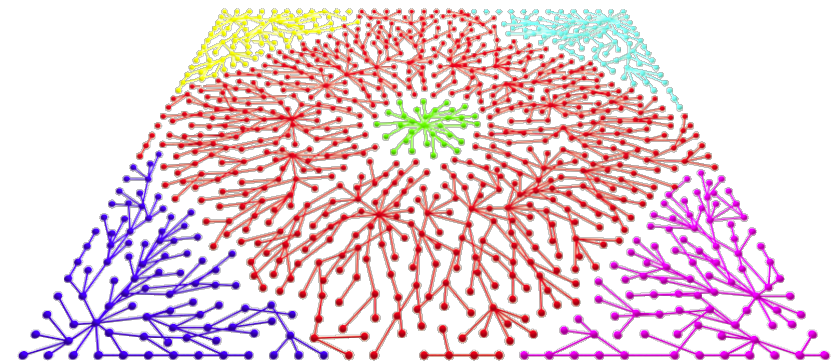
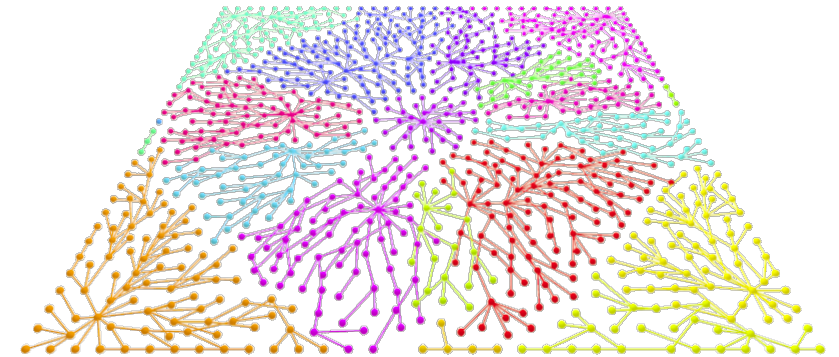


# CS233, CME251: Geometric and Topological Data Analysis

Leonidas Guibas  
Computer Science Department  
Stanford University



Lecture 17  
08 June 2020

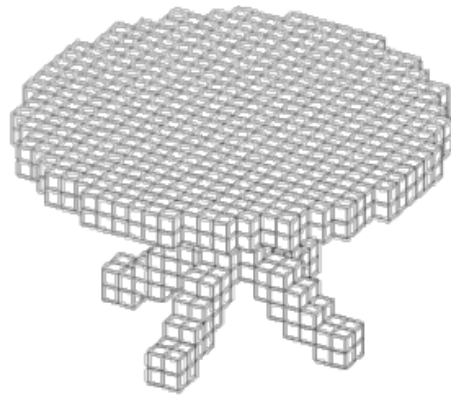


Previously: functional maps,  
CNN for 3D representations

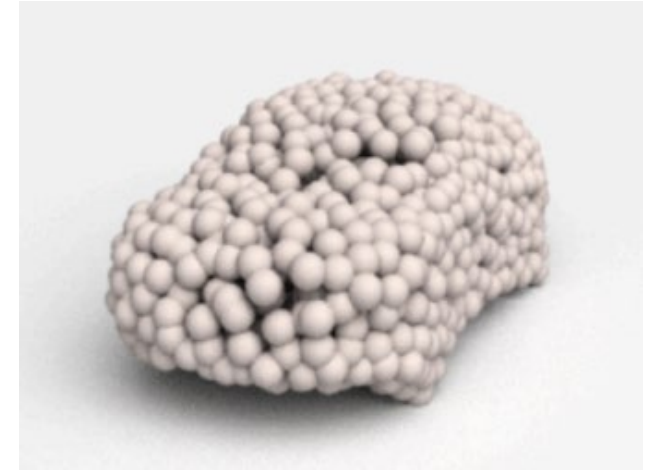
# 3D representations



Multi-view



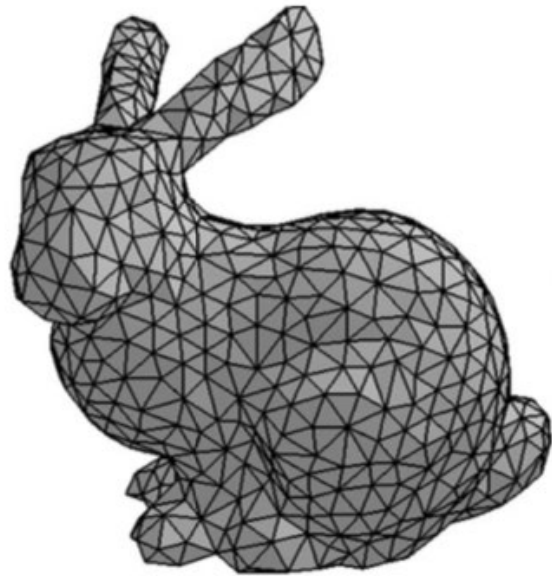
Volumetric



Point cloud

# Today: CNN for graphs and meshes

# What about non-Euclidean domains?



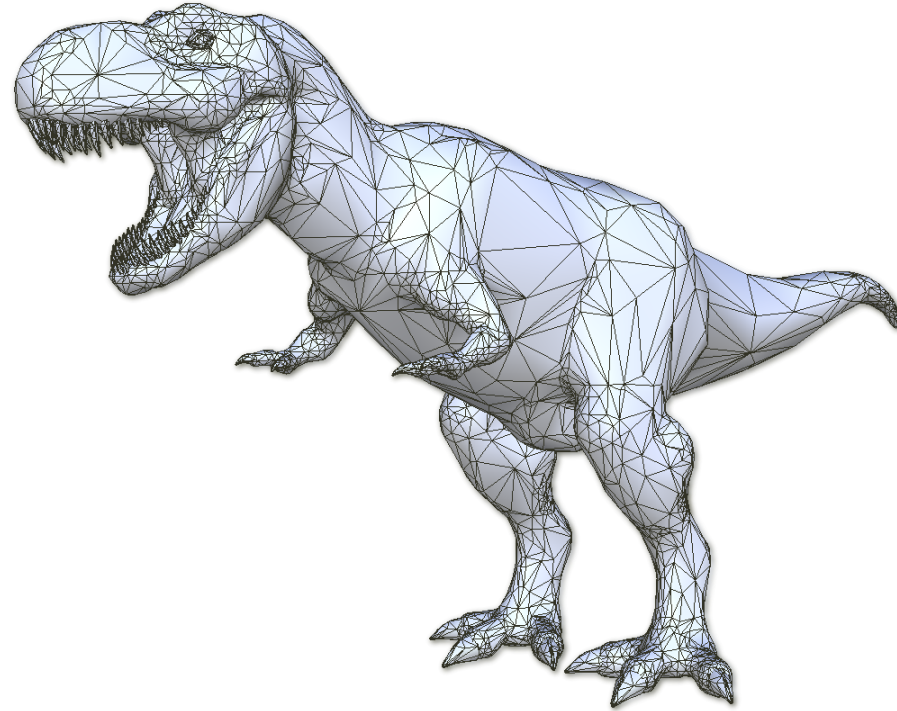
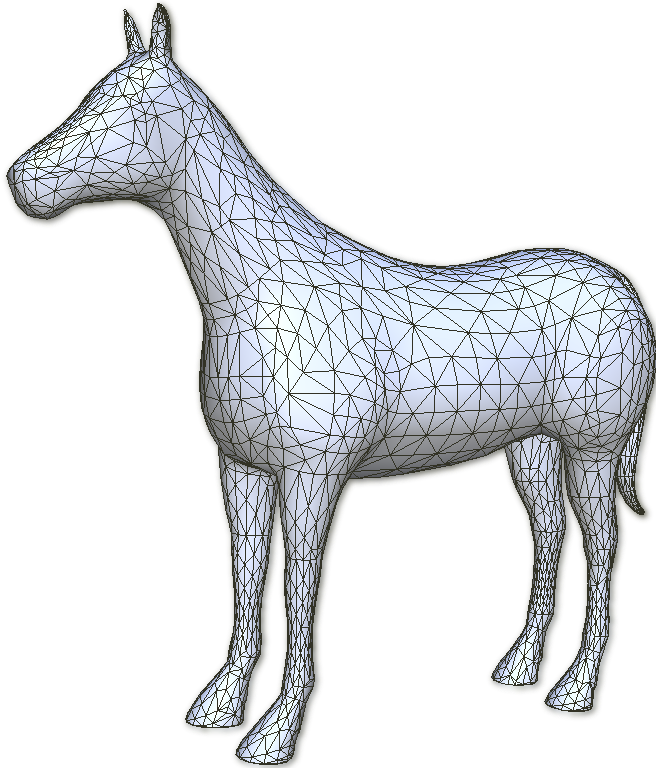
3D shape graph



General graphs

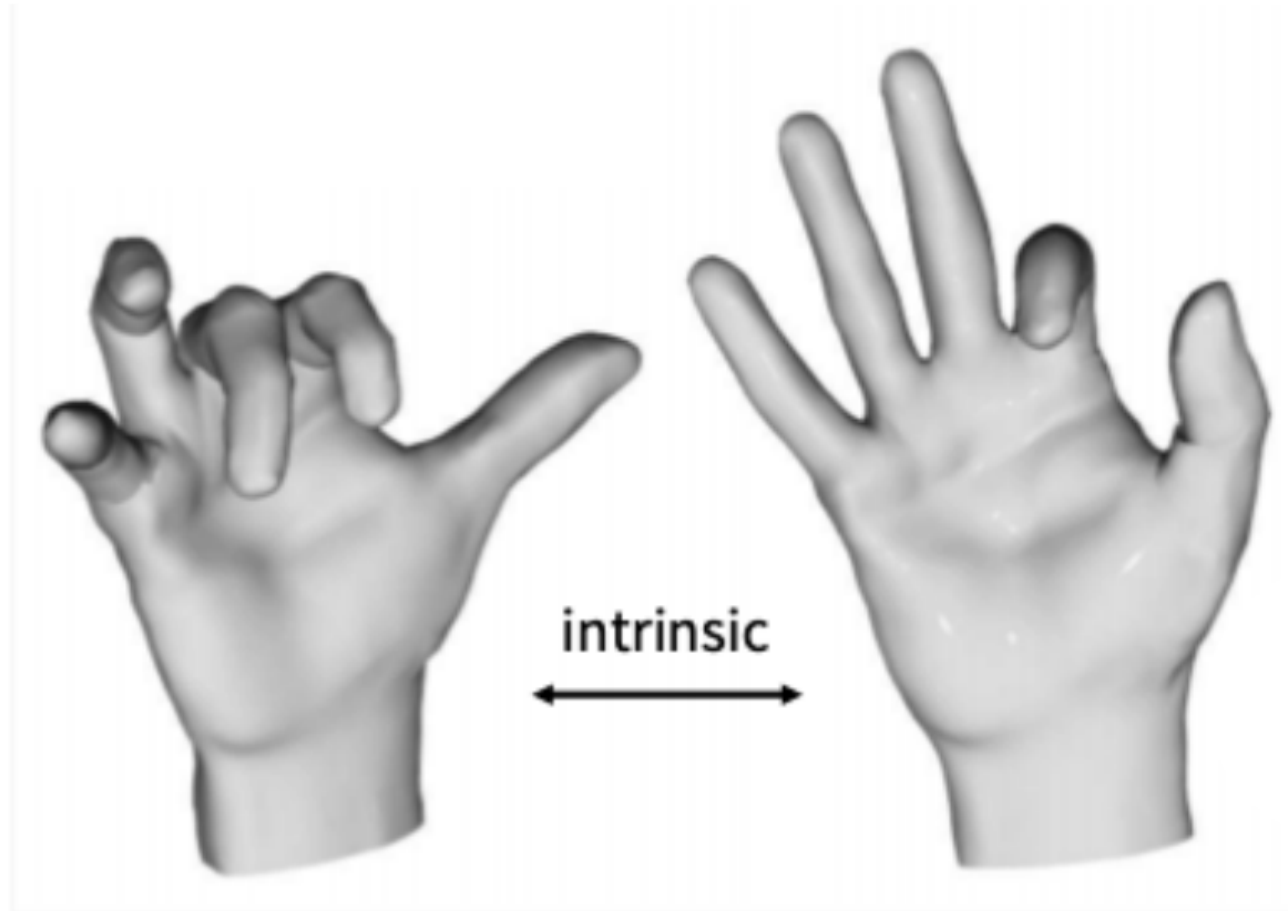
# Mesh representation

- Effective in encoding fine details

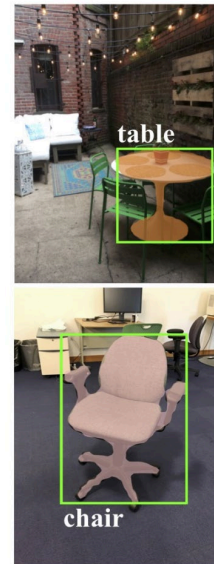
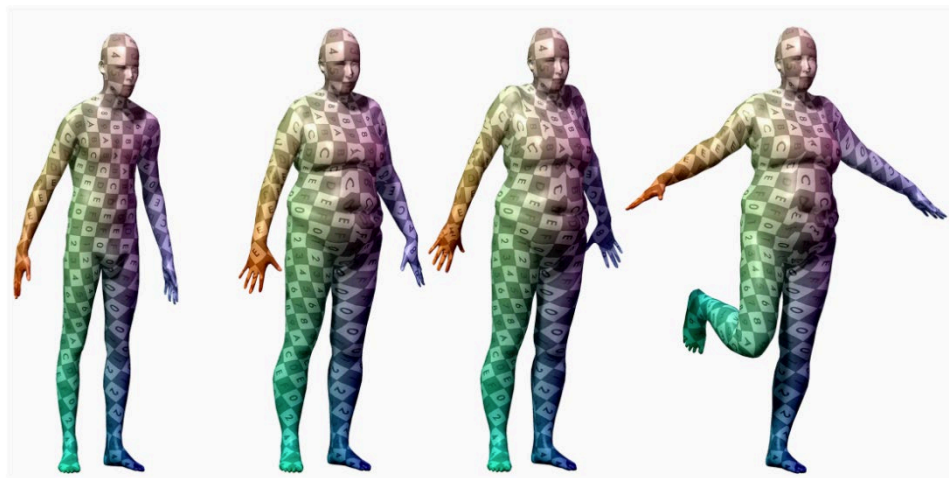
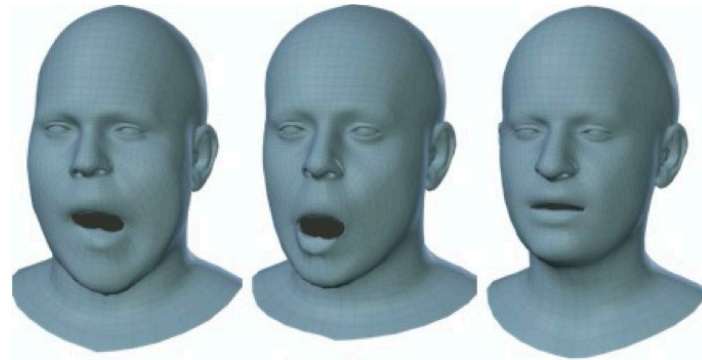
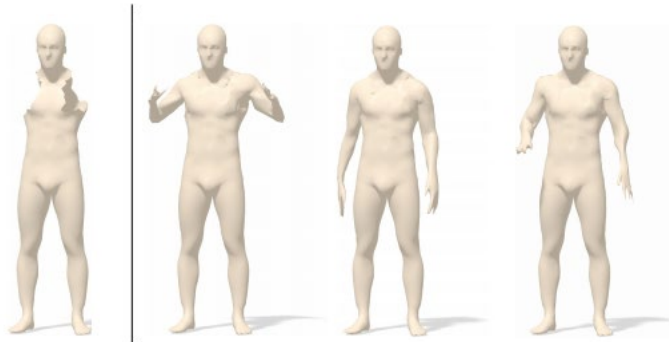


# Mesh representation

- Natural for describing intrinsic similarity



# Applications on shapes

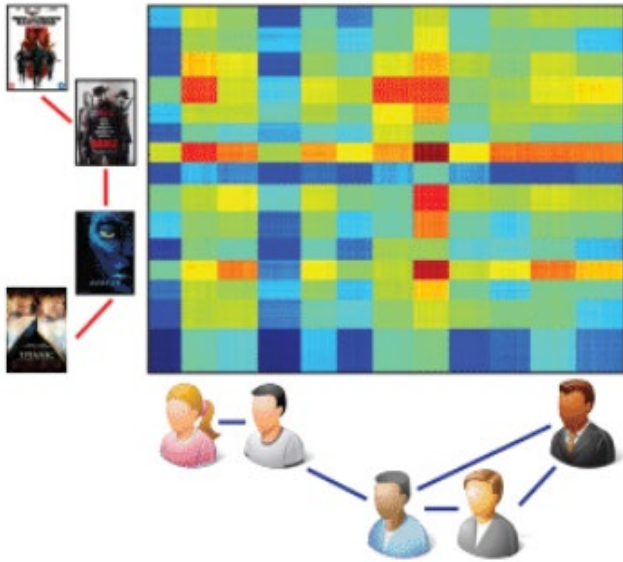


# Graph representation

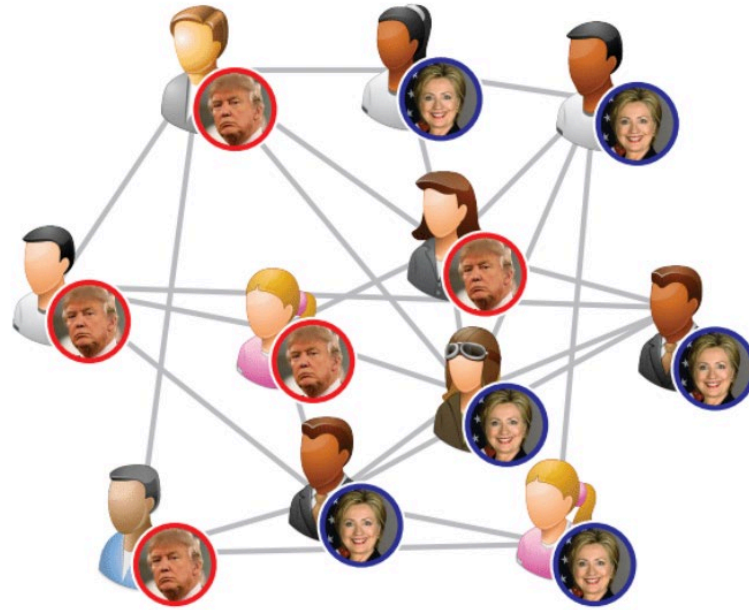


Graphs

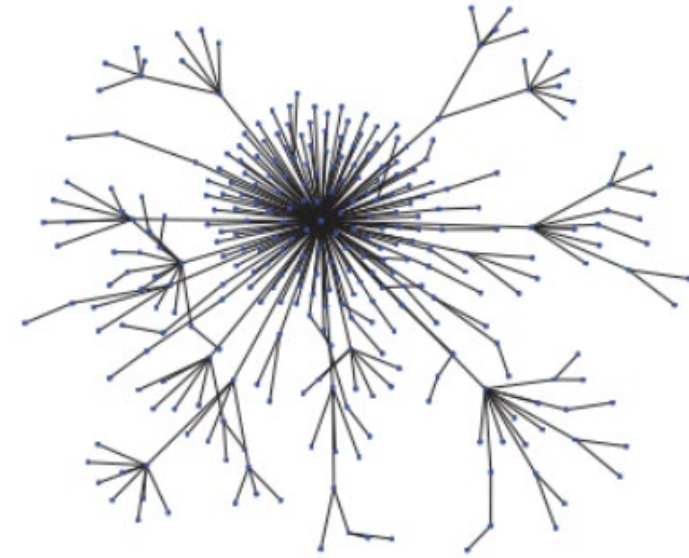
# Application on graphs



Recommendation systems



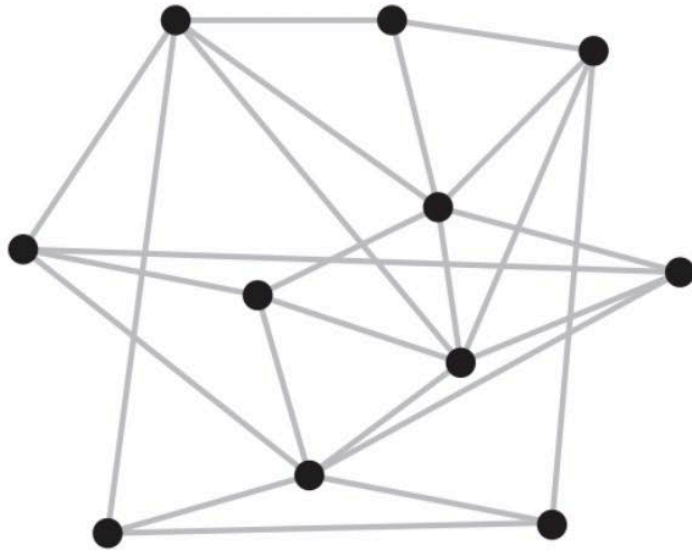
Labeling



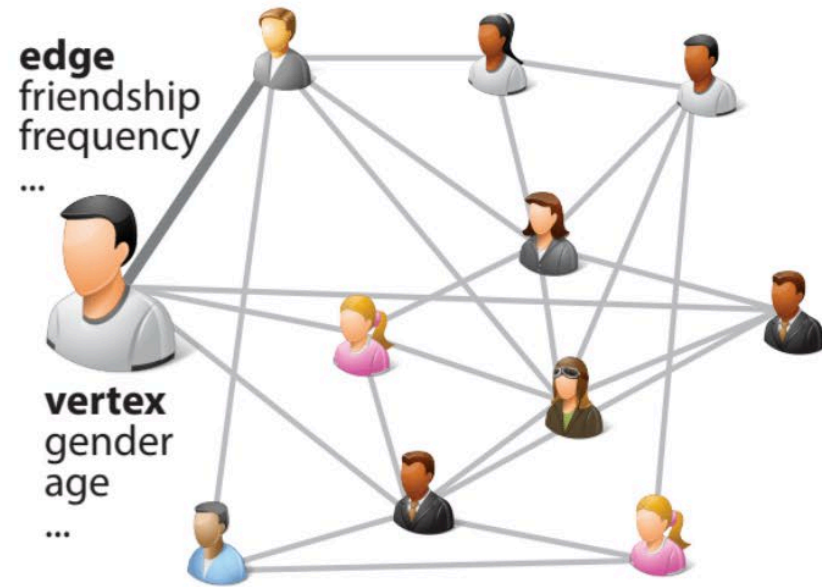
Fake news detection

# Challenges

- Domain structure vs. Data on a domain



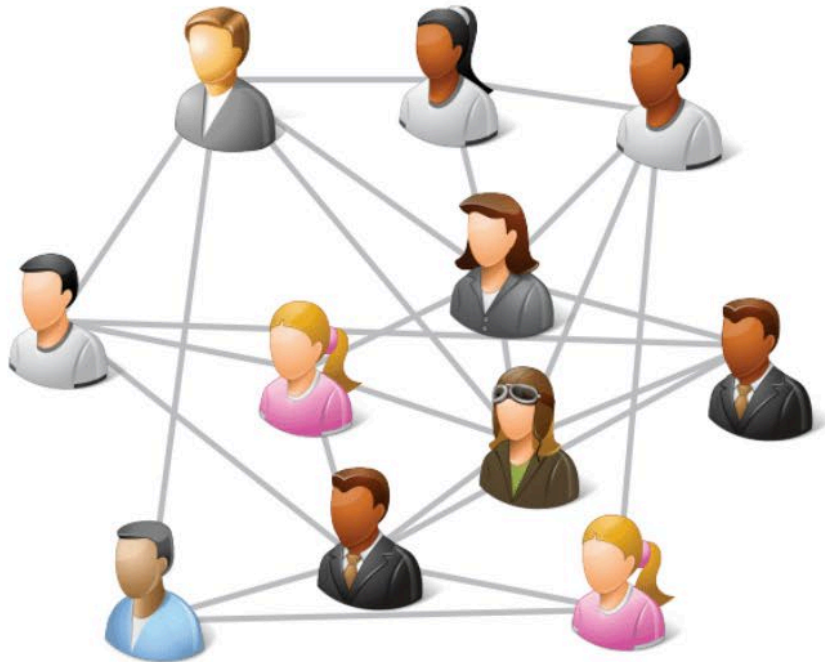
Domain structure



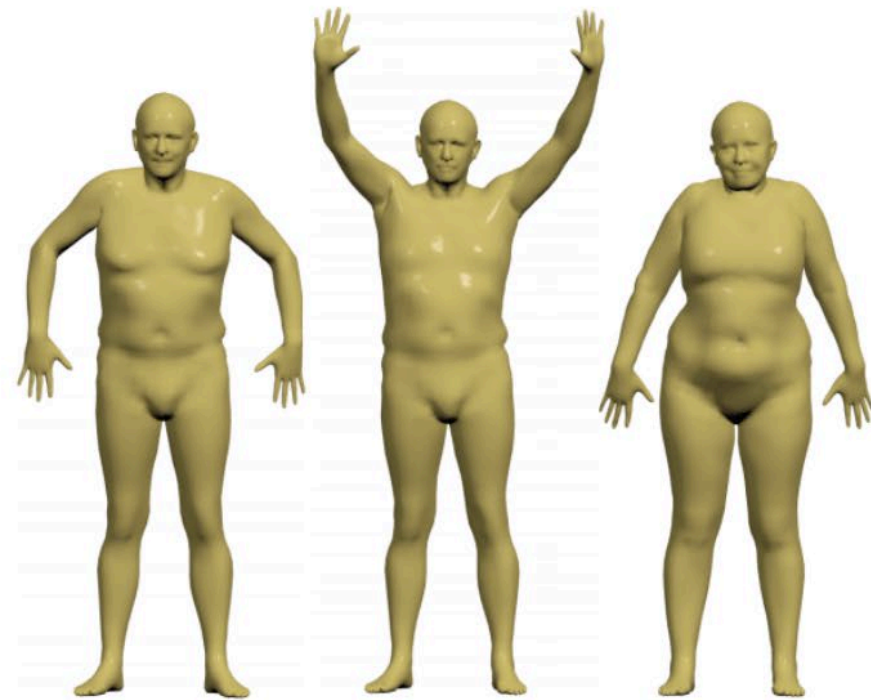
Data on a domain

# Challenges

- Fixed vs different domain



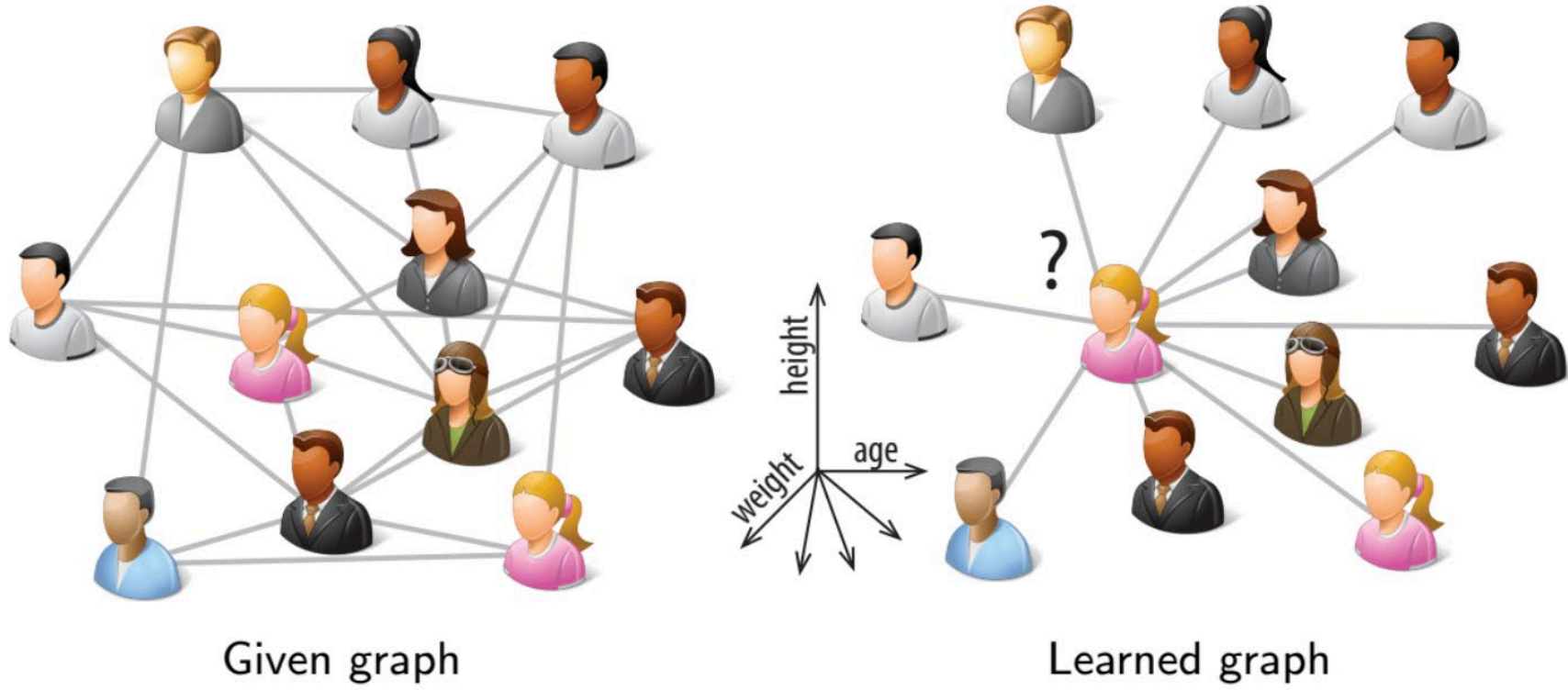
Social network  
(fixed graph)



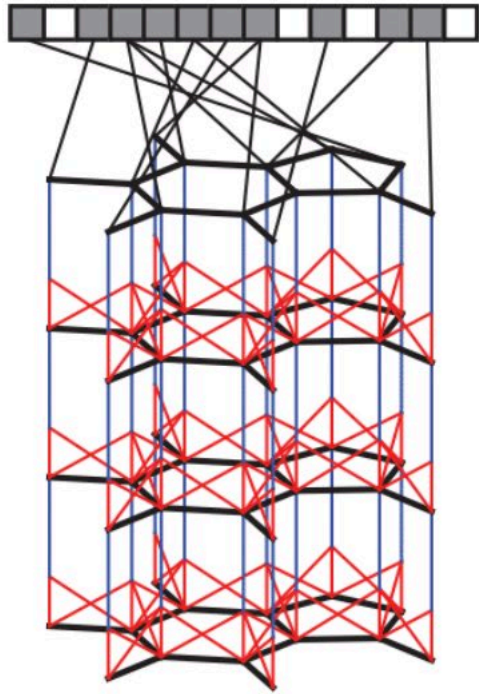
3D shapes  
(different manifolds)

# Challenges

- Known vs unknown graph



# Global tasks

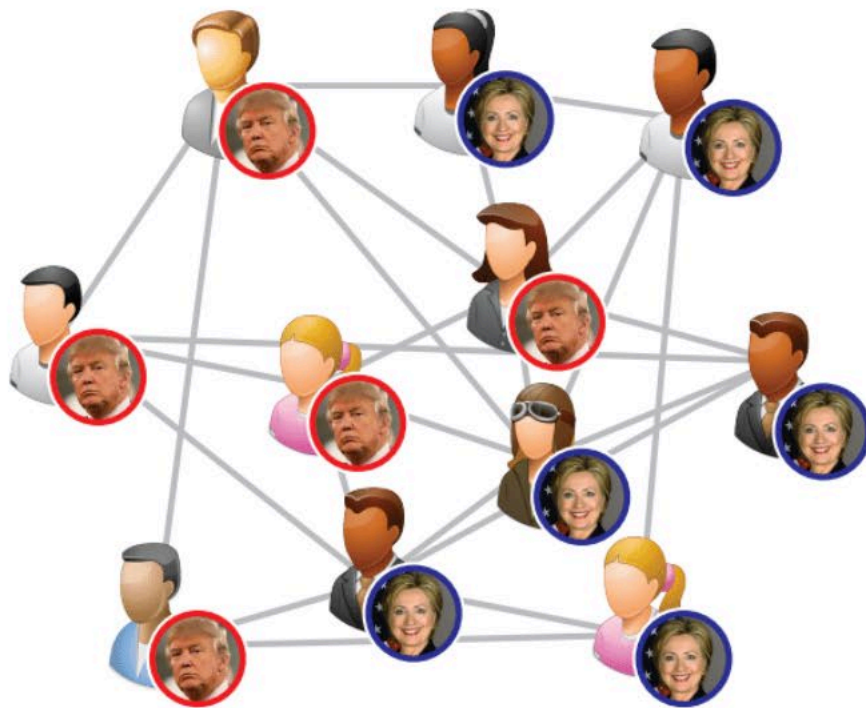


Molecule graph

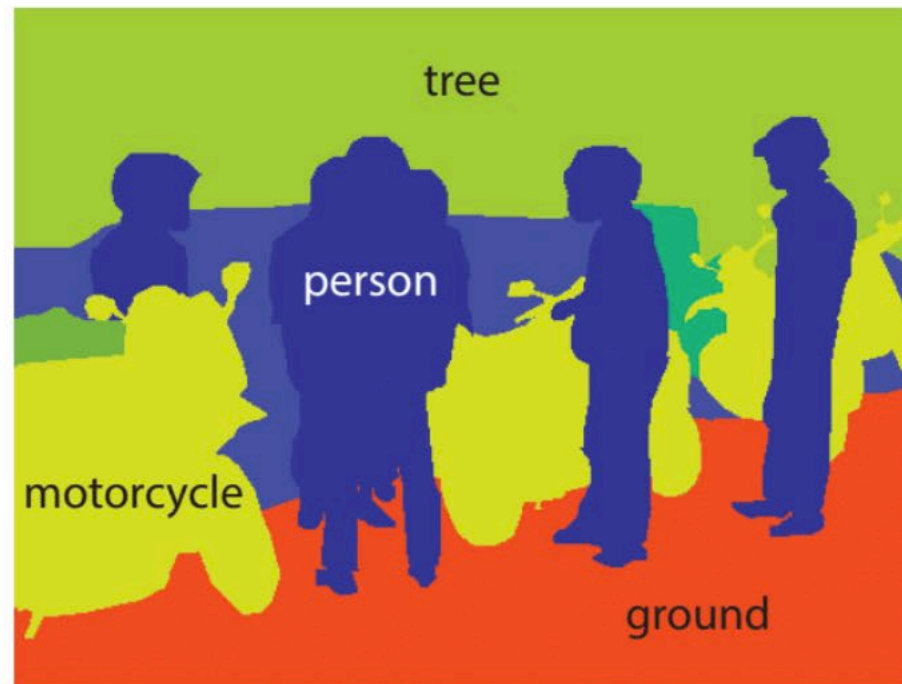


Image recognition

# Vertex-wise tasks



Social network



Semantic image segmentation

# Different formulations of non-Euclidean CNNs

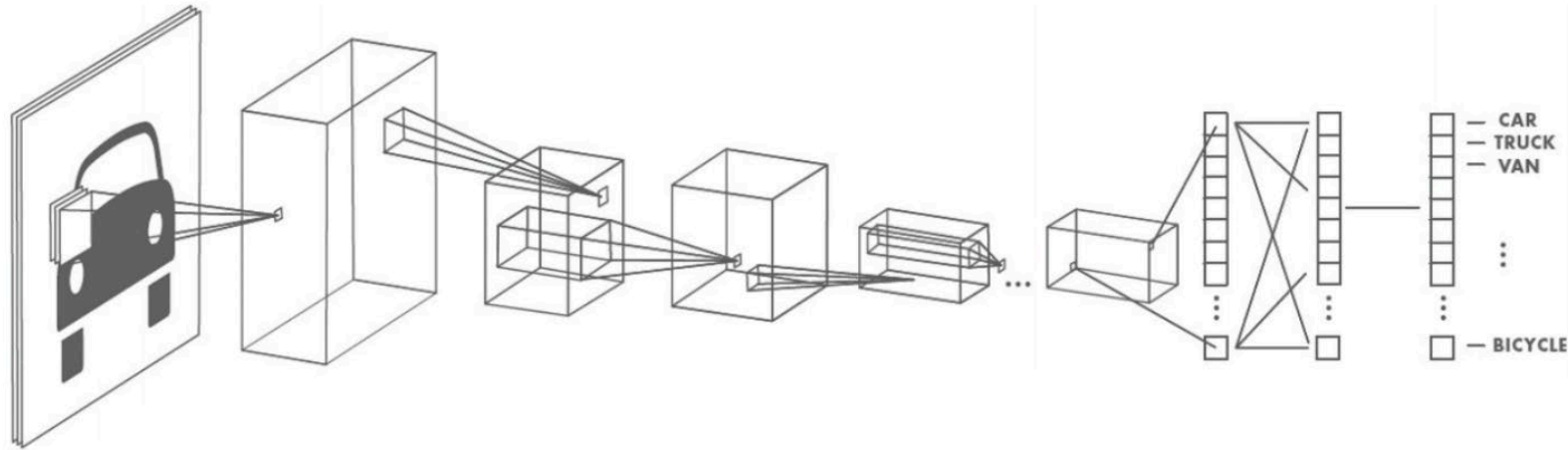


Spectral domain



Spatial domain

# Key properties of CNN



- ☺ Convolutional filters (**Translation invariance+Self-similarity**)
- ☺ Multiple layers (**Compositionality**)
- ☺ Filters localized in space (**Locality**)
- ☺  $\mathcal{O}(1)$  parameters per filter (independent of input image size  $n$ )
- ☺  $\mathcal{O}(n)$  complexity per layer (filtering done in the spatial domain)
- ☺  $\mathcal{O}(\log n)$  layers in classification tasks

# Spectral methods: Laplacian is key



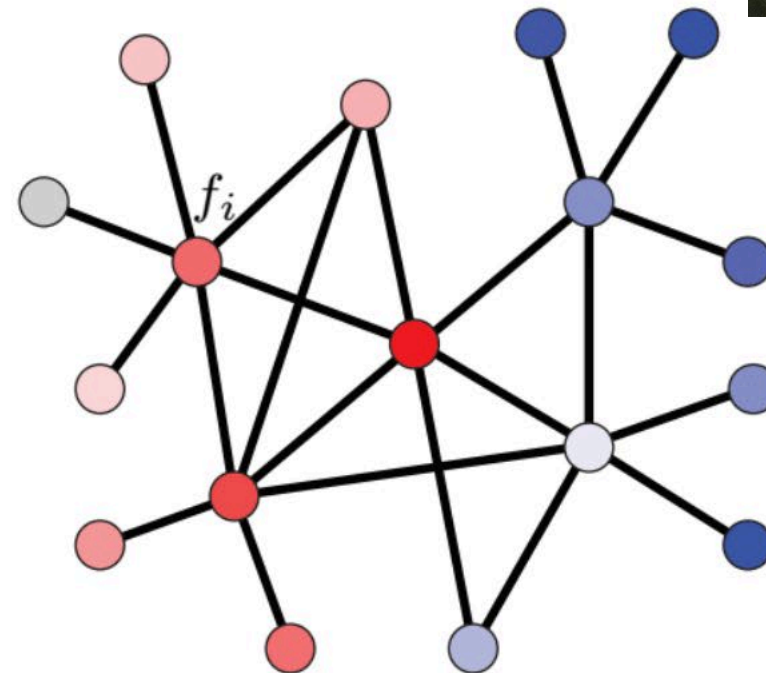
- Weighted undirected graph  $\mathcal{G}$  with vertices  $\mathcal{V} = \{1, \dots, n\}$ , edges  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  and edge weights  $w_{ij} \geq 0$  for  $(i, j) \in \mathcal{E}$

- Functions over the vertices

$L^2(\mathcal{V}) = \{f : \mathcal{V} \rightarrow \mathbb{R}\}$  represented as vectors  $\mathbf{f} = (f_1, \dots, f_n)$

- Hilbert space with inner product

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} f_i g_i = \mathbf{f}^\top \mathbf{g}$$



# Spectral methods: Laplacian is key

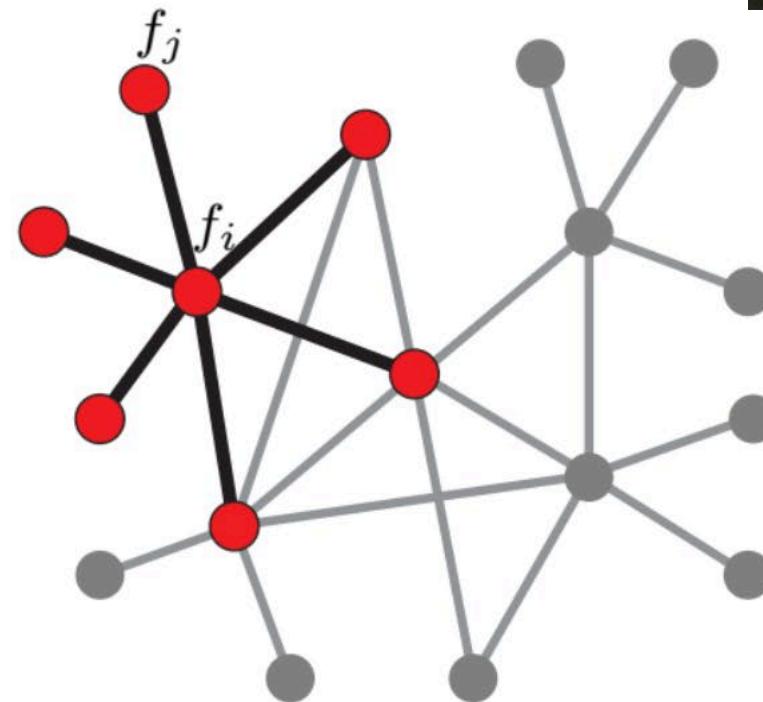


- Unnormalized Laplacian  $\Delta : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{V})$

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

(up to scale) difference between  $f$  and its local average

- Represented as a **positive semi-definite**  $n \times n$  matrix  $\Delta = \mathbf{D} - \mathbf{W}$  where  $\mathbf{W} = (w_{ij})$  and  $\mathbf{D} = \text{diag}(\sum_{j \neq i} w_{ij})$



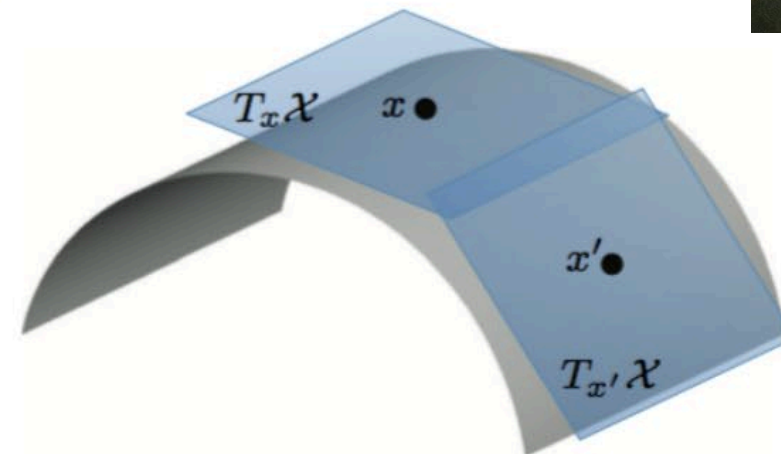
# Spectral methods: Laplacian is key



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

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

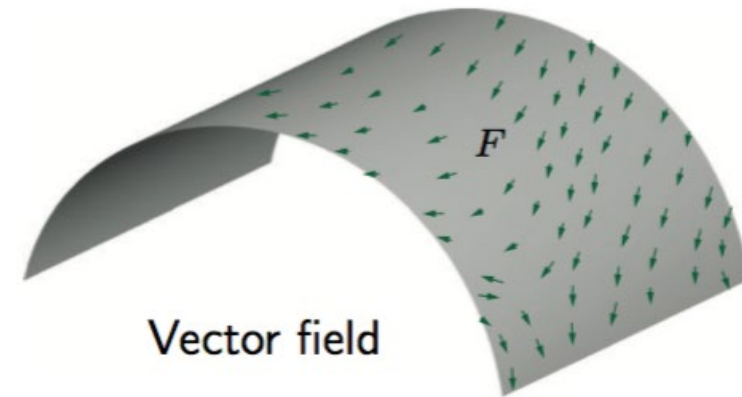
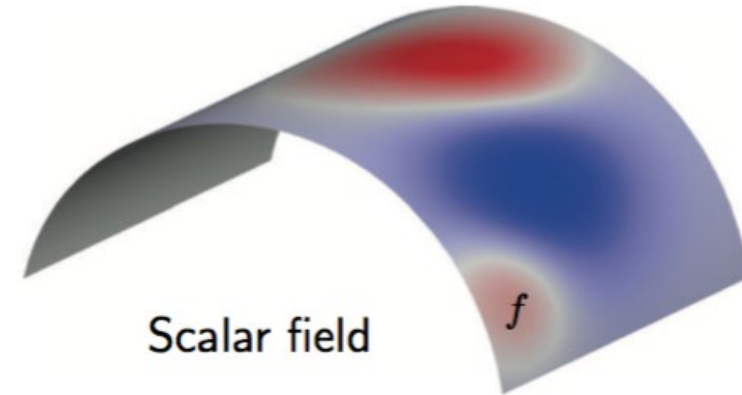
depending smoothly on  $x$



# Spectral methods: Laplacian is key



- Scalar field  $f : \mathcal{X} \rightarrow \mathbb{R}$
- Vector field  $F : \mathcal{X} \rightarrow T\mathcal{X}$



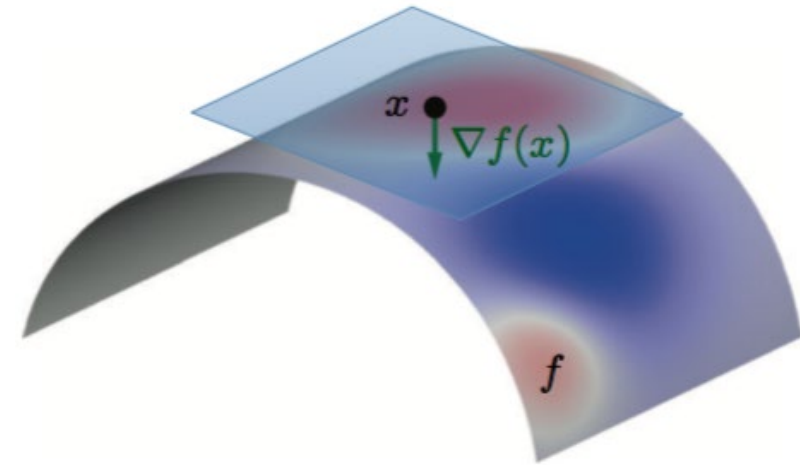
# Spectral methods: Laplacian is key



- Intrinsic gradient operator

$$\nabla f : L^2(\mathcal{X}) \rightarrow L^2(T\mathcal{X})$$

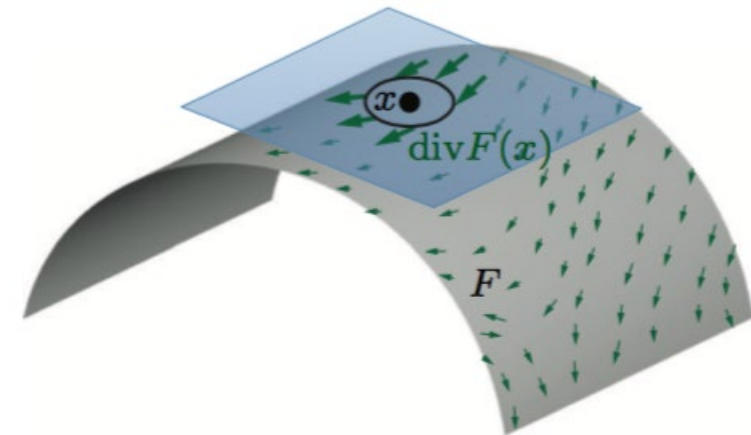
“direction of steepest change of  $f$ ”



- Intrinsic divergence operator

$$\text{div} : L^2(T\mathcal{X}) \rightarrow L^2(\mathcal{X})$$

“net flow of field  $F$  at  $x$ ”



# Spectral methods: Laplacian is key

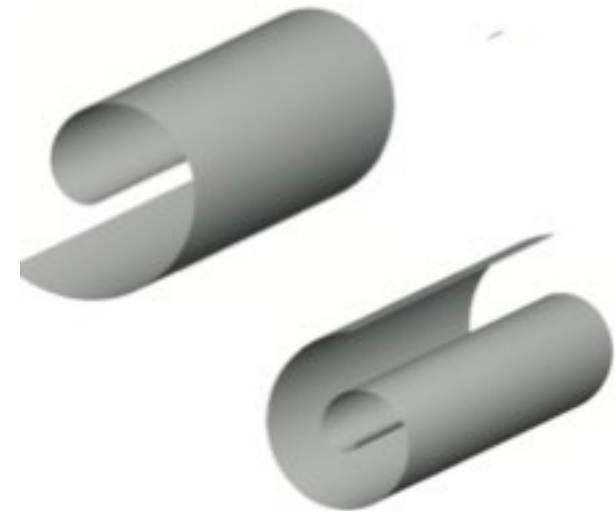
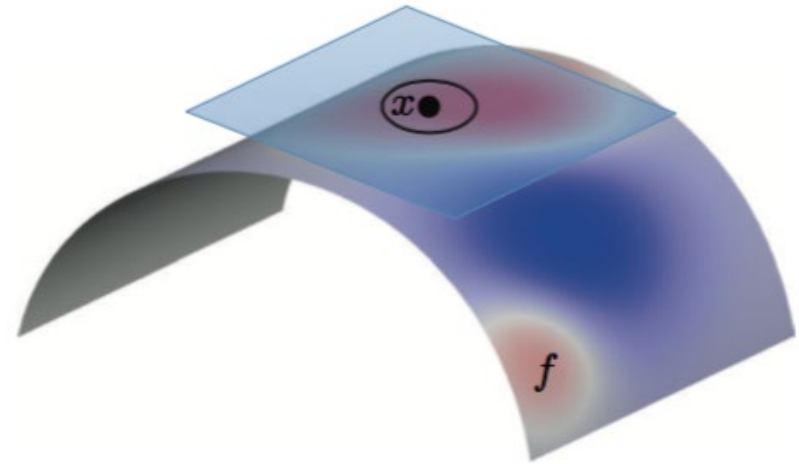


- Laplacian  $\Delta : L^2(\mathcal{X}) \rightarrow L^2(\mathcal{X})$

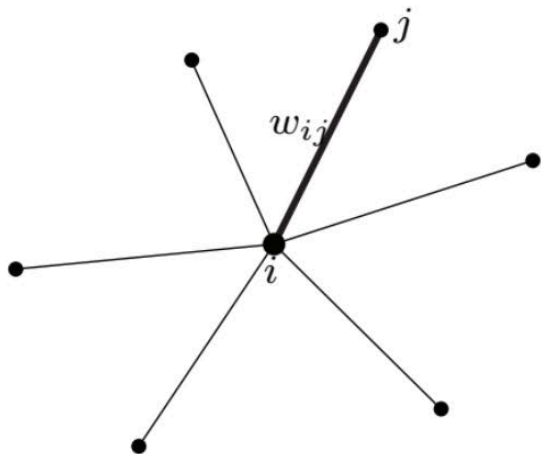
$$\Delta f = -\operatorname{div}(\nabla f)$$

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

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

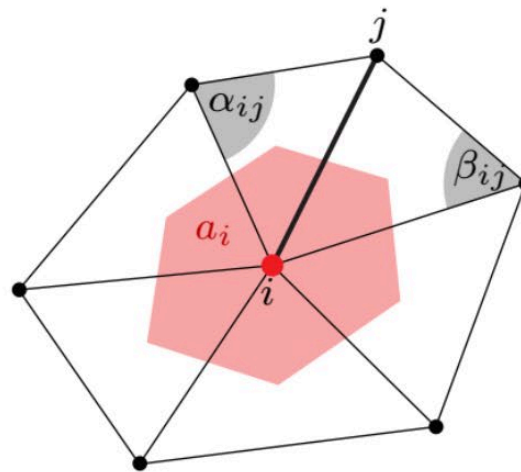


# Going non-Euclidean: Laplacian is key



**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

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

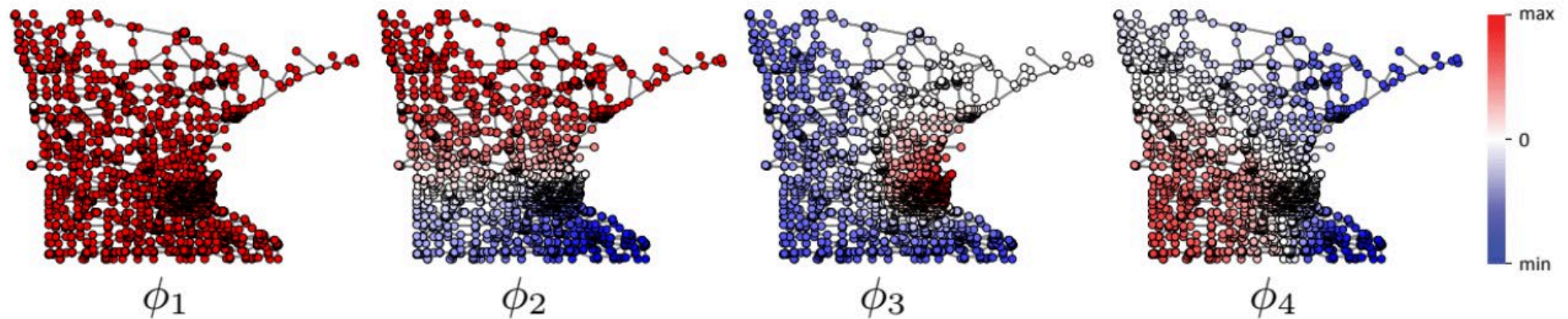
where  $\mathbf{f} = (f_1, \dots, f_n)^\top$ ,  $\mathbf{W}$  is the **stiffness matrix**,  $\mathbf{A} = \text{diag}(a_1, \dots, a_n)$  is the **mass matrix**, and  $\mathbf{D} = \text{diag}(\sum_{j \neq 1} w_{1j}, \dots, \sum_{j \neq n} w_{nj})$

# Going non-Euclidean: Laplacian is key

Eigendecomposition of a graph Laplacian

$$\Delta = \Phi \Lambda \Phi^\top$$

where  $\Phi = (\phi_1, \dots, \phi_n)$  are **orthogonal eigenvectors** ( $\Phi^\top \Phi = \mathbf{I}$ ) and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$  the corresponding **non-negative eigenvalues**



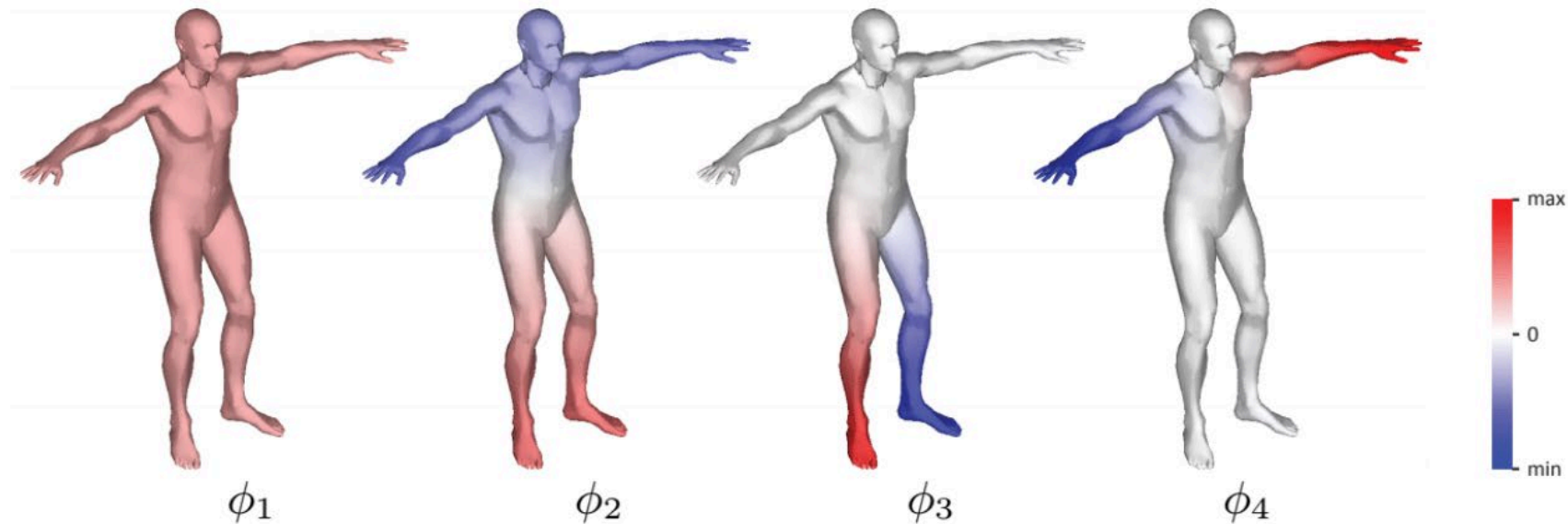
First eigenfunctions of a graph Laplacian

# Going non-Euclidean: Laplacian is key

Eigendecomposition of a graph Laplacian

$$\Delta = \Phi \Lambda \Phi^\top$$

where  $\Phi = (\phi_1, \dots, \phi_n)$  are **orthogonal eigenvectors** ( $\Phi^\top \Phi = \mathbf{I}$ ) and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$  the corresponding **non-negative eigenvalues**

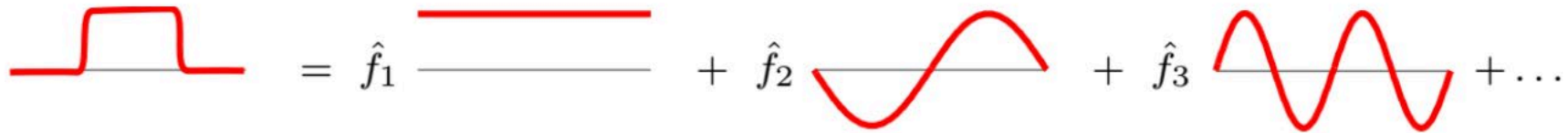


First eigenfunctions of a manifold Laplacian

# Fourier analysis on Euclidean spaces

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

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

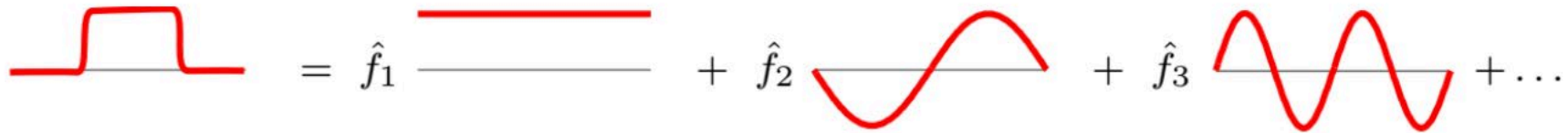


The diagram illustrates the Fourier series expansion of a square wave function. On the left, a red square wave is shown. This is followed by an equals sign and a series of terms: a constant term  $\hat{f}_1$  (represented by a horizontal line), a sine wave term  $\hat{f}_2$  (represented by a single cycle of a sine wave), a higher-order sine wave term  $\hat{f}_3$  (represented by two cycles of a sine wave), and an ellipsis  $+\dots$ .

# Fourier analysis on Euclidean spaces

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

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

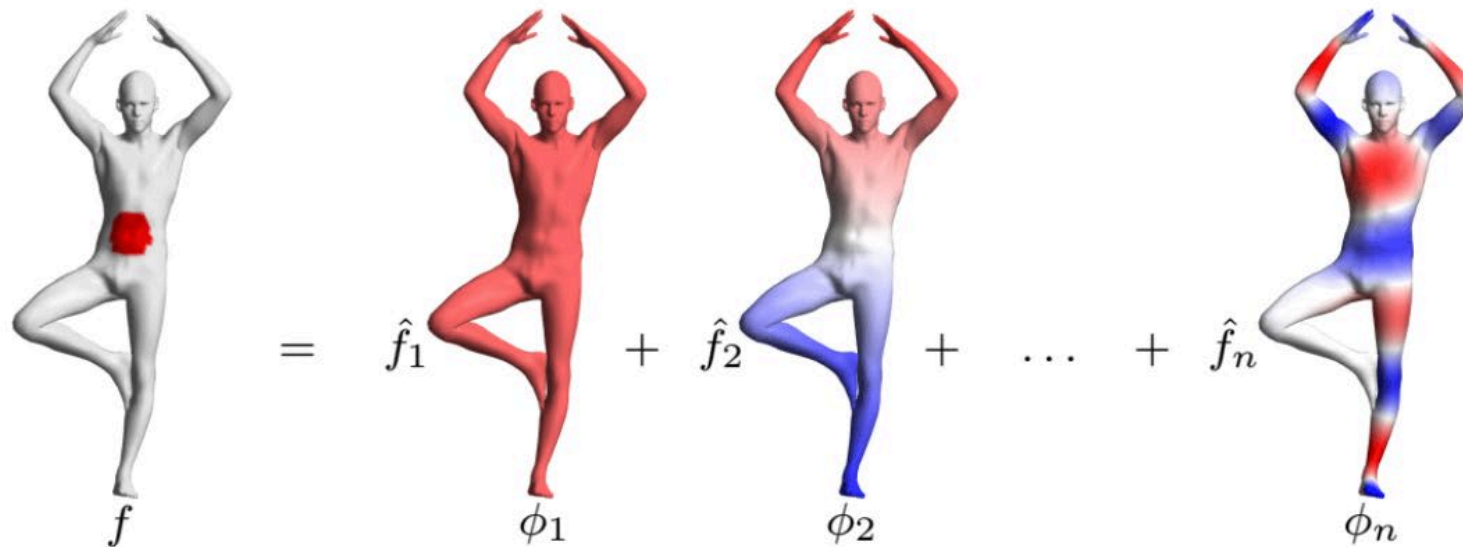


Fourier basis = **Laplacian eigenfunctions**:  $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

# Fourier analysis on graphs and manifolds

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

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



Fourier basis = **Laplacian eigenfunctions**:  $\Delta \phi_k = \lambda_k \phi_k$

# Convolution: Euclidean space

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

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

- **Shift-invariance:**  $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
- **Convolution 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

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

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

# Convolution: Euclidean space

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

$$\begin{aligned}\mathbf{f} \star \mathbf{g} &= \begin{bmatrix} g_1 & g_2 & \dots & \dots & g_n \\ g_n & g_1 & g_2 & \dots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \dots & g_1 & g_2 \\ g_2 & g_3 & \dots & \dots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \\ &= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f}\end{aligned}$$

# Convolution: Euclidean space

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

$$\begin{aligned} \mathbf{f} \star \mathbf{g} &= \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \\ &= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & & & \\ & \ddots & & & \\ & & \hat{g}_n & & \end{bmatrix} \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix} \end{aligned}$$

# Convolution: Euclidean space

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

$$\begin{aligned} \mathbf{f} \star \mathbf{g} &= \begin{bmatrix} g_1 & g_2 & \dots & \dots & g_n \\ g_n & g_1 & g_2 & \dots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \dots & g_1 & g_2 \\ g_2 & g_3 & \dots & \dots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \\ &= \Phi \begin{bmatrix} \hat{f}_1 \cdot \hat{g}_1 \\ \vdots \\ \hat{f}_n \cdot \hat{g}_n \end{bmatrix} \end{aligned}$$

# Spectral convolution

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

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

# 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

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

In matrix-vector notation

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

# Spectral convolution

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

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

In matrix-vector notation

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

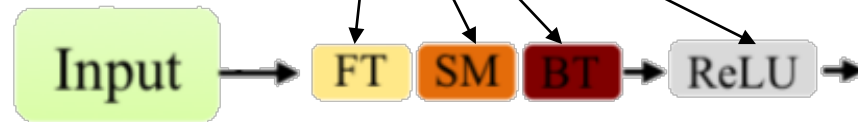
- Not shift-invariant! ( $\mathbf{G}$  has no circulant structure)
- Filter coefficients depend on basis  $\phi_1, \dots, \phi_n$

# Spectral CNN

Convolutional layer expressed in the **spectral domain**

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

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

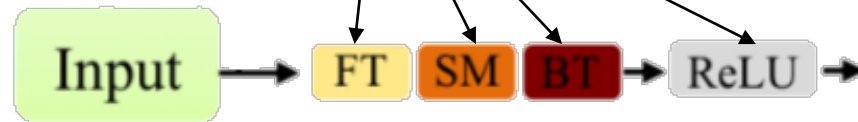


# Spectral CNN

Convolutional layer expressed in the **spectral domain**

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

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



Great! Are we done?

# Spectral CNN

Convolutional layer expressed in the **spectral domain**

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

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

# Localization and smoothness

**Vanishing moments:** In the Euclidean setting

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

**Localization in space = smoothness in frequency domain**

# Localization and smoothness

**Vanishing moments:** In the Euclidean setting

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

**Localization in space = smoothness in frequency domain**

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

Application of the filter

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

# Localization and smoothness

**Vanishing moments:** In the Euclidean setting

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

**Localization in space = smoothness in frequency domain**

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

Application of the parametric filter with learnable parameters  $\alpha$

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

# Spectral gCNN with polynomial filters (ChebNet)

Represent spectral transfer function as a **polynomial** of order  $r$

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

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

# Spectral gCNN with polynomial filters (ChebNet)

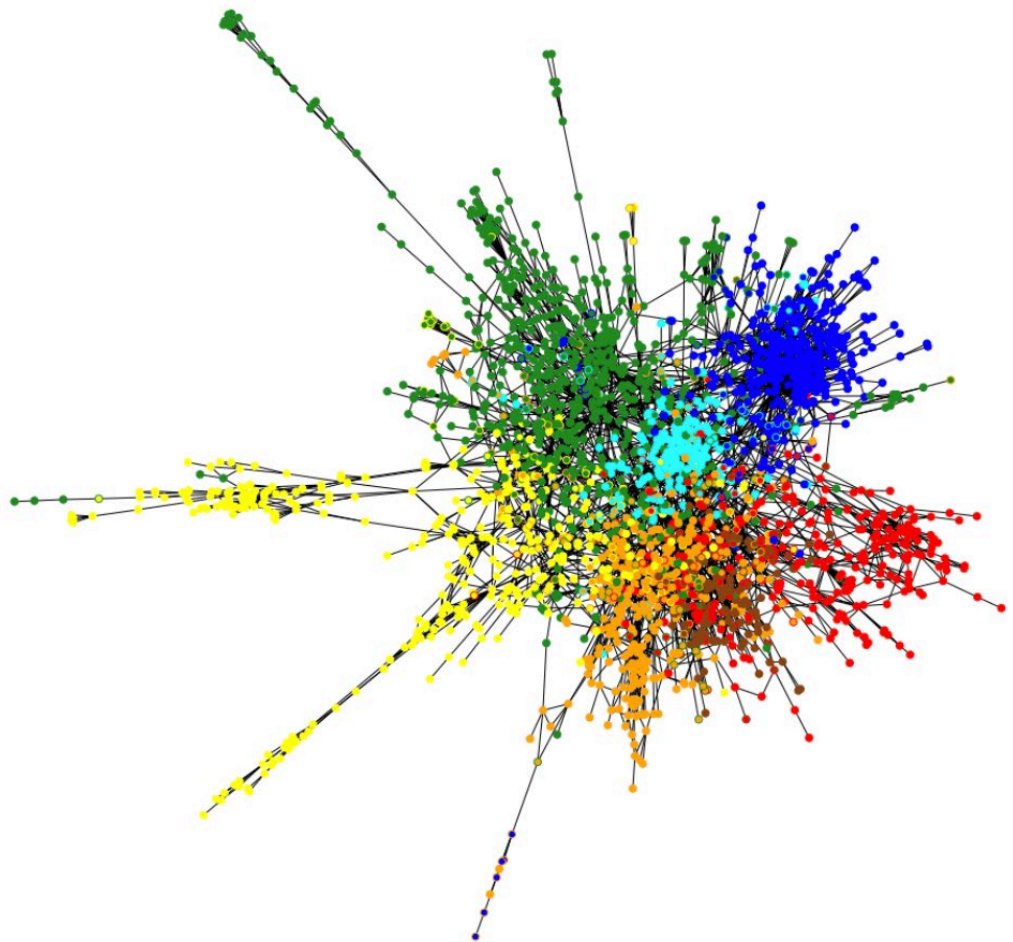
Represent spectral transfer function as a **polynomial** of order  $r$

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

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

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

# Example: citation networks



Method	Cora <sup>1</sup>	PubMed <sup>2</sup>
Manifold Regularization <sup>3</sup>	59.5%	70.7%
Semidefinite Embedding <sup>4</sup>	59.0%	71.1%
Label Propagation <sup>5</sup>	68.0%	63.0%
DeepWalk <sup>6</sup>	67.2%	65.3%
Planetoid <sup>7</sup>	75.7%	77.2%
<b>GCN<sup>8</sup></b>	<b>81.6%</b>	<b>78.7%</b>

# Basis dependence



Function  $f$

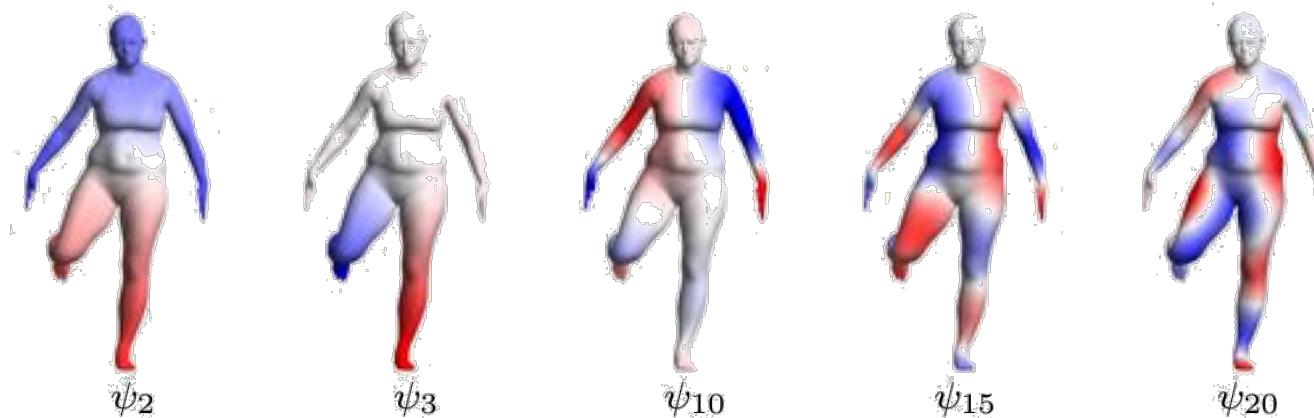
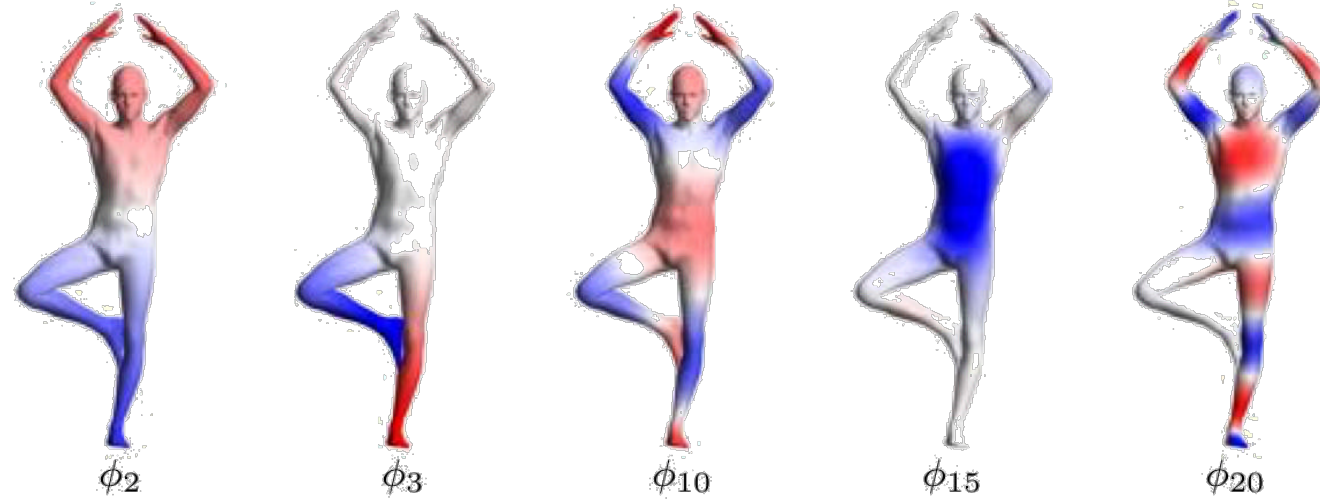


Filtered function  $\tilde{f}$

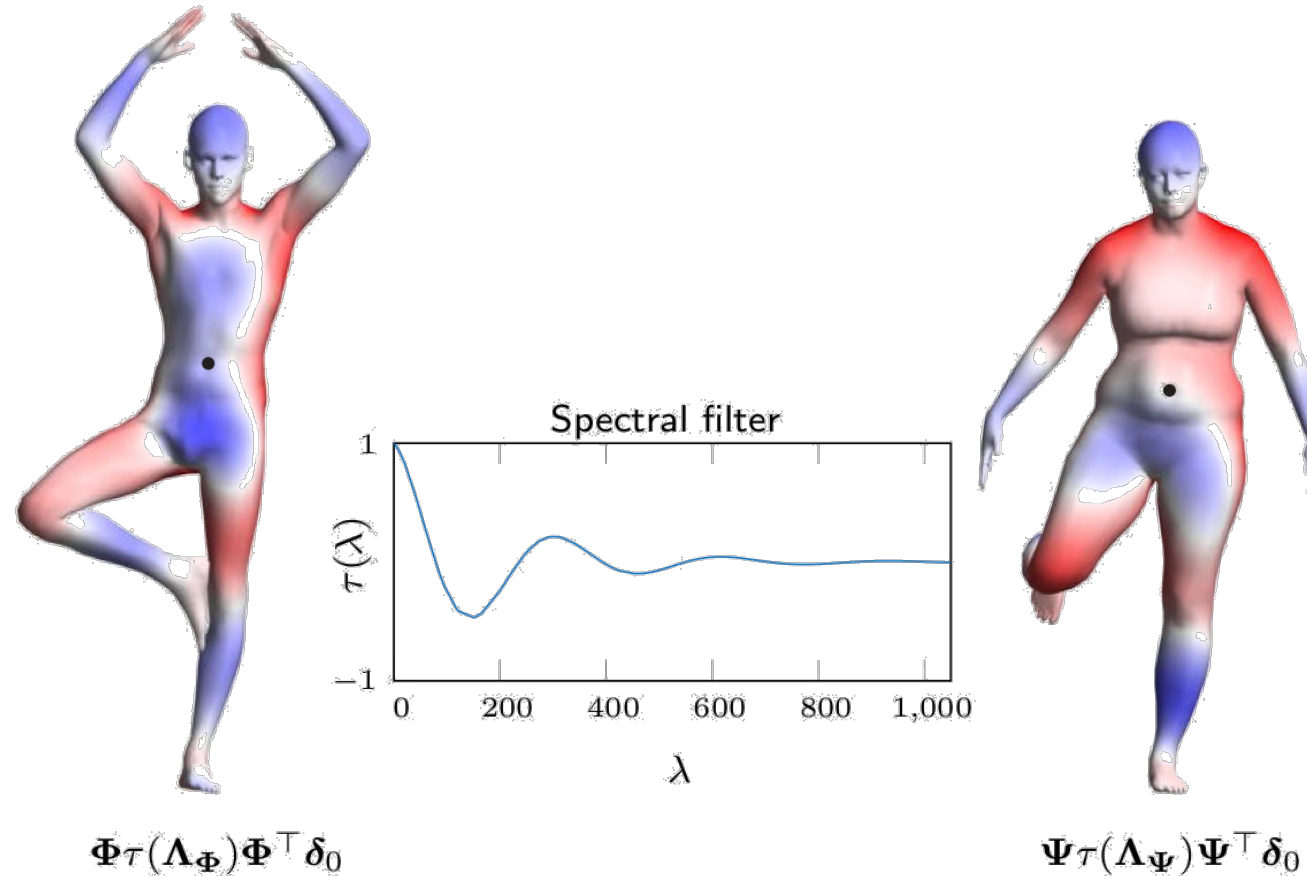


Same function,  
same filter,  
another shape

# Filtering in different bases



# Filtering in different bases



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

# Filtering in different bases



$$\Phi \tau(\Lambda_{\Phi}) \Phi^{\top} \delta_0$$



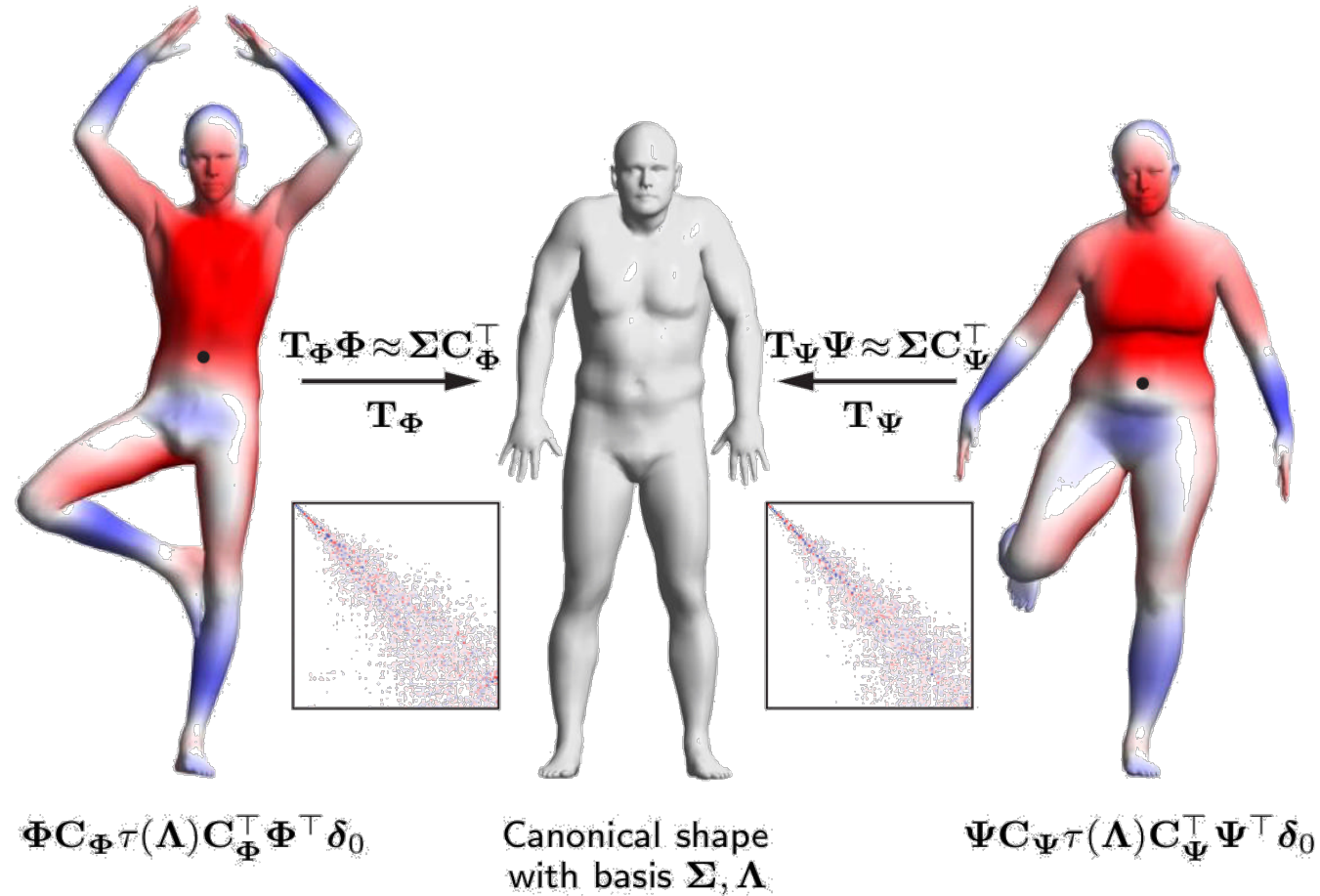
Canonical shape  
with basis  $\Sigma, \Lambda$



$$\Psi \tau(\Lambda_{\Psi}) \Psi^{\top} \delta_0$$

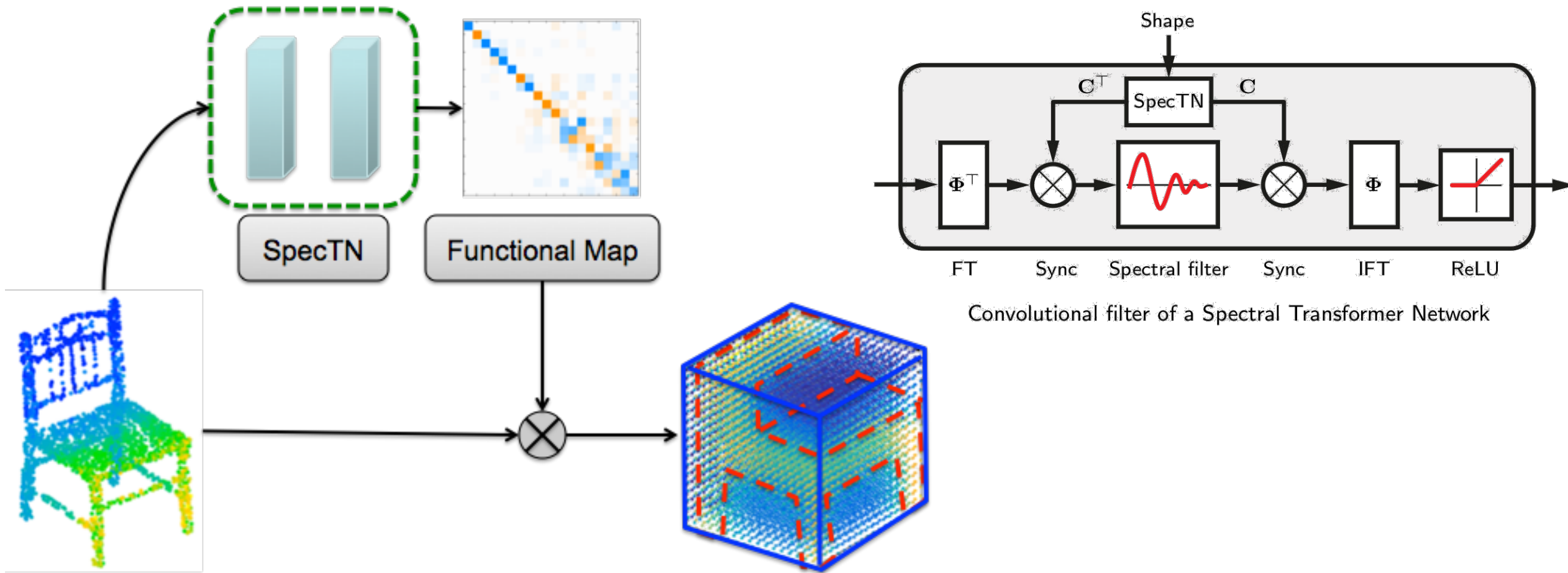
Apply spectral filter  $\tau(\lambda)$  in **different bases**  $\Phi$  and  $\Psi$   
 $\Rightarrow$  **different results!**

# Filtering in different bases



Apply spectral filter  $\tau(\lambda)$  in **synchronized bases**  $\Phi C_\Phi$  and  $\Psi C_\Psi$   
 $\Rightarrow$  **similar results!**

# Spectral transformer network



# Spectral transformer network

- Normal estimation



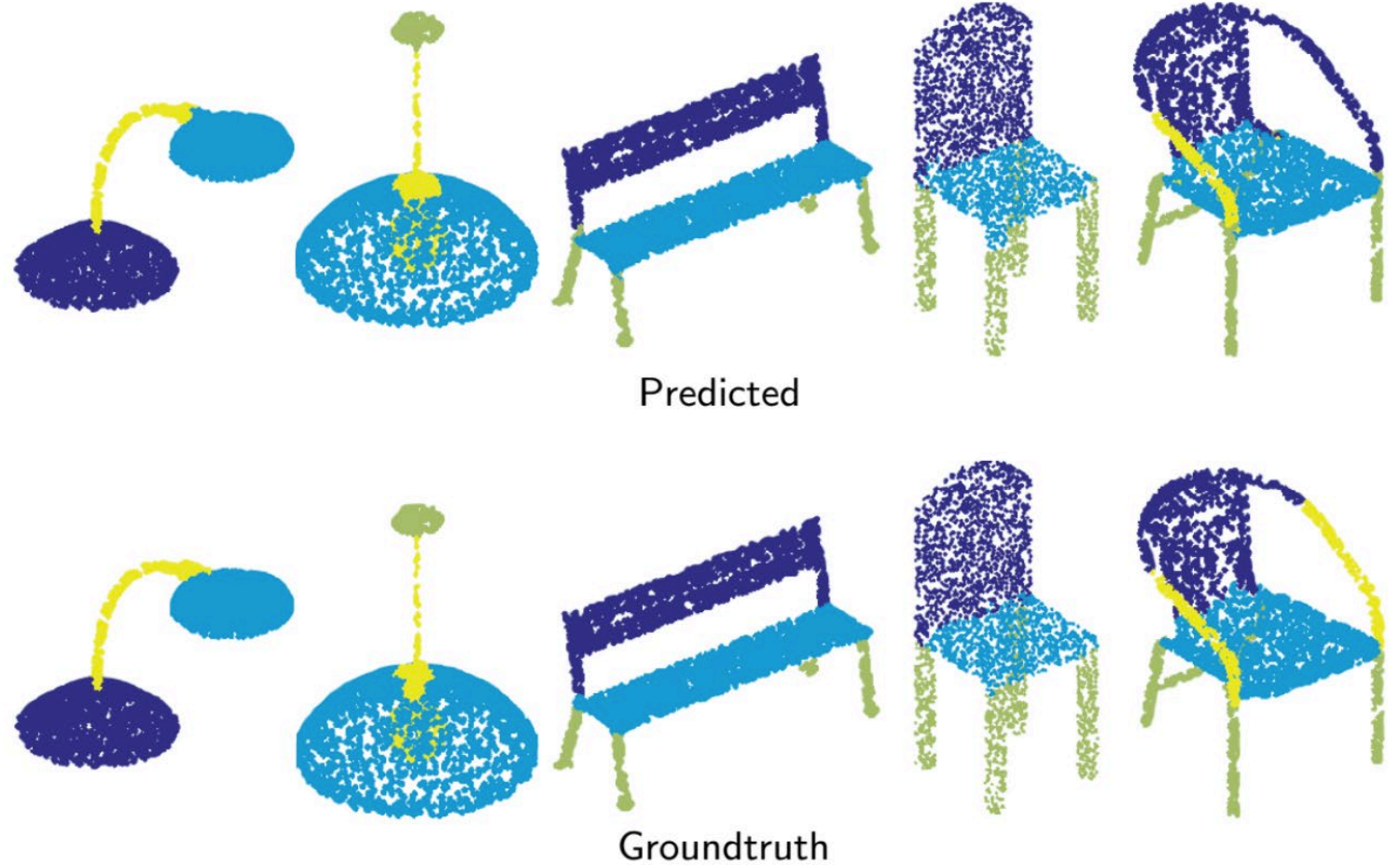
Predicted



Groundtruth

# Spectral transformer network

- Shape segmentation



# Spatial domain (charting-based) CNN

## Euclidean

Spatial domain

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

Spectral domain

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

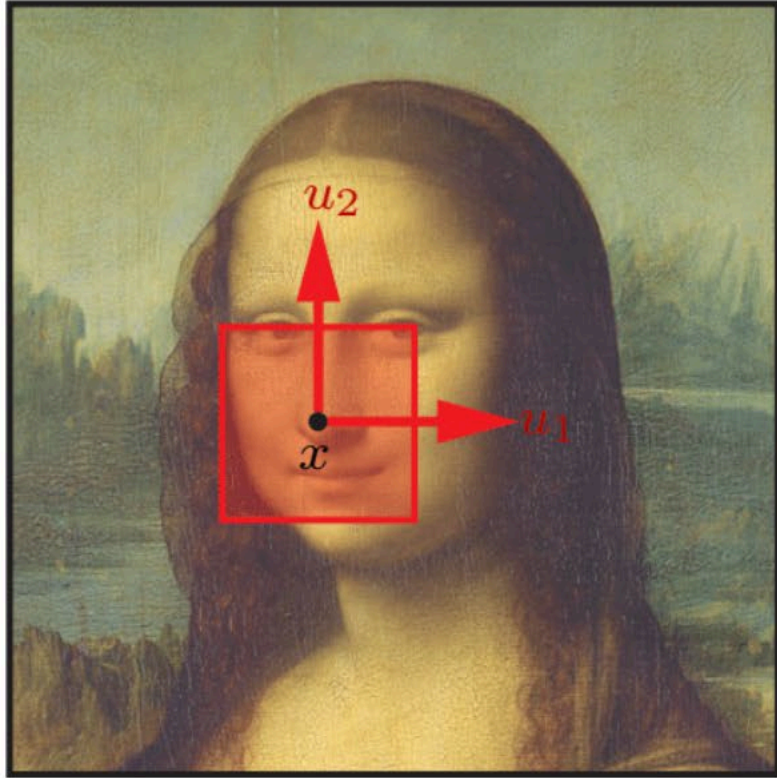
'Convolution Theorem'

## Non-Euclidean

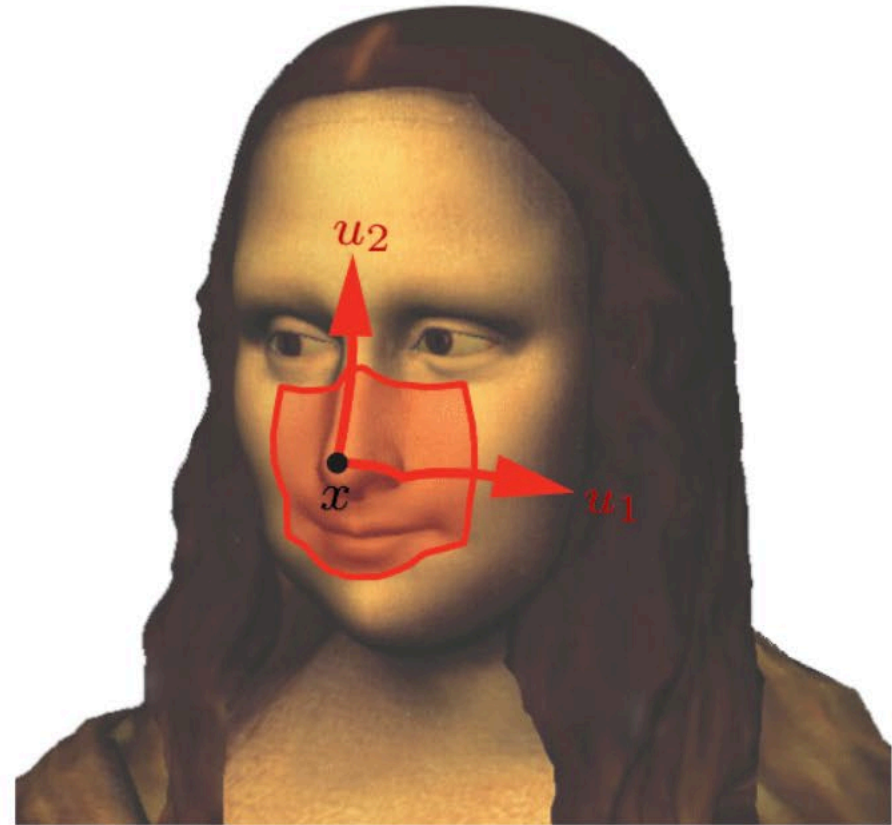
?

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

# Patch operators

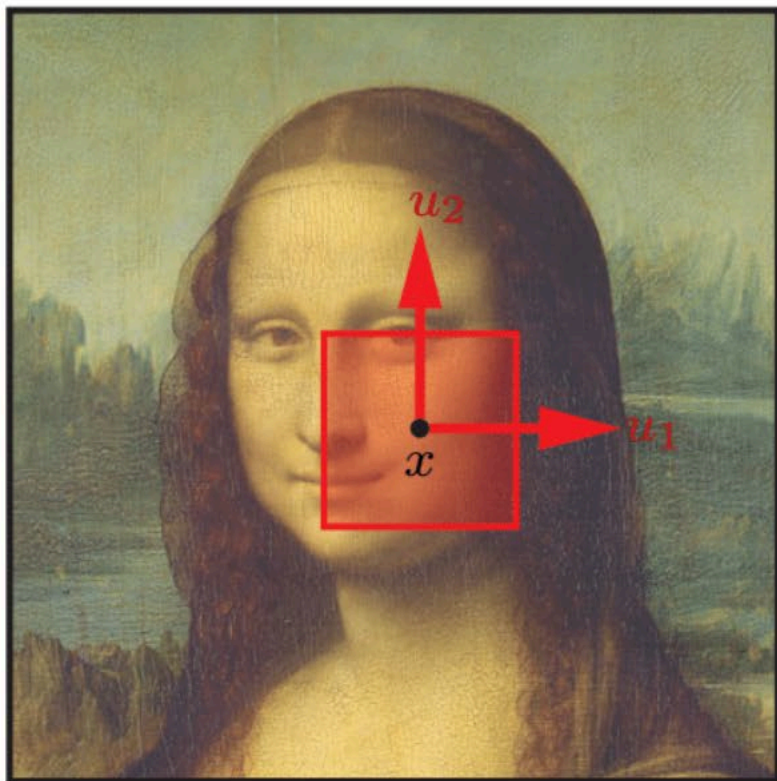


Image

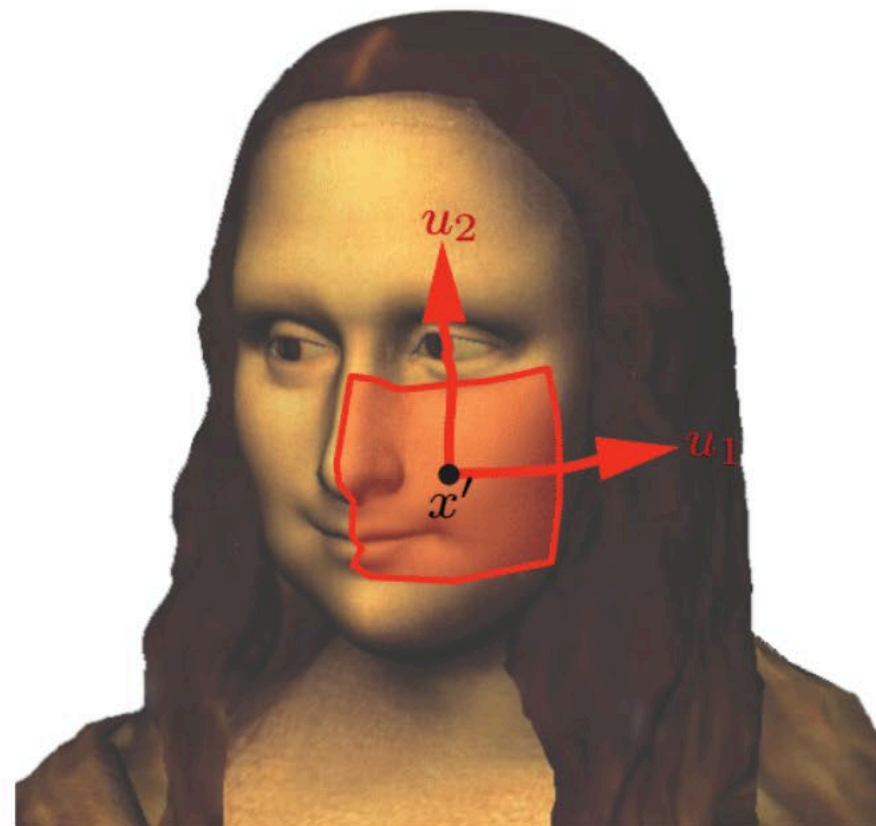


Manifold

# Patch operators



Image



Manifold

# Patch operators

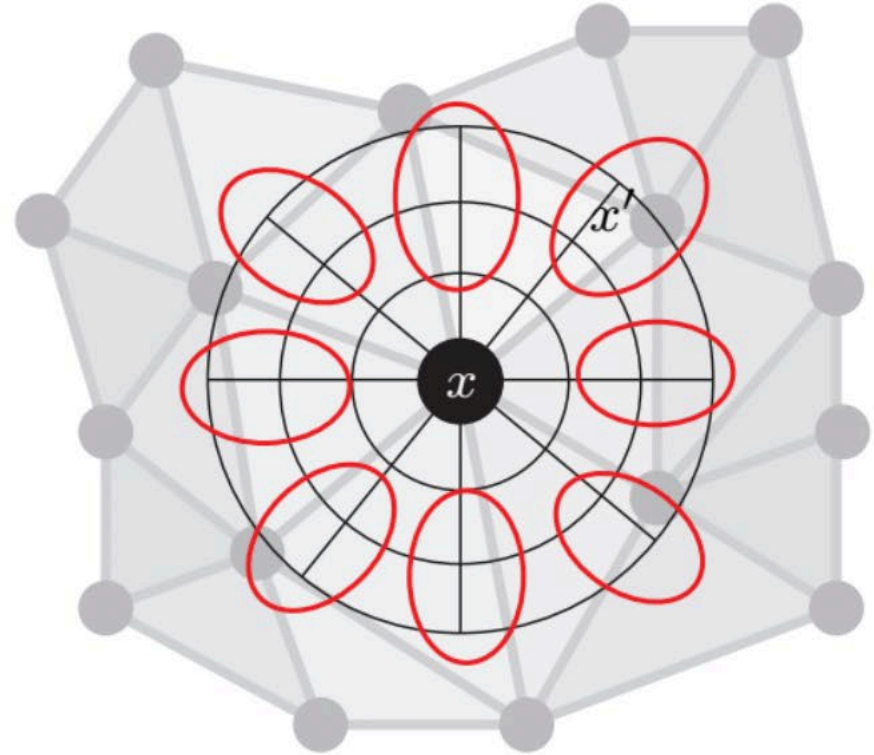
- **Local system of coordinates**  
 $\mathbf{u}(x, x')$  around  $x'$  (e.g. geodesic polar)

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

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

- **Spatial convolution with filter  $g$**

$$(f \star g)(x) \propto \sum_{\ell=1}^L g_\ell \int_{\mathcal{X}} w_\ell(\mathbf{u}(x, x')) f(x') dx'$$



# Patch operators

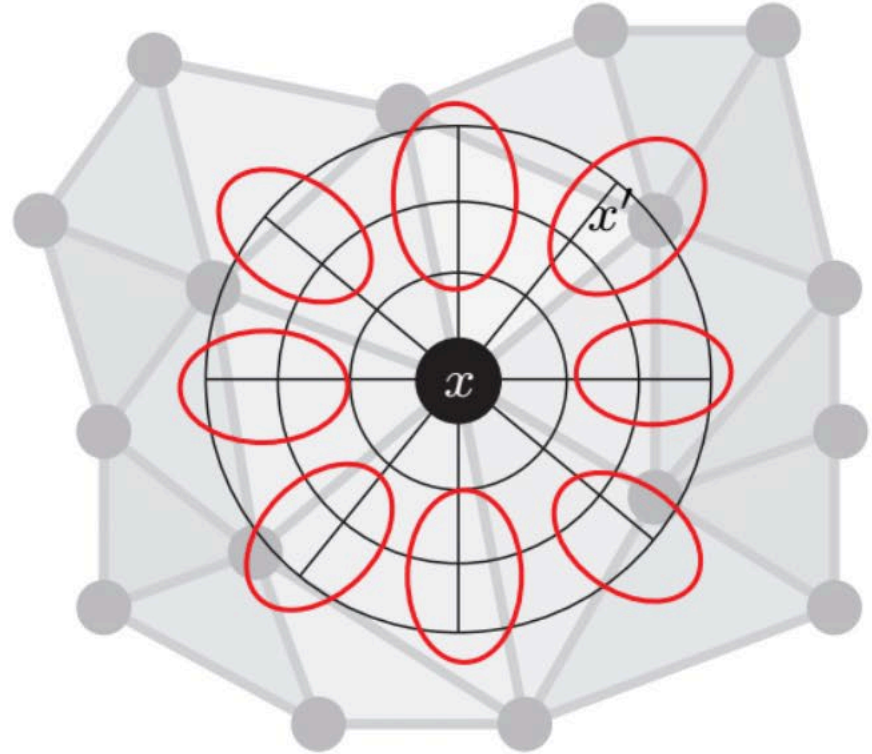
- Local system of coordinates  $\mathbf{u}(x, x')$  around  $x'$  (e.g. geodesic polar)

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

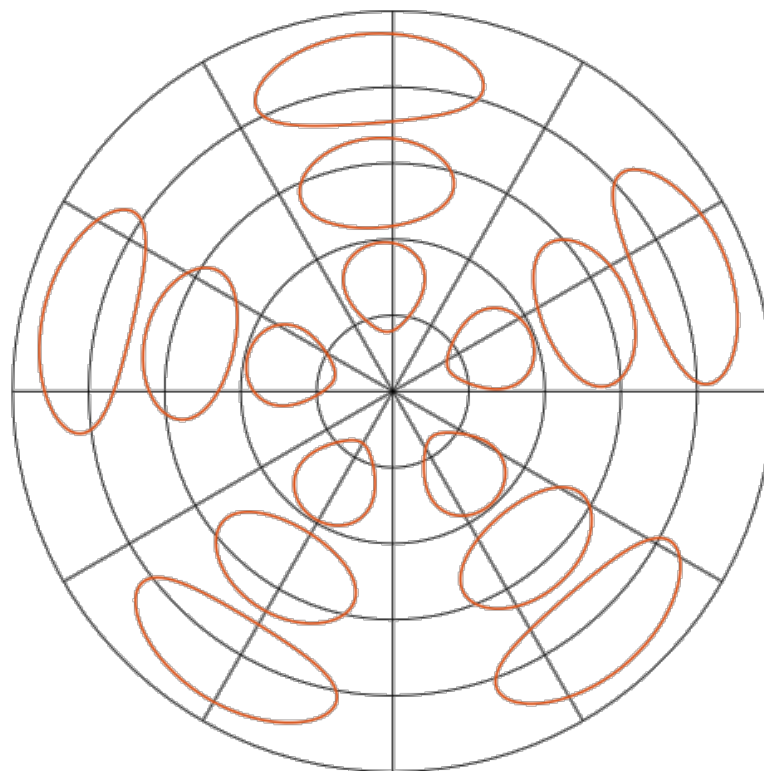
$$w_\ell = \exp\left(-(\mathbf{u} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1} (\mathbf{u} - \boldsymbol{\mu}_\ell)\right)$$

- Spatial convolution with filter  $g$

$$(f \star g)(x) \propto \sum_{\ell=1}^L g_\ell \underbrace{\int_{\mathcal{X}} w_\ell(\mathbf{u}(x, x')) f(x') dx'}_{\text{patch operator}}$$



# Geodesic CNN

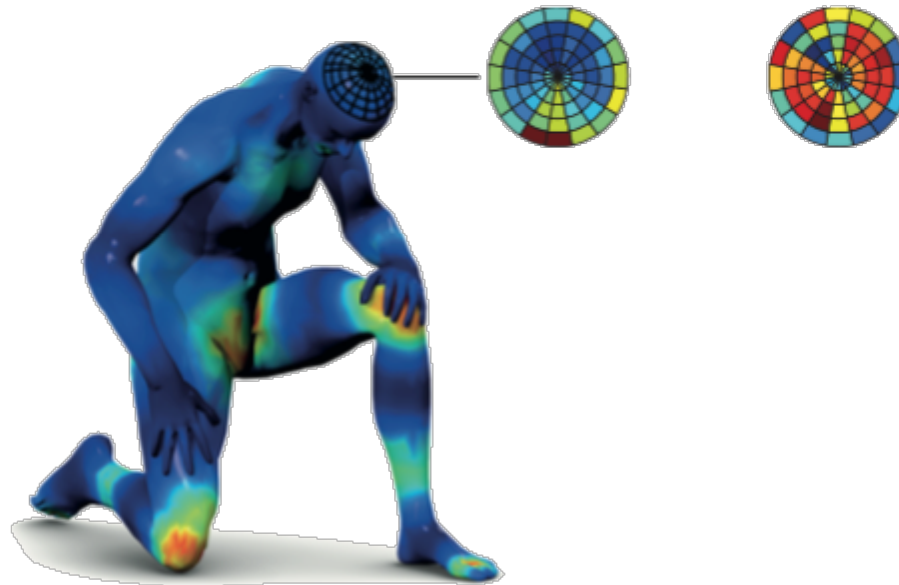


Weighting functions of the geodesic polar patch operator shown in  $(\rho, \theta)$  coordinates (contours mark the  $\frac{1}{2}$ -level set)

# Geodesic CNN

- **Geodesic convolution** = apply filter  $a$  to patches extracted from  $f \in L^2(X)$  in local geodesic polar coordinates

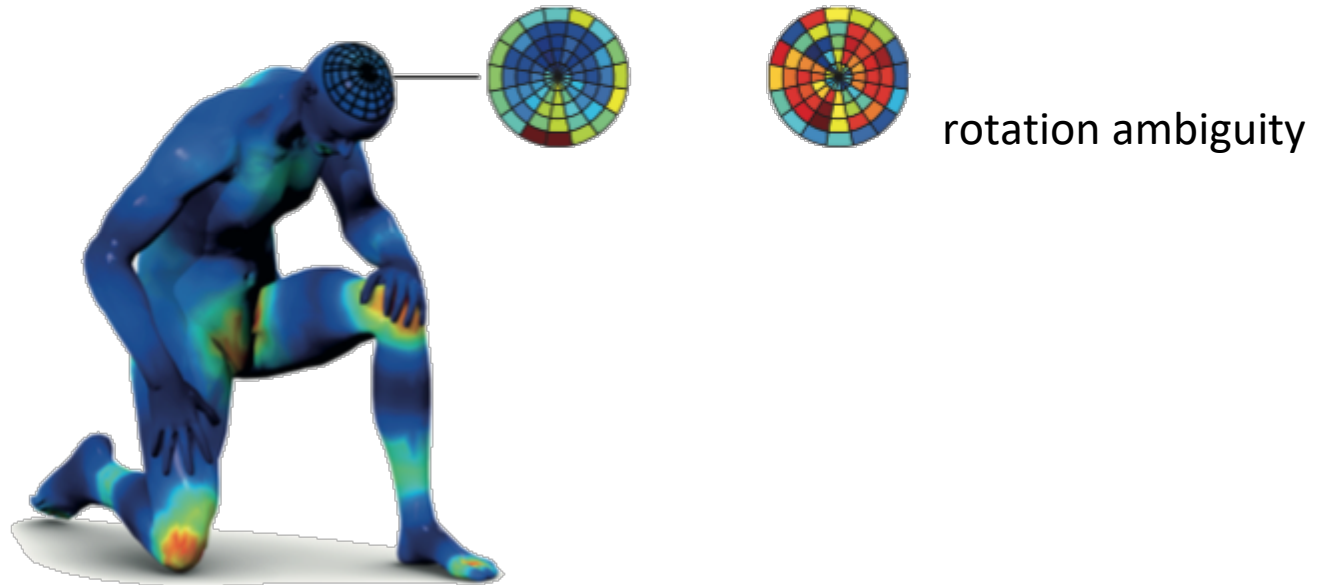
$$(f \star a)(x) = \sum_{\theta, r} \underbrace{(D(x)f)(r, \theta)}_{\text{patch}} \underbrace{a(\theta, r)}_{\text{filter}}$$



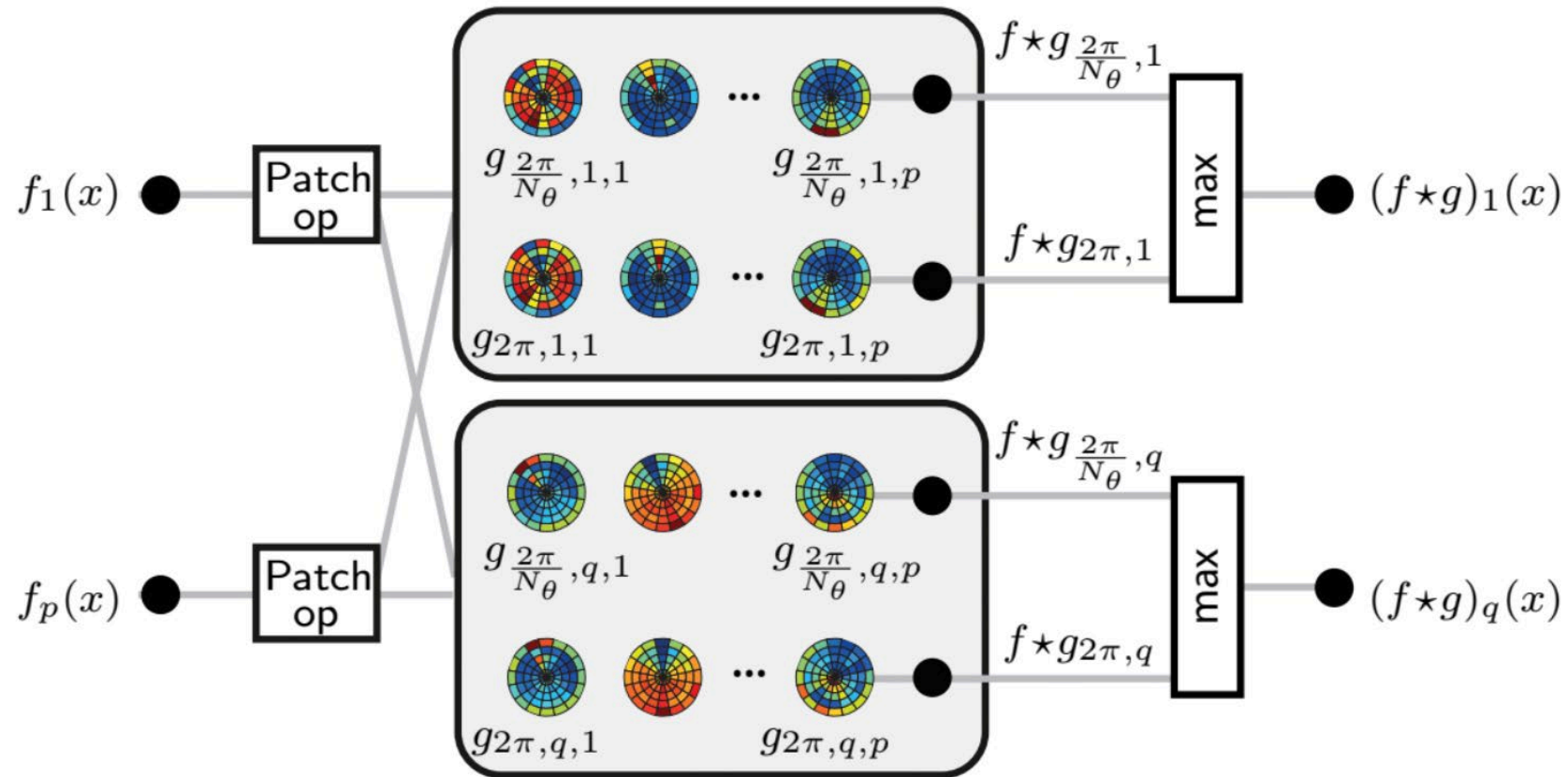
# Geodesic CNN

- **Geodesic convolution** = apply filter  $a$  to patches extracted from  $f \in L^2(X)$  in local geodesic polar coordinates

$$(f \star a)(x) = \sum_{\theta, r} \underbrace{(D(x)f)(r, \theta)}_{\text{patch}} \underbrace{a(\theta, r)}_{\text{filter}}$$



# Handling rotation ambiguity via pooling

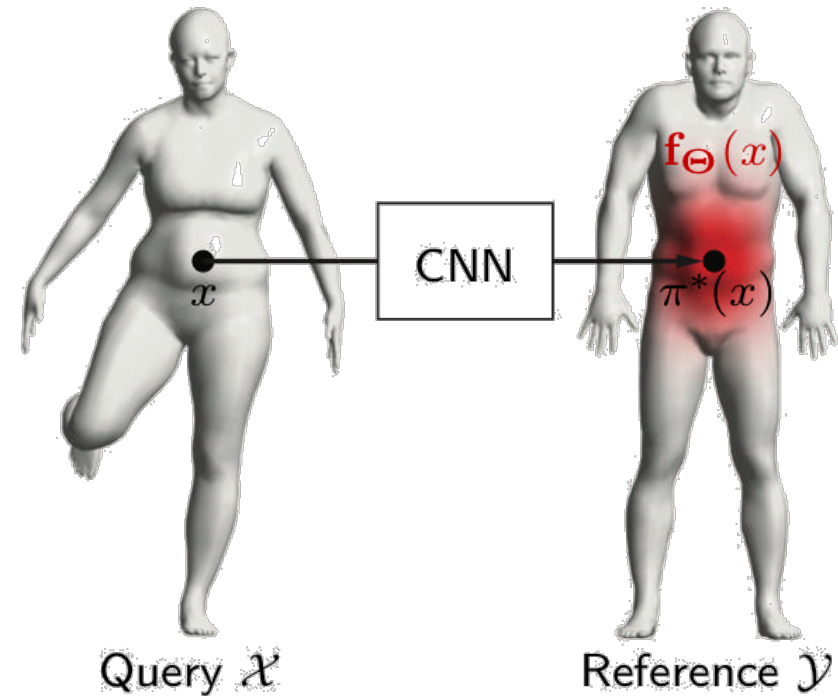


Conv. layer  $(f_l \star g)_{\Delta\theta, l}(x) = \xi \left( \sum_{\ell=1}^p (f_\ell \star g_{\Delta\theta, l, \ell})(x) \right)$

Angular max pooling  $(f \star g)_l(x) = \max_{\Delta\theta} (f \star g)_{\Delta\theta, l}(x)$

# Application: Shape matching

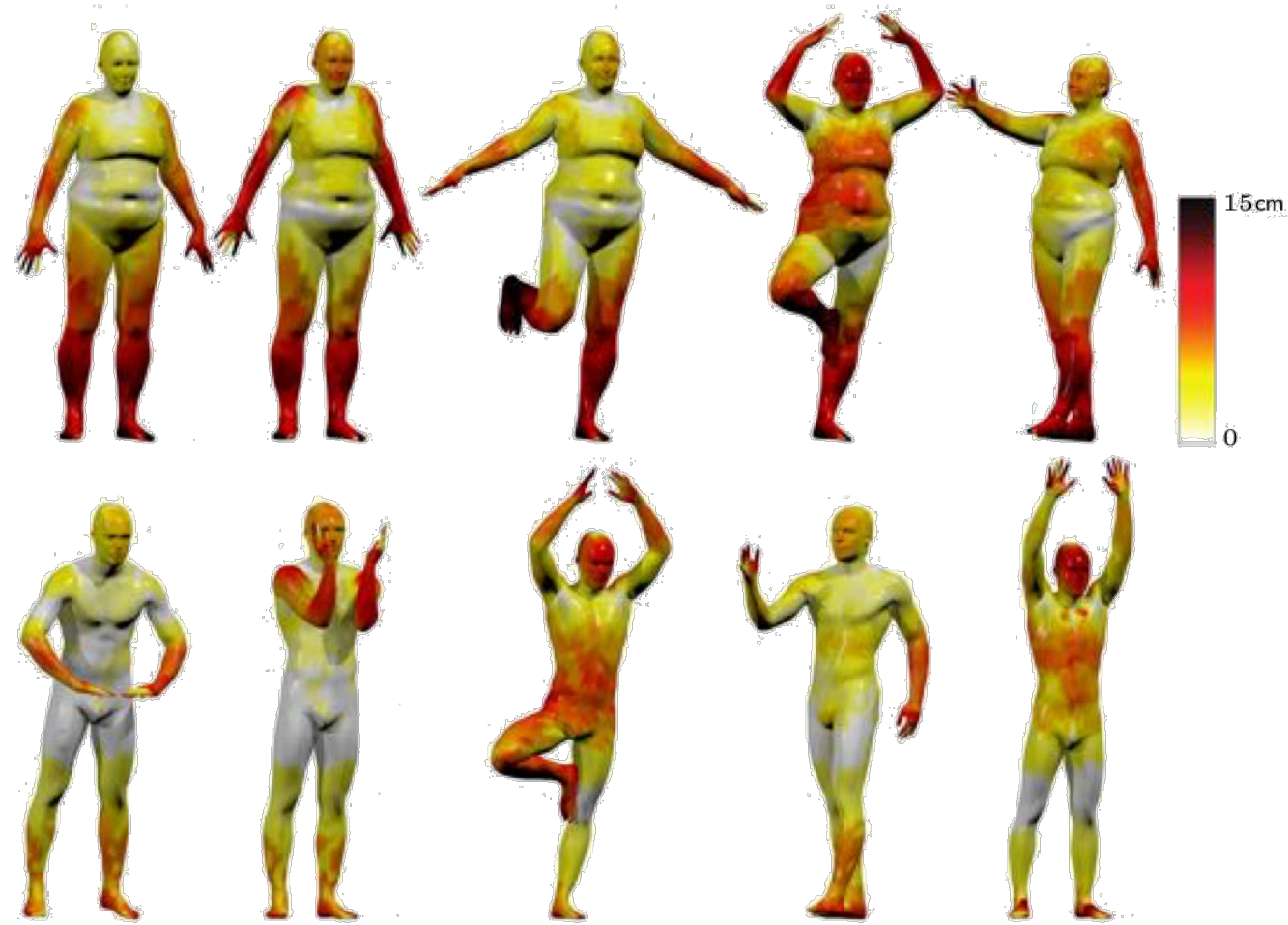
- Groundtruth correspondence  $\pi^* : \mathcal{X} \rightarrow \mathcal{Y}$  from query shape  $\mathcal{X}$  to some **reference shape**  $\mathcal{Y}$  (discretized with  $n$  vertices)
- Correspondence = **label** each query vertex  $x$  as reference vertex  $y$
- Net output at  $x$  after softmax layer  
 $\mathbf{f}_\Theta(x) = (f_{\Theta,1}(x), \dots, f_{\Theta,n}(x))$   
= probability distribution on  $\mathcal{Y}$



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

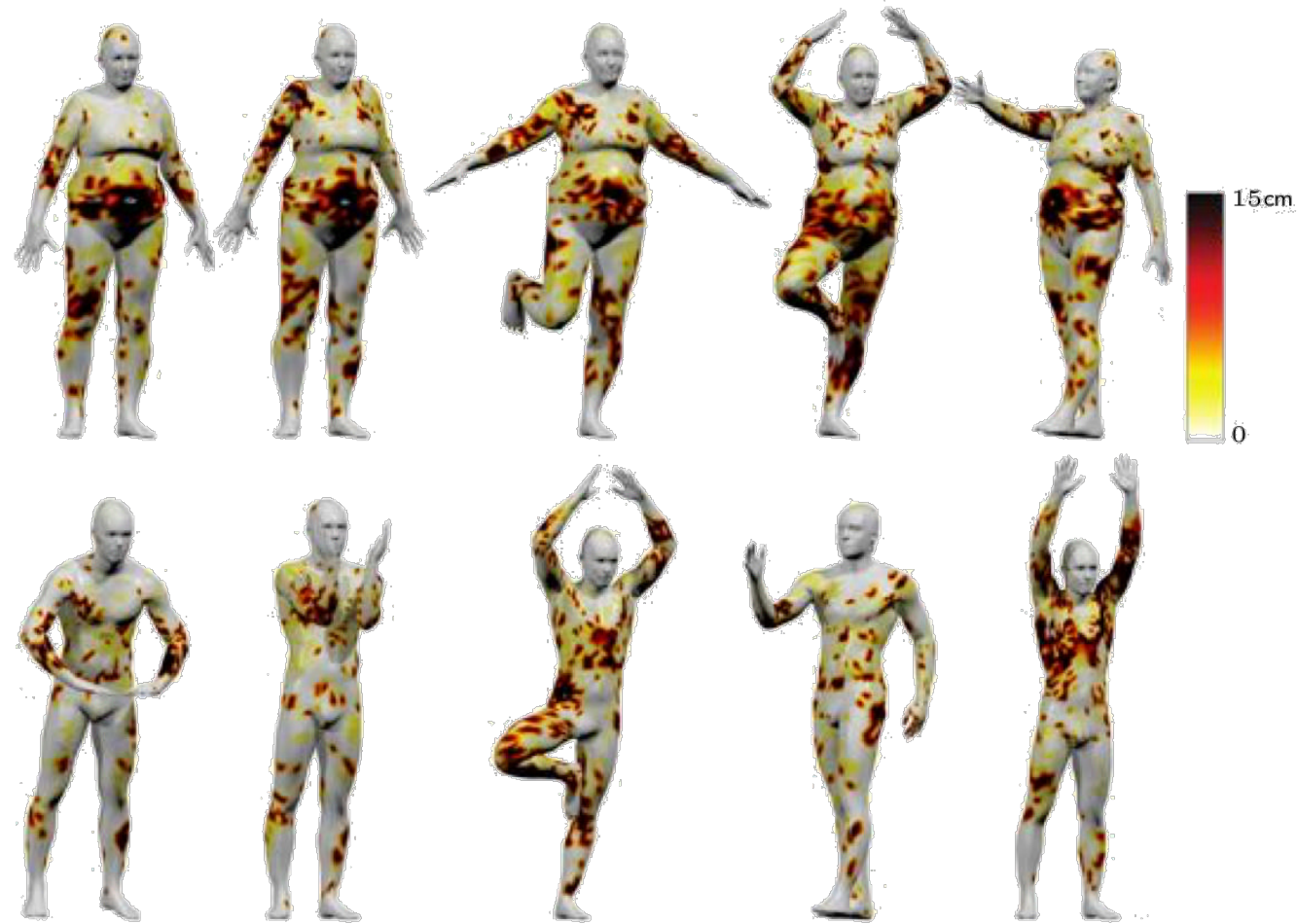
$$\min_{\Theta} \sum_x H(\delta_{\pi^*(x)}, \mathbf{f}_\Theta(x))$$

# Application: Shape matching



Pointwise correspondence error (geodesic distance from groundtruth)

# Application: Shape matching



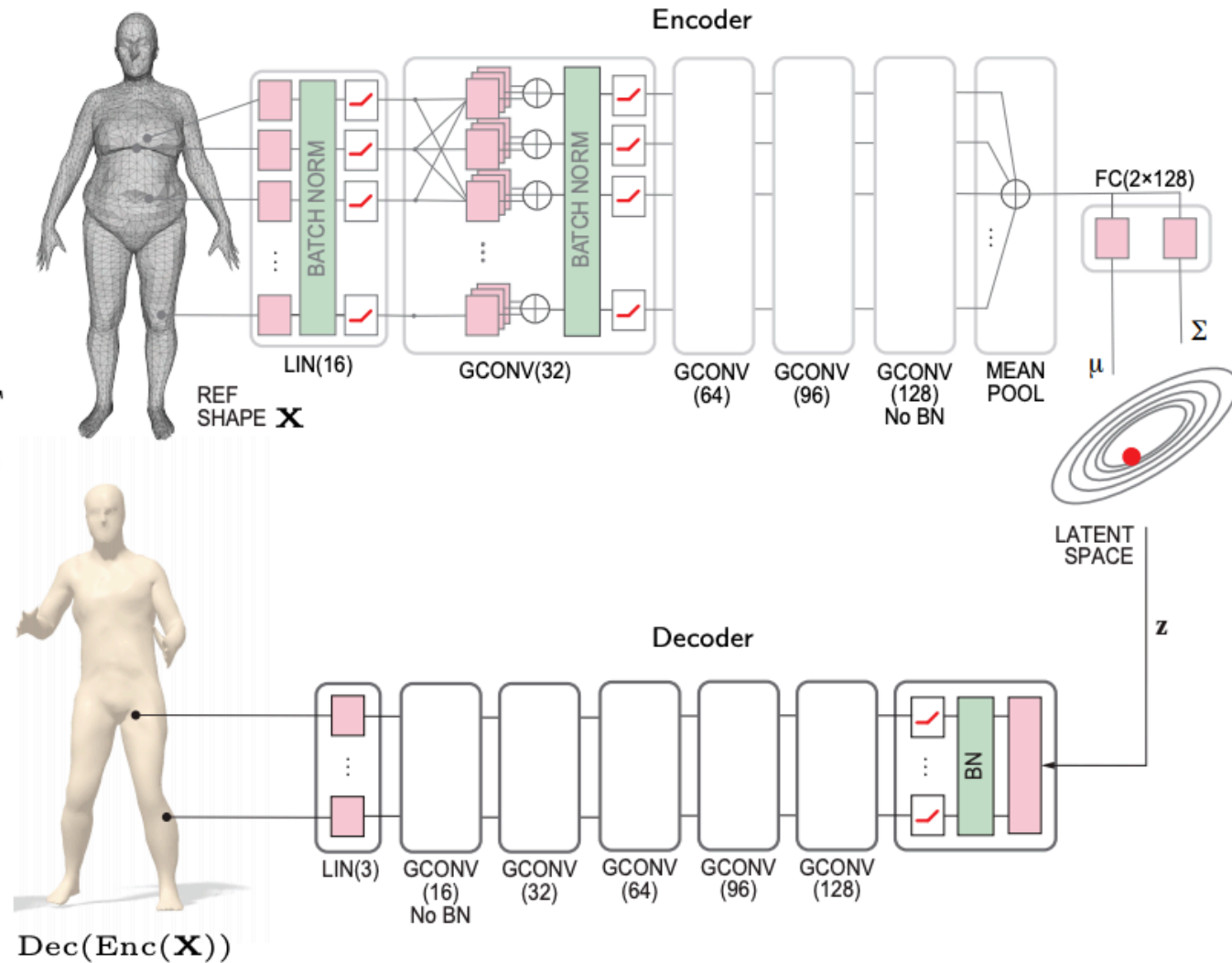
Pointwise correspondence error (geodesic distance from groundtruth)

# Application: Shape completion

Minimize

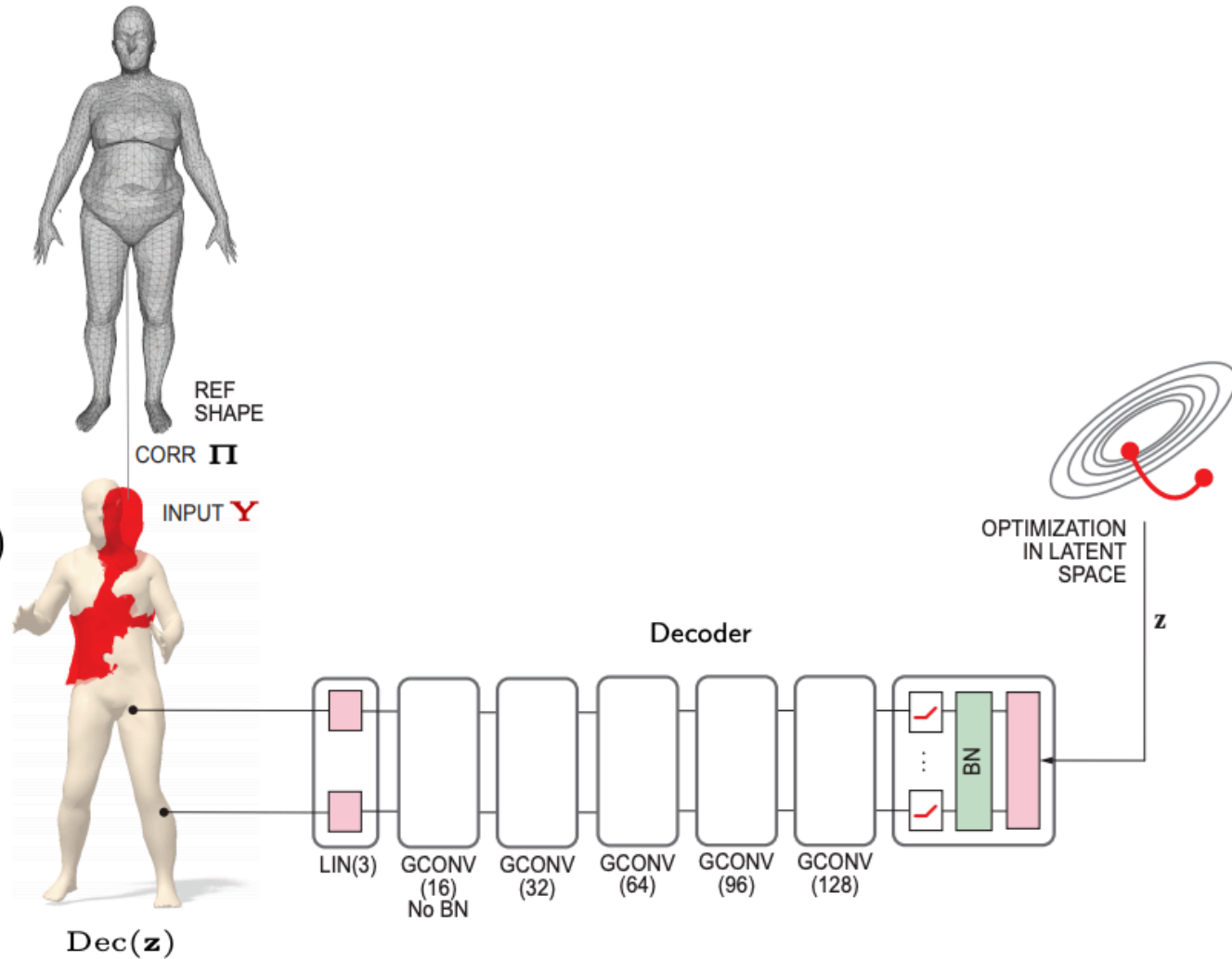
$$\|\text{Dec}(\text{Enc}(\mathbf{X})) - \mathbf{X}\|_F + \lambda D_{\text{KL}}(q(\mathbf{z}|\mathbf{X})||p(\mathbf{z}))$$

w.r.t. net parameters

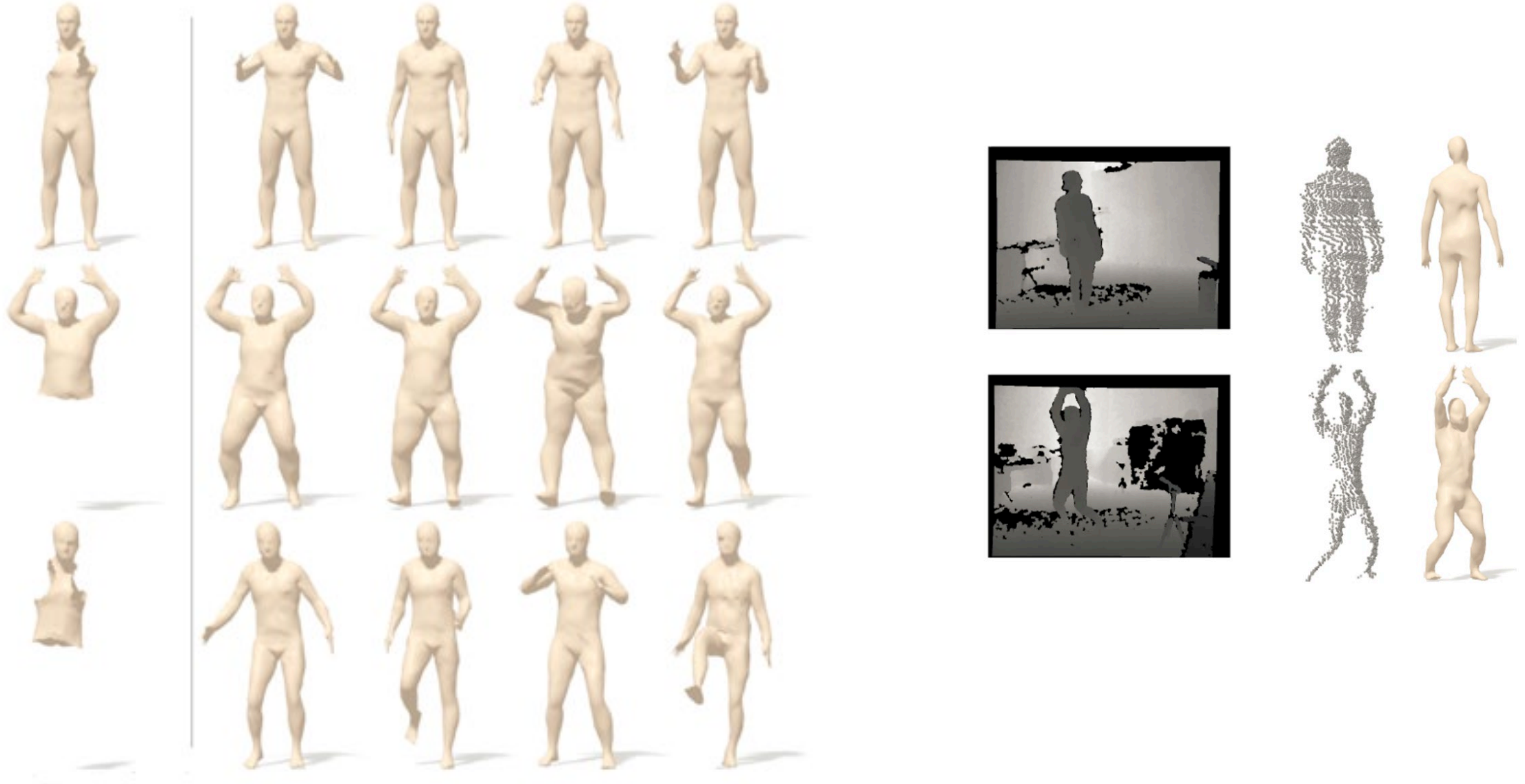


# Application: Shape completion

Minimize  
 $\|\text{Dec}(\mathbf{z})\mathbf{\Pi} - \mathbf{T}\mathbf{Y}\|_F$   
in alternating manner  
w.r.t.  $\mathbf{z}$  and  $\mathbf{T} \in \text{SO}(3)$



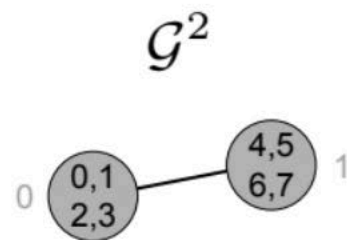
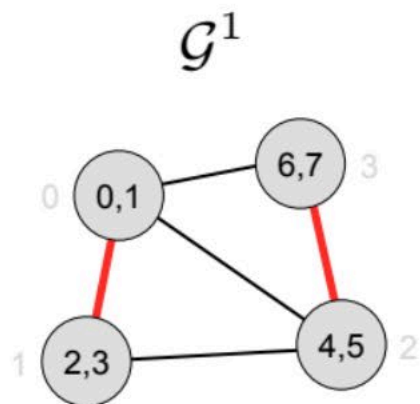
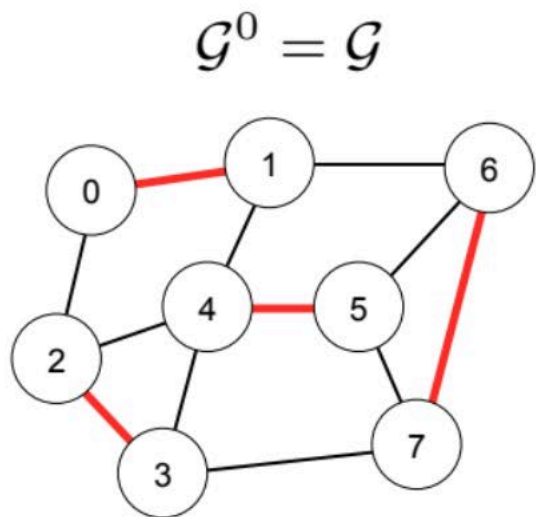
# Application: Shape completion



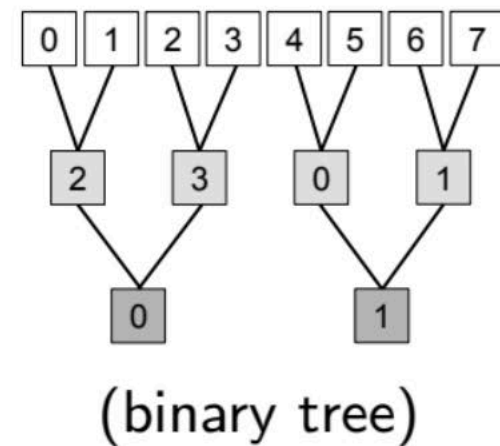
# Many more challenges...

- Pooling
- Input features
- gCNN for edges
- Unknown connectivity
- ...

# Graph pooling

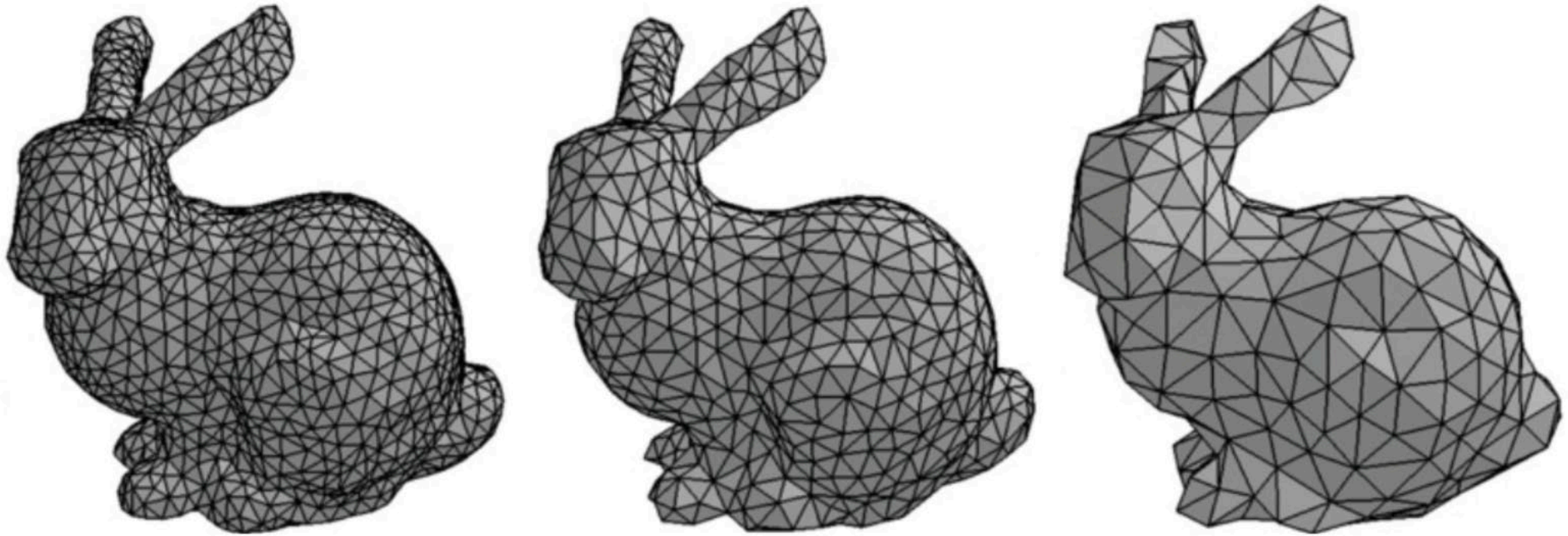


Coarsening structure



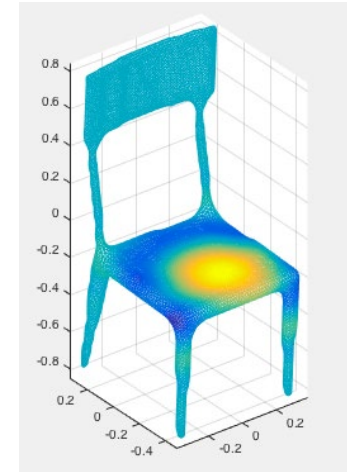
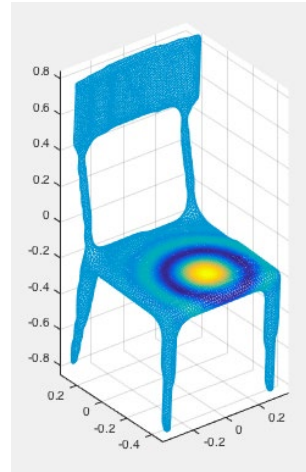
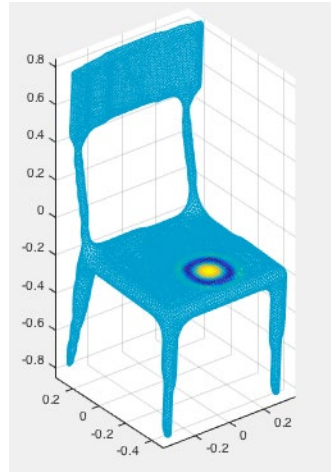
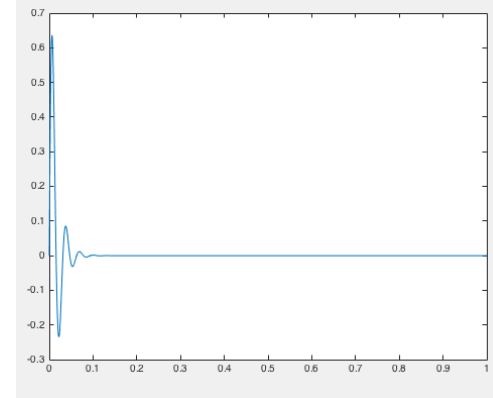
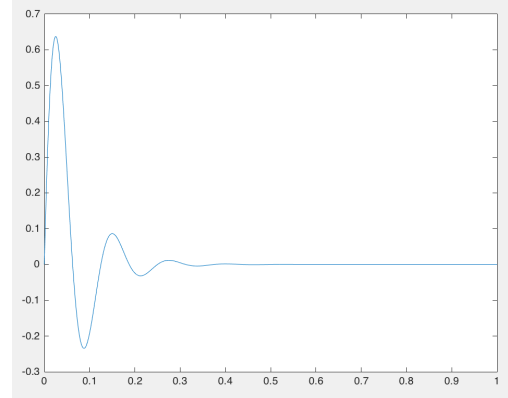
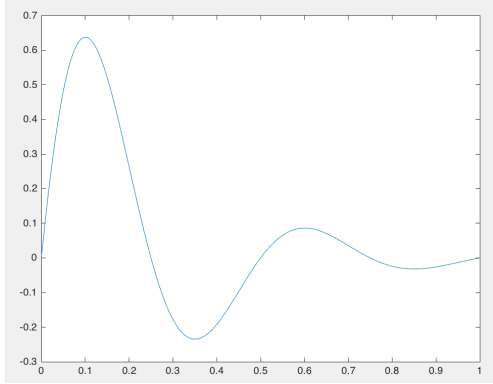
- Produce a sequence of coarsened graphs
- Max or average pooling of collapsed vertices
- Binary tree arrangement of node indices

# Mesh pooling

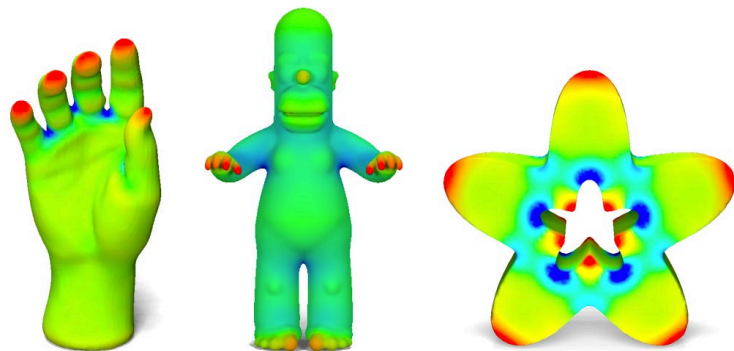


Example of progressive coarsening of a mesh

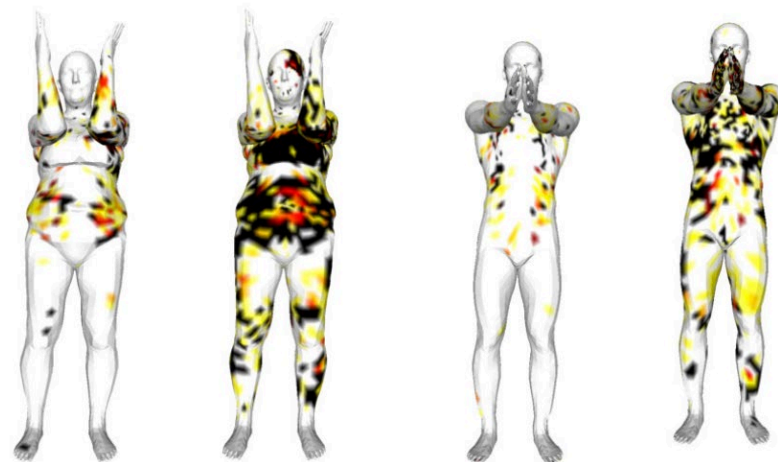
# Spectral pooling



# Features



HKS

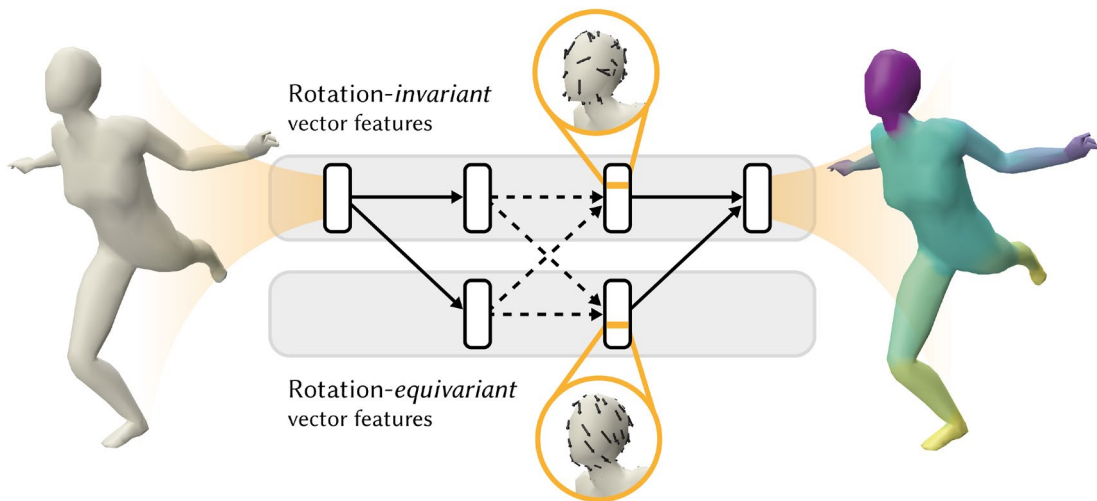
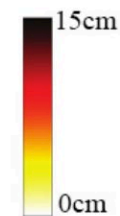


XYZ

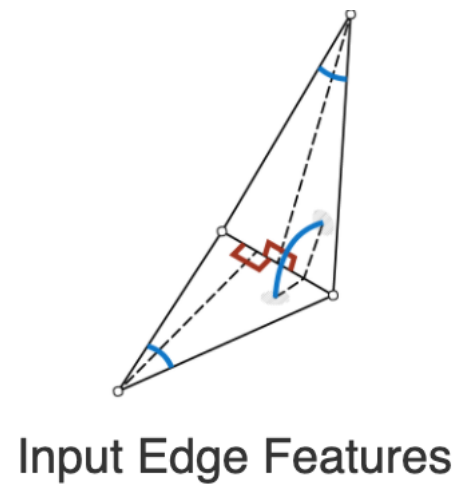
SHOT

XYZ

SHOT

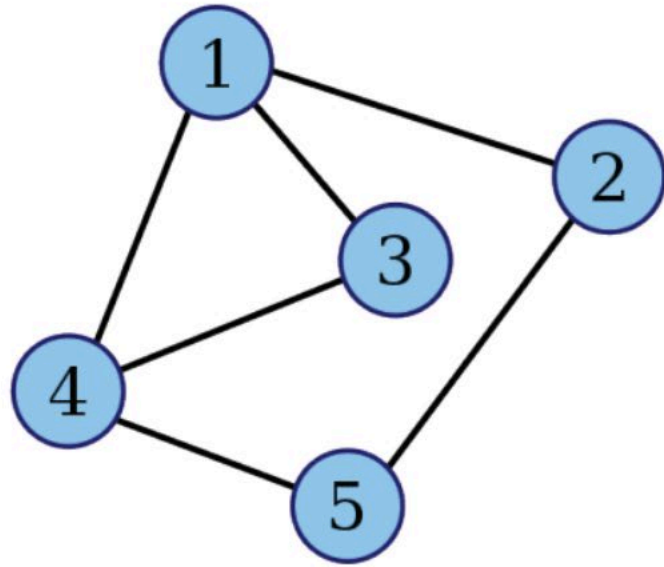


Circular Harmonics

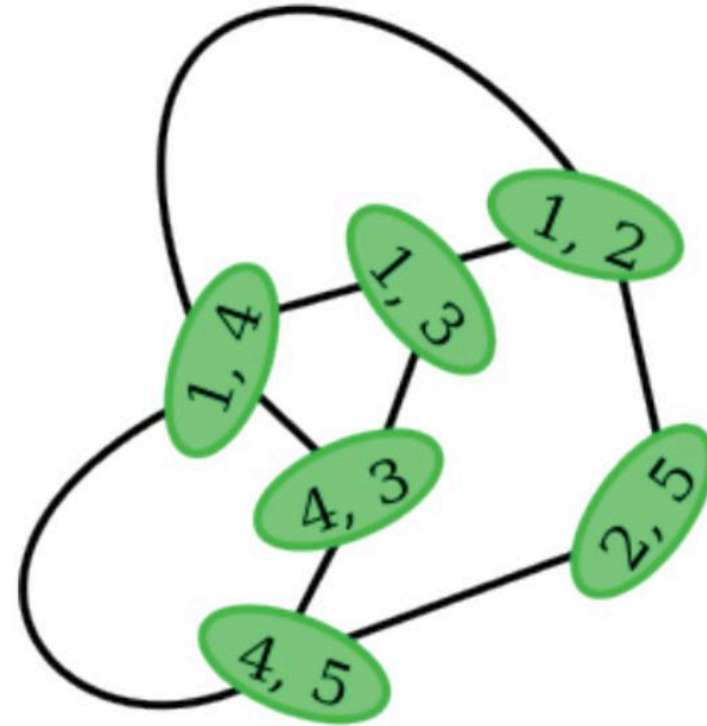


Input Edge Features

# Dual-primal gCNN

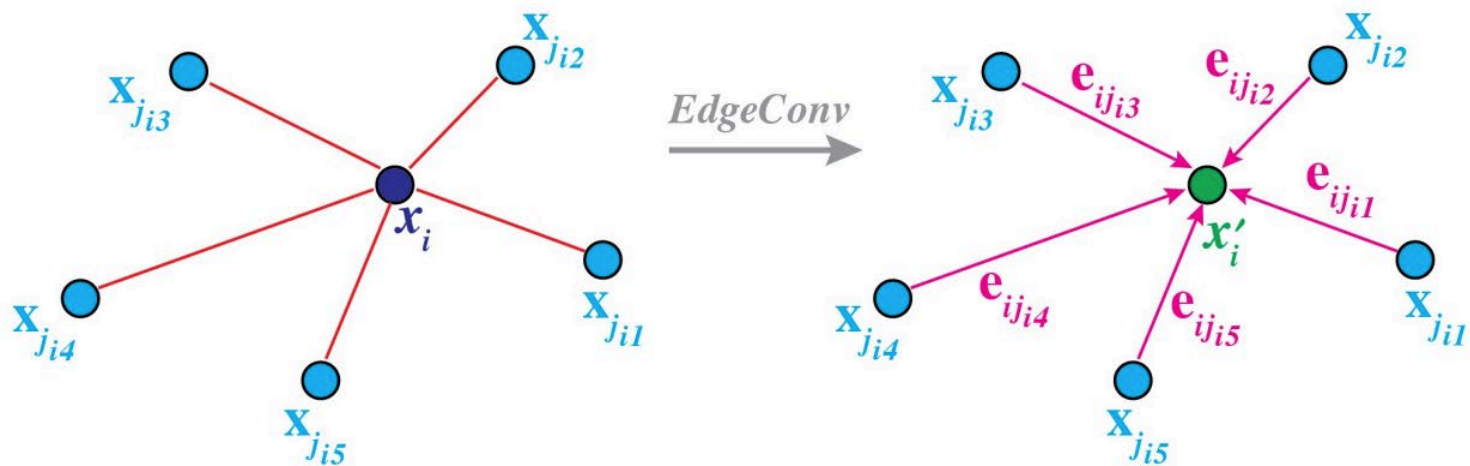
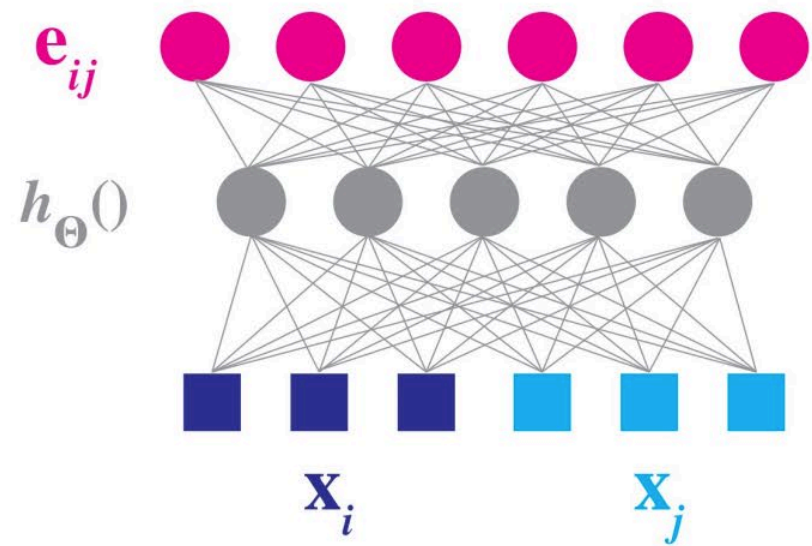


Primal graph  
 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

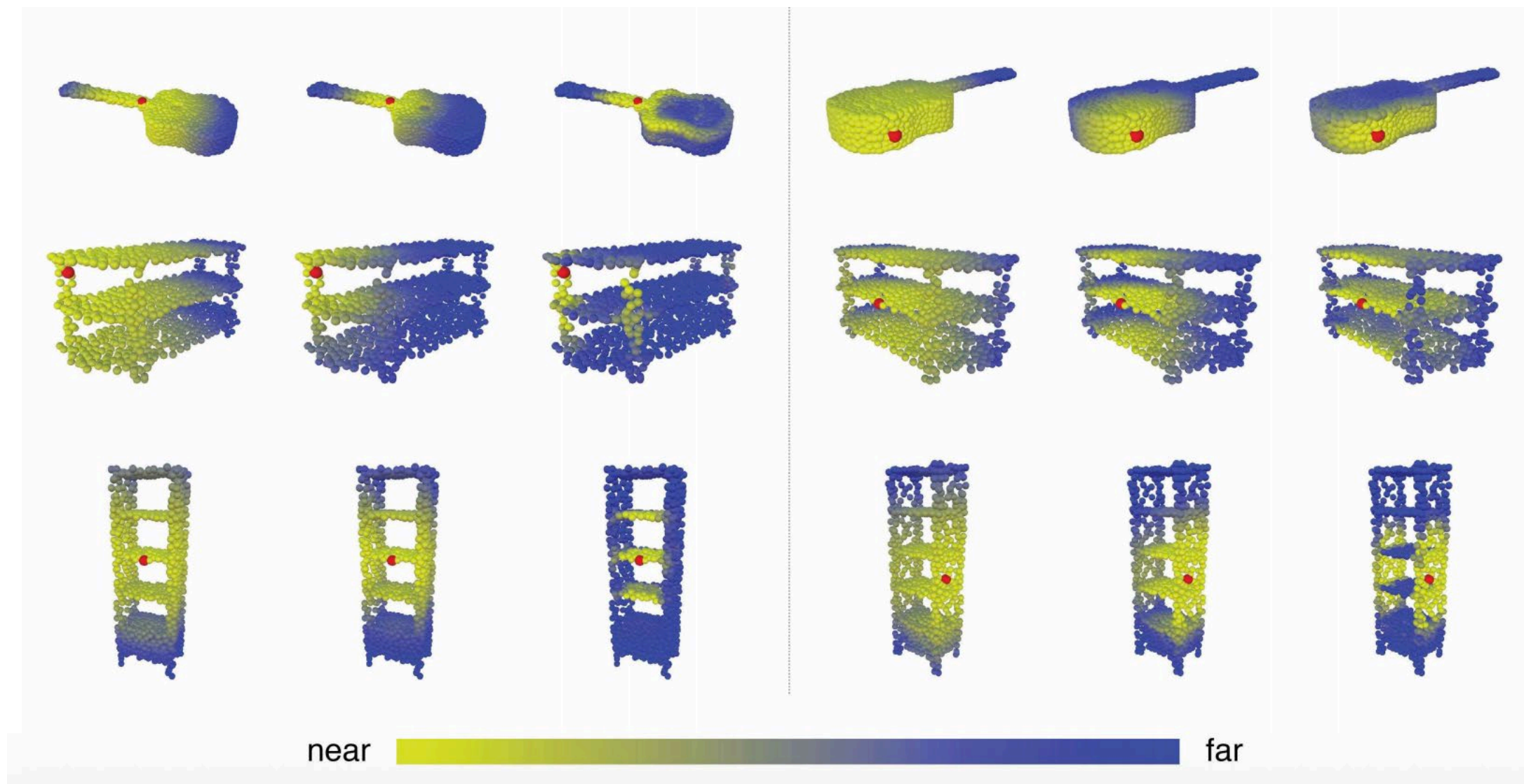


Dual or line graph  
 $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}} = \mathcal{E}, \tilde{\mathcal{E}})$

# Unknown connectivity (Dynamic Graph CNN)



# Unknown connectivity (Dynamic Graph CNN)



# Discussion

- Spatial construction is usually more efficient but less principled
- Spectral construction is more principled but usually slow (computing Laplacian eigenvectors for large scale data could be painful)
- On going research tries to bridge the gap
- Generalization issue on generic graphs is still a challenge