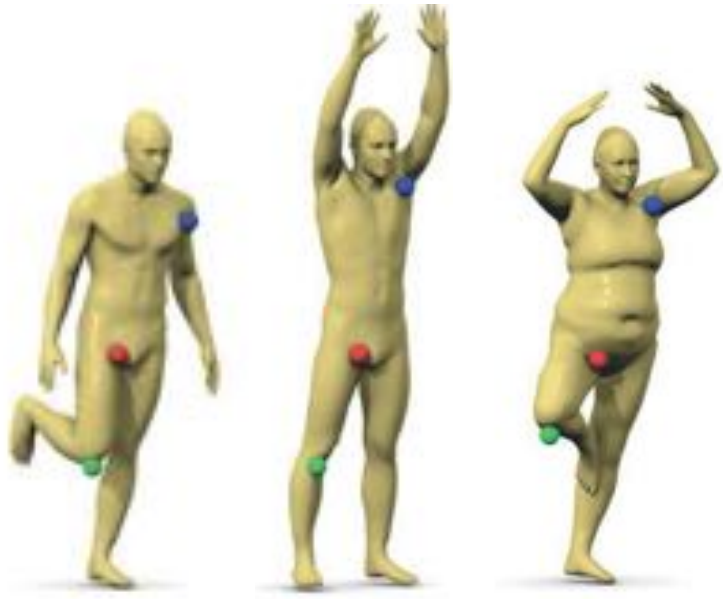
where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \ldots, w_{l,l',J})$ are spatial filter coefficients

- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ All operations are local $\Rightarrow$ $\mathcal{O}(n)$ computational complexity
- ☹ Angular max pooling potentially reduces discriminativity

Boscaini et al. 2015

Example: Learning local descriptors with GCNN





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

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

Poitwise feature cost

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

Boscaini et al. 2015

Example: 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: Bronstein et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

Example: 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)

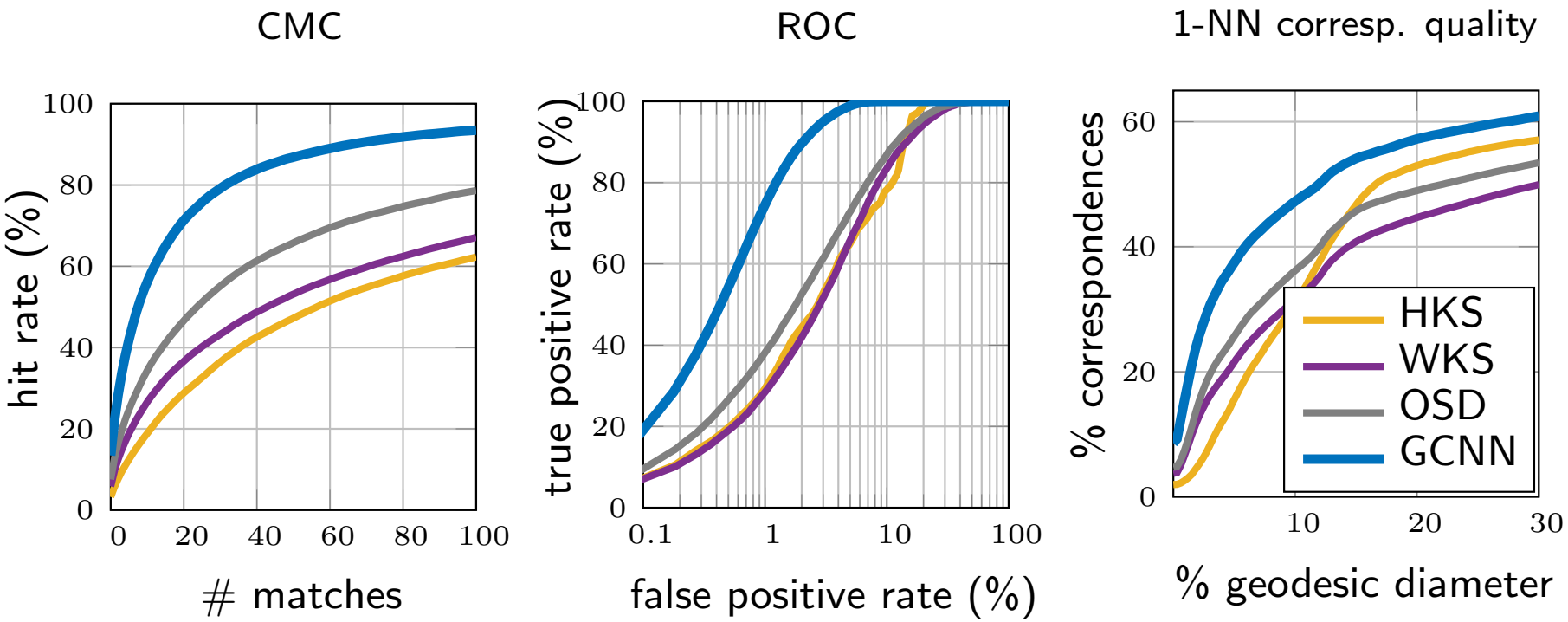
Descriptor: Aubry, Schlickewei, Cremers 2011 (WKS); data: Bronstein et al. 2008 (TOSCA); Anguelov et al. 2005 (SCAPE); Bogo et al. 2014 (FAUST)

Example: descriptor learning 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: Bronstein 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, Bronstein 2014 (OSD); Masci et al. 2015 (GCNN);
data: Bogo et al. 2014 (FAUST); benchmark: Kim et al. 2011

Homogeneous diffusion

$$f_t(x) = -c\Delta f(x)$$

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

Anisotropic diffusion

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

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

diffusion

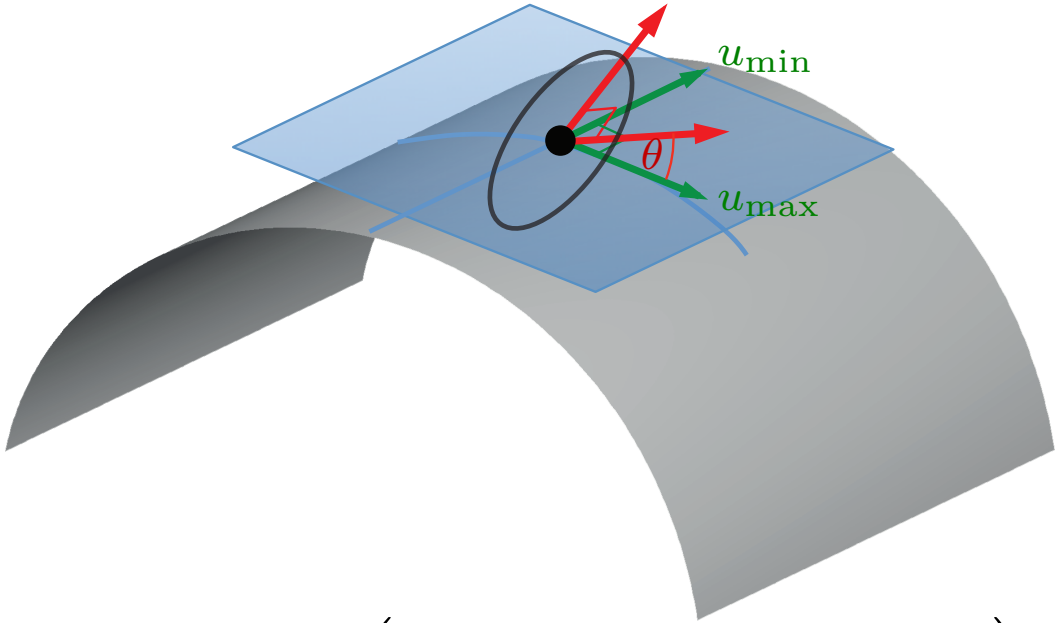
$$f_t(x) = -\mathrm{div}(\mathbf{A}(x)\nabla f(x))$$

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

Anisotropic diffusion

Anisotropic

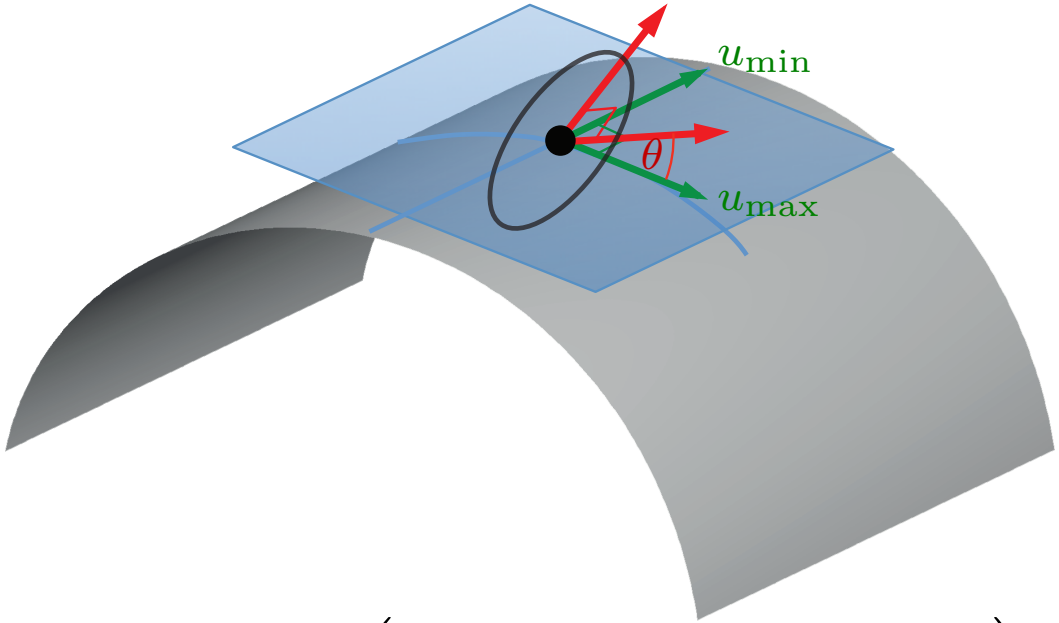
Anisotropic diffusion on manifolds



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

Andreux et al. 2014; Boscaini et al. 2016

Anisotropic diffusion on manifolds



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

- 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

Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

$$(\mathcal{D}(x)f)_j = \int_{\mathcal{X}} f(x')h_{\alpha_i,\theta_i,t_i}(x,x')dx' \qquad j = 1,\dots,J$$

for a discrete set of angles/scales/anisotropic constants

Boscaini et al. 2016

Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

$$(\mathcal{D}(x)f)_j = \int_{\mathcal{X}} f(x')h_{\alpha_i,\theta_i,t_i}(x,x')dx' \qquad j = 1,\dots,J$$

for a discrete set of angles/scales/anisotropic constants

Boscaini et al. 2016

Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

$$(\mathcal{D}(x)f)_j = \int_{\mathcal{X}} f(x')h_{\alpha_i,\theta_i,t_i}(x,x')dx' \qquad j = 1,\dots,J$$

for a discrete set of angles/scales/anisotropic constants

Convolutional layer expressed in the spatial domain

$$g_l(x) = \xi\left(\sum_{l'=1}^{p} \mathbf{w}_{l,l'}^{\top}\mathcal{D}(x)f\right) \qquad \begin{matrix} l = 1,\dots,q \\ l' = 1,\dots,p \end{matrix}$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1},\dots,w_{l,l',J})$ are spatial filter coefficients

Boscaini et al. 2016

Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

$$(\mathcal{D}(x)f)_j = \int_{\mathcal{X}} f(x')h_{\alpha_i,\theta_i,t_i}(x,x')dx' \qquad j = 1,\dots,J$$

for a discrete set of angles/scales/anisotropic constants

Convolutional layer expressed in the spatial domain

$$g_l(x) = \xi\left(\sum_{l'=1}^{p} \mathbf{w}_{l,l'}^{\top}\mathcal{D}(x)f\right) \qquad \begin{matrix} l = 1,\dots,q \\ l' = 1,\dots,p \end{matrix}$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1},\dots,w_{l,l',J})$ are spatial filter coefficients

- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ $\mathcal{O}(1)$ parameters per layer

Boscaini et al. 2016

Anisotropic CNN (ACNN)

Use anisotropic heat kernels as weighting functions of the patch operator

$$(\mathcal{D}(x)f)_j = \int_{\mathcal{X}} f(x')h_{\alpha_i,\theta_i,t_i}(x,x')dx' \qquad j = 1,\dots,J$$

for a discrete set of angles/scales/anisotropic constants

Convolutional layer expressed in the spatial domain

$$g_l(x) = \xi\left(\sum_{l'=1}^{p} \mathbf{w}_{l,l'}^{\top}\mathcal{D}(x)f\right) \qquad \begin{matrix} l = 1,\dots,q \\ l' = 1,\dots,p \end{matrix}$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1},\dots,w_{l,l',J})$ are spatial filter coefficients

- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ $\mathcal{O}(1)$ parameters per layer
- ☹ Expensive computation of heat kernels for many orientations

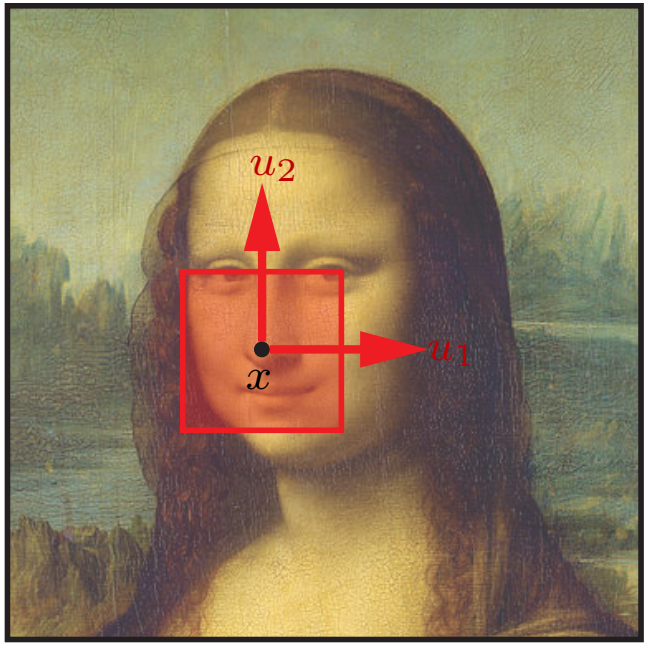
Boscaini et al. 2016

Example: descriptor learning with ACNN

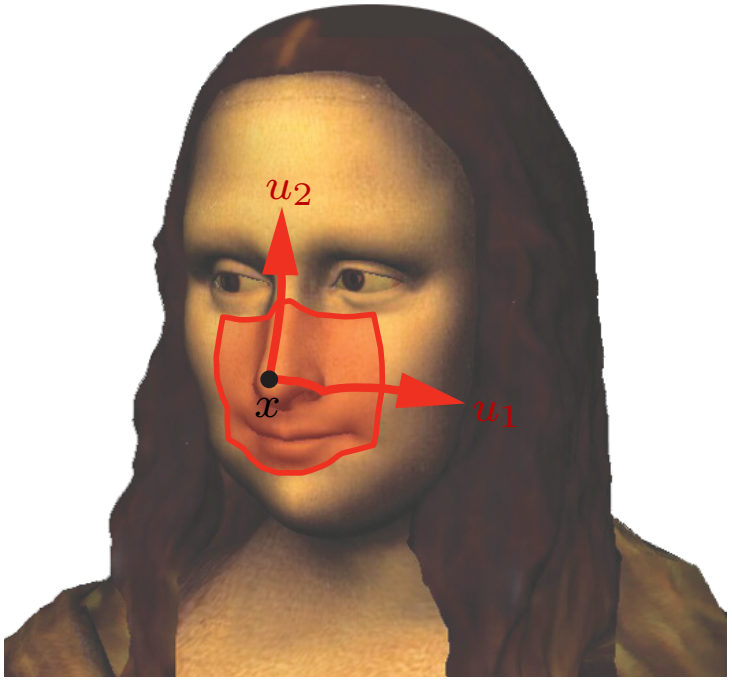
Distance in the space of local ACNN features
(shown is distance from a point on the shoulder marked in white)

Boscaini et al. 2015; Boscaini et al. 2016

Learnable patch operator



Image

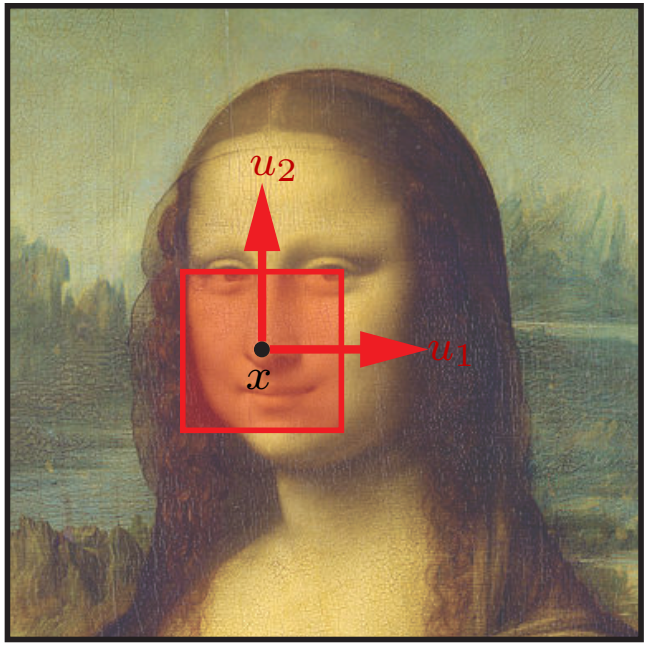


Manifold

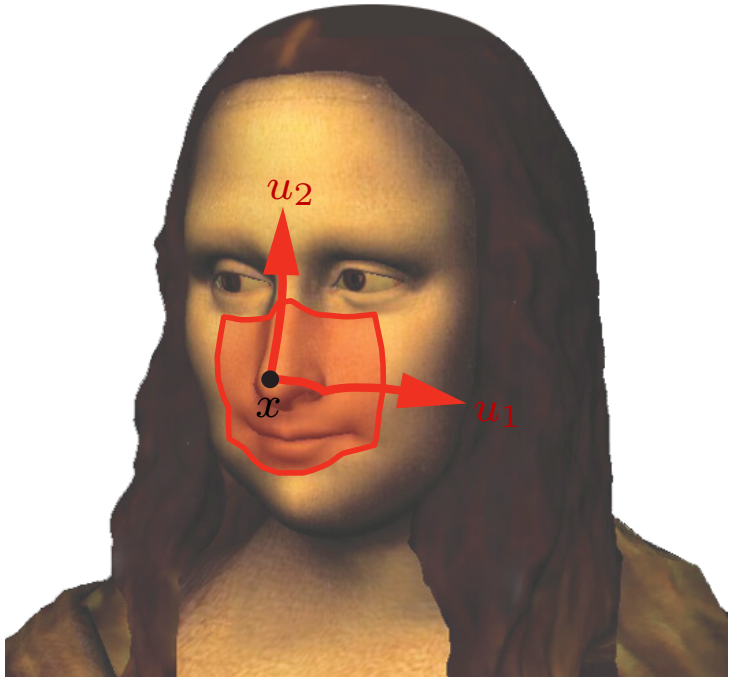
- Local system of coordinates $\mathbf{u}(x, x')$ around point x

Monti et al. 2016

Learnable patch operator



Image

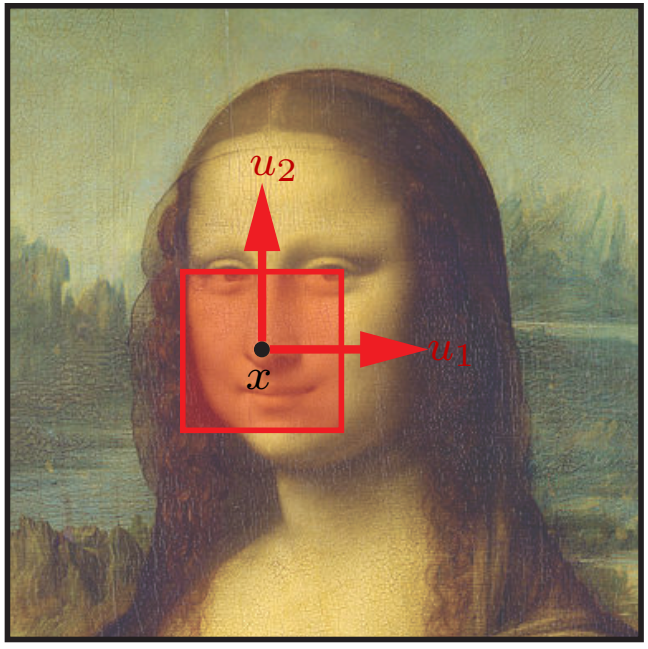


Manifold

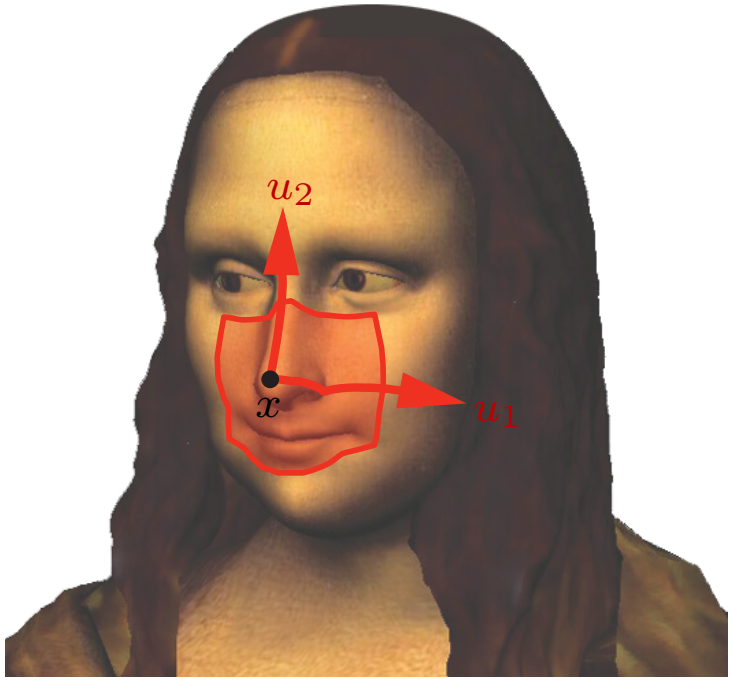
- Local system of coordinates $\mathbf{u}(x, x')$ around point x
- Parametric weighting functions $\mathbf{w}_{\boldsymbol{\Theta}}(\mathbf{u})$

Monti et al. 2016

Learnable patch operator



Image



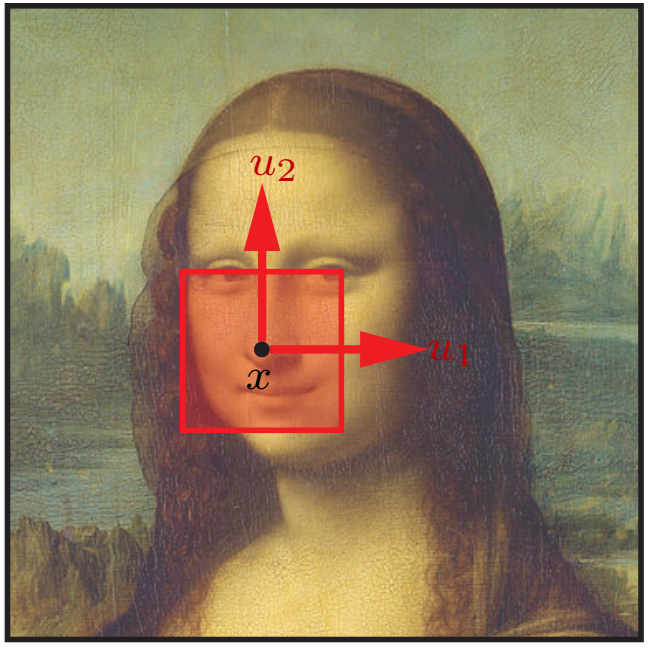
Manifold

- Local system of coordinates $\mathbf{u}(x, x')$ around point x
- Parametric weighting functions $\mathbf{w}_{\boldsymbol{\Theta}}(\mathbf{u})$
- Parametric patch operator applying J such weighting functions

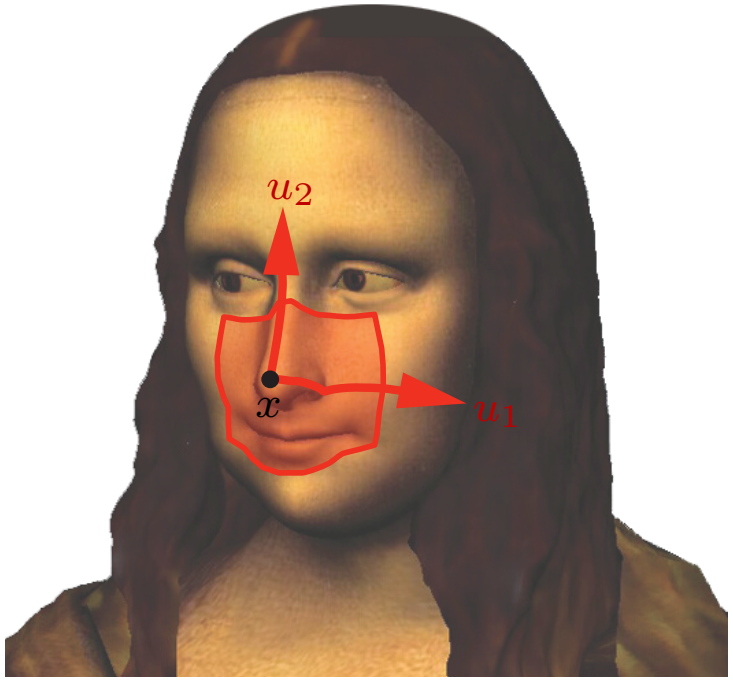
$$(\mathcal{D}_{\boldsymbol{\Theta}_1,\dots,\boldsymbol{\Theta}_J}(x)f)_j = \int_{\mathcal{X}} f(x')w_{\boldsymbol{\Theta}_j}(\mathbf{u}(x,x'))dx' \qquad j = 1,\dots,J$$

Monti et al. 2016

Learnable patch operator



Image



Manifold

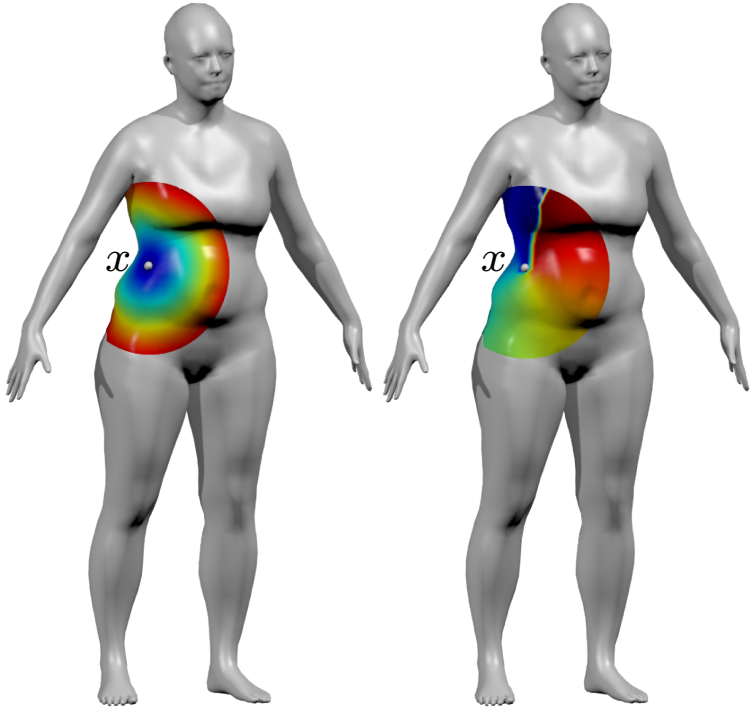
- Local system of coordinates $\mathbf{u}(x, x')$ around point x
- Parametric weighting functions, e.g. $\mathbf{w}_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{u}) = e^{-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})}$
- Parametric patch operator applying J such weighting functions

$$(\mathcal{D}_{\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1,\ldots,\boldsymbol{\mu}_J,\boldsymbol{\Sigma}_J}(x)f)_j = \int_{\mathcal{X}} f(x') w_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}(\mathbf{u}(x,x'))dx' \qquad j = 1,\ldots,J$$

Monti et al. 2016

Learnable patches on manifolds

- Geodesic polar coordinates
  $\mathbf{u}(x, y) = (\rho(x, y), \theta(x, y))$



Monti et al. 2016

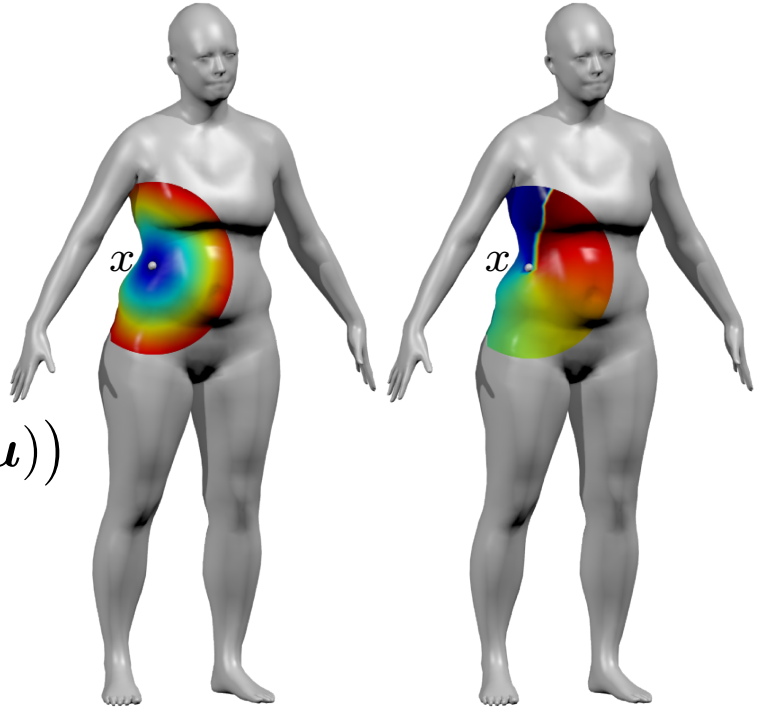
Learnable patches on manifolds

- Geodesic polar coordinates
  $\mathbf{u}(x, y) = (\rho(x, y), \theta(x, y))$

- Gaussian weighting functions

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

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



Monti et al. 2016

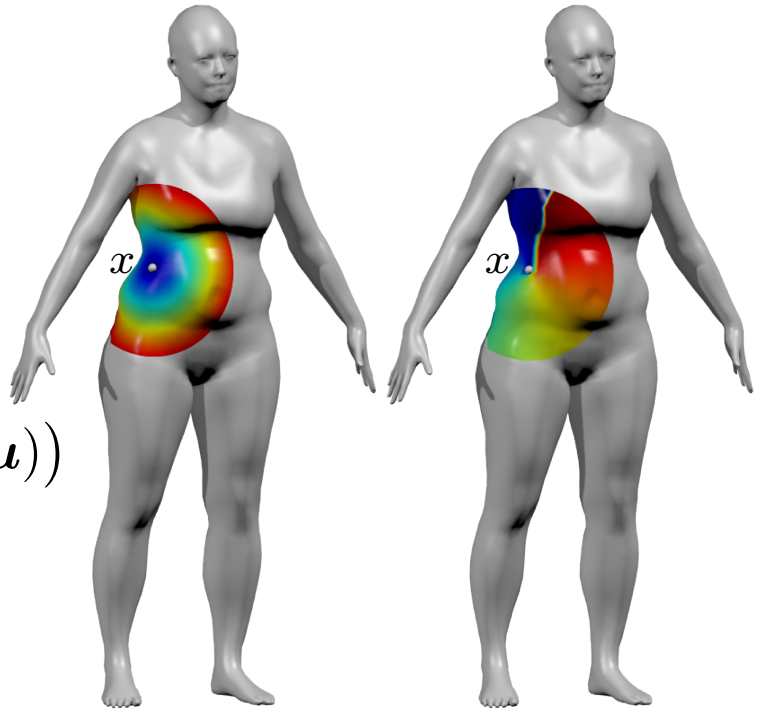
Learnable patches on manifolds

- Geodesic polar coordinates
  $\mathbf{u}(x,y) = (\rho(x,y), \theta(x,y))$

- Gaussian weighting functions

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

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



Spatial convolution

$$(f \star g)(x) = \sum_{j=1}^{J} g_j \int_{\mathcal{X}} w_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}(\mathbf{u}(x,x'))\, f(x')\, dx'$$

Monti et al. 2016

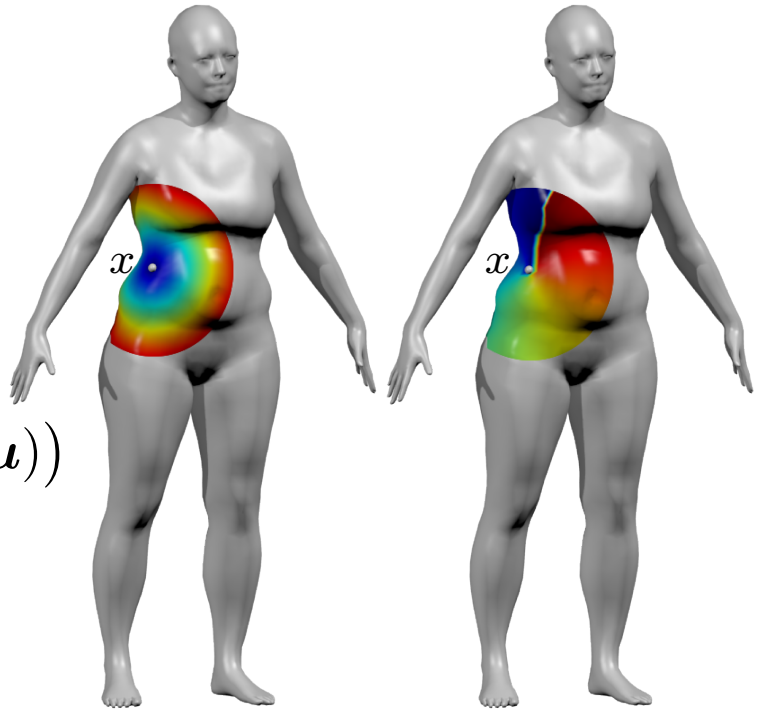
Learnable patches on manifolds

- Geodesic polar coordinates
  $\mathbf{u}(x,y) = (\rho(x,y), \theta(x,y))$

- Gaussian weighting functions

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

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



Spatial convolution

$$(f \star g)(x) = \int_{\mathcal{X}} \sum_{j=1}^{J} g_j w_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}(\mathbf{u}(x,x'))\, f(x')\, dx'$$

Monti et al. 2016

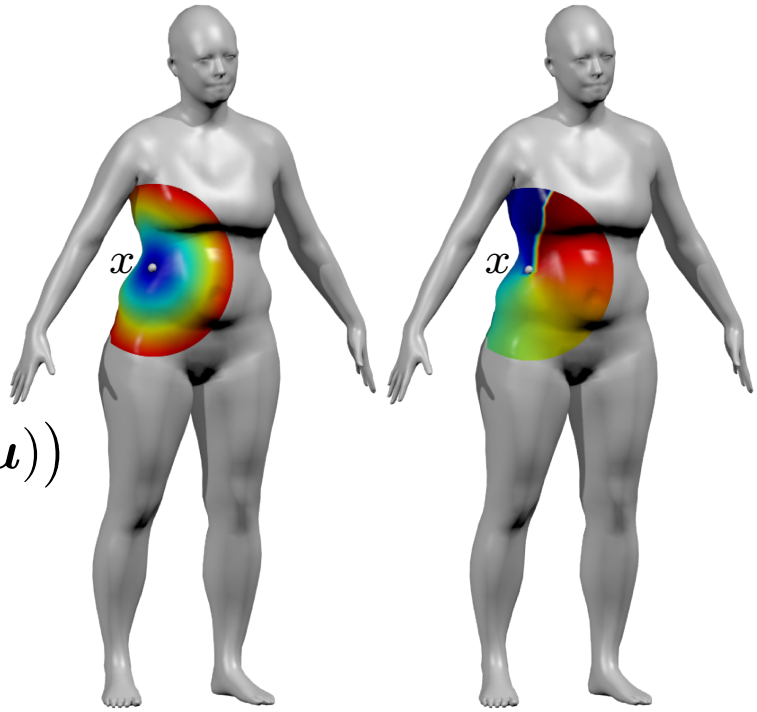
Learnable patches on manifolds

- Geodesic polar coordinates
  $\mathbf{u}(x,y) = (\rho(x,y), \theta(x,y))$

- Gaussian weighting functions

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

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



Spatial convolution

$$(f \star g)(x) = \int_{\mathcal{X}} \underbrace{\sum_{j=1}^{J} g_j w_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}(\mathbf{u}(x,x'))}_{\text{Gaussian mixture } g(\mathbf{u}(x,x'))} f(x')\, dx'$$

Monti et al. 2016

Mixture Model Networks (MoNet)

Convolutional layer expressed in the spatial domain using a learnable patch operator

$$g_l(x) = \xi\left(\sum_{l'=1}^{p} \mathbf{w}_{l,l'}^{\top} \mathcal{D}_{\boldsymbol{\Theta}}(x) f\right) \qquad \begin{matrix} l = 1, \dots, q \\ l' = 1, \dots, p \end{matrix}$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \dots, w_{l,l',J})$ are spatial filter coefficients and $\boldsymbol{\Theta} = (\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\mu}_J, \boldsymbol{\Sigma}_J)$ are the patch parameters

Monti et al. 2016

Mixture Model Networks (MoNet)

Convolutional layer expressed in the spatial domain using a learnable patch operator

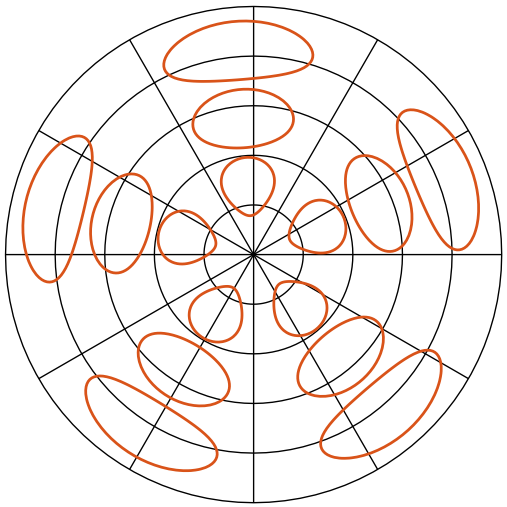
$$g_l(x) = \xi\left(\sum_{l'=1}^{p} \mathbf{w}_{l,l'}^{\top} \mathcal{D}_{\boldsymbol{\Theta}}(x) f\right) \qquad \begin{array}{l} l = 1, \dots, q \\ l' = 1, \dots, p \end{array}$$

where $\mathbf{w}_{l,l'} = (w_{l,l',1}, \dots, w_{l,l',J})$ are spatial filter coefficients and $\boldsymbol{\Theta} = (\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\mu}_J, \boldsymbol{\Sigma}_J)$ are the patch parameters

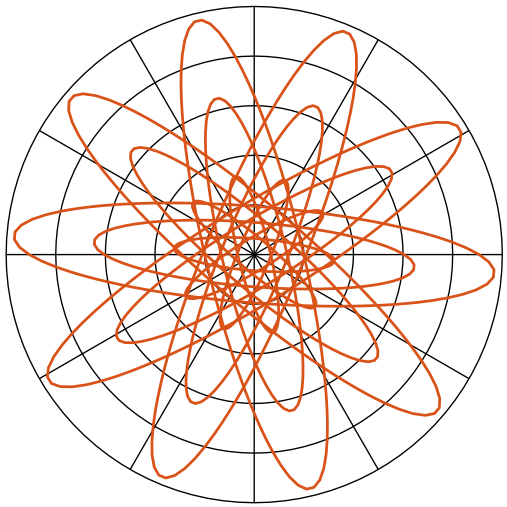
- ☺ Directional filters
- ☺ Spatially-localized filters
- ☺ Learnable patch operator
- ☺ $\mathcal{O}(1)$ parameters per layer

Monti et al. 2016

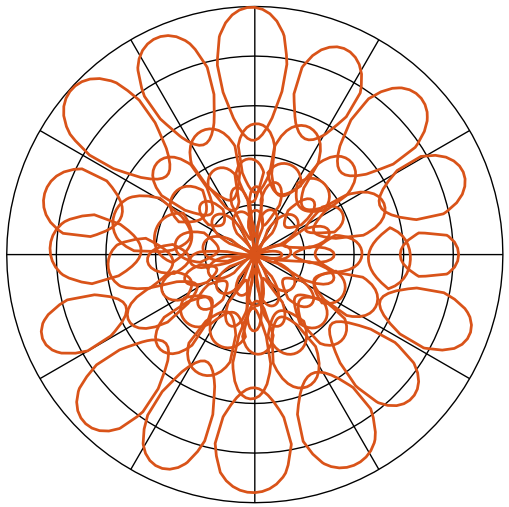
Patch operator weight functions



GCNN



ACNN



MoNet

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

MoNet as generalization of previous methods

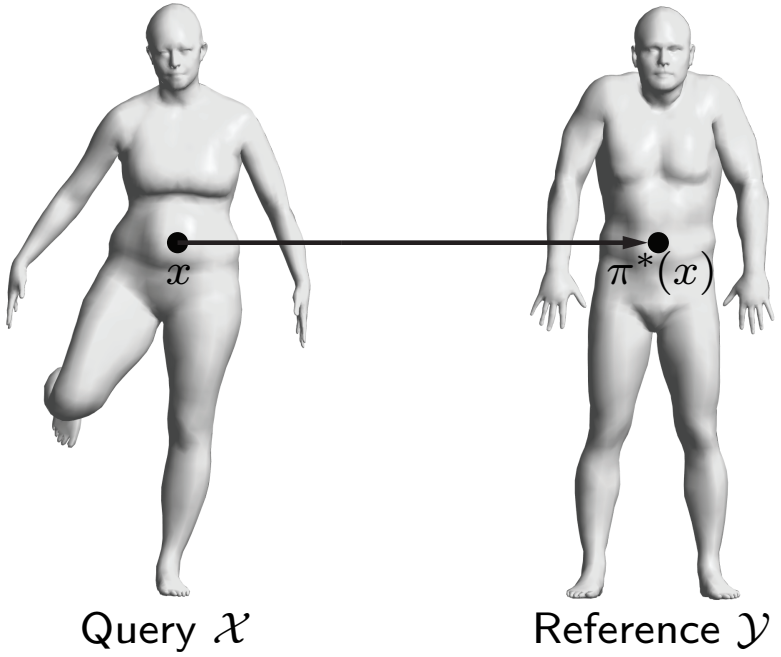
Method	**Coordinates** $\mathbf{u}(x,x')$	**Weight function** $w_{\boldsymbol{\Theta}}(\mathbf{u})$
CNN[1]	$\mathbf{u}(x') - \mathbf{u}(x)$	$\delta(\mathbf{u}-\mathbf{v})$ fixed parameters $\boldsymbol{\Theta} = \mathbf{v}$
GCNN[2]	$\rho(x,x'), \theta(x,x')$	$\exp\left(-\frac{1}{2}(\mathbf{u}-\mathbf{v})^\top \begin{pmatrix} \sigma_\rho^2 & \\ & \sigma_\theta^2 \end{pmatrix}^{-1} (\mathbf{u}-\mathbf{v})\right)$ fixed parameters $\boldsymbol{\Theta} = (\mathbf{v}, \sigma_\rho, \sigma_\theta)$
ACNN[3]	$\rho(x,x'), \theta(x,x')$	$\exp\left(-t\mathbf{u}^\top \mathbf{R}_\varphi \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\varphi^\top \mathbf{u}\right)$ fixed parameters $\boldsymbol{\Theta} = (\alpha, \varphi, t)$
MoNet[4]	$\rho(x,x'), \theta(x,x')$	$\exp\left(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\right)$ learnable parameters $\boldsymbol{\Theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$

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

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$ (discretized with n vertices)
- Correspondence = label each query vertex x as reference vertex y



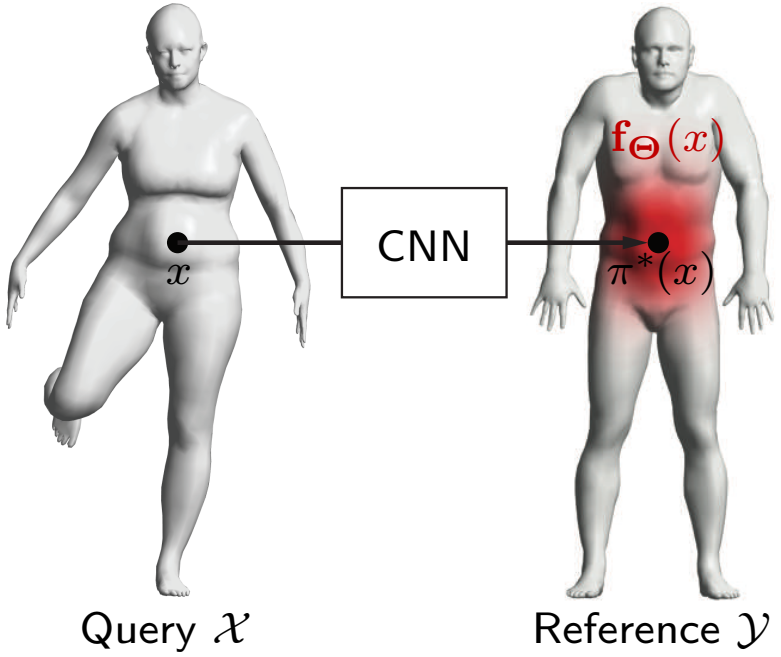
Rodolà et al. 2014; Masci et al. 2015

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$ (discretized with n vertices)
- Correspondence = label each query vertex x as reference vertex y
- Net output at x after softmax layer

  $$\mathbf{f}_{\boldsymbol{\Theta}}(x) = (f_{\boldsymbol{\Theta},1}(x), \ldots, f_{\boldsymbol{\Theta},n}(x))$$

  = probability distribution on $\mathcal{Y}$



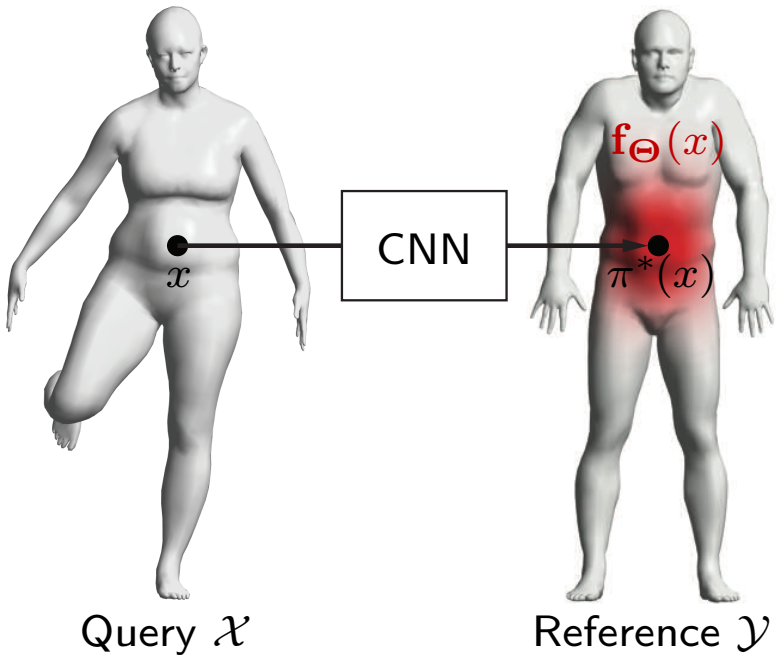
Rodolà et al. 2014; Masci et al. 2015

Learning deformation-invariant correspondence

- Groundtruth correspondence $\pi^* : \mathcal{X} \to \mathcal{Y}$ from query shape $\mathcal{X}$ to some reference shape $\mathcal{Y}$ (discretized with n vertices)
- Correspondence = label each query vertex x as reference vertex y
- Net output at x after softmax layer

$$\mathbf{f}_{\boldsymbol{\Theta}}(x) = (f_{\boldsymbol{\Theta},1}(x), \ldots, f_{\boldsymbol{\Theta},n}(x))$$

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

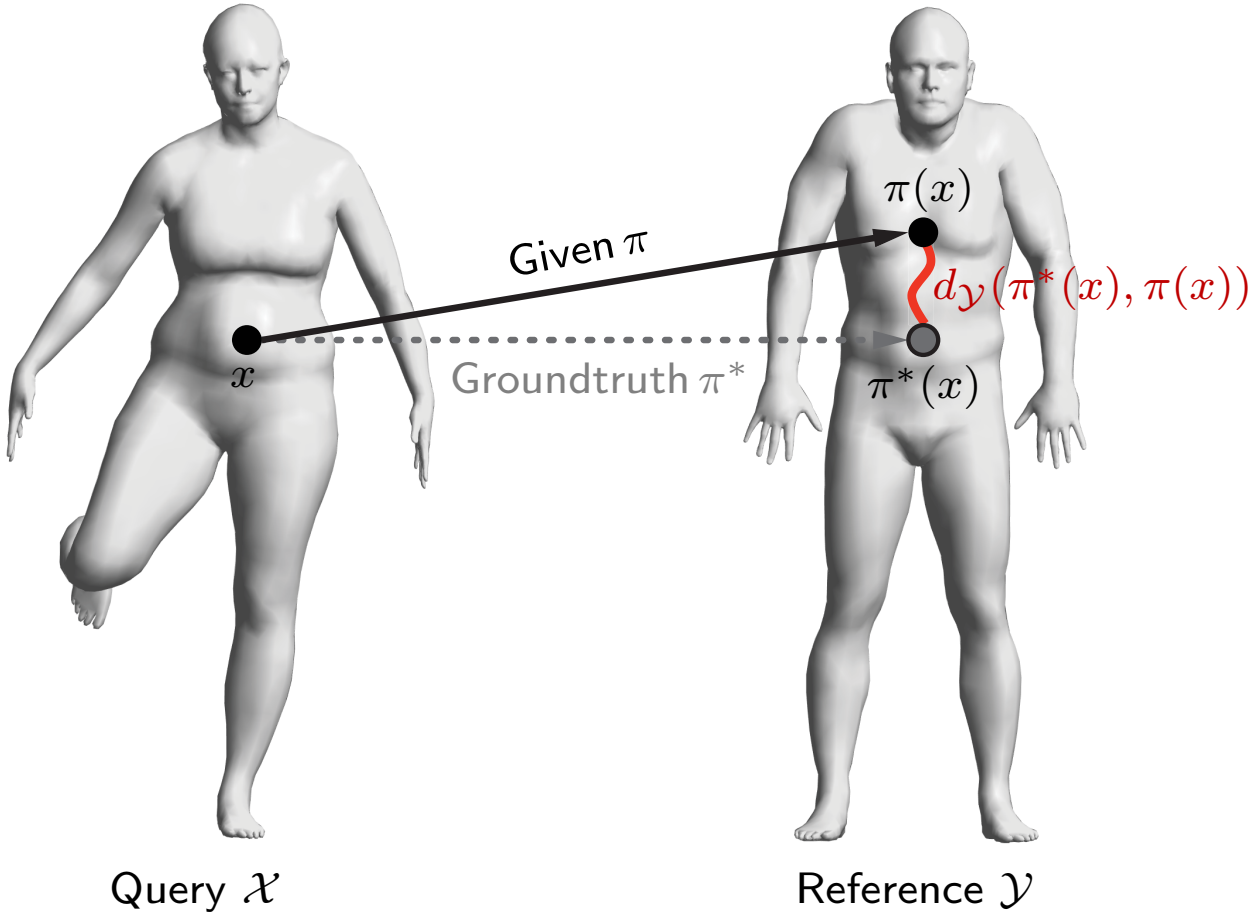


Minimize on training set the cross entropy between groundtruth correspondence and output probability distribution w.r.t. net parameters $\boldsymbol{\Theta}$

$$\min_{\boldsymbol{\Theta}} \sum_{x} H(\boldsymbol{\delta}_{\pi*(x)}, \mathbf{f}_{\boldsymbol{\Theta}}(x))$$

Rodolà et al. 2014; Masci et al. 2015

Correspondence evaluation: Princeton benchmark

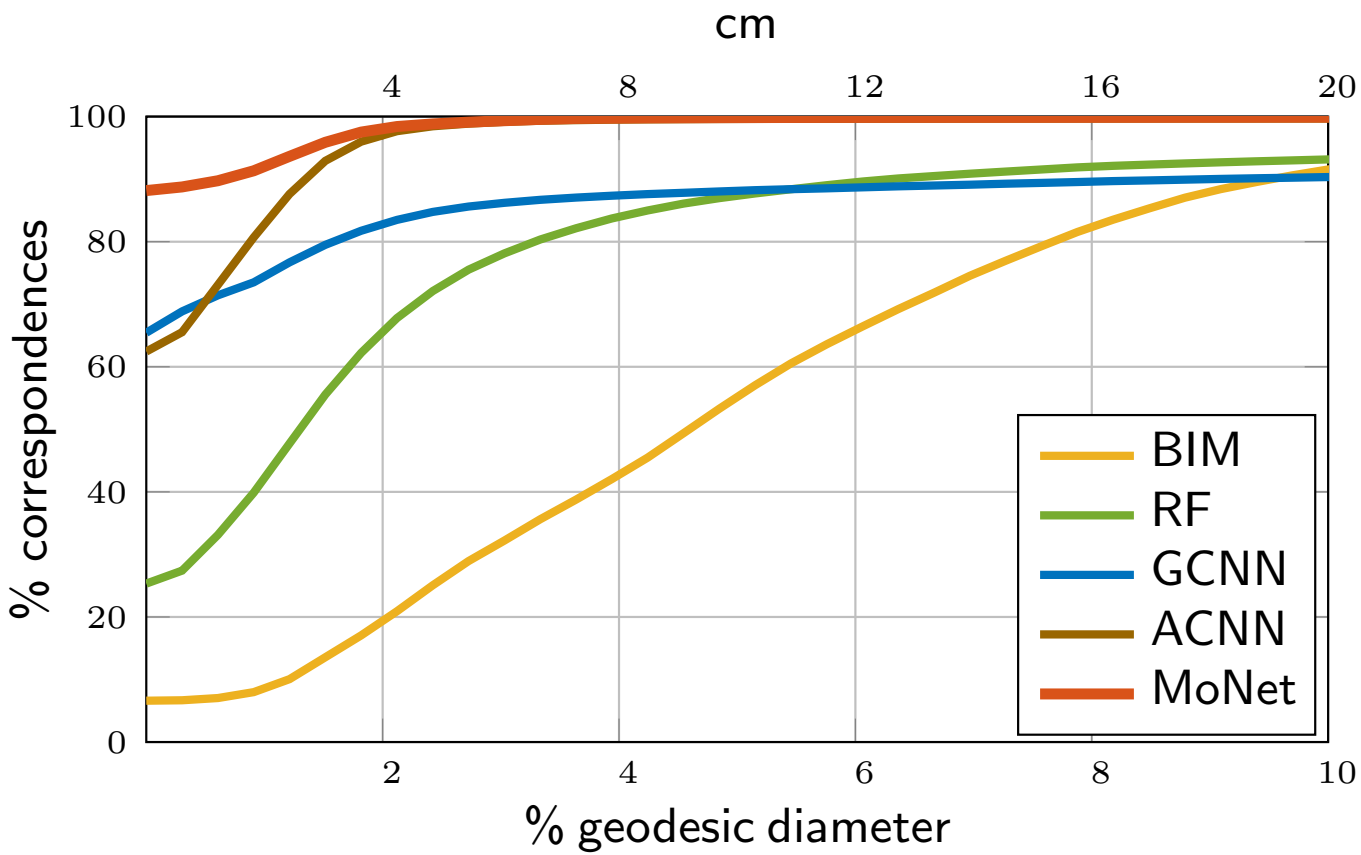


Pointwise correspondence error = geodesic distance from the groundtruth

$$\epsilon(x) = d_{\mathcal{Y}}(\pi^*(x), \pi(x))$$

Kim et al. 2011

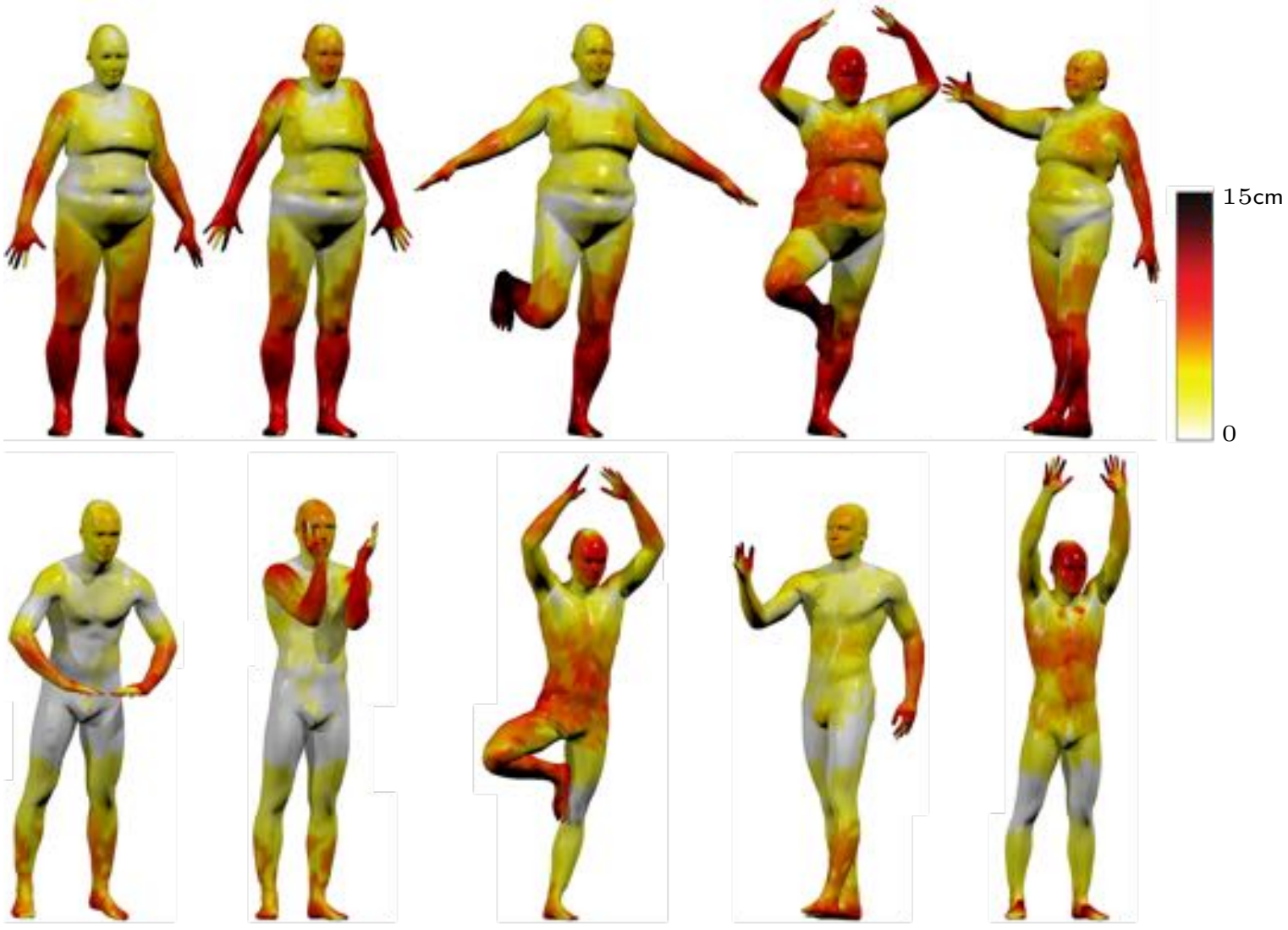
Correspondence quality comparison



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)

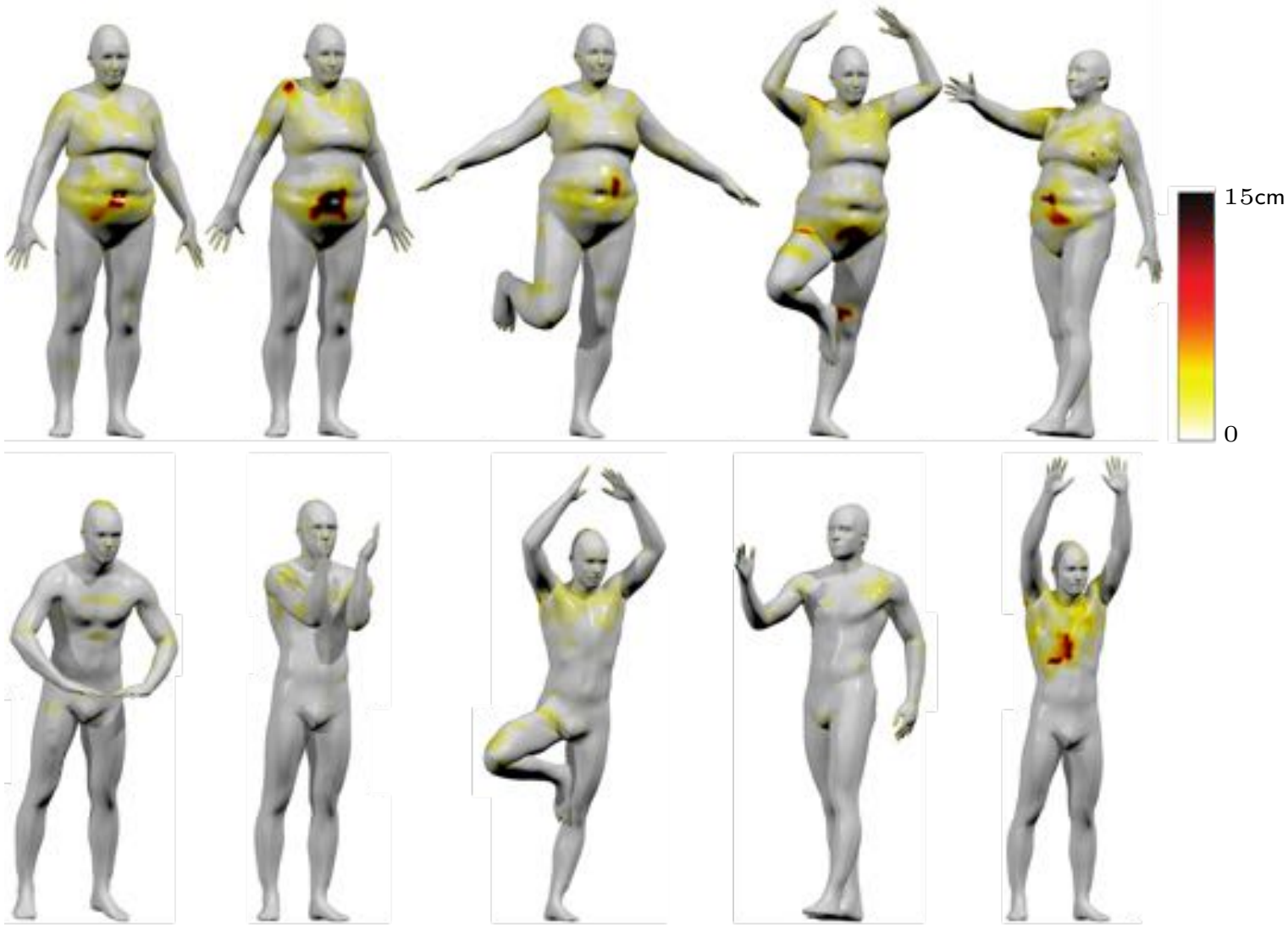
Kim, Lipman, Funkhouser 2011

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

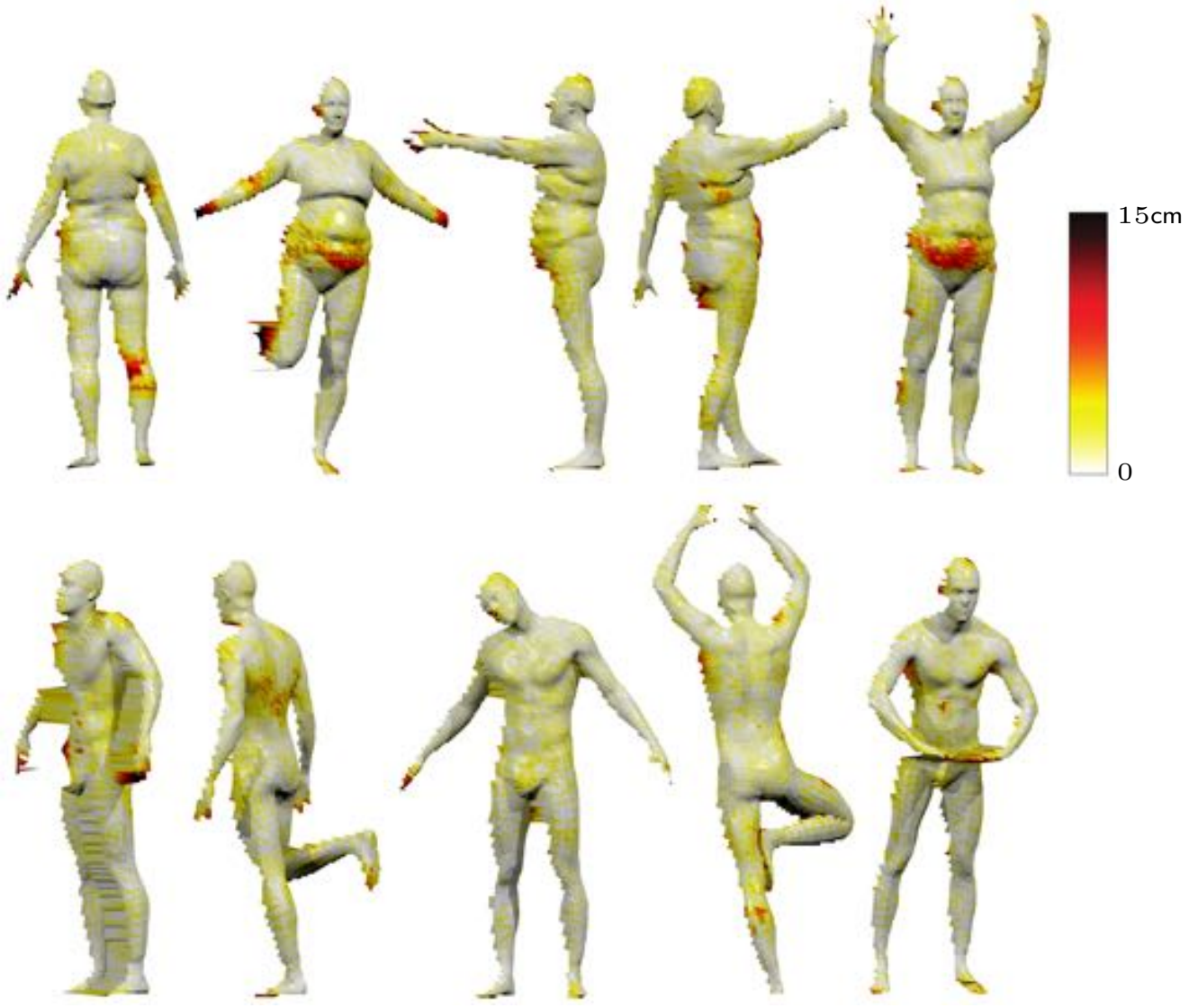
Shape correspondence visualization: MoNet

Reference

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

Reference

Correspondence visualization (similar colors encode corresponding points)

Monti et al. 2016

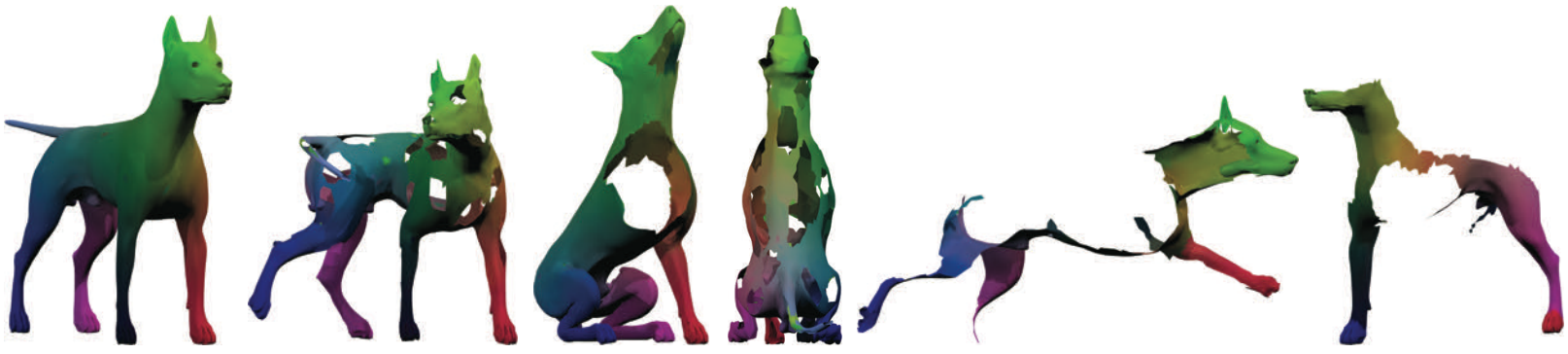
Partial correspondence with ACNN

Correspondence

Correspondence error

Boscaini et al. 2016

Partial correspondence with ACNN

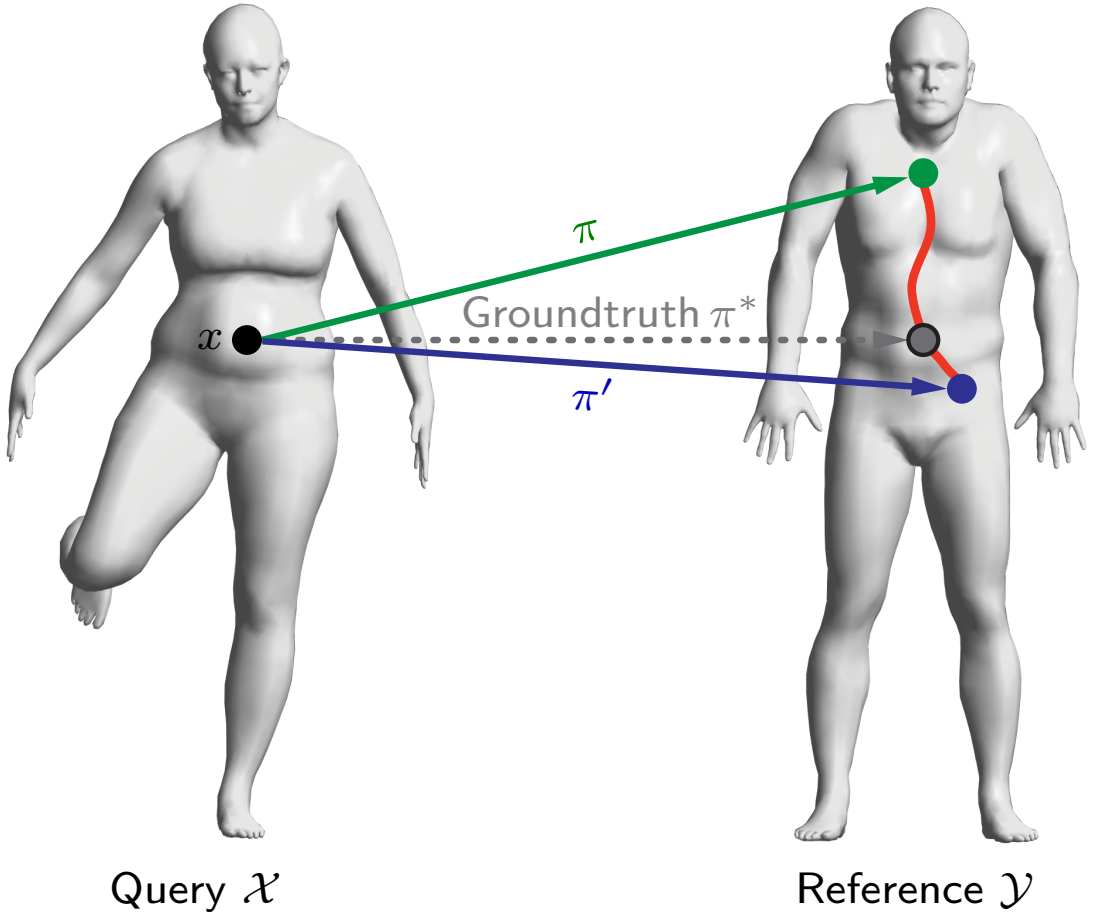


Correspondence

Correspondence error

Boscaini et al. 2016

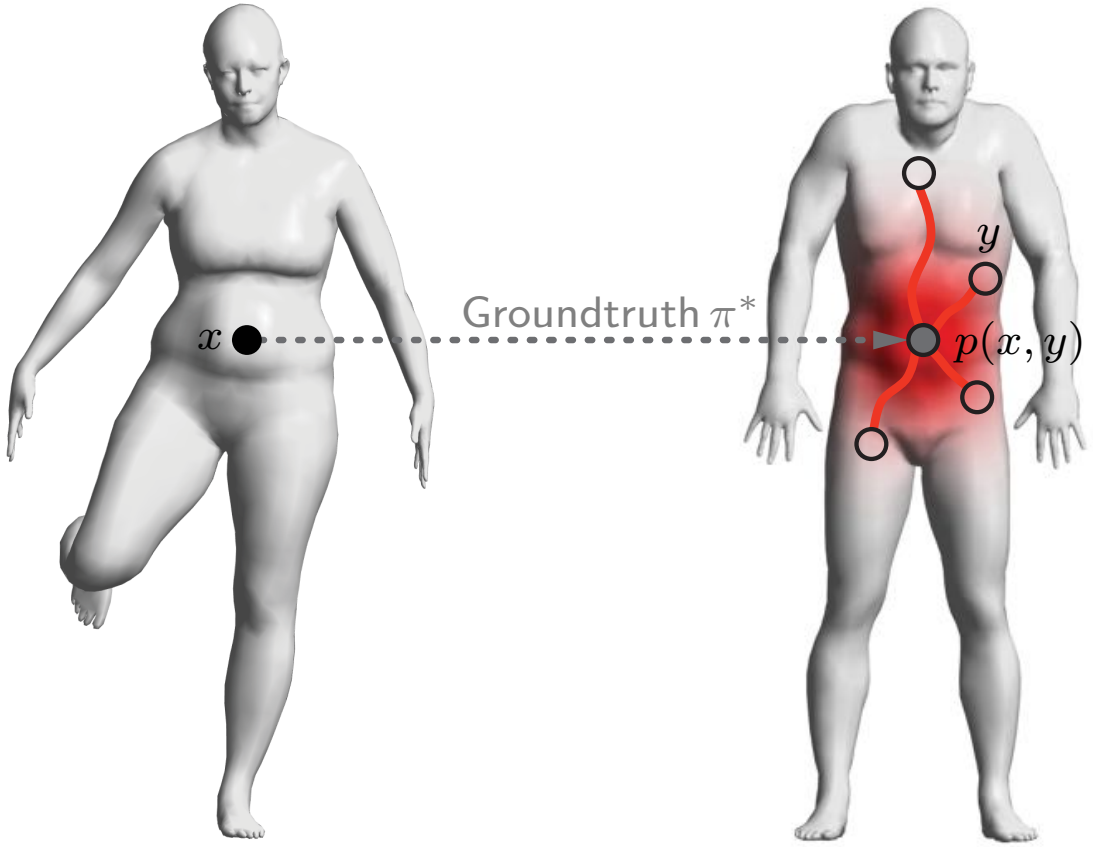
Correspondence as classification problem, revisited



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

Kim et al. 2011

Soft correspondence error

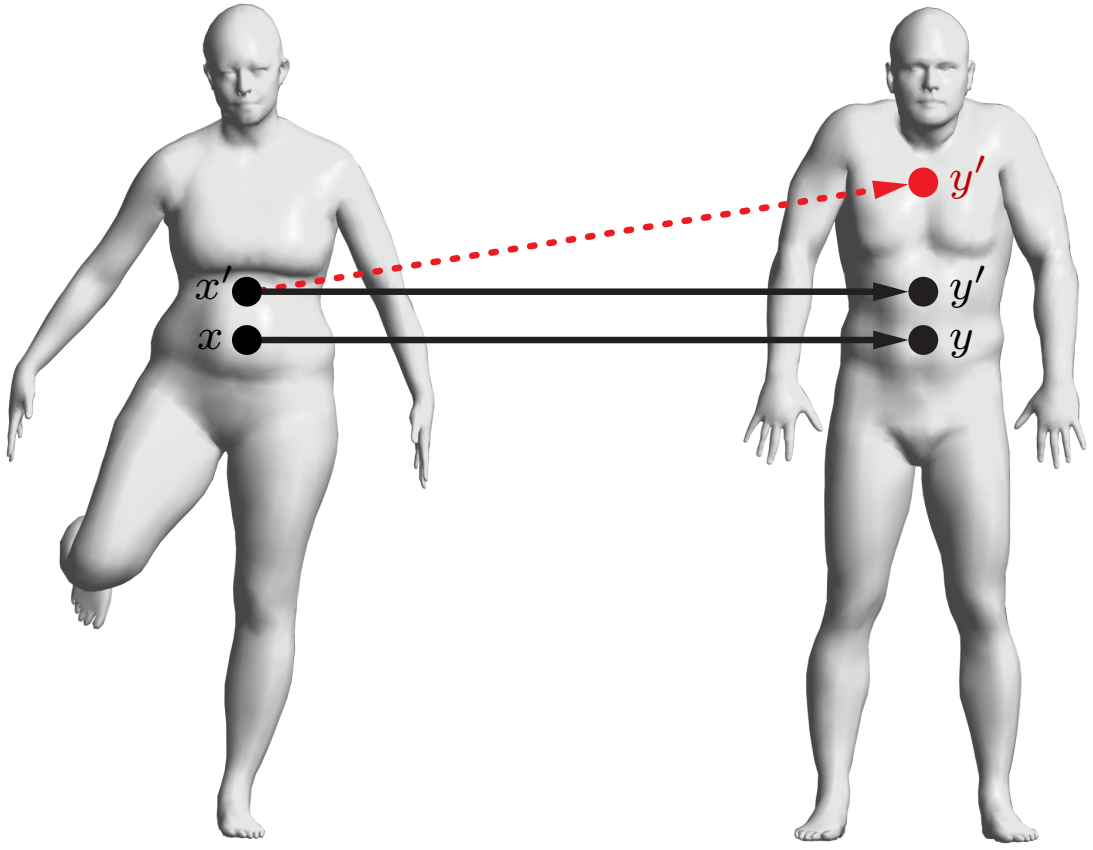


Soft correspondence error = probability-weighted geodesic distance from the groundtruth

$$\bar{\epsilon}(x) = \int_{\mathcal{Y}} p(x, y) d_{\mathcal{Y}}(\pi^*(x), y) dy$$

Kovnatsky et al. 2015; Litany et al. 2017

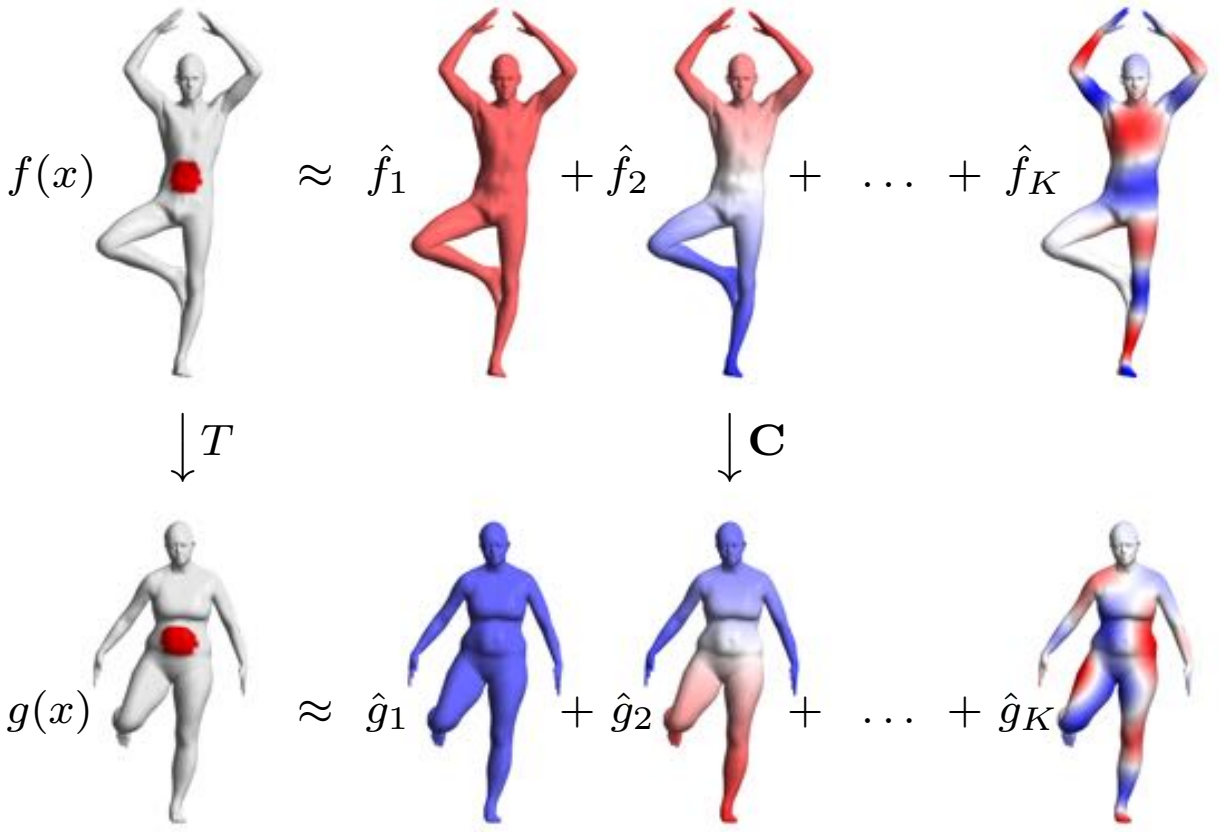
Pointwise vs Structured learning



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

Litany et al. 2017

Functional maps: spectral domain

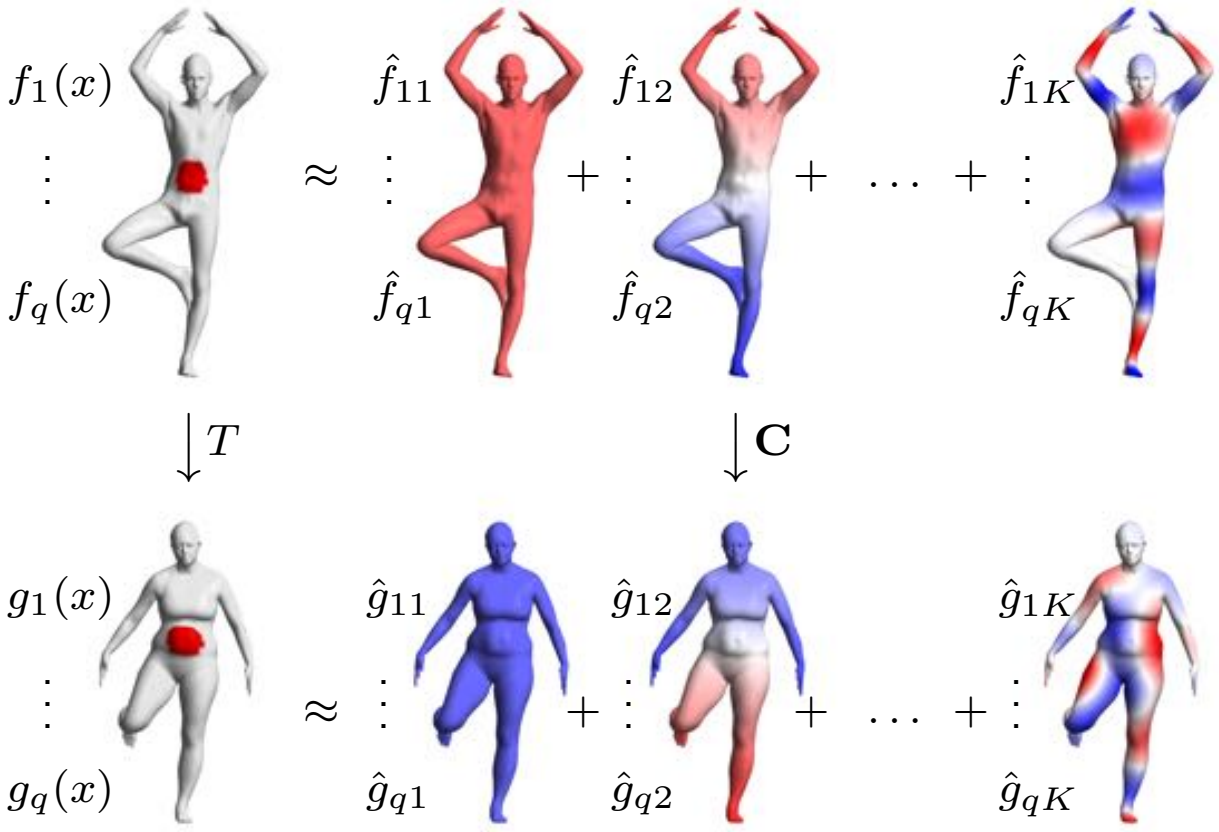


Functional correspondence T = linear map $\mathbf{C}$ between Fourier coefficients

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

Ovsjanikov et al. 2012

Functional maps: spectral domain

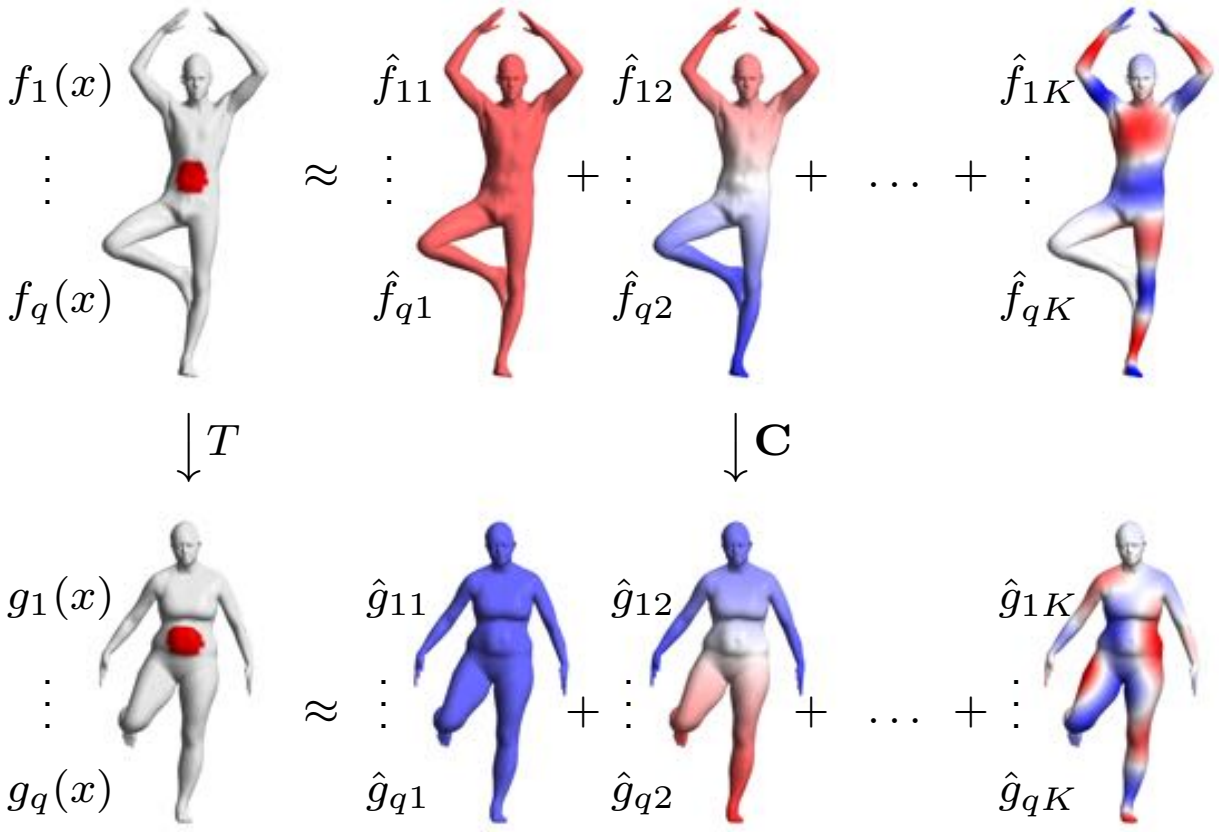


Recover correspondence from $q \geq k$ dimensional pointwise features

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

Ovsjanikov et al. 2012

Functional maps: spectral domain

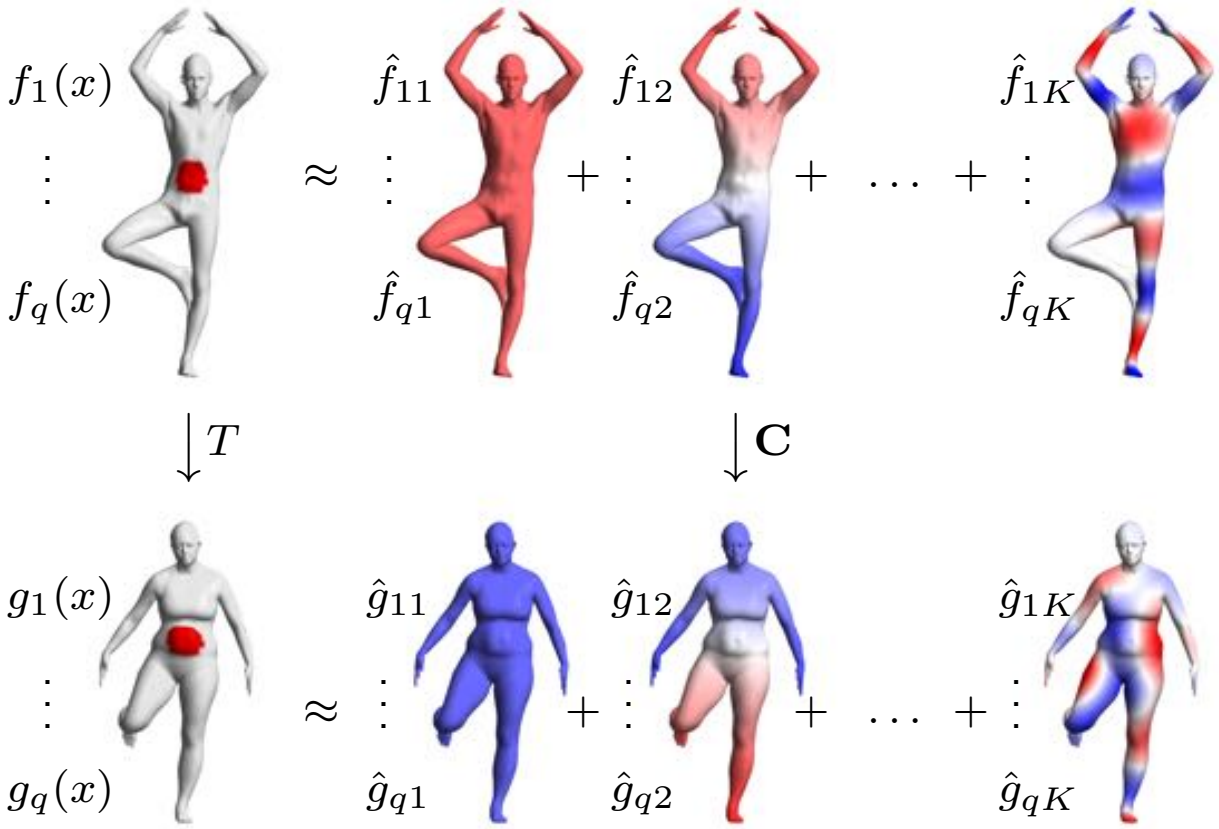


Recover correspondence from $q \geq k$ dimensional pointwise features

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

Ovsjanikov et al. 2012

Functional maps: spectral domain

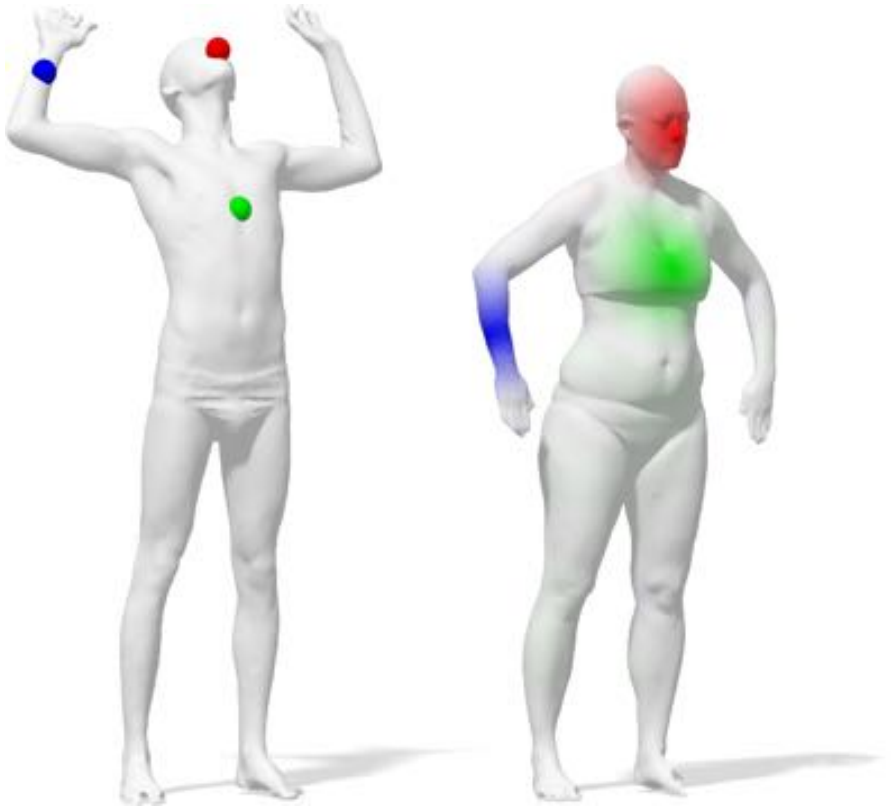


Recover correspondence from $q \geq k$ dimensional pointwise features

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

Ovsjanikov et al. 2012

Functional maps: spatial domain

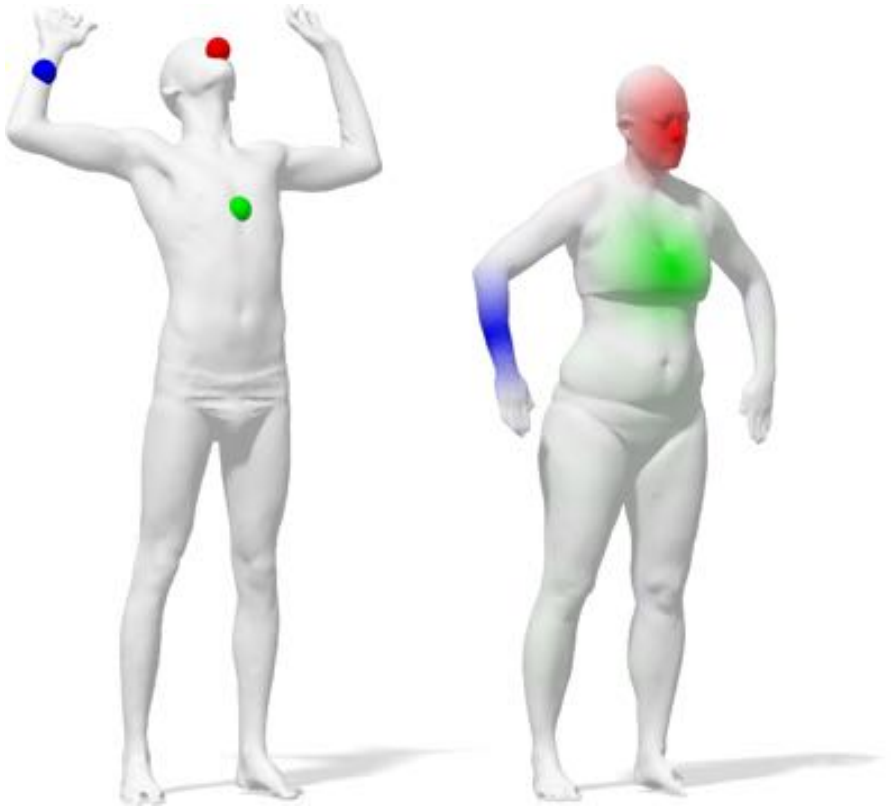


Rank-K approximation of spatial correspondence

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

Ovsjanikov et al. 2012

Functional maps: spatial domain

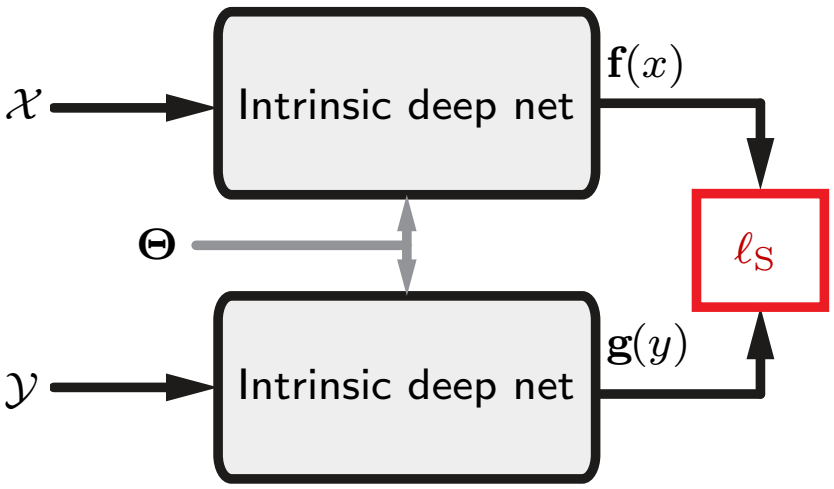


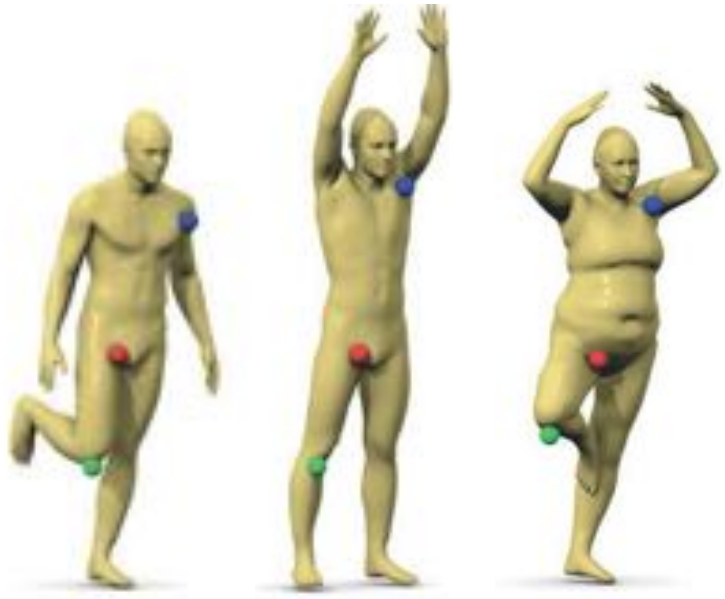
Probability $p(x, y)$ of point x mapping to y

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

Ovsjanikov et al. 2012

Siamese metric learning





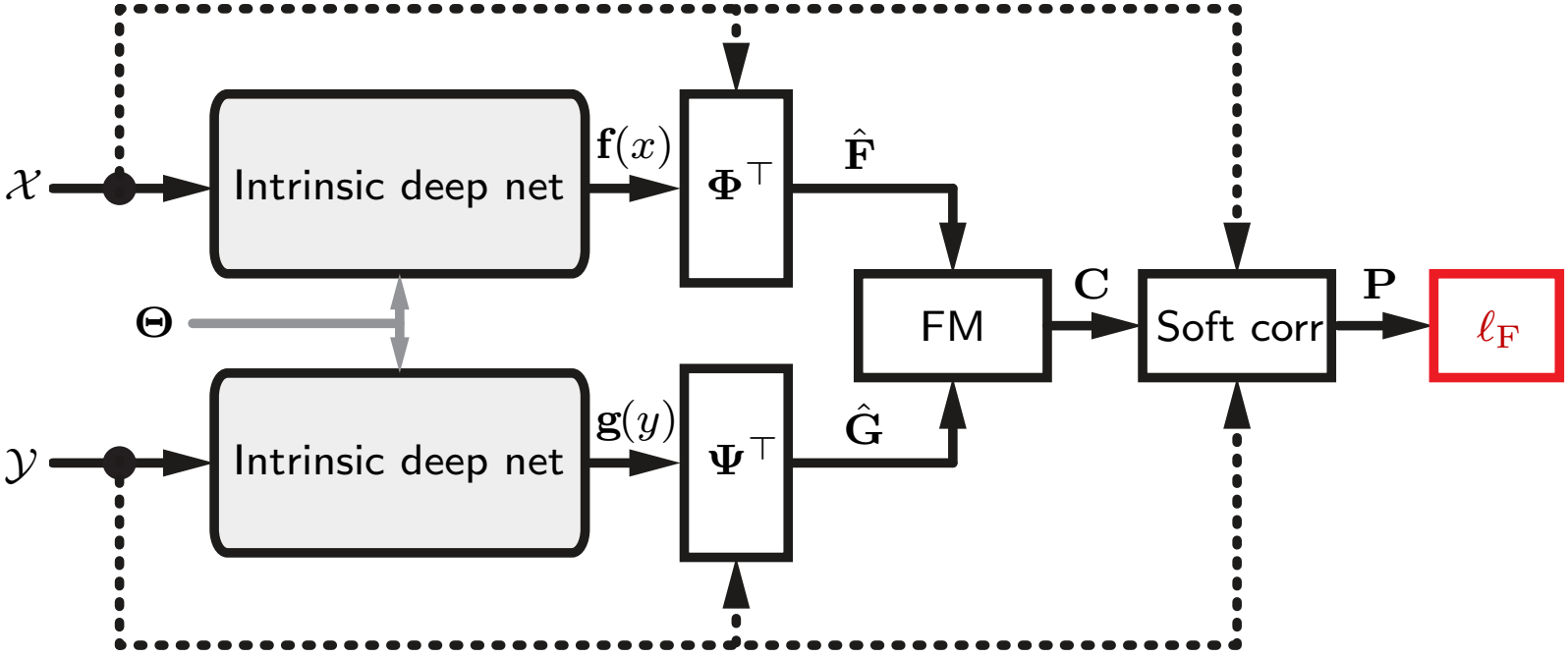
Siamese net — two net instances with shared parameters $\boldsymbol{\Theta}$

Poitwise feature cost

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

Boscaini et al. 2015; Cosmo et al. 2016

Structured correspondence with FMNet



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

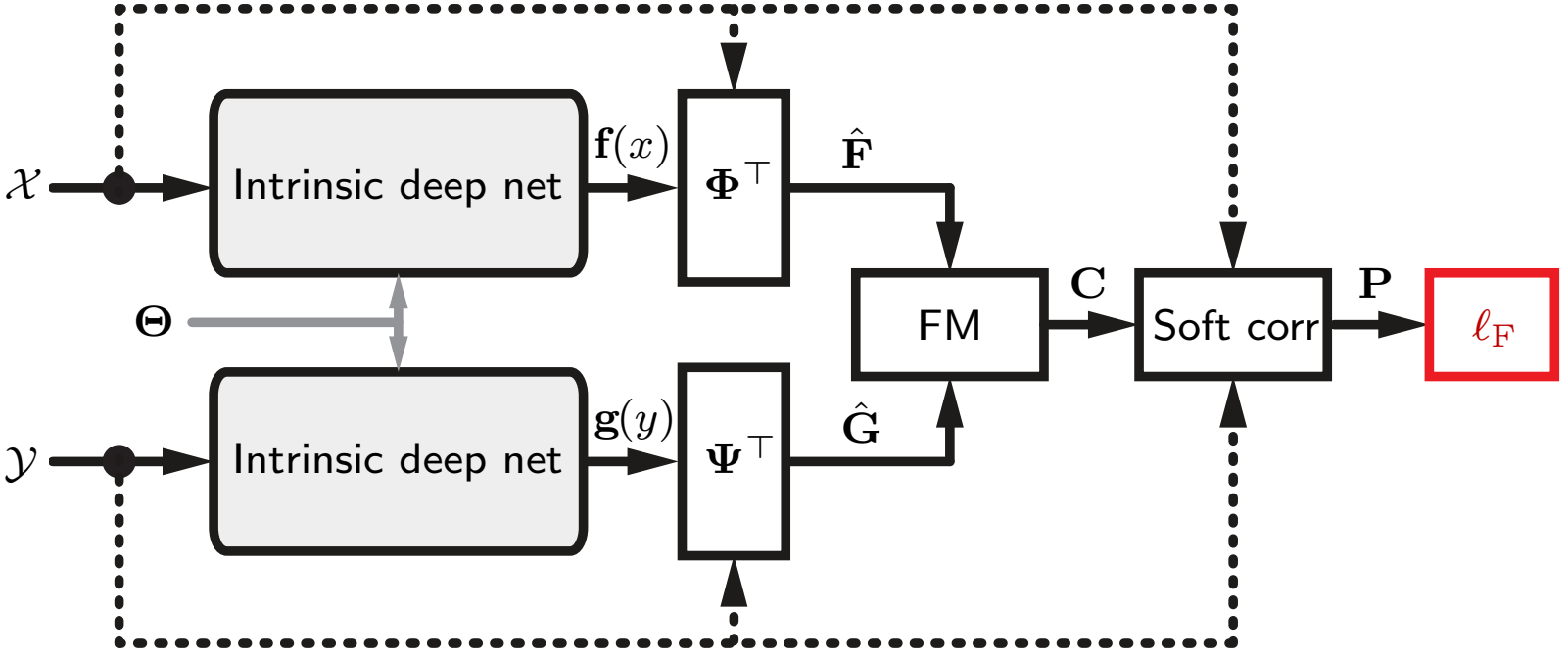
Functional map layer — $\mathbf{C}^*_{\mathbf{\Theta}} = \underset{\mathbf{C}}{\operatorname{argmin}} \|\hat{\mathbf{F}}_{\mathbf{\Theta}}\mathbf{C} - \hat{\mathbf{G}}_{\mathbf{\Theta}}\|^2_{\mathrm{F}}$

Soft correspondence layer — $\mathbf{P}_{\mathbf{\Theta}} = |\mathbf{\Psi}\mathbf{C}_{\mathbf{\Theta}}\mathbf{\Phi}^\top|_{\|\cdot\|}$

Soft error cost — $\ell_{\mathrm{F}}(\mathbf{\Theta}) = \int_{\mathcal{X}}\int_{\mathcal{Y}} p_{\mathbf{\Theta}}(x,y) d_{\mathcal{Y}}(\pi^*(x), y)\, dx\, dy$

Litany et al. 2017

Structured correspondence with FMNet



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

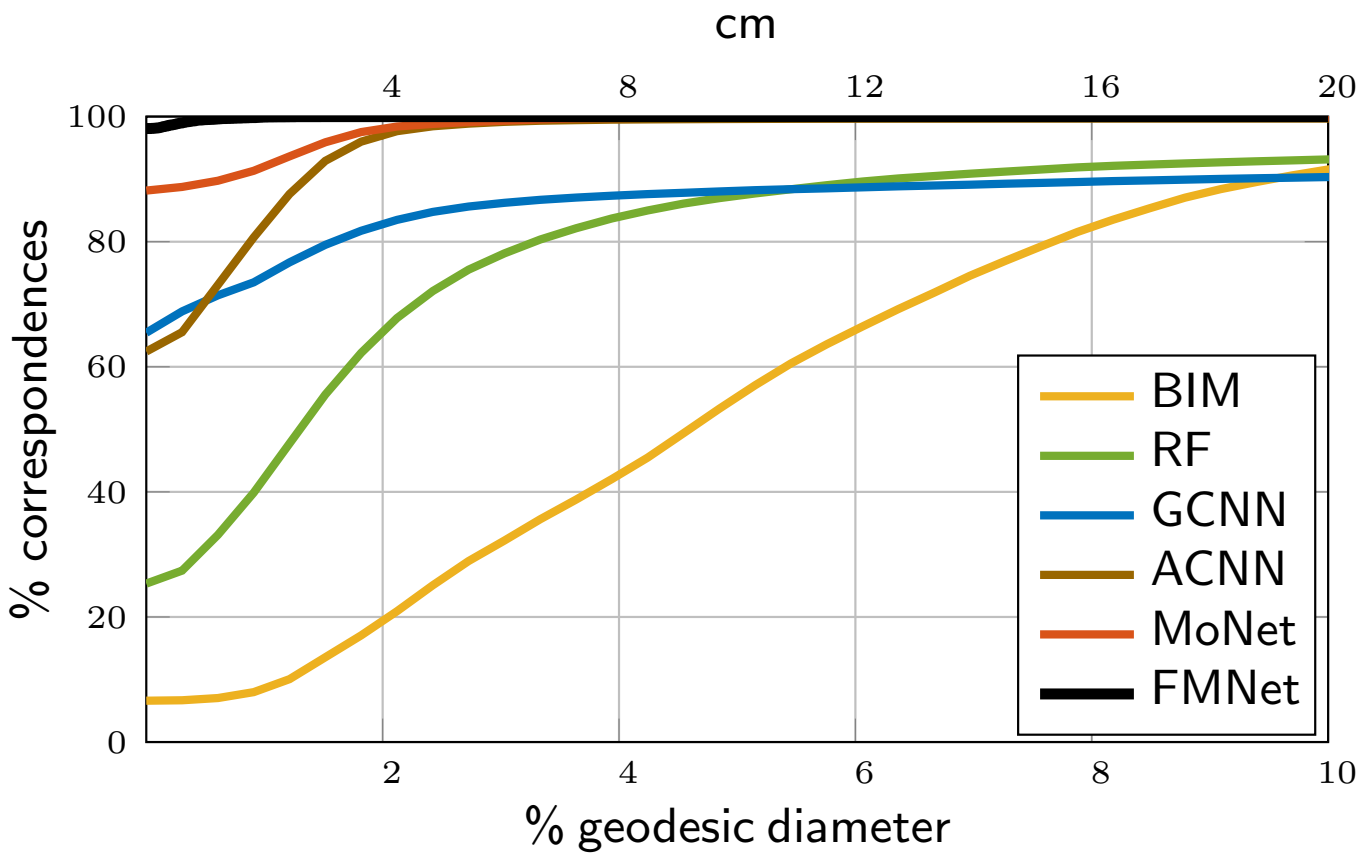
Functional map layer — $\mathbf{C}^*_{\mathbf{\Theta}} = \hat{\mathbf{F}}^\dagger_{\mathbf{\Theta}} \hat{\mathbf{G}}_{\mathbf{\Theta}}$

Soft correspondence layer — $\mathbf{P}_{\mathbf{\Theta}} = |\mathbf{\Psi} \mathbf{C}_{\mathbf{\Theta}} \mathbf{\Phi}^\top|_{\|\cdot\|}$

Soft error cost — $\ell_{\mathrm{F}}(\mathbf{\Theta}) = \|\mathbf{P}_{\mathbf{\Theta}} \circ \mathbf{D}_{\mathcal{Y}}\|$

Litany et al. 2017

Correspondence quality comparison



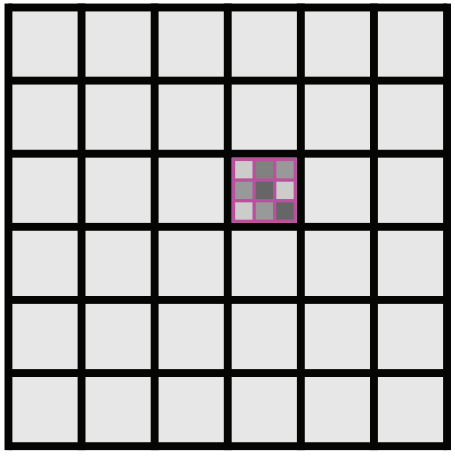
Correspondence evaluated using asymmetric Princeton benchmark
(training and testing: disjoint subsets of FAUST)

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

Different formulations of non-Euclidean CNNs

Spectral domain

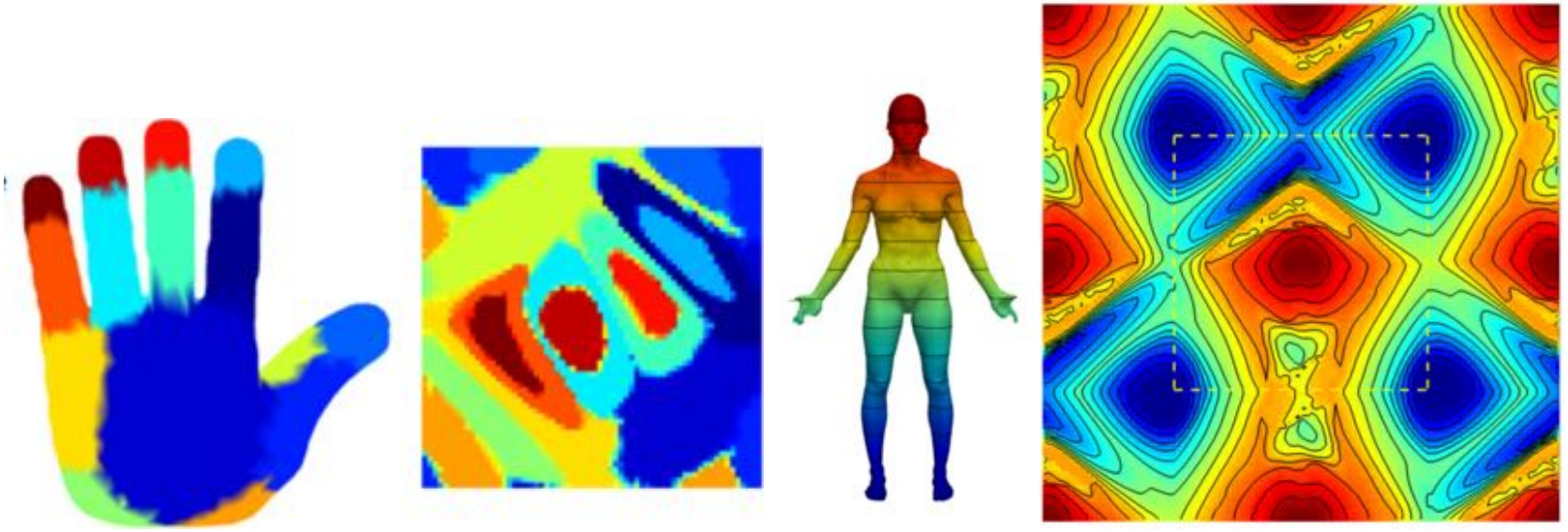
Spatial domain



Embedding domain

Global parametrization

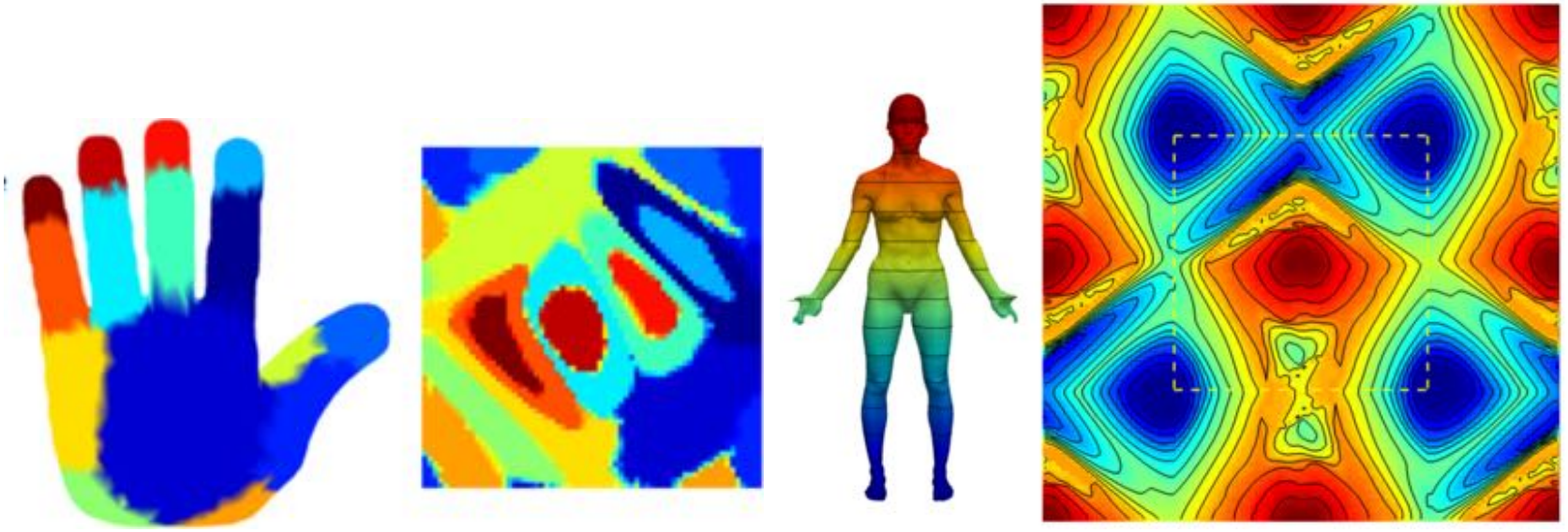
Map the input surface to some parametric domain with shift-invariant structure



Sinha et al. 2016; Maron et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure

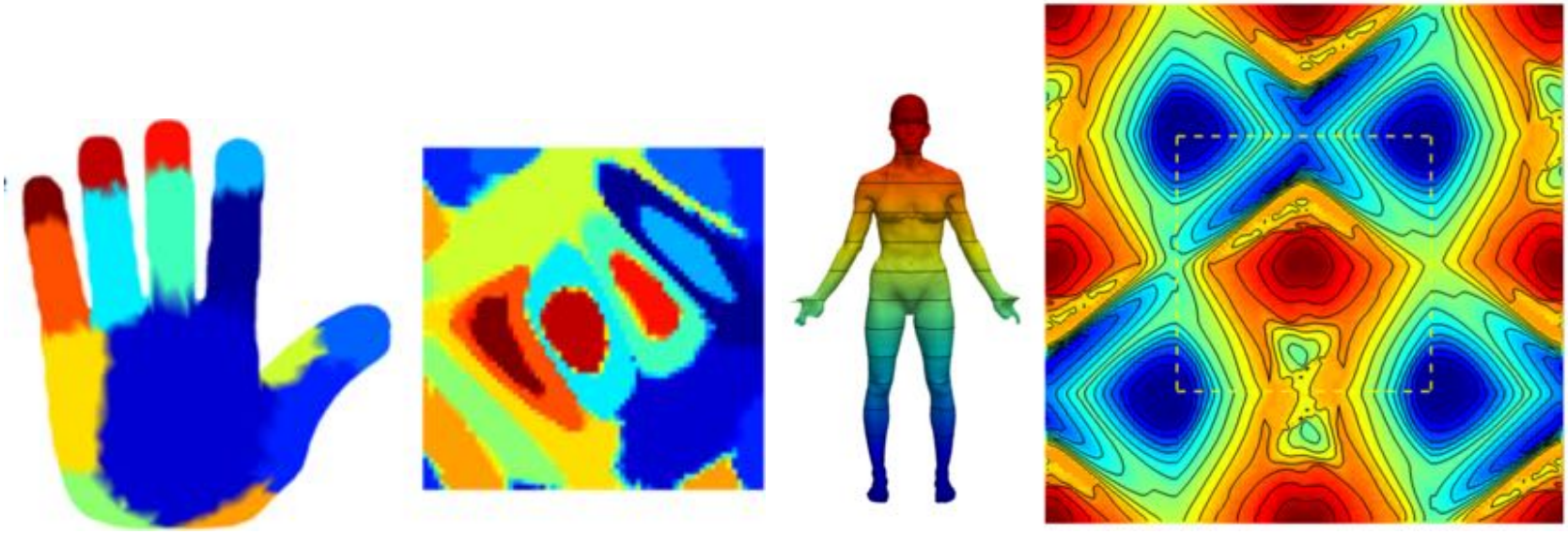


☺ Allows to use standard CNNs (pull back convolution from the parametric space)

Sinha et al. 2016; Maron et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure



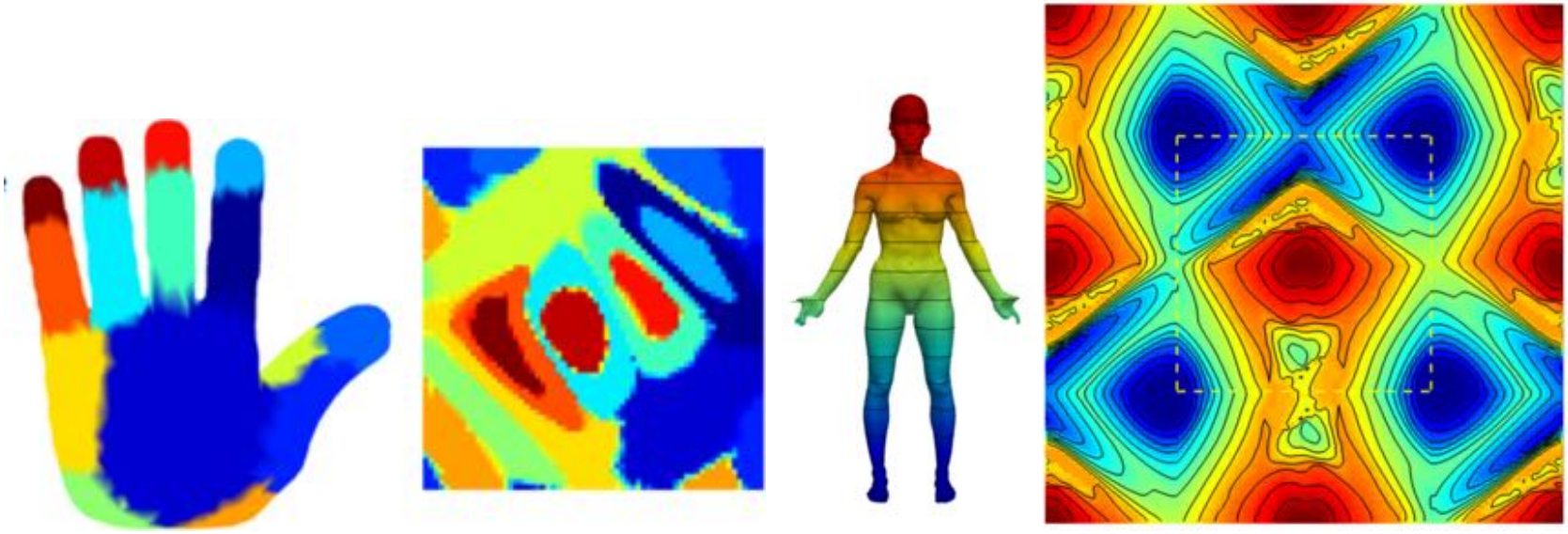
☺ Allows to use standard CNNs (pull back convolution from the parametric space)

☺ Guaranteed invariance to some classes of transformations

Sinha et al. 2016; Maron et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure

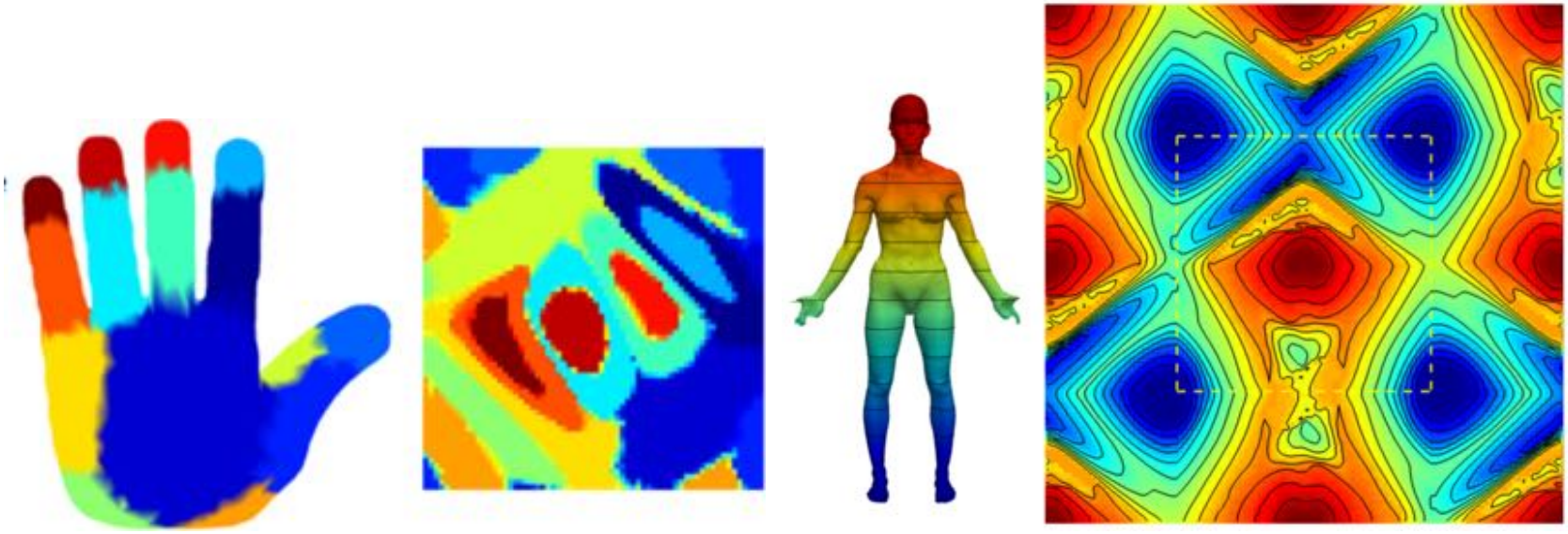


- ☺ Allows to use standard CNNs (pull back convolution from the parametric space)
- ☺ Guaranteed invariance to some classes of transformations
- ☹ Parametrization may not be unique

Sinha et al. 2016; Maron et al. 2017

Global parametrization

Map the input surface to some parametric domain with shift-invariant structure



- ☺ Allows to use standard CNNs (pull back convolution from the parametric space)
- ☺ Guaranteed invariance to some classes of transformations
- ☹ Parametrization may not be unique
- ☹ Embedding may introduce distortion

Sinha et al. 2016; Maron et al. 2017

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation = flow along a non-vanishing vector field

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation = flow along a non-vanishing vector field

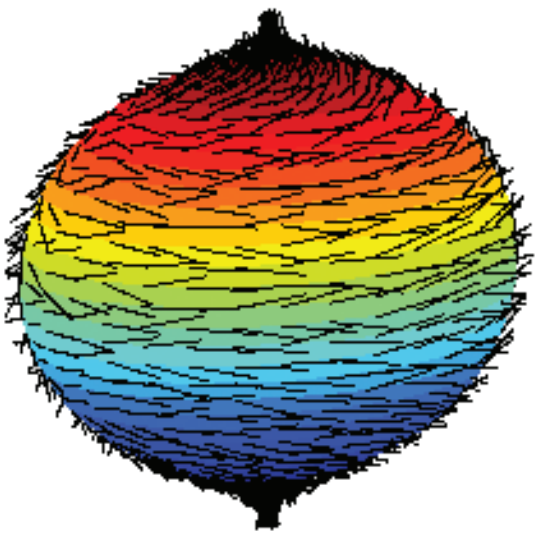
Poincaré-Hopf Theorem Non-vanishing vector field on a closed orientable compact 2-manifold implies manifold of genus 1 (torus)

Poincaré 1881; Hopf 1926

Translation invariance on manifolds

Translation on manifold = locally Euclidean translation = flow along a non-vanishing vector field

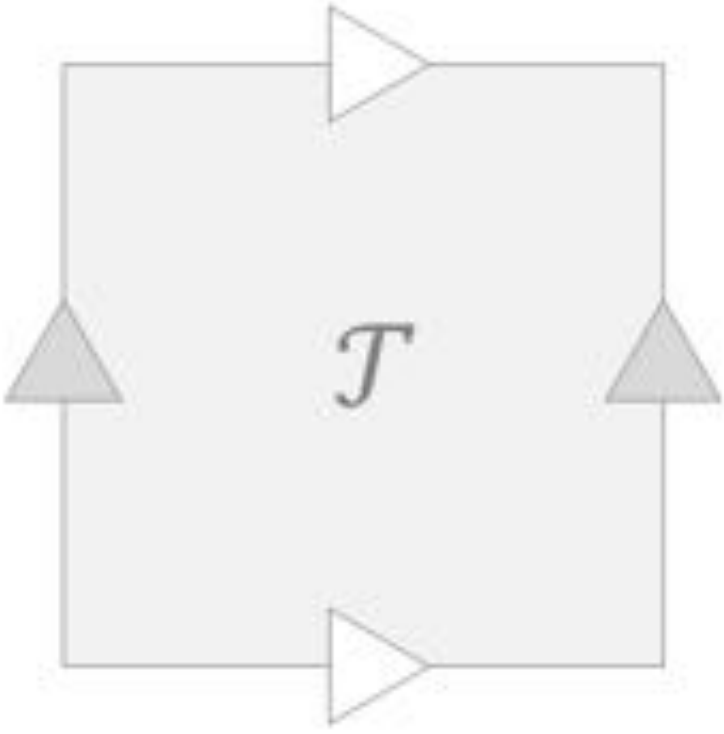
Poincaré-Hopf Theorem Non-vanishing vector field on a closed orientable compact 2-manifold implies manifold of genus 1 (torus)



'Hairy Ball Theorem' states that a sphere cannot be combed

Poincaré 1881; Hopf 1926

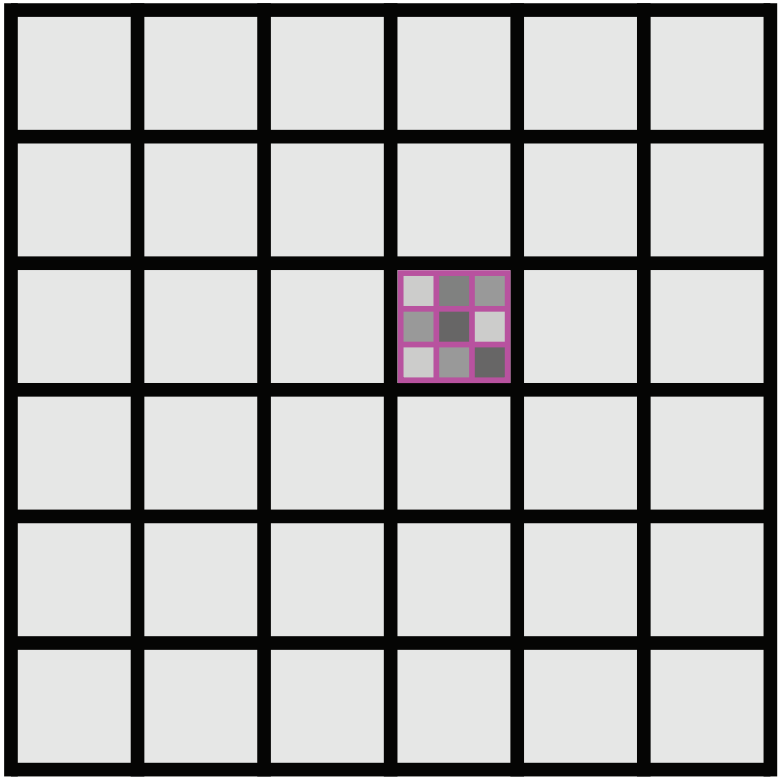
Translation invariance on the torus



Torus is the only closed orientable surface admitting a translation group

Convolution on torus

For any triplet of points on $\mathcal{X}$, construct conformal homeomorphism from the 4-cover $\mathcal{X}^4$ to $\mathcal{T}$ using orbifold-Tutte method



Maron et al. 2017

Conformal zoom

- Embedding depends on the choice of the triplets of points

Maron et al. 2017

Conformal zoom

- Embedding depends on the choice of the triplets of points
- 'Conformal zoom' effect

Maron et al. 2017

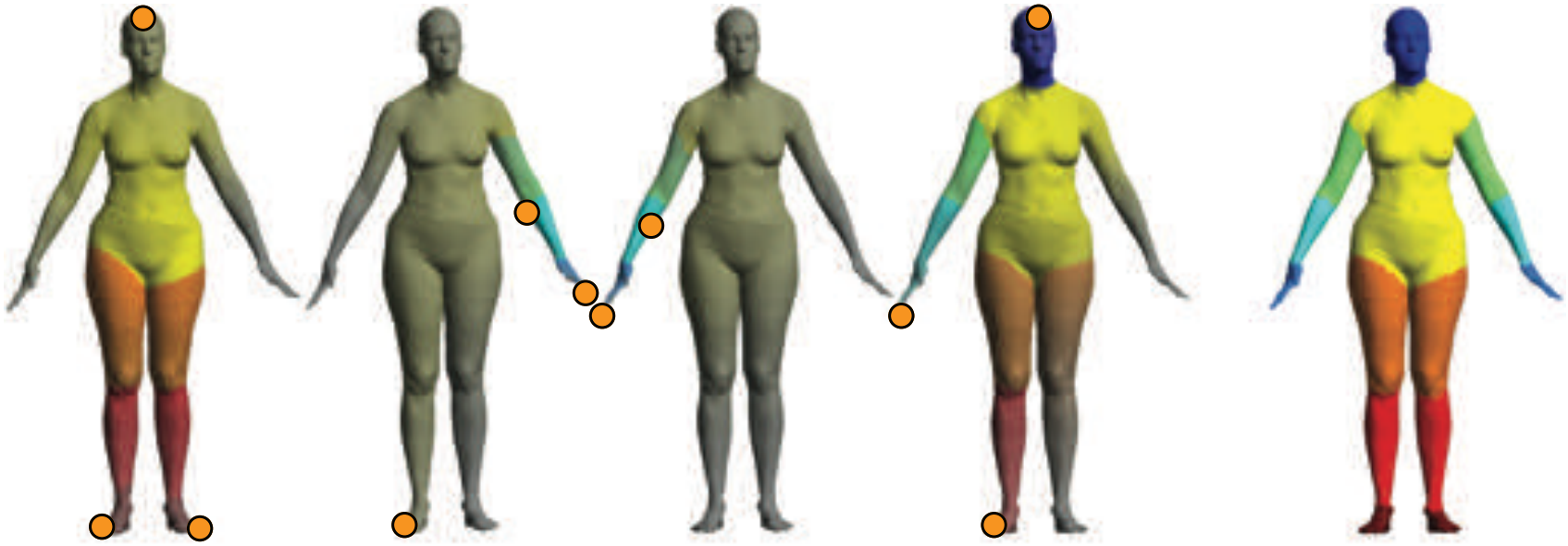
Conformal zoom

- Embedding depends on the choice of the triplets of points
- 'Conformal zoom' effect
- Choose multiple triples and aggregate results in training / test phase

Maron et al. 2017

Conformal zoom

- Embedding depends on the choice of the triplets of points
- 'Conformal zoom' effect
- Choose multiple triples and aggregate results in training / test phase

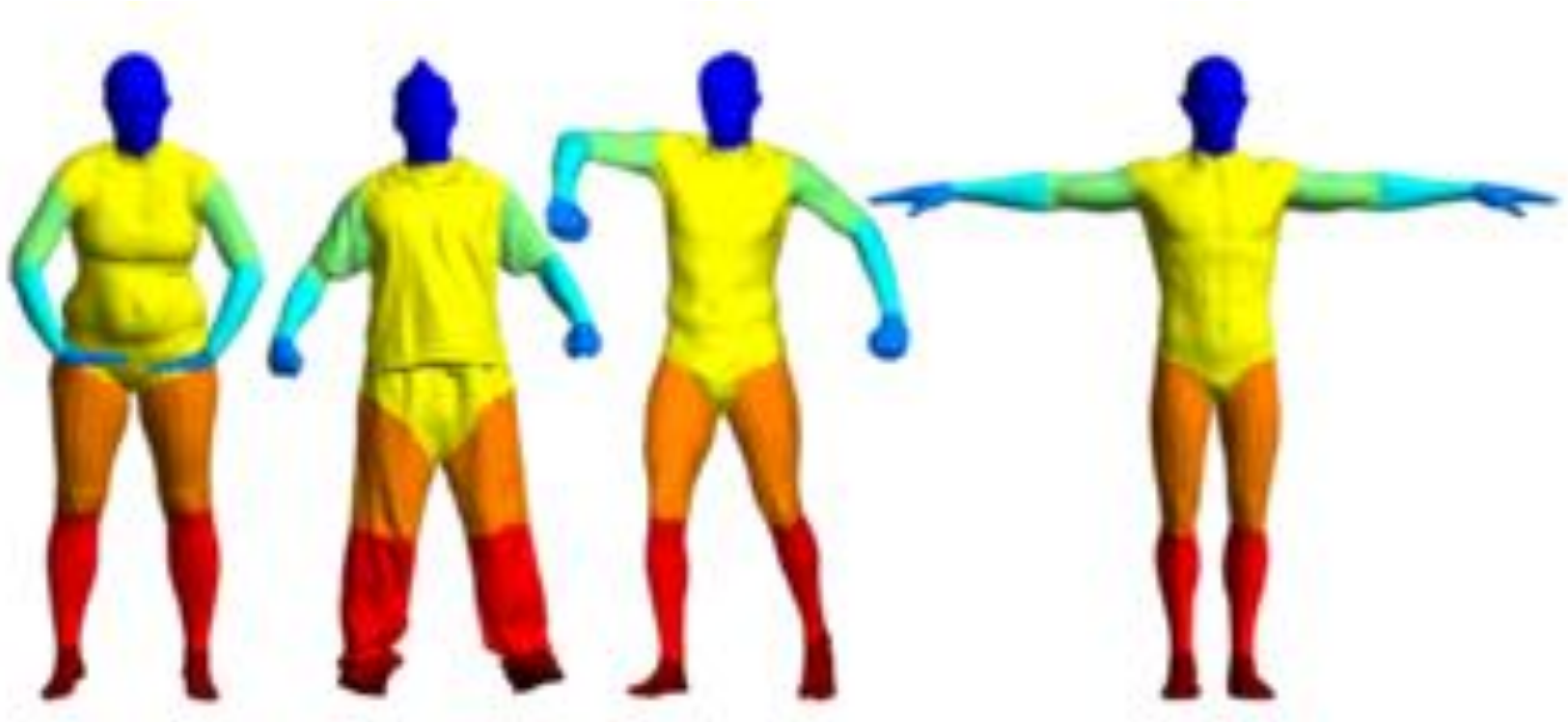


different triplets

aggregated result

Maron et al. 2017

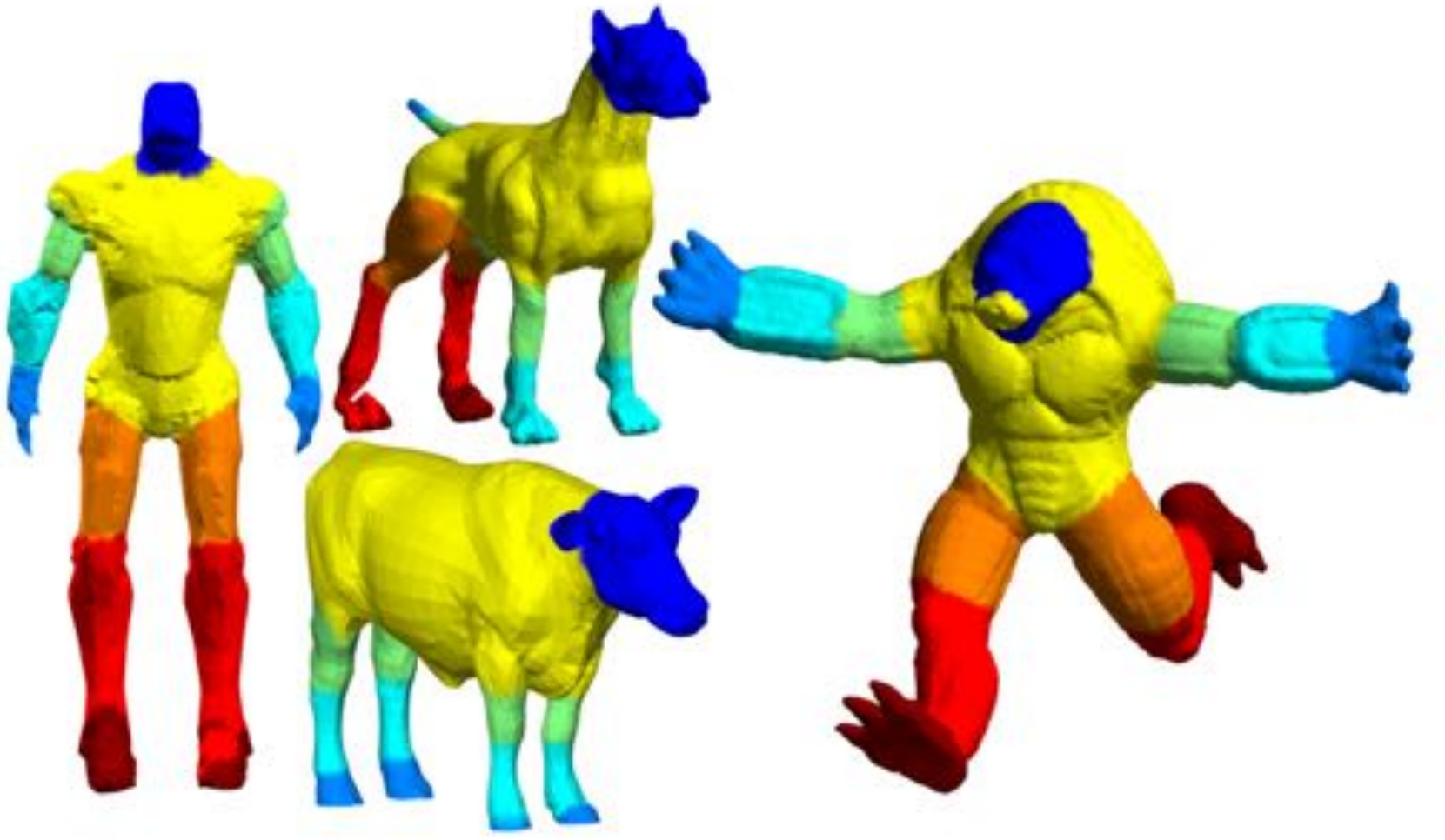
Example: shape segmentation with Toric CNN



Examples of shape segmentation obtained with Toric CNN

Maron et al. 2017

Example: shape segmentation with Toric CNN



Examples of shape segmentation obtained with Toric CNN

Maron et al. 2017