Beyond regular grids

Generalization of Convolutional Architecture to Non-Euclidean Domains

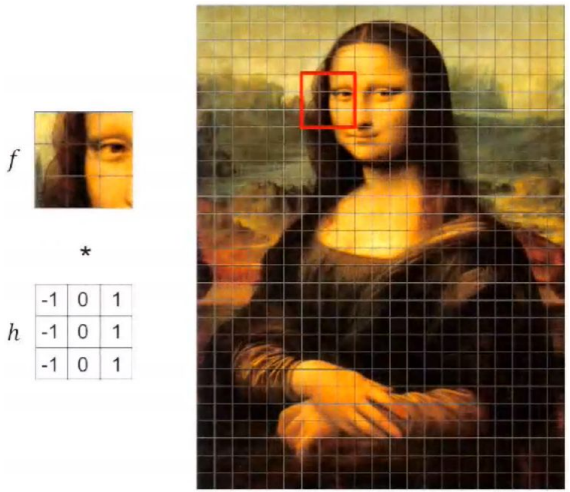
Main source

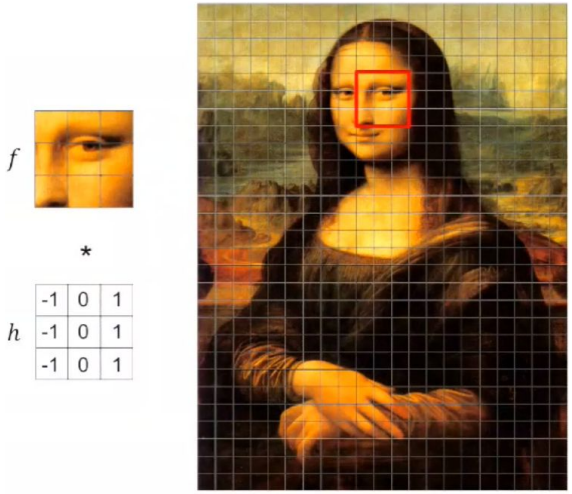
The work outlined in this presentation was published

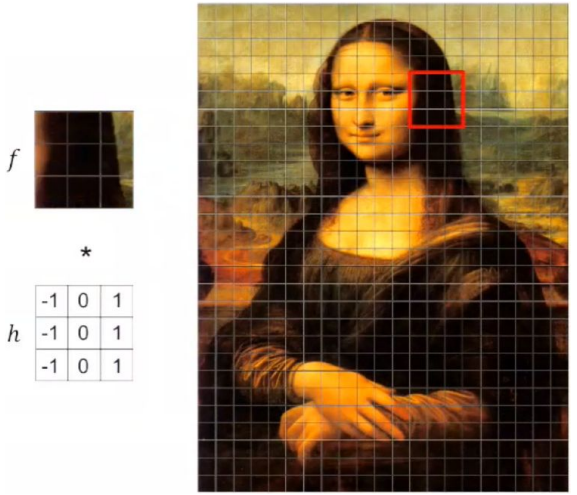
- Monti et al. 2016 - Geometric deep learning on graphs and manifolds using mixture model CNNs (http://arxiv.org/abs/1611.08402)

This work is a generalization of multiple previous attempts at implementing spatial convolution on surfaces representing boundaries of 3D volumes.

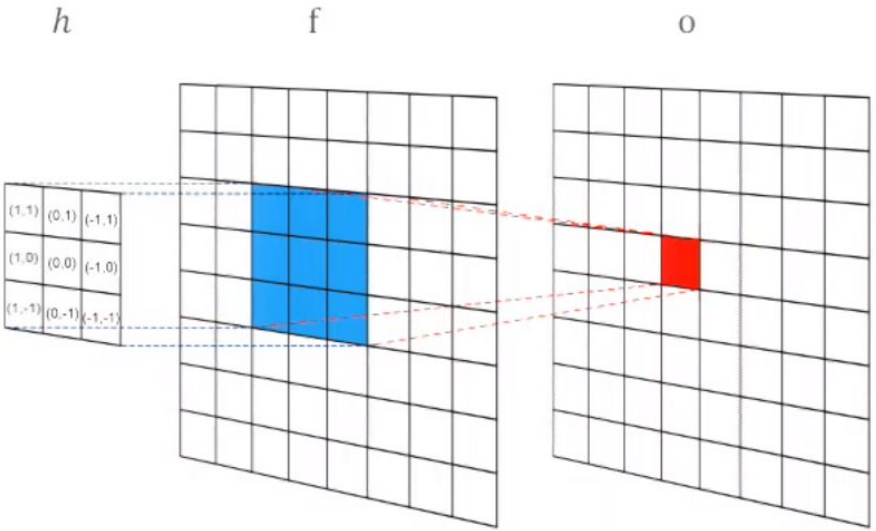
Remember Convolutions?







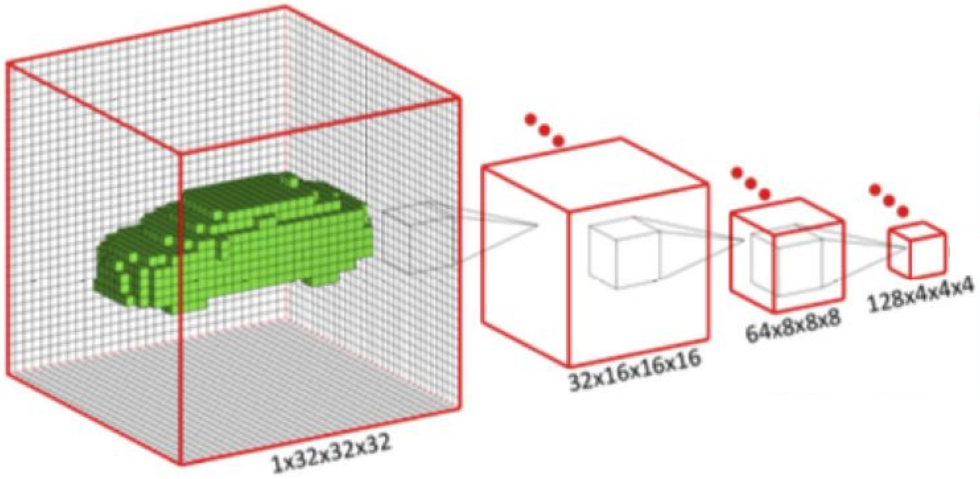
Well defined on regular grids (euclidean spaces)



$$g(m,n) = (h * f)(m,n) = \sum_{i,j=-w}^{w} h(i,j) \cdot f(m-i, n-j)$$

- To do a convolution g of weight matrix h (size wxw) over f at coordinates m,n
  - A patch of size wxw centered at coordinates m,n is extracted from f
  - Each cell (eg. pixel) from the extracted patch is multiplied with its corresponding weight from h
  - Output of the convolution is a sum of the these multiplications
- To keep the shape of o equal to the shape of f we need **padding** of $w/2$, and a stride of 1

Well defined on regular grids (euclidean spaces)

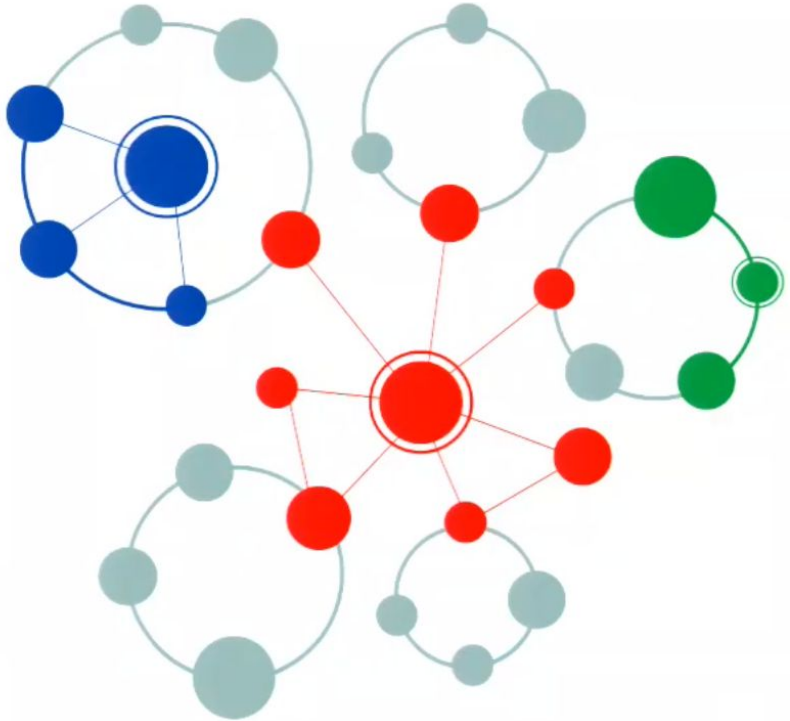


$$g(m,n,p) = (h * f)(m,n,p) = \sum_{i,j,k=-w}^{w} h(i,j,k) \cdot f(m-i, n-j, p-k)$$

- Things are very similar in 3D, however with very significant implementation drawbacks which we will discuss later in the presentation

But what about non-euclidean spaces

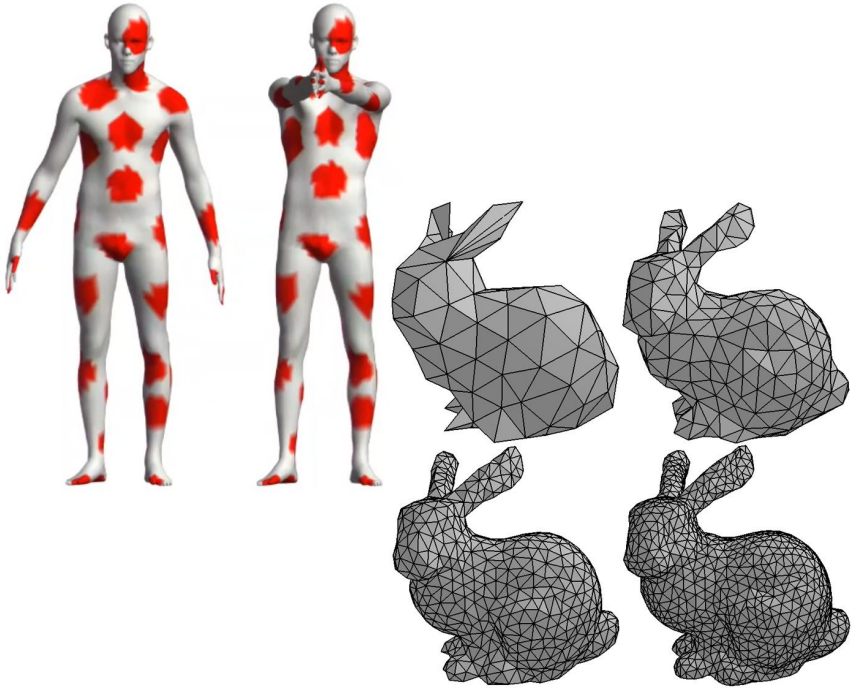
- What if we wanted to do pattern recognition over graphs (eg. social or gene regulatory networks)
  - Neighbourhood/distance is not uniquely defined (could be number of hops, sum of edge weights or any other function)
  - Vertices have very different neighbourhoods, thus making it impossible to apply convolution as previously defined (as it's impossible to define a weight matrix upfront)



Graphs

But what about non-euclidean spaces

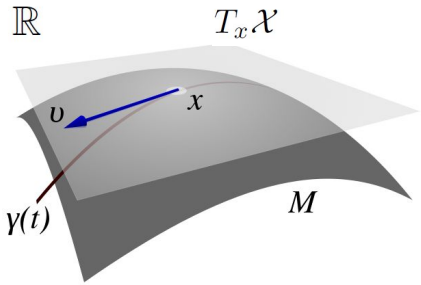
- What if we wanted to detect patterns in signals defined over surfaces (2-manifolds)
- Note that in most cases surfaces are implemented as triangular meshes (sets of triangles connected on shared edges)
- Triangular meshes can also be viewed as graphs where each vertex has x,y,z coordinates, and a list of neighboring vertices



Manifolds

Riemannian manifolds

- $\mathcal{X}$ - d-dimensional differentiable manifold, possibly with boundary $\partial\mathcal{X}$
- $T_x\mathcal{X}$ - tangent space, d-dimensional Euclidean space to which the manifold is around point $x \in \mathcal{X}$ to which the manifold is homeomorphic
- Riemannian metric - An inner product $\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$ depending smoothly on x
- $f : \mathcal{X} \to \mathbb{R}$ - smooth real functions (scalar fields) on the manifold
- $y \in \mathcal{N}(x)$ - point/vertex in the neighbourhood of x



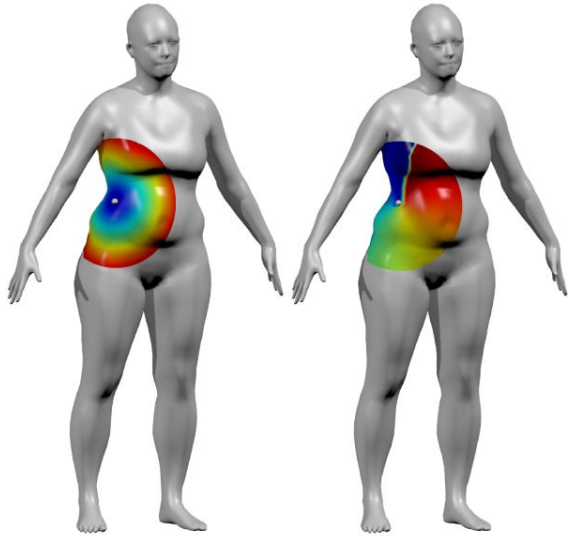
Geodesic CNN (GCNN)

Patch Operator:

$$(D(x)f)(\rho,\theta) = \int_{\mathcal{X}} w_{\rho,\theta}(x,y)f(y)dy$$

Weighing function:

$$w_{\rho,\theta}(x,y) = \frac{v_\rho(x,y)v_\theta(x,y)}{\int_X v_\rho(x,\gamma)v_\theta(x,\gamma)d\gamma}$$



Polar coordinates ρ, θ

*Masci et al. 2015 - Geodesic convolutional neural networks on Riemannian manifolds

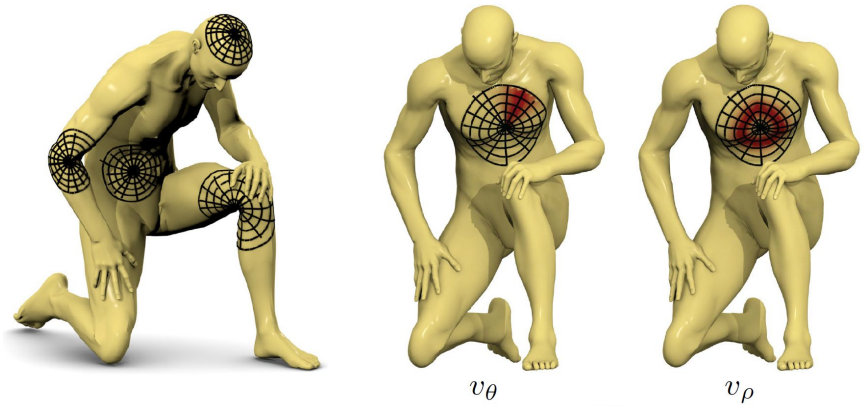
Geodesic CNN (GCNN)

v_ρ - Gaussian of the geodesic distance from x, centered around ρ:

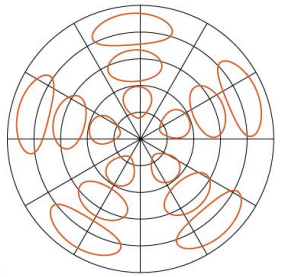
$$v_\rho(x,y) \propto e^{-(d_X(x,y)-\rho)^2/\sigma_\rho^2}$$

v_θ - Gaussian of the point-to-set distance to the geodesic $\Gamma(x,\theta)$:

$$v_\theta(x,y) \propto e^{-d_X^2(\Gamma(x,\theta),y)/\sigma_\theta^2}$$



$$w_{\rho,\theta}(x,y) = \frac{v_\rho(x,y)v_\theta(x,y)}{\int_X v_\rho(x,y)v_\theta(x,y)dy}$$



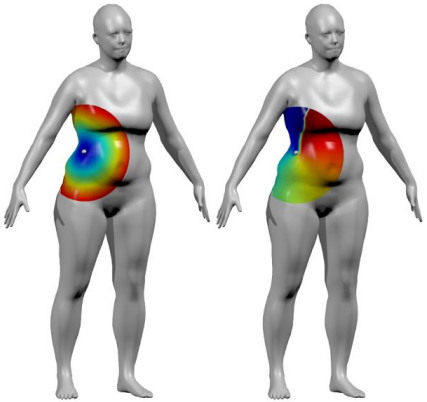
GCNN

Geodesic CNN (GCNN)

Geodesic Convolution:

$$(f \star g)(x) = \max_{\Delta\theta \in [0,2\pi)} \int_0^{2\pi} \int_0^{\rho_{\max}} g(\rho, \theta + \Delta\theta)(D(x)f)(\rho, \theta) d\rho d\theta$$

For all possible ρ, θ extract a patch of f at point x, multiply by a weight from g for this distance and angle, and max-pool across all rotations to account for the fact that the angle origin could be arbitrary



Polar coordinates ρ, θ

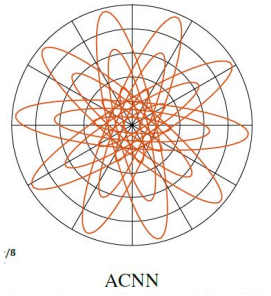
Anisotropic CNN (ACNN)

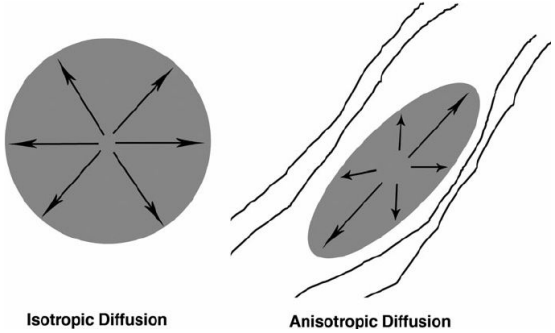
Anisotropic diffusion equation:

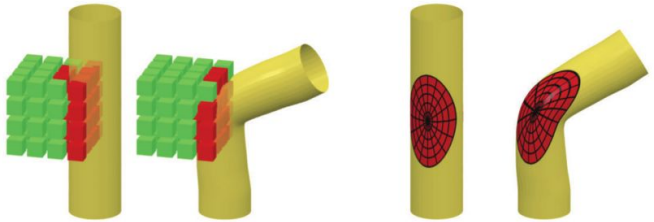
$$f_t(x,t) = -\mathrm{div}_{\mathcal{X}}(\mathbf{A}(x)\nabla_{\mathcal{X}} f(x,t))$$

$$\mathbf{A}_{\alpha\theta}(x) = \mathbf{R}_\theta(x) \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\theta^\top(x)$$

Rotation by θ in tangent space TxX







Extrinsic methods

Intrinsic methods

*Boscaini et al. 2016 - Learning shape correspondence with anisotropic convolutional neural networks

Generalized framework

- $\mathbf{u}(x, y)$ - pseudo coordinates, d-dimensional vector assigned to each y
- $\mathbf{w}_{\Theta}(\mathbf{u}) = (w_1(\mathbf{u}), \ldots, w_J(\mathbf{u}))$ - kernel parametrized by learnable params Θ
- J - dimensionality of the extracted patch x

$$D_j(x)f = \sum_{y \in \mathcal{N}(x)} w_j(\mathbf{u}(x, y))f(y), \quad j = 1, \ldots, J$$

$$(f \star g)(x) = \sum_{j=1}^{J} g_j \, D_j(x)f$$

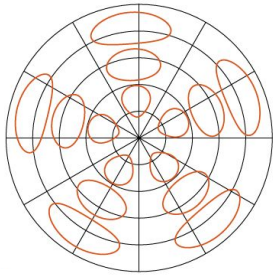
*Monti et al. 2016 - Geometric deep learning on graphs and manifolds using mixture model CNNs

GCNN Implementation

$$(D(x)f)(\rho,\theta) = \int_{\mathcal{X}} w_{\rho,\theta}(x,y)f(y)dy$$

$$v_\rho(x,y) \propto e^{-(d_X(x,y)-\rho)^2/\sigma_\rho^2}$$

$$v_\theta(x,y) \propto e^{-d_X^2(\Gamma(x,\theta),y)/\sigma_\theta^2}$$



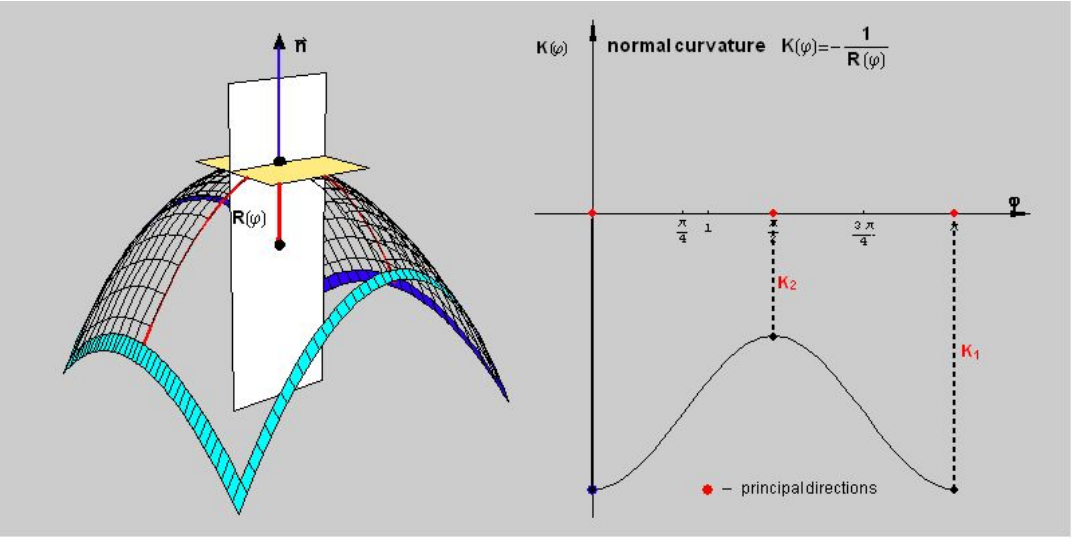
GCNN

$$D_j(x)f = \sum_{y\in\mathcal{N}(x)} w_j(\mathbf{u}(x,y))f(y), \quad j = 1,\dots,J$$

$$u(x,y) = \rho(x,y), \theta(x,y)$$

$$w_j(u) = \exp(-\frac{1}{2}(u-\bar{u}_j)^T \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1}(u-\bar{u}_j))$$

Principal curvatures



$$(f \star g)(x) = \max_{\Delta\theta \in [0,2\pi)} \int_0^{2\pi} \int_0^{\rho_{\max}} g(\rho, \theta + \Delta\theta)(D(x)f)(\rho, \theta) d\rho d\theta$$

$\theta = 0$ in the direction of maximum curvature

$$(f \star g)(x) = \sum_{j=1}^{J} g_j \, D_j(x) f$$

Other generalized implementations

Table 1. Several CNN-type geometric deep learning methods on graphs and manifolds can be obtained as a particular setting of the proposed framework with an appropriate choice of the pseudo-coordinates and weight functions in the definition of the patch operator. x denotes the reference point (center of the patch) and y a point within the patch. $\mathbf{x}$ denotes the Euclidean coordinates on a regular grid. $\bar{\alpha}, \bar{\sigma}_\rho, \bar{\sigma}_\theta$ and $\bar{\mathbf{u}}_j, \bar{\theta}_j, j = 1, \dots, J$ denote fixed parameters of the weight functions.

Method	Pseudo-coordinates	$\mathbf{u}(x, y)$	Weight function $w_j(\mathbf{u}), j = 1, \dots, J$
CNN [28]	Local Euclidean	$\mathbf{x}(x, y) = \mathbf{x}(y) - \mathbf{x}(x)$	$\delta(\mathbf{u} - \bar{\mathbf{u}}_j)$
GCNN [32]	Local polar geodesic	$\rho(x, y), \theta(x, y)$	$\exp(-\frac{1}{2}(\mathbf{u} - \bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u} - \bar{\mathbf{u}}_j))$
ACNN [7]	Local polar geodesic	$\rho(x, y), \theta(x, y)$	$\exp(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} \begin{pmatrix} \bar{\alpha} & \\ & 1 \end{pmatrix} \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u})$
GCN [26]	Vertex degree	$\deg(x), \deg(y)$	$\left(1 - \lvert 1 - \frac{1}{\sqrt{u_1}} \rvert\right)\left(1 - \lvert 1 - \frac{1}{\sqrt{u_2}} \rvert\right)$
DCNN [2]	Transition probability in r hops	$p^0(x, y), \dots, p^{r-1}(x, y)$	$\mathrm{id}(u_j)$

MoNet - Learnable patch operator

$$u(x, y) = \rho(x, y), \theta(x, y)$$

$$w_j(\mathbf{u}) = \exp(-\tfrac{1}{2}(\mathbf{u} - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\mathbf{u} - \boldsymbol{\mu}_j))$$

w_j is now a Gaussian Mixture Model of J gaussians with trainable $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

Entire model can be jointly trained by backpropagation, ie. both the convolution kernel and the patch operator

MoNet - Learnable patch operator

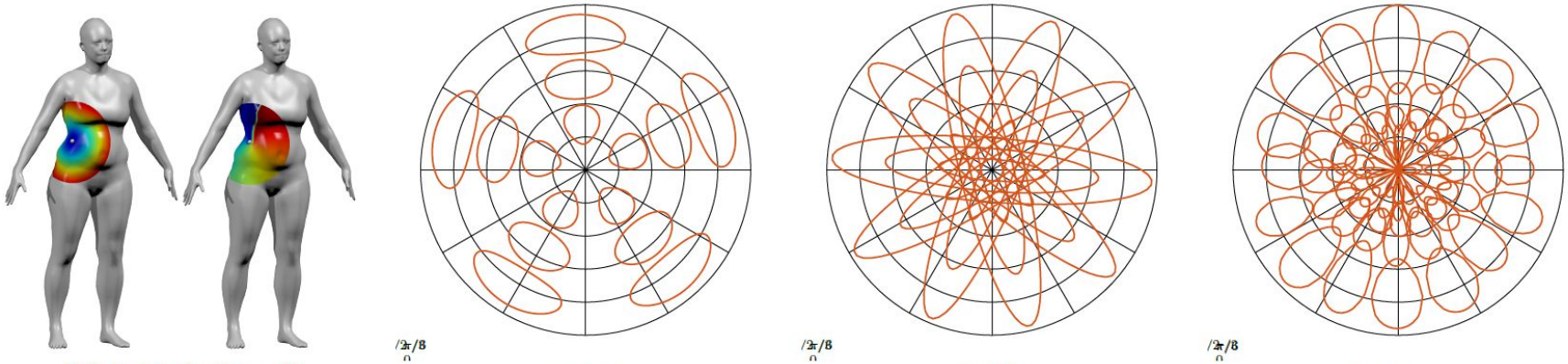
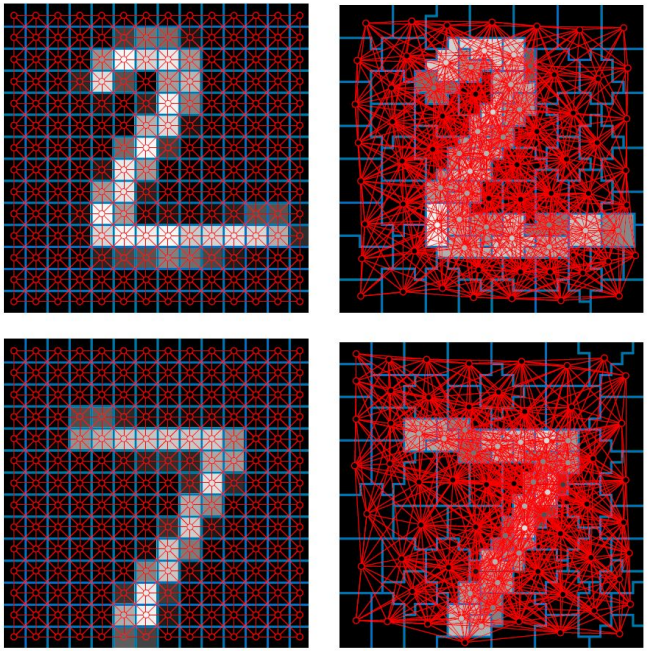


Figure 1. Left: intrinsic local polar coordinates ρ, θ on manifold around a point marked in white. Right: patch operator weighting functions $w_i(\rho, \theta)$ used in different generalizations of convolution on the manifold (hand-crafted in GCNN and ACNN and learned in MoNet). All kernels are L_∞-normalized; red curves represent the 0.5 level set.

MNIST Superpixel graph



Regular grid Superpixels

Figure 2. Representation of images as graphs. Left: regular grid (the graph is fixed for all images). Right: graph of superpixel adjacency (different for each image). Vertices are shown as red circles, edges as red lines.

- Data
  - superpixels - adjacent pixels with similar values
  - (super)pixel graphs - each pixel a vertex and all adjacent pixels are connected with edges
  - MNIST 28x28 = 784 pixels
  - 300, 150 and 75 superpixels
- MoNet
  - $\mathbf{u} = (\rho, \theta)$ with respect to superpixel barycenter
  - 25 gaussian kernels
  - 3 convolutional layers interleaved with pooling*
  - dropout = 0.5
  - batch size = 10

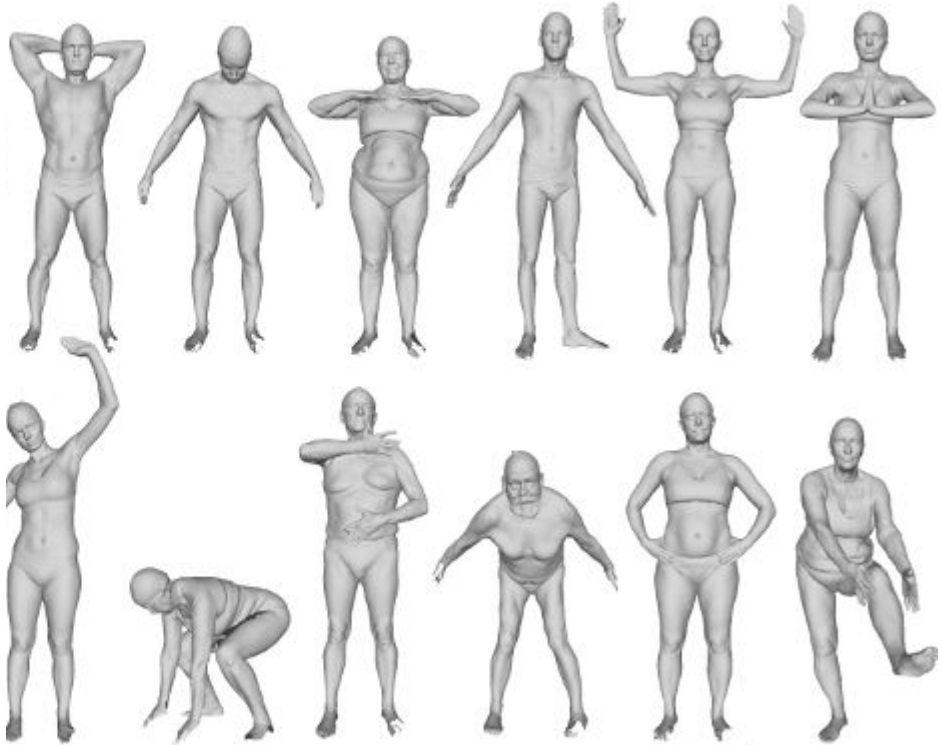
MNIST Superpixel graph

Table 2. Classification accuracy of classical Euclidean CNN (LeNet5), spectral CNN (ChebNet) and the proposed approach (MoNet) on different versions of the MNIST dataset. The setting of all the input images sharing the same graph is marked with *.

Dataset	LeNet5 [28]	ChebNet [15]	**MoNet**
*Full grid	99.33%	99.14%	99.19%
$*\frac{1}{4}$ grid	98.59%	97.70%	98.16%
300 Superpixels	-	88.05%	**97.30%**
150 Superpixels	-	80.94%	**96.75%**
75 Superpixels	-	75.62%	**91.11%**

Shape correspondence problem

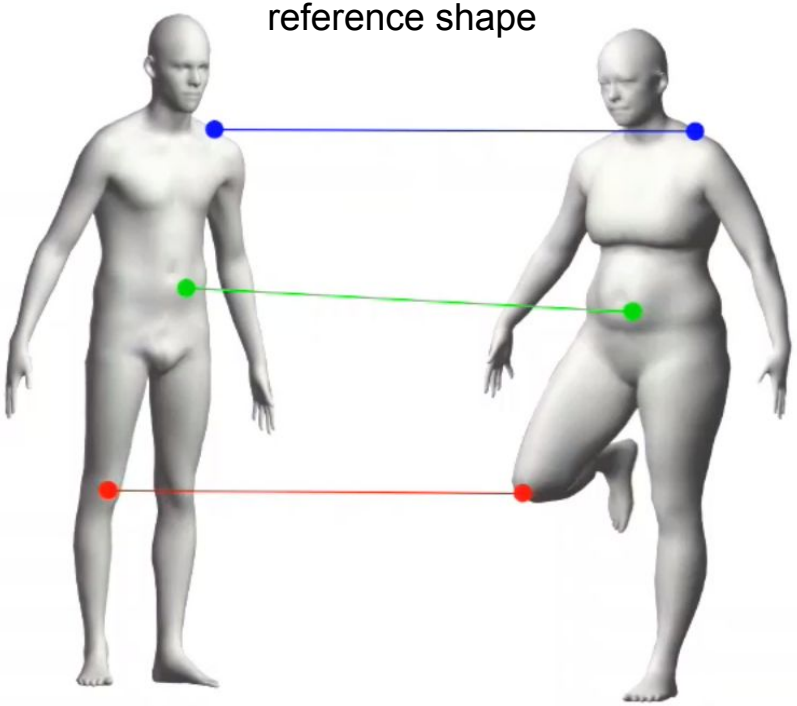
- FAUST dataset
  - 100 high resolution watertight meshes
  - 10 different people
  - 10 poses each
  - Exact ground-truth correspondence
  - Each shape has 6890 vertices
  - vertex-wise 544-dimensional SHOT descriptors (local histogram of normal vectors)



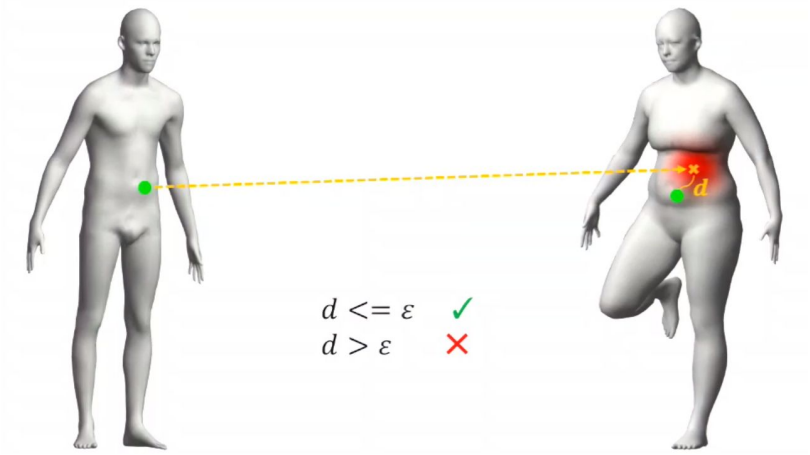
Shape correspondence problem

- Presented as a labeling problem
  - Input - vertex of a query shape X
  - Output - softmax over vertex indices on reference shape Y
  - Standard cross entropy loss
  - First subject in first pose used as ref
  - Used 8 subjects (80 shapes) for training, remaining for test set
- Network
  - LIN16+ReLU, MC32+ReLU, MC64+ReLU, MC128+ReLU, LIN256, LIN6890
  - Output refined refined using the intrinsic Bayesian filter to remove local outliers

For each point on a given shape, determine the corresponding point on the reference shape



Shape correspondence problem



Geodesic error d is the geodesic distance of predicted correspondence point from the true correspondence

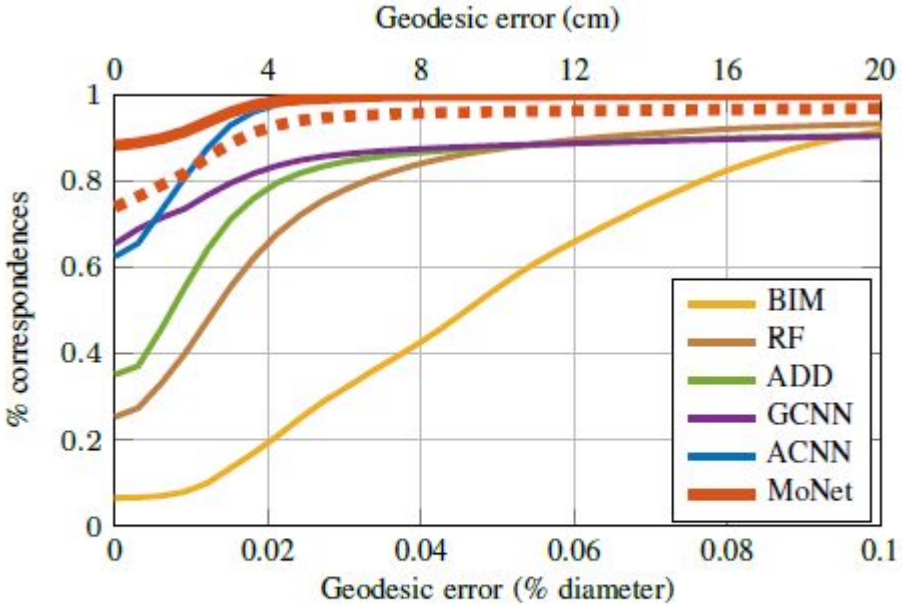


Figure 4. Shape correspondence quality obtained by different methods on the FAUST humans dataset. The raw performance of MoNet is shown in dotted curve.

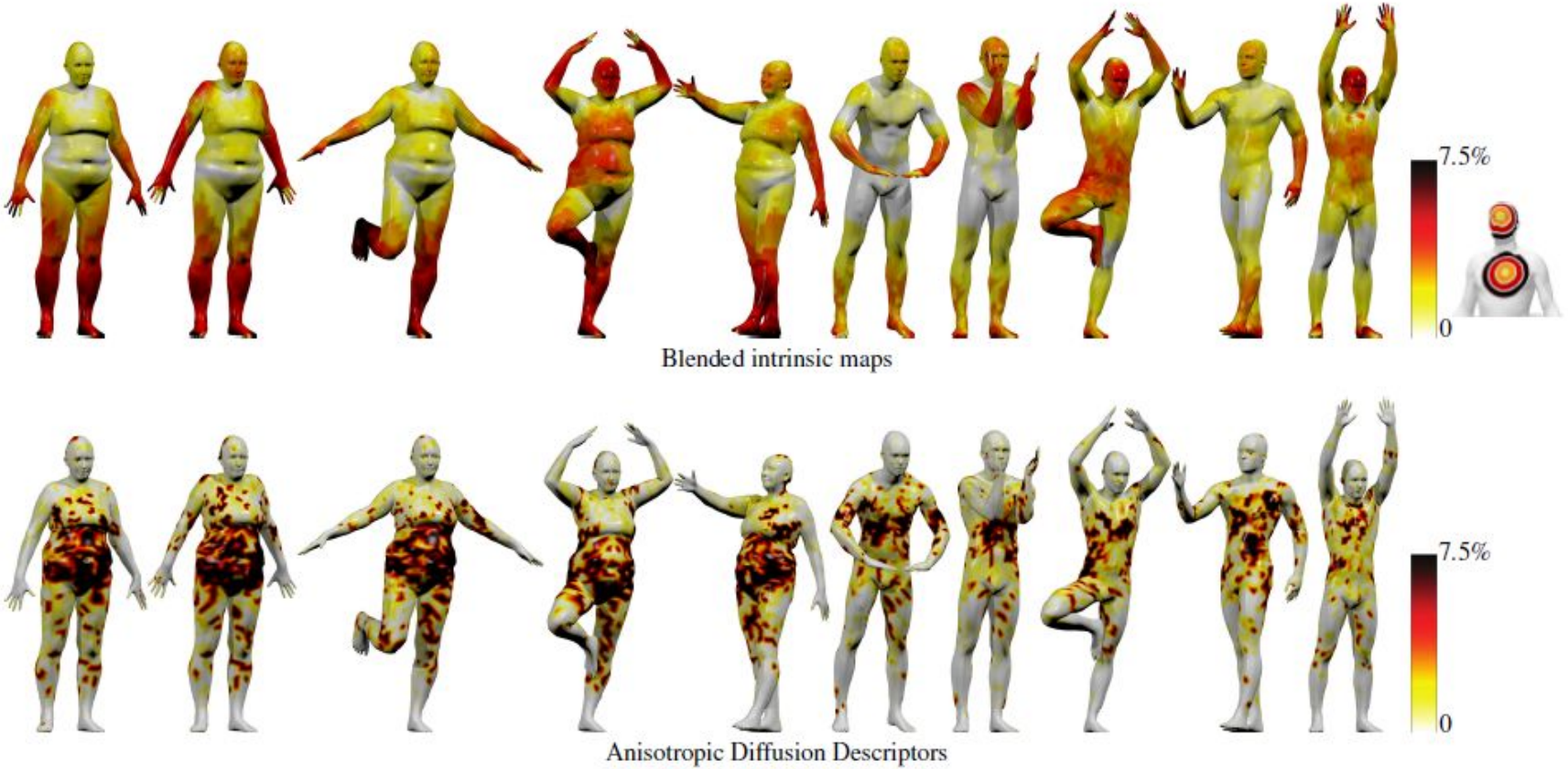
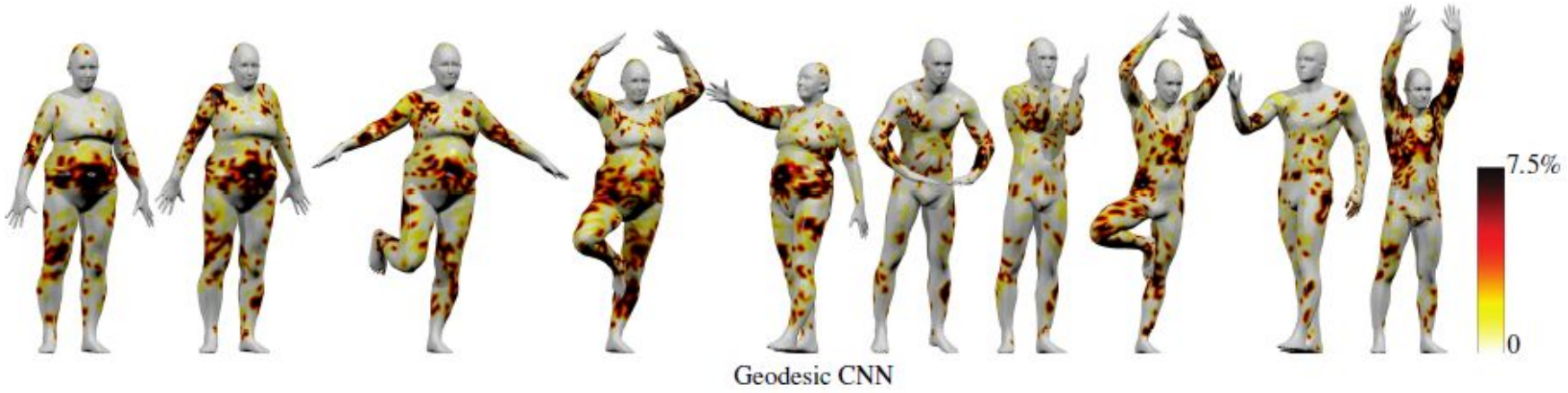


Figure 6. Pointwise error (geodesic distance from groundtruth) of different correspondence methods on the FAUST humans dataset. For visualization clarity, the error values are saturated at 7.5% of the geodesic diameter, which corresponds to approximately 15 cm. Hot colors represent large errors.



Geodesic CNN

Anisotropic CNN

Figure 6. Pointwise error (geodesic distance from groundtruth) of different correspondence methods on the FAUST humans dataset. For visualization clarity, the error values are saturated at 7.5% of the geodesic diameter, which corresponds to approximately 15 cm. Hot colors represent large errors.

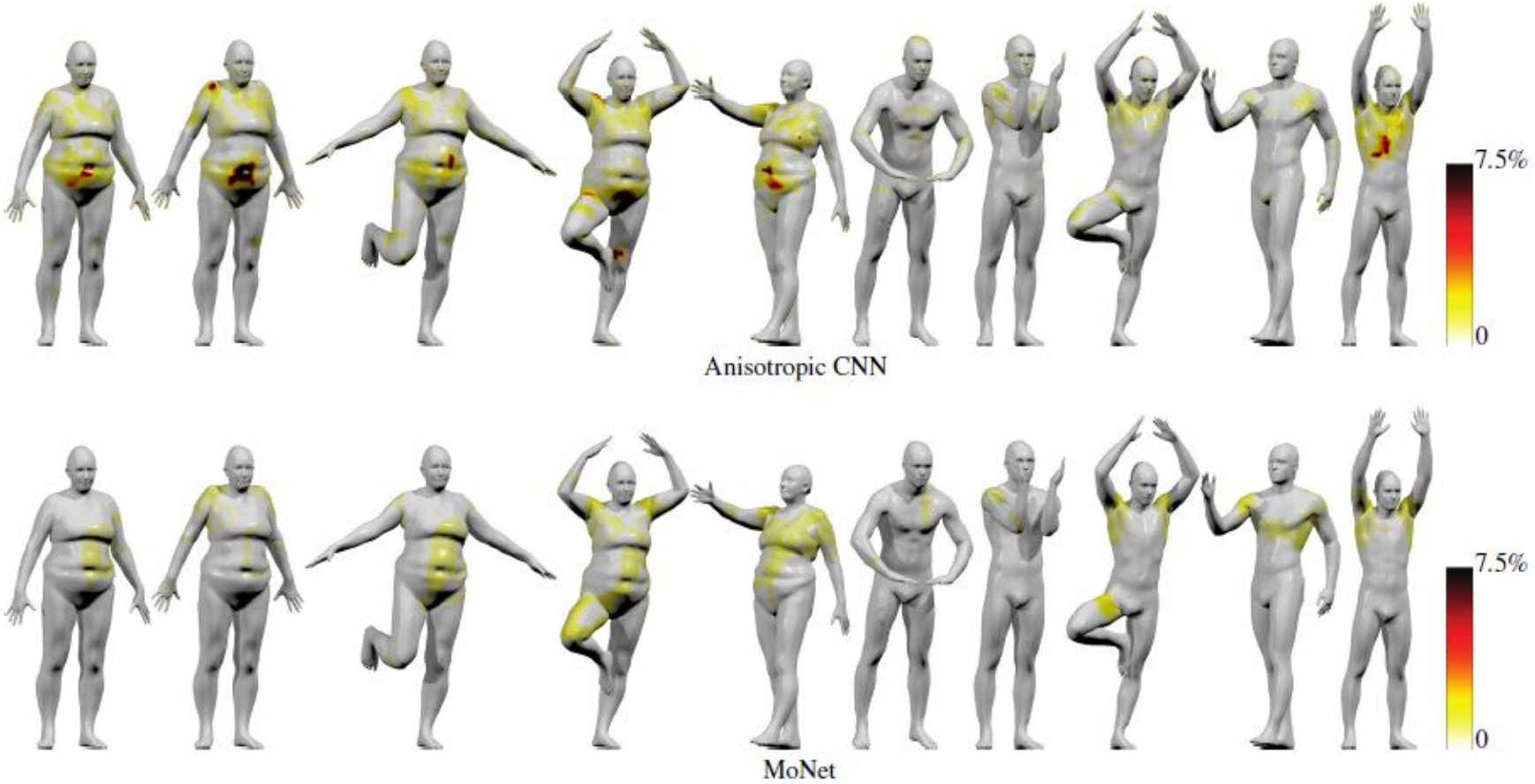


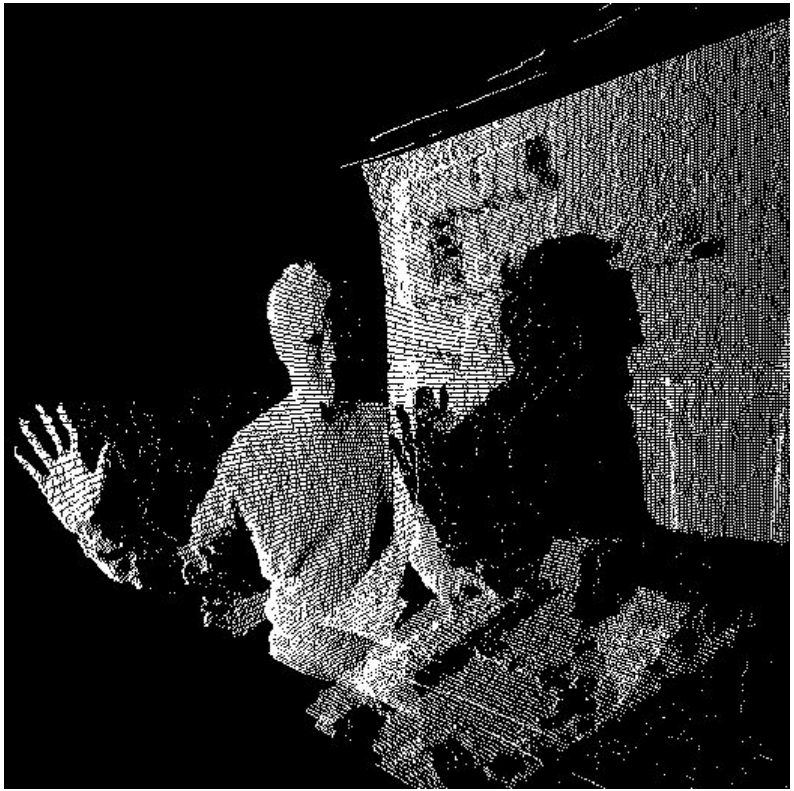
Figure 6. Pointwise error (geodesic distance from groundtruth) of different correspondence methods on the FAUST humans dataset. For visualization clarity, the error values are saturated at 7.5% of the geodesic diameter, which corresponds to approximately 15 cm. Hot colors represent large errors.

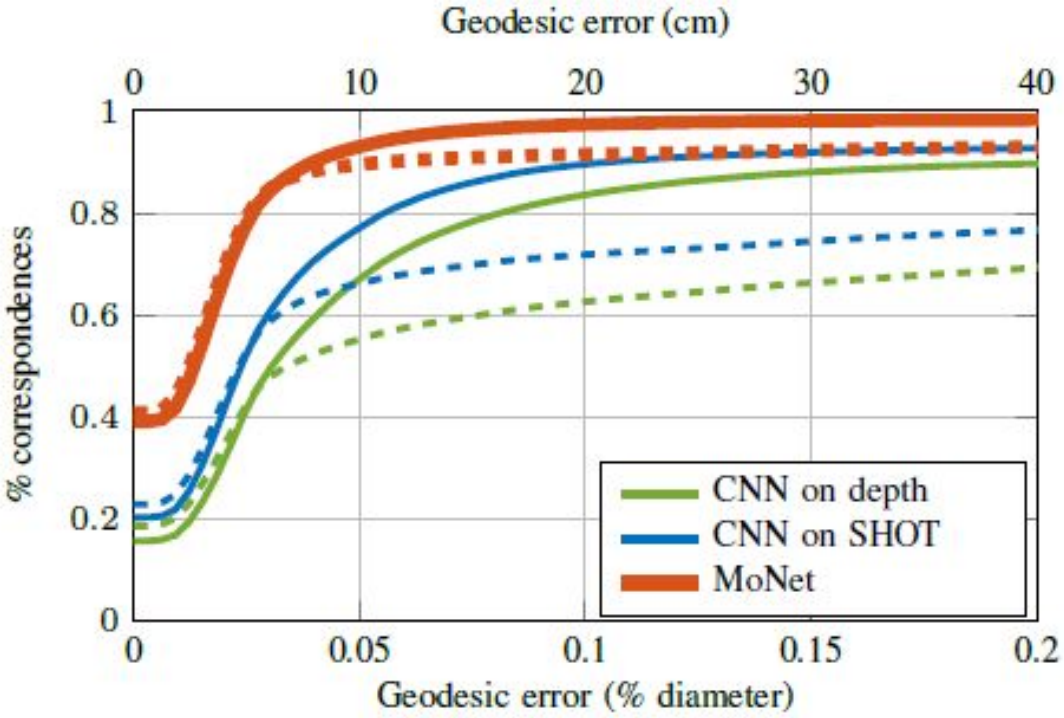
Texture retargeting using correspondences

Figure 7. Examples of correspondence on the FAUST humans dataset obtained by the proposed MoNet method. Shown is the texture transferred from the leftmost reference shape to different subjects in different poses by means of our correspondence.

Synthetic range maps

- Synthetically generated noisy range (depth) maps from FAUST meshes
- 10 range maps for each subject and pose
- Resolution 100x180
- z-axis rotations with increments of 63 degrees
- Total of 1000 range maps
- Kept the groundtruth correspondence.




Geodesic error (cm)
0
10
20
30
40
1
0.8
0.6
0.4
0.2
0
% correspondences
CNN on depth
CNN on SHOT
MoNet
0
0.05
0.1
0.15
0.2
Geodesic error (% diameter)

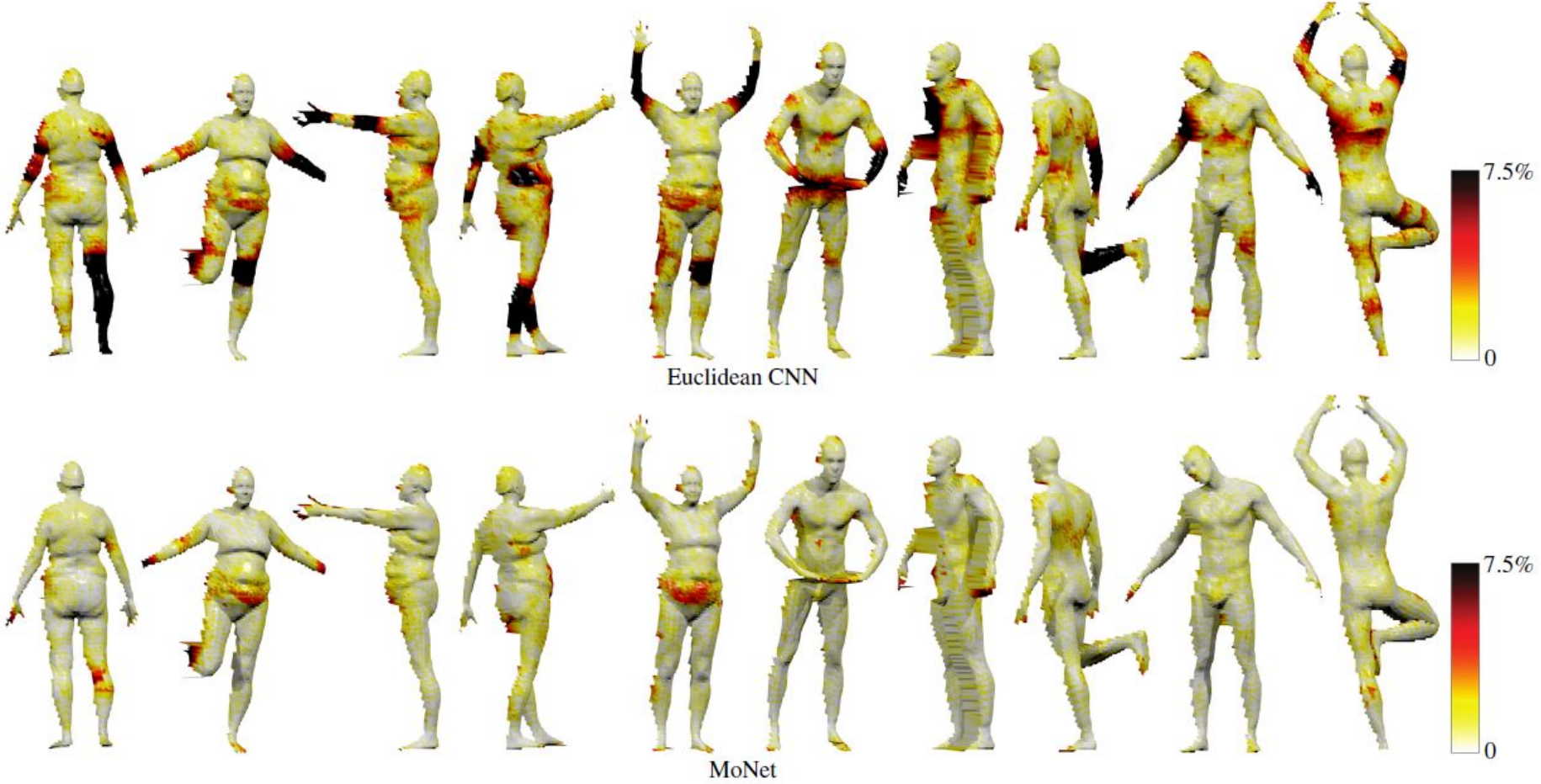


Figure 8. Pointwise error (geodesic distance from groundtruth) of different methods on FAUST range maps. For visualization clarity, the error values are saturated at 7.5% of the geodesic diameter, which corresponds to approximately 15 cm. Hot colors represent large errors.

Figure 9. Visualization of correspondence on FAUST range maps as color code (corresponding points are shown in the same color). Full reference shape is shown on the left. Bottom row show examples of additional shapes from SCAPE and TOSCA datasets.

Q&A

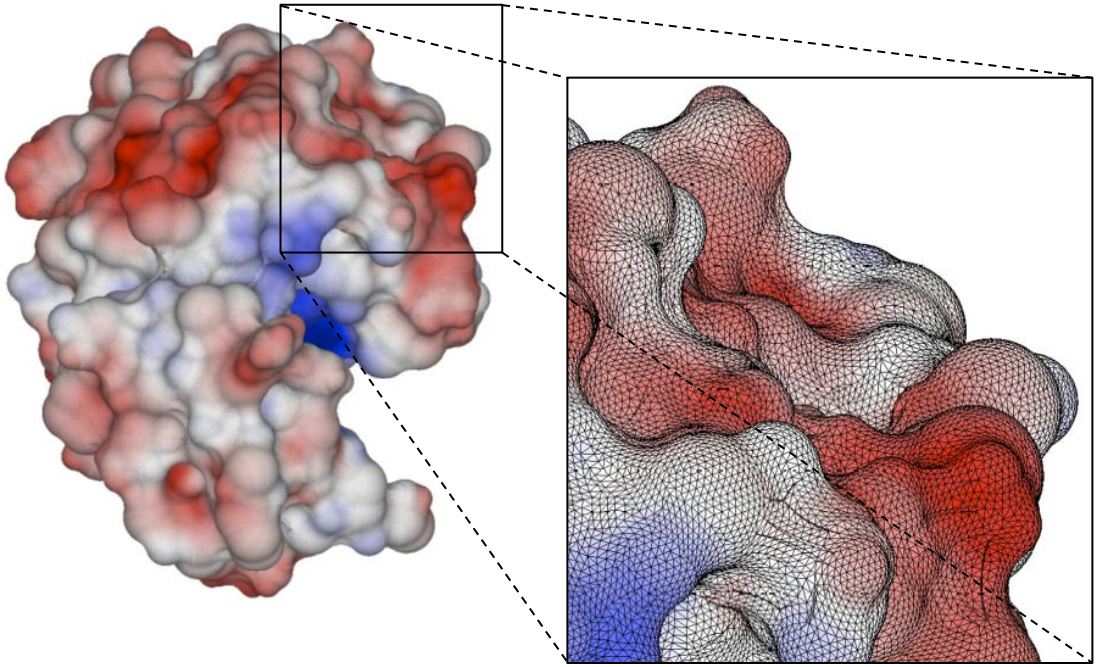
Appendix A

Extra slides

Technical advantages of MoNet

- Euclidean 3D convolution is extremely memory intensive
  - Let’s assume we’re decide on 32x32x32 cube
  - 4 bytes * 32^3 voxels * 50 filters * 10 layers * 100 examples = 6.5536 gigabytes
  - 1 data point every 2m/32 = 6.25cm
- Human body
  - ~2m max height
  - Arm span ~ 1:1 with height
  - 2m**2 skin surface area
- MoNet
  - 2500 vertices
  - sqrt(2m**2 / 2500) = 2.83cm
  - 2500^2*4b*2*100 + 2500*4b*50*10*100 = 5.5 gigabytes

Why do we need this at Totient



To create a molecule similarity measurement tailored to a specific problem based on local 3D shape and surface charge patterns

Appendix B

Spectral approaches

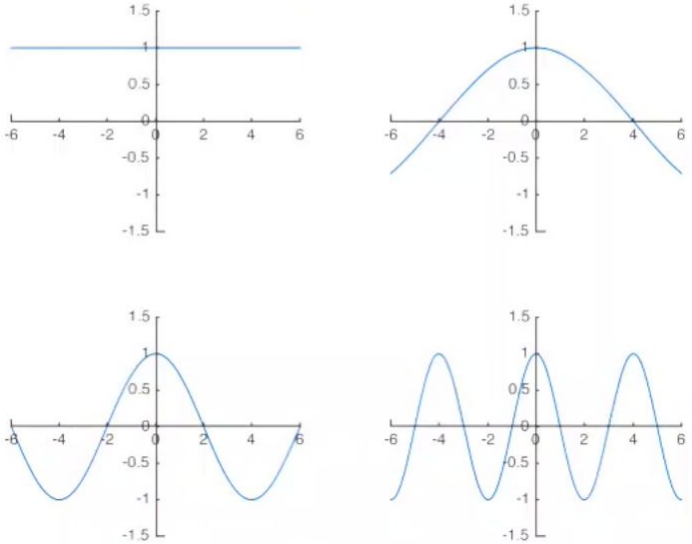
Spectral Approaches

Fourier Convolution Theorem:

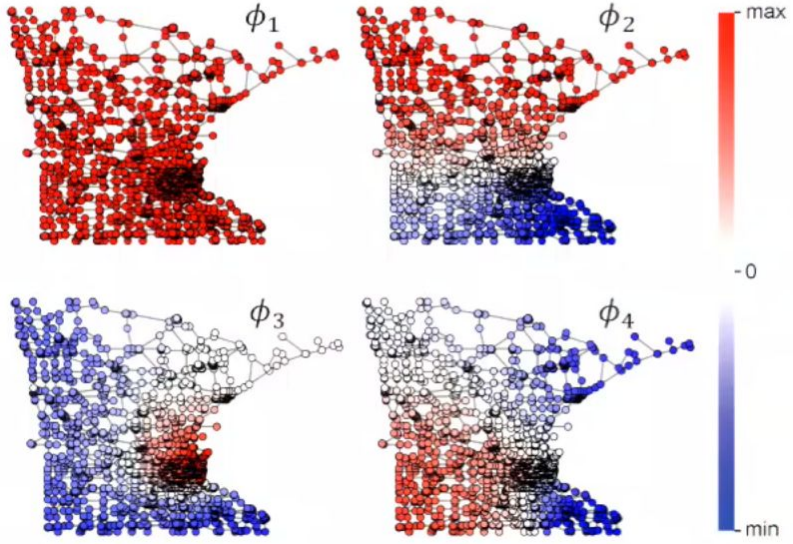
$$\mathcal{F}\{h * f\} = \mathcal{F}\{h\} \cdot \mathcal{F}\{f\}$$

- $\Delta e^{j\omega x} = \frac{d^2}{dx^2} e^{j\omega x} = -\omega^2 e^{j\omega x}$
- $\mathbf{\Delta} = \mathbf{D} - \mathbf{W} = \mathbf{\Phi \Lambda \Phi}^{\mathrm{T}}$

Grids



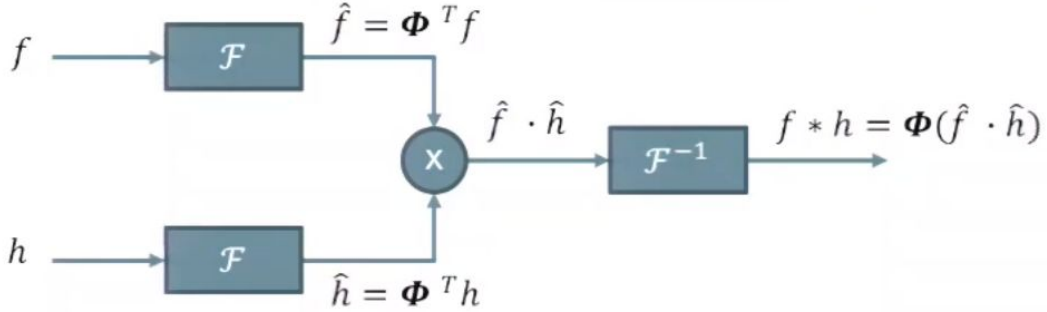
Graphs



Spectral Approaches

Fourier Convolution Theorem:

$$\mathcal{F}\{h * f\} = \mathcal{F}\{h\} \cdot \mathcal{F}\{f\}$$



$$f * h = \boldsymbol{\Phi}\,(\boldsymbol{\Phi}^T h) \cdot (\boldsymbol{\Phi}^T f)$$

Bruna et al. 2013

$$g = \boldsymbol{\Phi}_K diag(\hat{h}_k) \boldsymbol{\Phi}_K^T f$$

Henaff et al. 2015

$$g = \boldsymbol{\Phi} diag(\mathcal{K}\omega) \boldsymbol{\Phi}^T f$$

Defferrard et al. 2016

$$g = \sum_{i=1}^{K} \omega_i T_i(\boldsymbol{\Delta}) f$$

Instability of spectral approaches

f

f

Same signal defined over corresponding points/vertices on two manifolds/graphs f_1 and f_2

Instability of spectral approaches

$\Phi diag(\vartheta)\Phi^T f$

$\Psi diag(\vartheta)\Psi^T f$

Result of the same convolution h over these to f_1 and f_2 gives very different results