Lecture 4:

3D and AI

Simone Melzi, Marco tarini

Milano, 28/07/2021

PhD School – Learning on 3D geometries

LA STATALE Università degli Studi di Milano

SAPIENZA Università di Roma

Why Geometric deep Learning?

Pixels (**Euclidean**)

Slide credit E. *Rodolà*

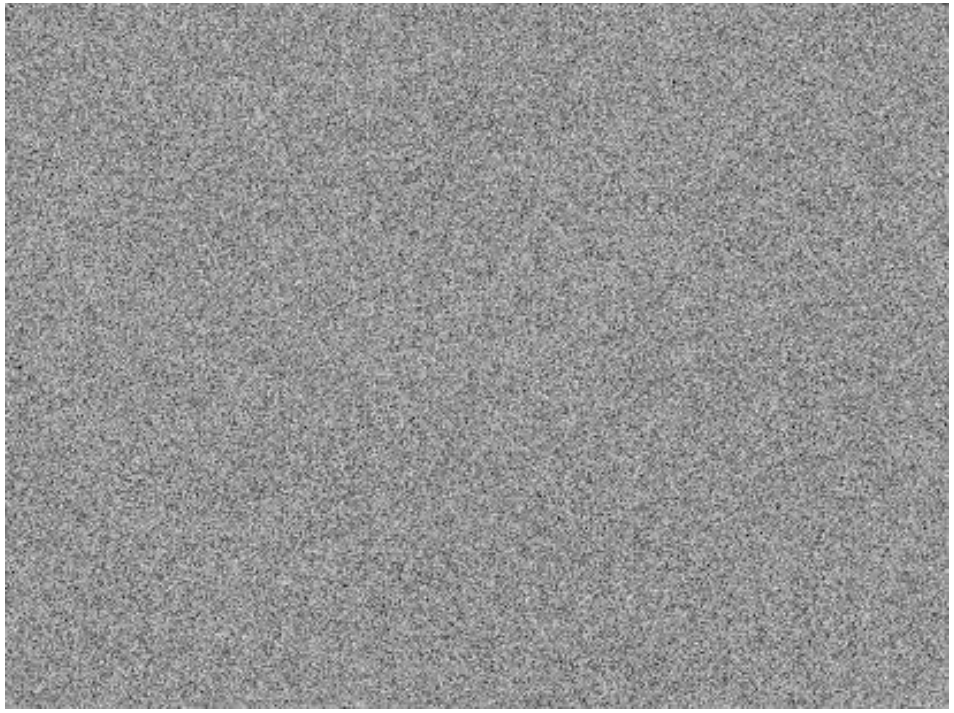
Why Geometric deep Learning?

Geometry (Non-**Euclidean**) Pixels (**Euclidean**)

Slide credit E. *Rodolà*

Need for priors:

Data often owns structural priors in terms of repeating patterns, compositionality, …



Slide credit E. *Rodolà*

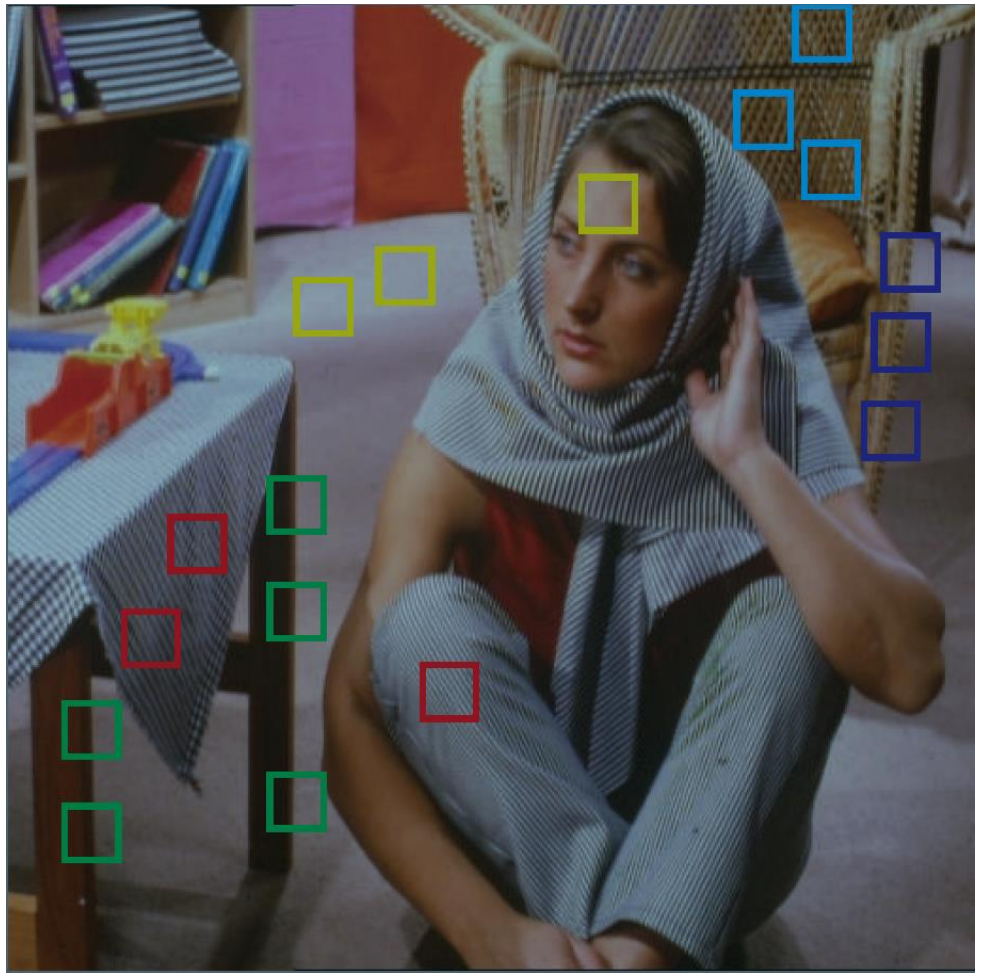
Need for priors:

Data often owns structural priors in terms of repeating patterns, compositionality, …

Slide credit E. *Rodolà*

Self similarity:

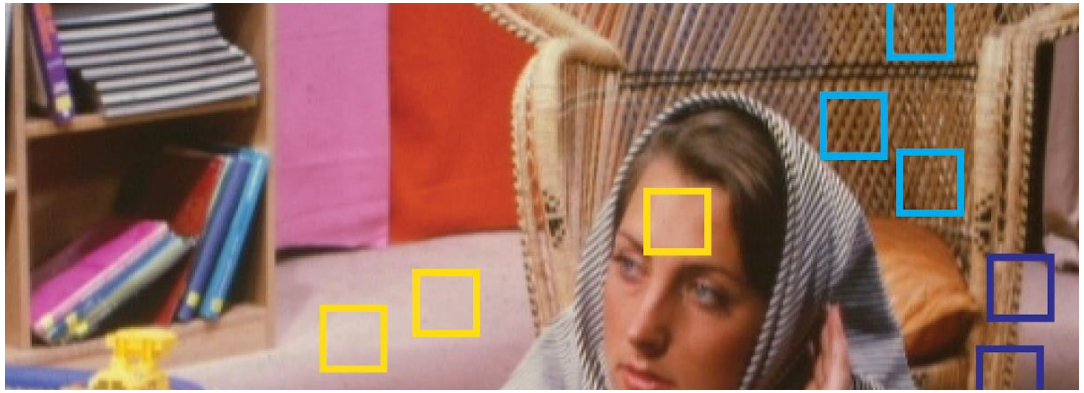
Data tends to be **self-similar** across the domain



Slide credit E. *Rodolà*

Hierarchy:

Translation invariance is desiderable across multiple scales



We expect local features to be invariant to their location in the image:

$$z(T_v p) = z(p), \qquad \forall p, T_v$$

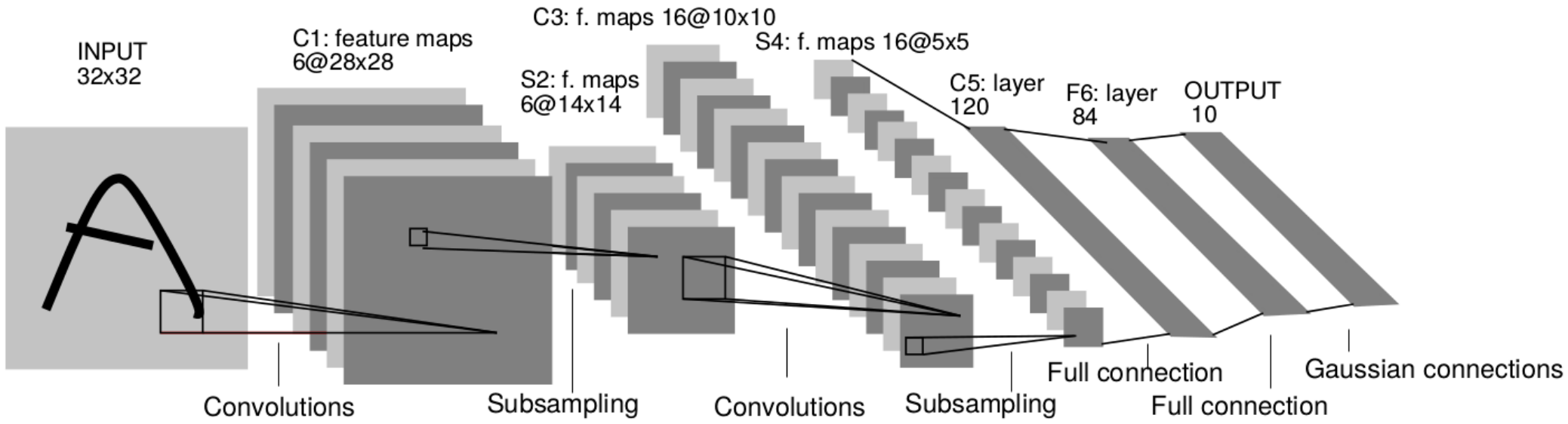
Where p are image patches of variable size.

Slide credit E. *Rodolà*

CNN:

Data are often composed of hierarchical, local, shift-invariant patterns.

CNNs directly exploit this fact as a prior.

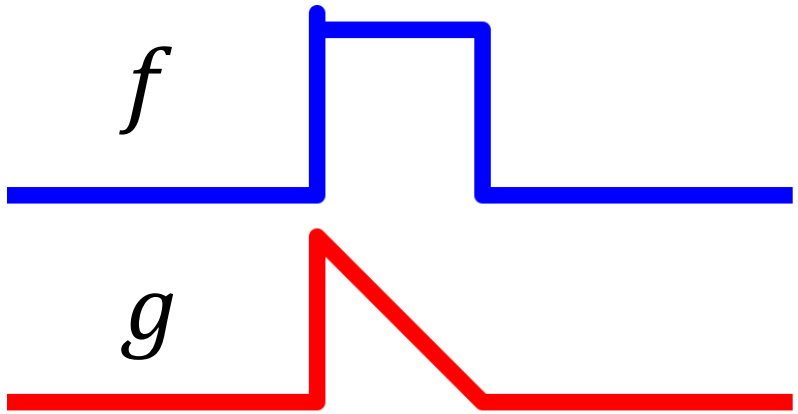


Slide credit E. *Rodolà*

Convolution:

Given two functions $f, g: D = [0, 2\pi] \to \mathbb{R}$
then their **convolution** is the function:

$$(f \star g)(x) = \int_0^{2\pi} f(t)g(x-t)\,dt$$

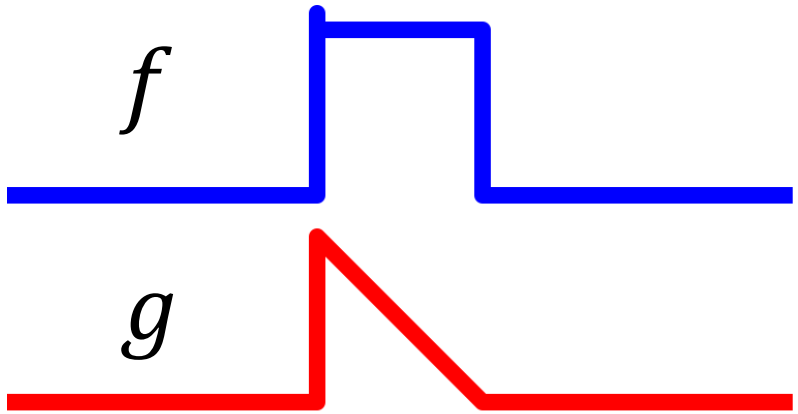


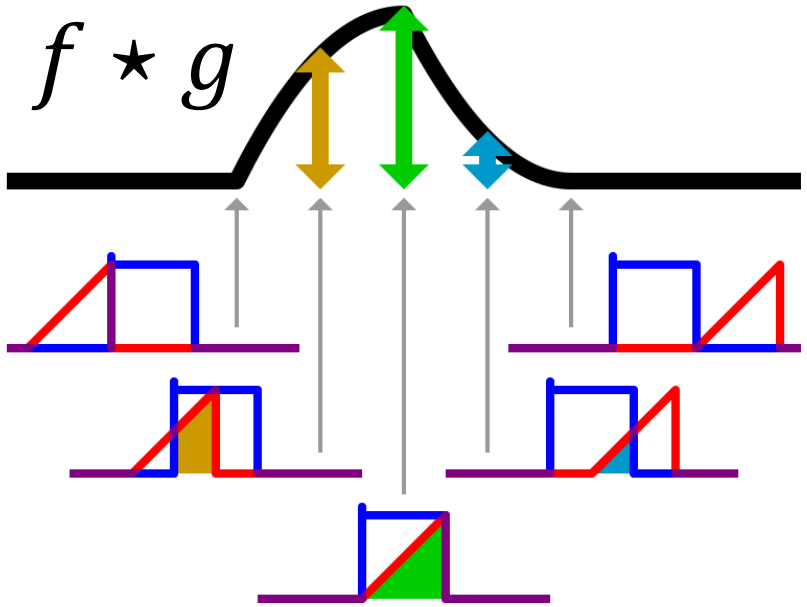
Slide credit E. *Rodolà*

Convolution:

Given two functions $f, g: D = [0, 2\pi] \to \mathbb{R}$
then their **convolution** is the function:

$$(f \star g)(x) = \int_0^{2\pi} f(t)g(x-t)\,dt$$





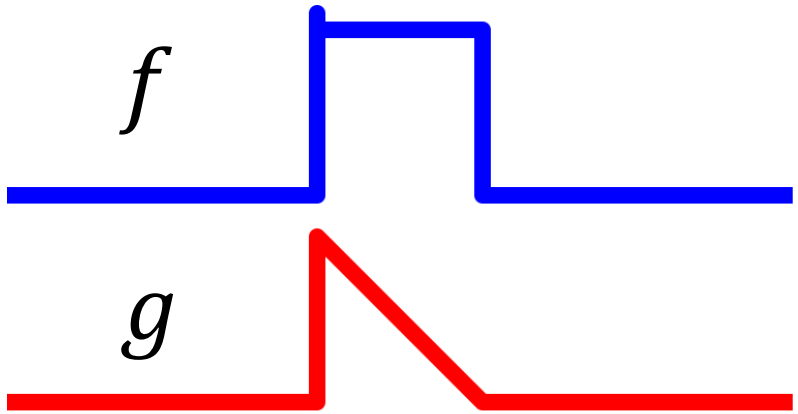
Slide credit E. *Rodolà*

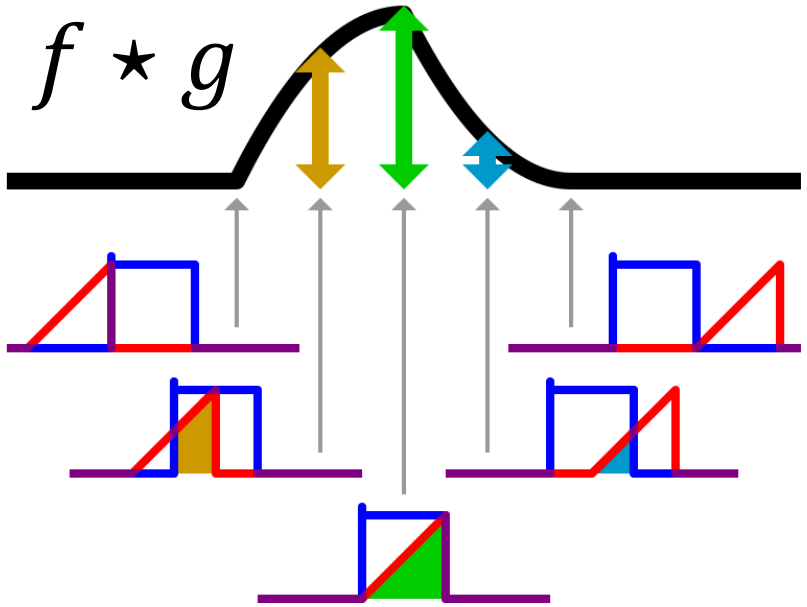
Convolution:

Given two functions $f, g: D = [0, 2\pi] \to \mathbb{R}$
then their **convolution** is the function:

$$(f \star g)(x) = \int_0^{2\pi} f(t)g(x-t)\,dt$$

Feature map — Kernel

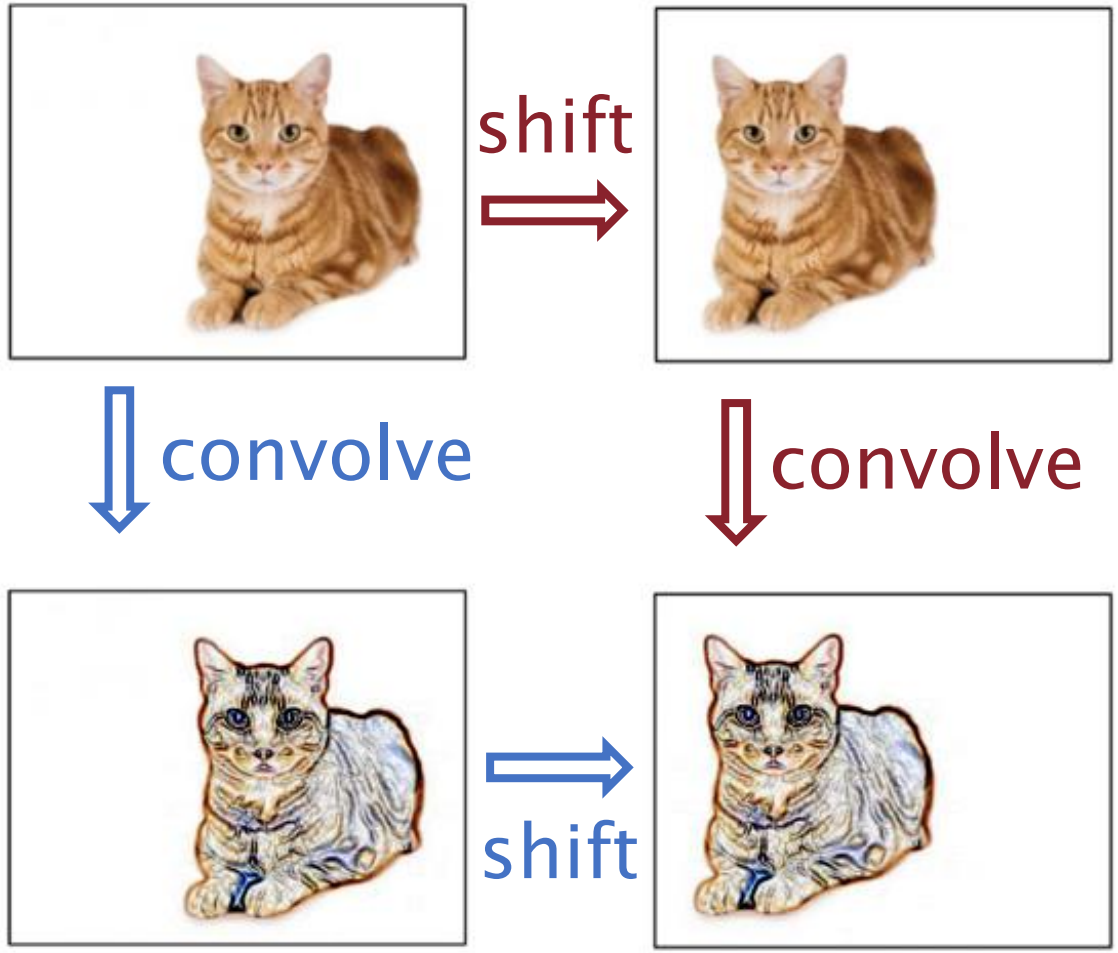




Slide credit E. *Rodolà*

Convolution: Shift-equivariance

Convolution is shift-equivariant



Slide credit E. *Rodolà*

Convolution: Linearity

Convolution could be seen as the application of a linear operator $\mathcal{G}$

$$\mathcal{G}f(x) = (f \star g)(x) = \int_0^{2\pi} f(t)g(x-t)\,dt$$

$\mathcal{G}$ is linear: $\mathcal{G}(\alpha f(x)) = \alpha\mathcal{G}(f(x))$

$$\mathcal{G}(f+h)(x) = \mathcal{G}\big(f(x)\big) + \mathcal{G}(h(x))$$

The shift-equivariance can be written as:

$\mathcal{G}(\mathcal{T}f) = \mathcal{T}(\mathcal{G}f)$ $\Leftrightarrow$ $\mathcal{G}$ and $\mathcal{T}$ commute

Slide credit E. *Rodolà*

Discrete convolution:

In the discrete setting we deal with vectors $\boldsymbol{f}$ and $\boldsymbol{g}$ and we define the convolution as:

$$(\boldsymbol{f} \star \boldsymbol{g})[x_i] = \sum_{j=1}^{n} \boldsymbol{f}[x_j]\, \boldsymbol{g}[x_i - x_j]$$

Assuming cyclic boundary conditions the convolution operator can be encoded as a matrix (Toeplitz matrix)

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

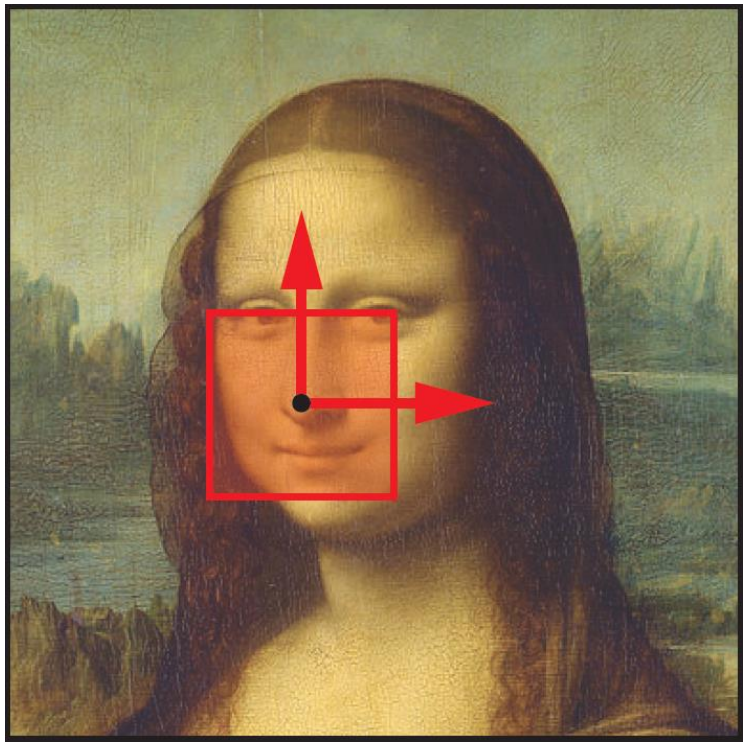
Slide credit E. *Rodolà*

Discrete convolution:

On 2D domains (RGB images $f: \mathbb{R}^2 \to \mathbb{R}^3$) for each channel

$$(\boldsymbol{f} \star \boldsymbol{g})[x_i, y_l] = \sum_{j=1}^{nr} \sum_{k=1}^{nc} \boldsymbol{f}[x_j, y_k]\, \boldsymbol{g}[x_i - x_j, y_l - y_k]$$

We get the classical interpretation of the moving window



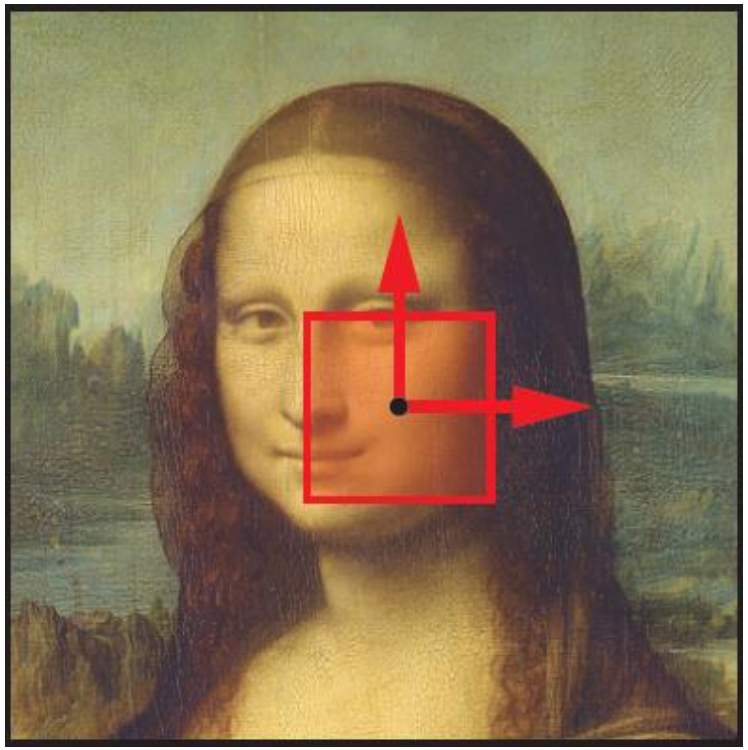
Slide credit E. *Rodolà*

Discrete convolution:

On 2D domains (RGB images $f: \mathbb{R}^2 \rightarrow \mathbb{R}^3$) for each channel

$$(\boldsymbol{f} \star \boldsymbol{g})[x_i, y_l] = \sum_{j=1}^{nr} \sum_{k=1}^{nc} \boldsymbol{f}[x_j, y_k]\, \boldsymbol{g}[x_i - x_j, y_l - y_k]$$

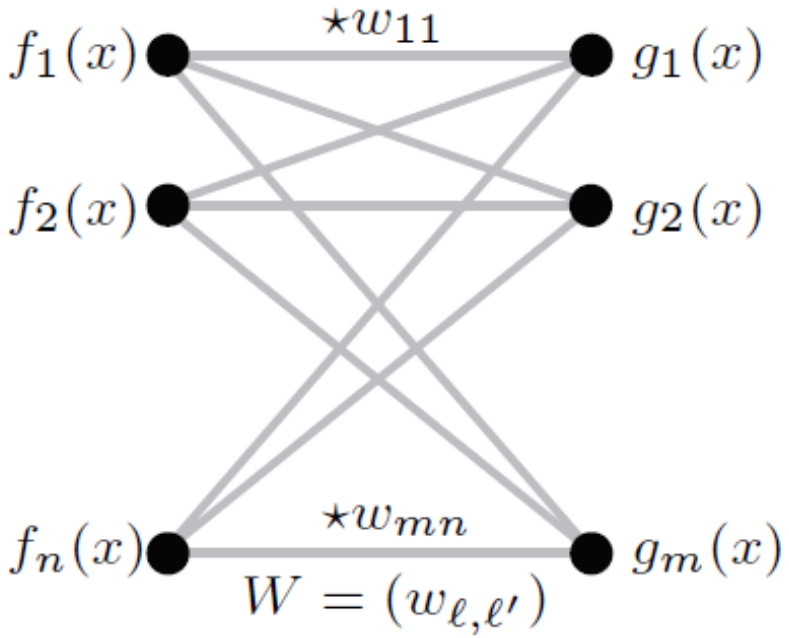
We get the classical interpretation of the moving window



Slide credit E. *Rodolà*

Convolutional Neural Network

Main Idea: Compose equivariant layers (based on convolutions)

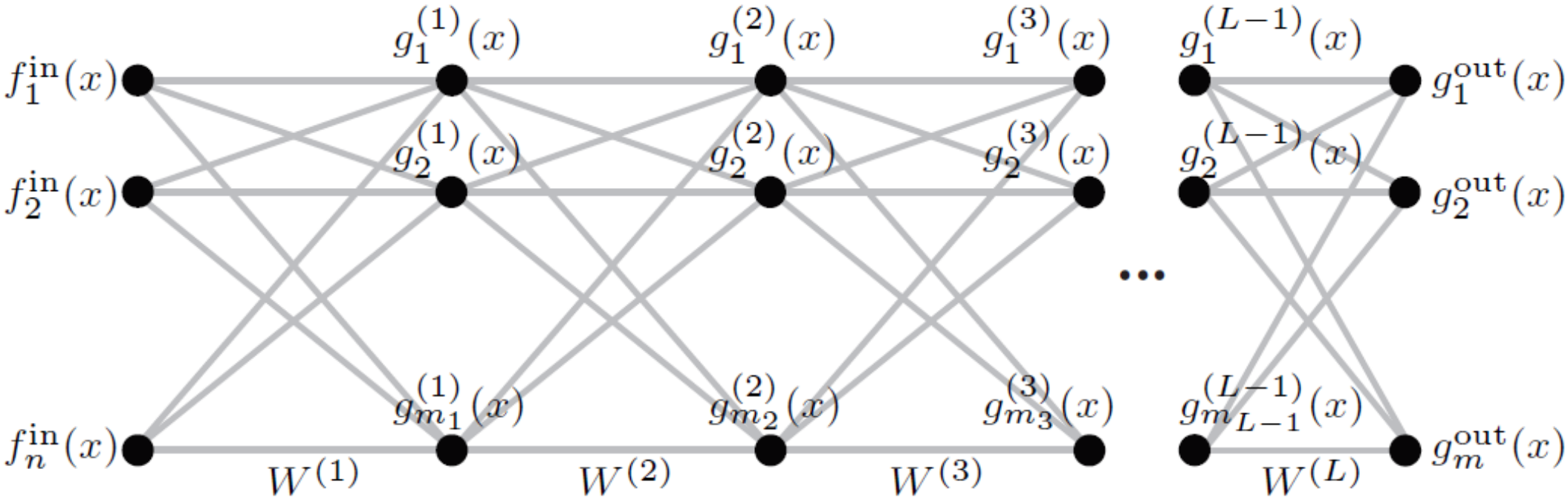


Single convolutional layer

Slide credit E. *Rodolà*

Convolutional Neural Network

Main Idea: Compose equivariant layers (based on convolutions)



CNN consisting of L convolutional layers

Slide credit E. *Rodolà*

Convolutional Neural Network

Main Idea: Compose equivariant layers (based on convolutions)

Conv. Layer:

$$g_l^{(k)}(x) = \sigma\left(\sum_{l'=1}^{m_{k-1}} (g_{l'}^{(k-1)} \star w_{l,l'}^{(k)})(x)\right) \quad \begin{matrix} l = 1, \dots, m_k \\ l' = 1, \dots, m_{k-1} \end{matrix}$$

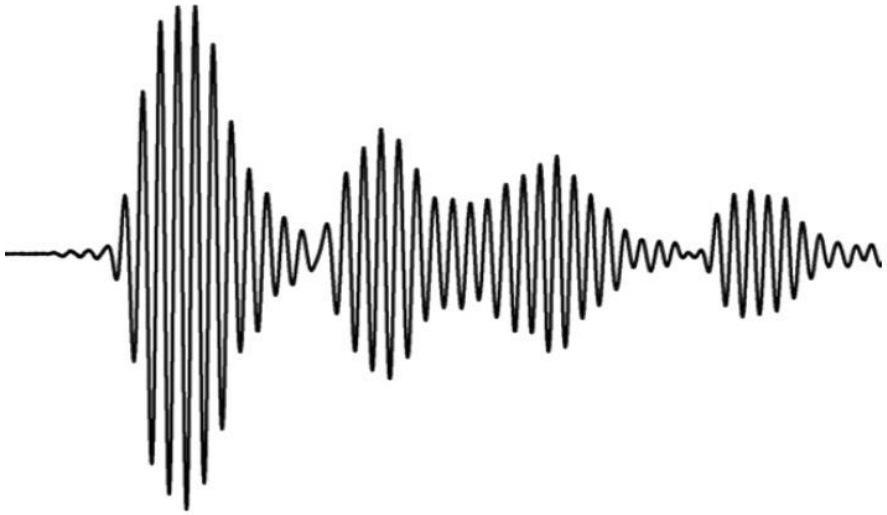
Activation: $\sigma(x) = \max(\{0, x\})$ rectified linear unit (ReLU)

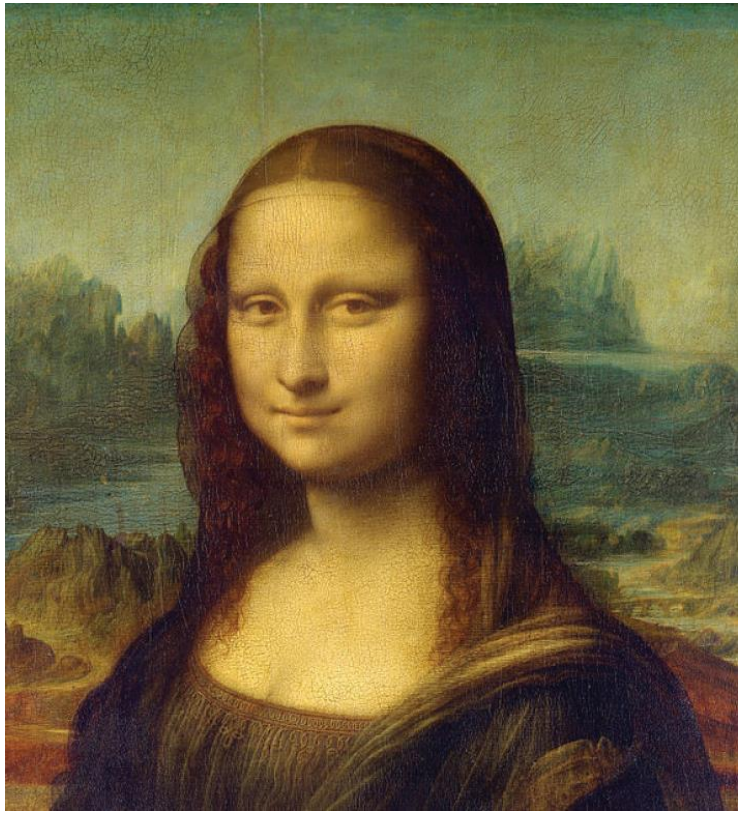
Parameters: Filters of all layers $W^{(1)}, \dots, W^{(L)}$

Slide credit E. *Rodolà*

Usual data for deep learning:

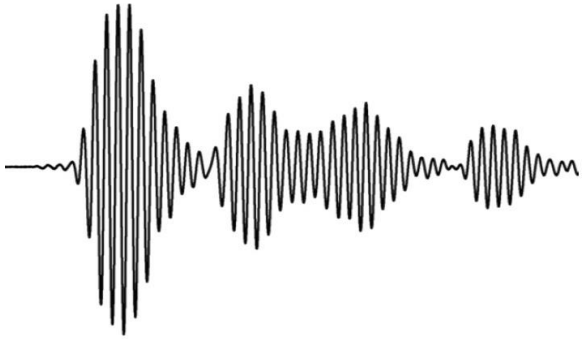
Audio signals and images





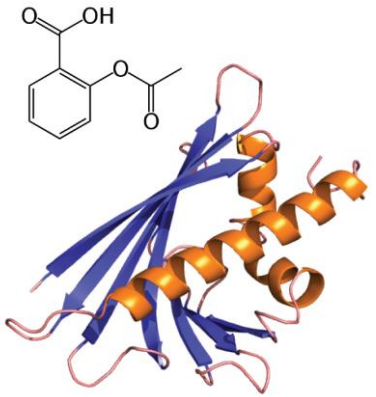
Slide credit E. *Rodolà*

Different signals:

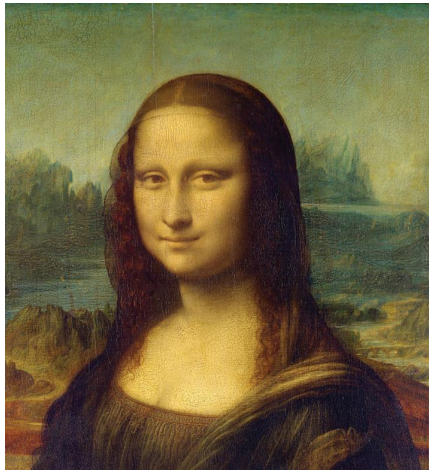


Audio

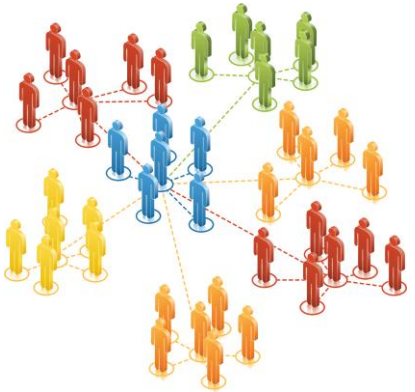
Road maps



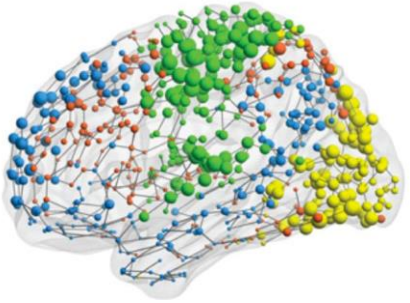
Molecules



Image



Social networks



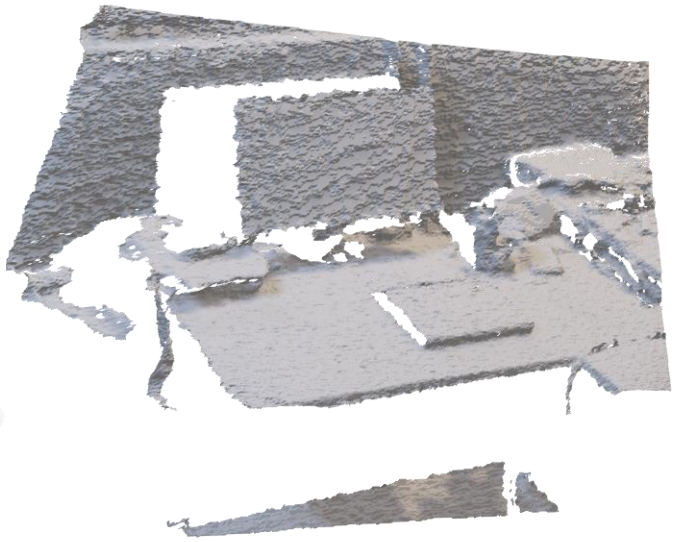
Functional networks

Slide credit E. *Rodolà*

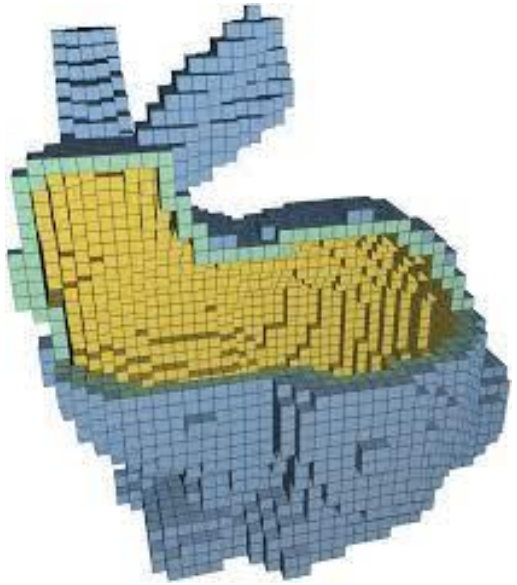
Different domains:

We want to consider non-Euclidean domains

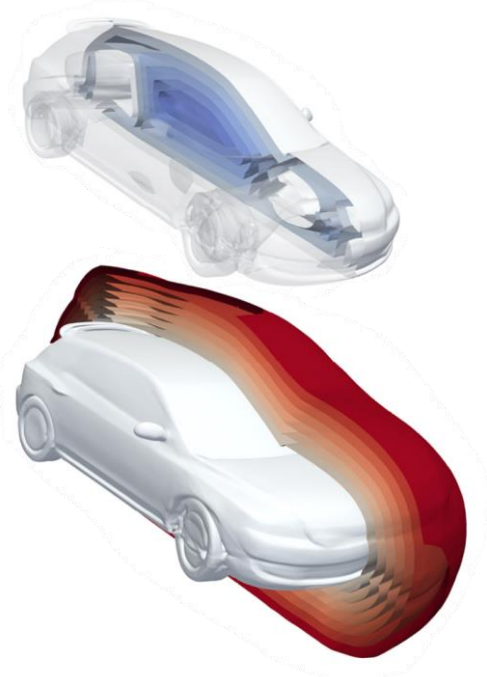
Point clouds



Range map



Volumetric



Implicit

"Fast Parallel Surface and Solid Voxelization on GPUs", M. Schwarz et al., 2010

"Implicit Geometric Regularization for Learning Shapes", A. Gropp et al., 2020

… and Meshes:

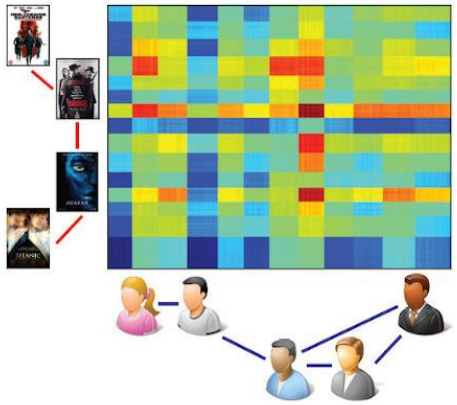
Piecewise linear approximation of manifolds.

We want to inject geometry in the Deep learning processing.

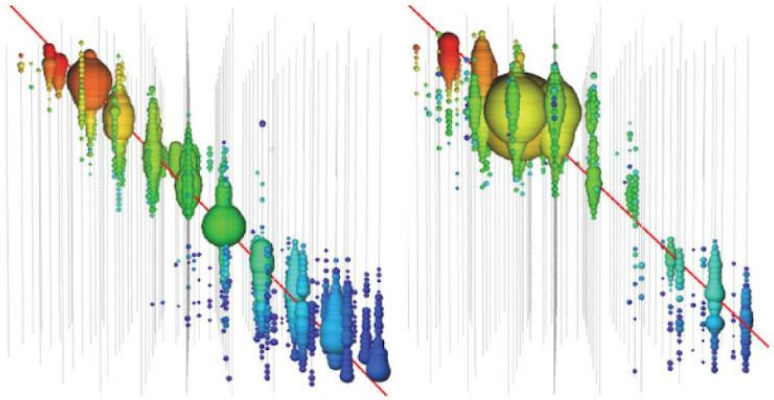
Is this new?

We want to do learning on the geometry of the domain!

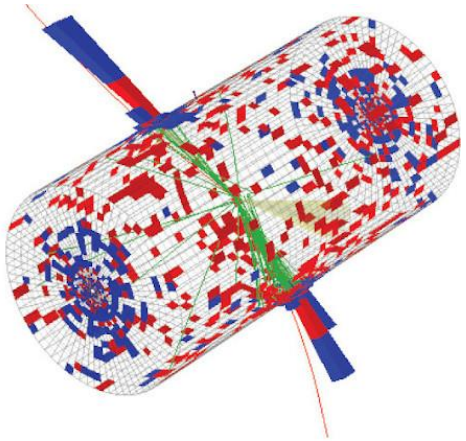
Applications:



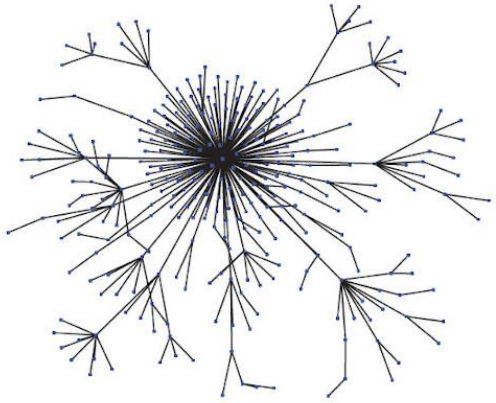
Recommender system



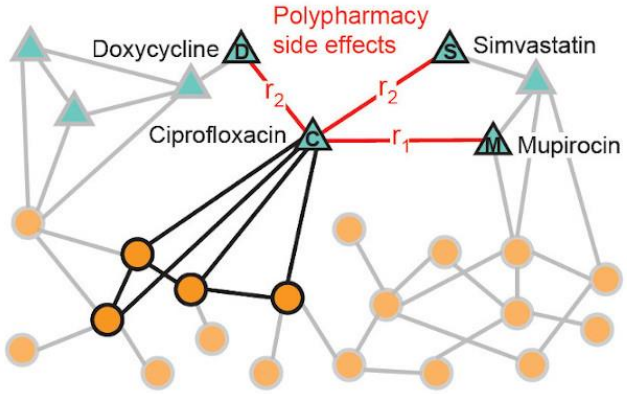
Neutrino detection



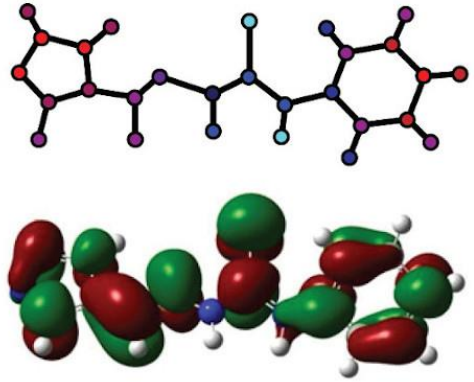
LHC



Fake news detection



Drug repurposing



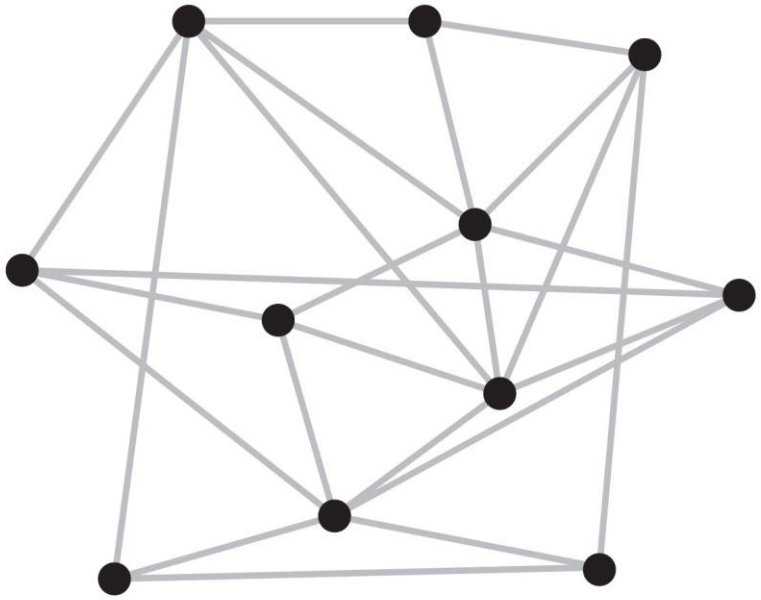
Chemistry

Slide credit E. *Rodolà*

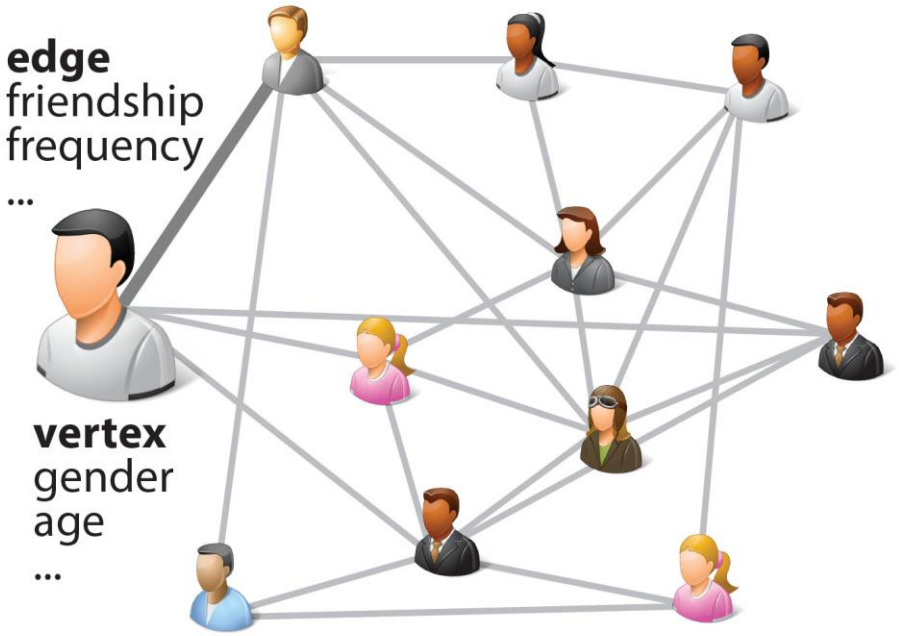
Domain structure:

Slide credit E. *Rodolà*

Signals on a domain:



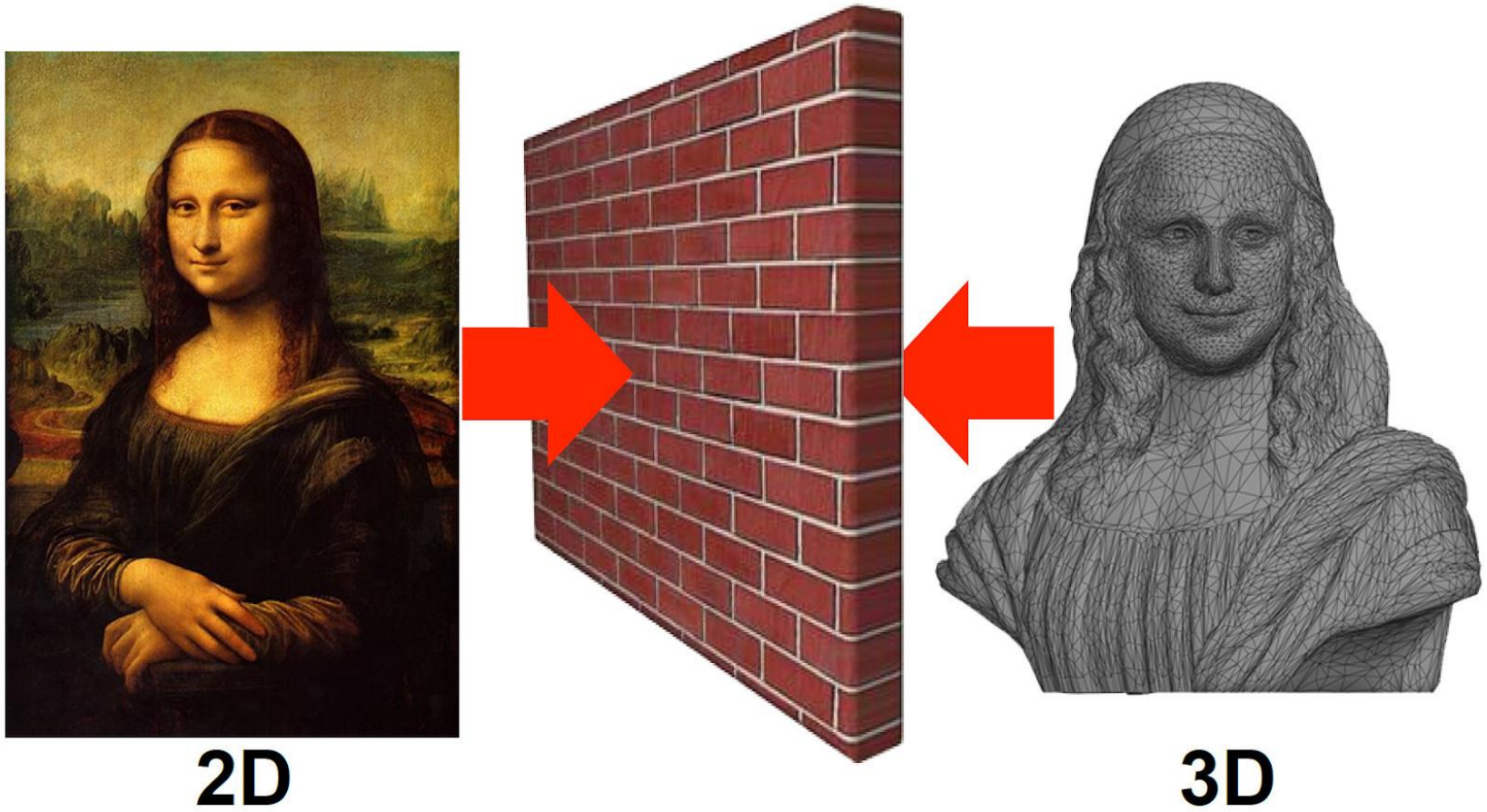
Domain structure



Signals on a domain

Slide credit E. *Rodolà*

Signals vs domain:



2D

3D

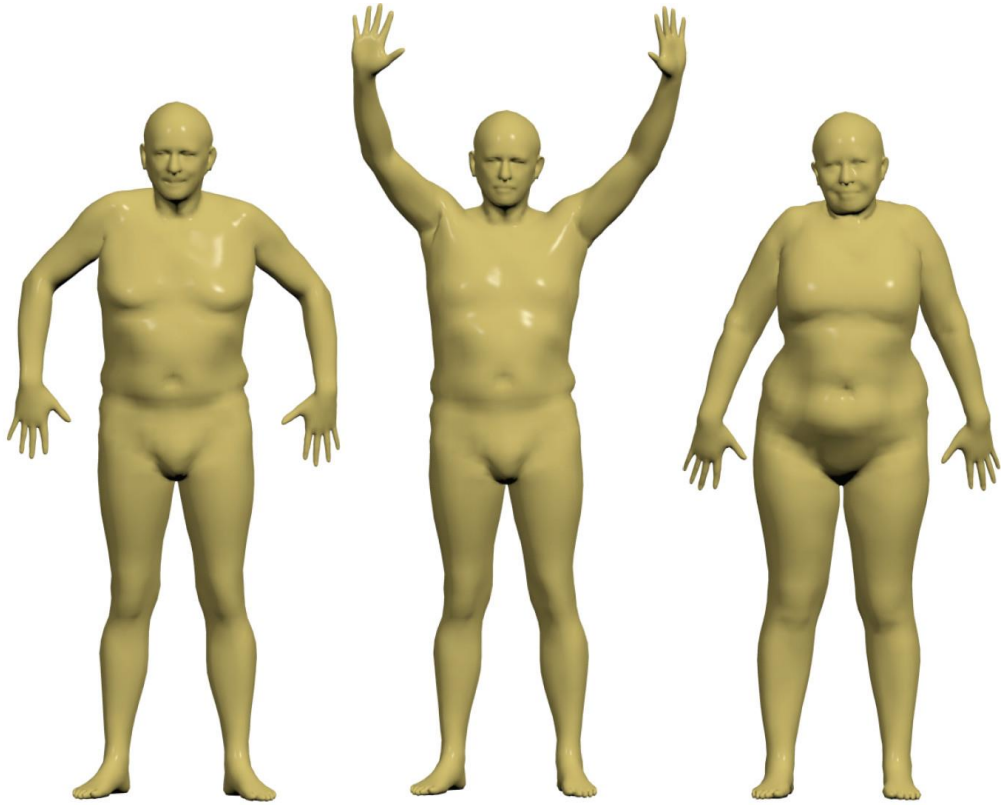
Only signal
(fixed domain/template)

Only domain
(constant/no signal)

Slide credit E. *Rodolà*

Fixed or non fixed domains:

Fixed domain

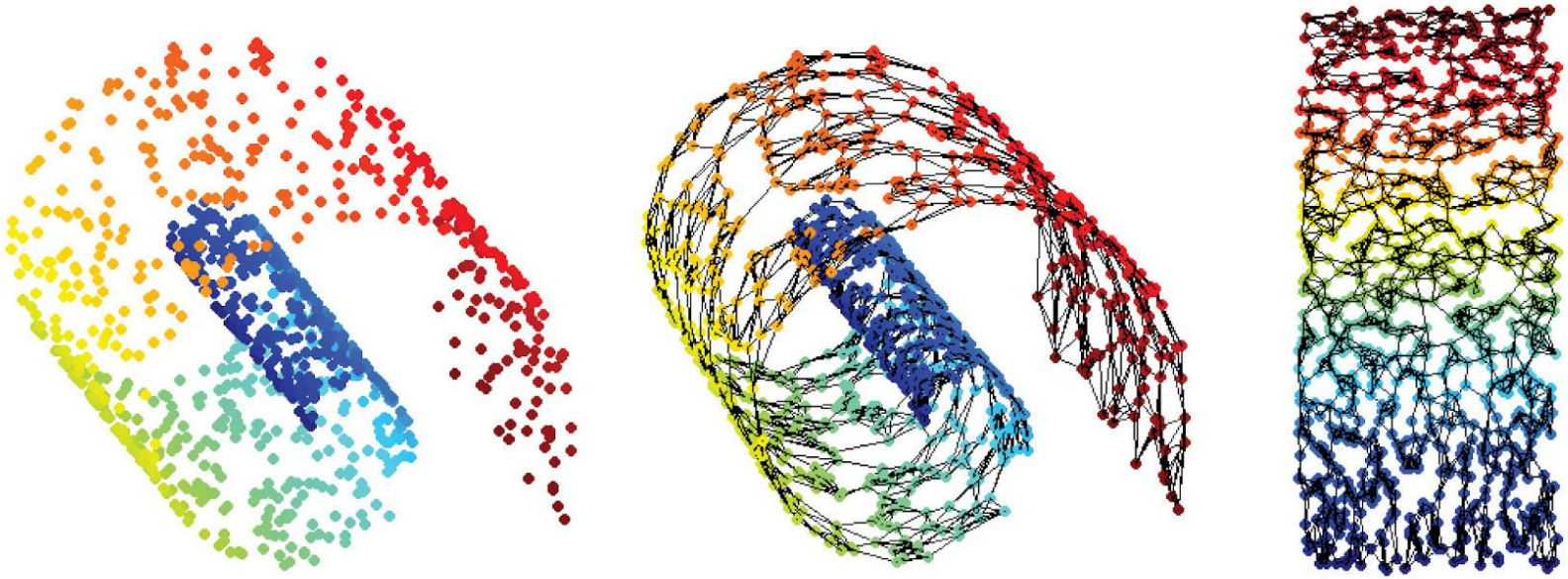


Deformable domain

Slide credit E. *Rodolà*

Geometric ≠ metric learning:

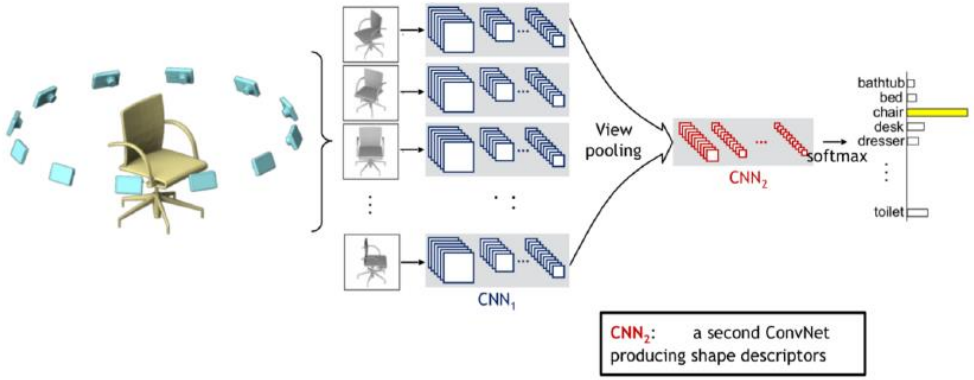
In manifold learning we look for a (high dimensional) manifold that justifies a given set of data samples



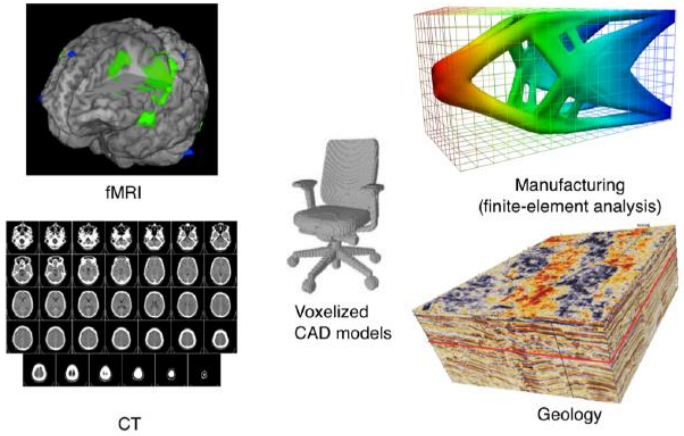
In geometric deep learning the data has a well-known geometric structure

Slide credit E. *Rodolà*

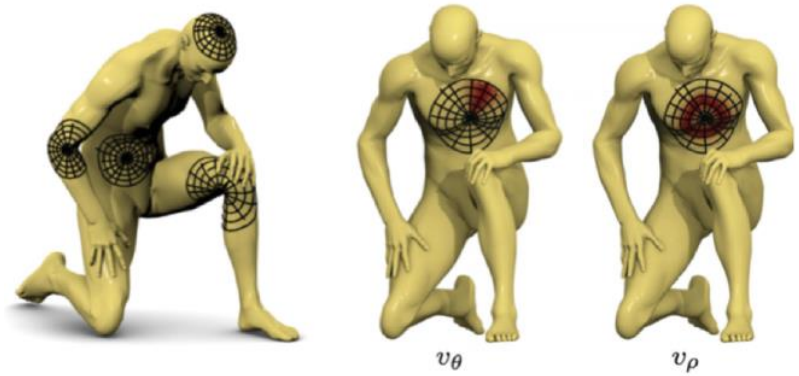
Geometric deep learning



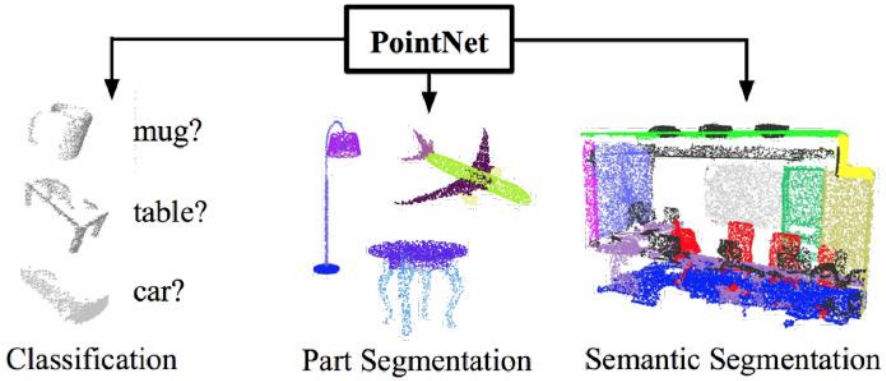
View-based



Volumetric



Intrinsic (surface-based)



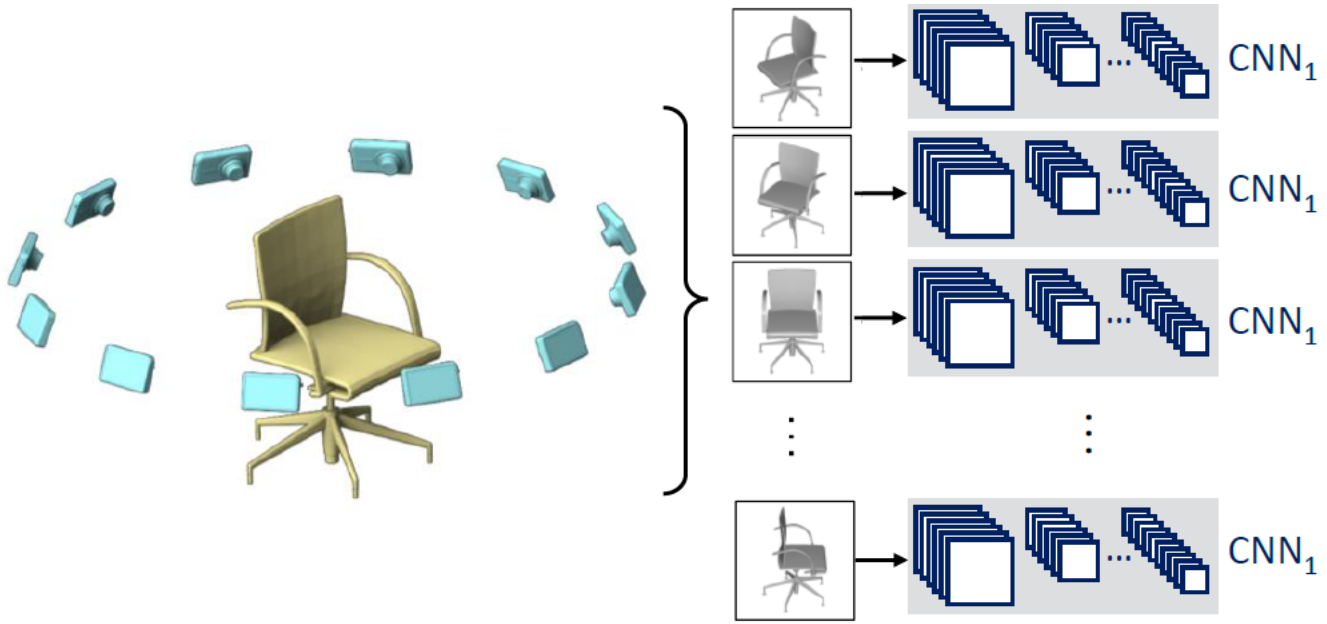
Point-based

https://geometricdeeplearning.com/

Slide credit M. *Ovsjanikov*

Multi-view CNN:

Slide credit E. *Rodolà*

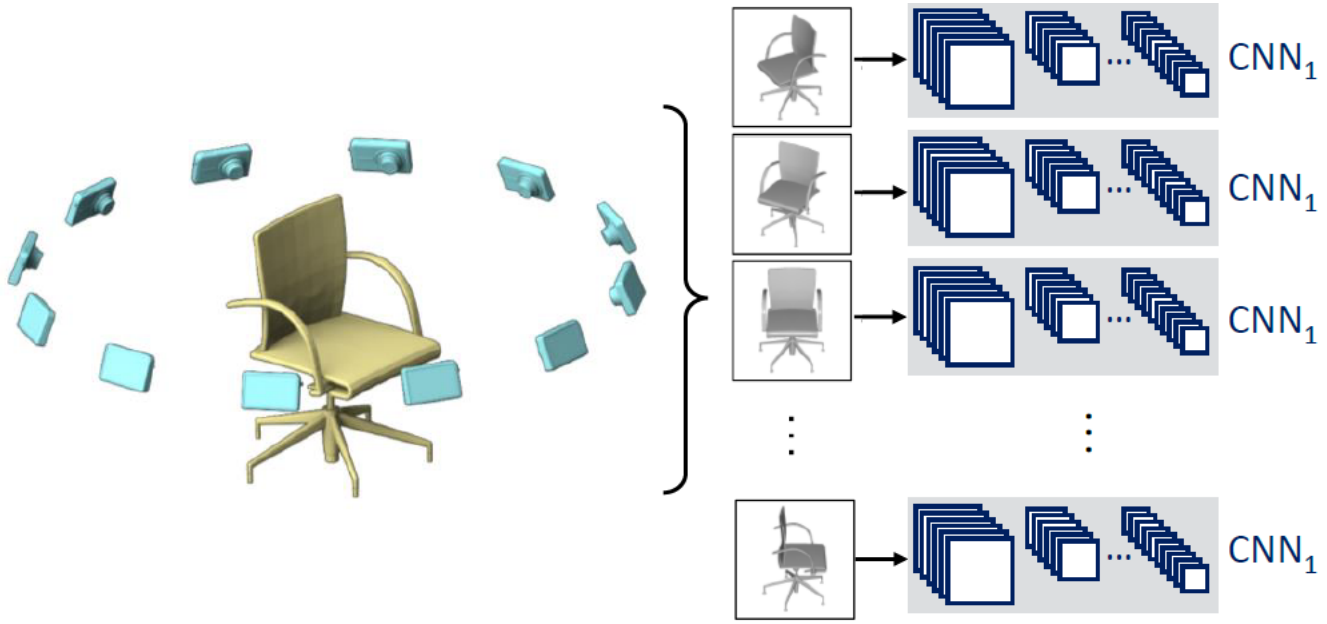


- Represent 3D object as a collection of range images.

"Multi-view Convolutional Neural Networks for 3D Shape Recognition", Su et al., 2015

Multi-view CNN:

Slide credit E. *Rodolà*

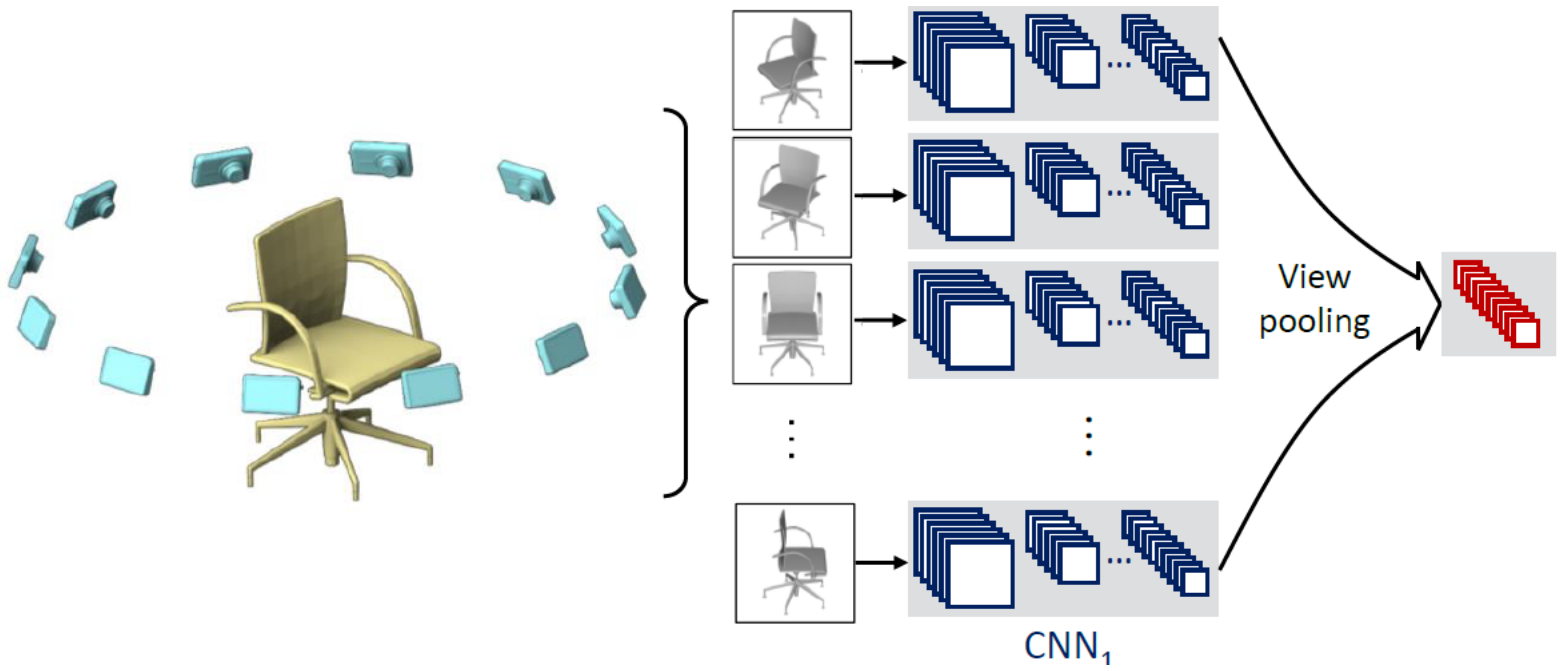


- Represent 3D object as a collection of range images.
- CNN_1: extracts image features (views share the parameters)

"Multi-view Convolutional Neural Networks for 3D Shape Recognition", Su et al., 2015

Multi-view CNN:

Slide credit E. *Rodolà*

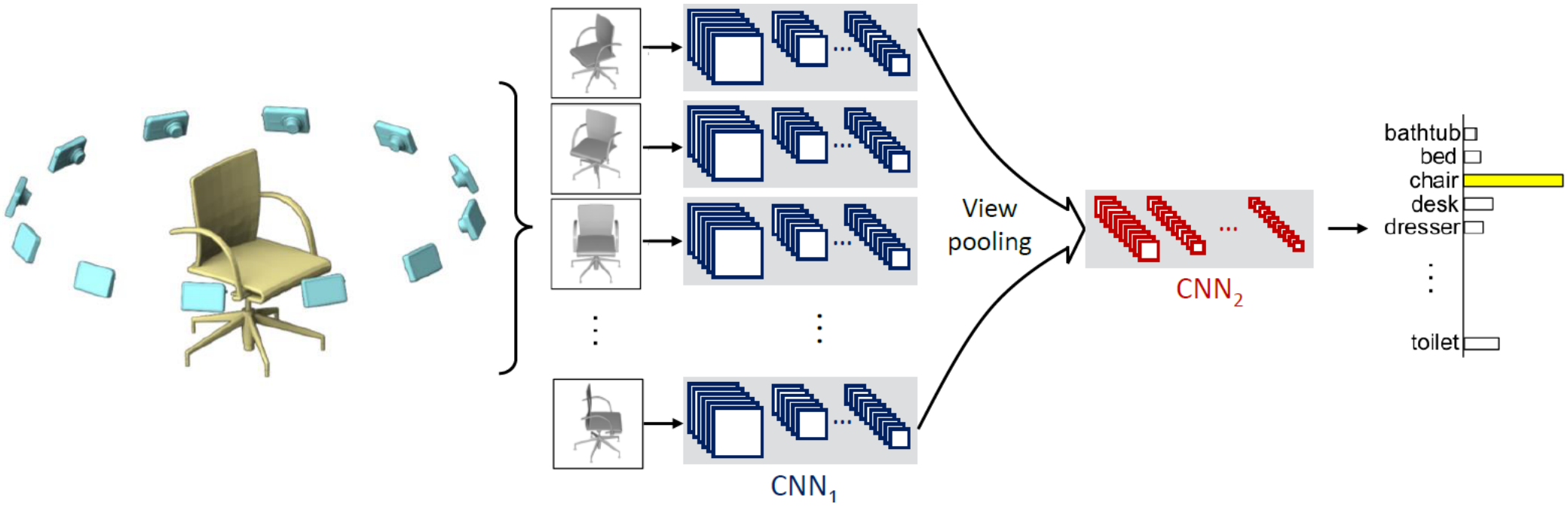


- Represent 3D object as a collection of range images.
- CNN_1: extracts image features (views share the parameters)
- Element-wise max pooling across all views

"Multi-view Convolutional Neural Networks for 3D Shape Recognition", Su et al., 2015

Multi-view CNN:

Slide credit E. *Rodolà*



- Represent 3D object as a collection of range images.
- CNN_1: extracts image features (views share the parameters)
- Element-wise max pooling across all views
- CNN_2: produces shape descriptors + final prediction

"Multi-view Convolutional Neural Networks for 3D Shape Recognition", Su et al., 2015

Multi-view CNN applications

Slide credit E. *Rodolà*

classify → «Chair»

classify → «Chair»

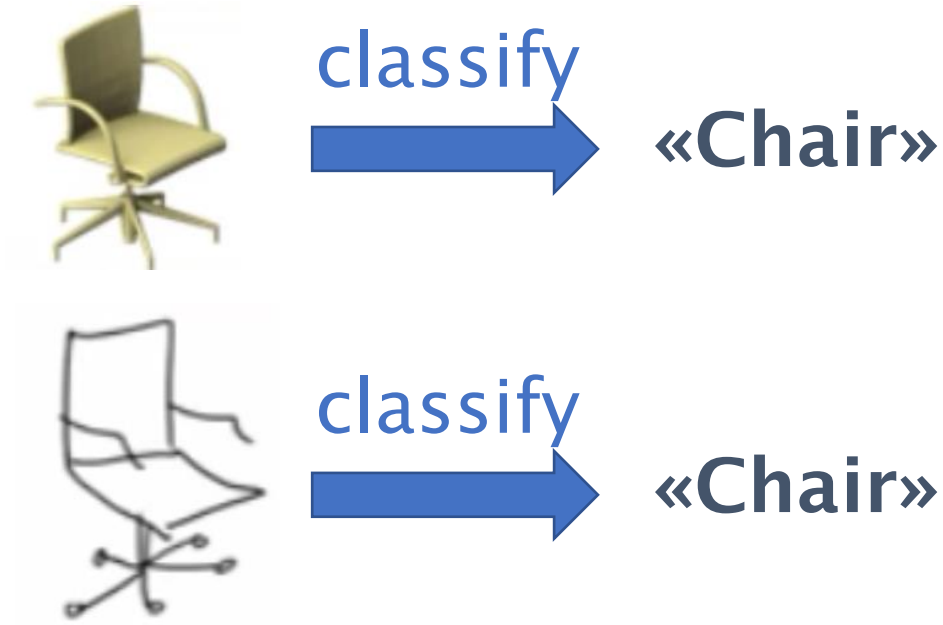
3D shape/sketch classification

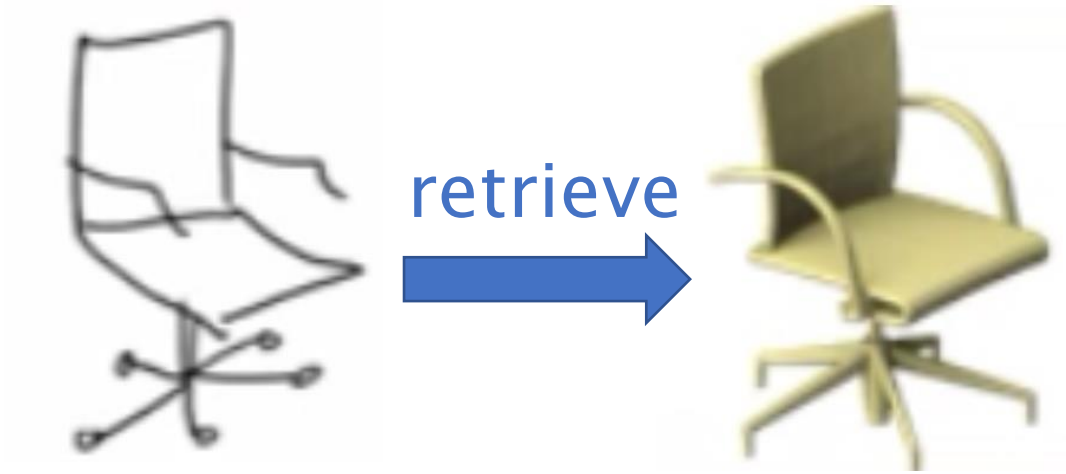
- Pretrained on ImageNet
- Fine-tuned on 2D views

"Multi-view Convolutional Neural Networks for 3D Shape Recognition", Su et al., 2015

Multi-view CNN applications

Slide credit E. *Rodolà*





3D shape/sketch classification

- Pretrained on ImageNet
- Fine-tuned on 2D views

Sketch-based shape retrieval

- Render views with hand-drawn style (edge maps)

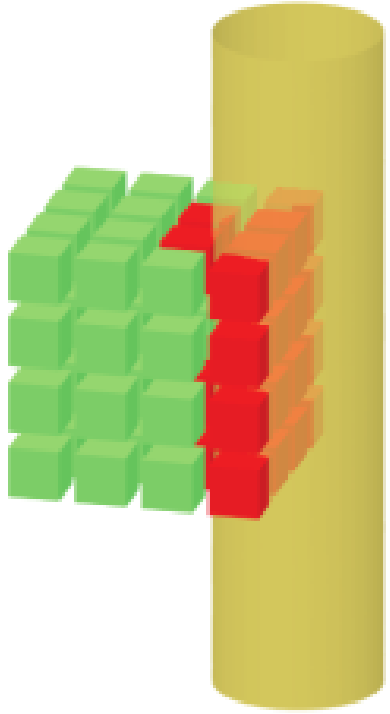
"Multi-view Convolutional Neural Networks for 3D Shape Recognition", Su et al., 2015

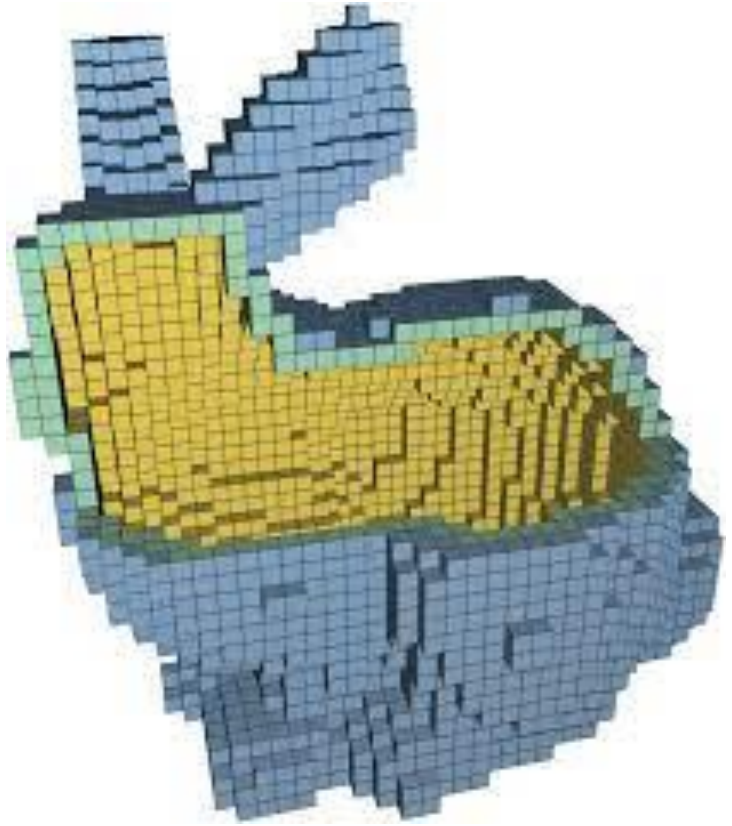
3D Shape-Net:

Slide credit
E. *Rodolà*

Volumetric representation
(binary voxels on 3D grid)

3D convolutional network



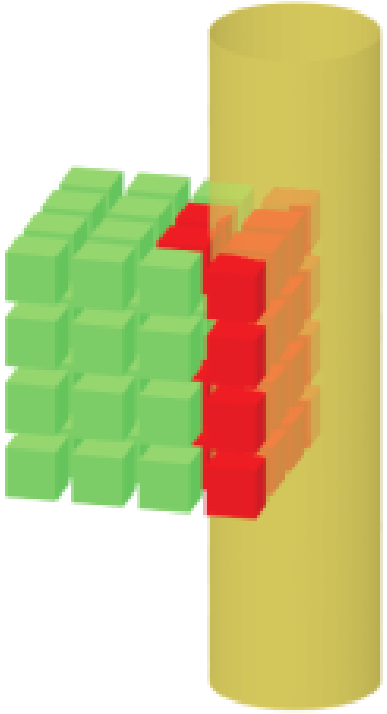


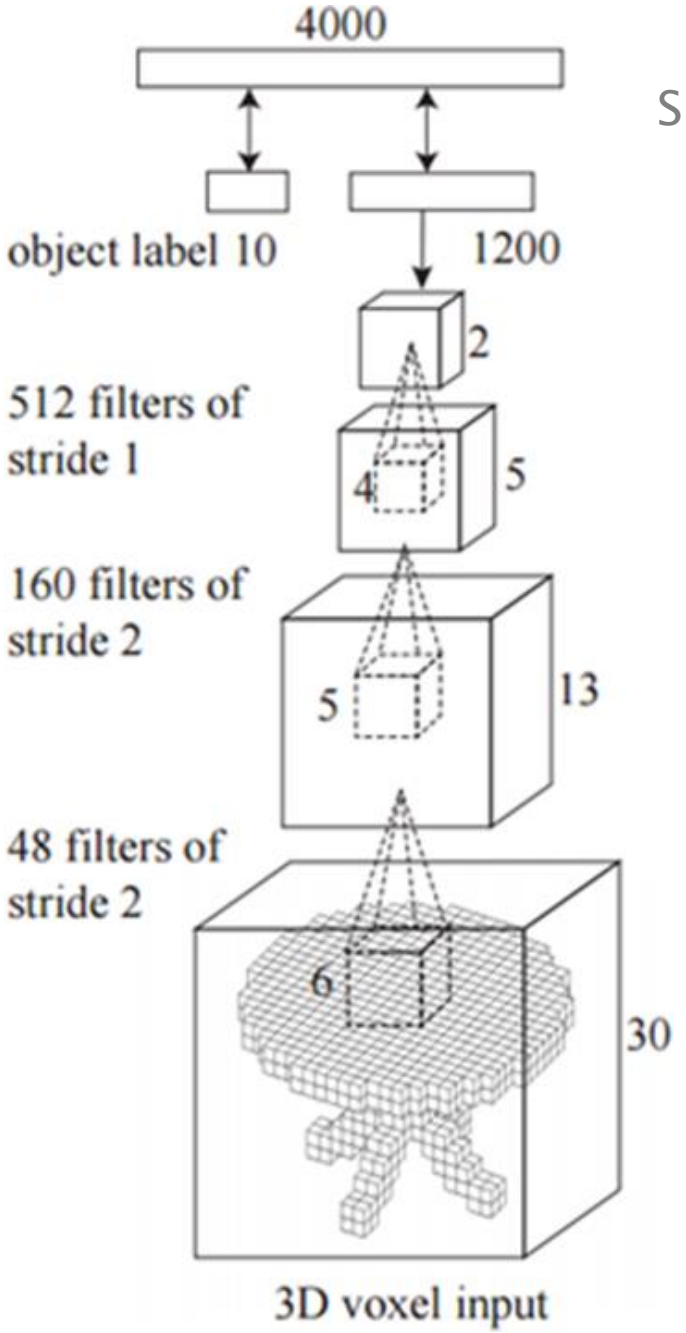
"3D ShapeNets: A Deep Representation for Volumetric Shapes", Wu et al., 2015

3D Shape-Net:

Volumetric representation (binary voxels on 3D grid)

3D convolutional network





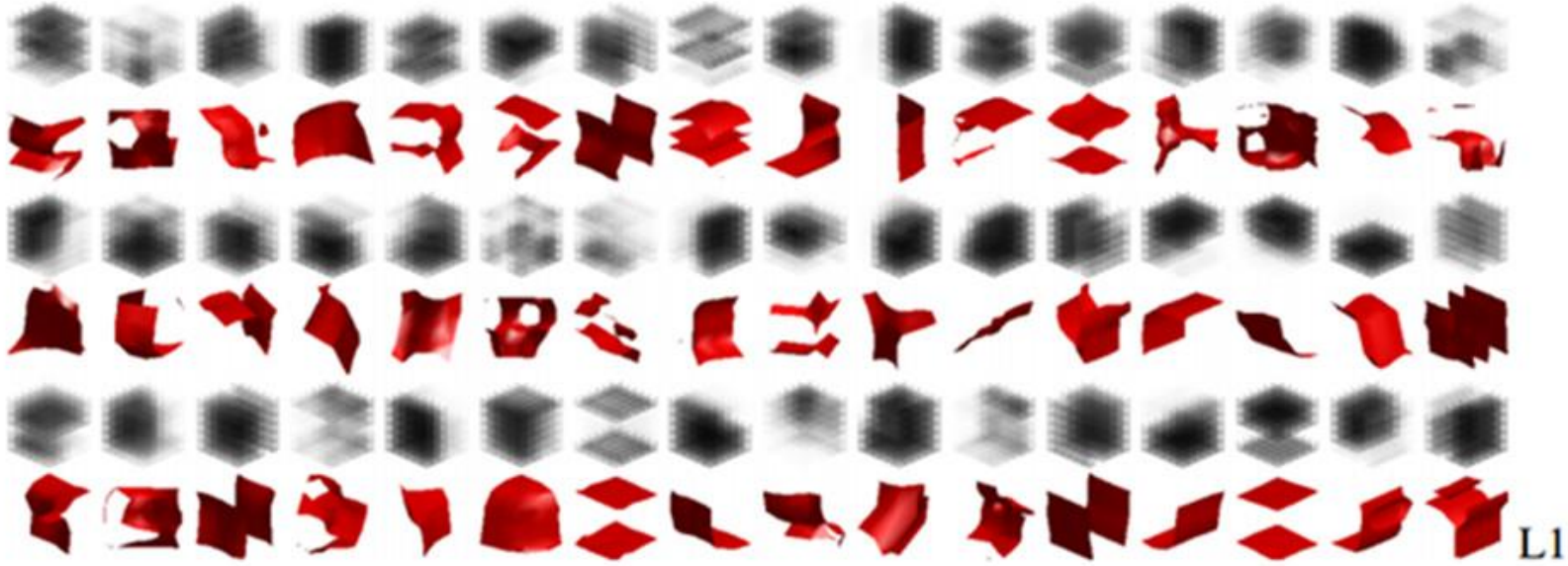
Slide credit E. *Rodolà*

"3D ShapeNets: A Deep Representation for Volumetric Shapes", Wu et al., 2015

3D Shape-Net learned features:

Slide credit
E. *Rodolà*

3D primitives

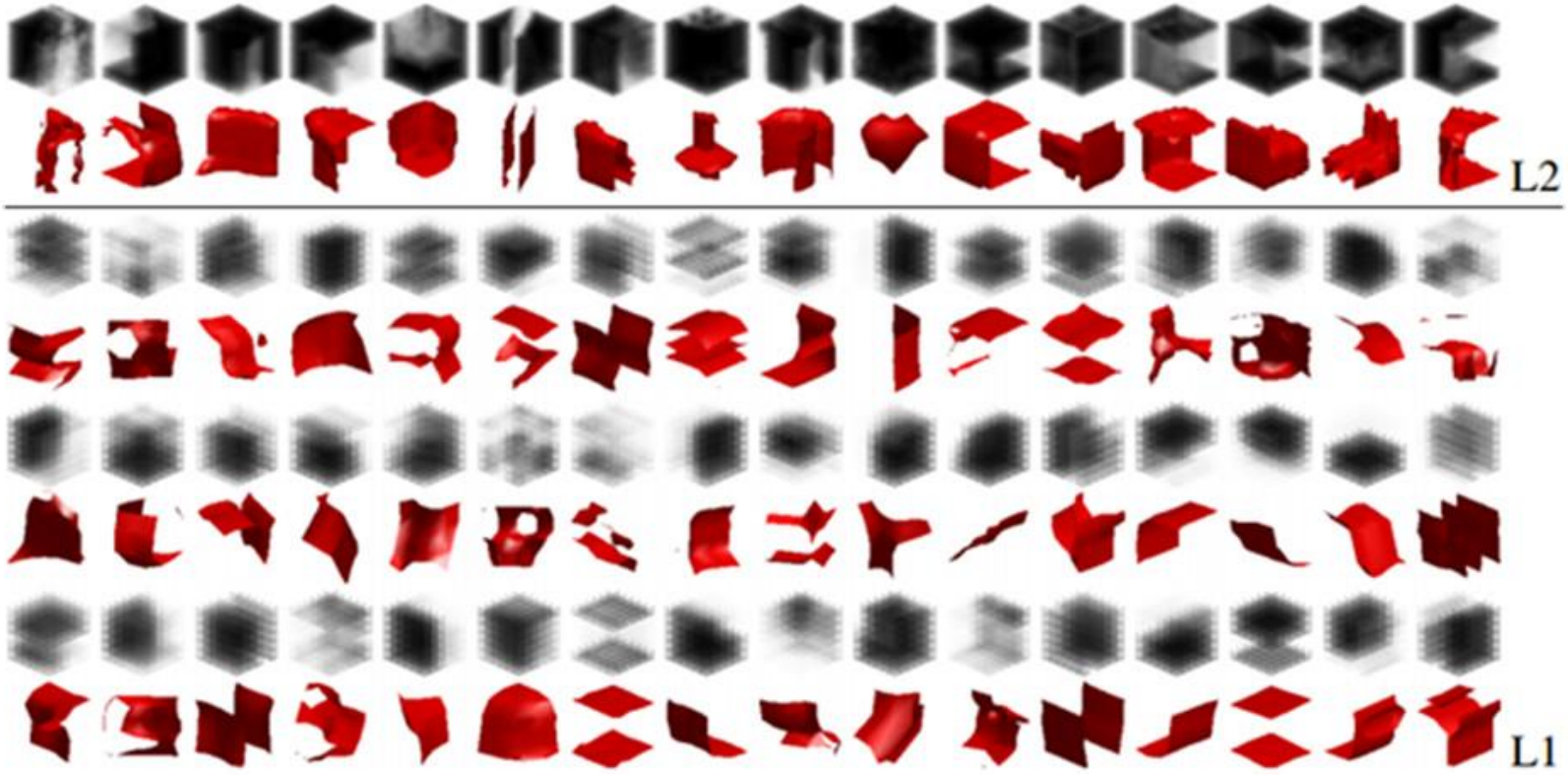


"3D ShapeNets: A Deep Representation for Volumetric Shapes", Wu et al., 2015

3D Shape-Net learned features:

Slide credit
E. *Rodolà*

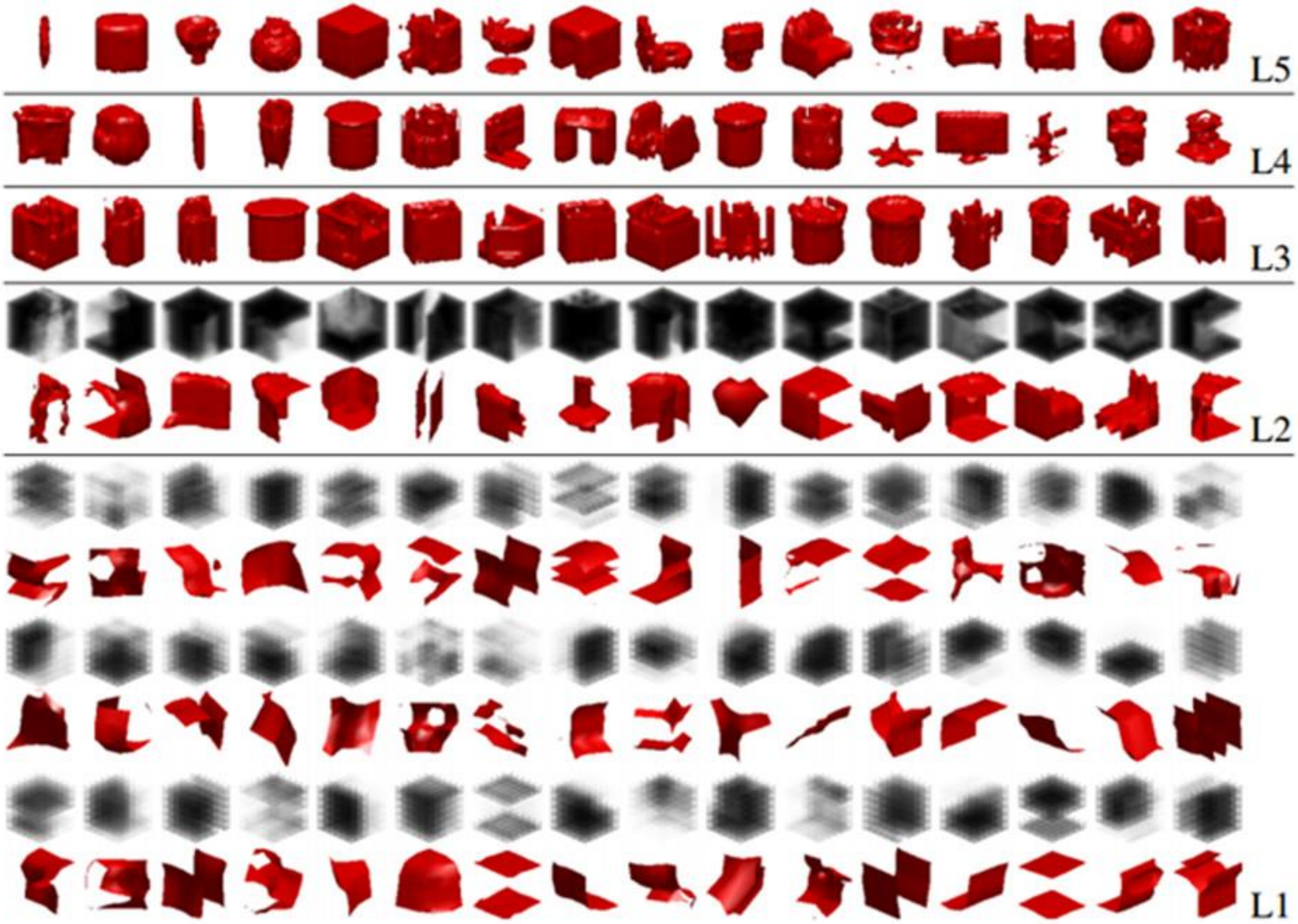
Different scales of features



"3D ShapeNets: A Deep Representation for Volumetric Shapes", Wu et al., 2015

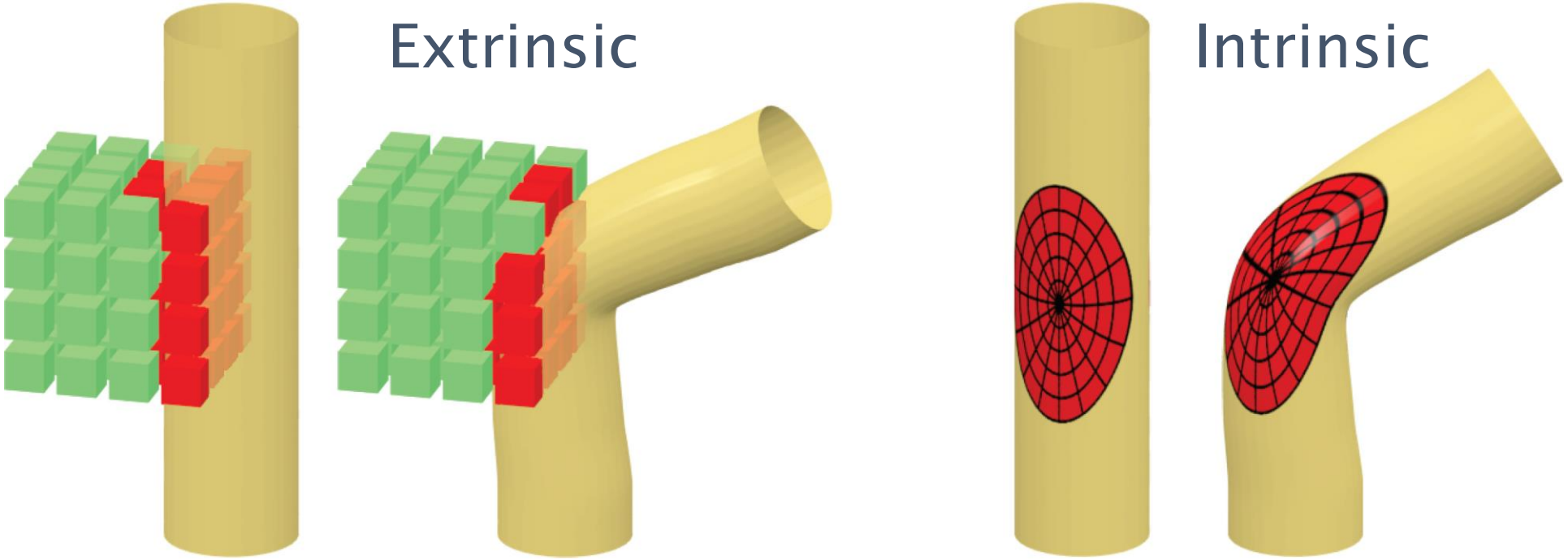
3D Shape-Net learned features:

Slide credit
E. *Rodolà*



"3D ShapeNets: A Deep Representation for Volumetric Shapes", Wu et al., 2015

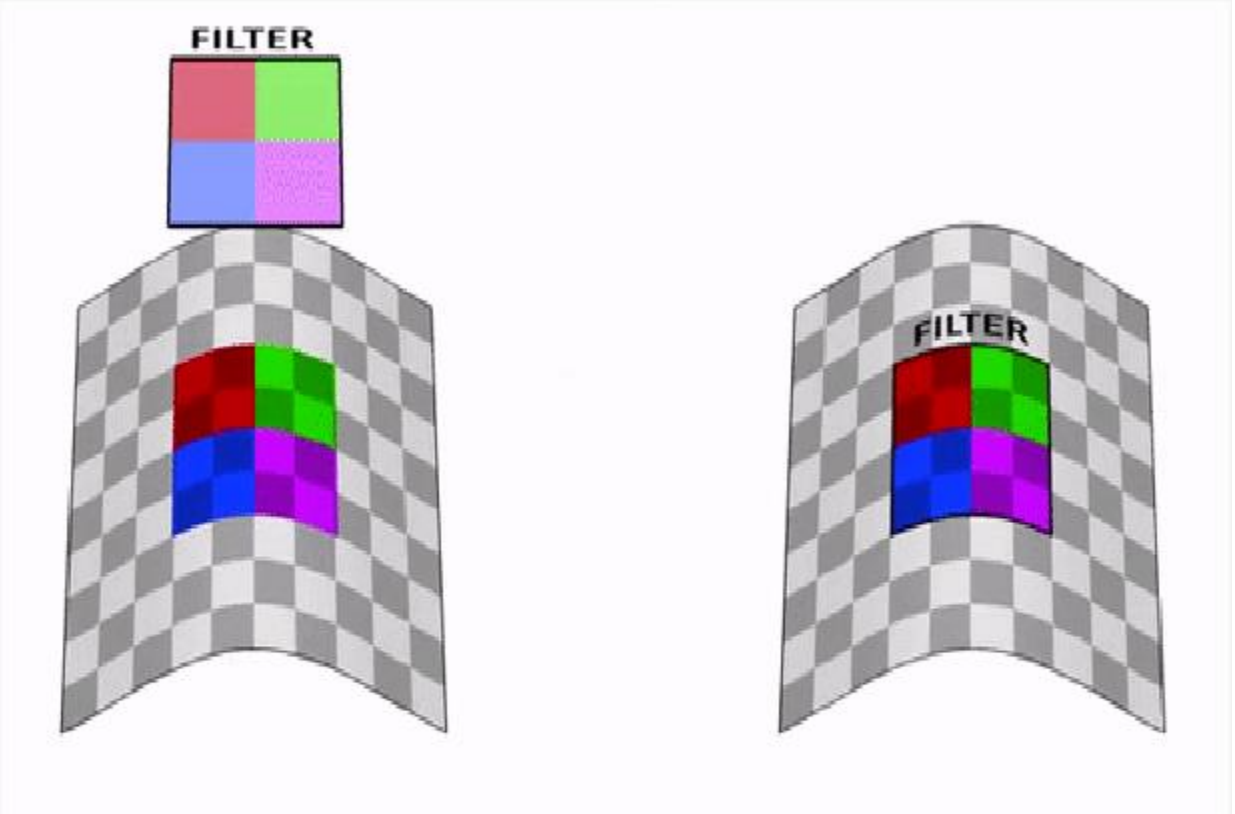
Challenges of Geometric DL:



- How to define **convolution**?
- How to do **pooling**?
- How to be **efficient**?

Slide credit E. *Rodolà*

Intrinsic vs extrinsic:

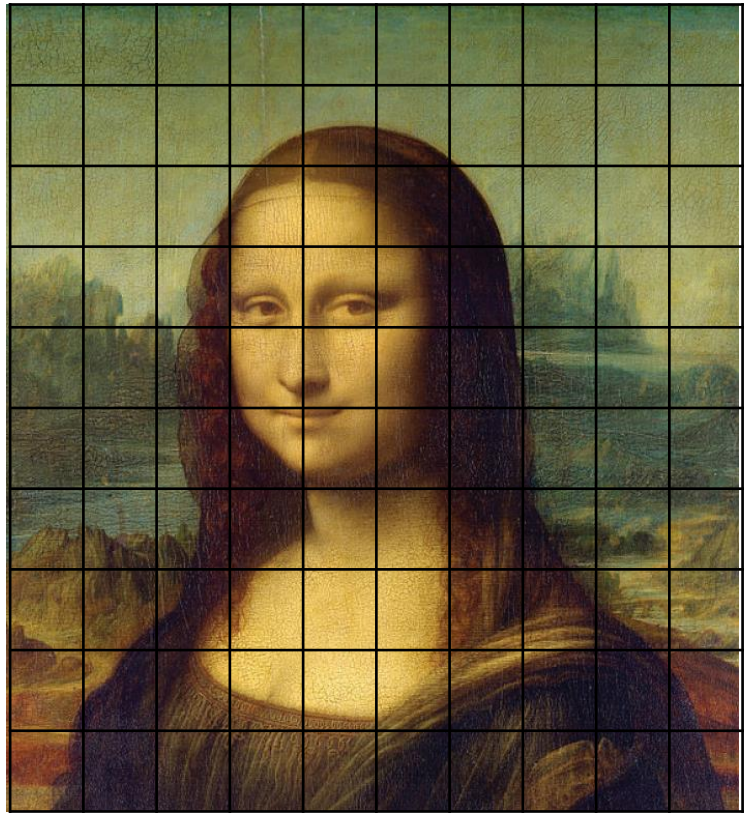


Extrinsic

Intrinsic

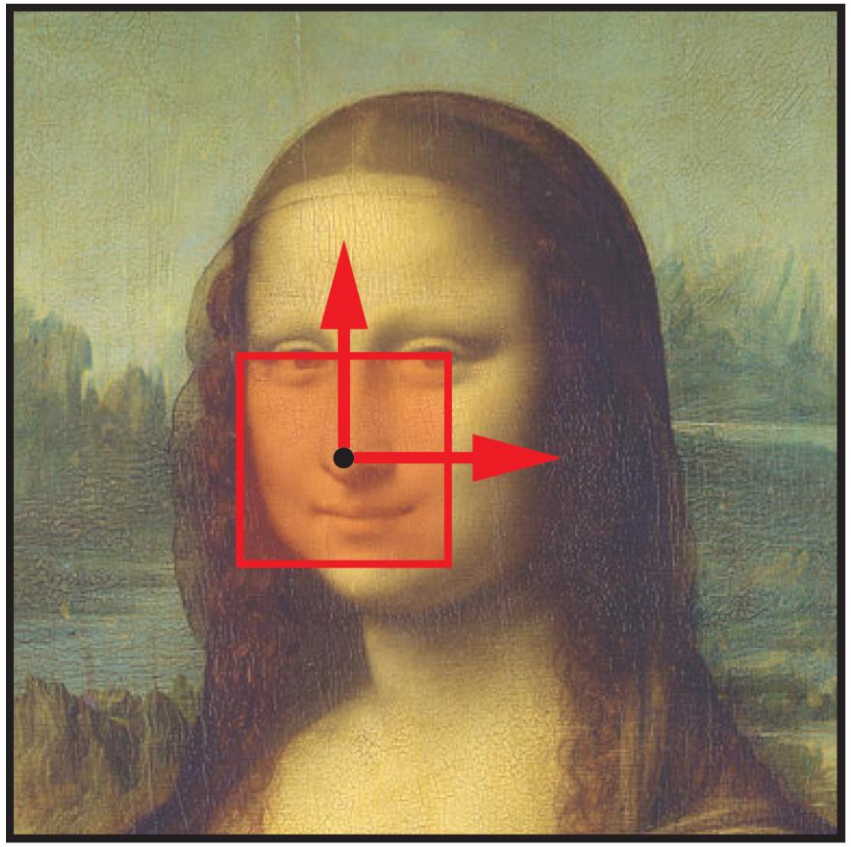
Slide credit E. *Rodolà*

Fixed pattern vs connectivity:

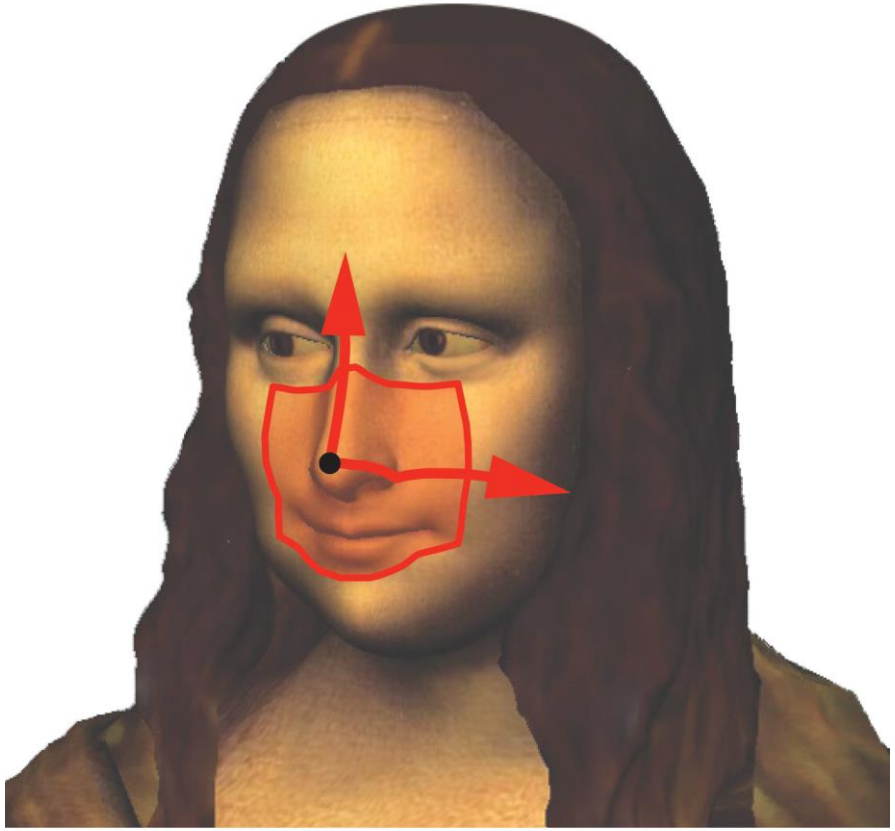


Slide credit E. *Rodolà*

Non-Euclidean convolution:



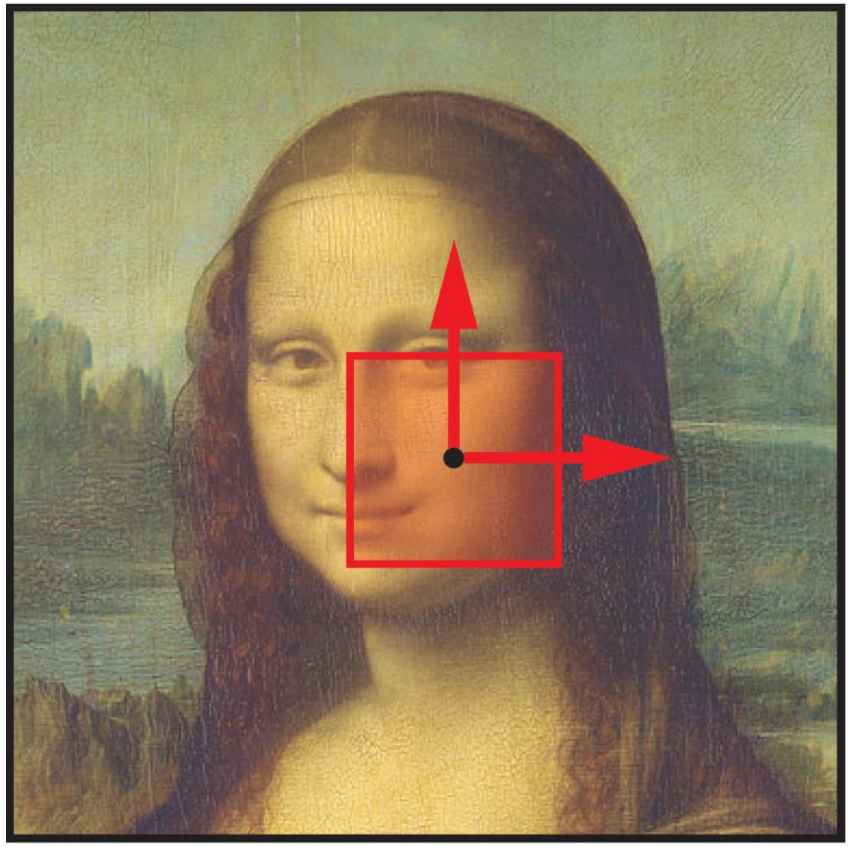
Euclidean



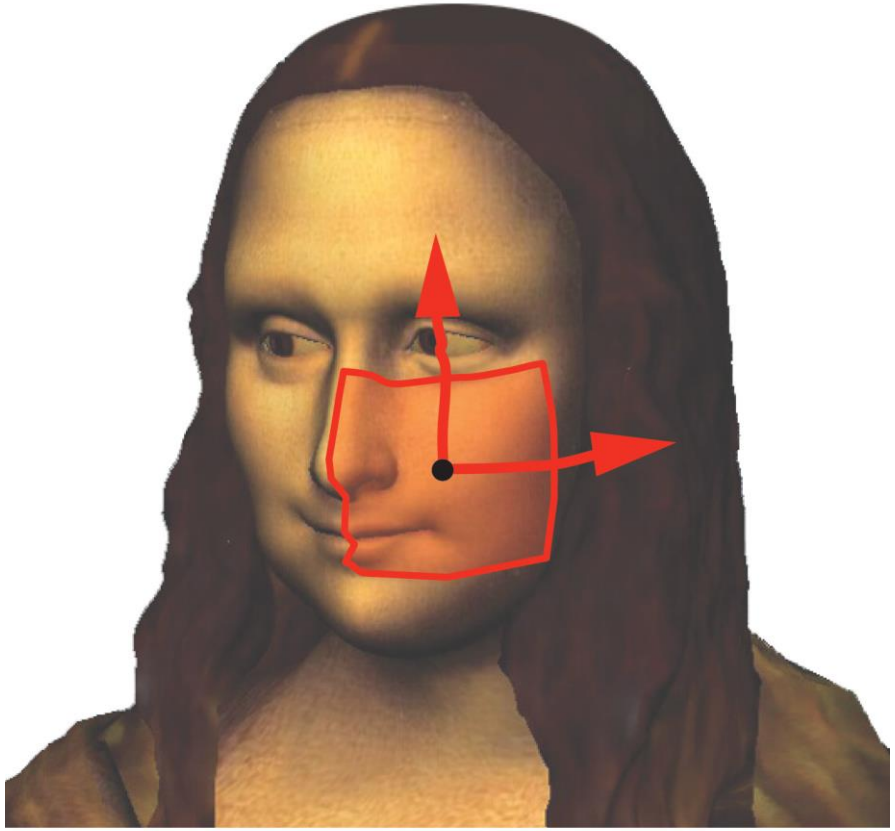
Non- Euclidean

Slide credit E. *Rodolà*

Non-Euclidean convolution:



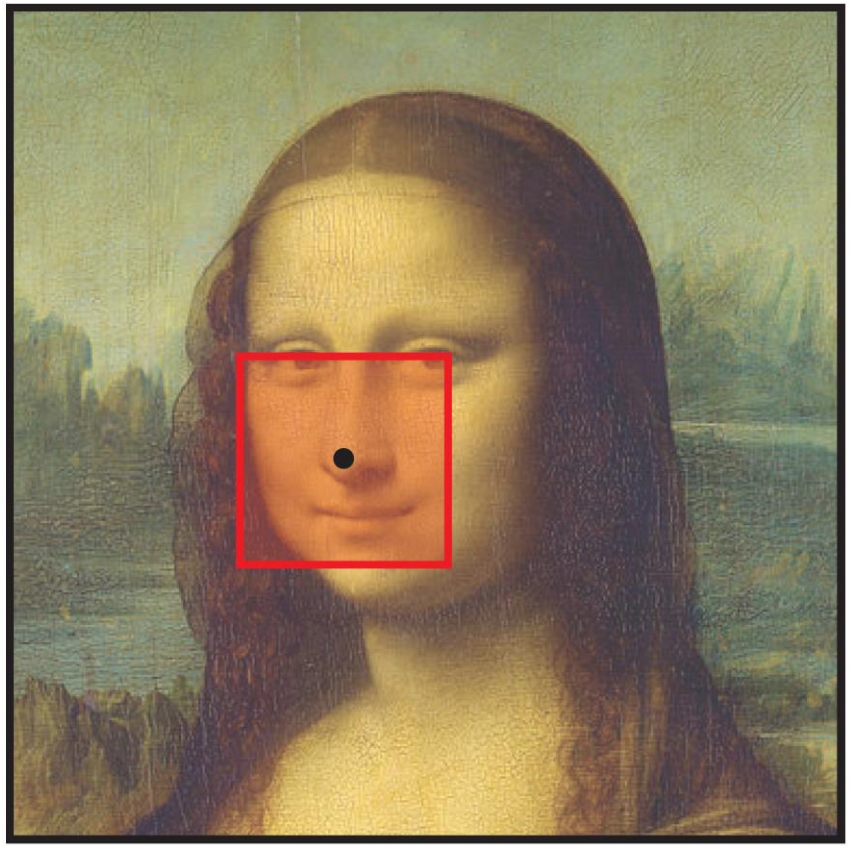
Euclidean



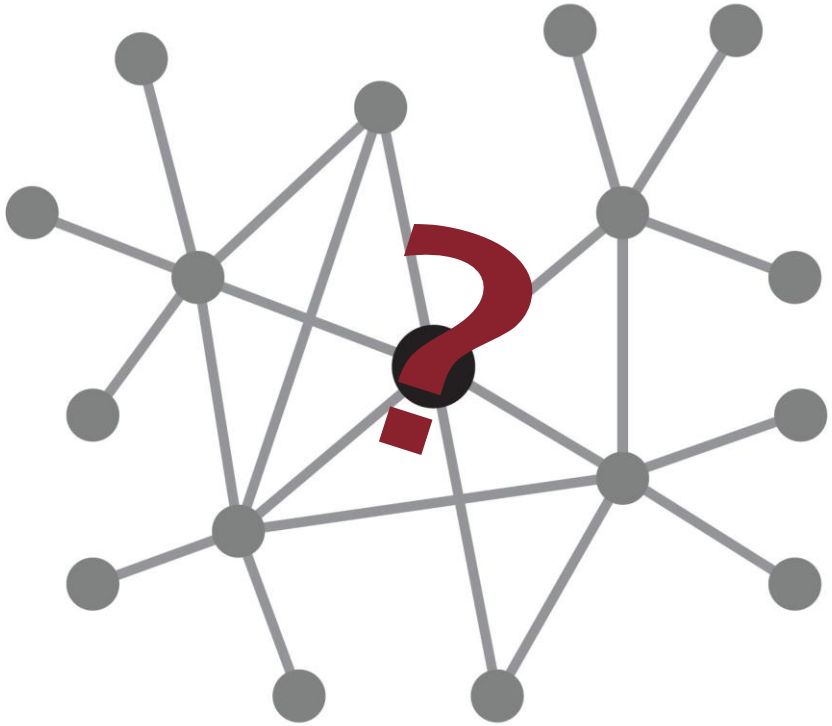
Non- Euclidean

Slide credit E. *Rodolà*

Non-Euclidean convolution:



Image

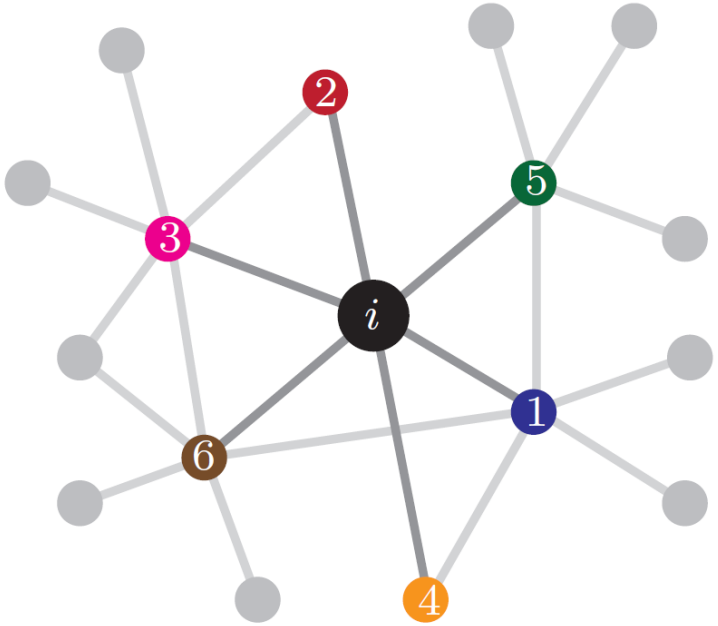


Graph

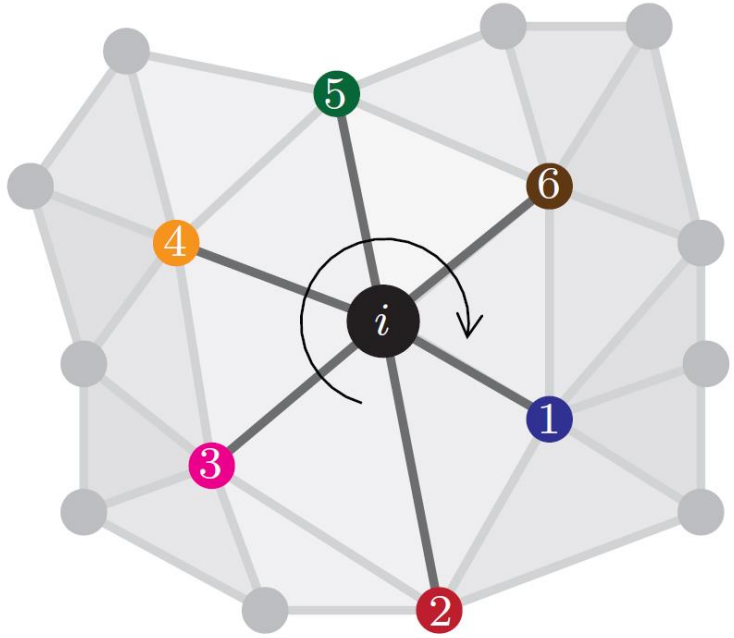
Slide credit E. *Rodolà*

Local ambiguity

Unlike images, there is not a **canonical ordering** of the points in the domain



Graph rotation

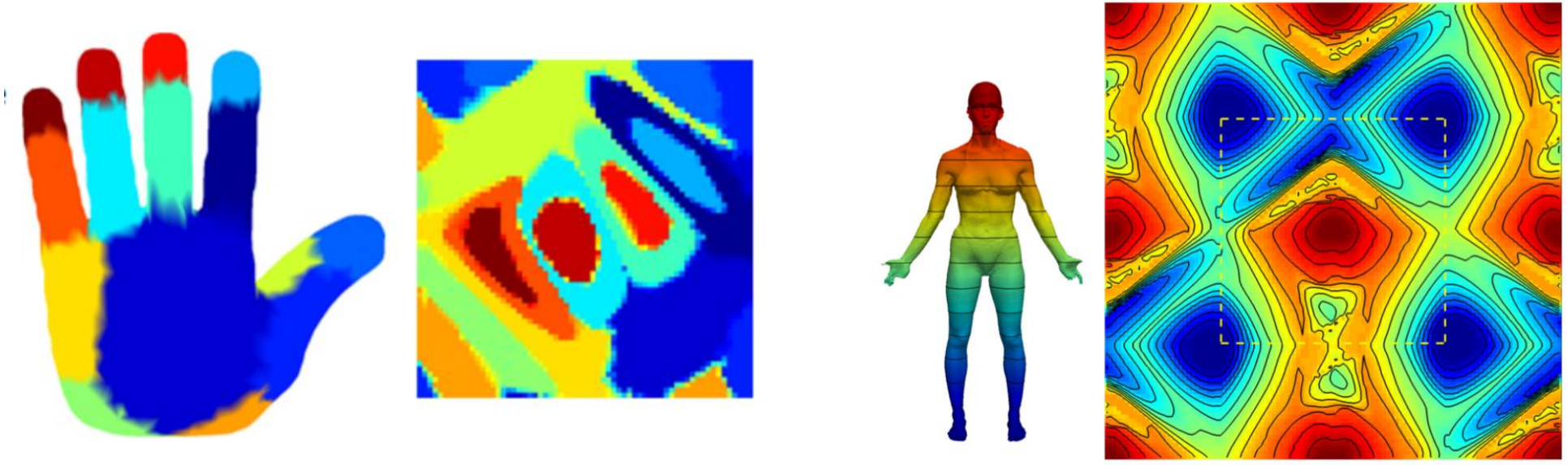


Mesh rotation

Slide credit E. *Rodolà*

Global parametrization: Slide credit E. *Rodolà*

We can map a geometric object in 2D and:

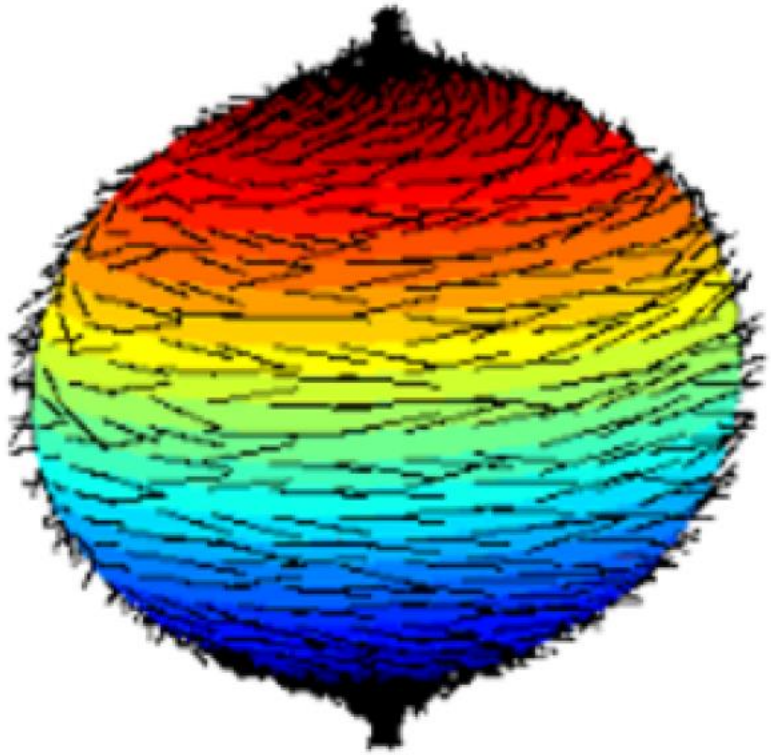


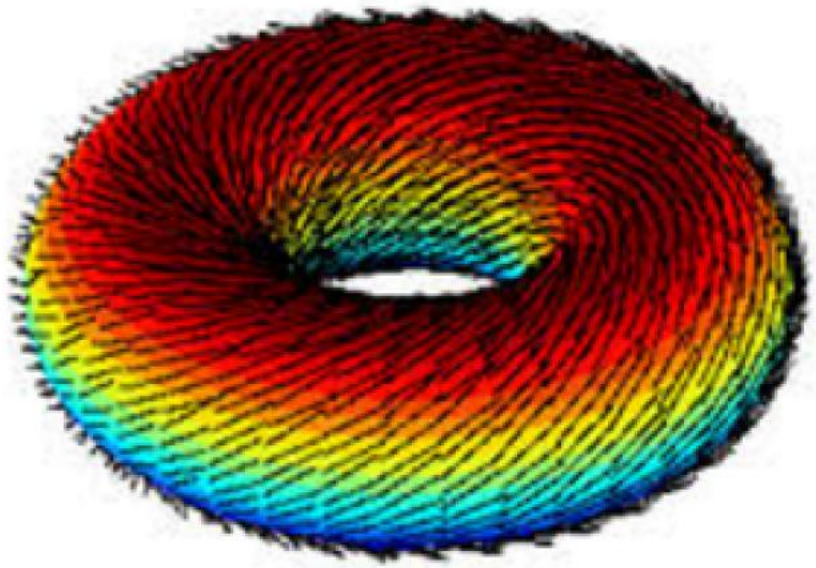
- Apply Euclidean techniques in the embedding space
- Provide invariance to certain tranformations
- Not-unique parametrization
- Distortion rised by the embedding map

"Deep learning 3d shape surfaces using geometry images", Sinha et al., 2016

Directly on manifolds:

Translation invariant convolution is not possible in general on surfaces due to the singularities in translation fields (proved by the «Hairy ball» theorem from Poincaré-Hopf)



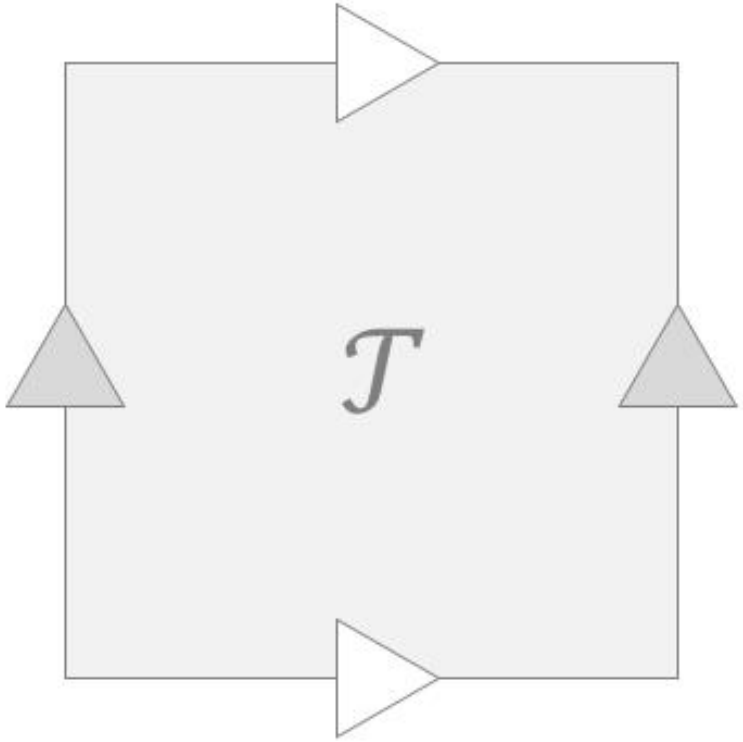


Slide credit E. *Rodolà*

Directly on manifolds:

Slide credit E. *Rodolà*

The torus is the only closed and orientable surface admitting a translational group



"Convolutional Neural Networks on Surfaces via Seamless Toric Covers", Maron et al., 2017

Spatial convolution:

Slide credit E. *Rodolà*

Local system of coordinates
(geodesic polar around i)

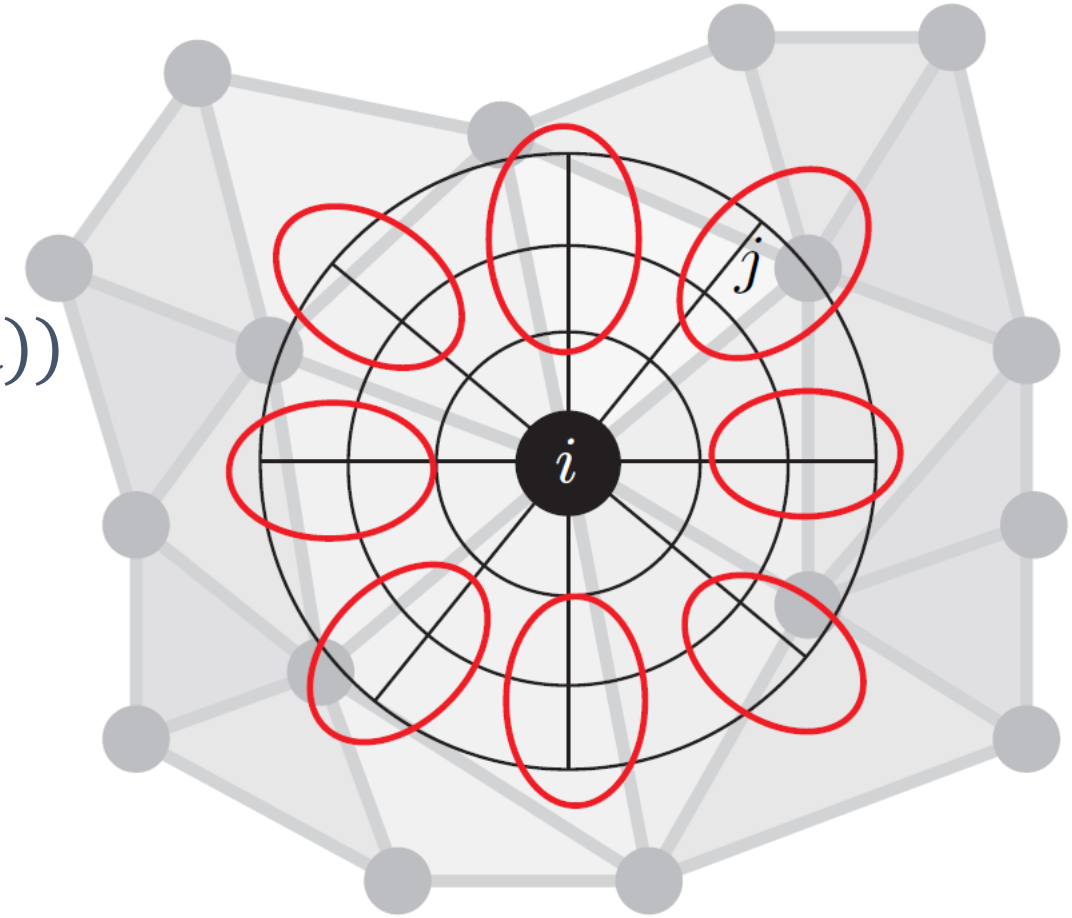
Local Weights $w(u_{ij})$ (Gaussians)

$$w(u_{ij}) = \exp(-(u_{ij} - \mu)^{\top}\Sigma^{-1}(u_{ij} - \mu))$$

Input signal: $\mathbf{f} \in \mathbb{R}^n$

Learnable filter: $\mathbf{g} \in \mathbb{R}^n$ (local)

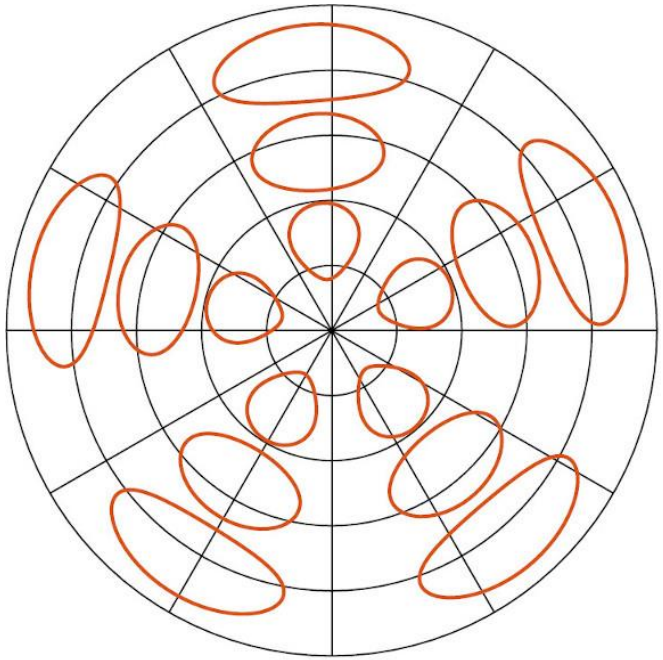
Apply the filter: $\boldsymbol{f}^{\top}\mathbf{g} \in \mathbb{R}^n$ (local)



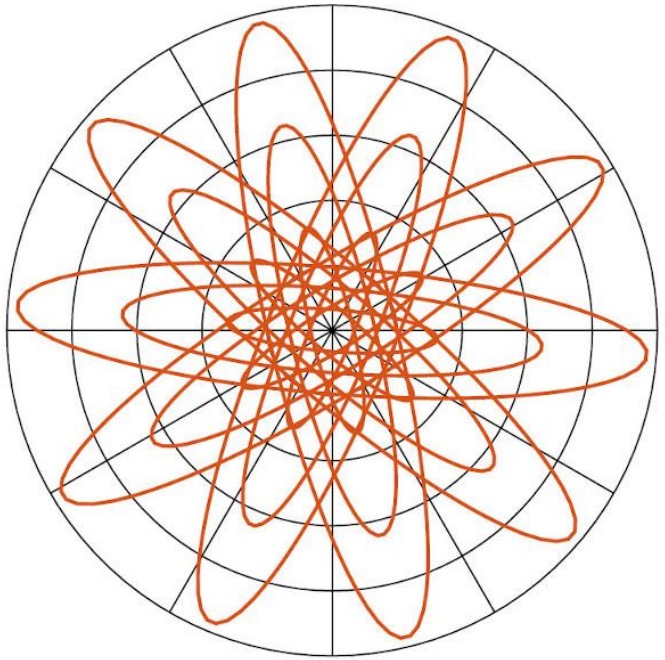
"Geometric deep learning on graphs and manifolds using mixture model CNNs", Monti et al., 2016

Spatial convolution:

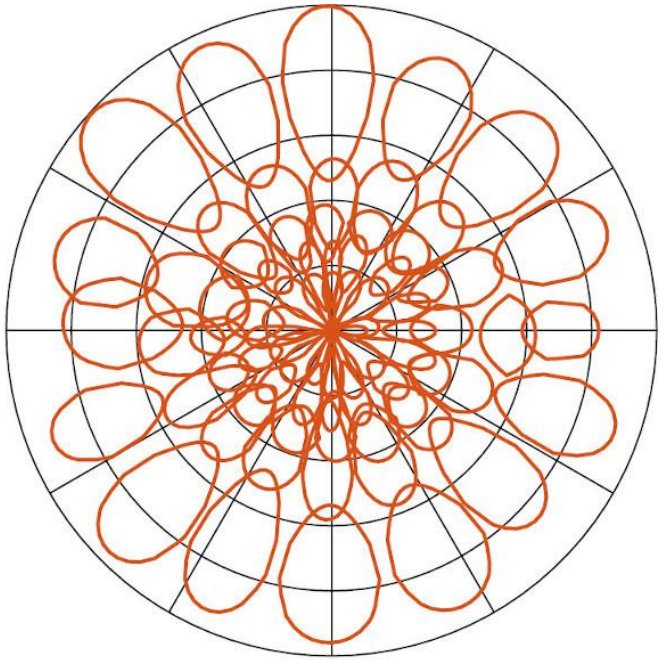
Slide credit E. *Rodolà*



GCNN



ACNN

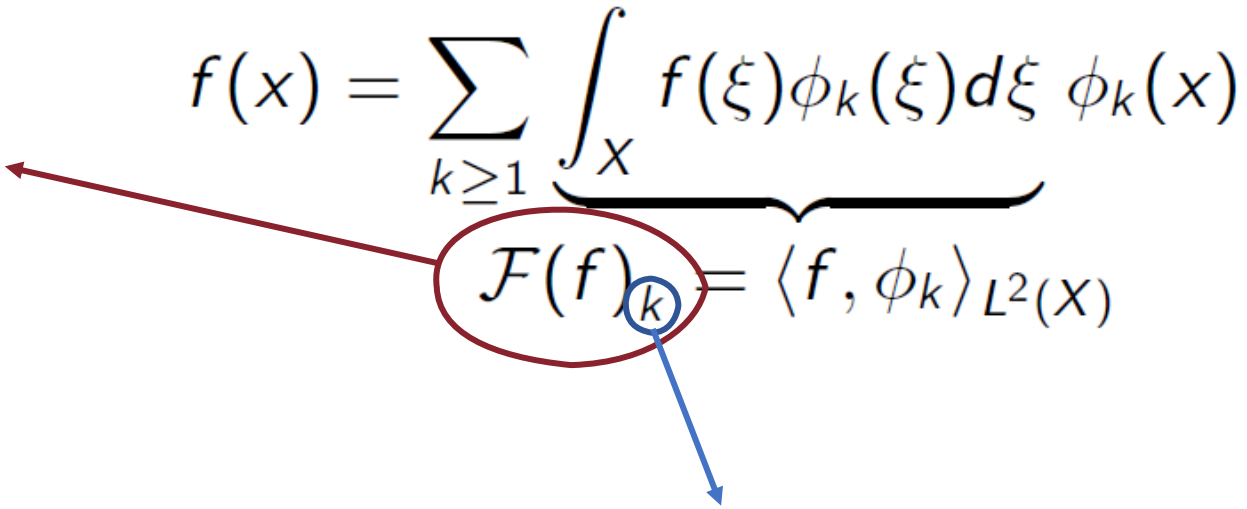


MoNet

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

Fourier analysis

The Fourier coefficients depend on the Global geometry of the surface



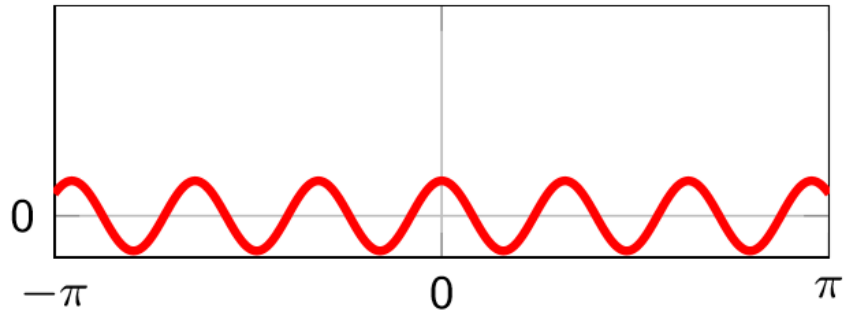
The only characterization of these coefficients is the frequency that each of them is representing

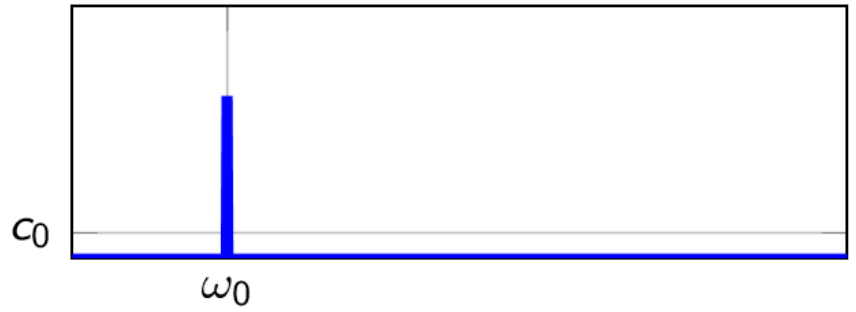
We want to compute pointwise descriptors

⇓

We would like to enforce LOCALIZATION of the Fourier analysis

Fourier and the need for localization

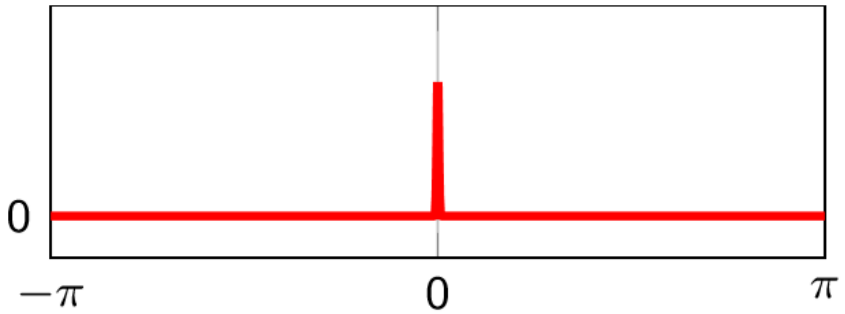


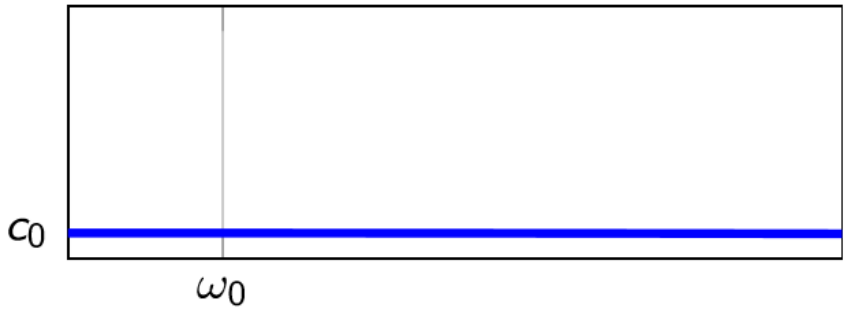


Poor spatial localization

⇕

Good spectral localization

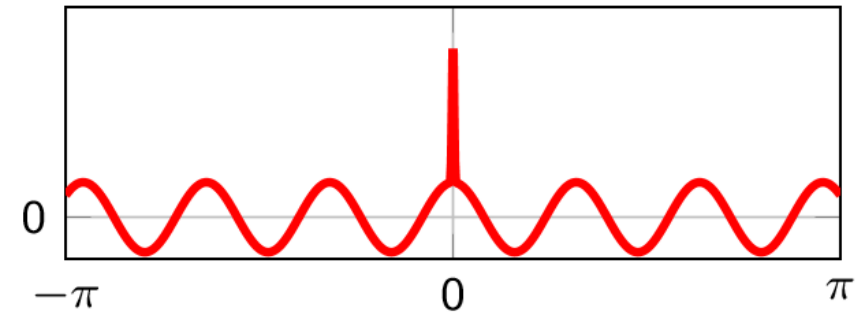


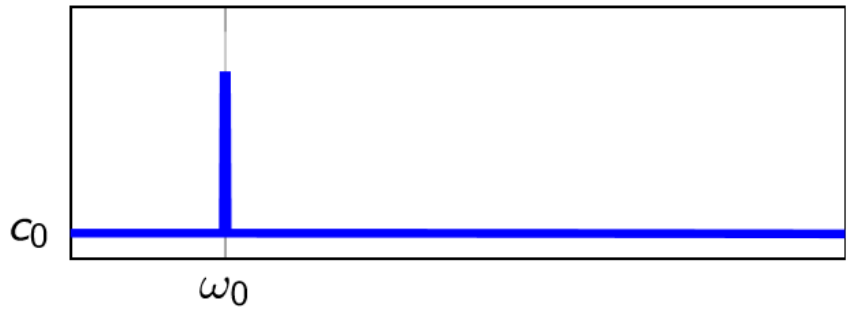


Good spatial localization

⇕

Poor spectral localization

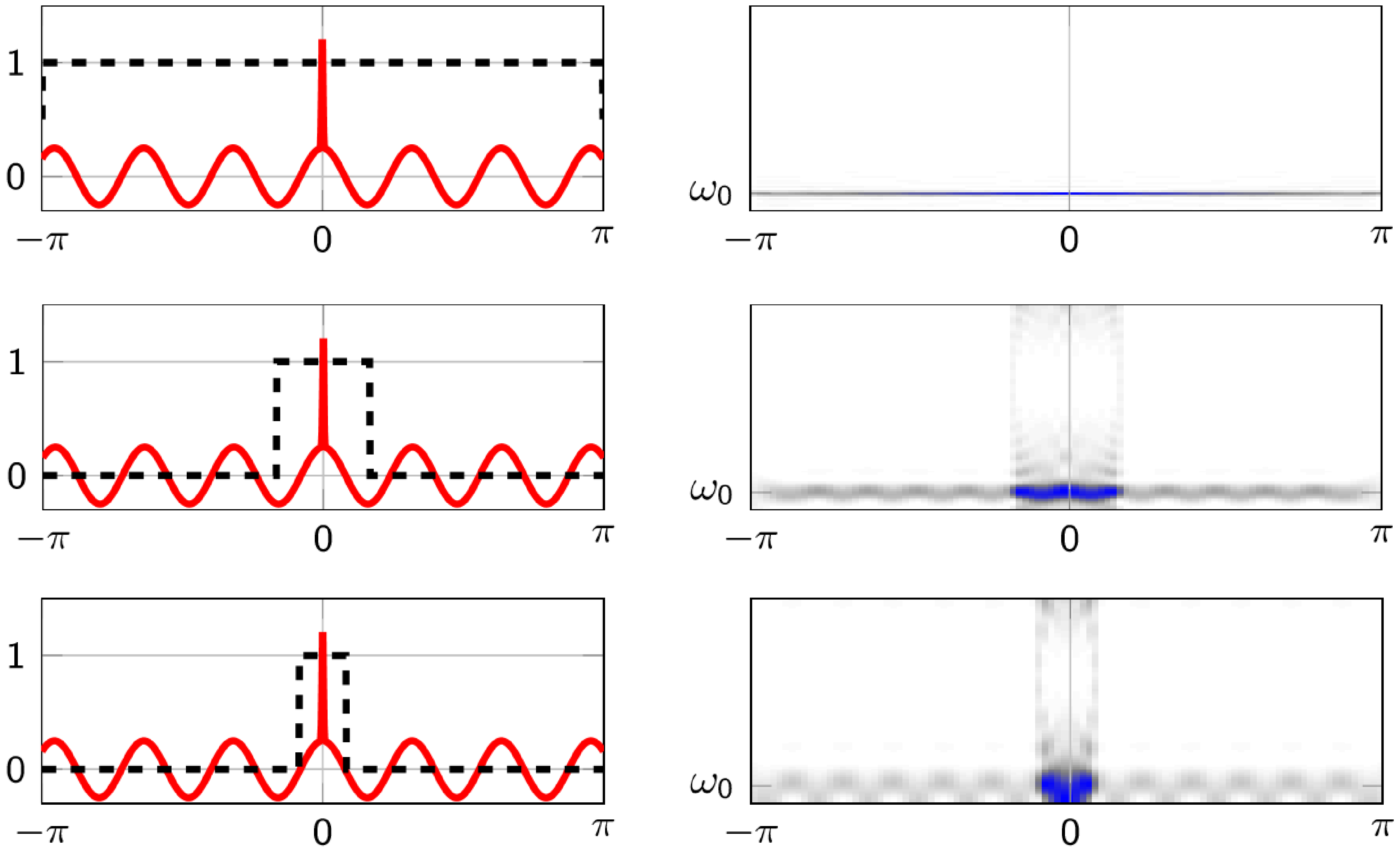




QUESTION: Do you know how this has been solved in signal processing?

"Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks", *Boscaini et al., 2015*

WFT is the standard solution



Windowed Fourier Transform of a signal enveloped by a **window** g

$$WFT(f)(\xi,\omega) = \underbrace{\int_D f(x)g(x-\xi)e^{-i\omega x}}_{\langle f(x), g_{\xi,\omega}(x)\rangle_{L^2(D)}}$$

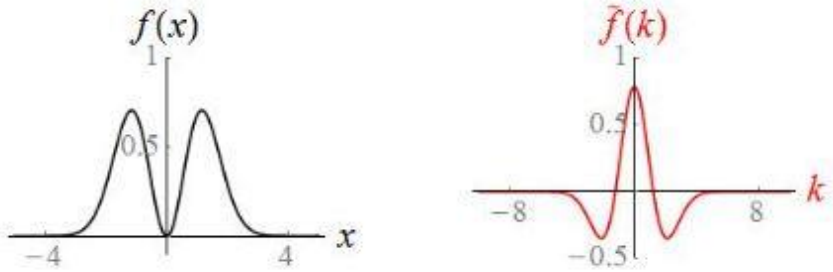
WFT

In some case the signal drastically changes in the space-time domain.

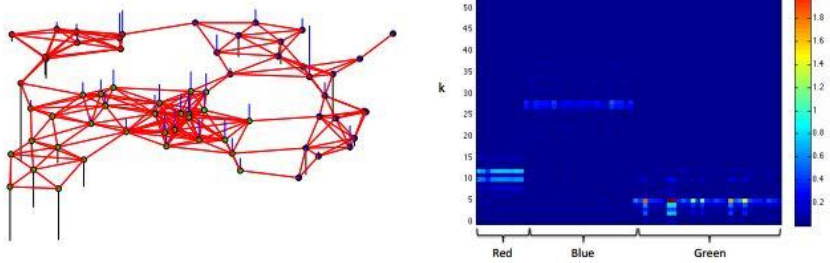
→ *It is desirable to localize signal and analyze it locally.*

The **Windowed Fourier Transform (WFT)** is the solution for this problem

Euclidean domains



→ Graphs



→ **And on manifolds ?**

Thorem of convolution

Convolution on Euclidean domain : $[-\pi, \pi]$ of two functions $f, g: [-\pi, \pi] \to \mathbb{R}$ is defined as:

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

Convolution Theorem: Fourier transform diagonalizes the convolution operator.

$\Longrightarrow$ ***Convolution can be computed in the spectral domain***

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

Thorem of convolution

Convolution on Euclidean domain : $[-\pi, \pi]$ of two functions $f, g: [-\pi, \pi] \to \mathbb{R}$ is defined as:

$$(\boldsymbol{f} \star \boldsymbol{g})[x_i] = \sum_{j=1}^{n} \boldsymbol{f}[x_j]\, \boldsymbol{g}[x_i - x_j]$$

$$\boldsymbol{f} \star \boldsymbol{g} = \begin{bmatrix} \boldsymbol{g_1} & \boldsymbol{g_2} & \cdots & \cdots & \boldsymbol{g_n} \\ \boldsymbol{g_n} & \boldsymbol{g_1} & \boldsymbol{g_2} & \cdots & \boldsymbol{g_{n-1}} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{g_2} & \boldsymbol{g_3} & \cdots & \cdots & \boldsymbol{g_1} \end{bmatrix} \begin{bmatrix} \boldsymbol{f_1} \\ \vdots \\ \vdots \\ \boldsymbol{f_n} \end{bmatrix} = \boldsymbol{\Phi \hat{g} \Phi^{\dagger} f}$$

Thorem of convolution

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

Thanks to the Convolution Theorem

$$\hat{g} = \begin{bmatrix} \hat{g}_1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \hat{g}_n \end{bmatrix}$$

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

WFT on non-Euclidean manifolds

Euclidean domains	Manifolds
Fourier Transform: $\hat{f}_\omega = \langle f,\ e^{2\pi i\omega x}\rangle$	Fourier Transform: $\hat{f}_k = \langle f, \phi_k\rangle_{L^2(X)}$
Convolution: $(f * g)(x) = \int_{-\infty}^{\infty} \hat{f}_\omega \hat{g}_\omega e^{2\pi i\omega x} dx$	Convolution: $(f \star g)(x) = \sum_{k\geq 1} \hat{f}_k \hat{g}_k \phi_k(x)$
Translation: $(T_u f)(x) = (f * \delta_u)(x) = f(x-u)$	Translation: $(T_{x'} f)(x) = (f \star \delta_{x'})(x) = \sum_{k\geq 1} \hat{f}_k \phi_k(x')\phi_k(x)$
Modulation: $(M_\omega f)(x) = e^{i\omega x} f(x)$	Modulation: $(M_k f)(x) = \phi_k(x) f(x)$

WFT on non-Euclidean manifolds

Euclidean domains

Basic Atom: $g_{u,\,\omega}(x) = (M_\omega T_u g)(x) = g(x-u)e^{2\pi i\omega x}$

Windowed Fourier Transform:

$$Sf(u,\omega) = \langle\, f,\; g_{u,\,\omega}\,\rangle$$

Manifolds

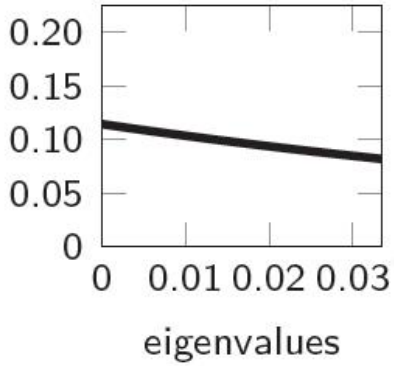
Basic Atom: $g_{x',k}(x) = (M_k T_{x'} g)(x) = \phi_k(x)\sum_{l\geq 1}\hat{g}_l\phi_l(x')\phi_l(x)$

Windowed Fourier Transform:

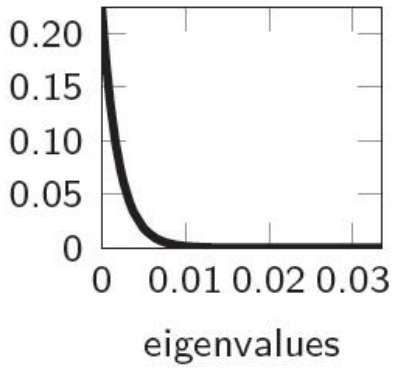
$$(Sf)_{x,k} = \langle f, g_{x,k}\rangle_{L^2(X)}$$

WFT atoms

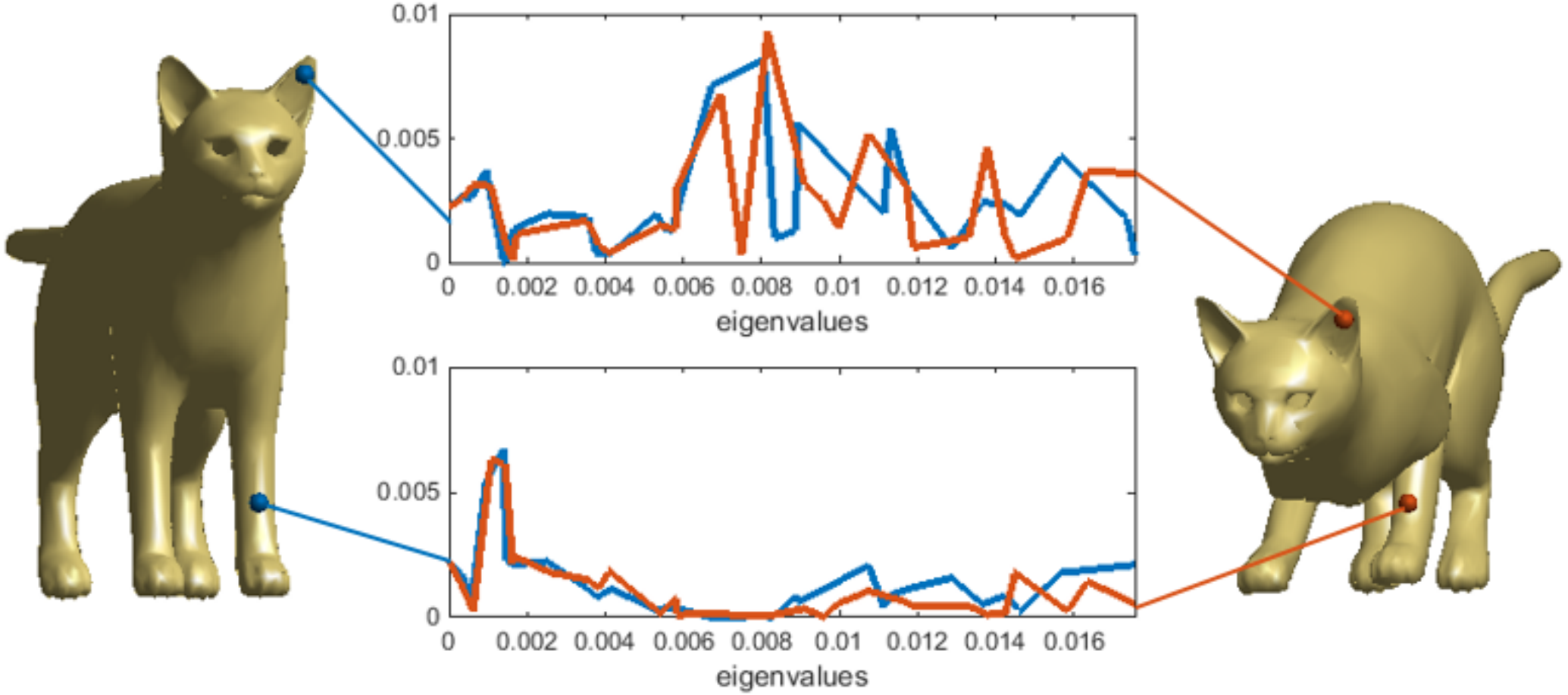
$\hat{g}_1$



$\hat{g}_2$



WFT problems



Spectral descriptors

A common structure is shared by the spectral descriptors
HKS and **WKS**

$$desc_q(x) = \sum_{l=1}^{k} g_{t_q}(\lambda_l)\phi_l^2(x), \quad \forall q \in 1, \ldots, Q$$

A set of filters on the frequencies

=

functions of the eigenvalues

The square of each dimension of the spectral embedding

A signal processing point of view

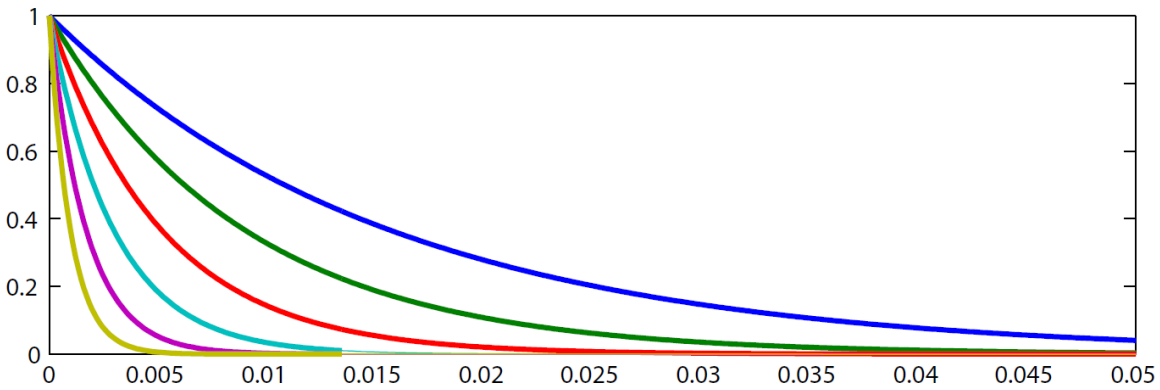
$$desc_q(x) = \sum_{l=1}^{k} g_{t_q}(\lambda_l)\phi_l^2(x), \quad \forall q \in 1, \ldots, Q$$

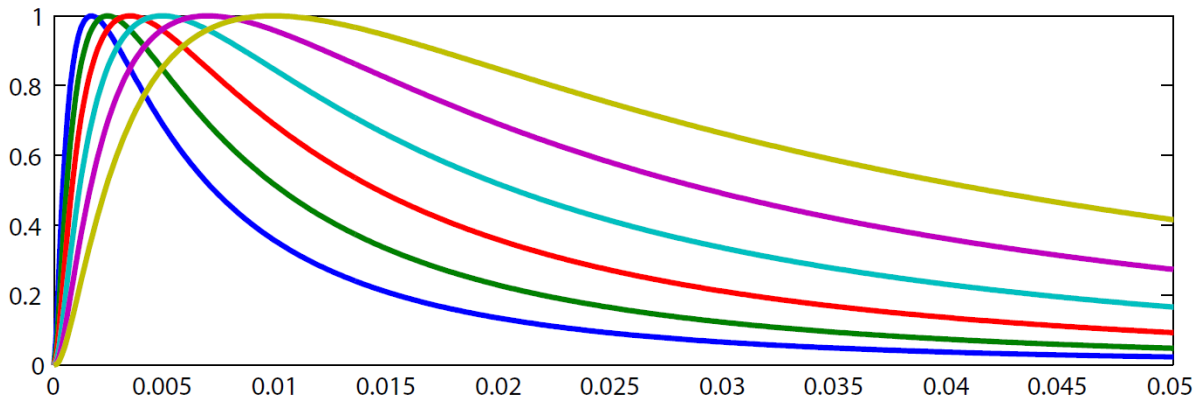
HKS:

$$g_t(\lambda_l) = e^{-t\lambda_l}$$

WKS:

$$g_t(\lambda_l) = e^{-\frac{(log(E) - log(\lambda_l))^2}{2\sigma^2}}$$





"Learning spectral descriptors for deformable shape correspondence", *Litman et al.*, 2014.

Could we obtain a stronger descriptor

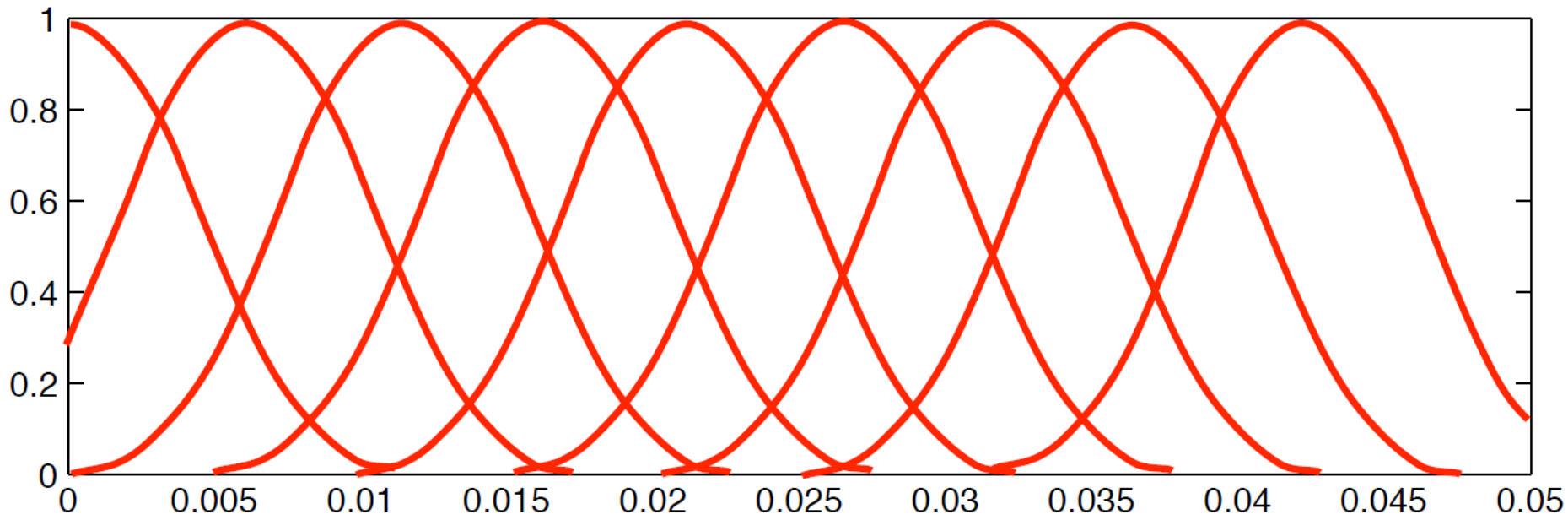
$$desc_q(x) = \sum_{l=1}^{k} g_{t_q}(\lambda_l)\phi_l^2(x), \quad \forall q \in 1, \ldots, Q$$

What are the best filters to apply in this equation to obtain the best descriptors?

Can we learn them?

Learn filters for spectral descriptors

Given a set of basis functions: $\beta_1(\lambda), \beta_2(\lambda), \ldots, \beta_Z(\lambda)$



We can learn the best coefficients to linearly combine them to obtain the best filters.

What do we need?

The q-th filter is obtained as a linear combination of the basis functions $\{\beta_z(\lambda_l)\}_{z=1}^{Z}$

$$g_q(\lambda_l) = \left(\sum_{z=1}^{Z} a_z^q \beta_z(\lambda_l) \right)$$

$$desc_q(x) = \sum_{l=1}^{k} g_{t_q}(\lambda_l)\phi_l^2(x), \quad \forall q \in 1, \ldots, Q$$

$$desc_q(x) = \sum_{l=1}^{k} \left(\sum_{z=1}^{Z} a_z^q \beta_z(\lambda_l) \right) \phi_l^2(x)$$

What do we need?

$$desc_q(x) = \sum_{l=1}^{k} \left(\sum_{z=1}^{Z} a_z^q \beta_z(\lambda_l) \right) \phi_l^2(x)$$

we should learn the set of coefficients:

$$a_z^q \; \forall q = 1, \ldots, Q \text{ and } z = 1, \ldots, Z$$

that is equivalent to learn a matrix:

$$\boldsymbol{A} \in \mathbb{R}^{Q \times Z} \text{ s.t. } \boldsymbol{A}_{q,z} = a_z^q$$

Learned descriptors

We can compute a learned kernel signature by learning the matrix $\boldsymbol{A} \in \mathbb{R}^{Q \times Z}$

$$\boldsymbol{LKS}(x) = [desc_1^{\boldsymbol{A}}(x), desc_2^{\boldsymbol{A}}(x), \ldots, desc_q^{\boldsymbol{A}}(x)]$$

These explixitly depend on the learned matrix

How could we learn this matrix A?

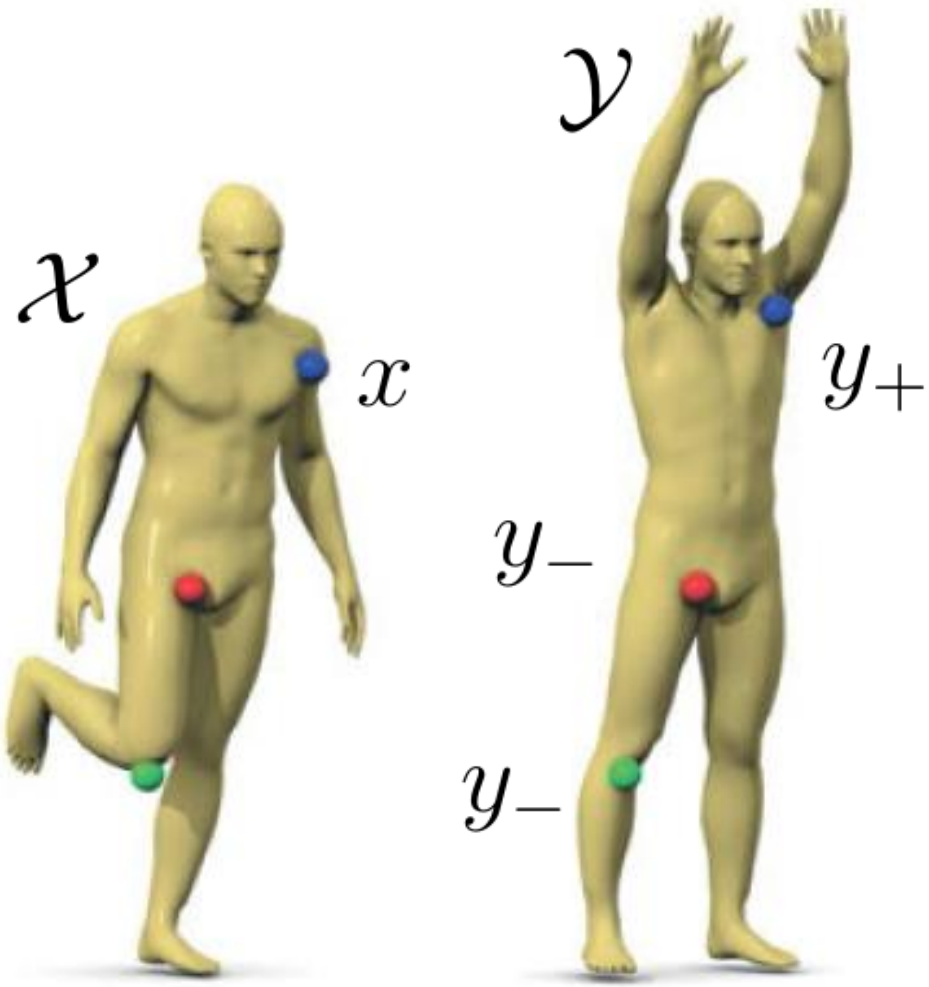
Loss definition

Given a pair of shapes $\mathcal{X}$ and $\mathcal{Y}$

We consider a set of points X on $\mathcal{X}$

such that $\forall x \in X$ we can define a set of points Y on $\mathcal{Y}$ that is composed by:

- similar points (**positive**) y_+
- dissimilar points (**negative**) y_-

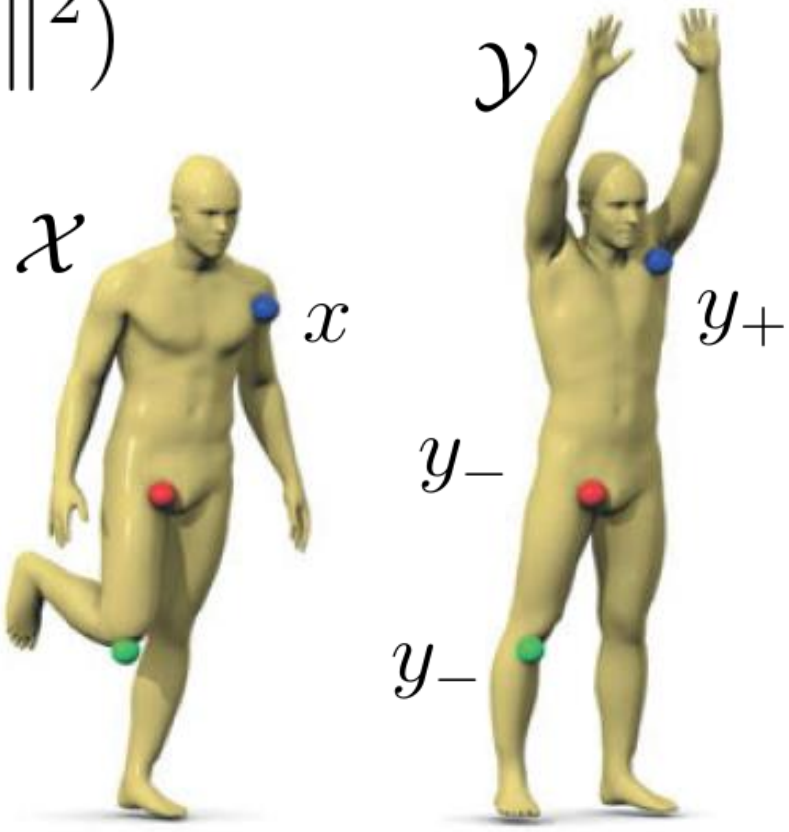


Loss definition

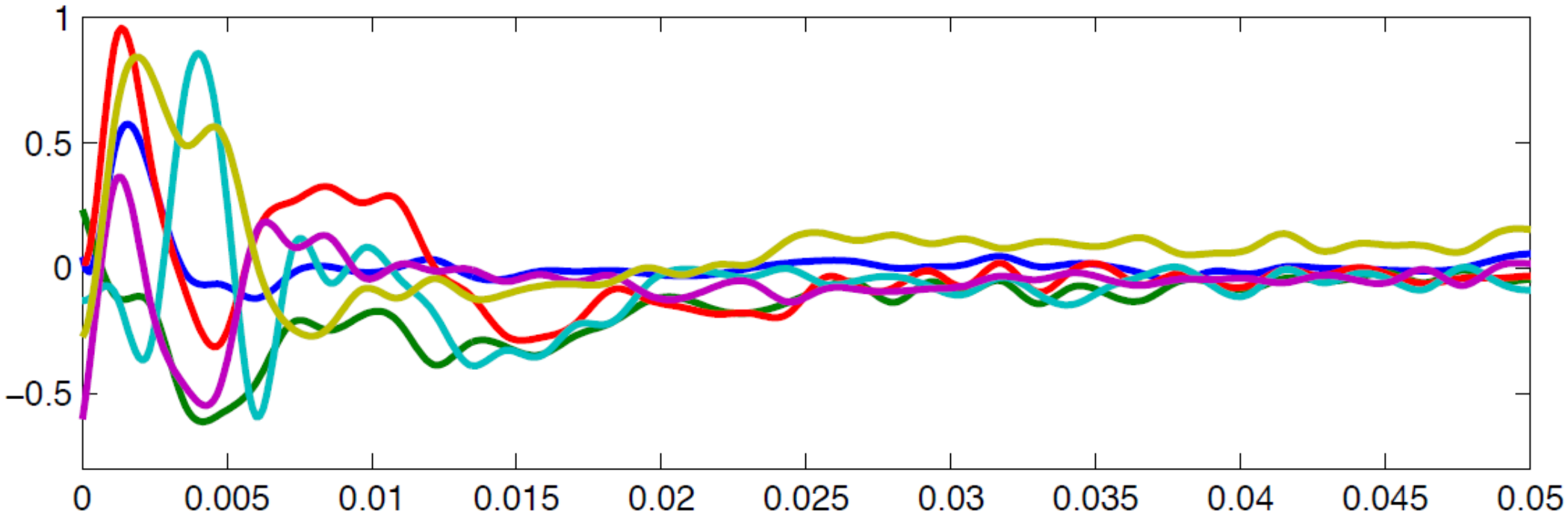
$$\boldsymbol{LKS}(x) = [desc_1^{\boldsymbol{A}}(x), desc_2^{\boldsymbol{A}}(x), \ldots, desc_q^{\boldsymbol{A}}(x)]$$

$$\underset{\boldsymbol{A}}{argmin} \sum_{x \in X} \gamma(\|LKS(x) - LKS(y_+)\|^2)$$

$$-(1-\gamma)(\|LKS(x) - LKS(y_-)\|^2)$$



Learned filters



Learn a new spectral descriptor

Given a set of $P \in \mathbb{N}$ functions $f_1, \ldots, f_P : \mathcal{X} \longrightarrow \mathbb{R}$

$$LSCNN_q(x) = \mathfrak{F}(f_1, \ldots, f_P), \ \forall q = 1, \ldots, Q$$

$$LSCNN_q(x) = \sum_{p=1}^{P} \left(\sum_{k=1}^{K} a_{q,p,k} (Sf_p)_{x,k} \right)$$

The Windowed Fourier atoms for the function f with translation in x and modulation at frequency k

QUESTION: what is not defined?

Learn the window for each input function

It is easier to learn it in the spectral domain!

We already did something similar, do you remeber where?

In the definition of the optimal shape descriptor!

The windows are obtained as a linear combination of the basis functions $\{\beta_z(\lambda_l)\}_{z=1}^{Z}$

$$g_p(\lambda_k) = \sum_{z=1}^{Z} b_z^p \beta_z \lambda_k$$

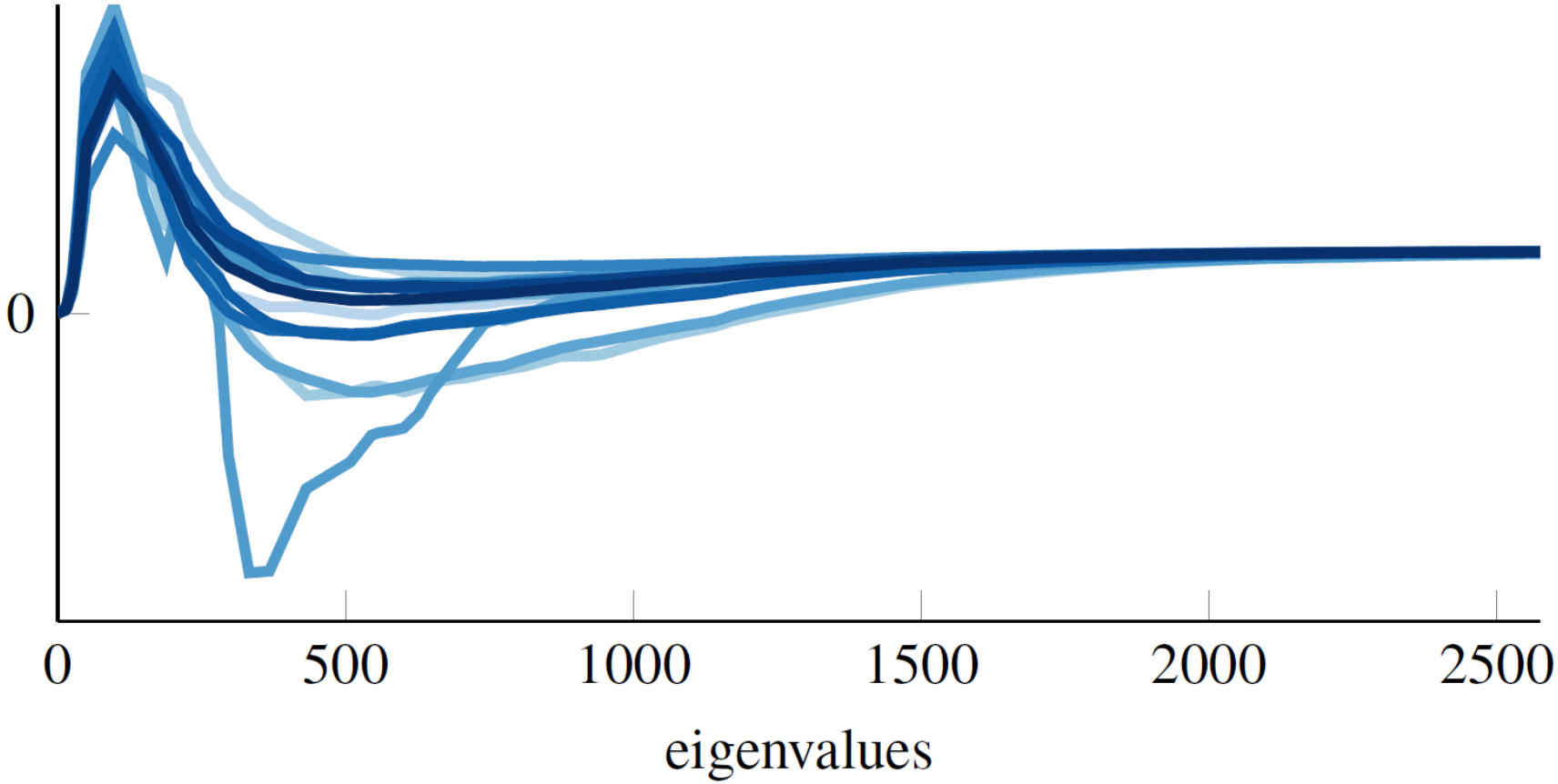
Localized spectral CNN descriptor (LSCNN)

$$desc(x) = [LSCNN_1(x), LSCNN_2(x), \ldots, LSCNN_Q(x)]$$

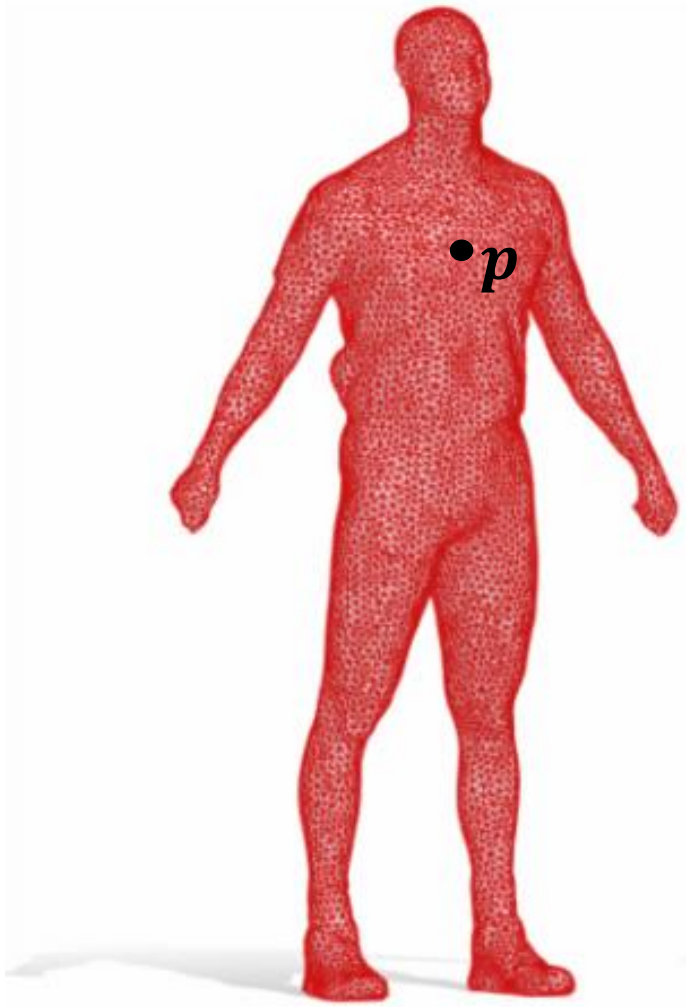
$$LSCNN_q(x) = \sum_{p=1}^{P} \left(\sum_{k=1}^{K} a_{q,p,k} (S f_p)_{x,k} \right)$$

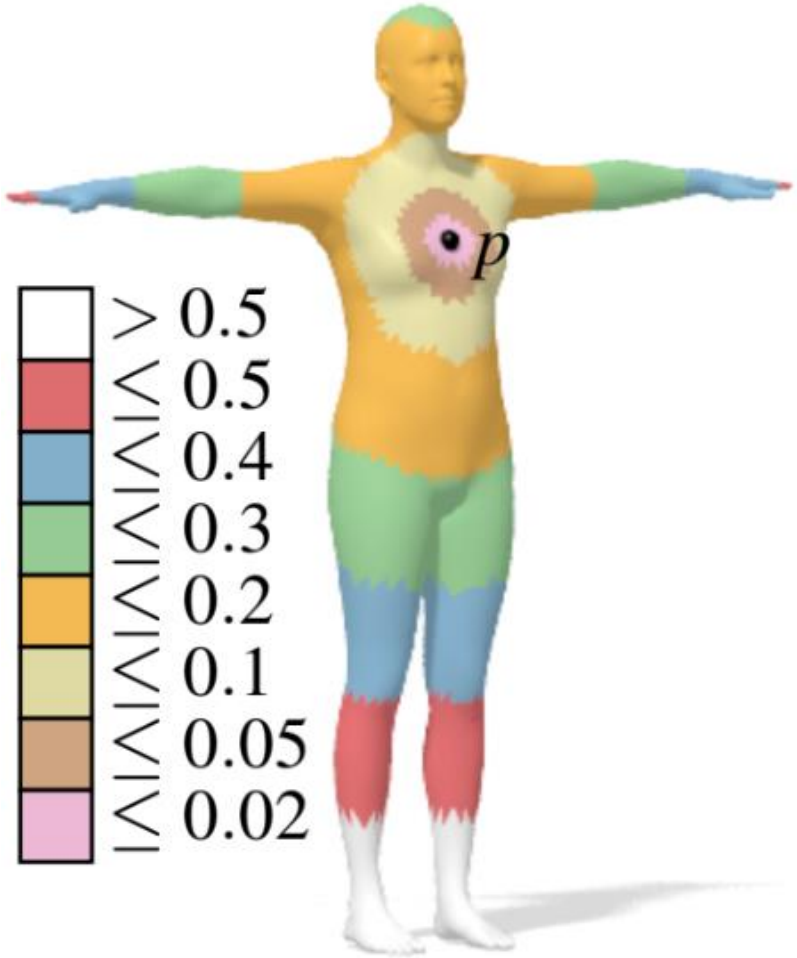
$$(S f_p)_{x,k} = \langle f_p, \sum_{l=1}^{K} \left(g_p(\lambda_l) \phi_l(x) \phi_k(x) \right) \phi_l \rangle_{\mathcal{X}}$$

Learned windows

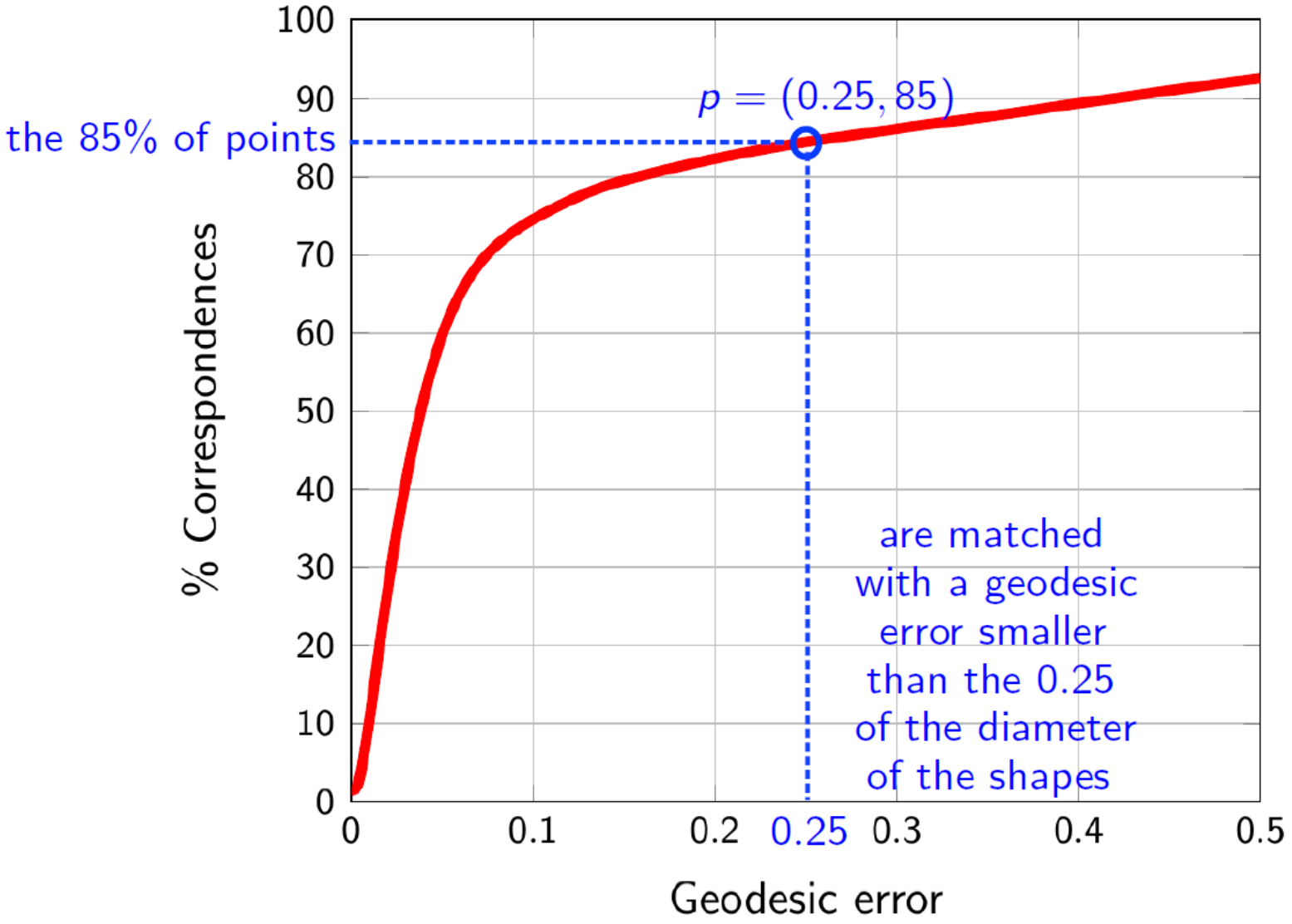


Geodesic error



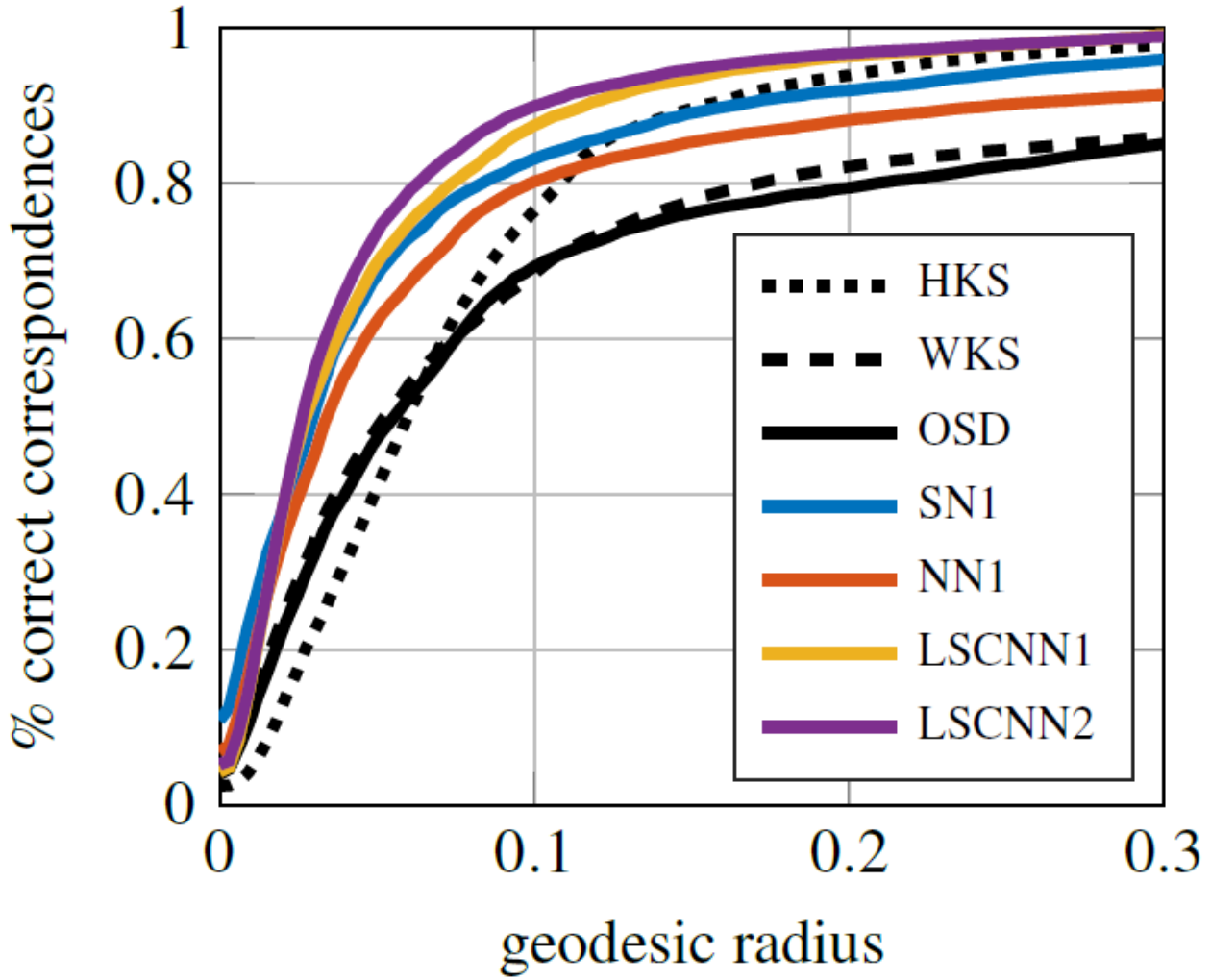


Cumulative error curve

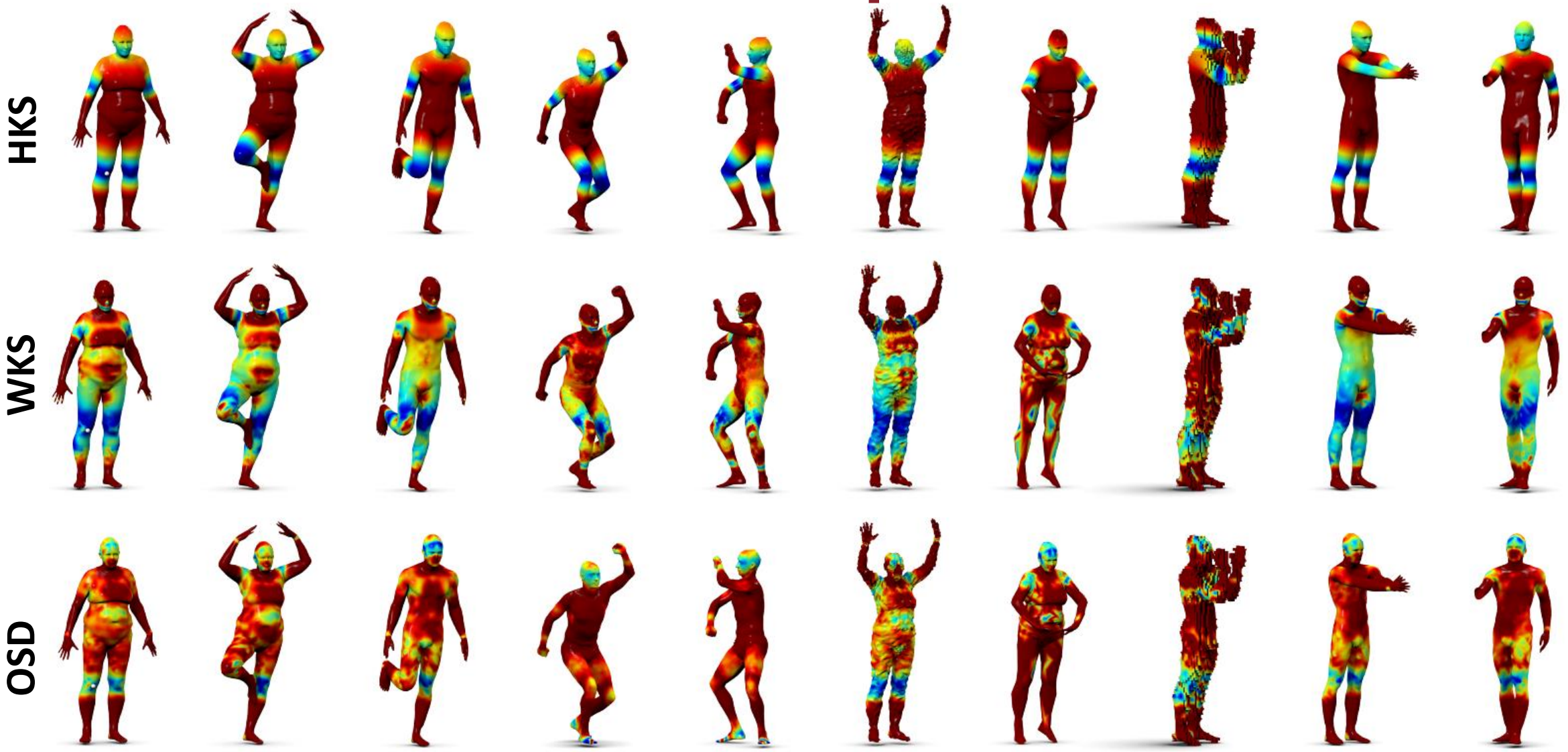


"Blended intrinsic maps", *Kim* et al, 2011

Quantitative comparison



Qualitative comparison



Qualitative comparison

Some conclusions

- Spectral descriptors are invariant to isometric deformations
- Spectral descriptors do not solve the symmetries
- Spectral descriptors can be generalized via data-driven approaches
- WFT can characterize locally the shape
- The data-driven approaches outperform the standard spectral ones
- Other deformations (for from isometries) can not be faced

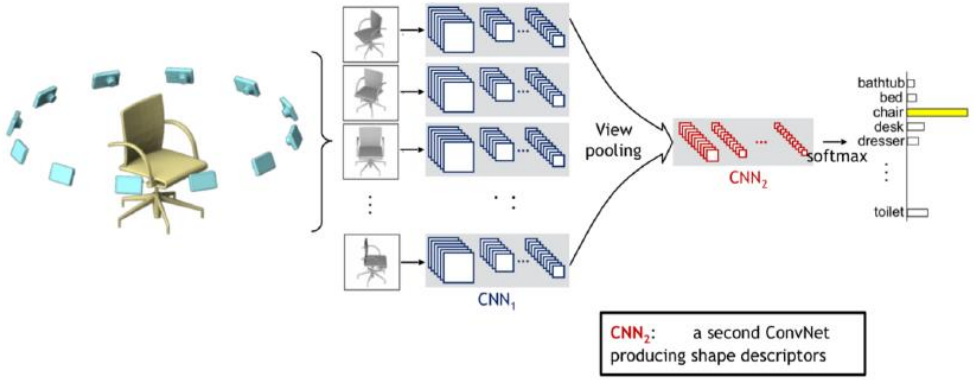
Other data-drive approaches

The data-driven approaches seem well-suited to solve the point-to-point mathcing problem

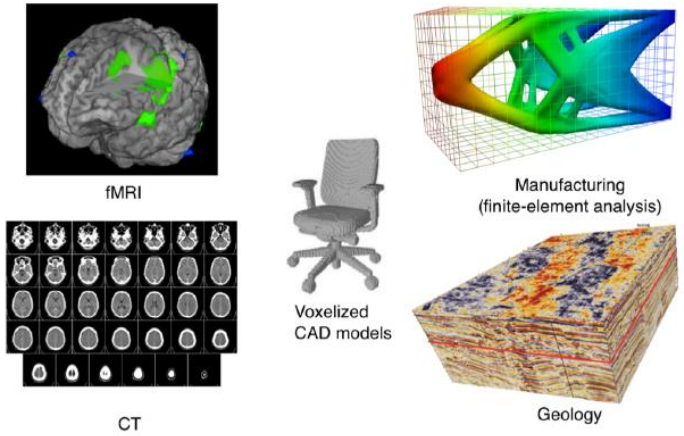
Recentrly this gives rise to a family of approaches that can be collected under the name of:

GEOMETRIC DEEP LEARNING

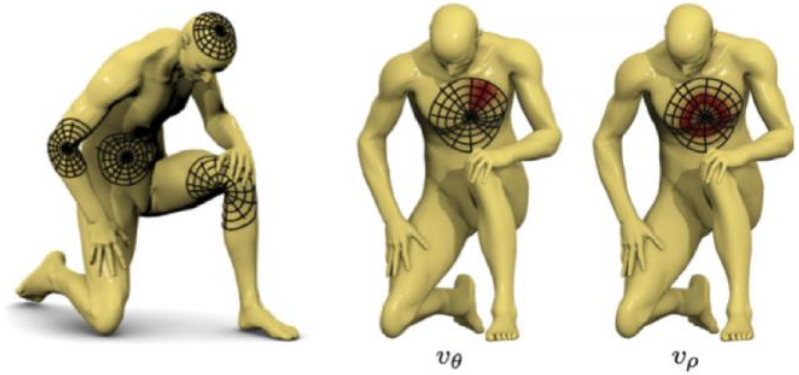
Geometric deep learning



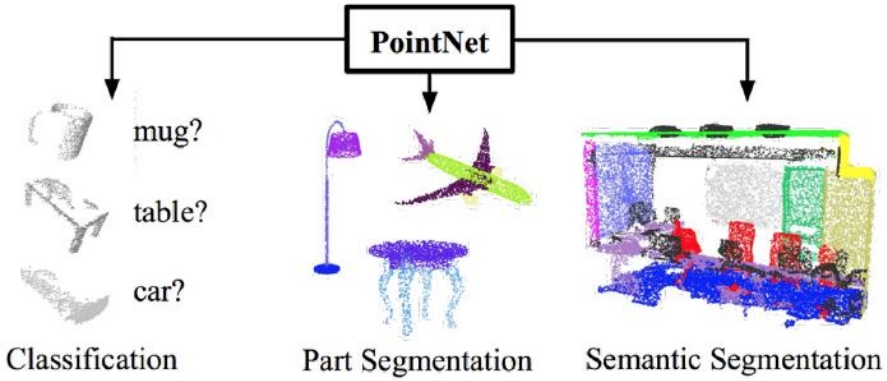
View-based



Volumetric



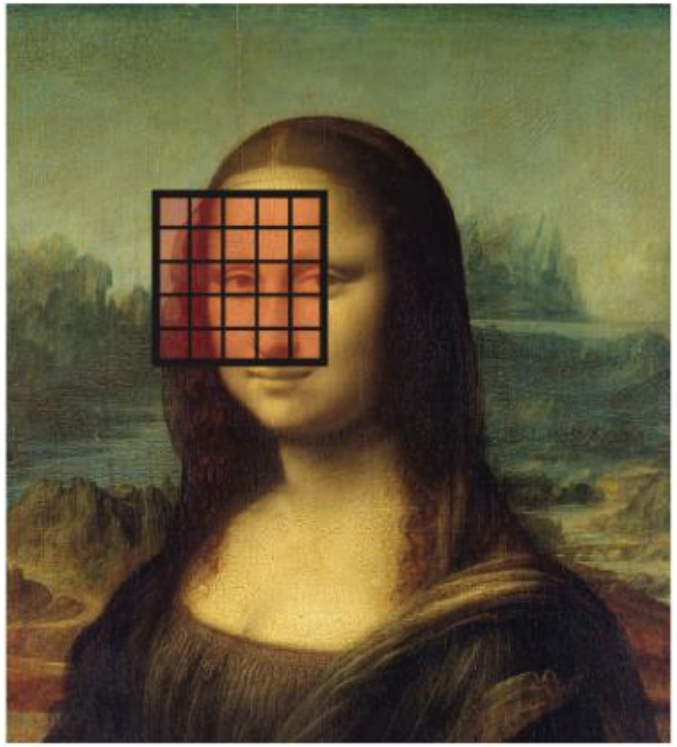
Intrinsic (surface-based)



Point-based

Slide credit M. *Ovsjanikov*

Alternative convolutions



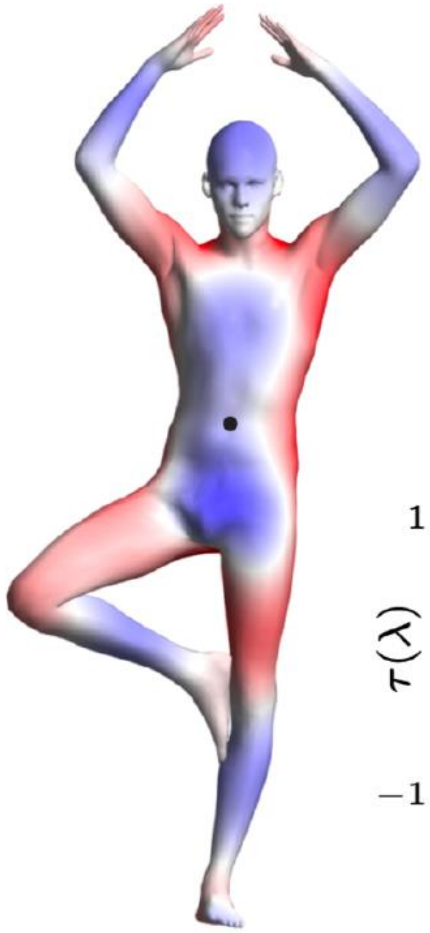
Euclidean

Non-Euclidean

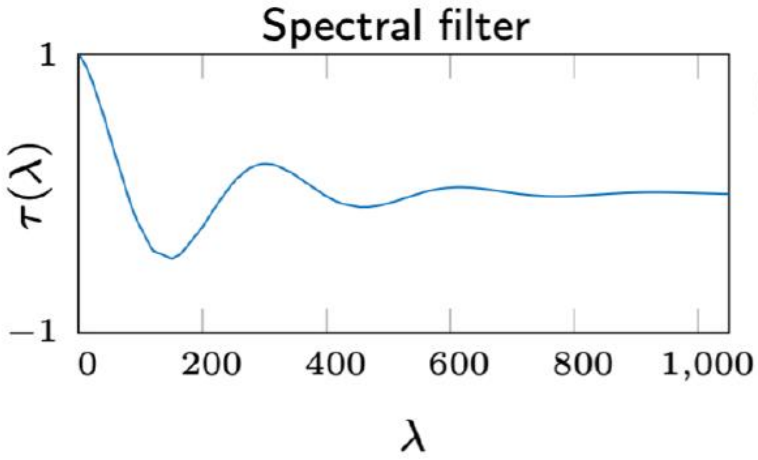
Slide credit M. *Ovsjanikov*

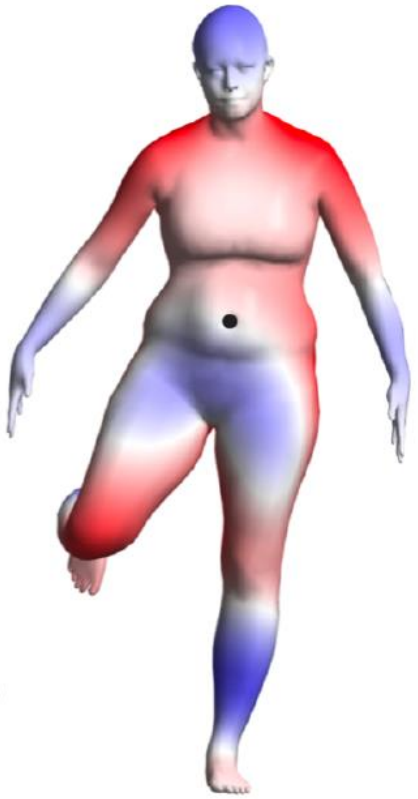
Limits of the spectral convolution

Unfortunately spectral convolution as many limitations (shape, shift invariances).



$\mathbf{\Phi}_1 \tau(\mathbf{\Lambda}_1)\mathbf{\Phi}_1^\top \boldsymbol{\delta}_0$





$\mathbf{\Phi}_2 \tau(\mathbf{\Lambda}_2)\mathbf{\Phi}_2^\top \boldsymbol{\delta}_0$

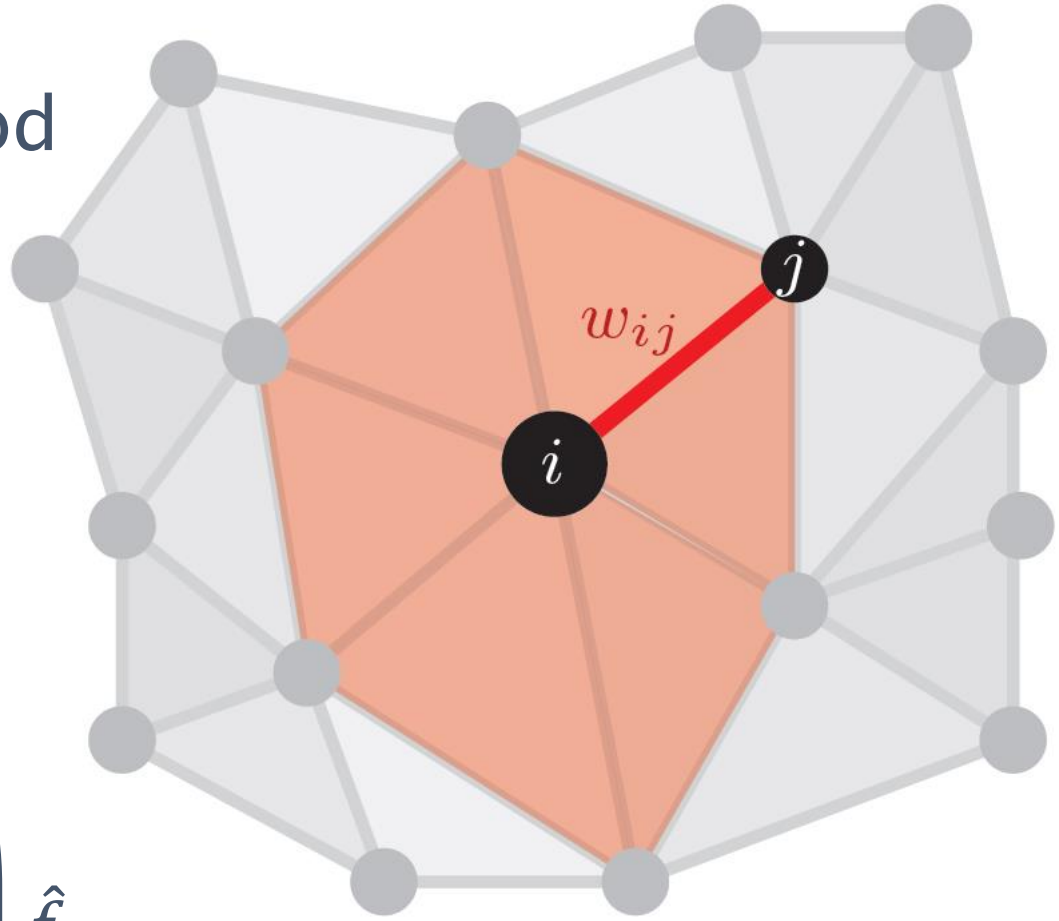
Slide credit E. *Rodolà*

Spectral convolution:

Slide credit E. *Rodolà*

Lapalcian operator Δ which
Acts locally on the neighborhood

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



LBO eigenvectors generalize
the Fourier basis $\hat{f} = \Phi^\dagger f$

Spectral convolution:

$$f \star g = \Phi \begin{pmatrix} \hat{g}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & & & & 0 \\ 0 & 0 & & \cdots & & \hat{g}_n \end{pmatrix} \hat{f}$$

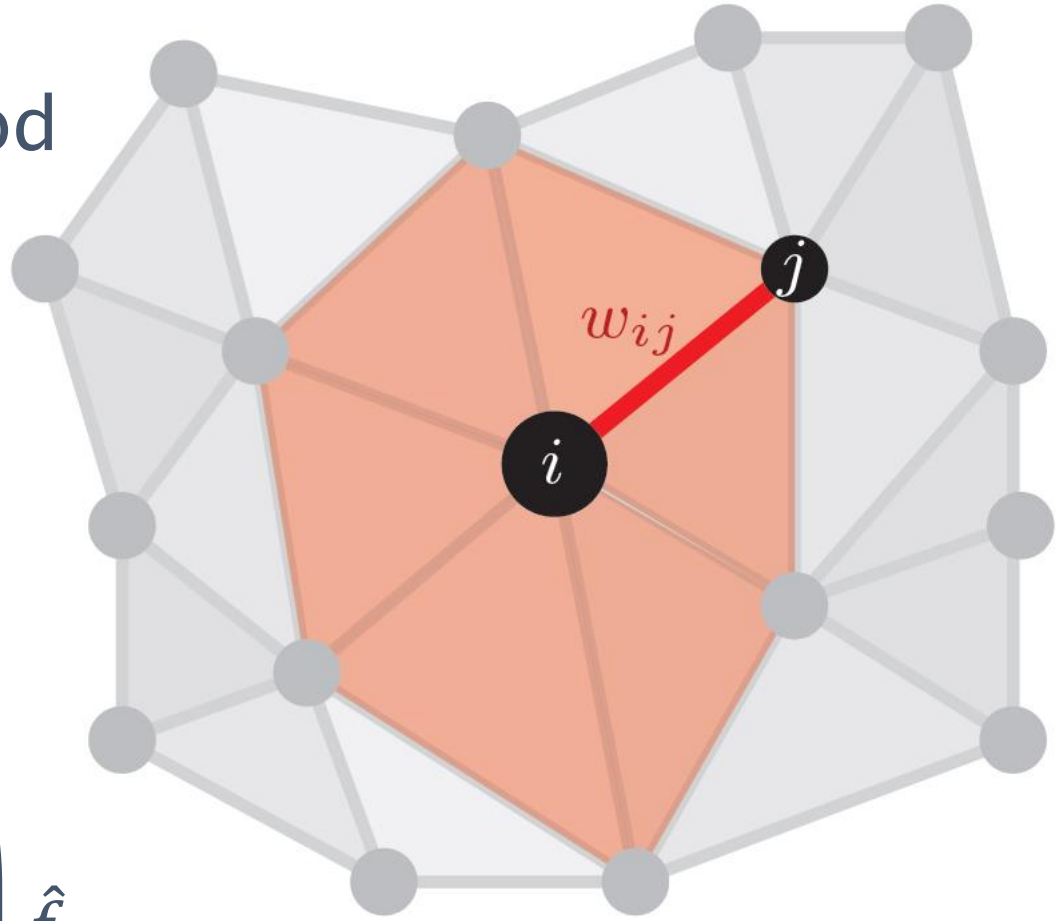
"Spectral Networks and Locally Connected Networks on Graphs", Bruna et al., 2014

Spectral convolution:

Slide credit E. *Rodolà*

Lapalcian operator Δ which
Acts locally on the neighborhood

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



LBO eigenvectors generalize
the Fourier basis $\hat{f} = \Phi^\dagger f$

Spectral convolution:

$$f \star g = \Phi \begin{pmatrix} \hat{g}_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & & & & 0 \\ 0 & 0 & & \cdots & & \hat{g}_n \end{pmatrix} \hat{f}$$

"Spectral Networks and Locally Connected Networks on Graphs", Bruna et al., 2014

Spectral convolution:

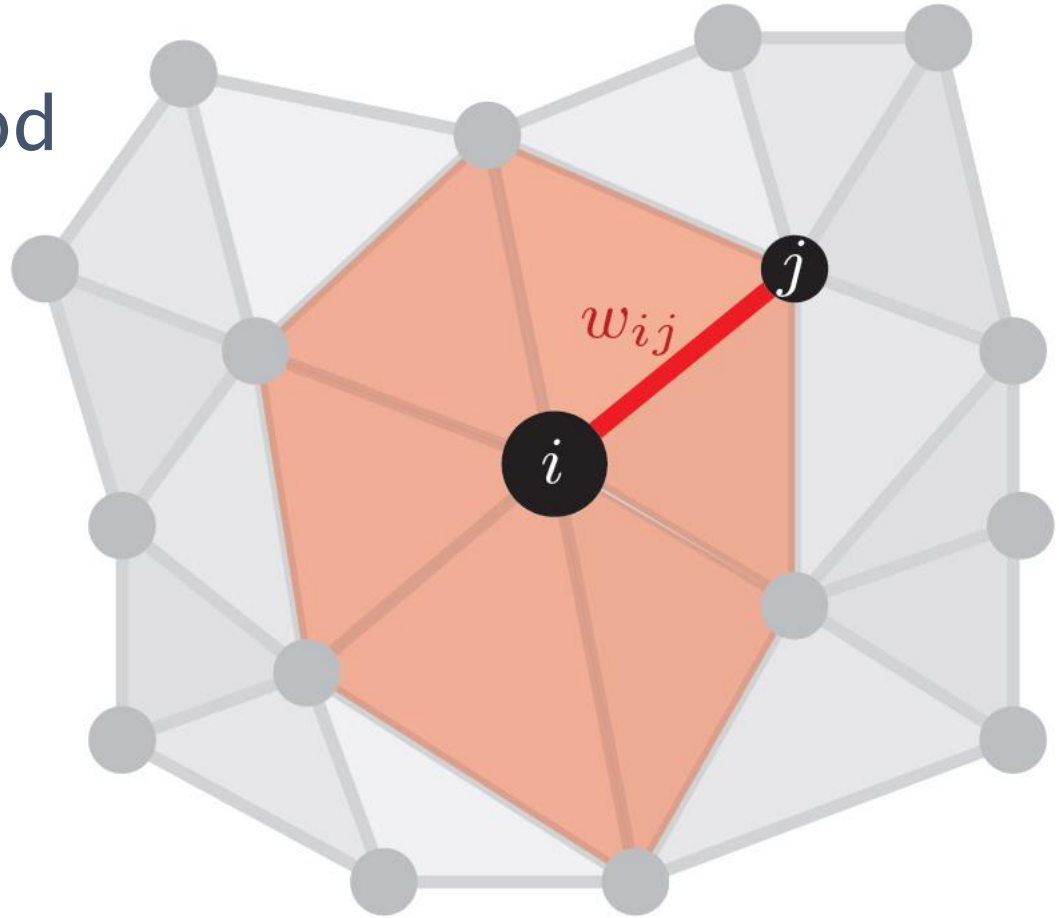
Slide credit E. *Rodolà*

Lapalcian operator Δ which
Acts locally on the neighborhood

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

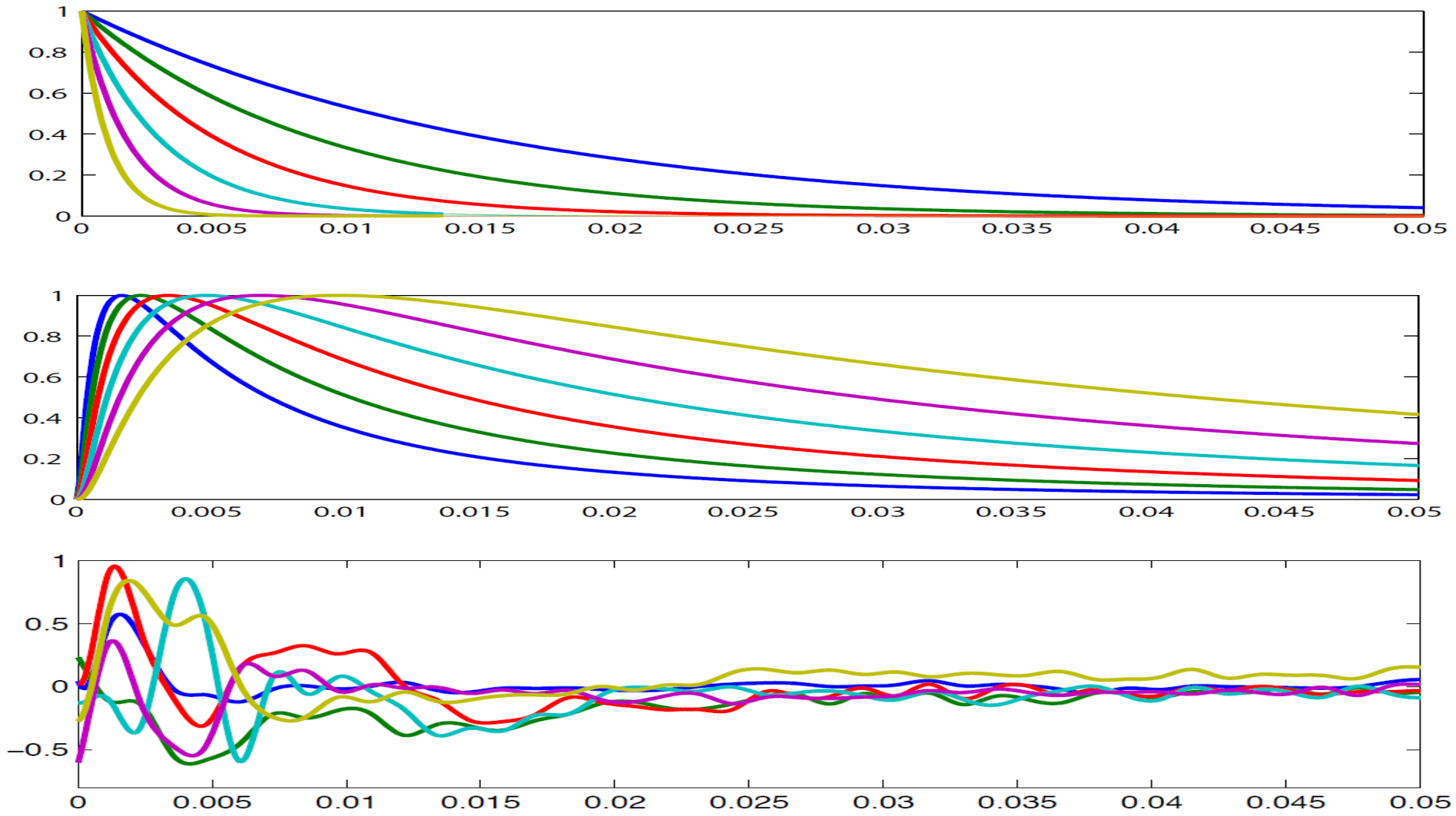
LBO eigenvectors generalize the Fourier basis $\hat{f} = \Phi^\dagger f$

Spectral convolution: $f' = \Phi\, \tau(\Lambda)\, \Phi^\dagger f$



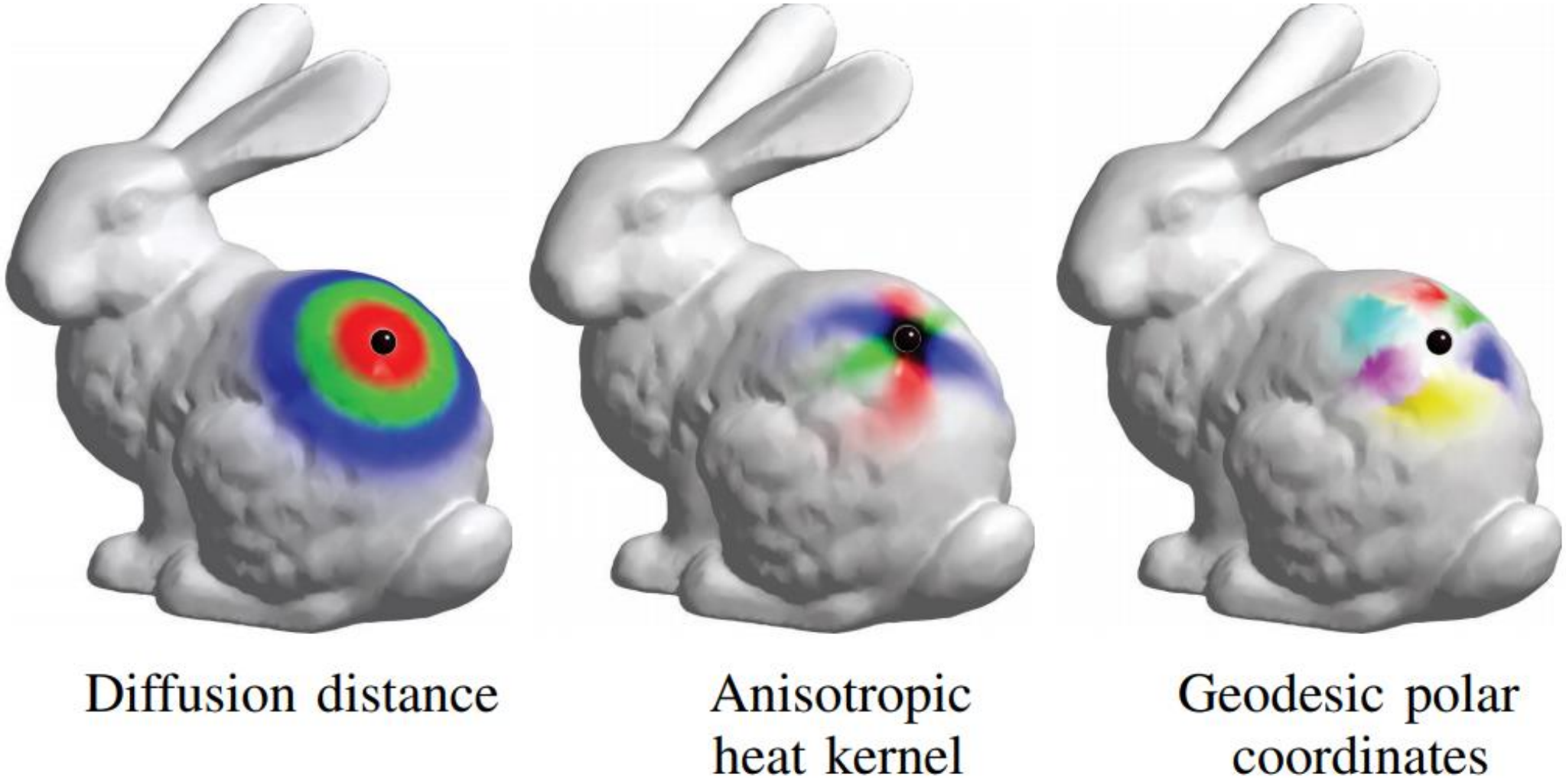
"Spectral Networks and Locally Connected Networks on Graphs", Bruna et al., 2014

Spectral filtering convolution:



Slide credit E. *Rodolà*

Geometric deep learning



"Geometric deep learning: going beyond Euclidean data" *Bronstein et al., 2016*