Lecture 3:

Geometry Processing

Simone Melzi, Marco tarini

Milano, 15/09/2021

PhD School – Learning on 3D geometries

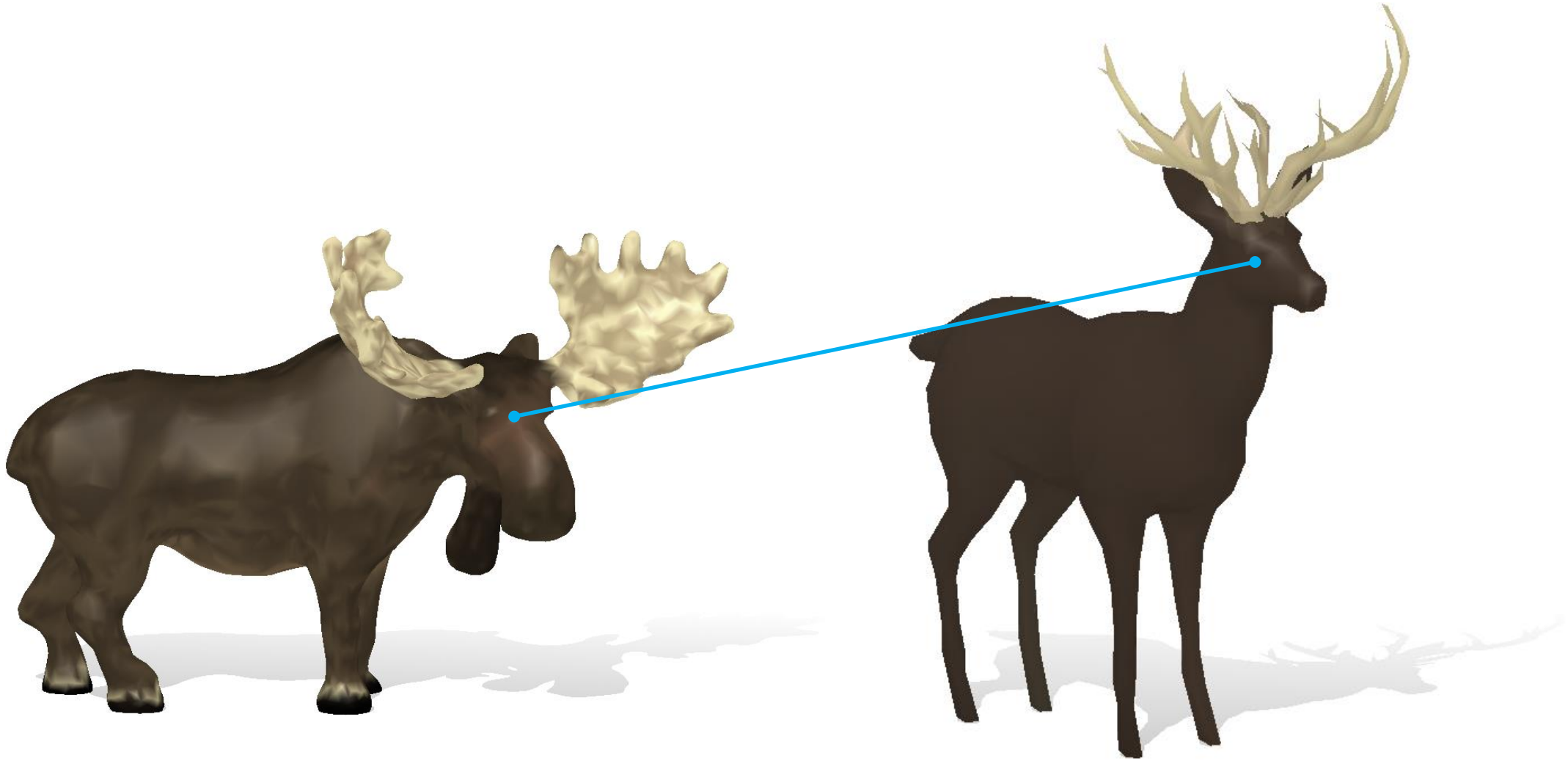
LA STATALE Università degli Studi di Milano

SAPIENZA Università di Roma

We saw what is a (single shape)

And if we have more than one shape?

An interesting problem

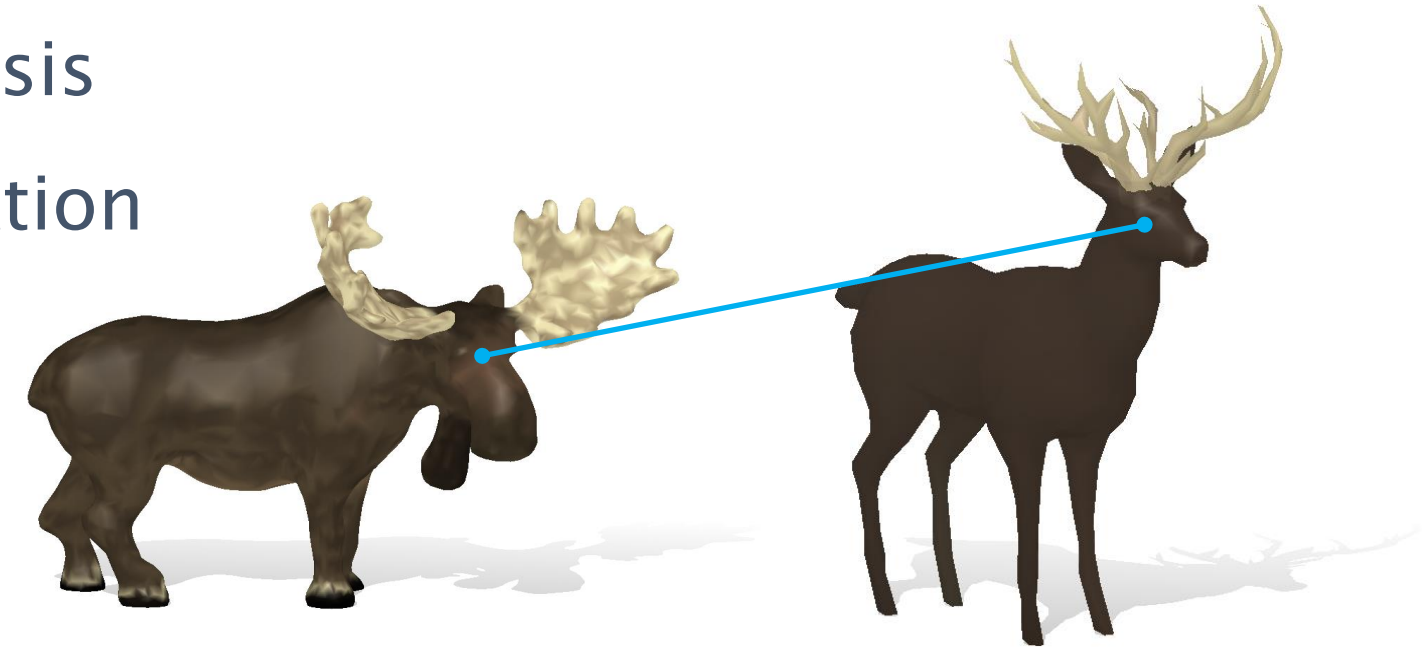


Shape matching

Requested by several applications

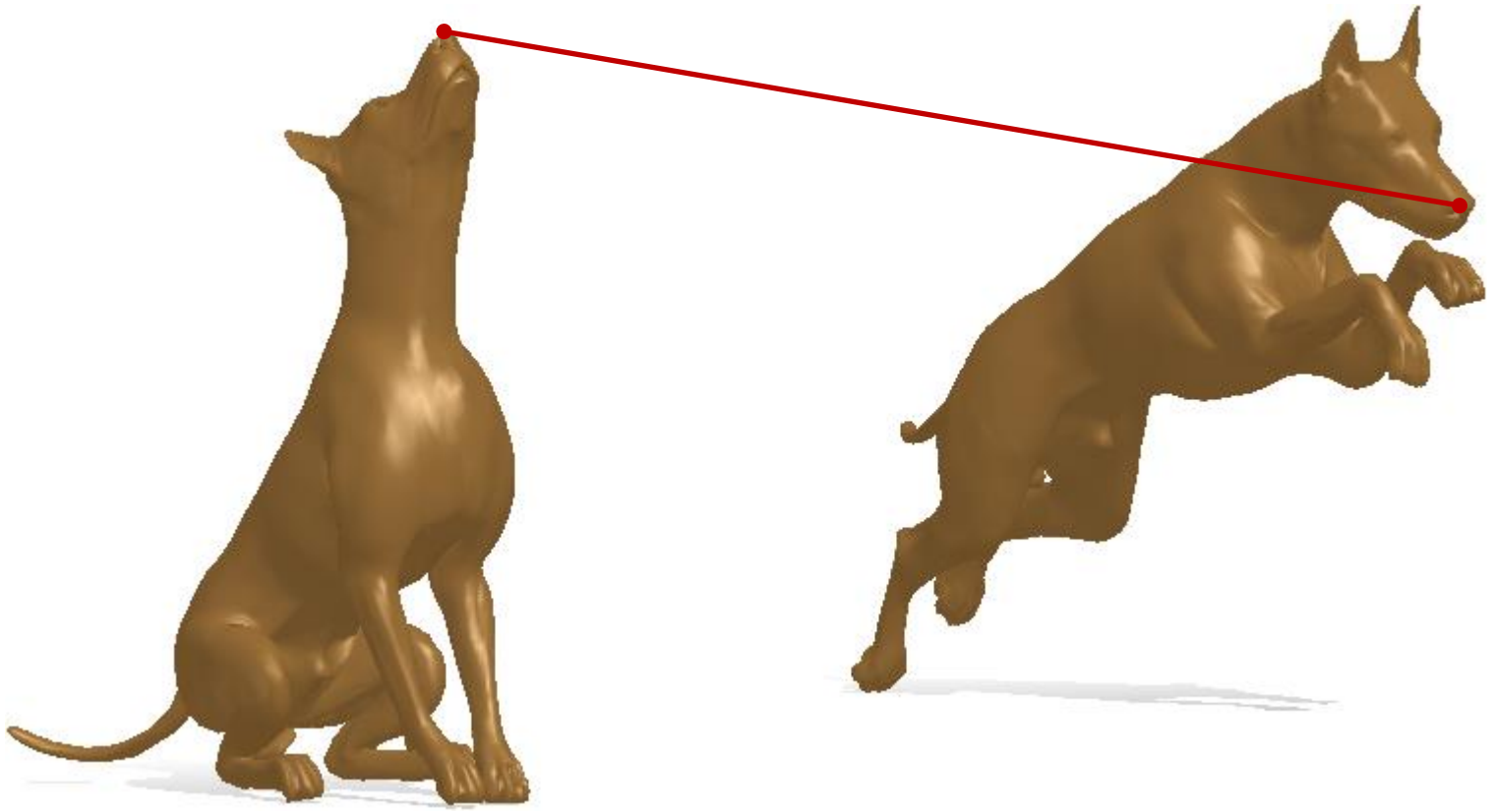
- Deformation transfer
- Texture transfer
- Registration
- Statistical analysis
- Shape interpolation
- Animation
- Tracking
- Meshing

…

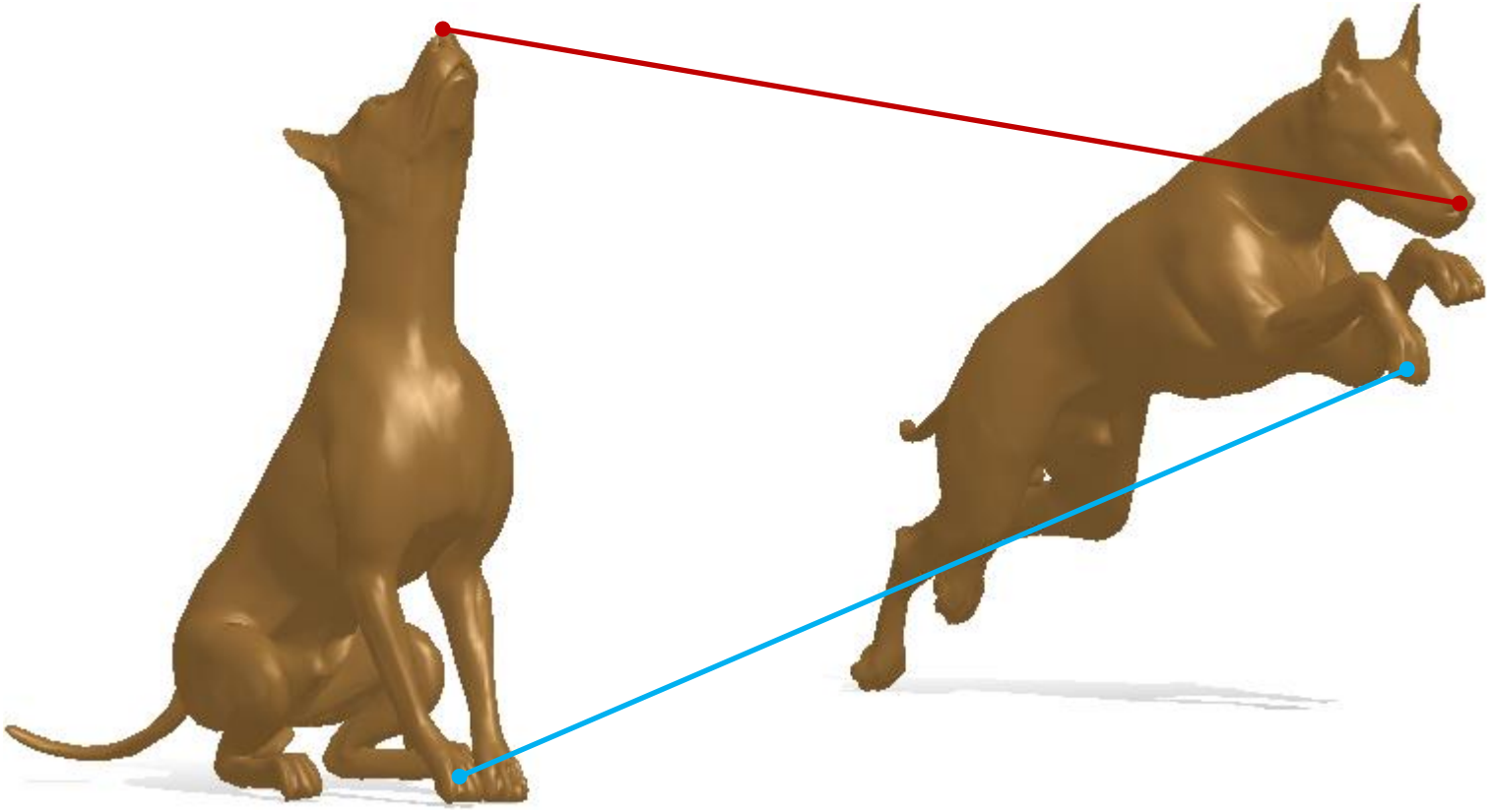


General Idea:

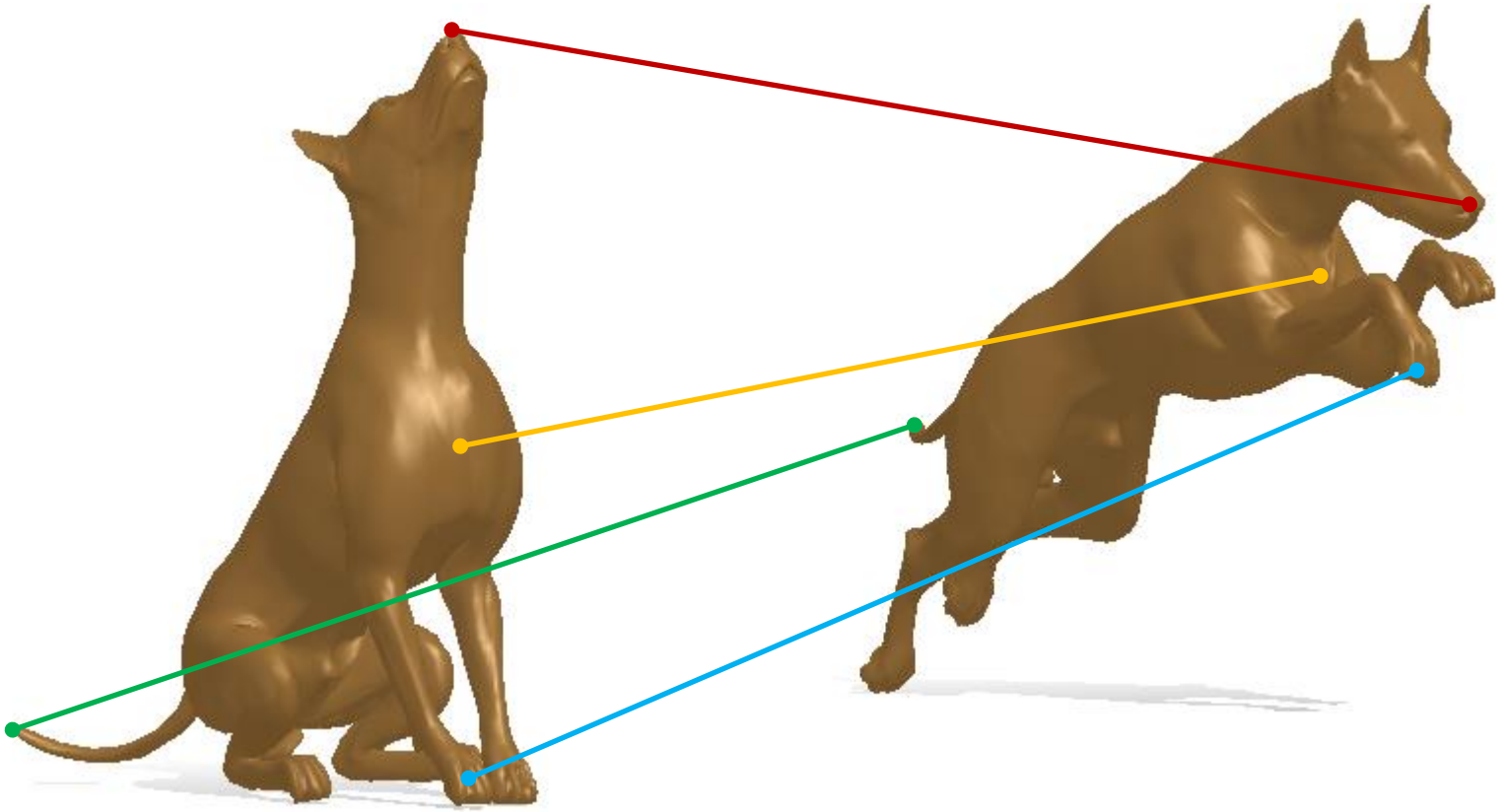
General Idea:



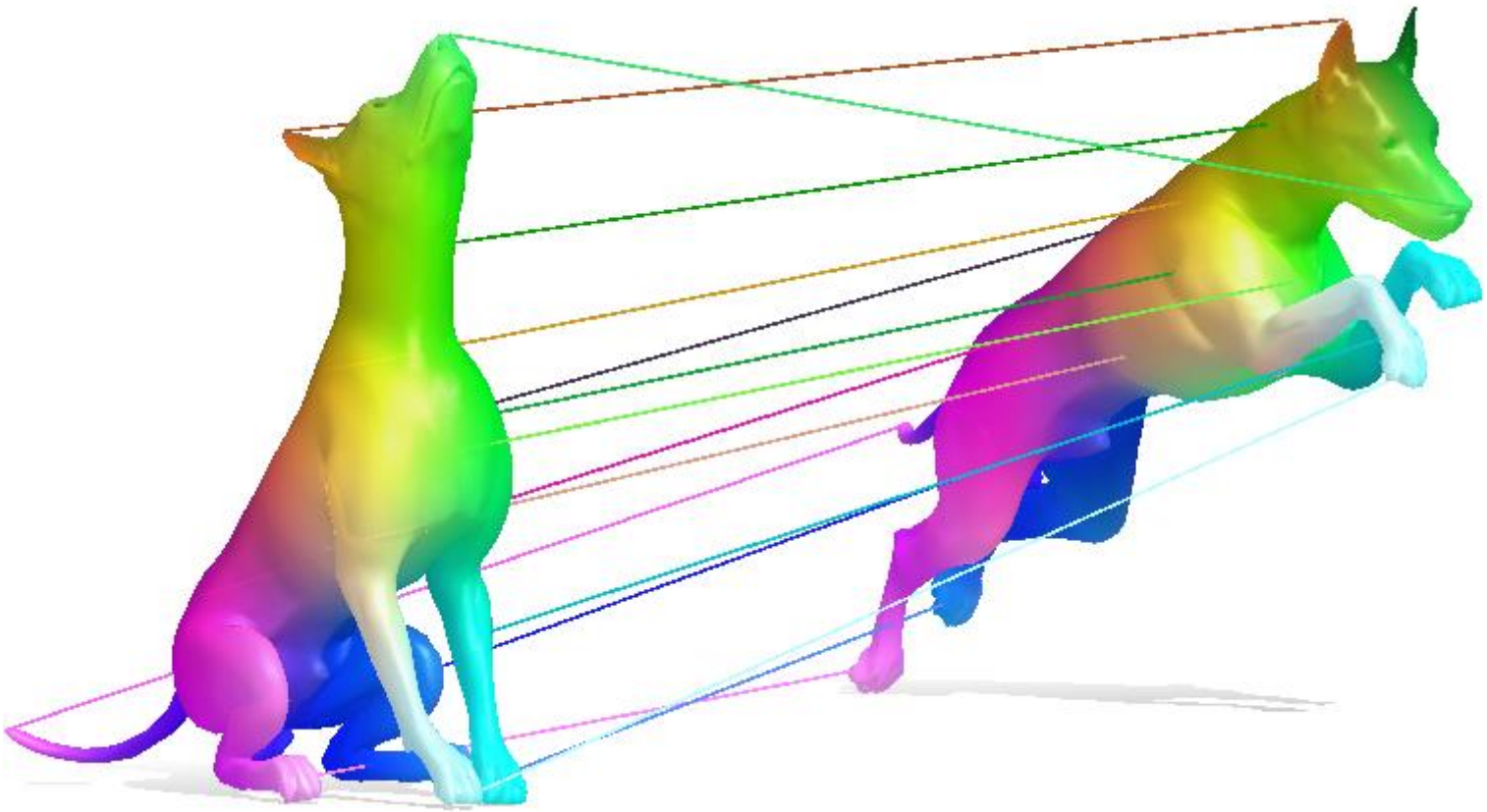
General Idea:



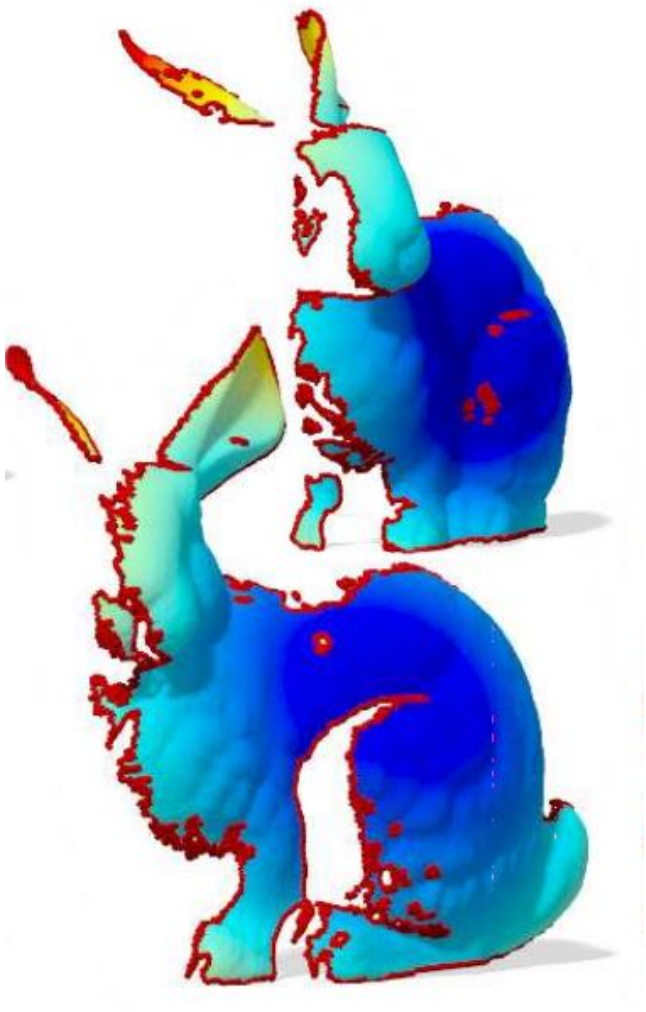
General Idea:



General Idea:



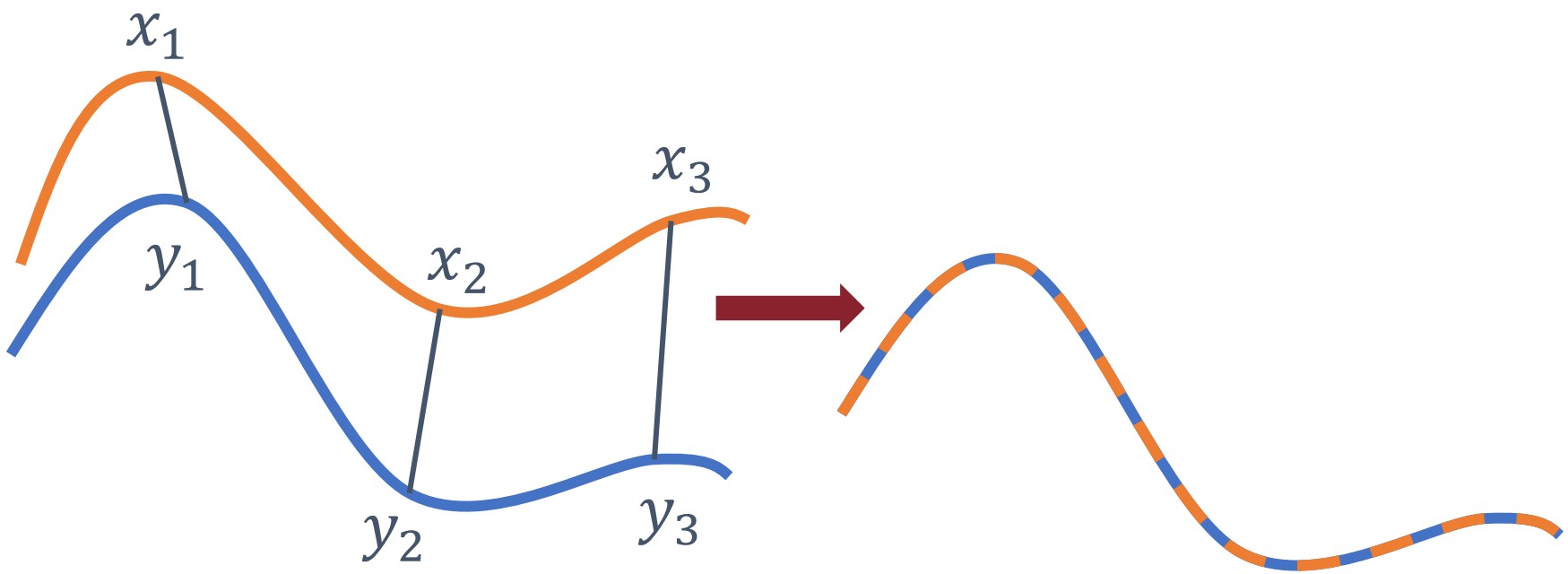
Not only non rigid:



These are 2 different views of the same object that its rigid

Not only meshes:

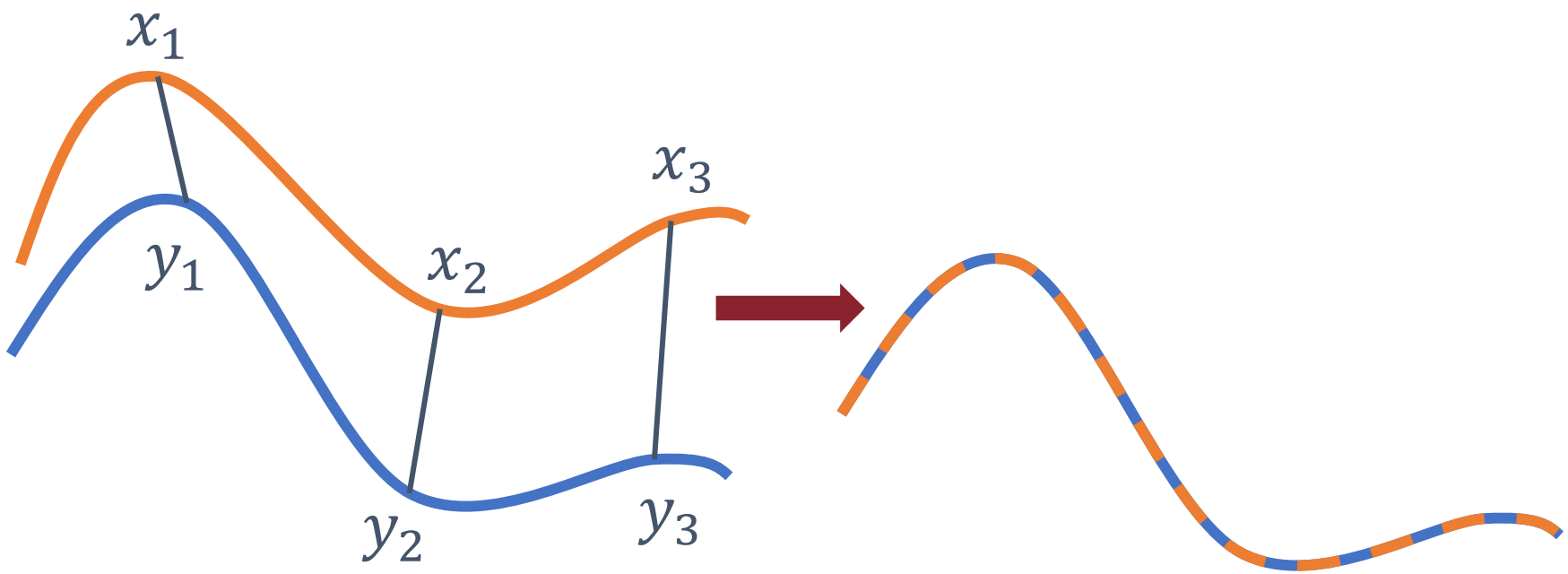
Starting case: 1D in $\mathbb{R}^2$:



Slide credits to M. Ovsjanikov

Starting case: 1D in $\mathbb{R}^2$:

The solution is a rigid transformation $\mathbb{R}^2 \to \mathbb{R}^2$



ICP approach = iterate alternating:
(1) finding correspondences;
(2) finding optimal transformation.

Slide credits to M. Ovsjanikov

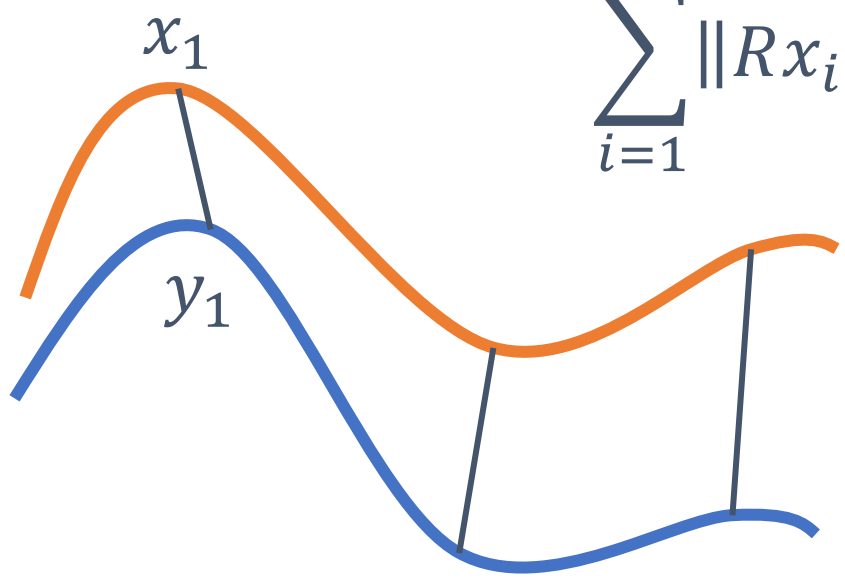
ICP algorithm:

ICP = Iterative Closest Point

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$
Iterate (a stop criteria is satisfied):

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R optimal rotation and t translation s.t.:

$$\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$$

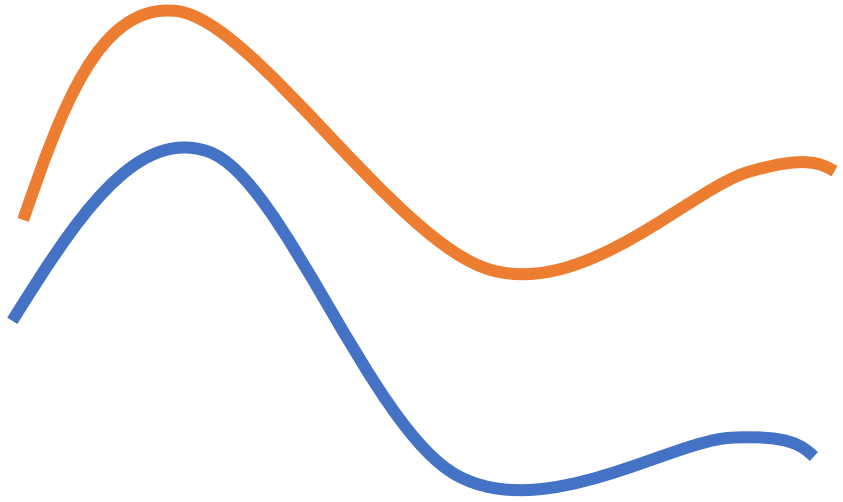


Slide credits to M. Ovsjanikov

ICP algorithm:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$

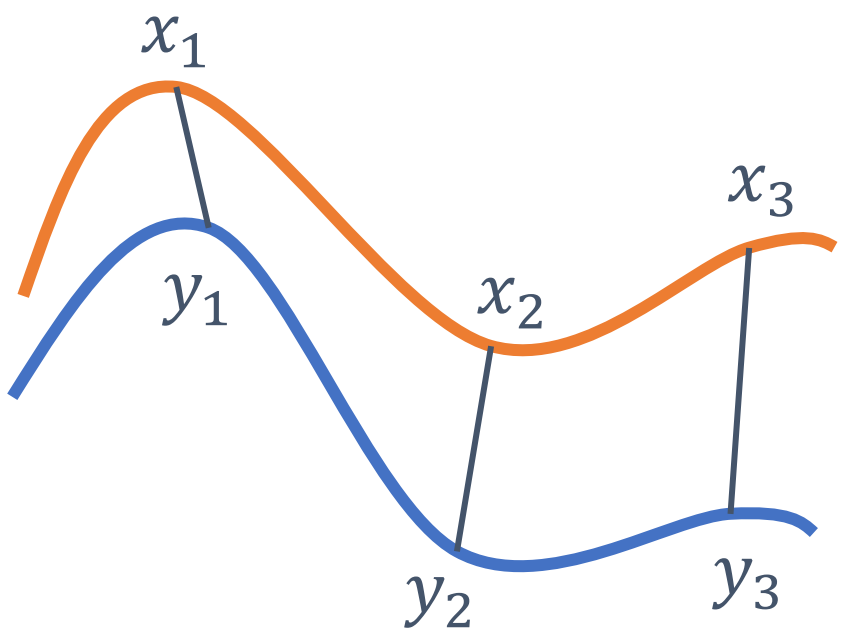


Slide credits to M. Ovsjanikov

ICP algorithm:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$

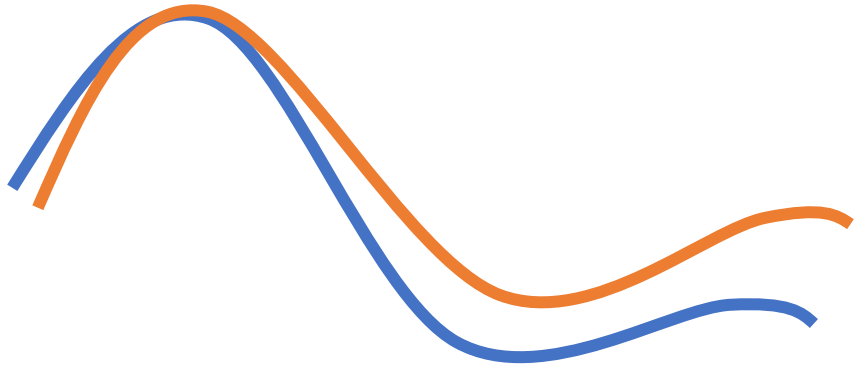


Slide credits to M. Ovsjanikov

ICP algorithm:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$

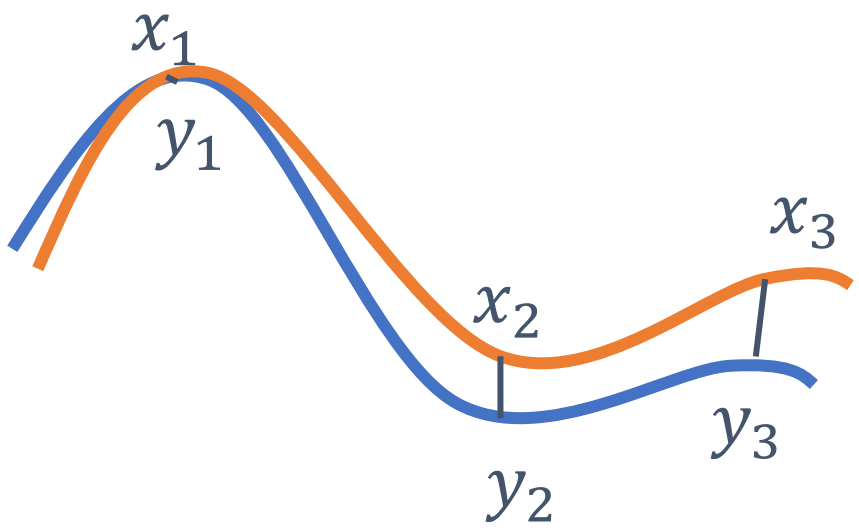


Slide credits to M. Ovsjanikov

ICP algorithm:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$

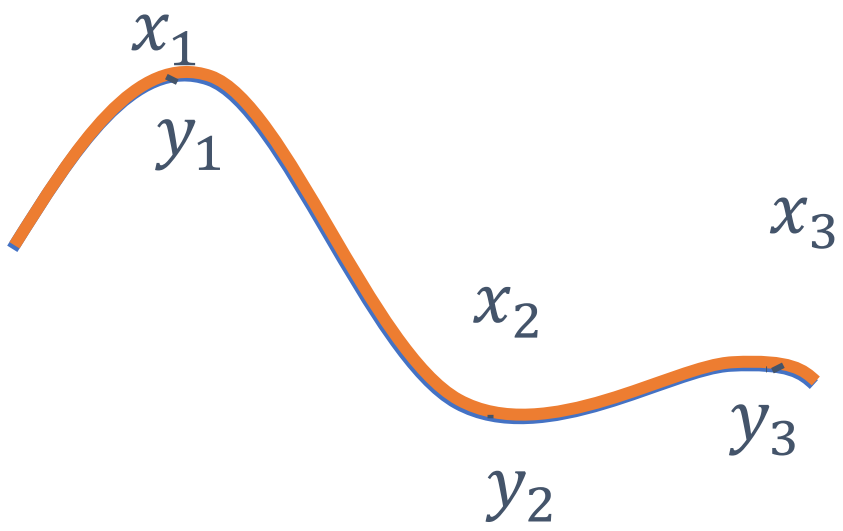


Slide credits to M. Ovsjanikov

ICP algorithm:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$

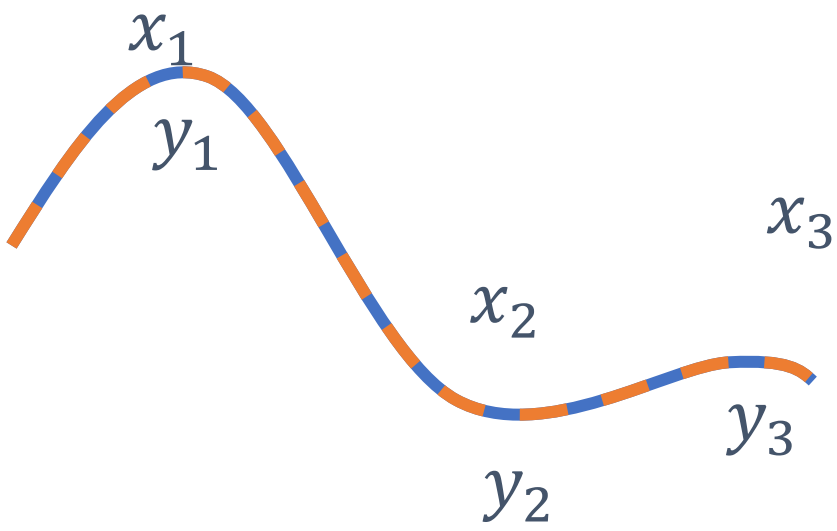


Slide credits to M. Ovsjanikov

ICP algorithm:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

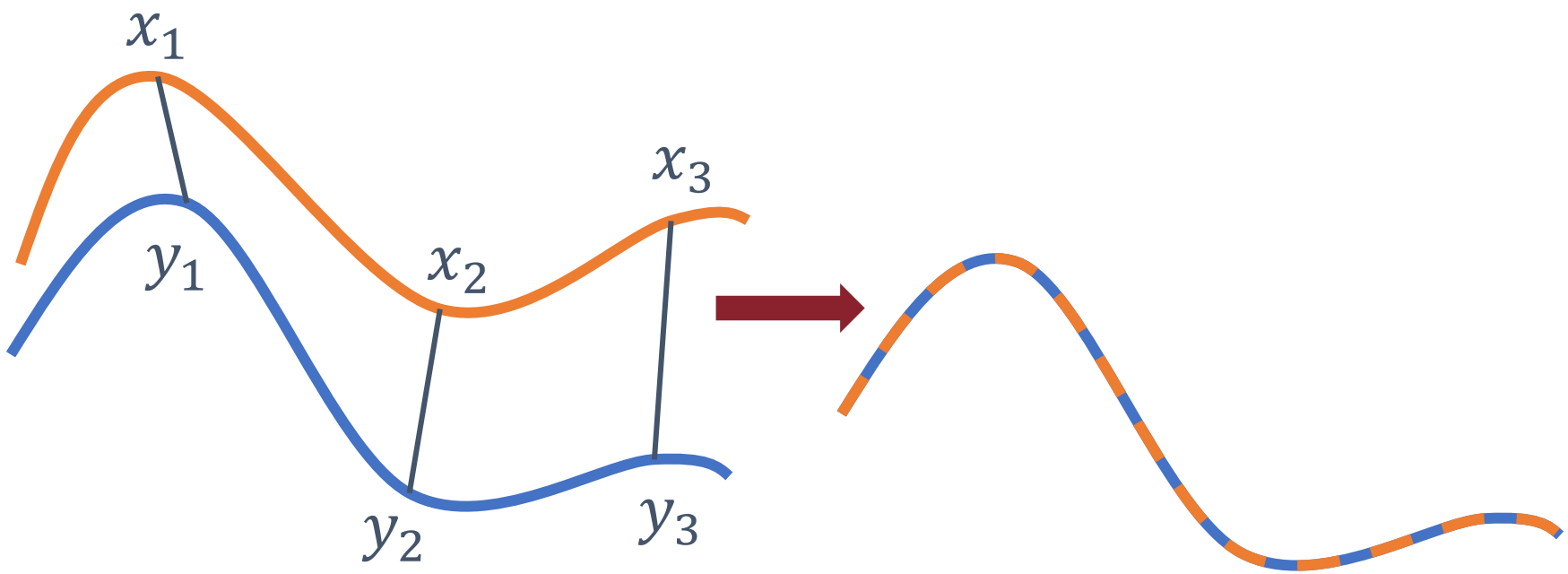
1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$



Slide credits to M. Ovsjanikov

ICP steps:

The ICP is for rigid matching



ICP requires 2 computations:
(1) finding correspondences;
(2) finding optimal transformation.

Slide credits to M. Ovsjanikov

ICP nearest neighbor:

Closest point

$$y_i = \underset{y \in \mathcal{Y}}{\operatorname{argmin}}\, d(y, x_i) = \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \|y - x_i\|, \forall x_i \in \mathcal{X}$$

How to solve it efficiently? $\mathcal{O}(MN)$ where M is the number of points on $\mathcal{Y}$ and N the ones of $\mathcal{X}$

Alternative divide $\mathbb{R}^2$ into Voronoi cells:

$$V(y \in \mathcal{Y}) = \{z \in \mathbb{R}^2 : \|y - z\| < \|y' - z\|, \forall y' \in \mathcal{Y} \neq y\}$$

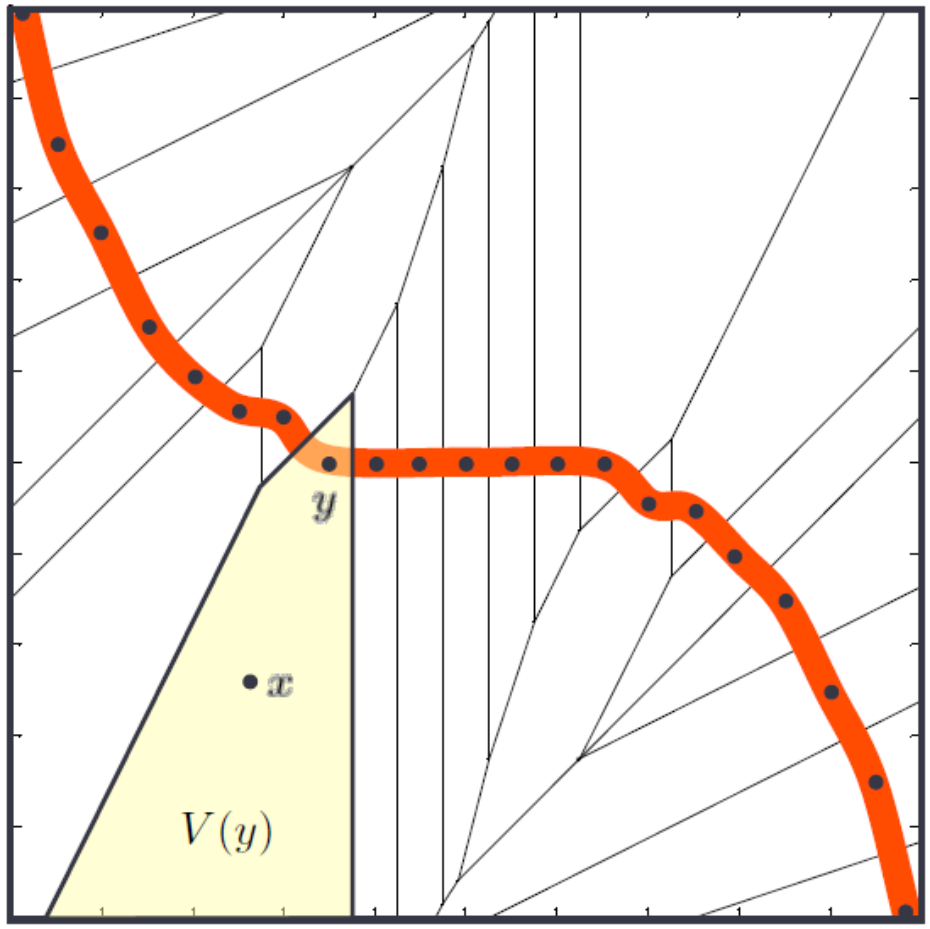
This is done once, then we should just check the Voronoi cell to which belongs $x_i, \forall x_i \in \mathcal{X}$

Slide credits to M. Ovsjanikov

Voronoi cells:

Voronoi cells:

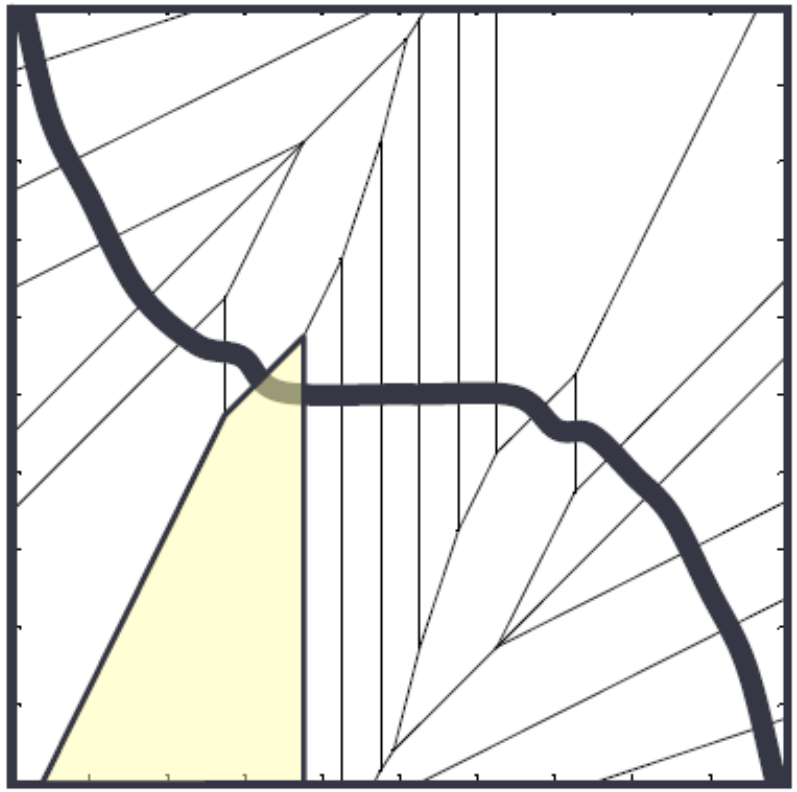
$$V(y \in \mathcal{Y}) = \{z \in \mathbb{R}^2 : \|y - z\| < \|y' - z\|, \forall y' \in \mathcal{Y} \neq y\}$$

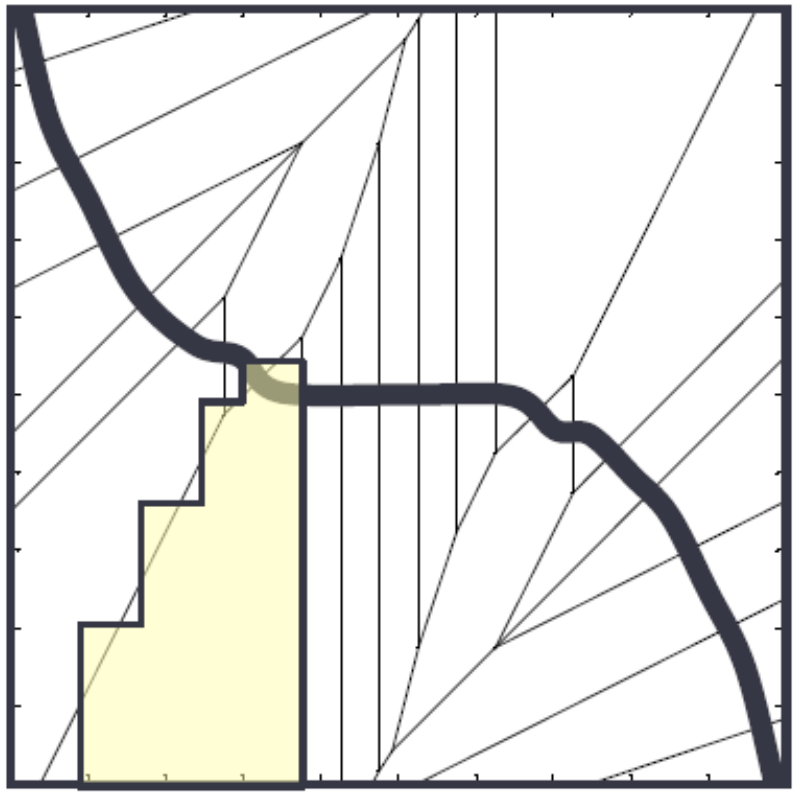


Slide credits to M. Bronstein

Approximate Voronoi cells:

Approximate Voronoi cells using binary space partition trees (e.g. kd-trees) $\mathcal{O}(Nlog(M))$





Slide credits to M. Bronstein

ICP optimal transformation:

Given two sets of ***corresponding*** points

$\{x_i\} \in \mathcal{X}, \{y_i\} \in \mathcal{Y}, \forall i \in \{1, \dots, N\}$;

Find R optimal rotation and t translation s.t.:

$$\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$$

Slide credits to M. Ovsjanikov

ICP optimal transformation:

Given two sets of ***corresponding*** points
$\{x_i\} \in \mathcal{X}, \{y_i\} \in \mathcal{Y}, \forall i \in \{1, \dots, N\}$;
Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$

1. Compute: $\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i$ and: $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$
2. Construct: $C = \sum_{i=1}^{N}(y_i - \mu_y)(x_i - \mu_x)^T$
3. Compute the SVD of C: $C = USV^T$
   If $\det(UV^T) = 1 \Longrightarrow R = UV^T$
   Else $R = U\tilde{S}V^T$, where $\tilde{S} = diag(1,1, \dots, -1)$
4. Set $t = \mu_y - R\mu_x$

Arun et al., Least-Squares Fitting of Two 3-D Point Sets

Slide credits to M. Ovsjanikov

ICP optimal transformation:

Given two sets of ***corresponding*** points
$\{x_i\} \in \mathcal{X}, \{y_i\} \in \mathcal{Y}, \forall i \in \{1, \dots, N\}$;
Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$

1. Compute: $\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i$ and: $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$
2. Construct: $C = \sum_{i=1}^{N}(y_i - \mu_y)(x_i - \mu_x)^T$
3. Compute the SVD of C: $C = USV^T$
   If $\det(UV^T) = 1 \Longrightarrow R = UV^T$
   Else $R = U\tilde{S}V^T$, where $\tilde{S} = diag(1,1, \dots, -1)$
4. Set $t = \mu_y - R\mu_x$

The SVD is very fast because $C \in \mathbb{R}^{2\times2}$ or $C \in \mathbb{R}^{3\times3}$

Arun et al., Least-Squares Fitting of Two 3-D Point Sets

Slide credits to M. Ovsjanikov

Stopping criteria:

1. Cumulative distance below a fixed threshold
2. A fixed point has been reached
3. Maximum number of iterations

ICP converges to a local minimum
ICP depends on the initialization

Good initial guess = global minimum
Bad initial guess = very bad matching

Slide credits to M. Ovsjanikov

Variations:

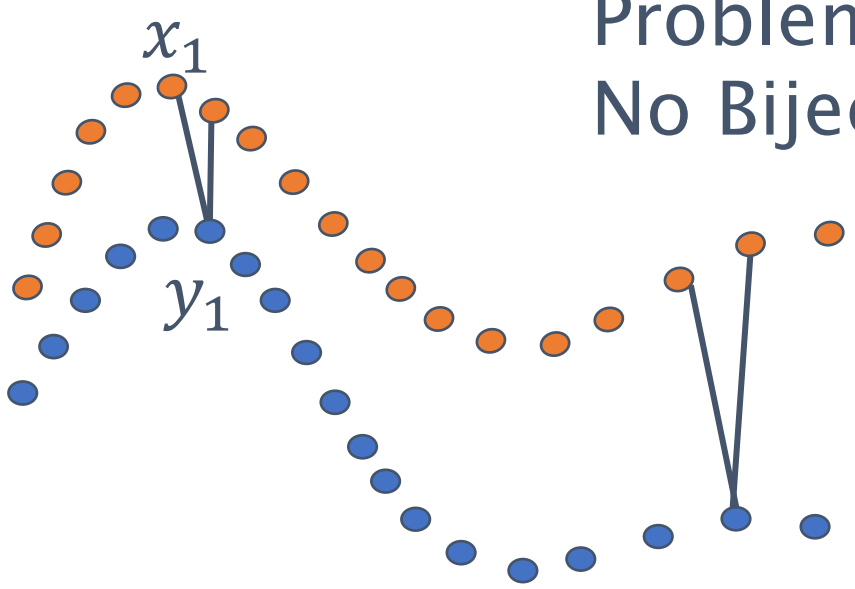
1. Selecting bi-directional matching
2. Weighting the correspondences
3. Rejecting outliers
4. Assigning an error metric to the current transformation
5. Minimizing the error metric of the transformation

Slide credits to M. Ovsjanikov

ICP Problems:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.: $\sum_{i=1}^{N} \|Rx_i + t - y_i\|_2^2$



Problem: multi-matching
No Bijection

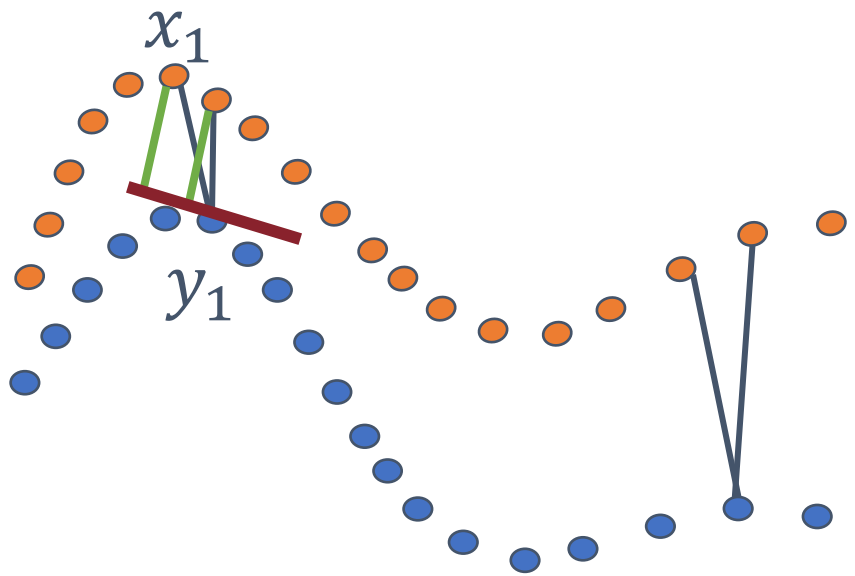
Slide credits to M. Ovsjanikov

ICP Problems:

Given 2 shape $\mathcal{X}$ and $\mathcal{Y}$ iterate:

1. $\forall x_i \in \mathcal{X}$ find the **nearest** neighbor $y_i \in \mathcal{Y}$;
2. Find R and t s.t.:

$$\sum_{i=1}^{N} d\left(Rx_i + t, P(y_i)\right) = \sum_{i=1}^{N} \left(\left(Rx_i + t, P(y_i)\right)^T \boldsymbol{n}_{y_i}\right)^2 =$$

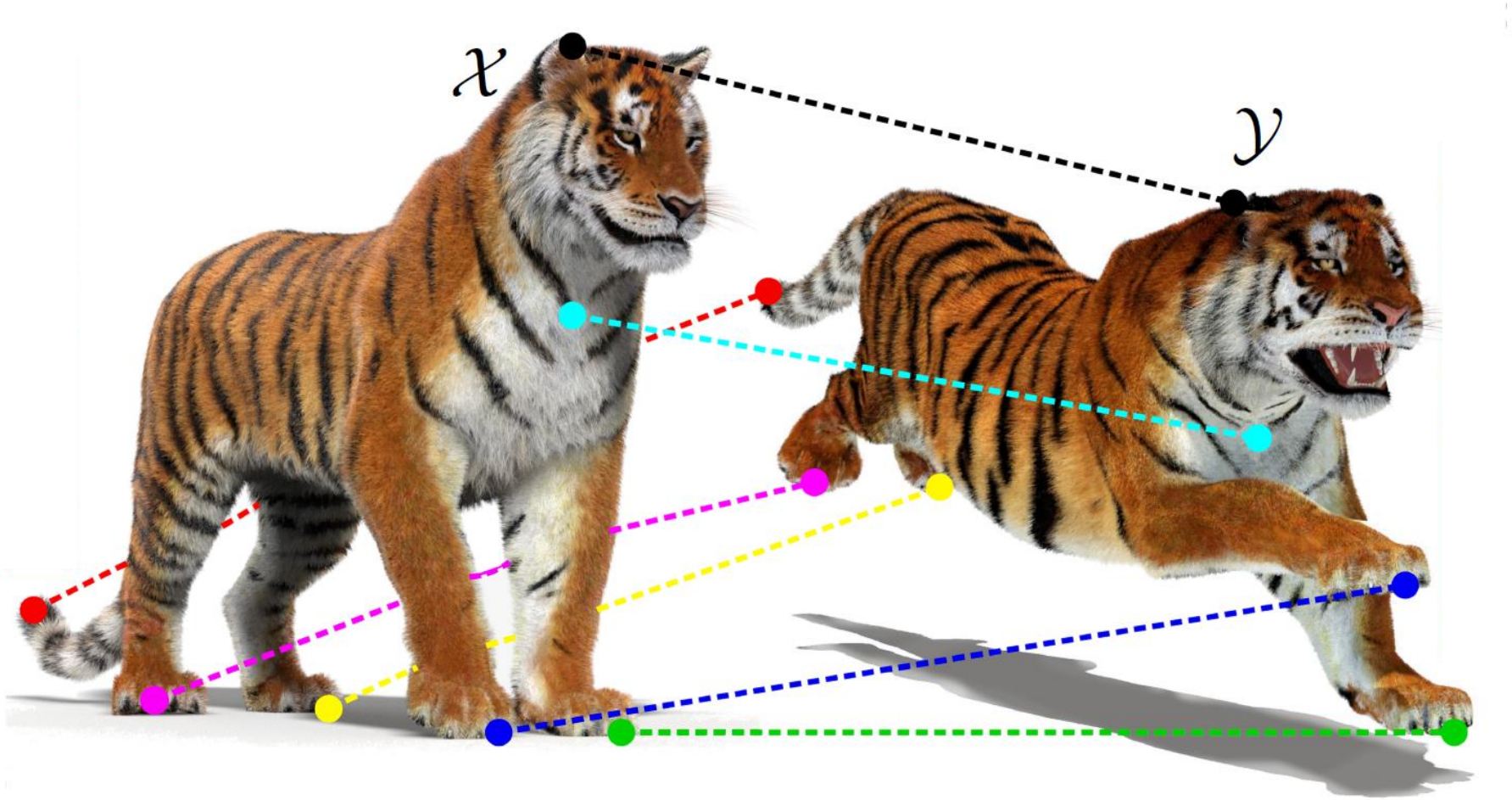


Solution:
minimize distance
to the tangent
plane

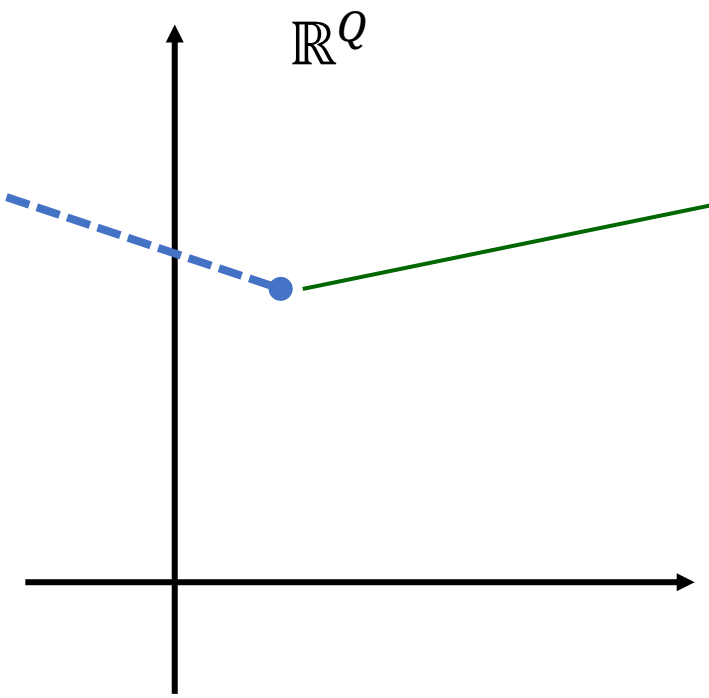
Slide credits to M. Ovsjanikov

Non-rigid matching

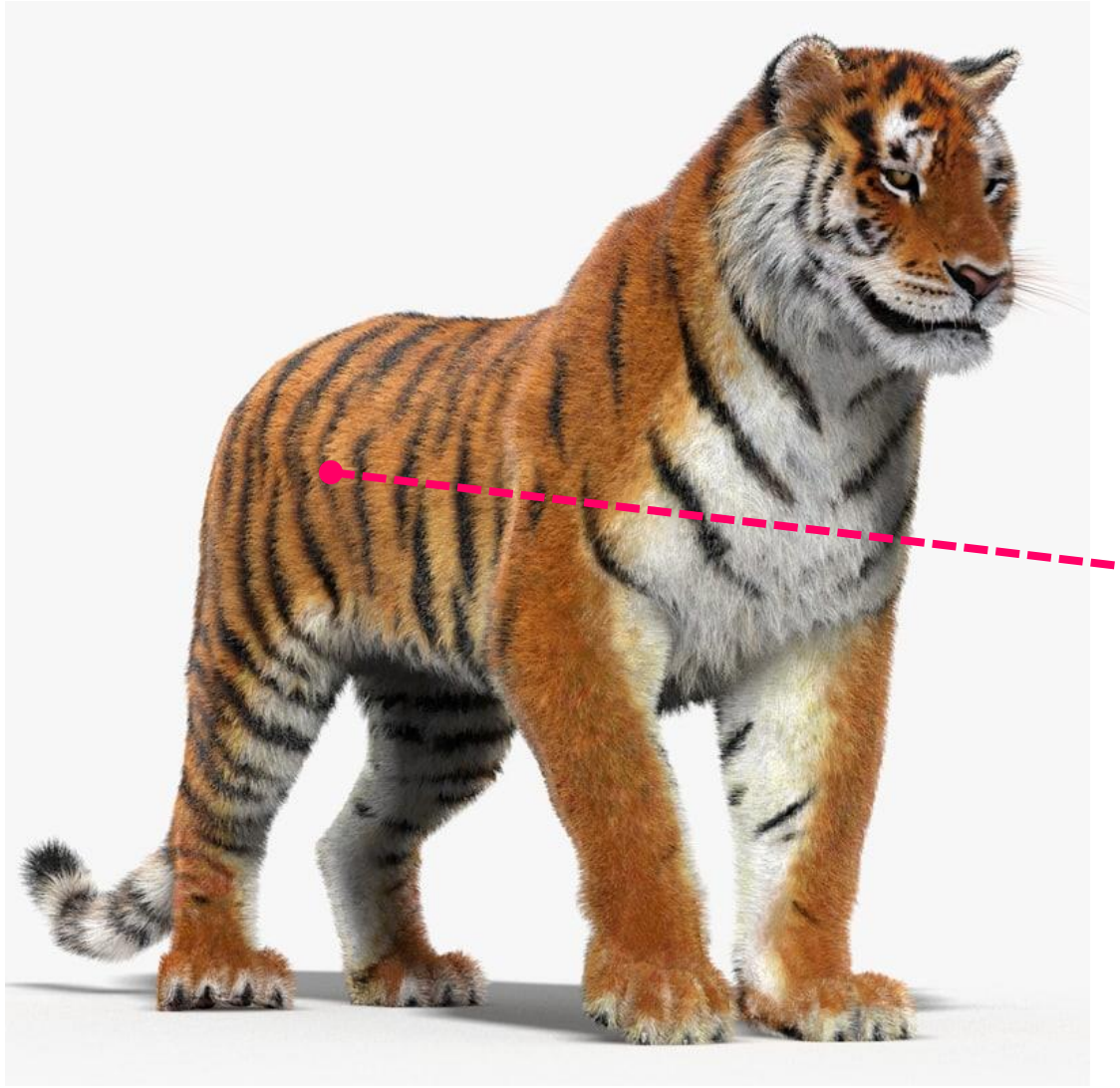
Non-rigid matching

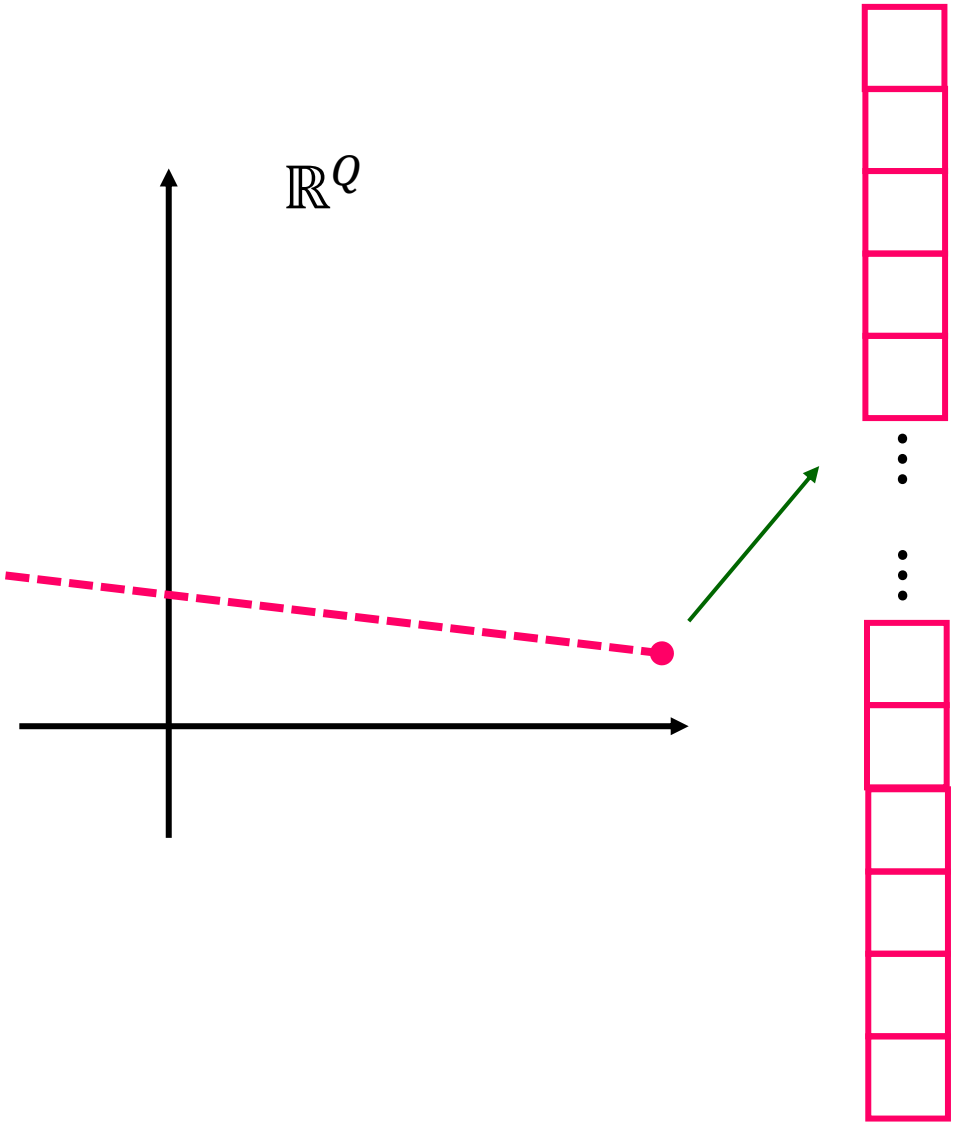


Pointwise signature

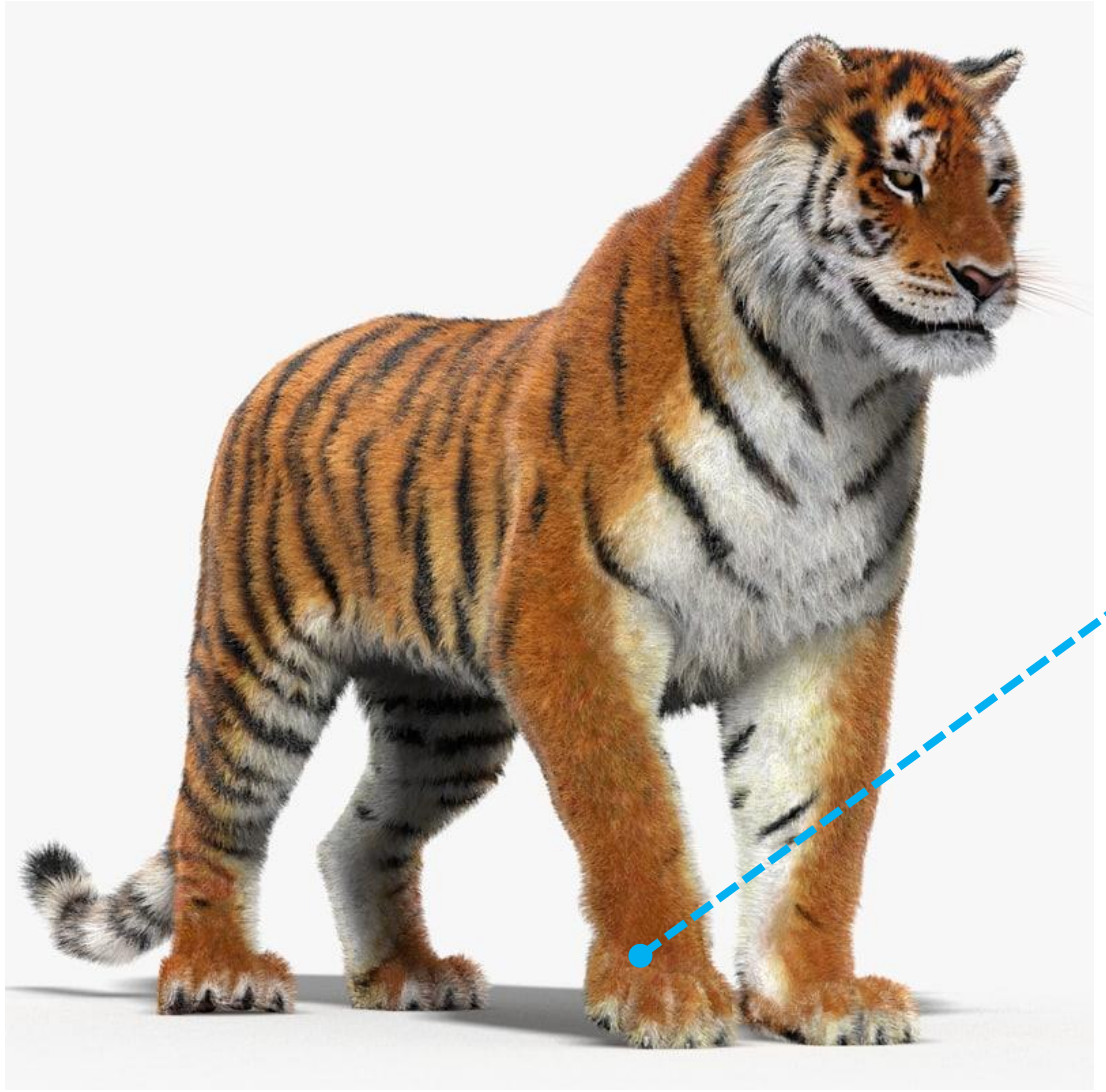


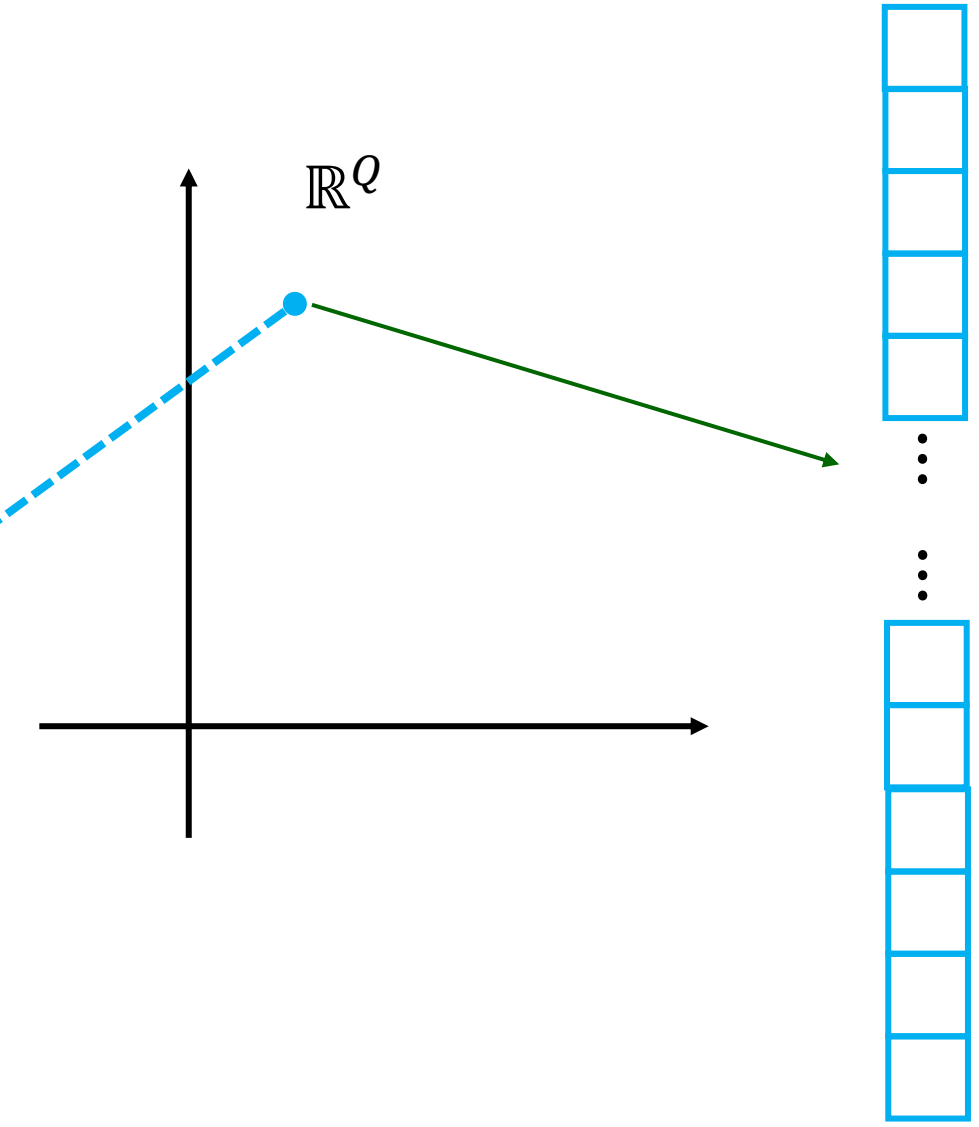
Pointwise signature



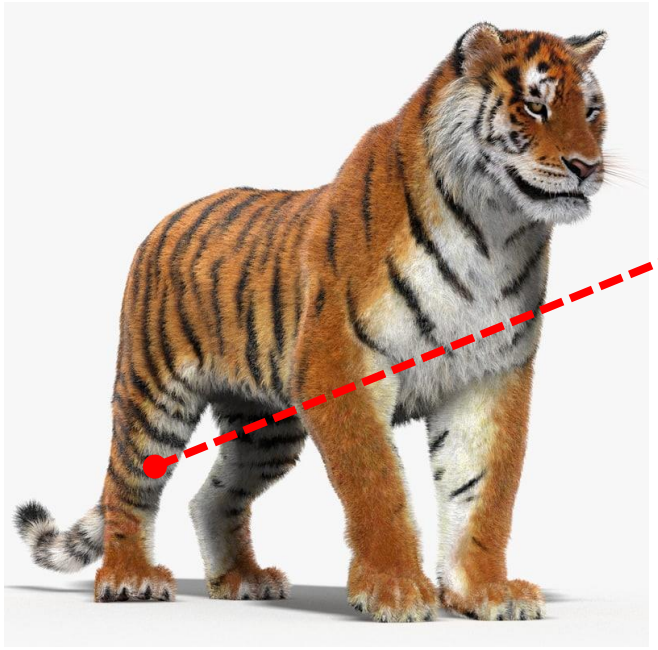


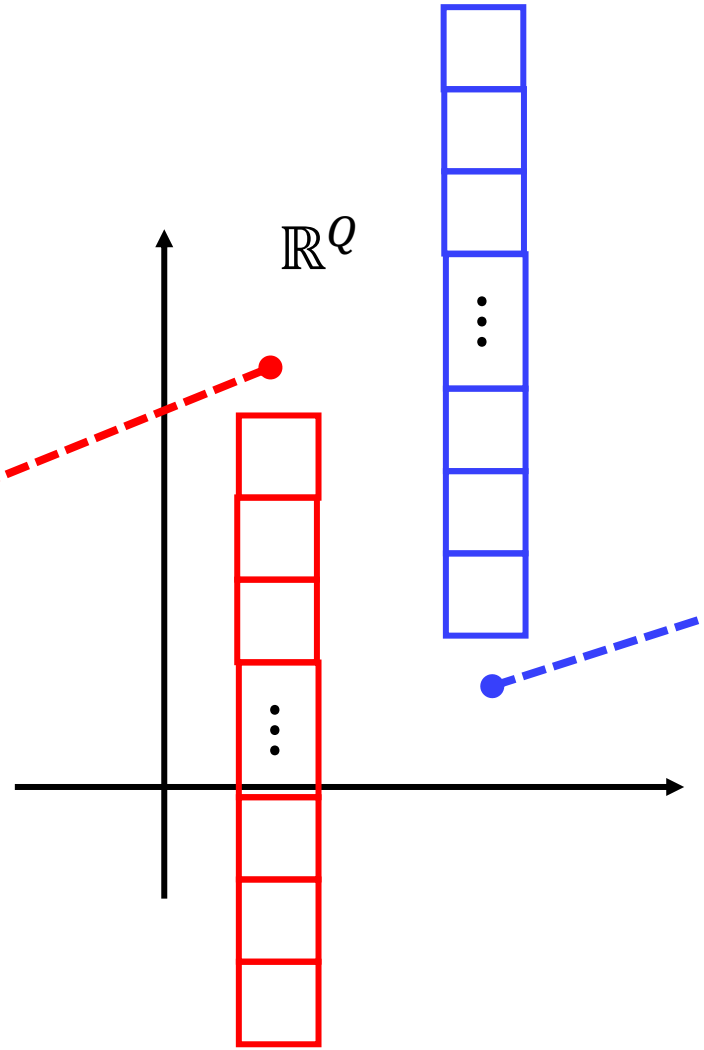
Pointwise signature



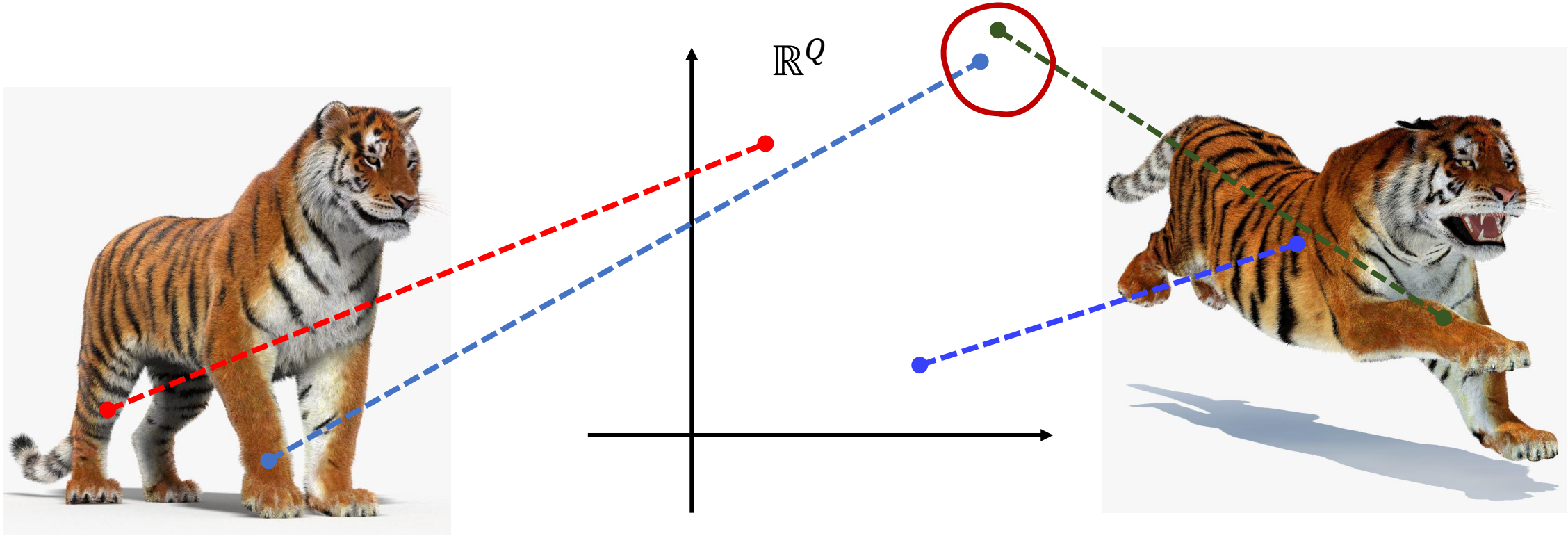


Pointwise signature



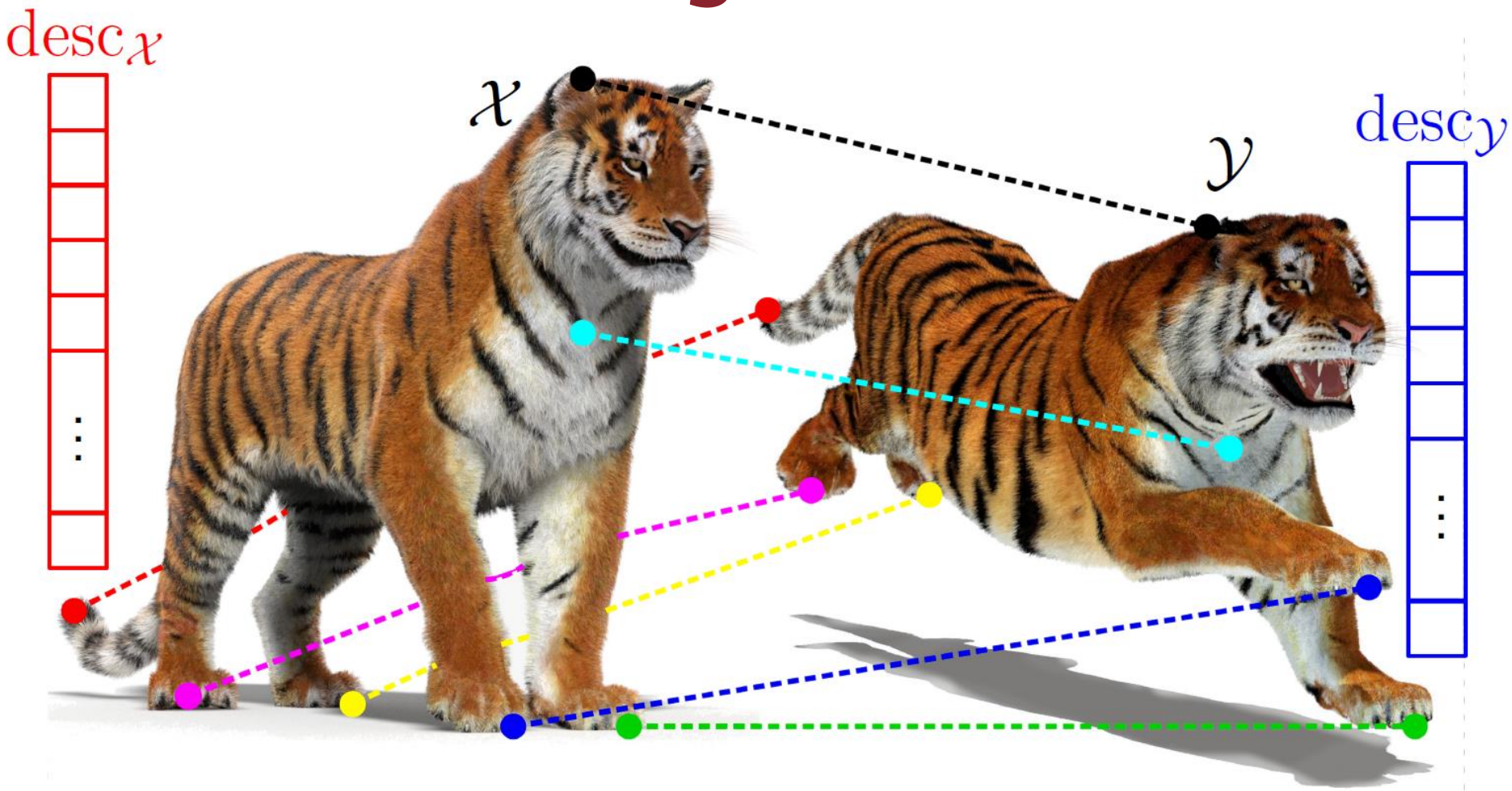


Pointwise signature



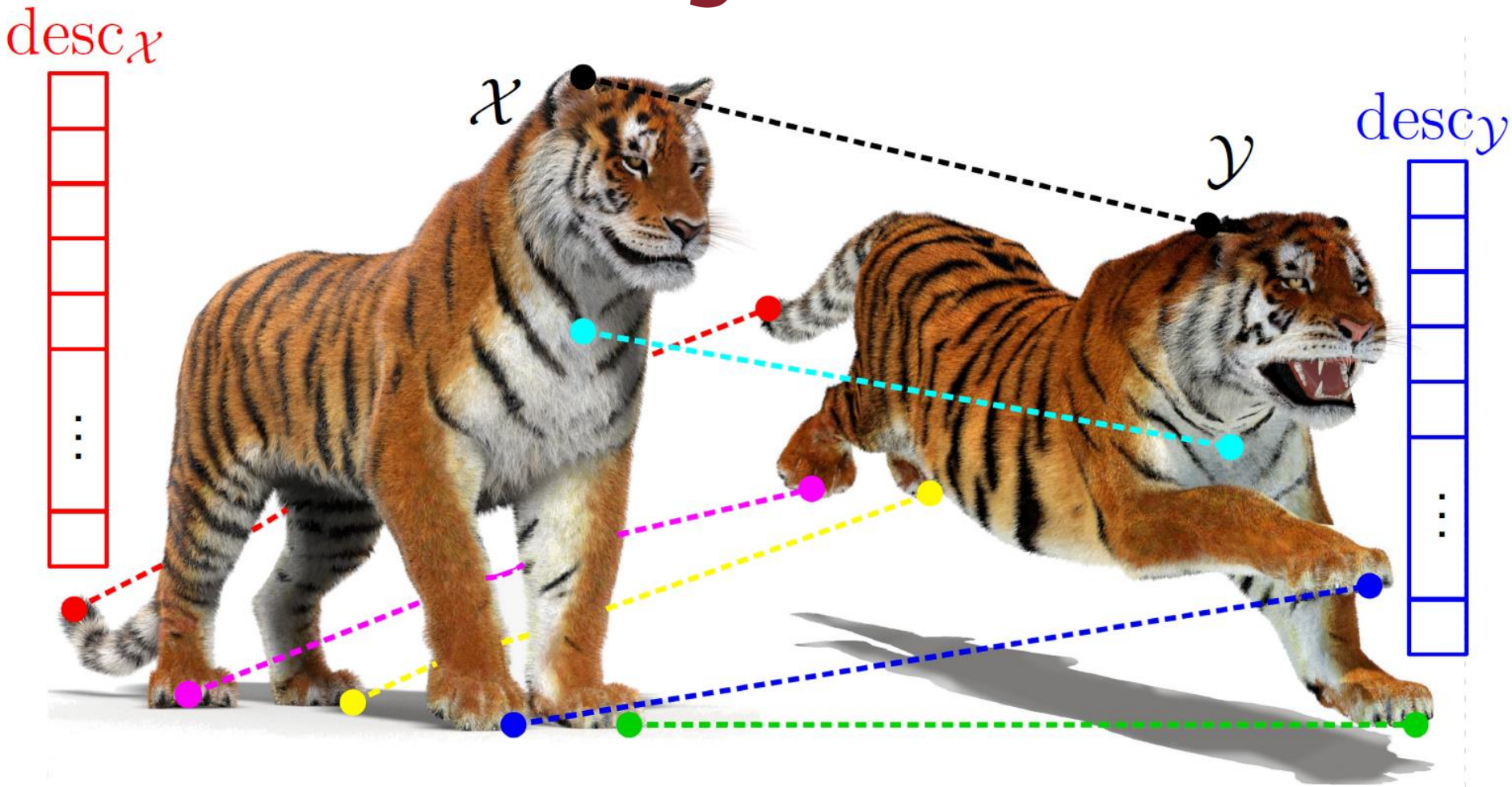
How do you suggest to find the most similar point to the orange one?

Pointwise signature



$$\text{distance} = \mathcal{D}(\mathrm{desc}_{\mathcal{X}}, \mathrm{desc}_{\mathcal{Y}}) = \|\mathrm{desc}_{\mathcal{X}} - \mathrm{desc}_{\mathcal{Y}}\|$$

Pointwise signature



$$y = \Pi(x) = \underset{y \in \mathcal{Y}}{argmin} \| desc_{\mathcal{X}}(x) - desc_{\mathcal{Y}}(y) \|$$

Desidered properties

A **descriptor (signature)** should be:

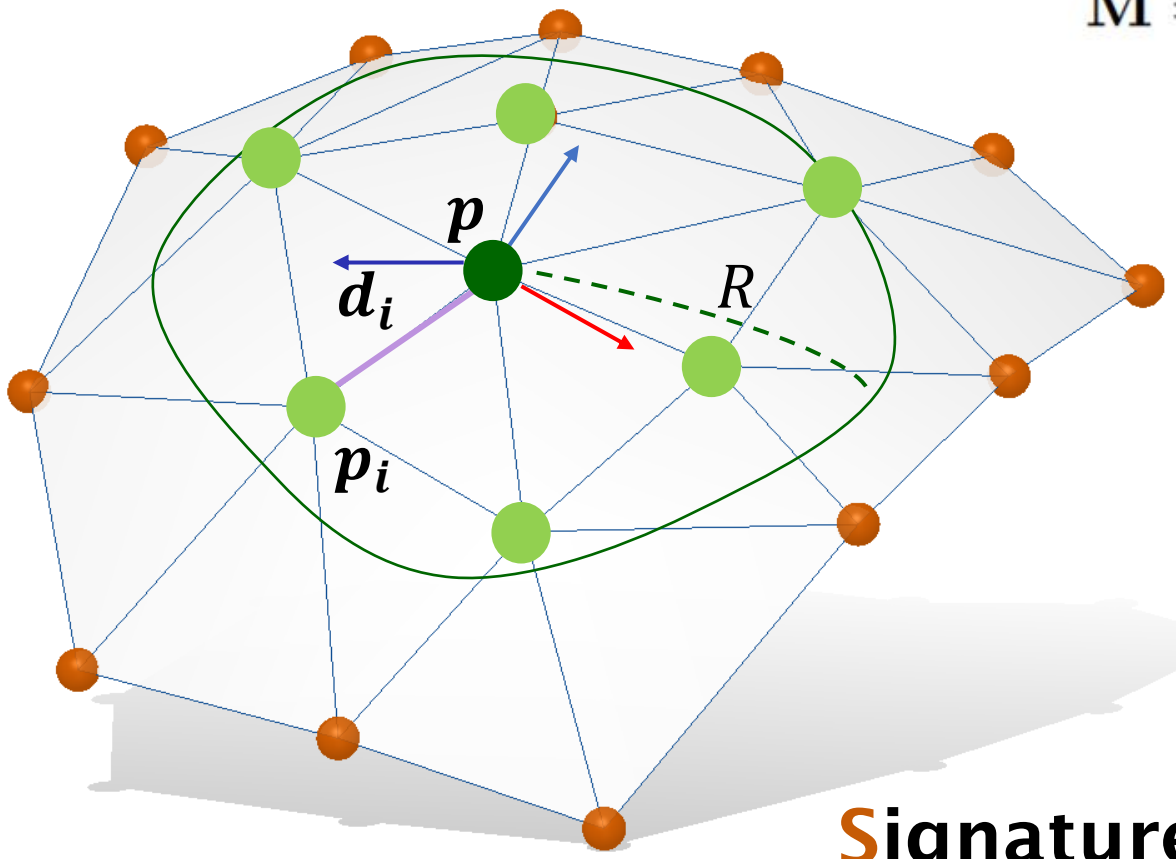
- Effective
- Concise
- Repeatable

The **properties of the descriptor** should be evaluated w.r.t. the kind of **deformations** that would be matched (**near isometric tiger deformation**)

SHOT: a first example

For all $\boldsymbol{p}$ we define the covariance matrix:

$$\mathbf{M} = \frac{1}{\sum_{i:d_i \leq R} (R - d_i)} \sum_{i:d_i \leq R} (R - d_i)(\mathbf{p}_i - \mathbf{p})(\mathbf{p}_i - \mathbf{p})^T$$



From the eigenvectors of M we obtain a LRF (x, y, z) that is then used to define:

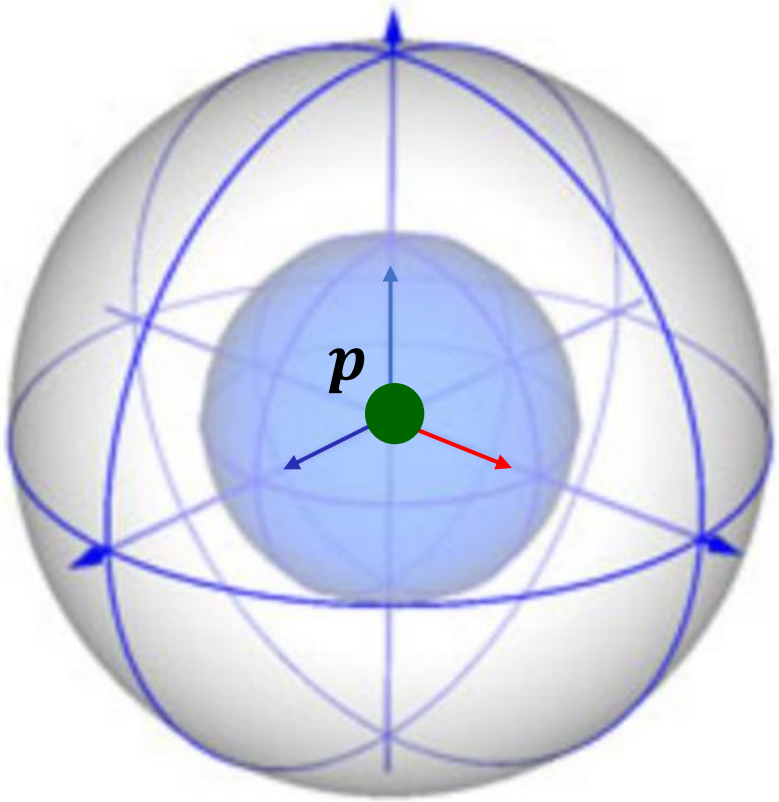
SHOT

Signature of Histograms of OrienTations

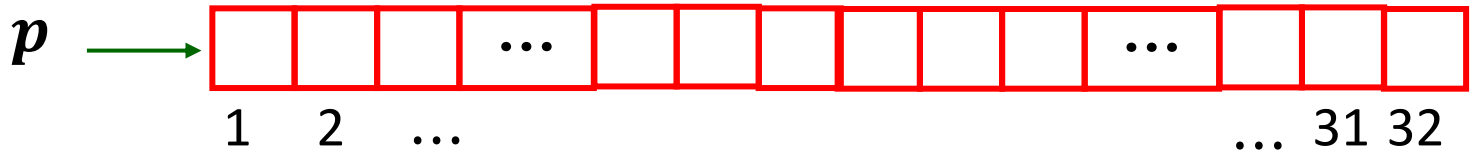
Unique Signatures of Histograms for Local Surface Description, *Tombari et al.*, ECCV 2010

SHOT: construction

Once we have the LRF for every point $\boldsymbol{p}$ we can define a **coherent 3D grid**



The 3D space around $\boldsymbol{p}$ is subdivided in 32 regions each of wich is a different bin of the histogram that describes the point.

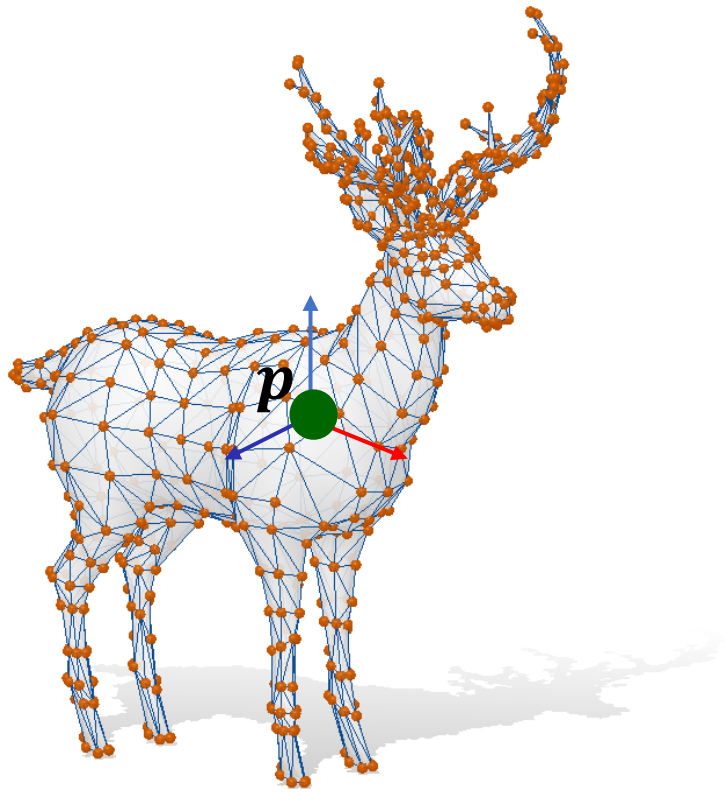


The value of each bin is a weighted sum of $cos\theta_i$ where θ_i is the angle between the normals of the point $\boldsymbol{p}$ and the point within each region of the 3D grid.

Unique Signatures of Histograms for Local Surface Description, *Tombari et al.*, ECCV 2010

SHOT: a comment

SHOT is an extrinsic descriptor: it depends on the 3D embedding of the shape



The analysis for the point $\boldsymbol{p}$ is performed looking at how the shape behaves around the point.

To obtain a coherent description of similar points and to be invariant to rigid deformations the LRF is necessary.

The SHOT descriptors is not invariant to non-rigid deformations.

Functions and signals on 3D shapes

Basis:

Given a vector space $\mathcal{V}$ a collection of vectors $B = \{b_1, b_2, \ldots, b_n\}$ is a basis for $\mathcal{V}$ iff:

1. Their vectors are all linear independent
2. They span the entire space $\mathcal{V}$

We are familiar with canonical basis for Euclidean spaces (e.g. $\mathbb{R}^3$)

$$\alpha_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{v}$$

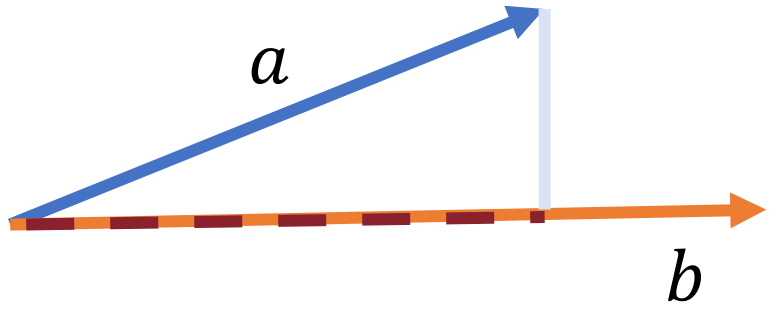
Slide credits to R. Marin

Inner product:

Given two vectors a, b in $\mathcal{V}$ the inner product is a function that gives us a scalar:

$$\langle a, b \rangle = \sum_{i=1}^{n} a_i b_i = a^T b$$

It can be seen as a projection of a on b



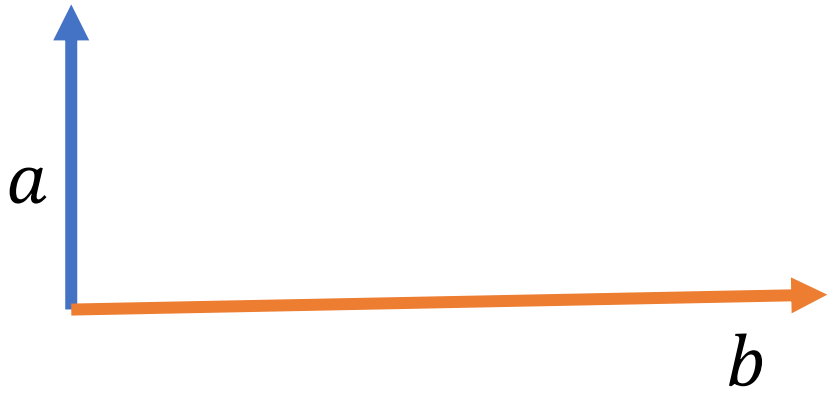
Slide credits to R. Marin

Inner product:

Given two vectors a, b in $\mathcal{V}$ the inner product is a function that gives us a scalar:

$$\langle a, b \rangle = \sum_{i=1}^{n} a_i b_i = a^T b$$

We say that 2 vectors are **orthogonal** if the projection is equal to zero.



Slide credits to R. Marin

Inner product:

Given two vectors a, b in $\mathcal{V}$ the inner product is a function that gives us a scalar:

$$\langle a, b \rangle = \sum_{i=1}^{n} a_i b_i = a^T b$$

We say that a vector has norm 1 or is normal if its inner product with itself is equal to 1

Slide credits to R. Marin

Inner product:

Given two vectors a, b in $\mathcal{V}$ the inner product is a function that gives us a scalar:

$$\langle a, b \rangle = \sum_{i=1}^{n} a_i b_i = a^T b$$

We say that a vector has norm 1 or is normal if its inner product with itself is equal to 1

We refer to the inner product of a vector with itself as its square norm (the square of its length)

$$\langle a, a \rangle = \sum_{i=1}^{n} a_i a_i = \sum_{i=1}^{n} {a_i}^2 = a^T a = \|a\|^2$$

Slide credits to R. Marin

Basis representation:

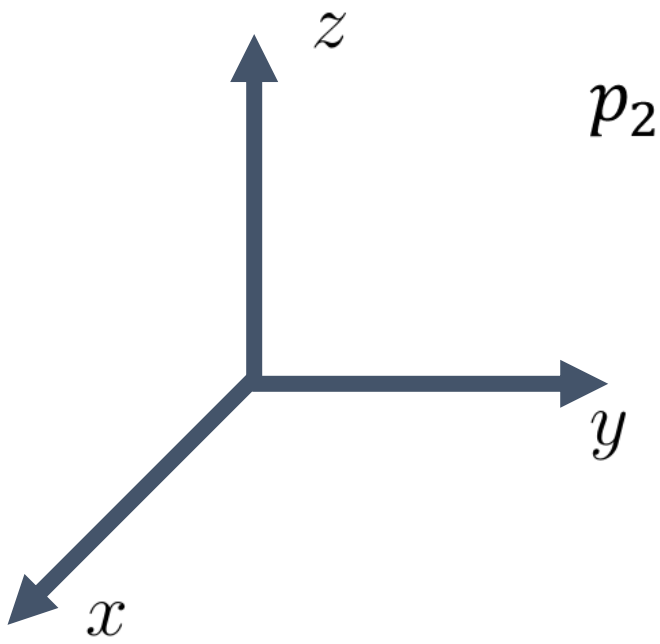
If we have a basis $B = \{b_1, b_2, \dots, b_n\}$ for a vector space $\mathcal{V}$ we can compute the representation in that basis for every vector $a \in \mathcal{V}$

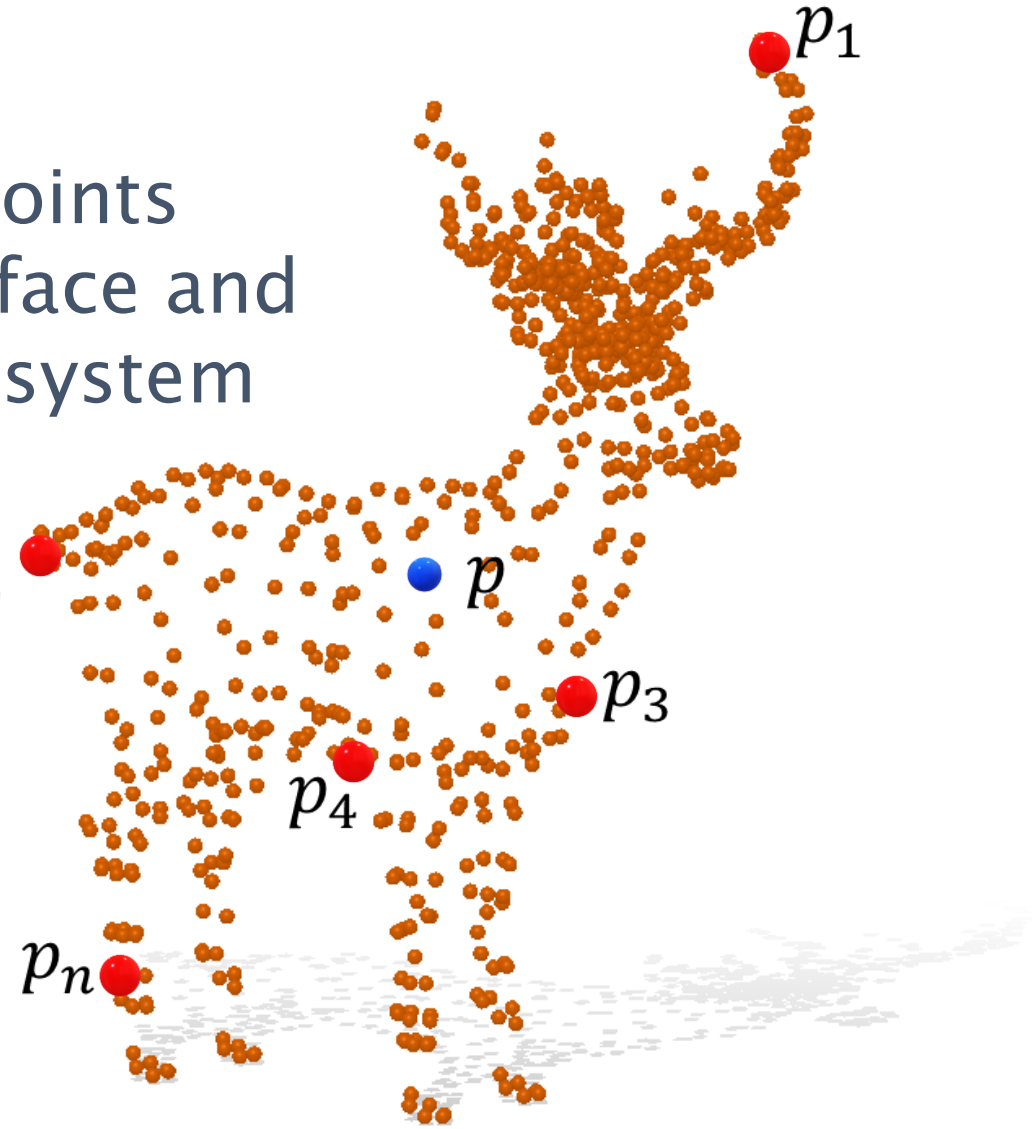
If we only consider a subset of our basis $B = \{b_1, b_2, \dots, b_n\}$ for the vector space $\mathcal{V}$ we can only represent a subspace of $\mathcal{V}$

Slide credits to R. Marin

A point of view:

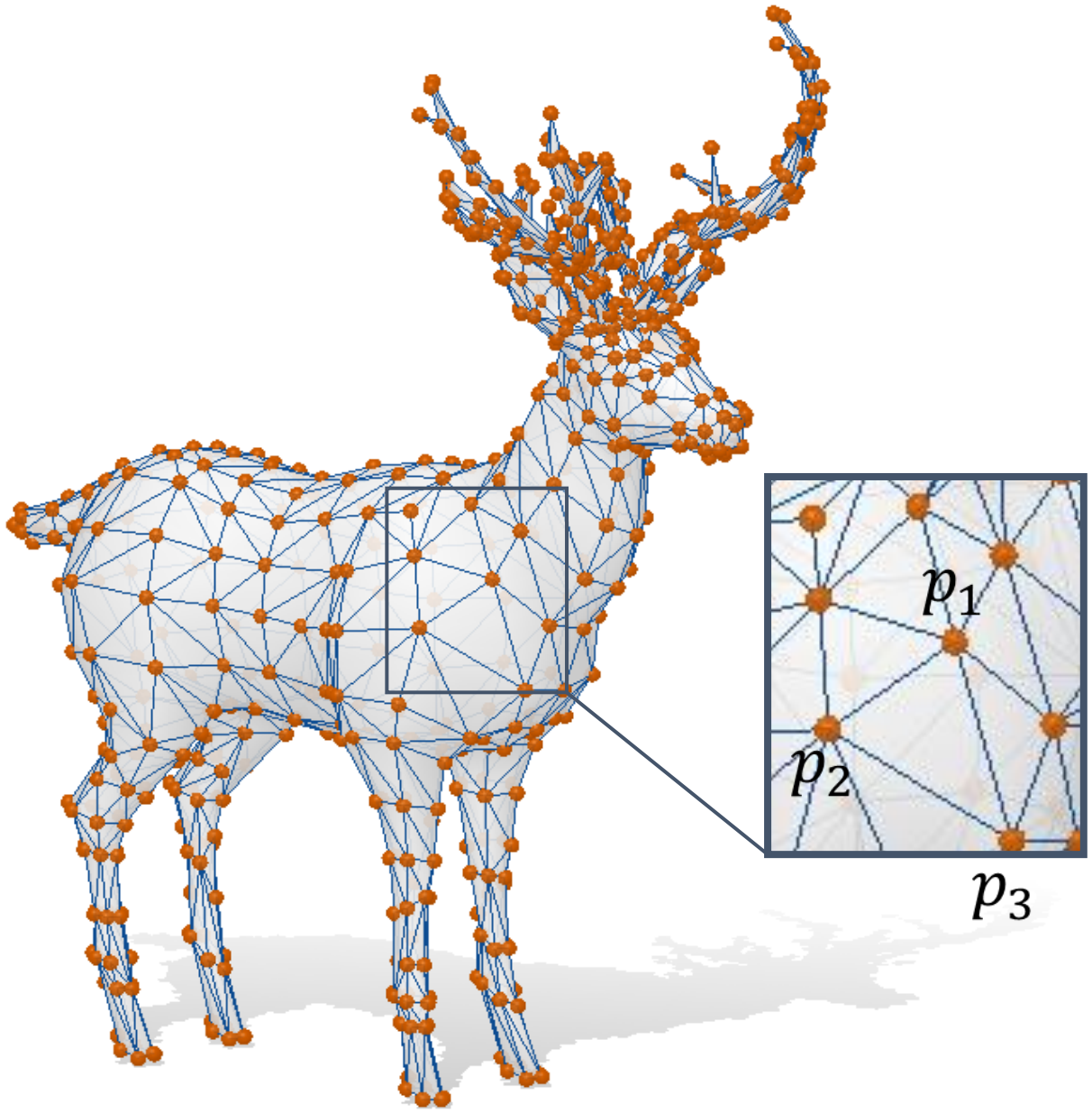
Given a set of n points sampled on a surface and a fixed reference system





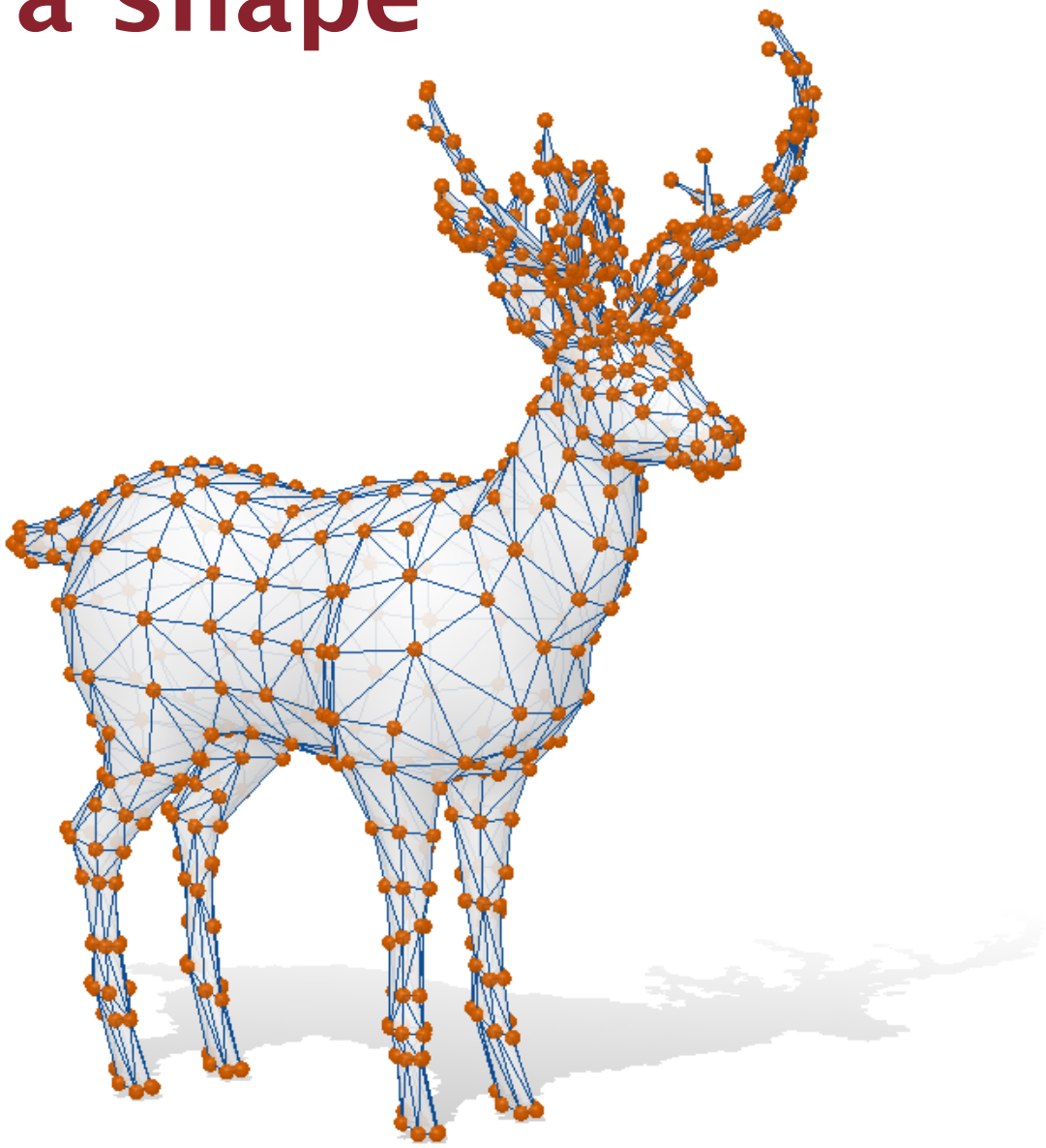
We can order the points $p_1, p_2, p_3, p_4, \cdots, p_n$

Triangular meshes

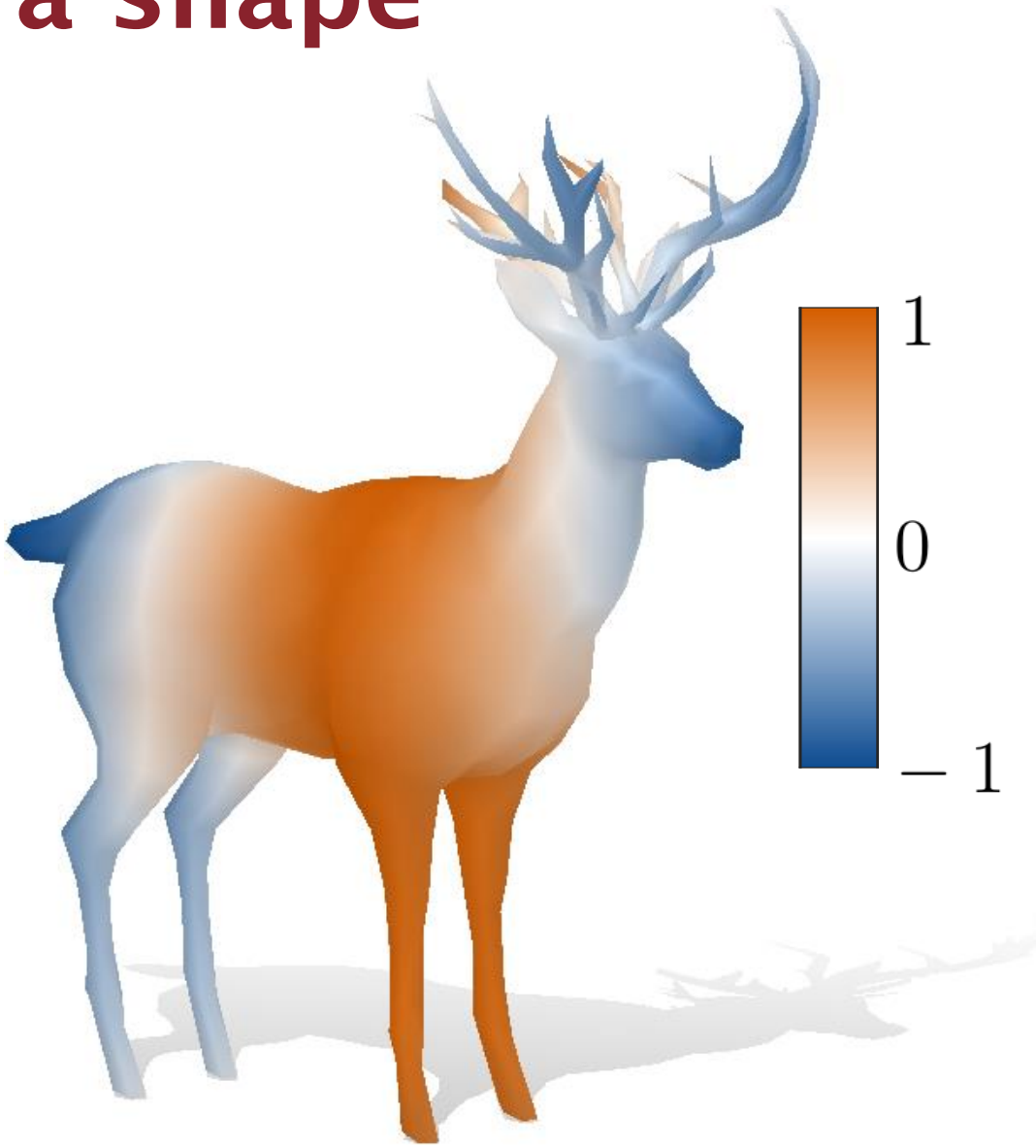


What is a function on a shape

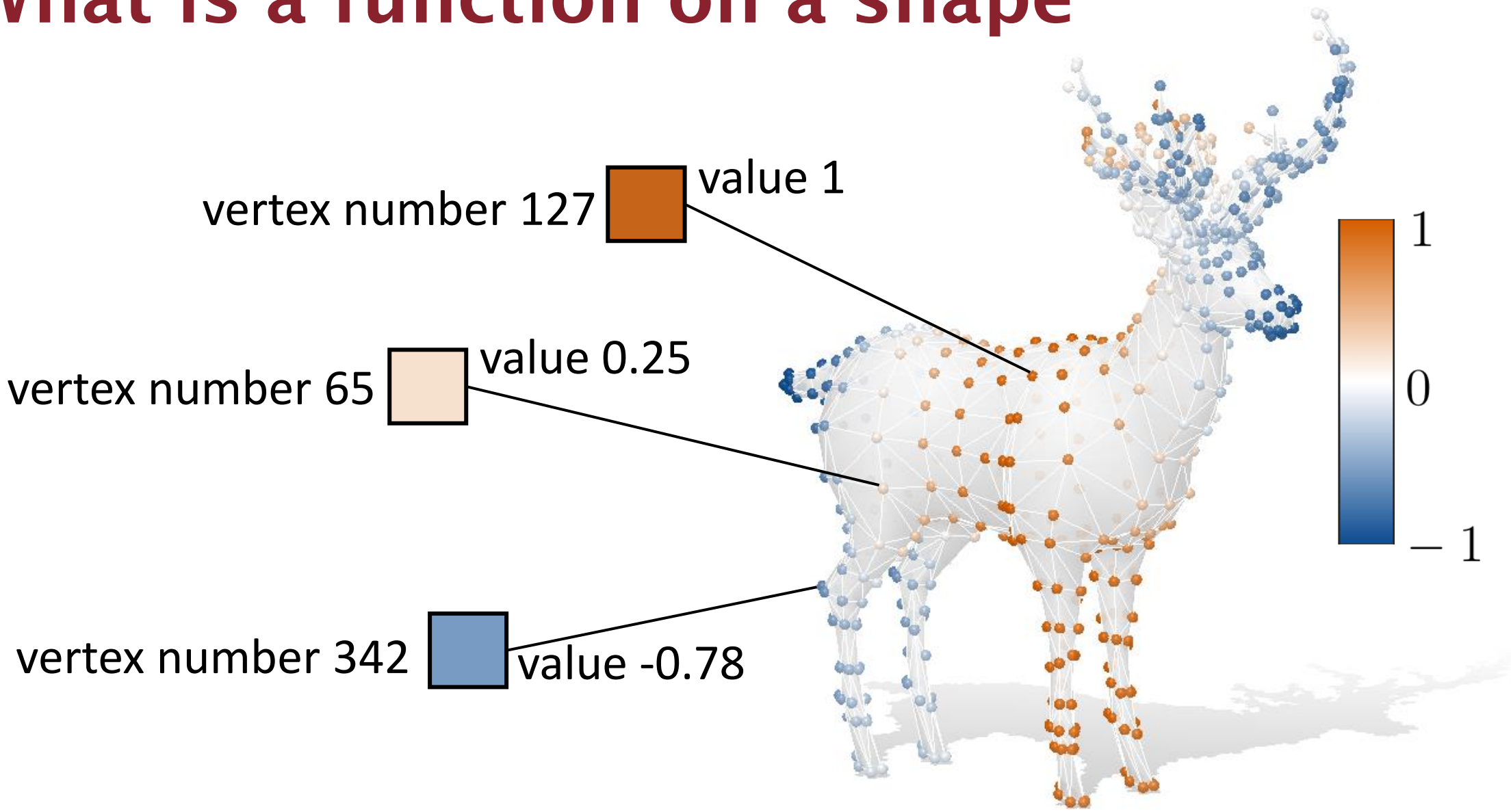
?



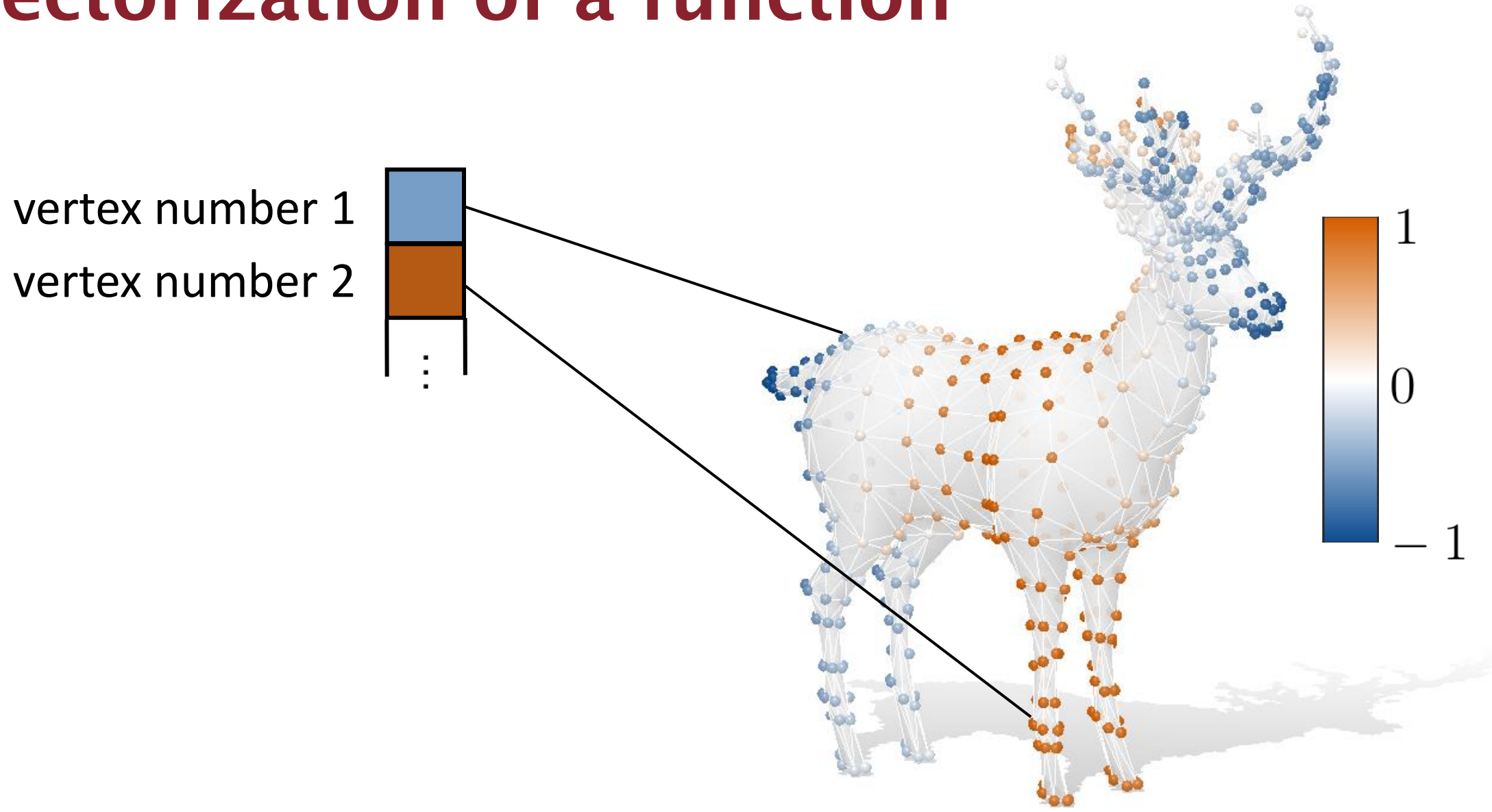
What is a function on a shape



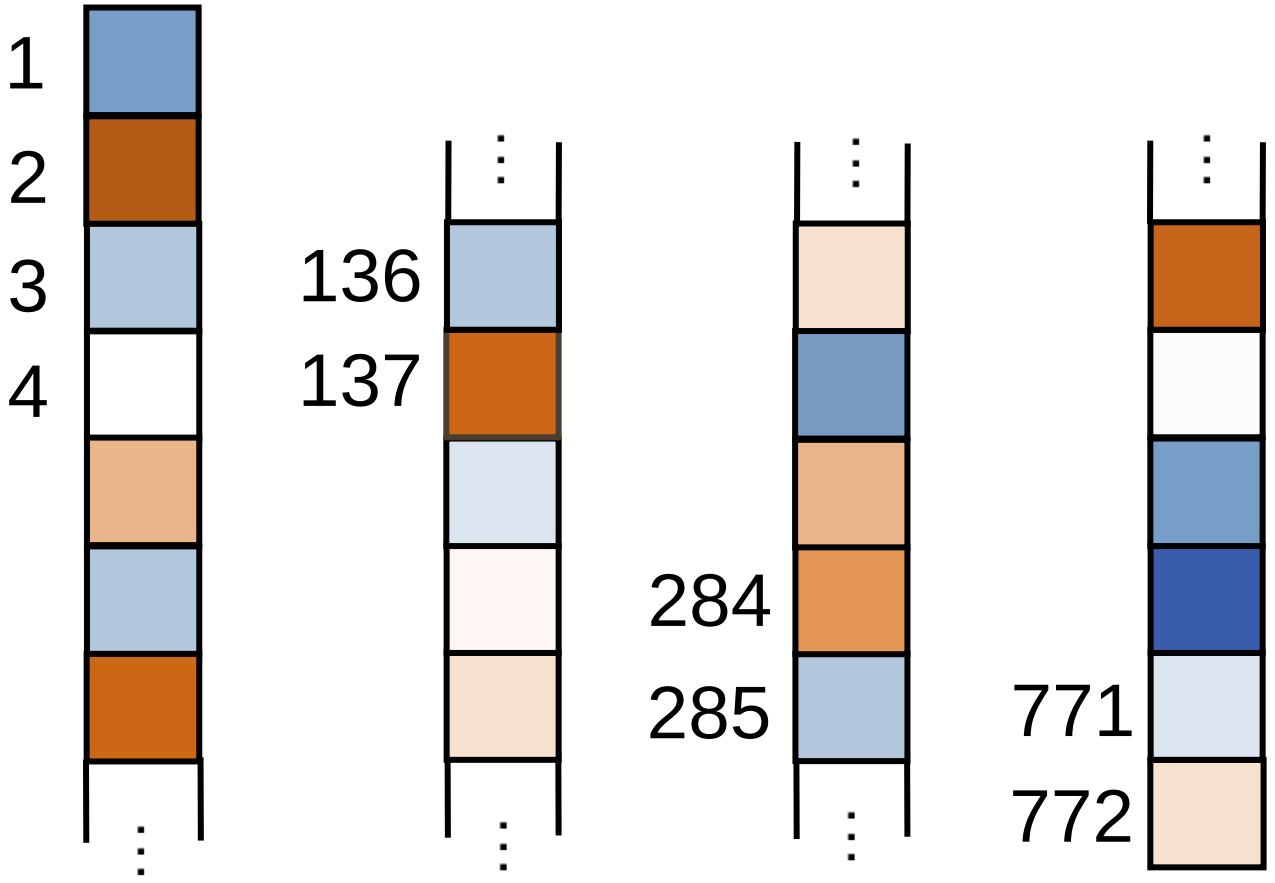
What is a function on a shape

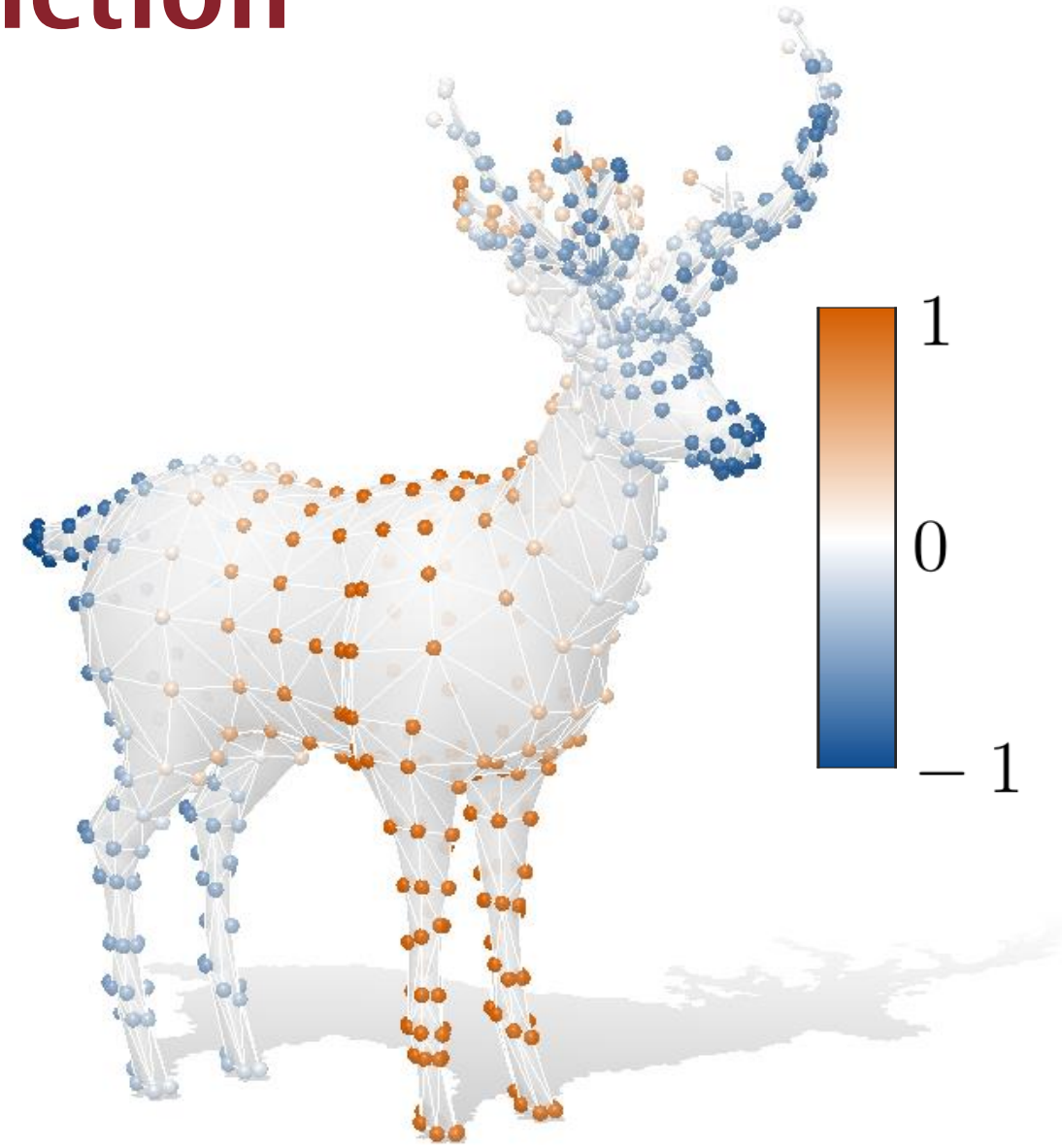


Vectorization of a function

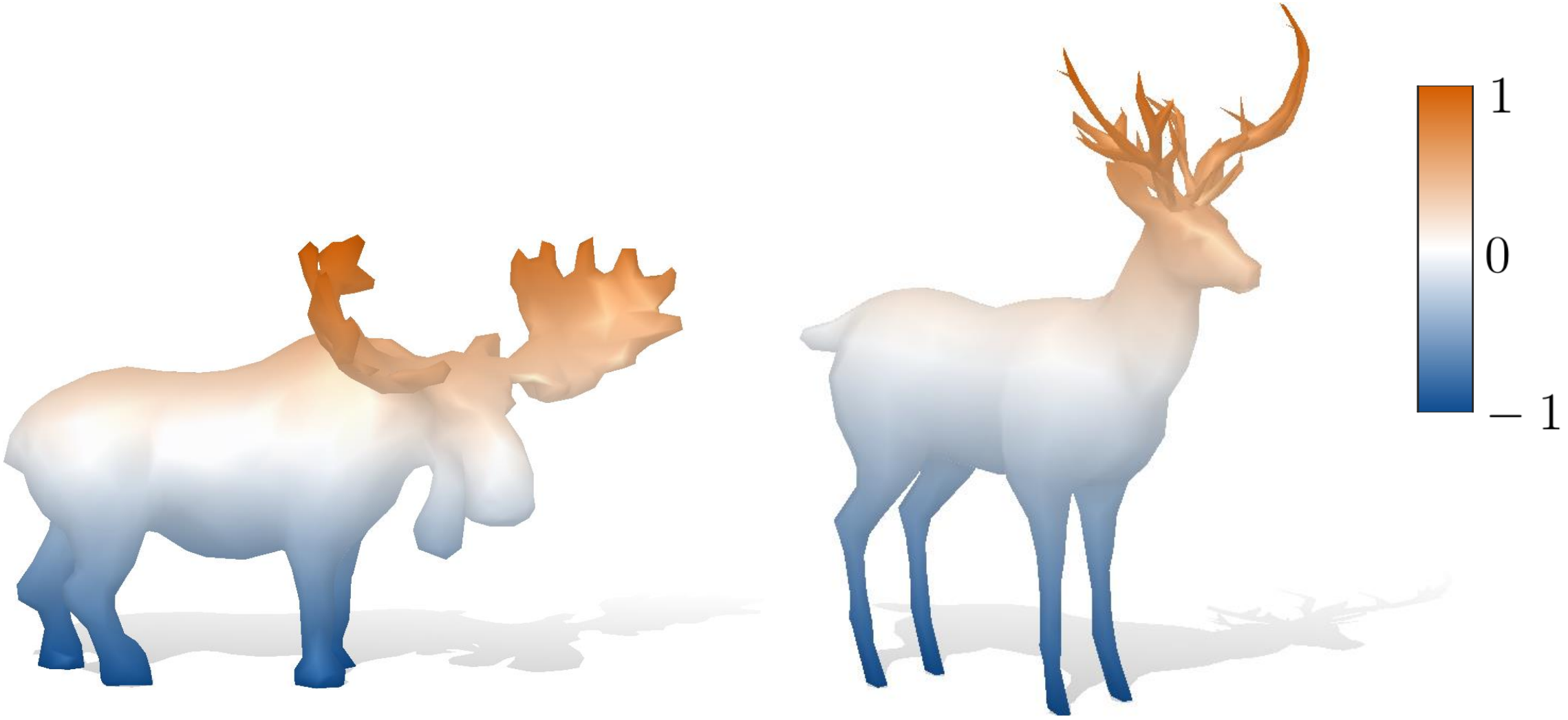


Vectorization of a function

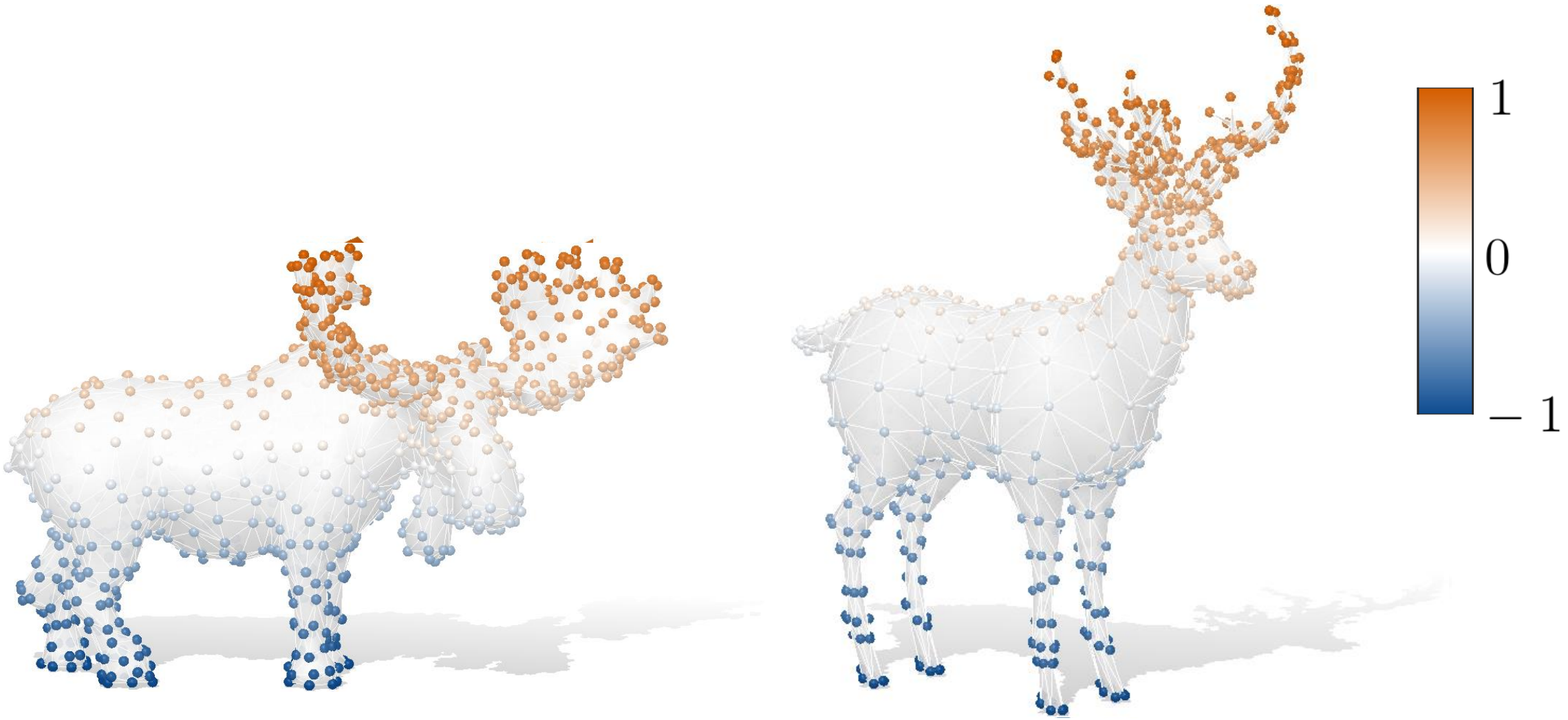




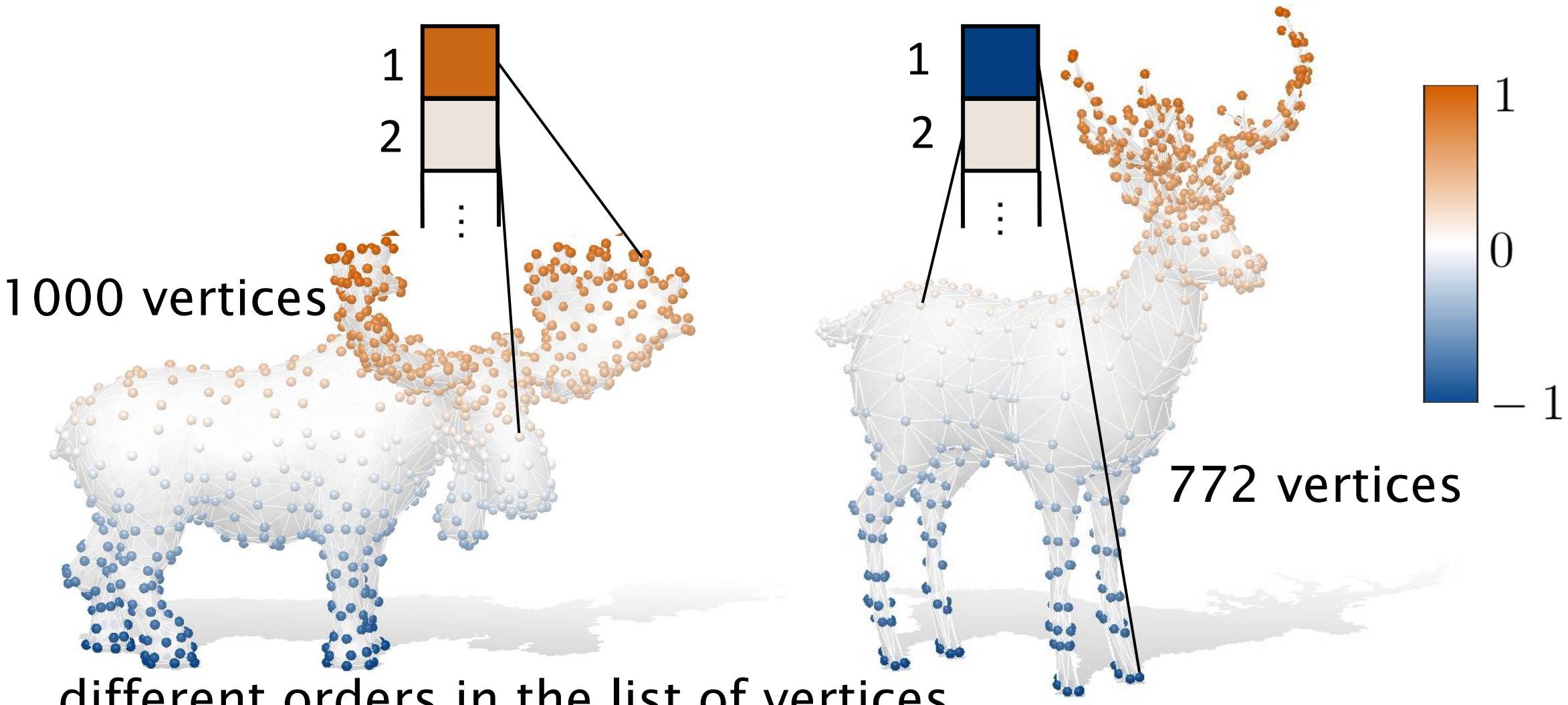
Functions on different domains



Not a common template



Not a common template

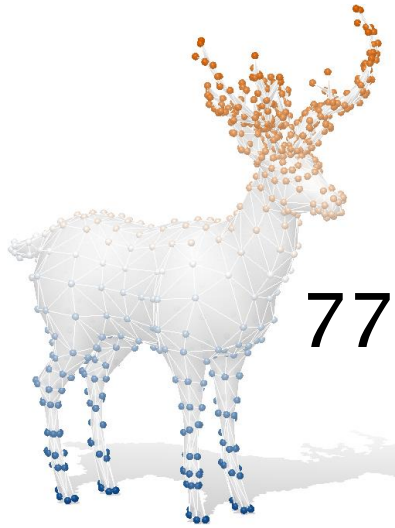


different orders in the list of vertices

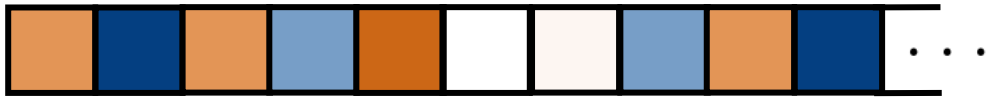
Same function – different vectors

1 2 3 4 5 6

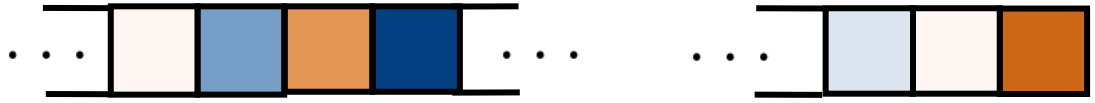
768 770 772



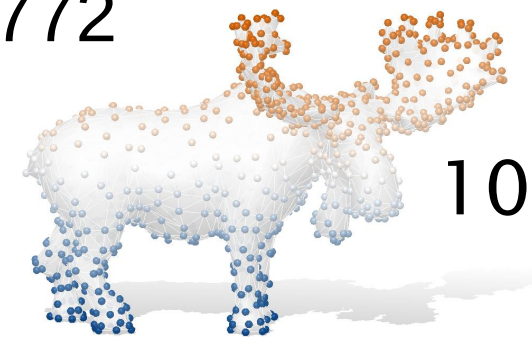
772 vertices



1 2 3 4 5 6 7 8 9 10



770 772 998 1000



1000 vertices

These representations are not comparable!

Functions on a shape:

Given a shape $\mathcal{M}$, represented as a mesh

we consider a function f defined on $\mathcal{M}$

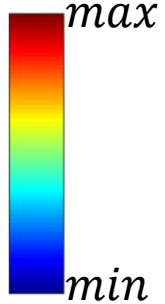
And the set $F(\mathcal{M}, \mathbb{R})$ of all the real-valued functions defined on $\mathcal{M}$.

What is a function defined on the discrete representation of $\mathcal{M}$?

Sum of functions

+

=

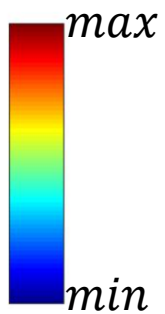


Sum of functions

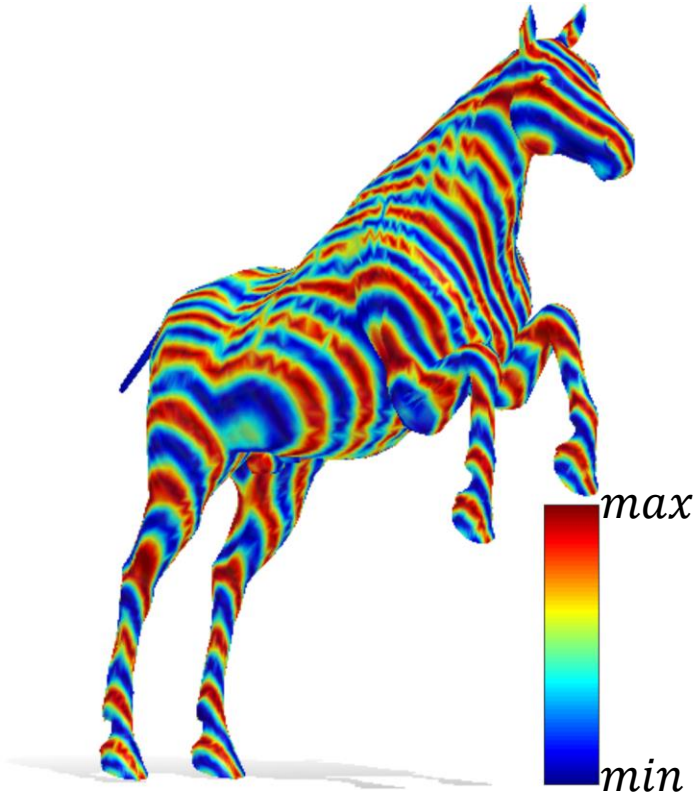
Scalar product

$5 \cdot$

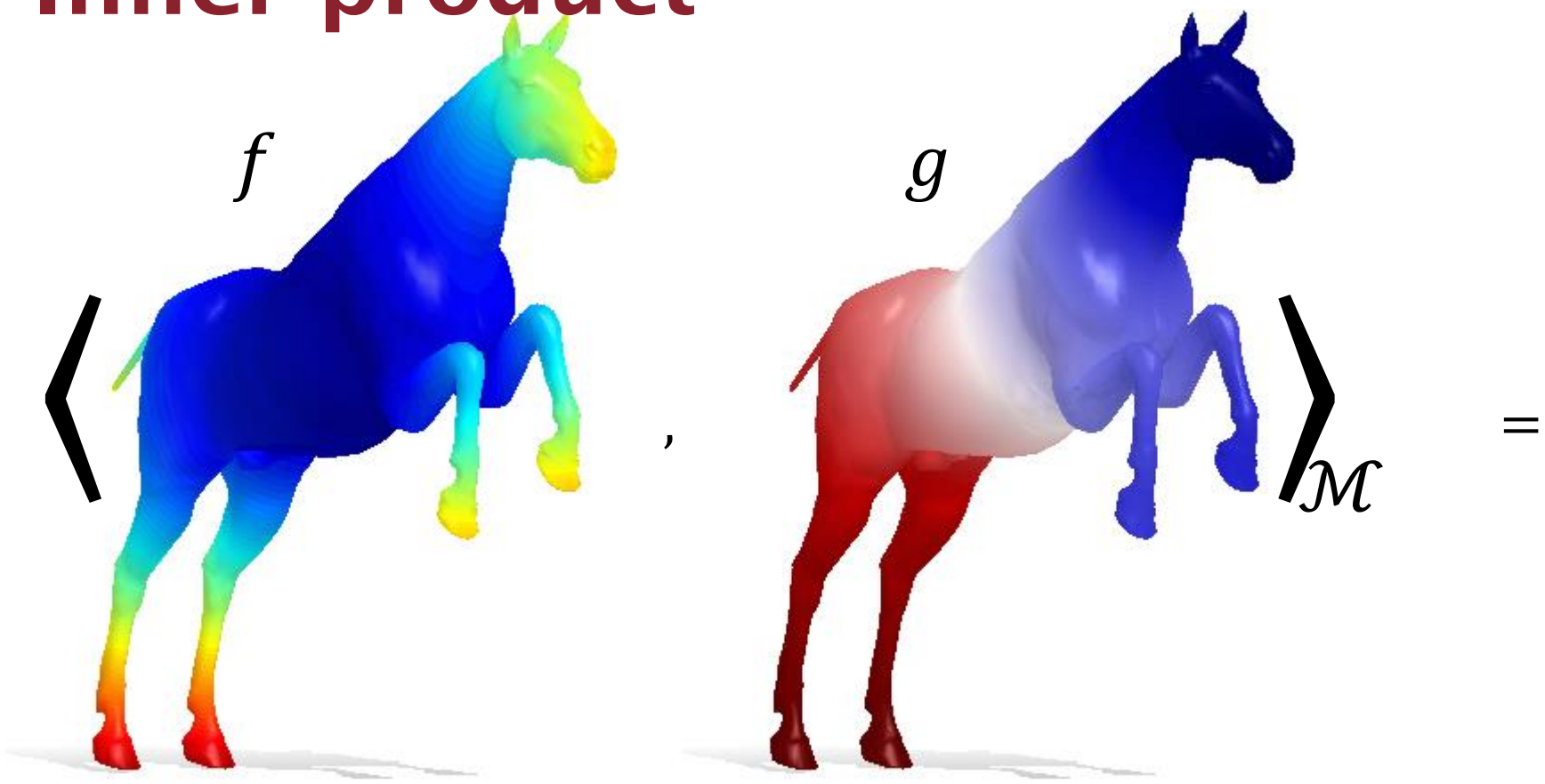
$=$



Scalar product

$5 \cdot$  $=$ 

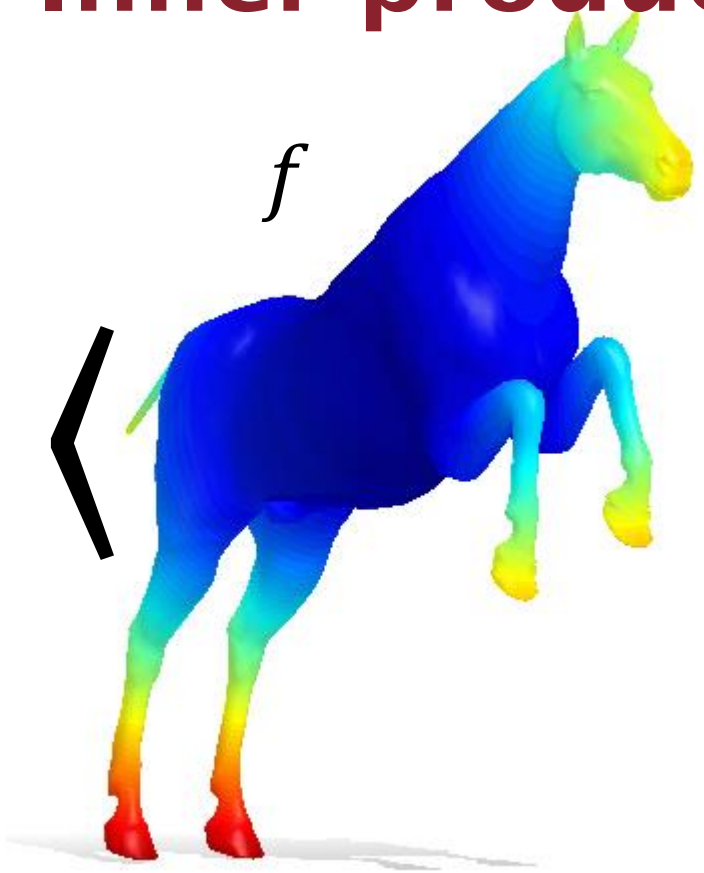
Inner product

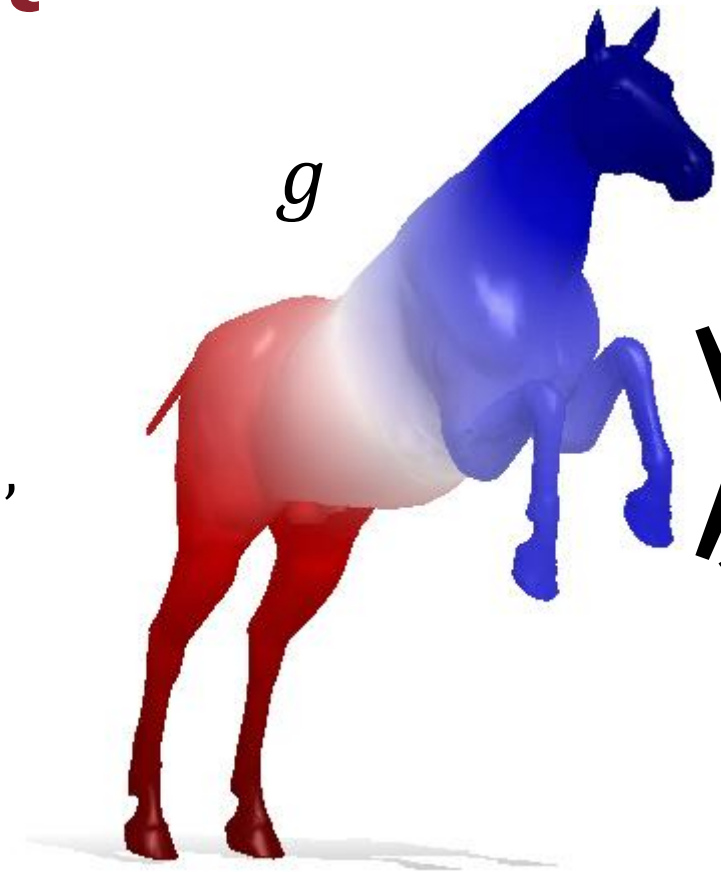


$$\langle \cdot, \cdot \rangle_{\mathcal{M}} : F(\mathcal{M}, \mathbb{R}) \times F(\mathcal{M}, \mathbb{R}) \longrightarrow \mathbb{R}$$

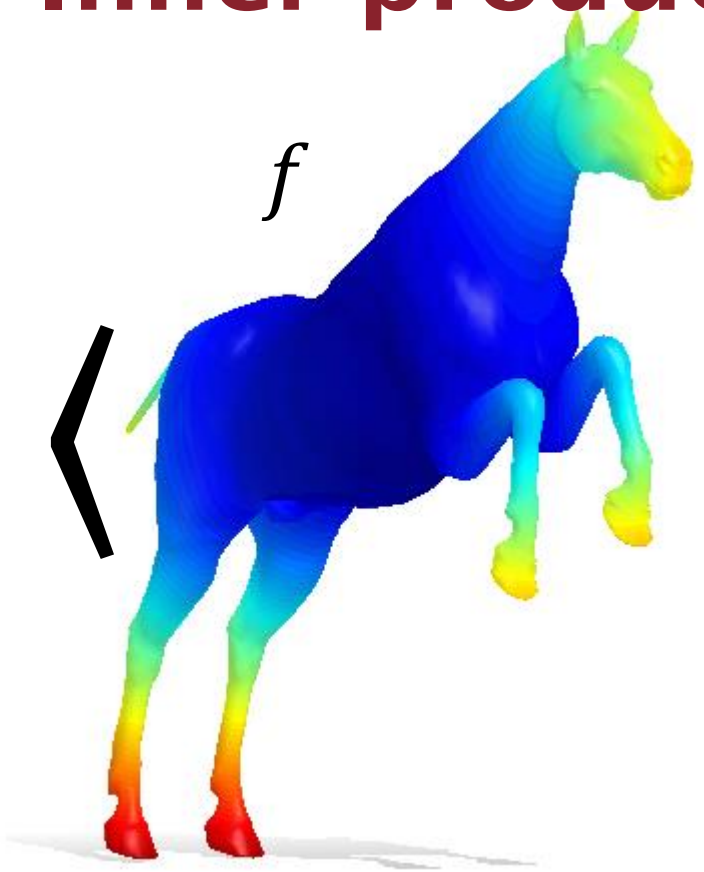
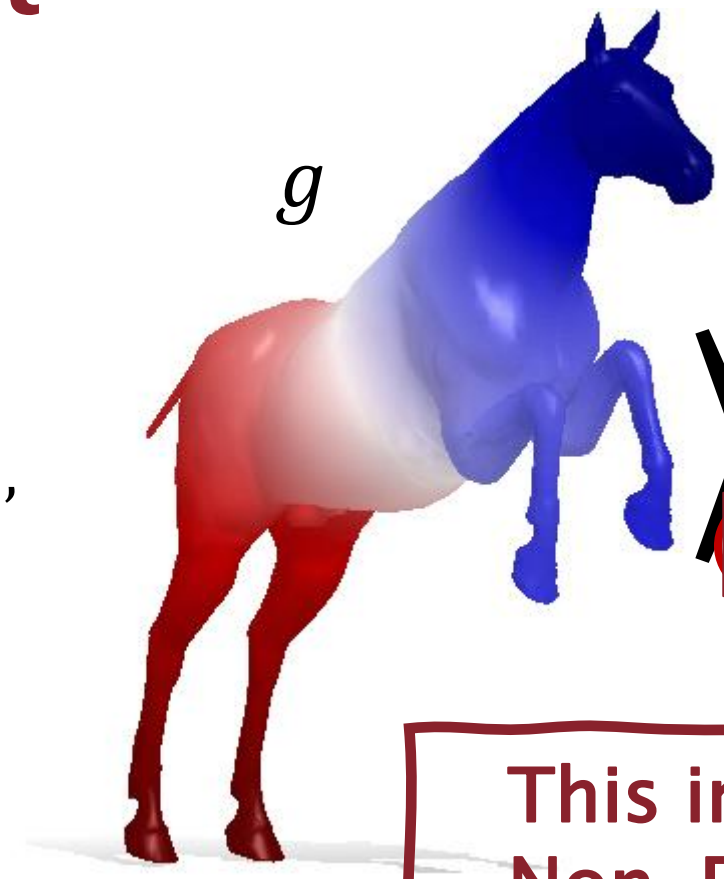
Inner product

$$\langle f, g \rangle_{\mathcal{M}} = 3,8953$$





Inner product

$\langle f$  $,$  $g \rangle_{\mathcal{M}} = 3,8953$

This inner product is on a Non-Euclidean domain so the METRIC is DIFFERENT

$$\langle f, g \rangle_{\mathcal{M}} = \int_{\mathcal{M}} f(x) g(x) d\mu(x)$$

Spectral Geometry Processing

Spectral:

- Proposed at the beginning of 90's (as me)
- Transfer the signal processing toolbox on 3D shapes

The idea in ONE sentence
Exploit the eigendecomposition (eigenvalues, eigenvectors …) of specific geometric operators to process 3D shapes

Levy and Zhang: «Spectral Mesh processing», 2009.

The Laplace Beltrami Operator:

- is an extension of the Euclidean Laplacian
- approximates the sum of the second partial derivatives
- is defined as: $\Delta f = div(\nabla f)$
- is invariant to isometries (maps that preserve geodesics)

The LBO umbrella:

The LBO as swiss-knife:

$\Delta =$

LBO eigendecomposition:

Definition: the LBO is given by the divergence of the gradient $\Delta f = div(\nabla f)$

Is the symmetric positive semi-definite operator:

$$\Delta: \mathcal{C}^\infty(\mathcal{X}) \to \mathcal{C}^\infty(\mathcal{X})$$

It admits a countable set of eigenfunctions with non negative associated eigenvalues

$$\lambda_l \varphi_l = \Delta \varphi_l$$

Slide credits to M. Ovsjanikov

Laplacian operators:

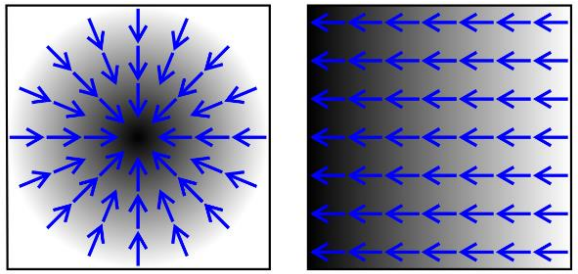
We would have a specific analysis of Laplacian

1. Laplacian on Euclidean space
2. Laplacian on Graph (discrete Laplacian)
3. An example of use (spectral clustering)
4. Laplace Beltrami Operator (LBO)
5. Applications

Gradient, divergence, Laplacian:

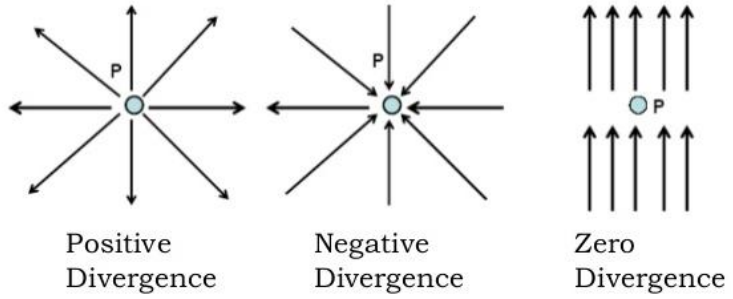
The gradient

$$\nabla : \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \ldots\right)$$

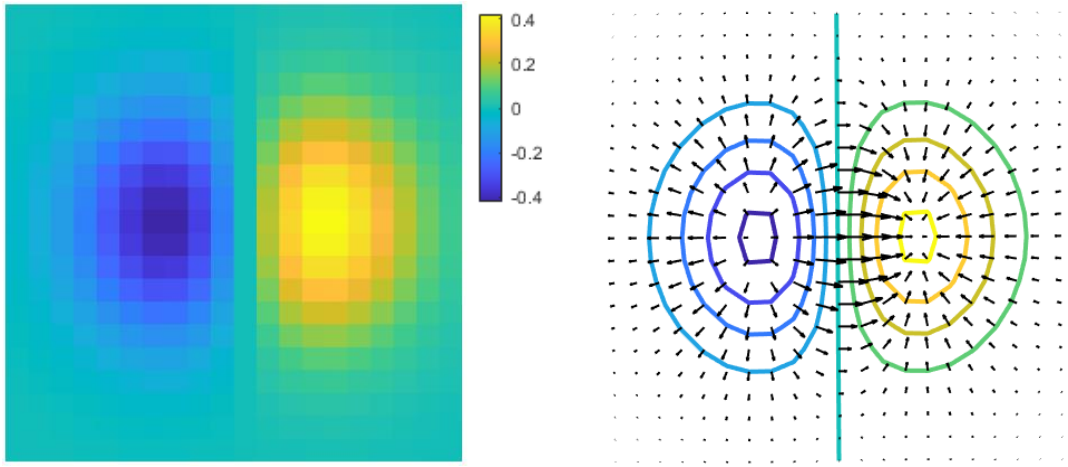


The divergence

$$div : \frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z} + \ldots.$$



The Laplacian $\Delta : div\, \nabla = \frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 y} + \frac{\partial^2}{\partial^2 z} + \ldots$



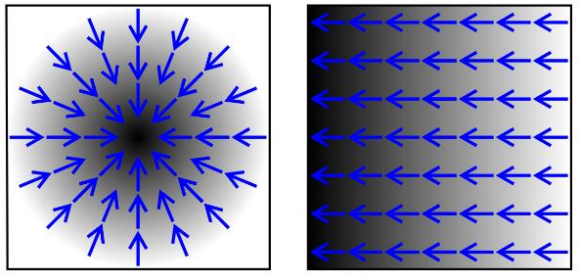
The change of
a rate of change

Slide credits to R. Marin

Gradient, divergence, Laplacian:

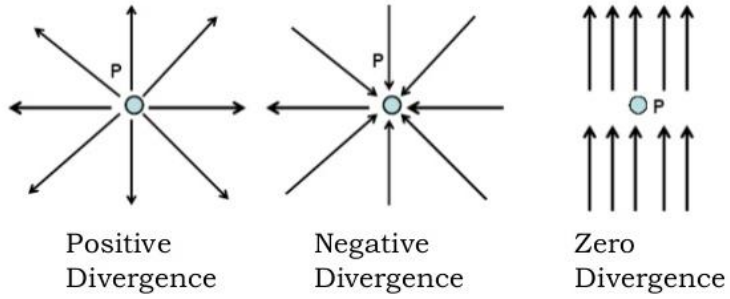
The gradient

$$\nabla : \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \dots\right)$$

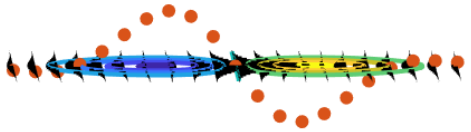


The divergence

$$div : \frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z} + \dots.$$



The Laplacian $\Delta : div\, \nabla = \frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 y} + \frac{\partial^2}{\partial^2 z} + \dots$



Note that in 1D, the Laplacian is equal to the second order derivative.

Slide credits to R. Marin

Euclidean Laplacian:

Given a function $f: \mathbb{R}^n \to \mathbb{R}^n$

Given a function $\Delta f = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2} f$

Examples:

$f(x) = \cos(x)\,,\ \Delta f = f''(x) = -\cos(x)$

$$f(x_1, x_2) = \exp(2x_1 + 7x_2)\,,$$

$$\Delta f = \frac{\partial^2}{\partial x_1^2} f + \frac{\partial^2}{\partial x_2^2} f = 4f + 49f = 54f$$

Slide credits to M. Ovsjanikov

The Laplace Beltrami Operator:

- is an extension of the Euclidean Laplacian
- approximates the sum of the second partial derivatives
- is defined as: $\Delta f = div(\nabla f)$
- is invariant to isometries (maps that preserve geodesics)

LBO continuous setting:

Definition: Given a surface $\mathcal{X}$ (eventually with boundary), we define the LBO as the unique operator given by the divergence of the gradient.

$$\Delta f = div(\nabla f)$$

As operator it maps smooth function defined on $\mathcal{X}$ to smooth function defined on $\mathcal{X}$, i.e.:

$$\Delta\colon \mathcal{C}^{\infty}(\mathcal{X}) \to \mathcal{C}^{\infty}(\mathcal{X})$$

Why Laplacian:

Suppose that we want to compute a orhonormal basis to represent a small subest of the function defined on $\mathcal{X}$

And we want the basis that minimize the Dirichlet energy (i.e. the one that has the smallest variation on the surface)

$$\langle f, \Delta f \rangle_{\mathcal{X}} = \int_{\mathcal{X}} f \Delta f \, d\mu = E(f),$$

This problem is related with the Laplacian and in particular with its eigendecomposition

Discrete Laplacian:

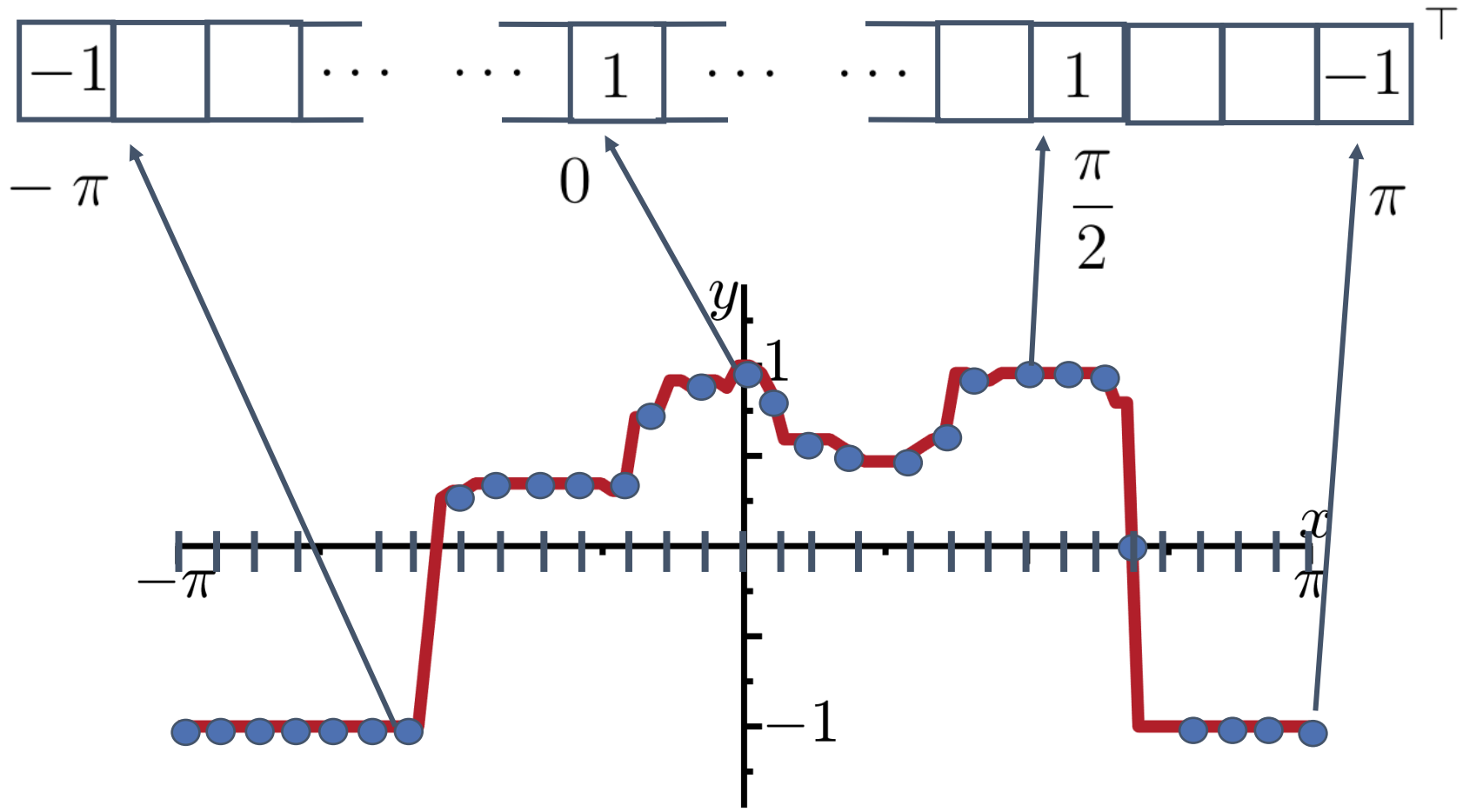
We know that we need to discretize everything for **Digital** representation

The Laplacian can be discretized as a **square Matrix**

Graphs are a good setting to start with discretization fo the Laplacian

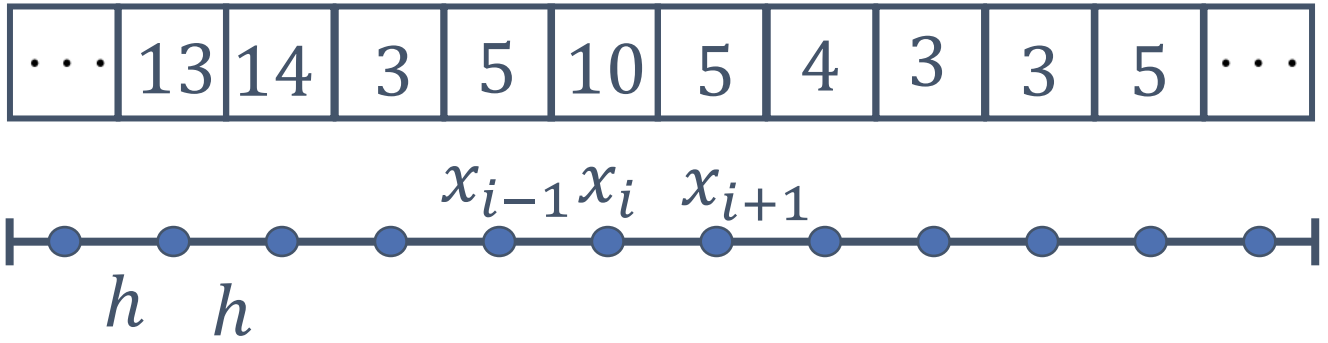
Discrete Laplacian on 1D:

Let us discretize the a real interval $[-\pi, \pi]$



Discrete Laplacian on 1D:

Let us discretize the a real interval $[-\pi, \pi]$



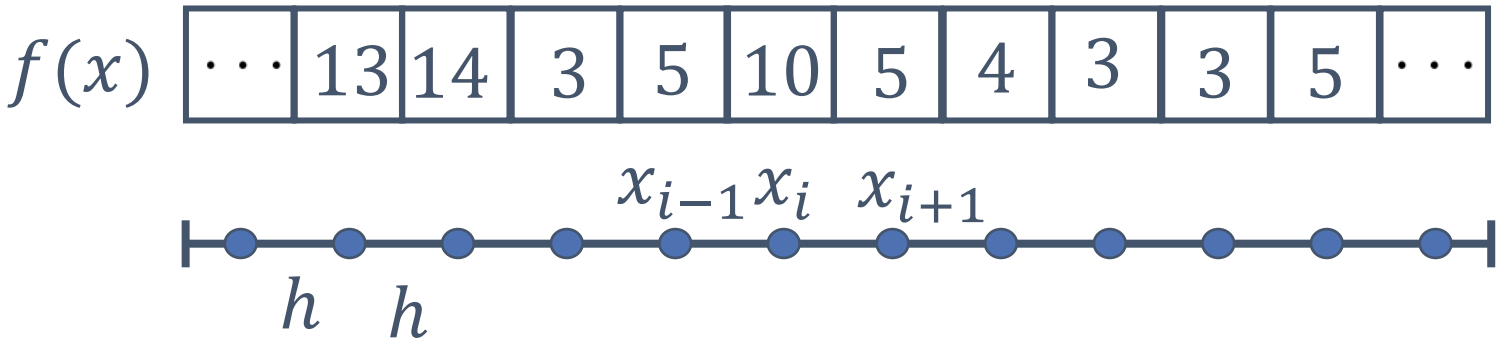
We can compute the finite difference approximation of the first order derivative

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

Slide credits to M. Ovsjanikov

Discrete Laplacian on 1D:

Let us discretize the a real interval $[-\pi, \pi]$



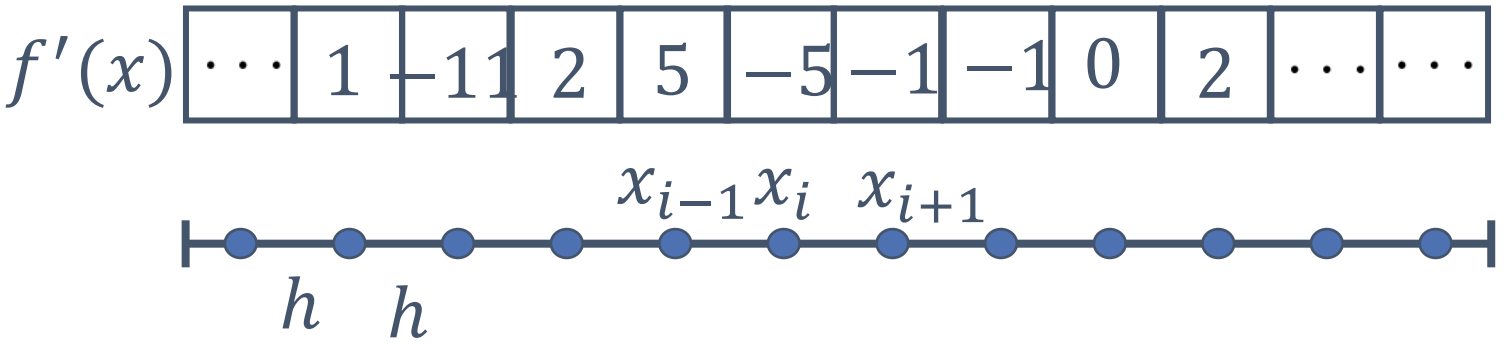
We can compute the finite difference approximation of the first order derivative

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$$

Slide credits to M. Ovsjanikov

Discrete Laplacian on 1D:

That for $h = 1\ is$:

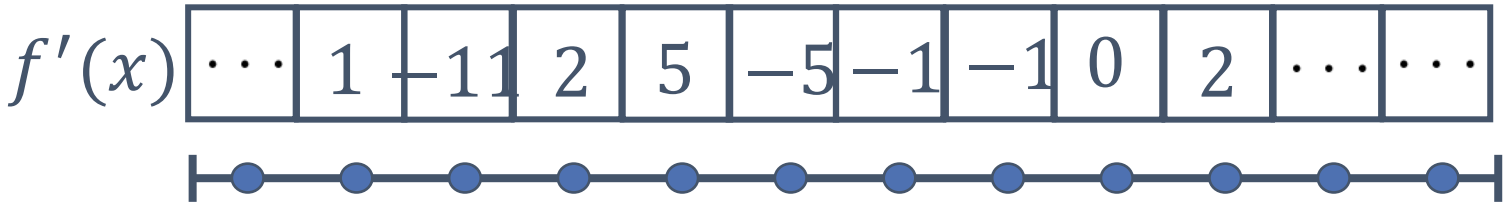


We can compute the finite difference approximation of the first order derivative

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$$

Slide credits to M. Ovsjanikov

Discrete Laplacian on 1D:



We can compute the finite difference approximation of the second order derivative

$$f''(x_i) = \frac{f'(x_{i+1}) - f'(x_i)}{h}$$

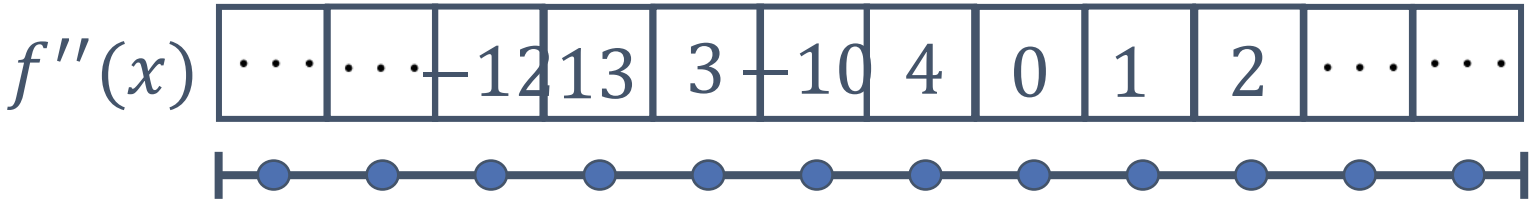
$$= \frac{\frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f(x_i) - f(x_{i-1})}{h}}{h}$$

$$= \frac{f(x_{i+1}) - f(x_i) - f(x_i) + f(x_{i-1})}{h^2}$$

$$= \frac{f(x_{i+1}) + f(x_{i-1}) - 2f(x_i)}{h^2}$$

Slide credits to M. Ovsjanikov

Discrete Laplacian on 1D:



We can compute the finite difference approximation of the second order derivative

$$f''(x_i) = \frac{f'(x_{i+1}) - f'(x_i)}{h}$$

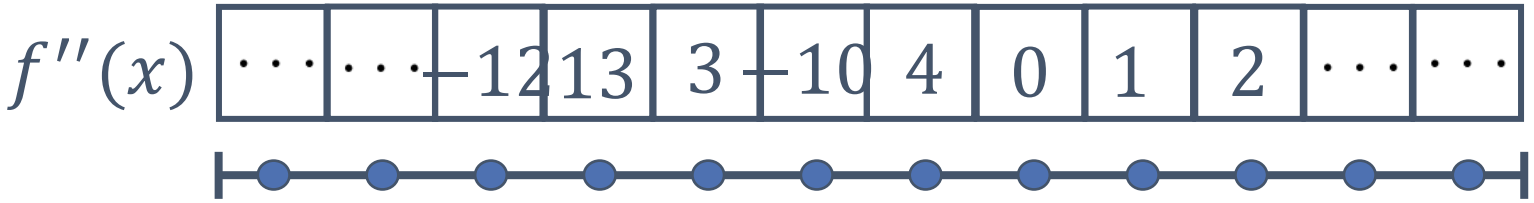
$$= \frac{\frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f(x_i) - f(x_{i-1})}{h}}{h}$$

$$= \frac{f(x_{i+1}) - f(x_i) - f(x_i) + f(x_{i-1})}{h^2}$$

$$= \frac{f(x_{i+1}) + f(x_{i-1}) - 2f(x_i)}{h^2}$$

Slide credits to M. Ovsjanikov

Discrete Laplacian on 1D:



With $h = 1$ we have that:

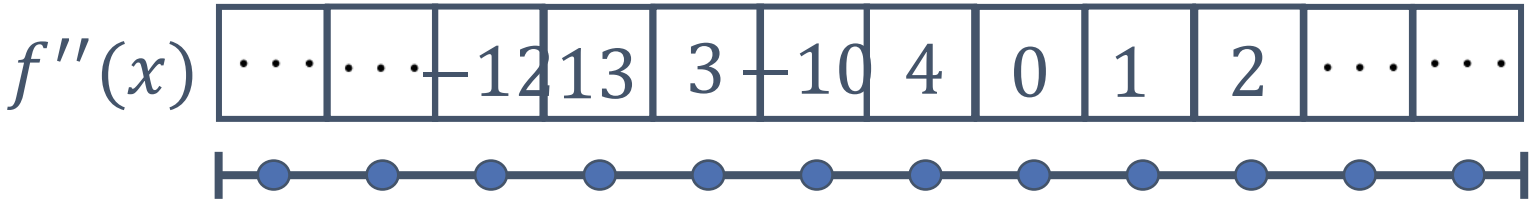
$$f''(x_i) = f(x_{i+1}) + f(x_{i-1}) - 2f(x_i)$$

If we store the values of f in a vector we have:

$$f'' = Lf \qquad L_{i,j} = \begin{cases} 1 \ if \ i \neq j \ and \ j \in \mathcal{N}_i \\ -d_i = -\sum_{\substack{k=1 \\ k \neq i}}^{n} L_{i,k} \ \ if \ i = j \\ 0 \ otherwise \end{cases}$$

Slide credits to M. Ovsjanikov

Discrete Laplacian on 1D:



More generally:

$$f'' = Lf \qquad L_{i,j} = \begin{cases} w_{i,j} \;\; if \;\; i \neq j \;\; and \;\; j \in \mathcal{N}_i \\ -d_i = -\sum_{\substack{k=1 \\ k \neq i}}^{n} w_{i,k} \;\; if \;\; i = j \\ 0 \;\; otherwise \end{cases}$$

Slide credits to M. Ovsjanikov

Discrete Laplacian 2D:

Similarly for the 2D

...	...	...	...	...	...
...	...	...	$f(x, y+1)$	...	...
...	...	$f(x-1, y)$	$f(x, y)$	$f(x+1, y)$	...
...	...	...	$f(x, y-1)$	...	...
...	...	...	...	...	...

$$\Delta f(x,y) =$$
$$= f(x+1,y) + f(x-1,y) + f(x,y-1) +$$
$$+ f(x,y+1) - 4f(x,y)$$

Slide credits to M. Ovsjanikov

Discrete Laplacian 2D:

Similarly for the 2D

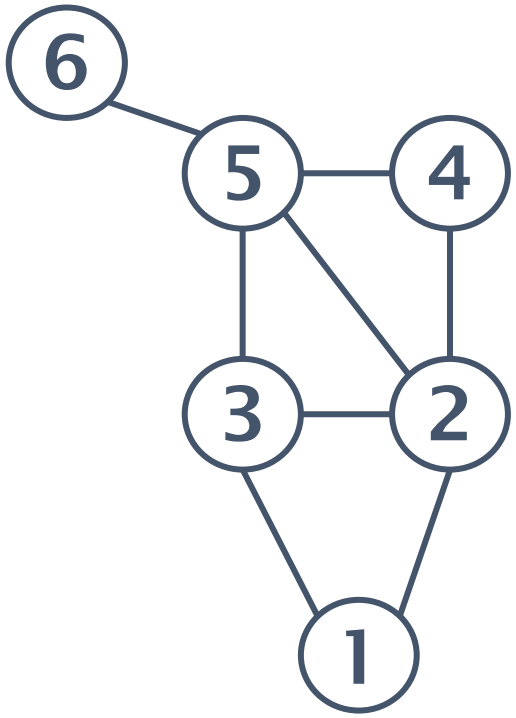
...	...	...	...	...	...
...	...	...	$f(x, y+1)$	...	...
...	...	$f(x-1, y)$	$f(x, y)$	$f(x+1, y)$	...
...	...	...	$f(x, y-1)$	...	...
...	...	...	...	...	...

$$f'' = \Delta f \qquad \Delta_{i,j} = \begin{cases} w_{i,j} \;\; if \;\; i \neq j \;\; and \;\; j \in \mathcal{N}_i \\ -d_i = -\sum_{\substack{k=1 \\ k \neq i}}^{n} w_{i,k} \;\; if \;\; i = j \\ 0 \;\; otherwise \end{cases}$$

Slide credits to M. Ovsjanikov

Graph Laplacian:

Instead of grids that approximate Euclidean domains we can consider graphs:

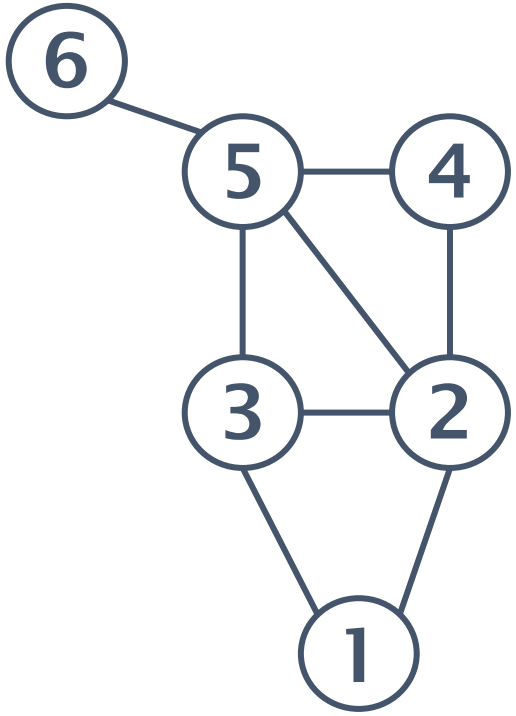


$$L_{i,j} = \begin{cases} -1 \;\; if \;\; i \;\; \neq j \;\; and \;\; j \in \mathcal{N}_i \\ d_i = -\sum_{\substack{k=1 \\ k \neq i}}^{n} L_{i,k} \;\; if \;\; i = j \\ 0 \;\; otherwise \end{cases}$$

5 has a neighborhood the nodes 2 3 4 6 i.e. $\mathcal{N}_5 = \{2,3,4,6\}$

Slide credits to M. Ovsjanikov

Graph Laplacian:



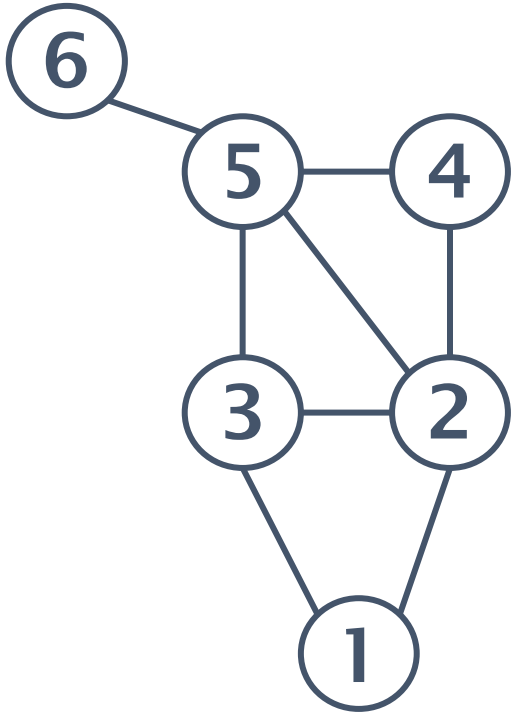
$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$L_{i,j} = \begin{cases} -1 \ if \ i \neq j \ and \ j \in \mathcal{N}_i \\ d_i = -\sum_{\substack{k=1 \\ k \neq i}}^{n} L_{i,k} \ \ if \ i = j \\ 0 \ otherwise \end{cases}$$

d_i is the degree of the node and is = to the number of edges connected to it ($d_5 = 4$)

Slide credits to M. Ovsjanikov

Graph Laplacian:



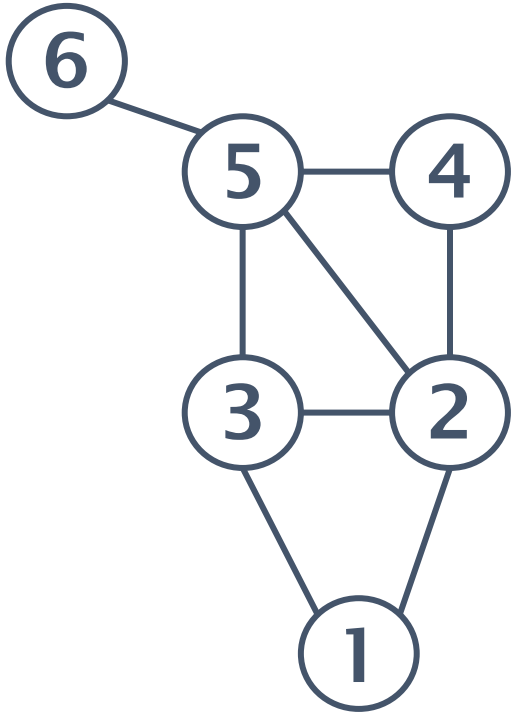
$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Given a pair of real valued functions $f, g : \mathcal{G} \to \mathbb{R}$

We can represent them as the vector $\vec{f}$ in $\mathbb{R}^6$ such that $\vec{f}_i = f(i)$ (and the same holds for g)

Slide credits to M. Ovsjanikov

Graph Laplacian:



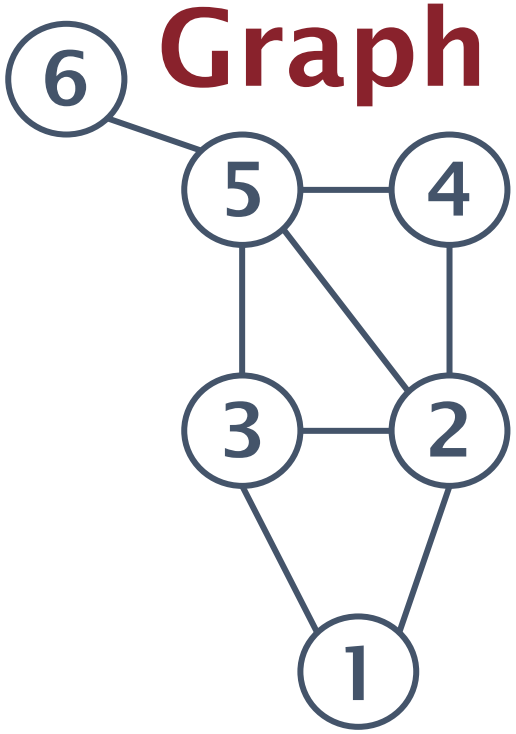
$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Given a pair of real valued functions $f, g : \mathcal{G} \to \mathbb{R}$

$$\int_{\mathcal{G}} g\Delta f \, d\mu = \sum_{i=1}^{6} g_i (\Delta f)_i = \sum_{i=1}^{6} g_i \sum_{j \in \mathcal{N}_i} (f_i - f_j) =$$

Slide credits to M. Ovsjanikov

Graph Laplacian:



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

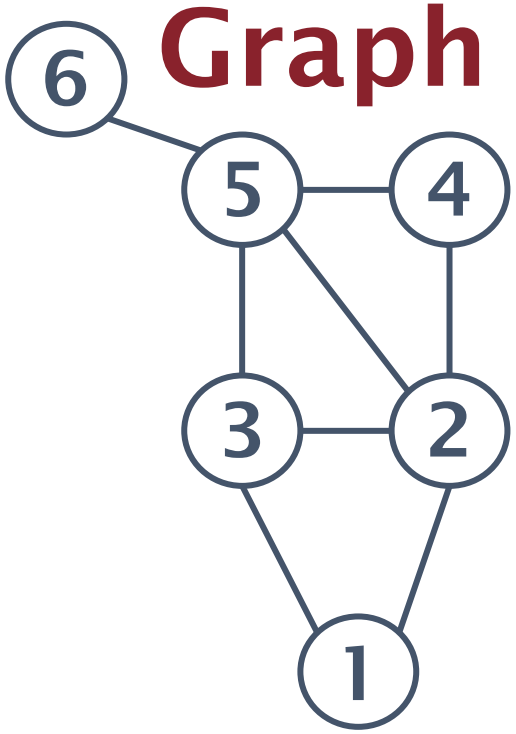
Given $f, g : \mathcal{G} \to \mathbb{R}$

$$\int_{\mathcal{G}} g\Delta f \, d\mu = \sum_{i=1}^{6} g_i(\Delta f)_i = \sum_{i=1}^{6} g_i \sum_{j \in \mathcal{N}_i} (f_i - f_j) =$$

$$= \sum_{\substack{(i,j)\in E \\ i<j}} (g_i f_i - g_i f_j) = \sum_{\substack{(i,j)\in E \\ i<j}} (g_i - g_j)(f_i - f_j) = \sum \langle \nabla g, \nabla f \rangle$$

Slide credits to M. Ovsjanikov

Graph Laplacian:



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

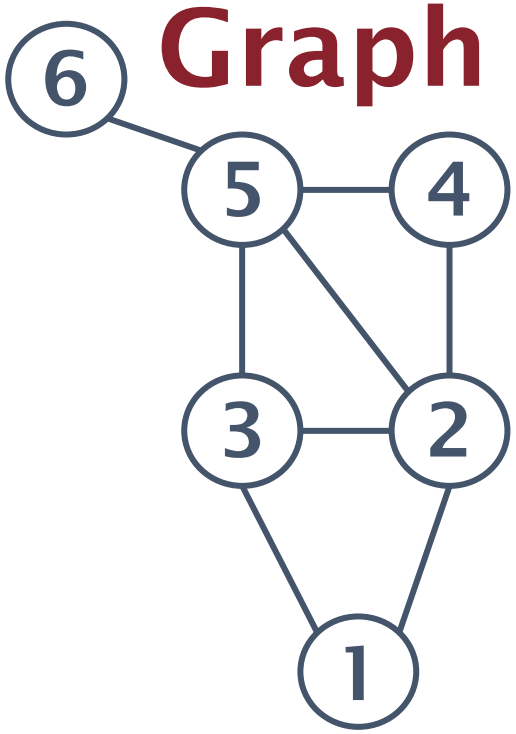
Given $f\colon \mathcal{G} \to \mathbb{R}$ $(f = g)$

$$\int_{\mathcal{G}} f \Delta f \, d\mu = \sum_{i=1}^{6} f_i (\Delta f)_i = \sum_{i=1}^{6} f_i \sum_{j \in \mathcal{N}_i} (f_i - f_j) =$$

$$= \sum_{\substack{(i,j)\in E \\ i<j}} (f_i f_i - f_i f_j) = \sum_{\substack{(i,j)\in E \\ i<j}} (f_i - f_j)(f_i - f_j) = \sum \langle \nabla f, \nabla f \rangle$$

Slide credits to M. Ovsjanikov

Graph Laplacian:



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

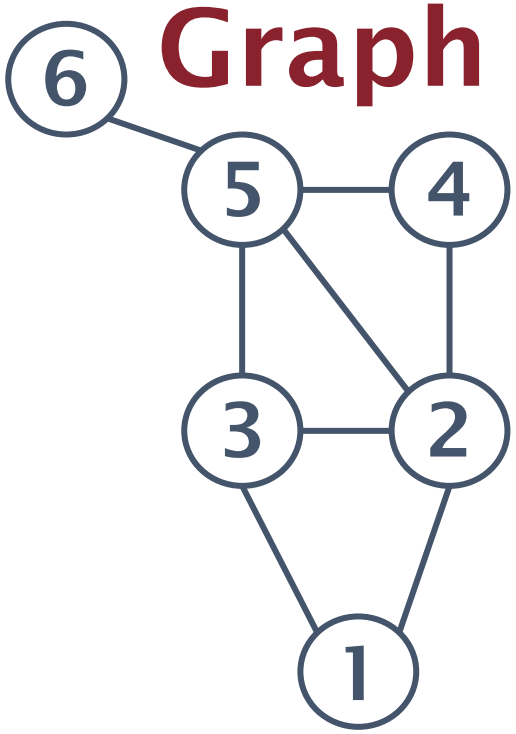
Given $f\colon \mathcal{G} \to \mathbb{R}$ $(f = g)$

$$\int_{\mathcal{G}} f \Delta f \, d\mu = \sum_{i=1}^{6} f_i (\Delta f)_i = \sum_{i=1}^{6} f_i \sum_{j \in \mathcal{N}_i} (f_i - f_j) =$$

$$= \sum_{\substack{(i,j) \in E \\ i<j}} (f_i f_i - f_i f_j) = \sum_{\substack{(i,j) \in E \\ i<j}} (f_i - f_j)^2 = \sum \langle \nabla f, \nabla f \rangle$$

Slide credits to M. Ovsjanikov

Graph Laplacian:



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Given $f\colon \mathcal{G} \to \mathbb{R}$ $(f = g)$

$$\int_{\mathcal{G}} f \Delta f \, d\mu = \sum_{i=1}^{6} f_i (\Delta f)_i = \sum_{i=1}^{6} f_i \sum_{j \in \mathcal{N}_i} (f_i - f_j) =$$

$$= f^T \Delta f = \sum_{\substack{(i,j) \in E \\ i<j}} (f_i - f_j)^2 = \sum \langle \nabla f, \nabla f \rangle$$

Slide credits to M. Ovsjanikov

Eigendecomposition:

$$f^T \Delta f = \sum_{(i,j)\in E} (f_i - f_j)^2$$

This **matrix multiplication** approximates the action of the **Laplacian operator** on the discrete setting encoding the **Geometry** of the graph.

Moreover, Δ is a real symmetric matrix → from linear Algebra we know that it has a full set of non-negative real eigenvalues λ_l and associated eigenvoectors φ_l:

$$\lambda_l = \varphi_l^T \Delta \varphi_l \geq 0$$

Slide credits to M. Ovsjanikov

Eigendecomposition:

$$\lambda_l = \varphi_l^T \Delta \varphi_l \geq 0$$

The eigenvalues are all non-negative and can be sorted in an ascendng order

$$0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

The eigenvoectors φ_l are orthonormal functions (vectors) i.e.:

$$\langle \varphi_l, \varphi_h \rangle = \begin{cases} 0 \; if \; l \neq h \\ 1 \; if \; l = h \end{cases}$$

A matrix view:

The **Laplcian** operator on a graph with n nodes can be encoded by a square **matrix:**

$$\Delta \in \mathbb{R}^{n \times n}$$

We can compute the **eigenvalues** λ_l and **eigenvectors** φ_l of this matrix (they are a finite number $= n$)

$$\lambda_l \varphi_l = \Delta \varphi_l$$

where:

$\lambda_l \in \mathbb{R}^1$

$\varphi_l \in \mathbb{R}^n$

We can organize them as:

$\Lambda = [\lambda_1, \ldots, \lambda_n] \in \mathbb{R}^n$

$\Phi = [\varphi_1, \ldots, \varphi_n] \in \mathbb{R}^{n \times n}$

A matrix view:

The **Laplcian** operator on a graph with n nodes can be encoded by a square **matrix:**

$$\Delta \in \mathbb{R}^{n \times n}$$

We can compute the **eigenvalues** λ_l and **eigenvectors** φ_l of this matrix (they are a finite number $= n$)

$$\lambda_l \varphi_l = \Delta \varphi_l$$

where:

$\lambda_l \in \mathbb{R}^1$

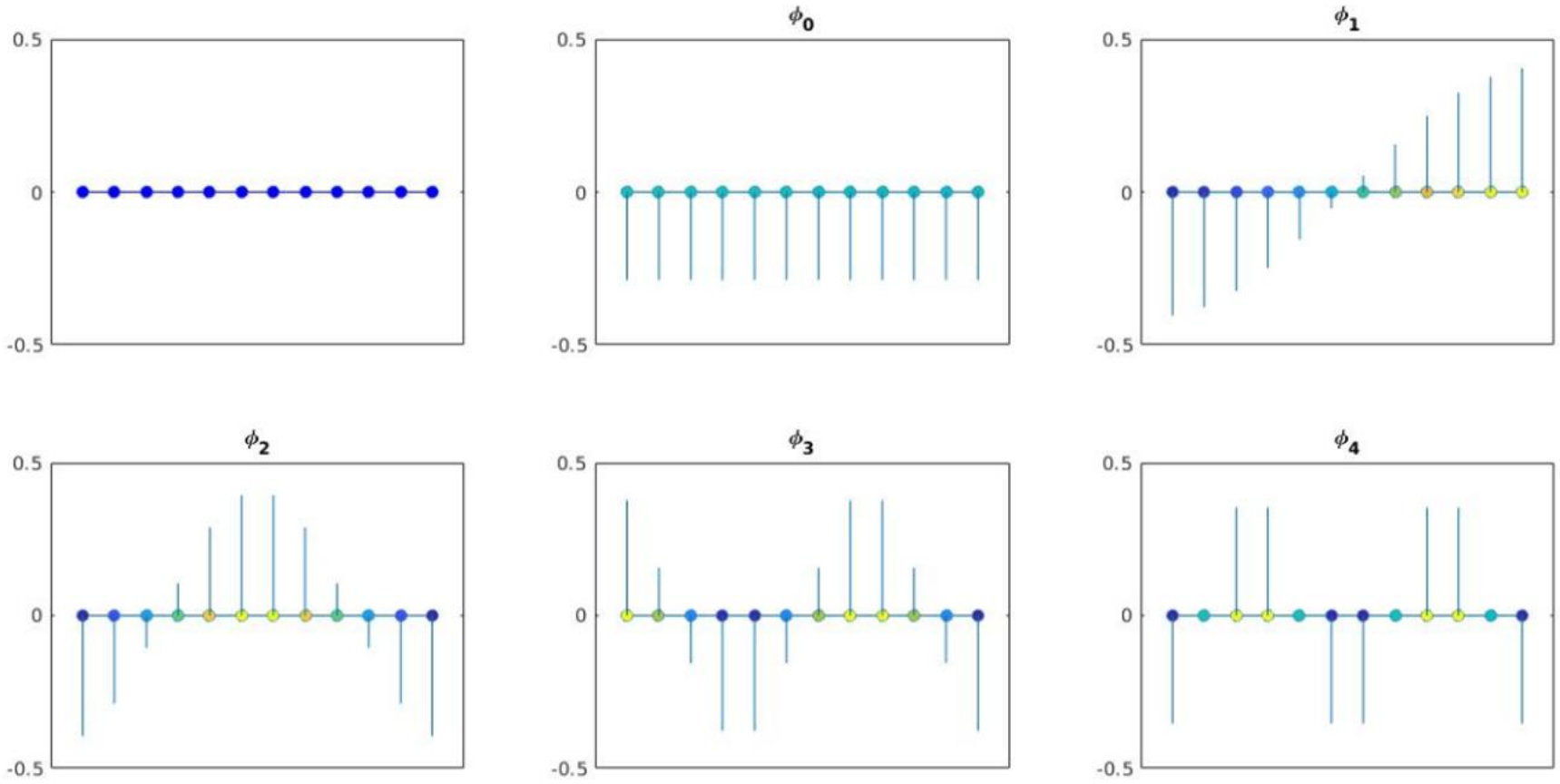
$\varphi_l \in \mathbb{R}^n$

Or just a subset with $k \ll n$:

$\Lambda = [\lambda_1, \ldots, \lambda_k] \in \mathbb{R}^k$

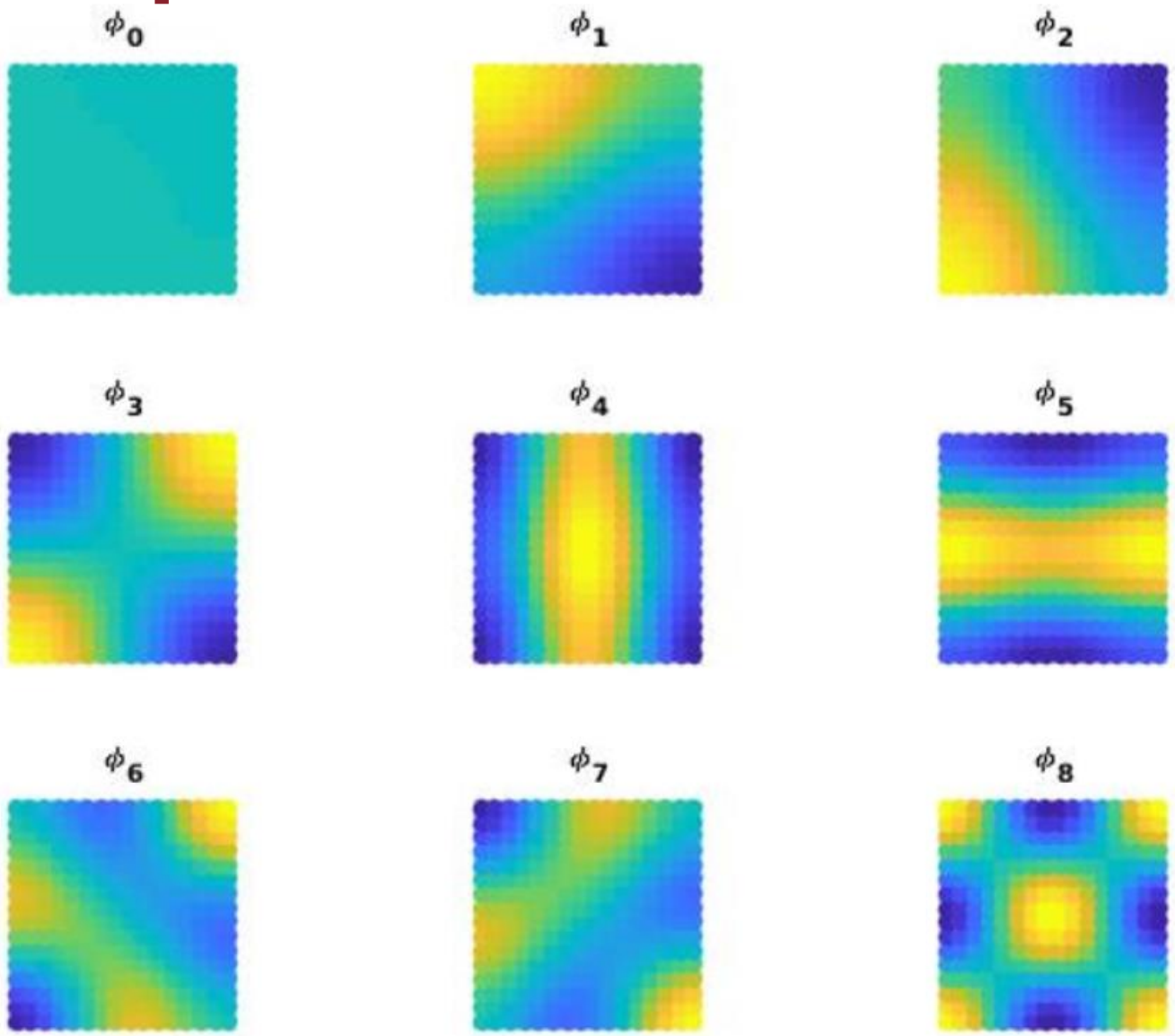
$\Phi = [\varphi_1, \ldots, \varphi_k] \in \mathbb{R}^{n \times k}$

Example 1D:



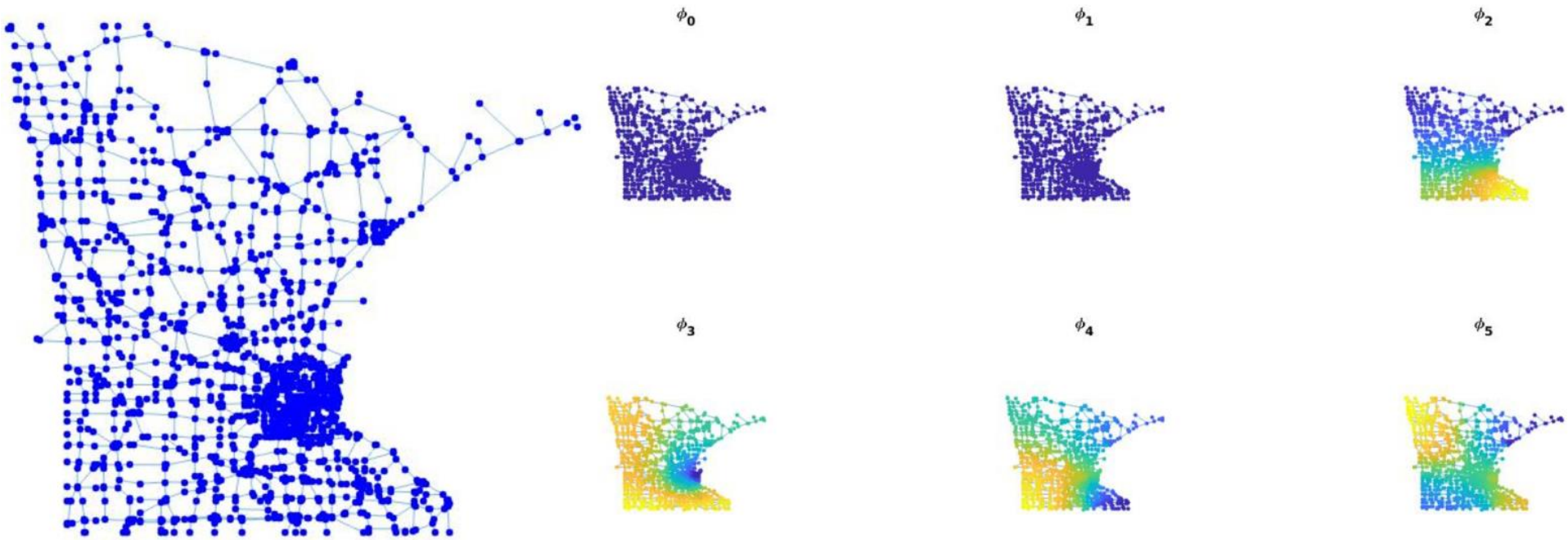
Slide credits to R. Marin

Example 2D:



Slide credits to R. Marin

Example Graph:



Minnesota Road Map, D. Gleich,2010,
https://www.cise.ufl.edu/research/sparse/matrices/Gleich/minnesota.html

Slide credits to R. Marin

LBO Discretetization:

The **LBO** in the discrete setting is represented as a matrix: $\boldsymbol{\Delta}_{\mathcal{M}} = \boldsymbol{\Omega}_{\mathcal{M}}^{-1}\boldsymbol{W}$

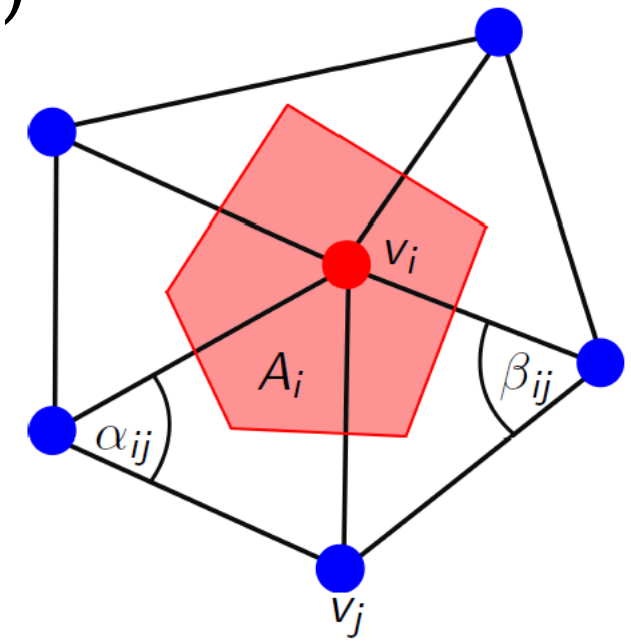
$\boldsymbol{\Omega}_{\mathcal{M}} =$ the mass matrix;

$\boldsymbol{\Omega}_{ii} = A_i$ (the area of the barycell)

$\boldsymbol{W} =$ the stiffness matrix

defined as:

$$w_{ij} = \begin{cases} -\frac{1}{2}(cot(\alpha_{ij}) + cot(\beta_{ij})) \text{ if } i \sim j \\ -\sum_j w_{ij} \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$



Computing discrete minimal surfaces and their conjugates, *Pinkall*, *Polthier*, EXPERIMENTAL MATHEMATICS,1993.

LBO discrete definition:

The **LBO** discretization should satisfy the discrete version of the **Divergence theorem**:

$$\int_{\mathcal{X}} g\Delta f \, d\mu = \int_{\mathcal{X}} -g \, div(\nabla f) \, d\mu = \int_{\mathcal{X}} \langle \nabla g, \nabla f \rangle \, d\mu, \qquad \forall f, g$$

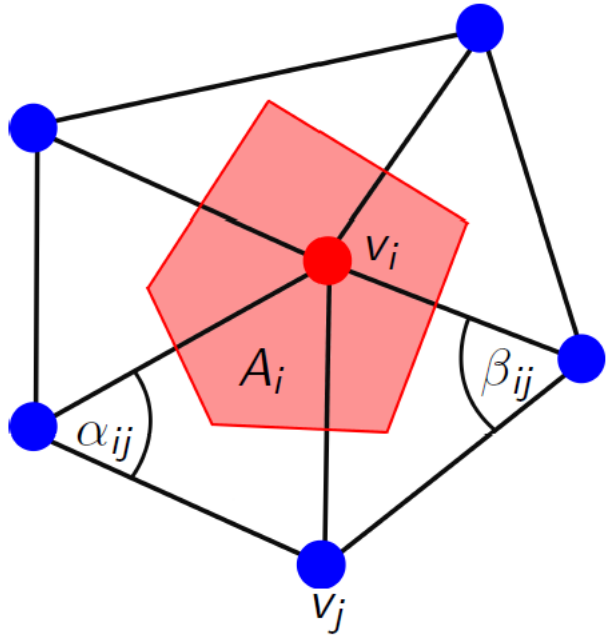
$f, g \in \mathbb{R}^n$ are discrete function defined on $\mathcal{X}$

$$\int_{\mathcal{X}} f \, d\mu = \sum_{i=1}^{n} A_i f(i)$$

where A_i is the area associated to the vertex i or v_i.

$$A_i = \frac{1}{3} \sum_{T \ni i} Area(T)$$

the area of the barycell



Slide credits to M. Ovsjanikov

LBO discrete definition:

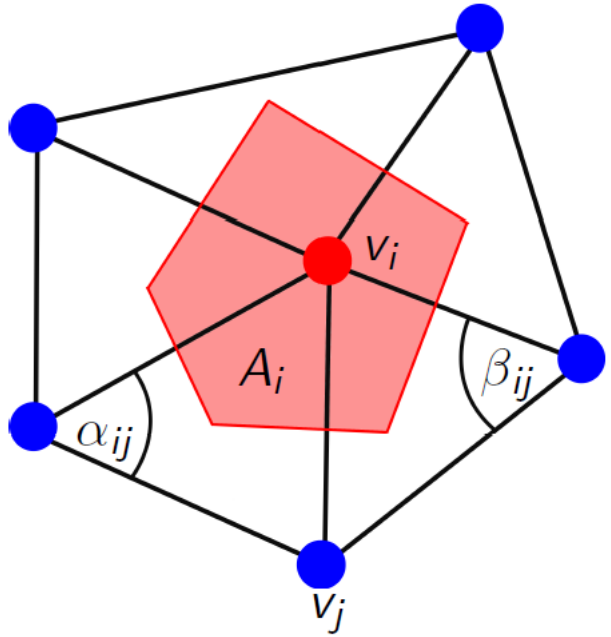
The **LBO** discretization should satisfy the discrete version of the **Divergence theorem**:

$$\int_{\mathcal{X}} g\Delta f \, d\mu = \int_{\mathcal{X}} -g \, div(\nabla f) \, d\mu = \int_{\mathcal{X}} \langle \nabla g, \nabla f \rangle \, d\mu, \qquad \forall f, g$$

$f, g \in \mathbb{R}^n$ are discrete function defined on $\mathcal{X}$

$$\int_{\mathcal{X}} g\Delta f \, d\mu = \sum_{i=1}^{n} A_i g(i) \, \Delta f(\mathrm{i})$$

$$\int_{\mathcal{X}} \langle \nabla g, \nabla f \rangle \, d\mu = \sum_{i=1}^{n} A_i \, \langle \nabla g, \nabla f \rangle (i)$$



Slide credits to M. Ovsjanikov

LBO discrete definition:

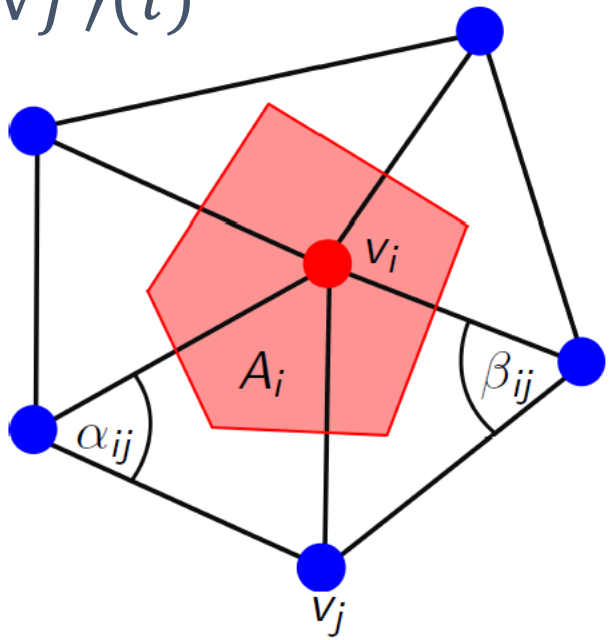
The **LBO** discretization should satisfy the discrete version of the **Divergence theorem**:

$$\int_{\mathcal{X}} g\Delta f \, d\mu = \int_{\mathcal{X}} -g \, div(\nabla f) \, d\mu = \int_{\mathcal{X}} \langle \nabla g, \nabla f \rangle \, d\mu, \qquad \forall f, g$$

$f, g \in \mathbb{R}^n$ are discrete function defined on $\mathcal{X}$

$$\sum_{i=1}^{n} A_i g(i) \, \Delta f(\mathrm{i}) = \sum_{i=1}^{n} A_i \, \langle \nabla g, \nabla f \rangle (i)$$

The gradient depends on the angles of the triangles as in the image.
Through standard computation can obtain



Slide credits to M. Ovsjanikov

LBO Discretetization:

The **LBO** in the discrete setting is represented as a matrix: $\boldsymbol{\Delta}_{\mathcal{M}} = \boldsymbol{\Omega}_{\mathcal{M}}^{-1}\boldsymbol{W}$

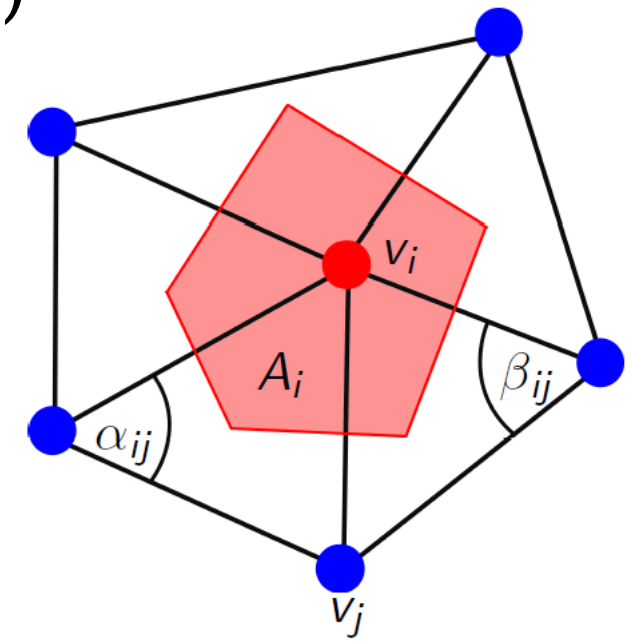
$\boldsymbol{\Omega}_{\mathcal{M}} =$ the mass matrix;

$\boldsymbol{\Omega}_{ii} = A_i$ (the area of the barycell)

$\boldsymbol{W} =$ the stiffness matrix

defined as:

$$w_{ij} = \begin{cases} -\frac{1}{2}(cot(\alpha_{ij}) + cot(\beta_{ij})) \text{ if } i \sim j \\ -\sum_j w_{ij} \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$



Computing discrete minimal surfaces and their conjugates, *Pinkall*, *Polthier*, EXPERIMENTAL MATHEMATICS,1993.

No Free lunch:

There is no discrete **LBO** that is:

- Symmetric
- Local
- Has Linear precision
- Always has positive weights

Discrete Laplace operators: No free lunch

For nice enough meshes the cotangent Laplacian converges weekly and provides a good approximation

No Free lunch:

	SYM	LOC	LIN	POS	PSD	CON
MEAN VALUE	○	●	●	●	○	○
INTRINSIC DEL	●	○	●	●	●	?
COMBINATORIAL	●	●	○	●	●	○
COTAN	●	●	●	○	●	●

M. Wardetzky et al. 2007: "Discrete Laplace operators: No free lunch"

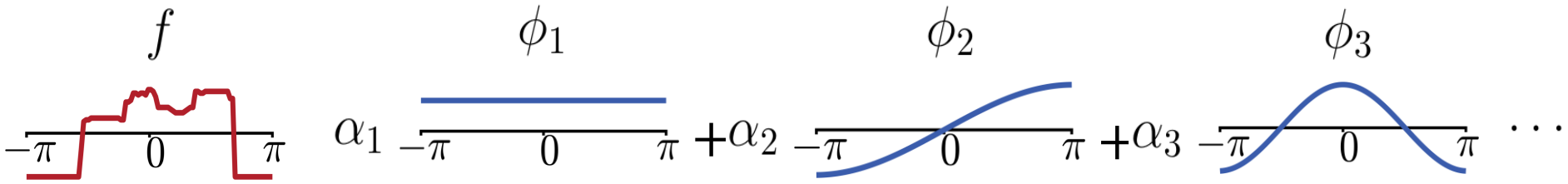
LBO on meshes:

All the applications that we saw for graphs
Are valid for LBO and meshes:

- Positive eigenvalues
- Informative Eigenvectors
- Number of connected components
- Fiedler vector
- Spectral clustering

Moreover there is a strong connection between eigenvectors of the LBO and the Fourier basis (Harmonics)

Fourier:

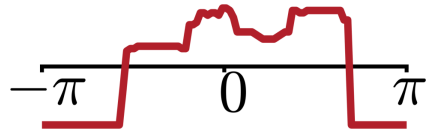


The Fourier basis functions can be estimated as the eigenfunctions of the Euclidean Laplacian

They are ordered w.r.t. the frequecncies that Correspond to the square root of the corresponding Euclidean Laplacian eigenvalues

Fourier Notation:

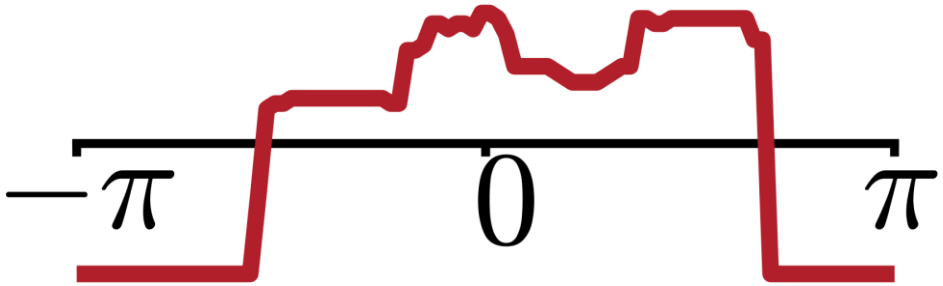
Given a signal: f



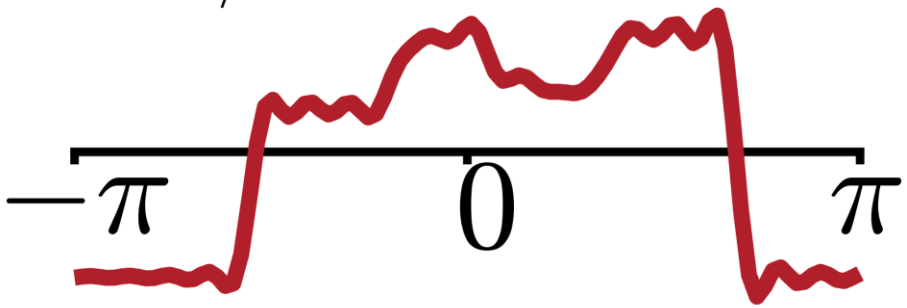
The analysis: $$\alpha_l = \langle f, \phi_l \rangle = \int_{-\pi}^{\pi} f(x)\phi_l(x)dx$$

The synthesis: $$f = \sum_{l=1}^{n} \alpha_l \phi_l = \sum_{l=1}^{n} \langle f, \phi_l \rangle \phi_l \approx \sum_{l=1}^{k<n} \alpha_l \phi_l$$

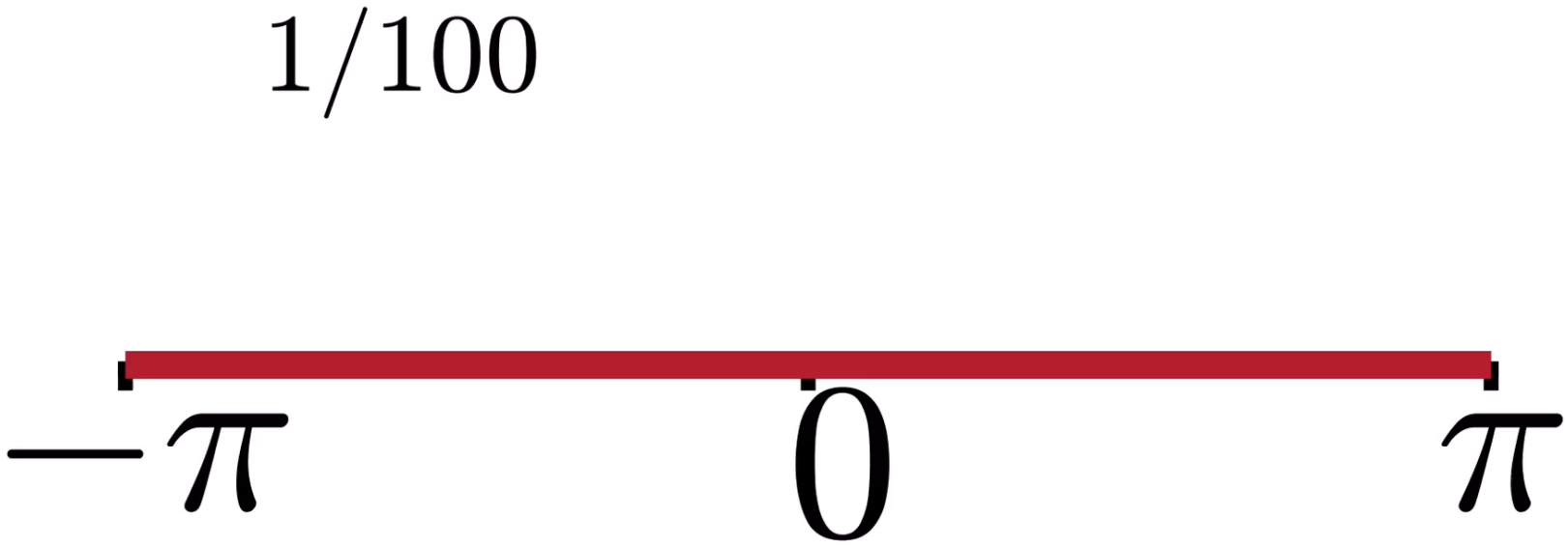
Fourier approximation:



40/100



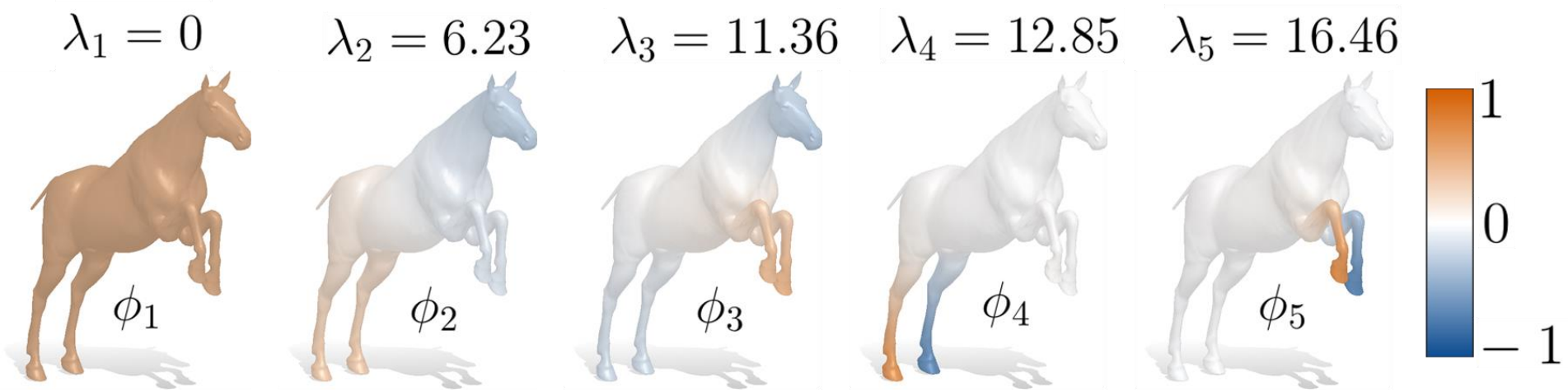
Fourier approximation:



LBO eigenfunction:

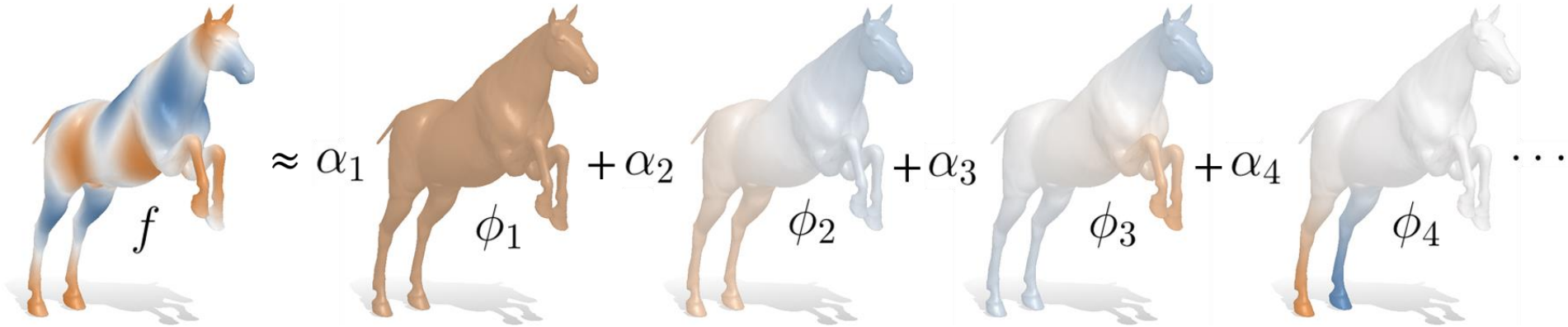
LBO eigenvectors are a basis for the space of functions defined on the mesh (graph, surface) and this basis has similar properties to the Fourier basis

$$\Delta_{\mathcal{M}}\phi_l = \lambda_l\phi_l \qquad \langle\phi_l, \phi_k\rangle_{\mathcal{M}} = \delta_l^k \qquad \lambda_l = \int_{\mathcal{M}} \|\nabla\phi_l\|^2 d\mu(x)$$



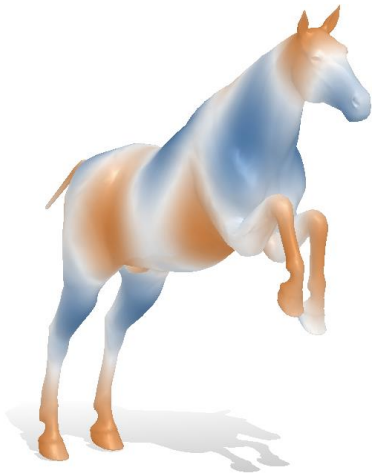
Laplace-beltrami eigenfunctions towards an algorithm that understands geometry, *Levy*, SMI, 2006

LBO representation:



LBO Fourier notation:

Given a signal: f



The analysis: $$\alpha_l = \langle f, \phi_l \rangle_{\mathcal{M}} = \int_{\mathcal{M}} f(x)\phi_l(x)d\mu(x)$$

The synthesis: $$f = \sum_{l=1}^{n} \alpha_l \phi_l = \sum_{l=1}^{n} \langle f, \phi_l \rangle_{\mathcal{M}} \phi_l \approx \sum_{l=1}^{k<n} \alpha_l \phi_l$$

LBO Fourier and matrices:

$$\alpha_l = \langle f, \phi_l \rangle_{\mathcal{M}} = \int_{\mathcal{M}} f(x)\phi_l(x)d\mu(x)$$

Fourier approximation:

Fourier approximation:

LBO Notation:

$$\langle f, g \rangle_{\mathcal{M}} = f^{\top} \mathbf{\Omega}_{\mathcal{M}} g$$

$$\mathbf{\Phi}_{\mathcal{M}} = [\phi_1,\ \phi_2,\ \ldots\ \phi_{k-1},\ \phi_k]$$

$$\mathbf{\Phi}_{\mathcal{M}}^{\dagger}\ s.t.\ \mathbf{\Phi}_{\mathcal{M}}^{\dagger} \mathbf{\Phi}_{\mathcal{M}} = I$$

$$\mathbf{\Phi}_{\mathcal{M}}\ s.t.\ \mathbf{\Phi}_{\mathcal{M}}^{\top} \mathbf{\Omega}_{\mathcal{M}} \mathbf{\Phi}_{\mathcal{M}} = I$$

$$\mathbf{\Phi}_{\mathcal{M}}^{\dagger} = \mathbf{\Phi}_{\mathcal{M}}^{\top} \mathbf{\Omega}_{\mathcal{M}}$$

Given a signal: f

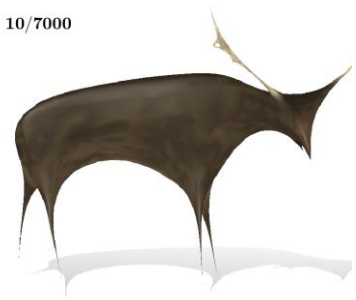
The analysis: $\boldsymbol{\alpha} = \mathbf{\Phi}_{\mathcal{M}}^{\dagger} f$

The synthesis: $f = \mathbf{\Phi}_{\mathcal{M}} \boldsymbol{\alpha} = \mathbf{\Phi}_{\mathcal{M}} \mathbf{\Phi}_{\mathcal{M}}^{\dagger} f$

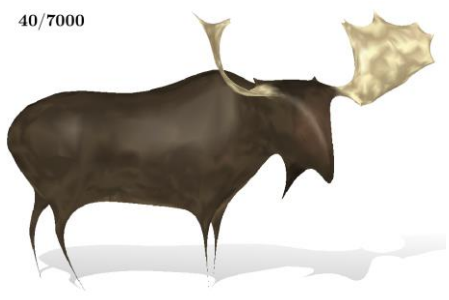
Coordinates approximation:

If as function we consider the 3 coordinates X, Y and Z then we reconstruct the geometry:

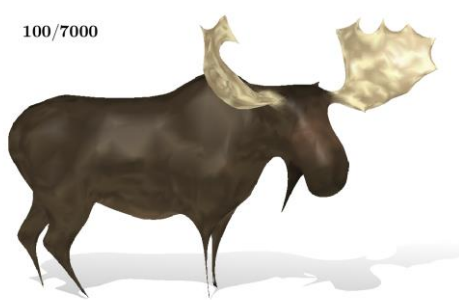
$\tilde{X} = \sum_{i=1}^{k} \alpha_i \varphi_i$, where $\alpha_i = \langle \varphi_i, X \rangle_\chi = {\varphi_i}^T \Omega_\chi X = {\varphi_i}^\dagger X$
The same for Y and Z and then plot $\tilde{X}, \tilde{Y}, \tilde{Z}$



$k = 10$



$k = 40$

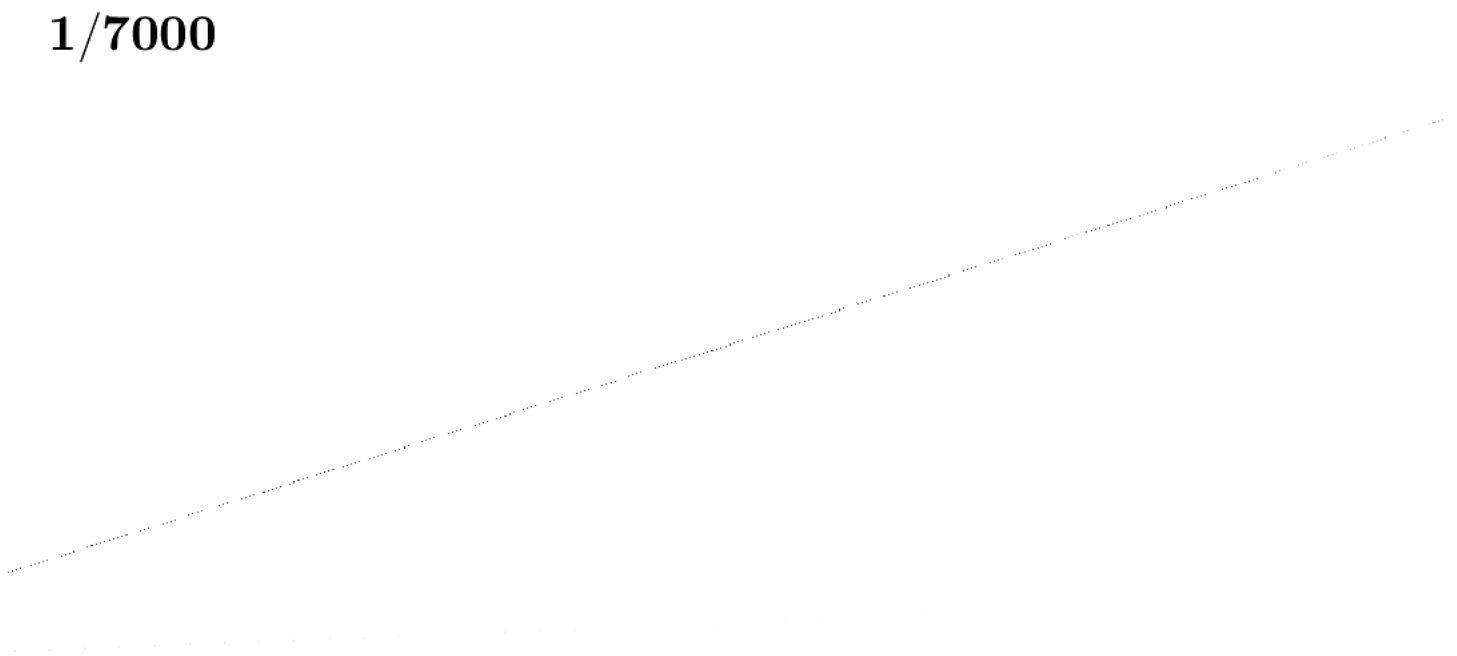


$k = 100$

$k = 200$

Slide credits to M. Ovsjanikov

Fourier approximation:

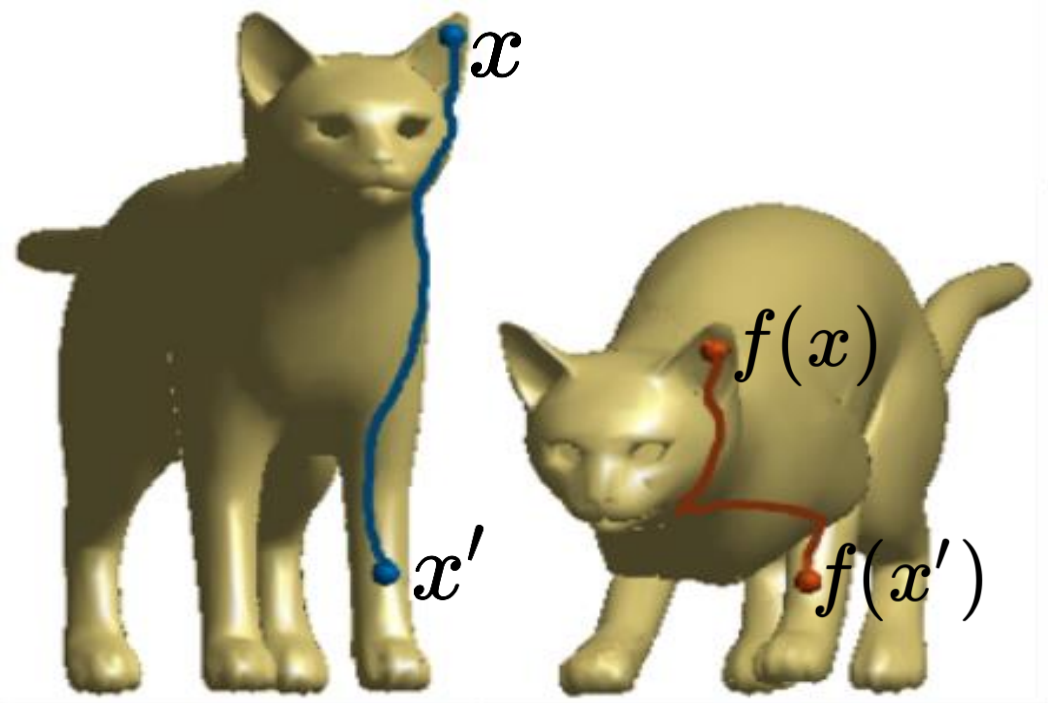


Spectral Solutions
For Shape Matching

LBO and isometries

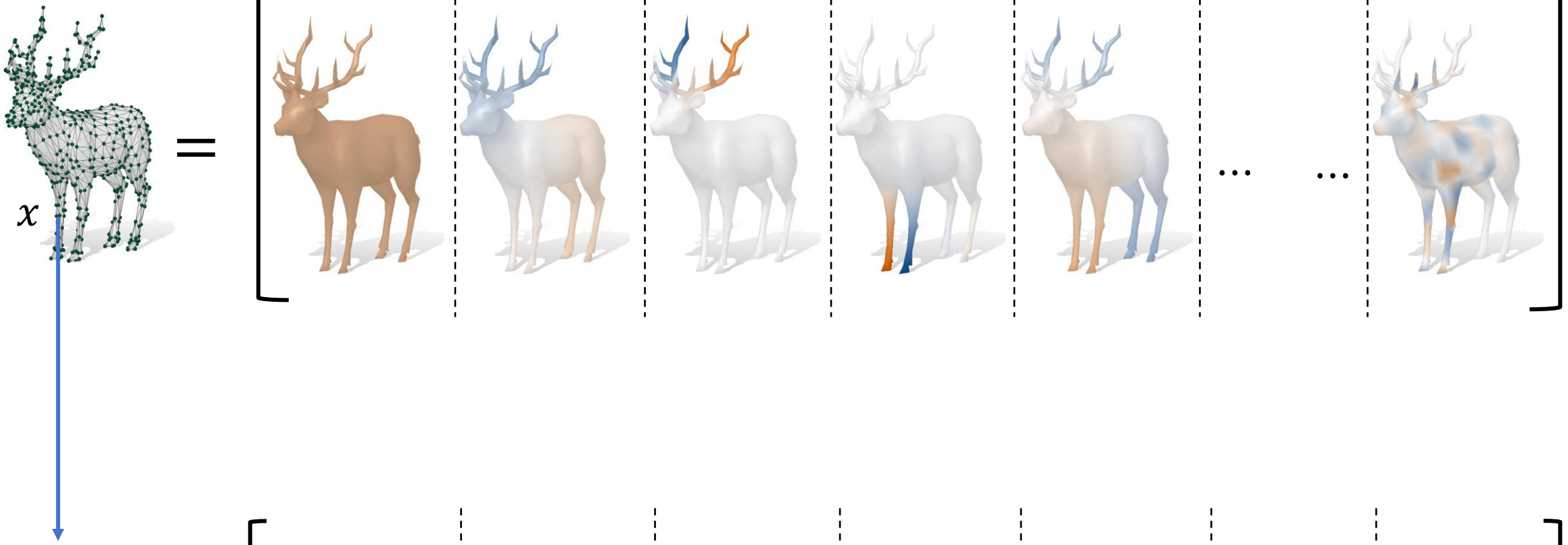
Two shapes are isometric ⇔ their LBO agree

$$d_{\mathcal{X}}(x, x') = d_{\mathcal{Y}}(f(x), f(x')), \ \forall\, x, x'$$



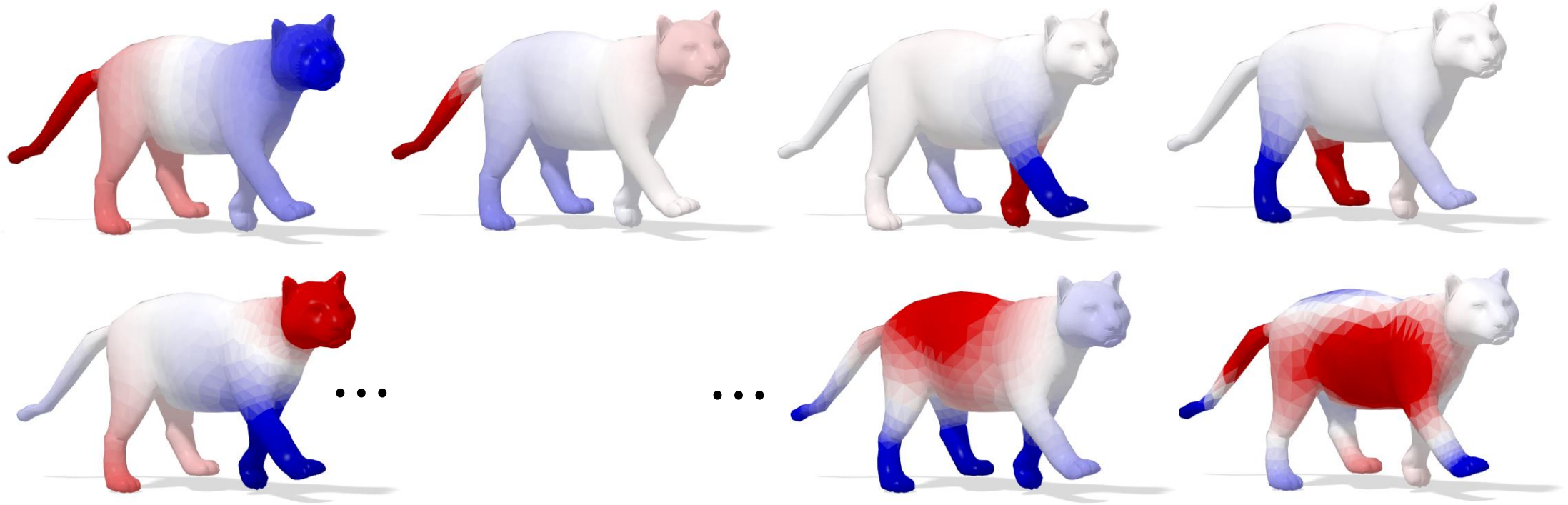
Any quantity derived from the LBO is invariant to isometry

Spectral Embedding



$$\boldsymbol{\Phi}(x) = \left[\begin{array}{c|c|c|c|c|c|c|c} \phi_1(x) & \phi_2(x) & \phi_3(x) & \phi_4(x) & \phi_5(x) & \cdots & \cdots & \phi_k(x) \end{array}\right]$$

GPS: Global Point Signature



$$GPS(x) = [-\frac{1}{\sqrt{\lambda_1}}\phi_1(x), -\frac{1}{\sqrt{\lambda_2}}\phi_2(x), \ldots, -\frac{1}{\sqrt{\lambda_Q}}\phi_Q(x)]$$

Laplace-Beltrami eigenfunctions for deformation invariant shape representation, R. Rustamov, 2007

Heat Equation:

$\mathcal{X}$ is a Riemannian surface, $u(x,t)$

is the amount of heat in a point

$x \in \mathcal{X}$ at time $t \in \mathbb{R}$

Given a initial distribution u_0

of heat on $\mathcal{X}$ at time $t = 0$,

$$u_0(x) = u(x,0)$$

$t = 0$

How is it diffused over time on the surface?

Sun et al., «A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion», 2009.

Heat Equation:

From physics that the heat diffusion is governed by the **heat equation**:

$$\Delta_{\mathcal{X}} u(x,t) = -\frac{\partial u(x,t)}{\partial t}$$

The LBO = derivatives in space

derivative in time

$u(x,t)$ solution of the heat equation is a function of $x \in \mathcal{X}$ and time $t \in \mathbb{R}$ which satisfies the **heat equation** for a given initial condition $u_0(x) = u(x,0)$

Sun et al., «A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion», 2009.

Heat Equation:

Given an initial heat distribution f on $\mathcal{X}$
The solution of the heat diffusion at time $t \in \mathbb{R}$ is given by the **heat operator** H_t **:**

$$H_t = e^{-t\boldsymbol{\Delta}_x}$$

$$e^{-t\lambda_l}$$

$$\boldsymbol{\Delta}_{\mathcal{X}} : \mathcal{F}(\mathcal{X}, \mathbb{R}) \longrightarrow \mathcal{F}(\mathcal{X}, \mathbb{R}) \quad H_t : \mathcal{F}(\mathcal{X}, \mathbb{R}) \longrightarrow \mathcal{F}(\mathcal{X}, \mathbb{R})$$

Sun et al., «A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion», 2009.

Heat Equation:

Given an initial heat distribution f on $\mathcal{X}$

$$H_t f(x) = \int_{\mathcal{X}} k_t(x, y) f(y) d\mu(y)$$

$H_t(f)$ is the heat distribution at time $t \in \mathbb{R}$ and

H_t is the **heat operator** (that depends on $\mathcal{X}$)

$$H_t = e^{-t\boldsymbol{\Delta}_{\mathcal{X}}}$$

$$\boldsymbol{\Delta}_{\mathcal{X}} : \mathcal{F}(\mathcal{X}, \mathbb{R}) \longrightarrow \mathcal{F}(\mathcal{X}, \mathbb{R}) \quad H_t : \mathcal{F}(\mathcal{X}, \mathbb{R}) \longrightarrow \mathcal{F}(\mathcal{X}, \mathbb{R})$$

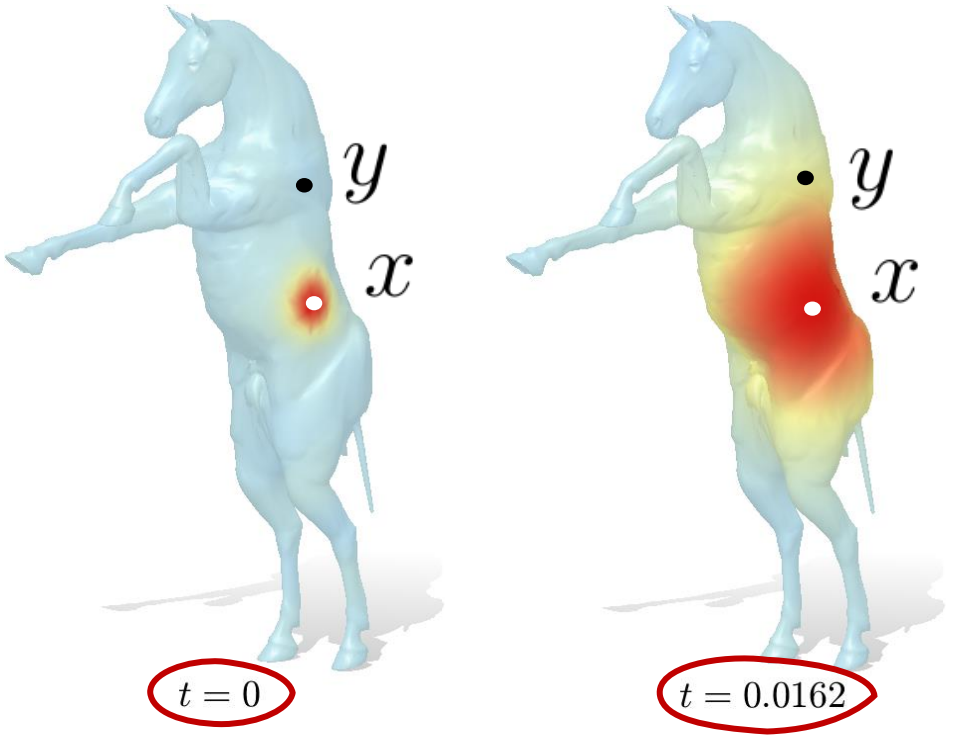
Sun et al., «A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion», 2009.

Heat Equation:

There is a function $k_t : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ such that:

$$H_t f(x) = \int_{\mathcal{X}} k_t(x, y) f(y) d\mu(y)$$

k_t is the heat kernel and $k_t(x, y)$ corresponds to the heat transferred from x to y in time $t \in \mathbb{R}$

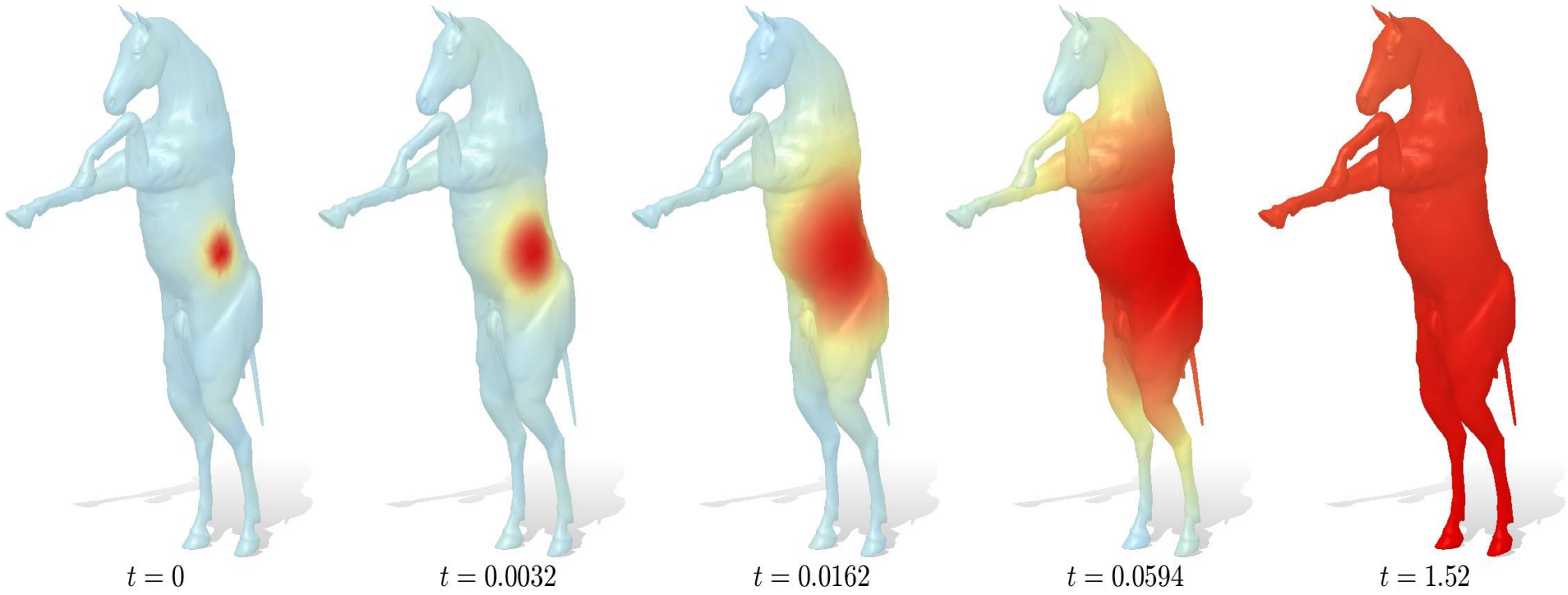


Sun et al., «A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion», 2009.

Heat Equation:

For an initial delta distribution of heat $\delta_x, \ x \in \mathcal{X}$

the heat kernel $k_t(x,y) = \sum_{l=0}^{\infty} e^{-t\lambda_l}\phi_l(x)\phi_l(y)$



Sun et al., «A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion», 2009.

HKS: Heat Kernel Signature

For an initial delta distribution of heat $\delta_x, \ x \in \mathcal{X}$

$$k_t(x,x) = \sum_{l=0}^{\infty} e^{-t\lambda_l} \phi_l(x) \phi_l(x)$$

Is the amount of heat remaining at x after the time $t \in \mathbb{R}$

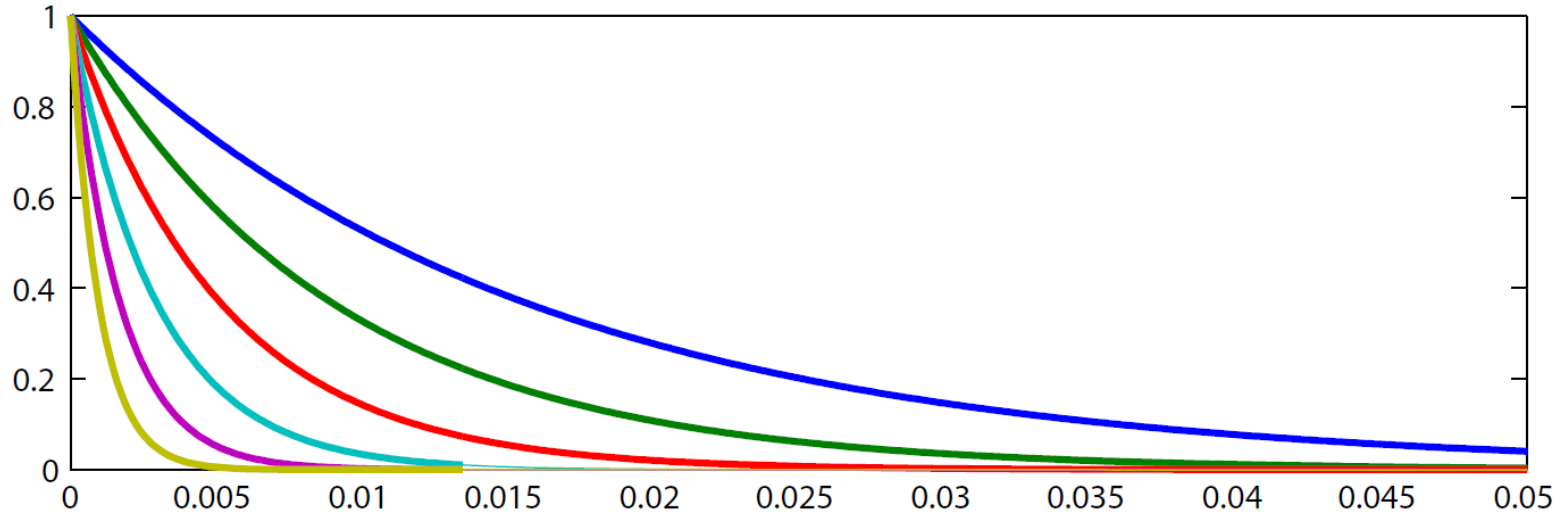
$$\textbf{\textit{HKS}}(x) = [k_{t_1}(x,x), k_{t_2}(x,x), \ldots, k_{t_Q}(x,x)] \quad t_1 < t_2 < \ldots t_Q \in \mathbb{R}$$

is the heat kernel signature (HKS) at the point $x \in \mathcal{X}$ for a given set of time scales $t_1, \ldots, t_Q$

HKS as a filter on frequencies

$$k_t(x,x) = \sum_{l=1}^{\infty} e^{-t\lambda_l}\phi_l(x)\phi_l(x) = \sum_{l=1}^{\infty} e^{-t\lambda_l}\phi_l(x)^2$$

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



A low-pass filter applied to the frequencies to produce the HKS

The wave equation (Schrödinger)

Heat equation: $\Delta_{\mathcal{X}} u(x,t) = -\dfrac{\partial u(x,t)}{\partial t}$

Wave equation: $i\Delta_{\mathcal{X}} u(x,t) = \dfrac{\partial u(x,t)}{\partial t}$

It governs the temporal evolution of a quantum particle

presence of the i

missing a minus

It encodes oscillation rather than dissipation as done by the heat equation

Idea: point $x \leftrightarrow$ the average probabilities of quantum particles of different energies to be measured at x

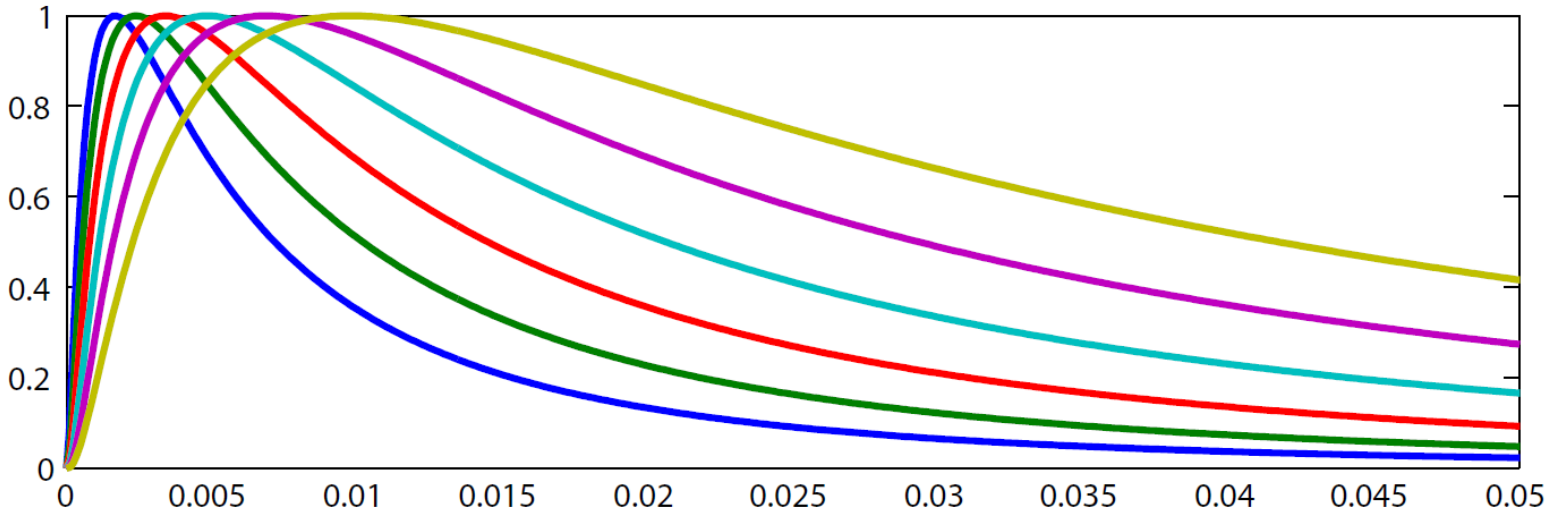
The wave kernel signature: A quantum mechanical approach to shape analysis, *Aubry et al.*, 2011.

WKS as a filter on the frequencies

$$k_E(x,x) = \sum_{l=1}^{\infty} e^{-\frac{(log(E) - log(\lambda_l))^2}{2\sigma^2}} \phi_l(x)^2$$

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

A band-pass filter applied to the frequencies to produce the WKS



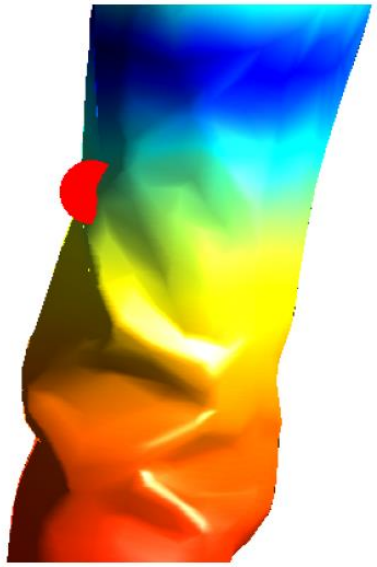
HKS and WKS

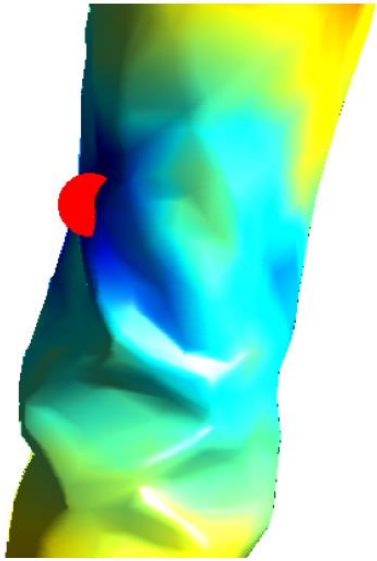
HKS

WKS

reference shape

new shape

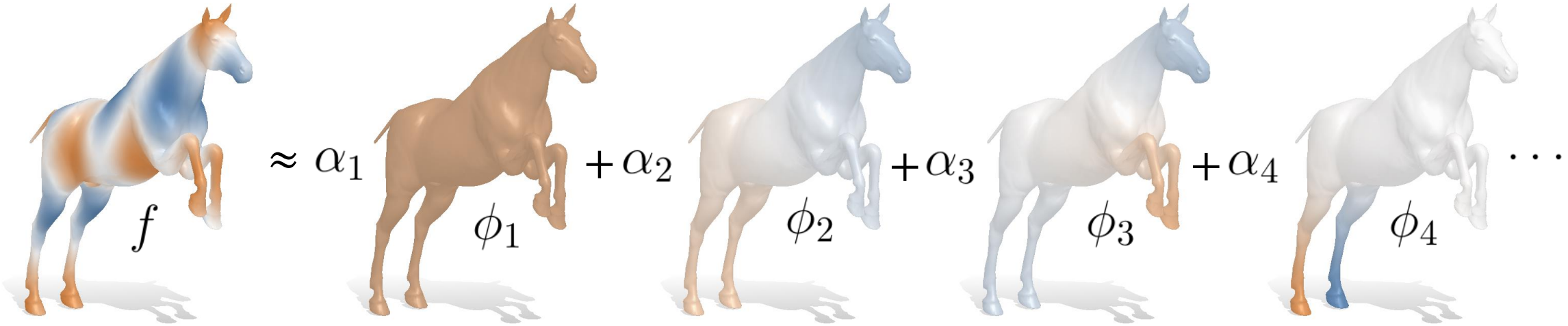




zoom on the new shape

Functional Maps

Fourier representation



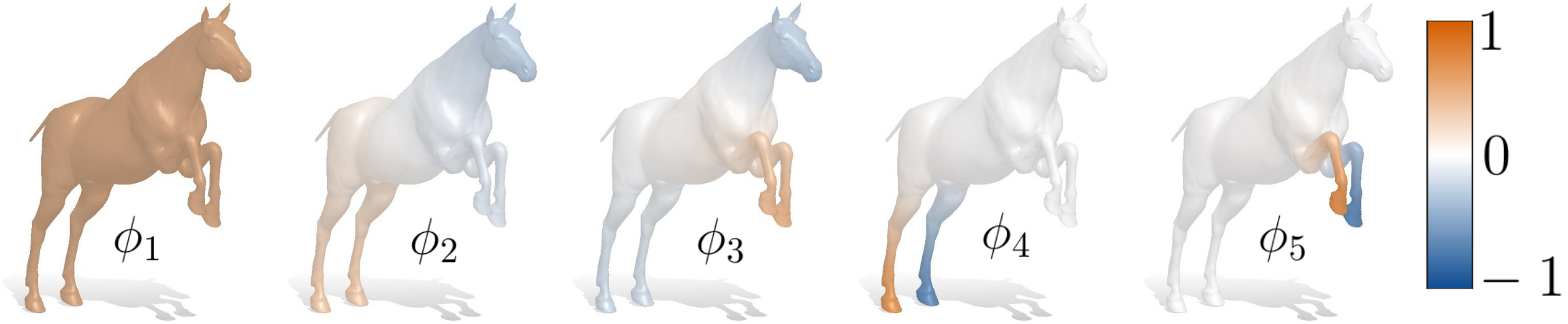
Laplace-beltrami eigenfunctions towards an algorithm that understands geometry, *Levy*, 2006

Fourier Basis on 3D shapes

The eigenfunctions of the Laplace Beltrami Operator (LBO)

$$\Delta_{\mathcal{M}}\phi_l = \lambda_l\phi_l \qquad \langle\phi_l, \phi_k\rangle_{\mathcal{M}} = \delta_l^k \qquad \lambda_l = \int_{\mathcal{M}} \|\nabla\phi_l\|^2 d\mu(x)$$

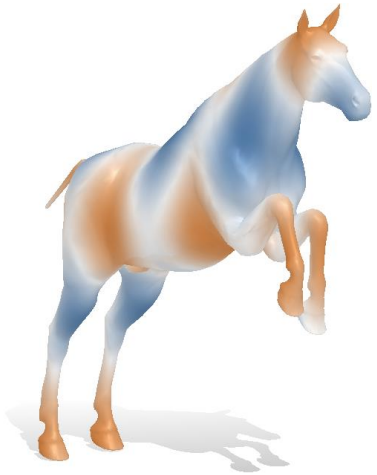
$\lambda_1 = 0$ $\lambda_2 = 6.23$ $\lambda_3 = 11.36$ $\lambda_4 = 12.85$ $\lambda_5 = 16.46$



Laplace-beltrami eigenfunctions towards an algorithm that understands geometry, *Levy*, 2006

Synthesis and analysis

Given a signal: f



The analysis:

$$\alpha_l = \langle f, \phi_l \rangle_{\mathcal{M}} = \int_{\mathcal{M}} f(x)\phi_l(x)d\mu(x)$$

The synthesis:

$$f = \sum_{l=1}^{n} \alpha_l \phi_l = \sum_{l=1}^{n} \langle f, \phi_l \rangle_{\mathcal{M}} \phi_l \approx \sum_{l=1}^{k<n} \alpha_l \phi_l$$

Laplace-beltrami eigenfunctions towards an algorithm that understands geometry, *Levy*, 2006

Notation

Given a signal: f

The analysis: $\boldsymbol{\alpha} = \boldsymbol{\Phi}_{\mathcal{M}}^{\dagger} f$

$$\langle f, g \rangle_{\mathcal{M}} = f^{\top} \boldsymbol{\Omega}_{\mathcal{M}} g$$

$$\boldsymbol{\Phi}_{\mathcal{M}} = [\phi_1,\ \phi_2,\ \ldots\ \phi_{k-1},\ \phi_k]$$

$$\boldsymbol{\Phi}_{\mathcal{M}}^{\dagger}\ s.t.\ \boldsymbol{\Phi}_{\mathcal{M}}^{\dagger} \boldsymbol{\Phi}_{\mathcal{M}} = I$$

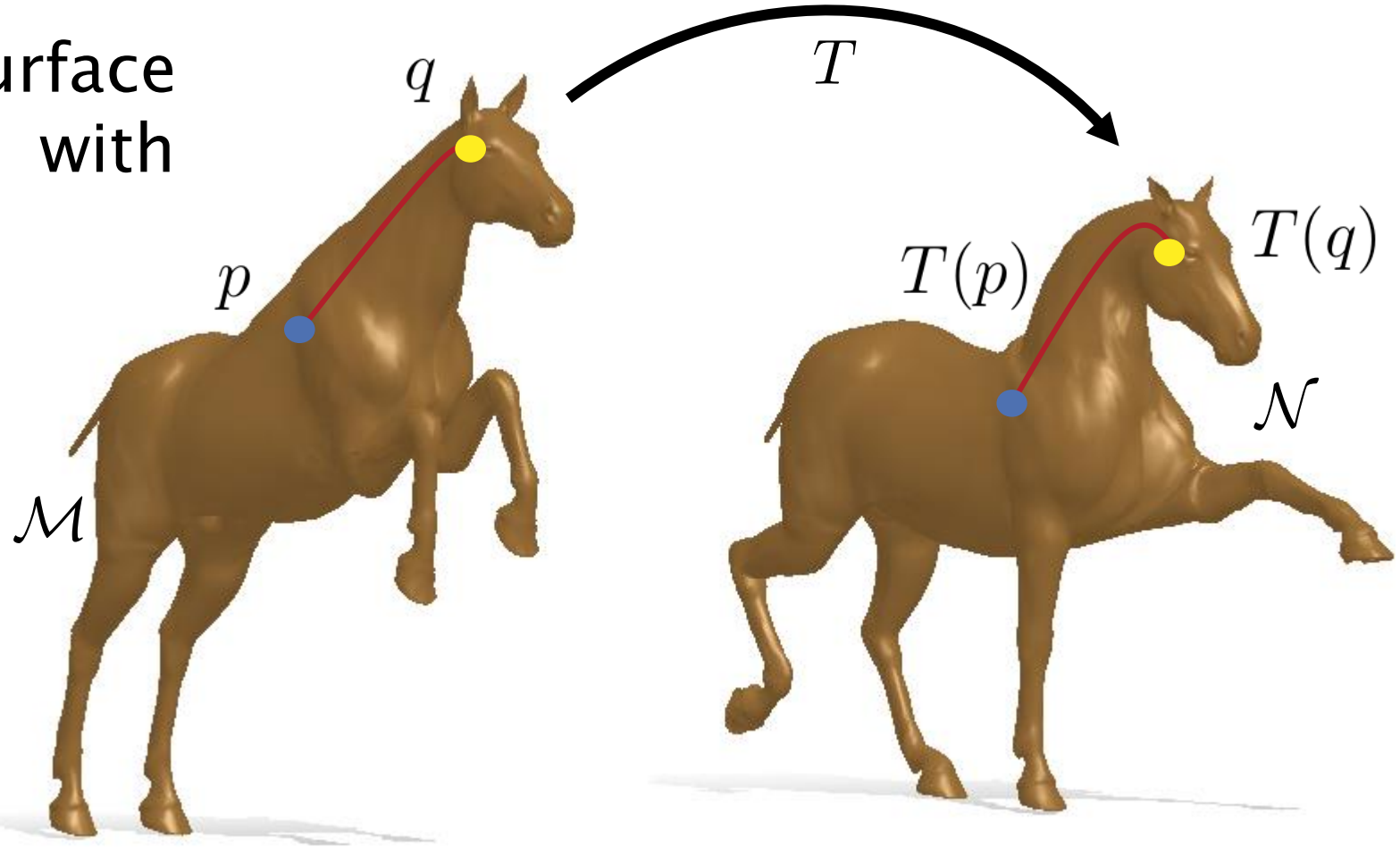
$$\boldsymbol{\Phi}_{\mathcal{M}}\ s.t.\ \boldsymbol{\Phi}_{\mathcal{M}}^{\top} \boldsymbol{\Omega}_{\mathcal{M}} \boldsymbol{\Phi}_{\mathcal{M}} = I$$

$$\boldsymbol{\Phi}_{\mathcal{M}}^{\dagger} = \boldsymbol{\Phi}_{\mathcal{M}}^{\top} \boldsymbol{\Omega}_{\mathcal{M}}$$

The synthesis: $f = \boldsymbol{\Phi}_{\mathcal{M}} \boldsymbol{\alpha} = \boldsymbol{\Phi}_{\mathcal{M}} \boldsymbol{\Phi}_{\mathcal{M}}^{\dagger} f$

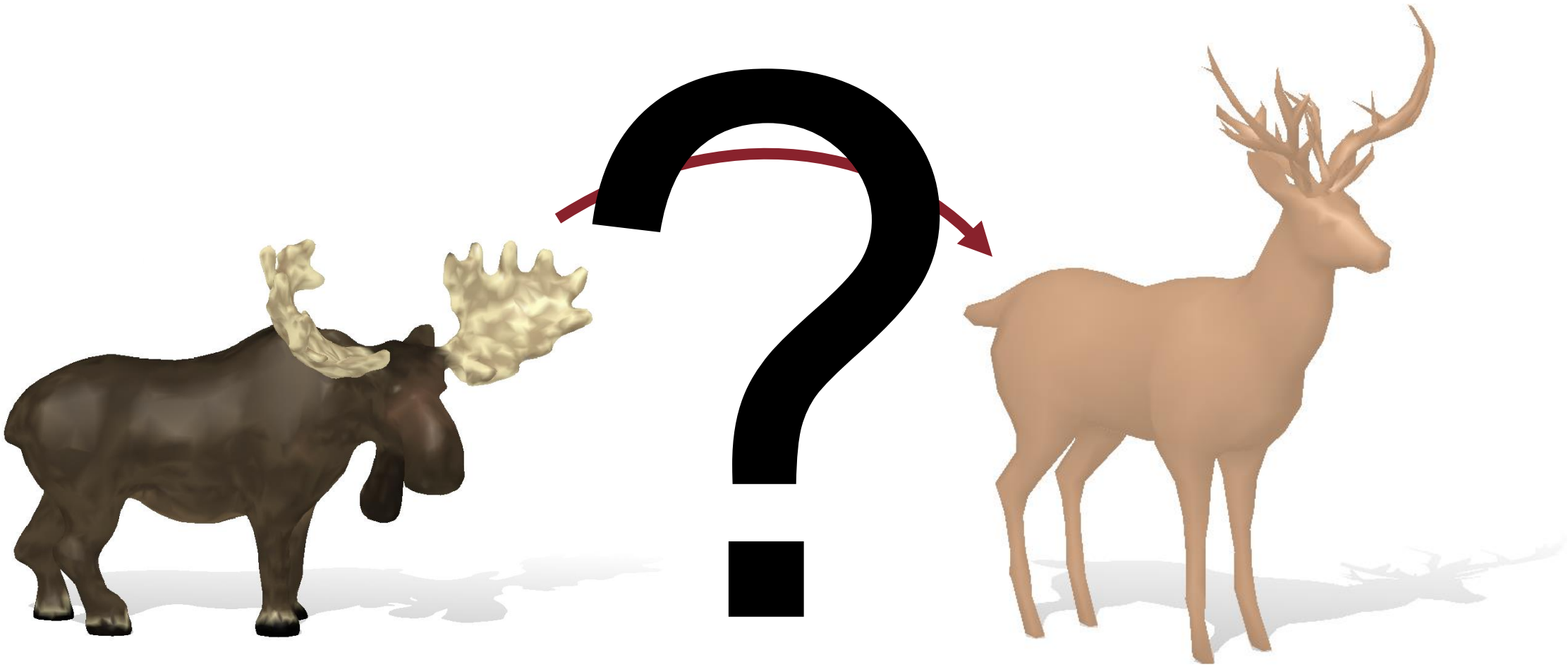
Geodesic distances

The path on the surface connecting p to q with minimum length



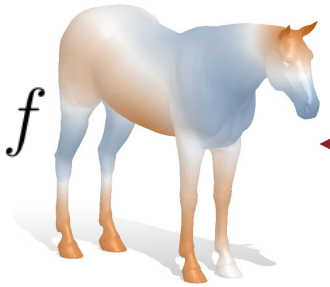
T is an isometry $\iff d_{\mathcal{M}}(p,q) = d_{\mathcal{N}}(T(p),T(q))\ \forall p,q \in \mathcal{M}$

Question 1



Not directly possible! They are 2 different templates

Question 2

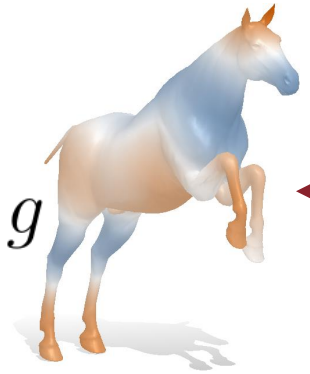


f

Fourier

$$[\alpha_1,\ \alpha_2,\ \alpha_3,\ \alpha_4,\ \alpha_5,\ \ldots\ \alpha_{29},\ \alpha_{30}]^\top = \boldsymbol{a}$$

What is the relation between the two set of coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$?

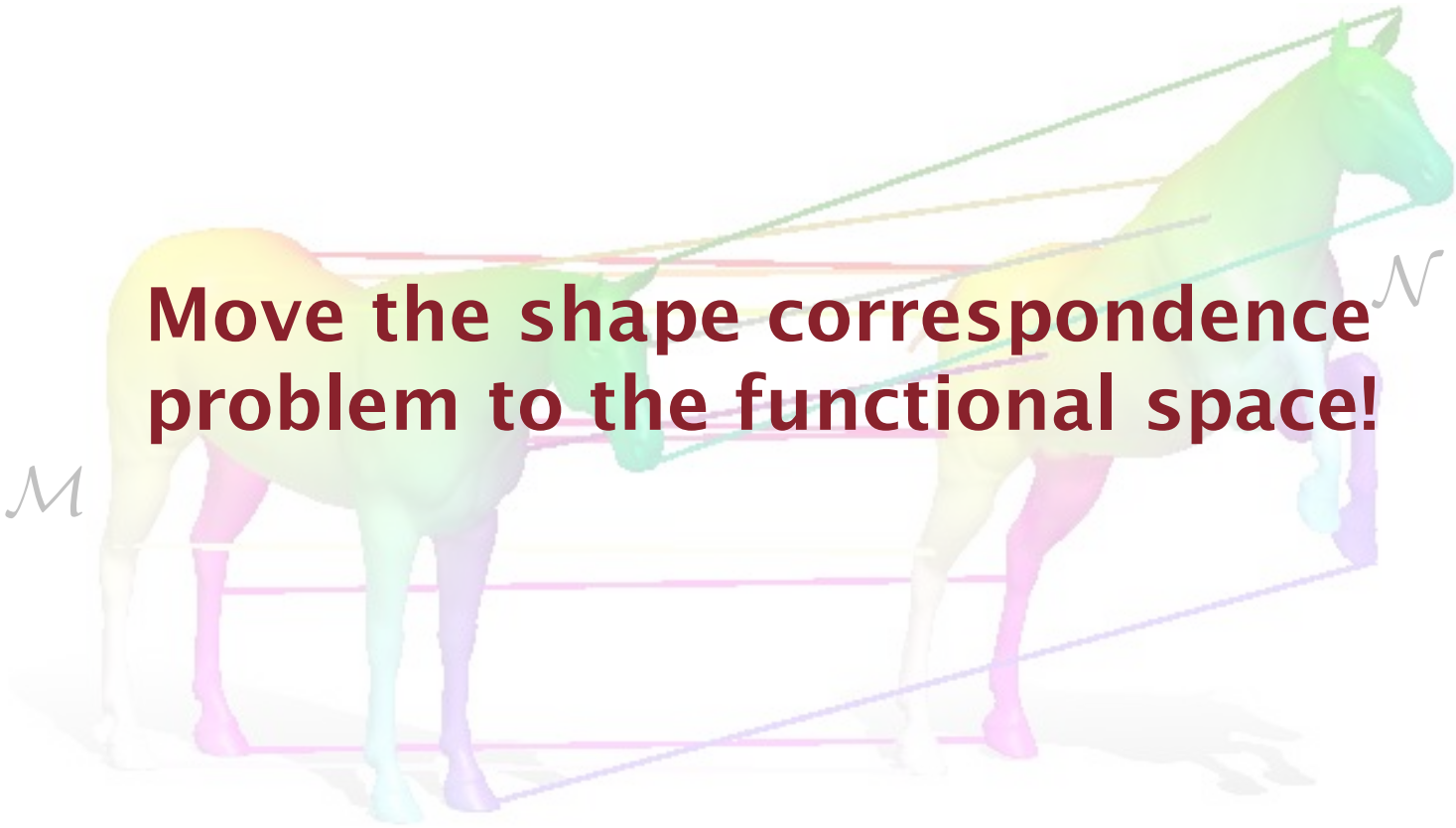


g

Fourier

$$[\beta_1,\ \beta_2,\ \beta_3,\ \beta_4,\ \beta_5,\ \ldots\ \beta_{29},\ \beta_{30}]^\top = \boldsymbol{b}$$

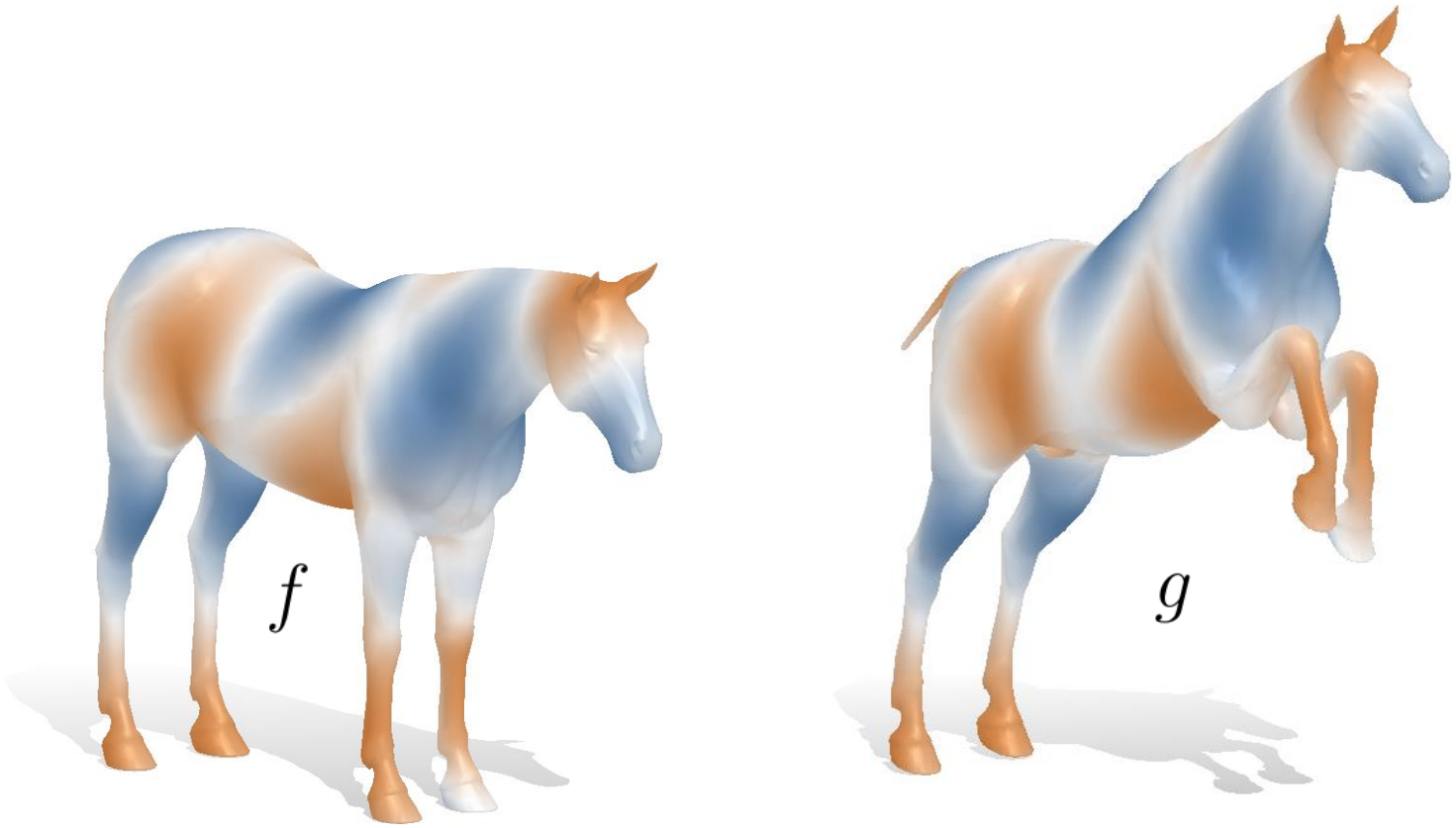
Shape correspondence



Move the shape correspondence problem to the functional space!

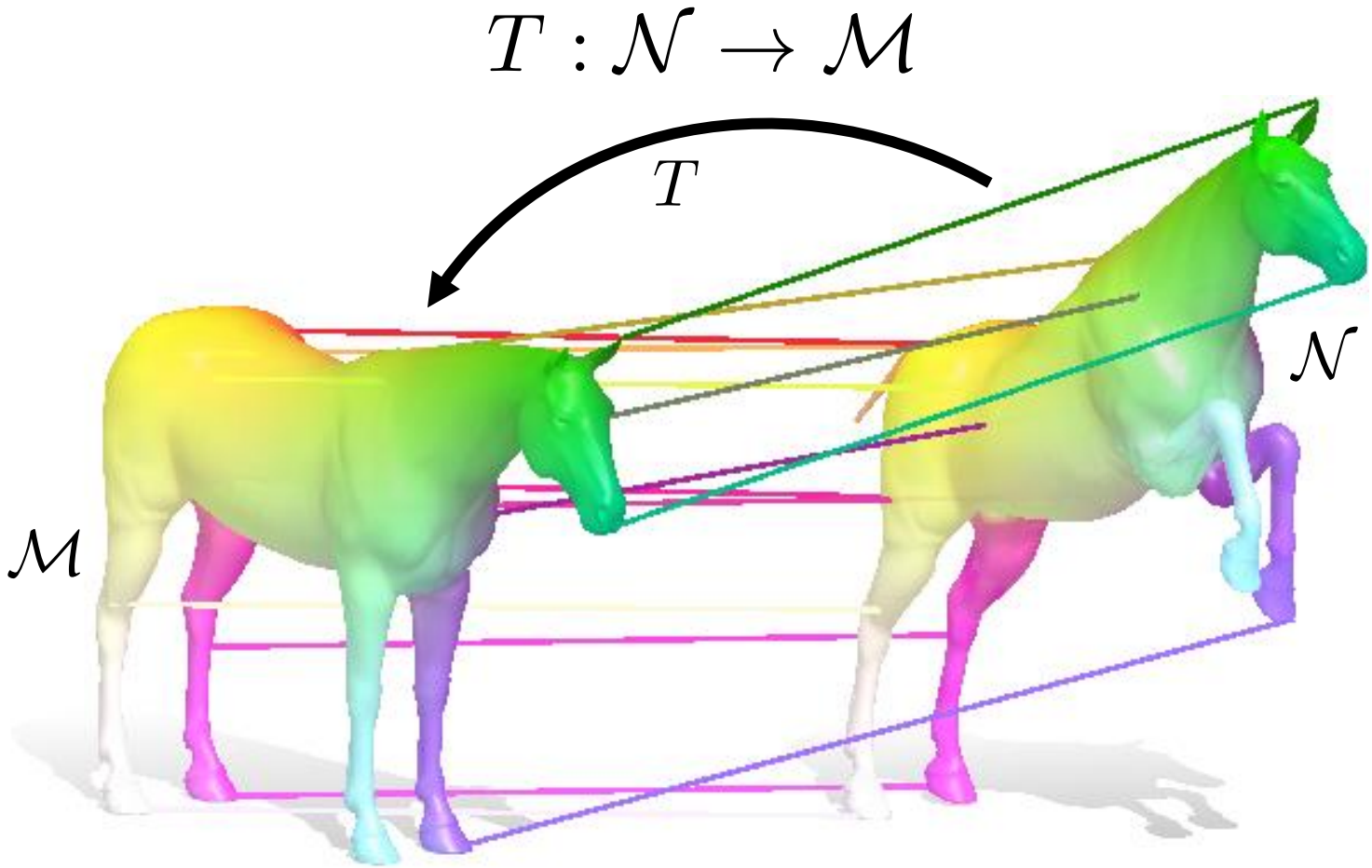
The problem to find a point-to-point map between $\mathcal{M}$ and $\mathcal{N}$

Corresponding functions



corresponding = arise from a point-to-point map

A point-to-point map

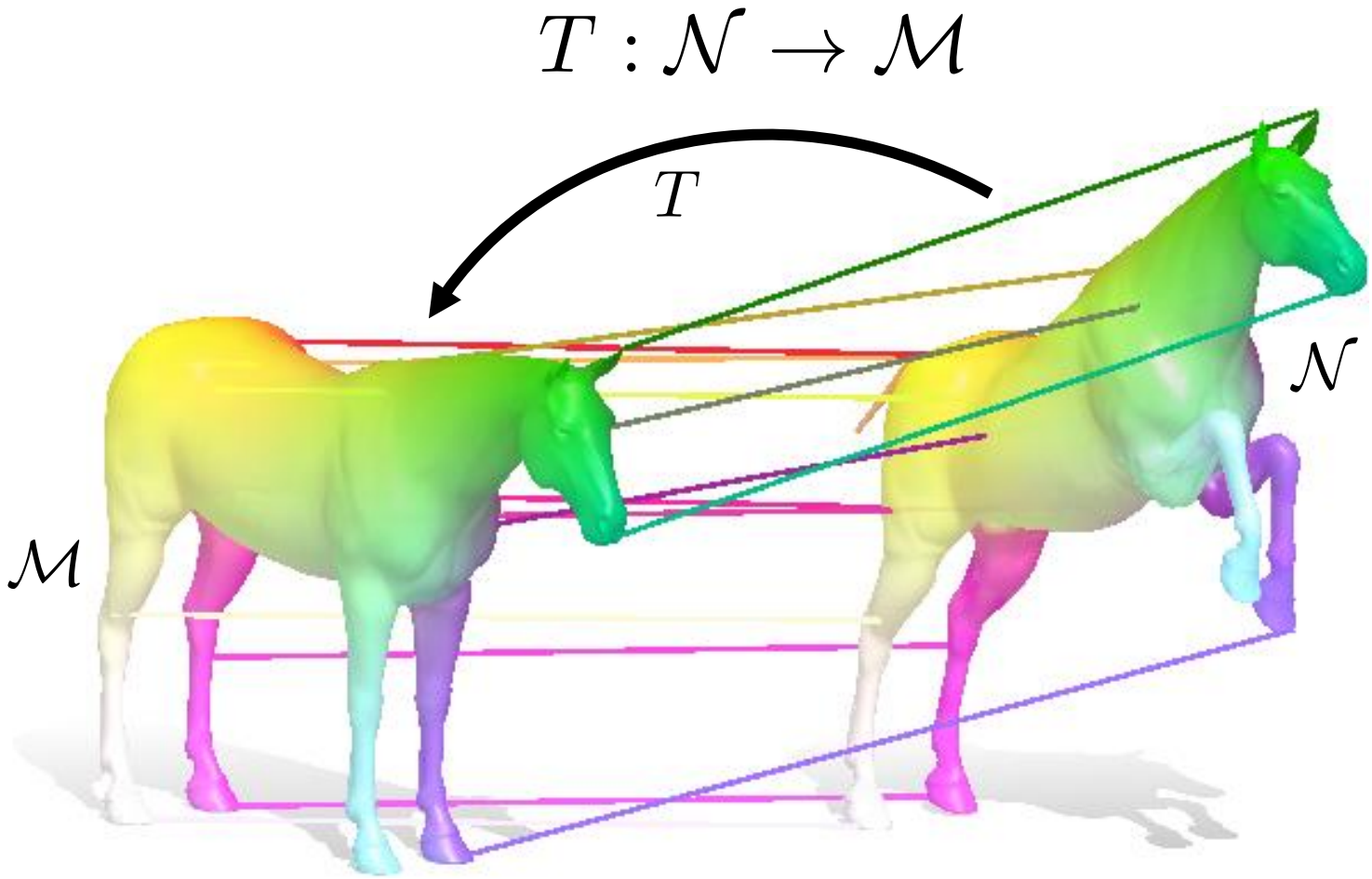


T is a point-to-point map

$$g(p) = f(T(p)) \; \forall p \in \mathcal{N}$$

Functional maps: a flexible representation of maps between shapes, *Ovsjanikov at al.*, 2012

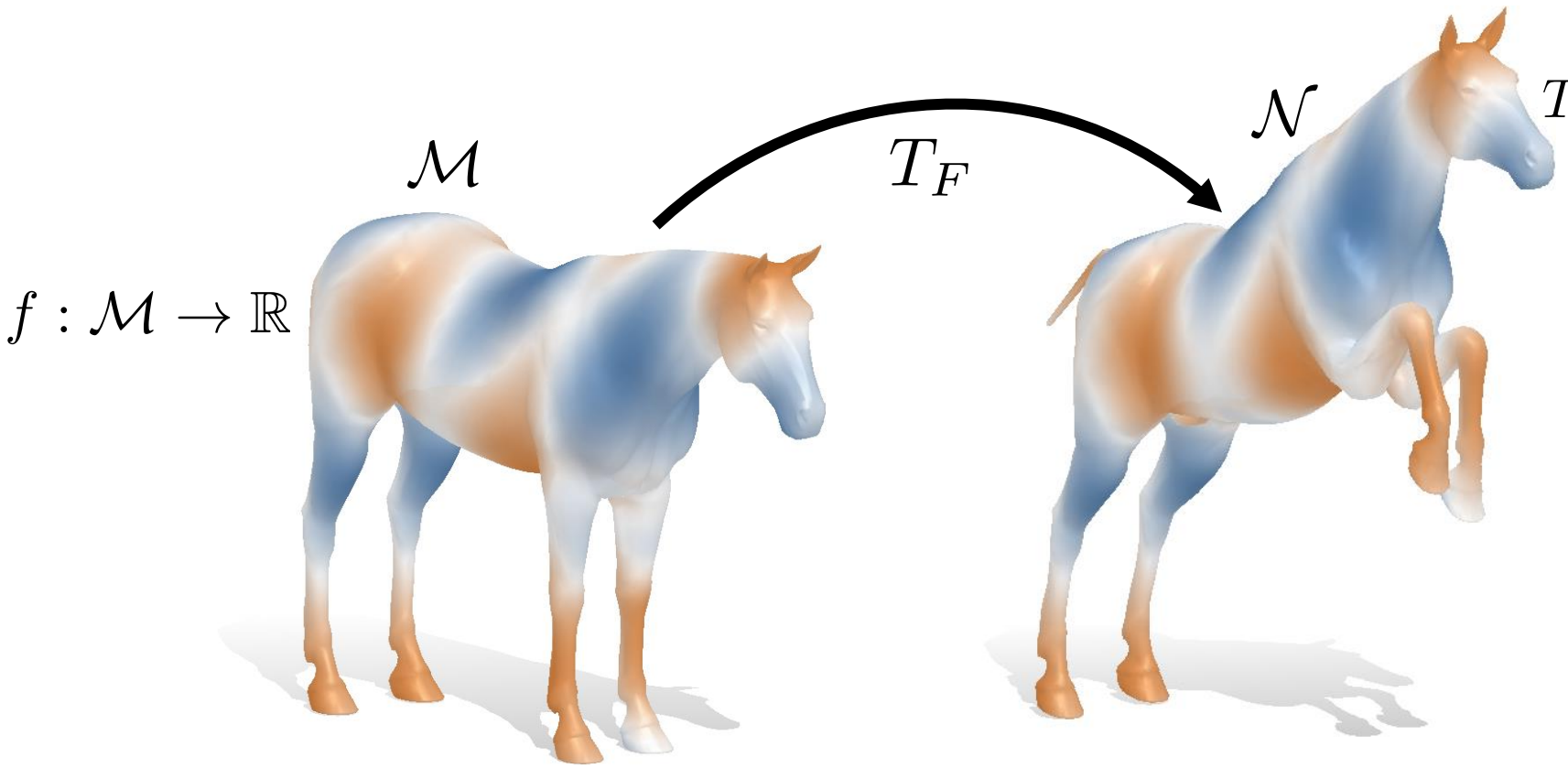
A point-to-point map



write it as a binary matrix $\Pi_{\mathcal{N}\mathcal{M}}(i,j) = 1 \Longleftrightarrow T(i) = j$

Induces a functional map

$$T_F : F(\mathcal{M}) \longrightarrow F(\mathcal{N})$$



$$T_F(f) = g : \mathcal{N} \to \mathbb{R}$$

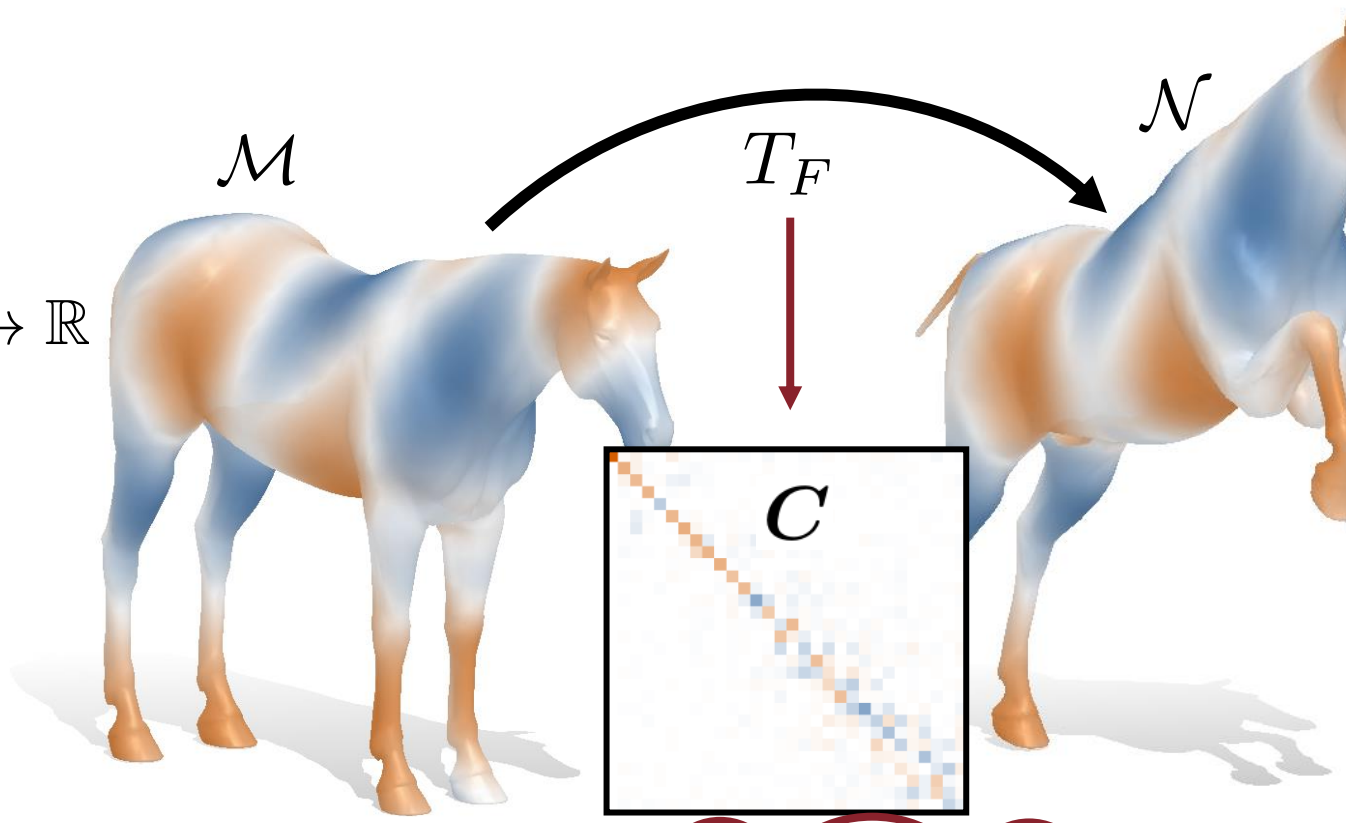
$$g(y) = f(T(y))$$

$$\begin{array}{ccc} \mathcal{N} & \xrightarrow{T} & \mathcal{M} \\ & {\scriptstyle g} \searrow & \downarrow {\scriptstyle f} \\ & & \mathbb{R} \end{array}$$

The transfer is defined as: $g = \Pi_{\mathcal{N}\mathcal{M}} f$.

Induces a functional map

$$T_F : F(\mathcal{M}) \longrightarrow F(\mathcal{N})$$



$f : \mathcal{M} \to \mathbb{R}$

$\boldsymbol{a}$

$T_F(f) = g : \mathcal{N} \to \mathbb{R}$

$g = \Pi_{\mathcal{N}\mathcal{M}} f$

$\boldsymbol{b}$

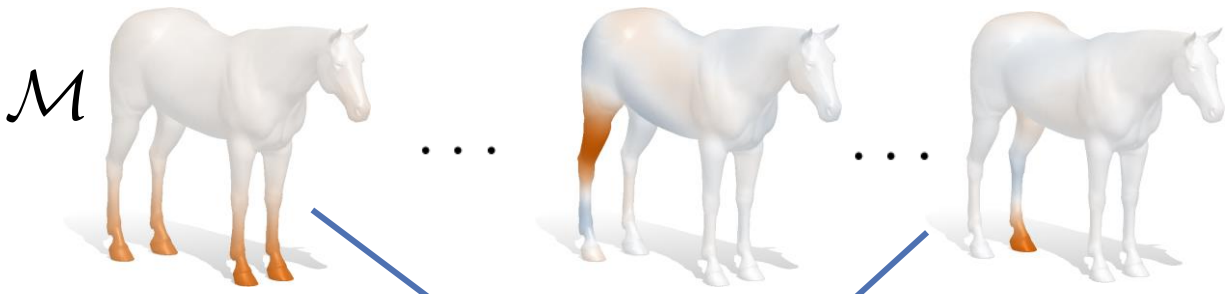
$$C = \Phi_{\mathcal{N}}^{\dagger} \Pi_{\mathcal{N}\mathcal{M}} \Phi_{\mathcal{M}}$$

$$\boldsymbol{b} = \boldsymbol{C}\boldsymbol{a}$$

Exploiting Fourier

Functional maps: a flexible representation of maps between shapes, *Ovsjanikov, Ben-Chen, Solomon, Butscher, Guibas*, SIGGRAPH 2012

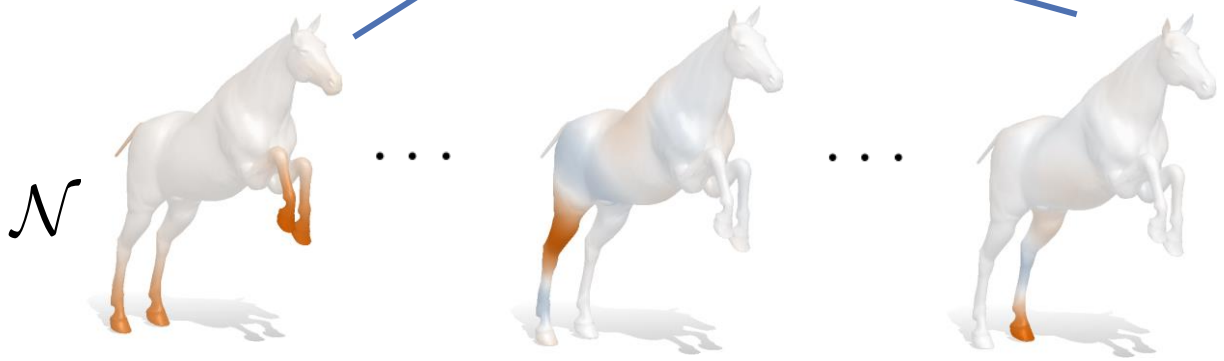
Functional maps estimation



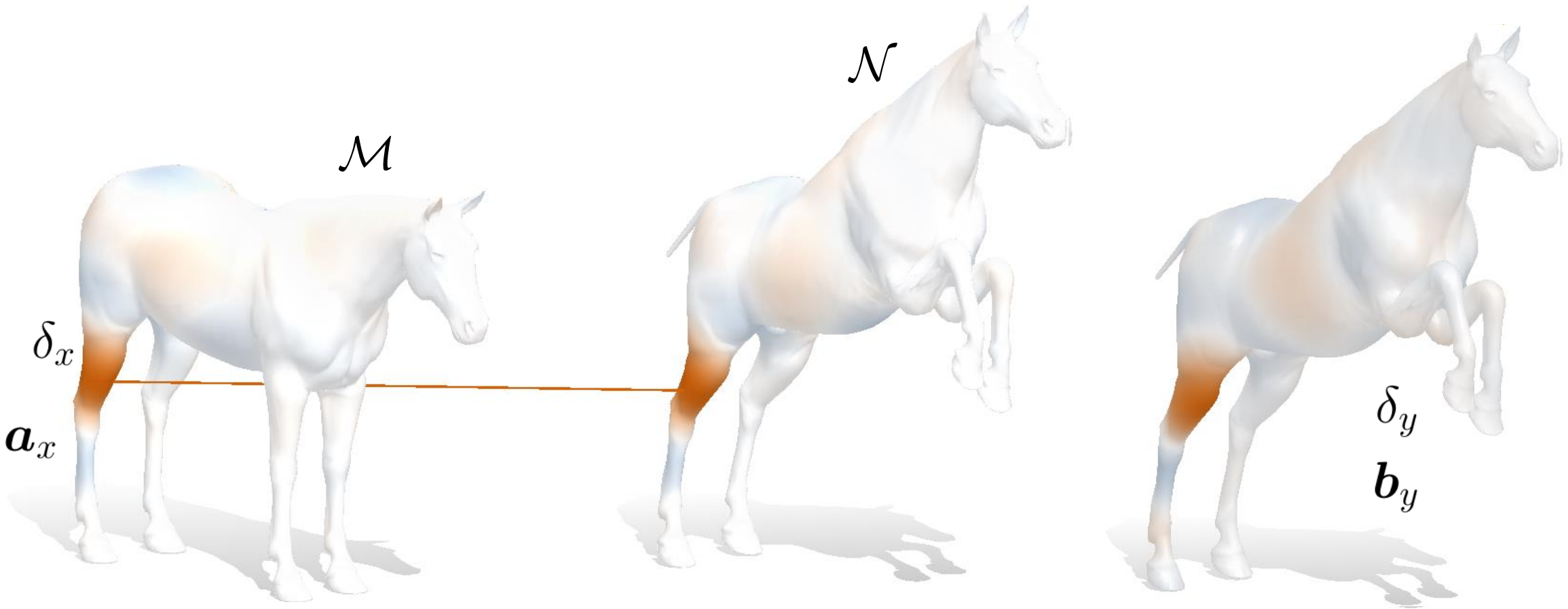
$$\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_q]$$

$$\boldsymbol{C} = \underset{\boldsymbol{C}}{argmin} \|\boldsymbol{C}\boldsymbol{A} - \boldsymbol{B}\|^2 + \mathcal{R}(\boldsymbol{C})$$

$$\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_q]$$



Conversion to a point-to- point map



$$T(x) = \underset{y}{argmin} \|\boldsymbol{b}_y - \boldsymbol{C}\boldsymbol{a}_x\|_2$$

Fmaps pipeline

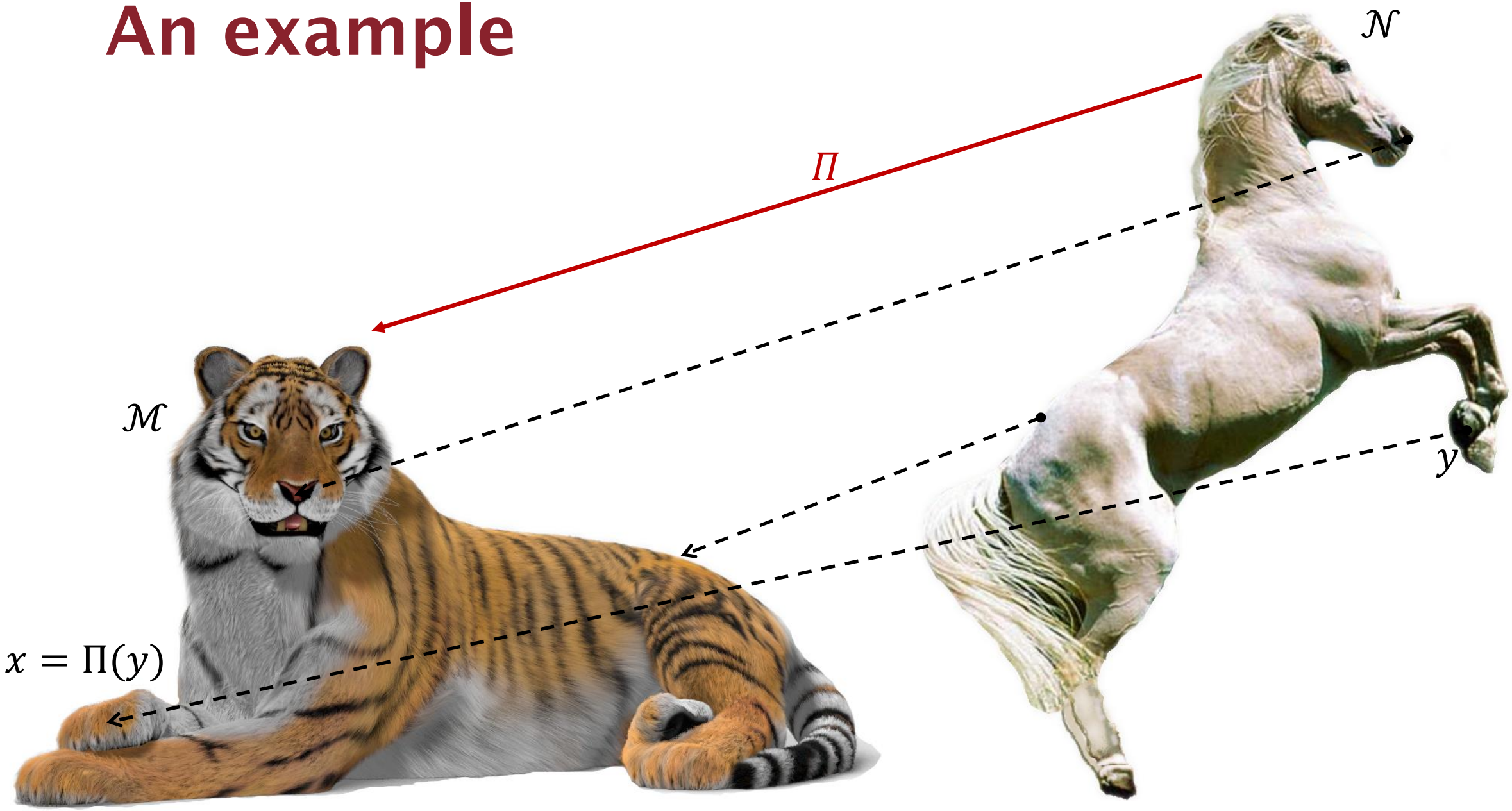
1. Compute the first *k* (~30-100) eigenfunctions of the LBO. Store them in matrices: $\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}$

2. Compute probe functions (e.g., landmarks or descriptors) on $\mathcal{M}, \mathcal{N}$. Express them in $\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}$, as columns of $\boldsymbol{A}$ and $\boldsymbol{B}$

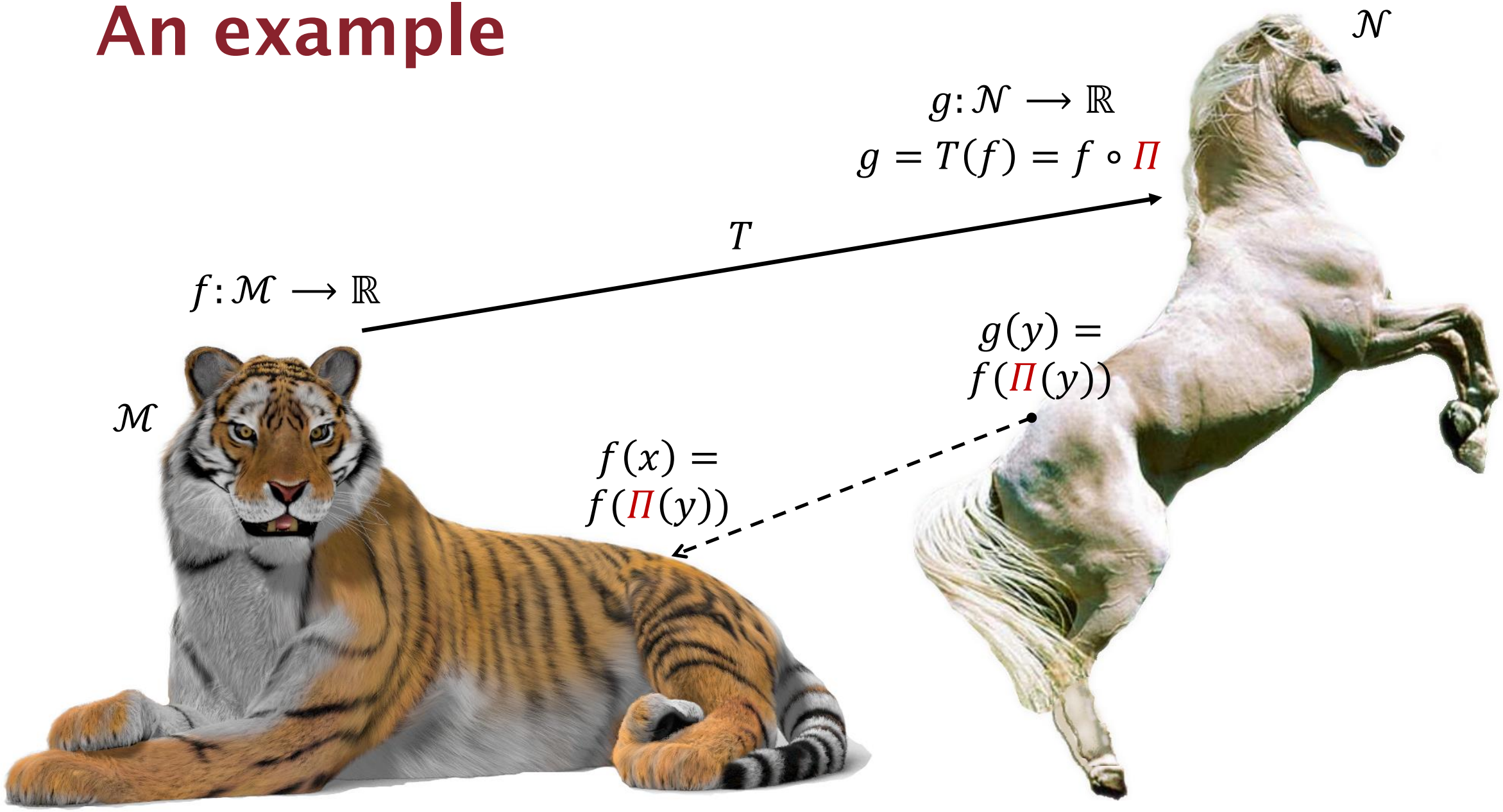
3. Solve $\underset{\boldsymbol{C}}{argmin} \|\boldsymbol{C}\boldsymbol{a} - \boldsymbol{b}\|_F^2 + \mathcal{R}(\boldsymbol{C})$

4. Convert the functional map to a point to point map *T*.

An example



An example


$\mathcal{N}$
$g: \mathcal{N} \longrightarrow \mathbb{R}$
$g = T(f) = f \circ \Pi$
T
$f: \mathcal{M} \longrightarrow \mathbb{R}$
$g(y) = f(\Pi(y))$
$\mathcal{M}$
$f(x) = f(\Pi(y))$

An example

$\mathcal{N}$

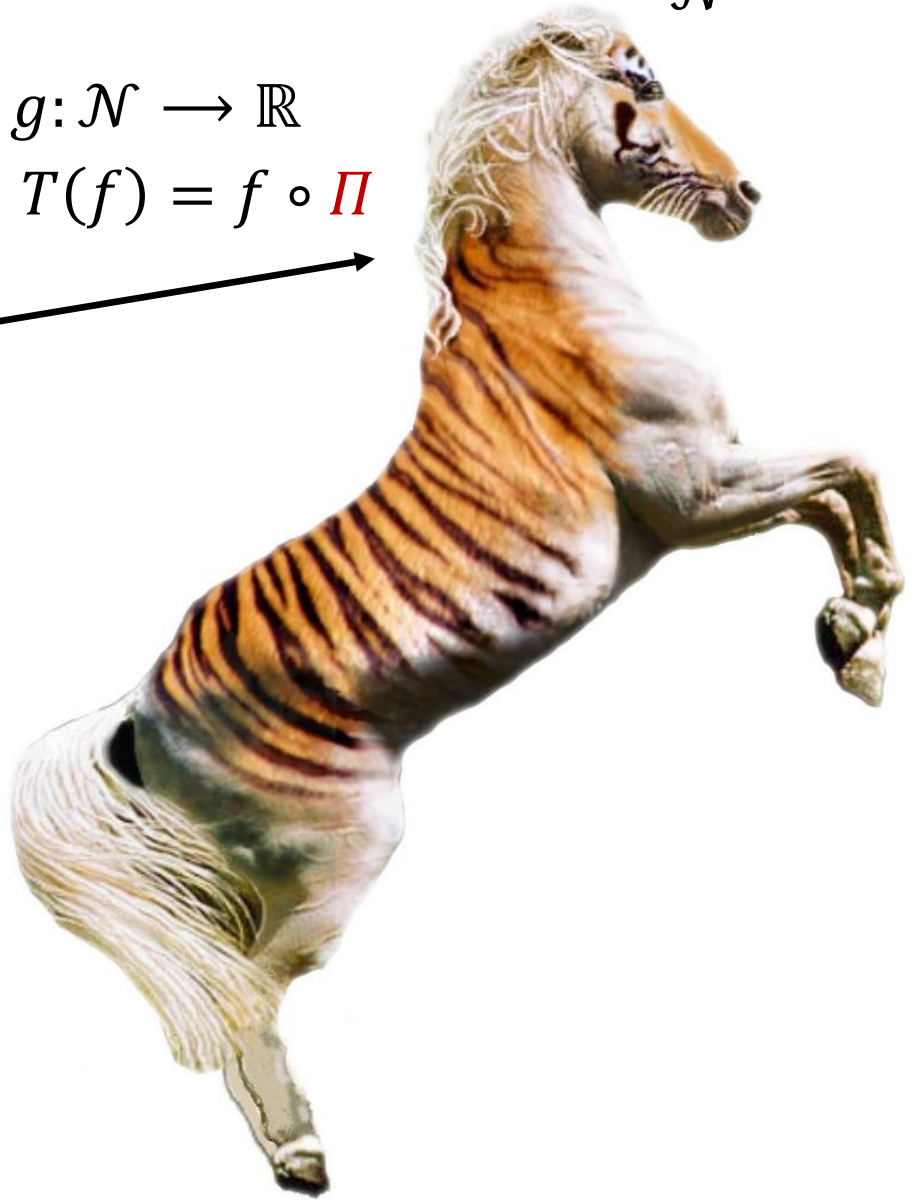
$g: \mathcal{N} \longrightarrow \mathbb{R}$

$g = T(f) = f \circ \Pi$

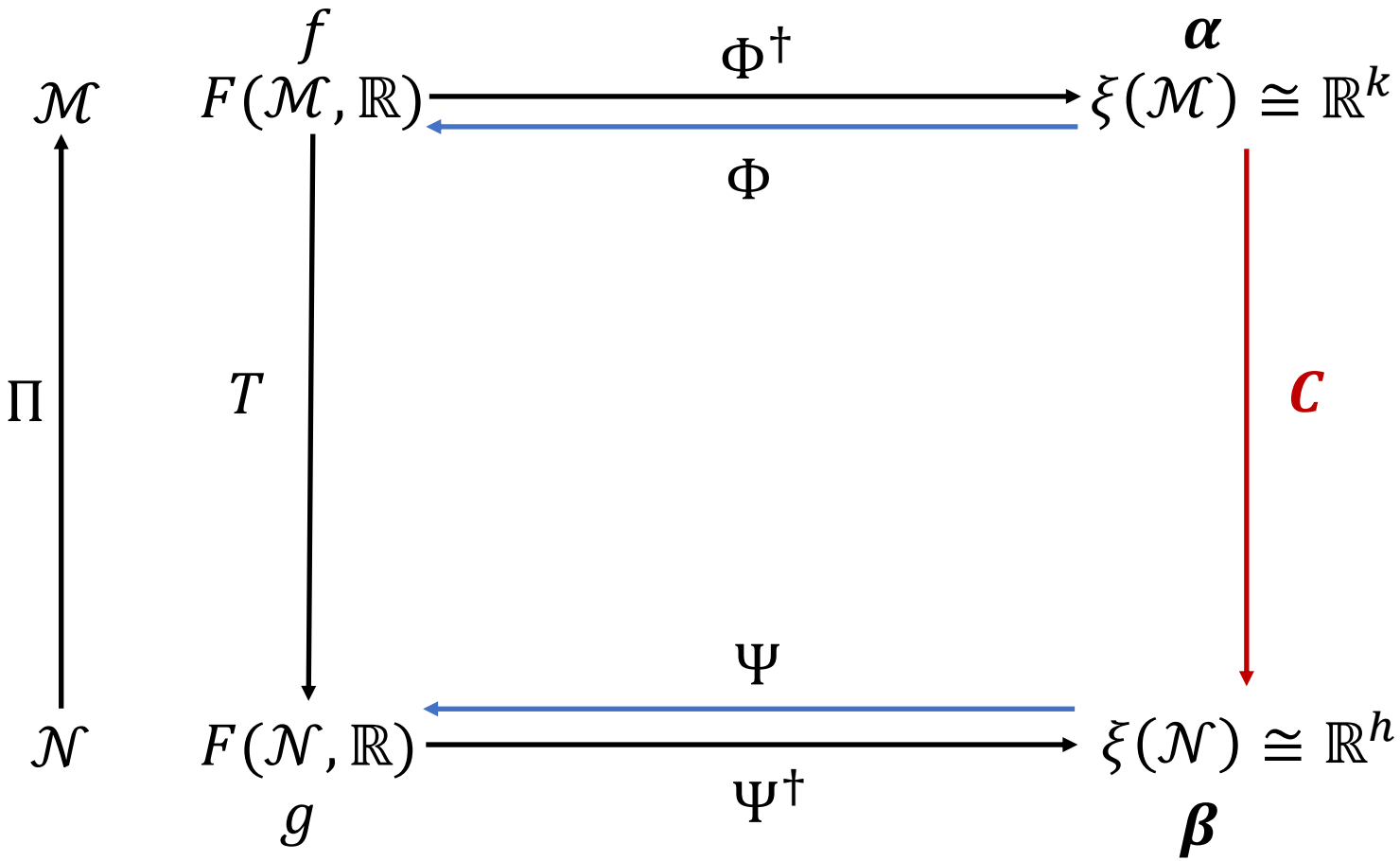
T

$f: \mathcal{M} \longrightarrow \mathbb{R}$

$\mathcal{M}$



A Fourier view



$$g = T(f) = f \circ \Pi = mat(\Pi) f$$

Given $\boldsymbol{\alpha} \in \xi(\mathcal{M})$

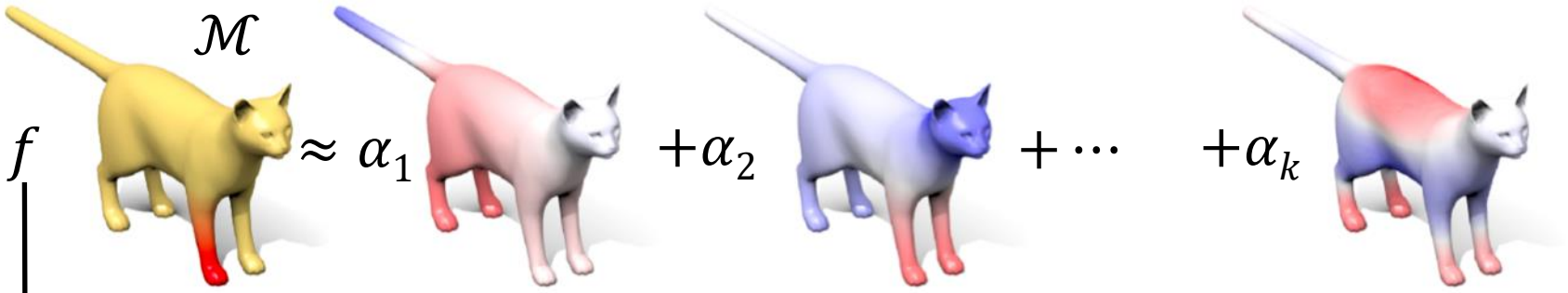
$$\Phi\boldsymbol{\alpha} \in F(\mathcal{M},\mathbb{R})$$

$$mat(\Pi)\Phi\boldsymbol{\alpha} = T(\Phi\boldsymbol{\alpha}) = \Phi\boldsymbol{\alpha} \circ \Pi \in F(\mathcal{N},\mathbb{R})$$

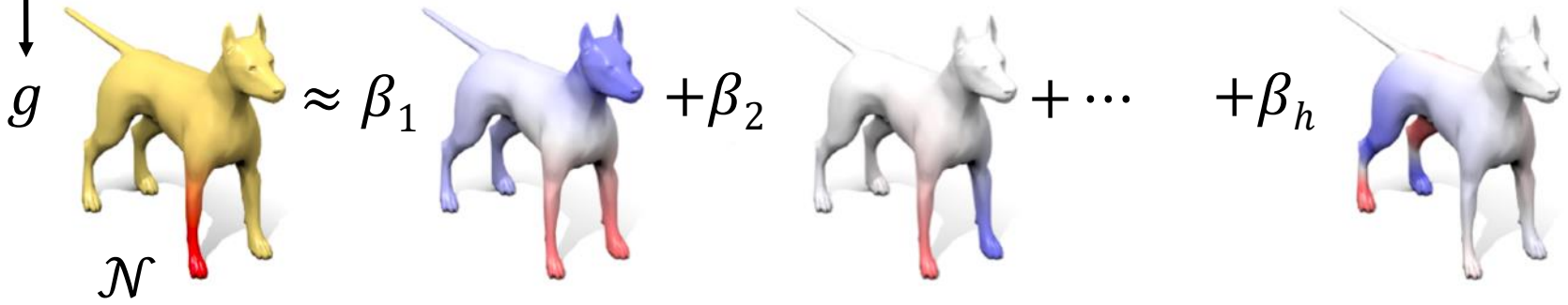
$$\boldsymbol{\beta} = \Psi^\dagger mat(\Pi)\Phi\boldsymbol{\alpha} \in \xi(\mathcal{N})$$

$$\boldsymbol{C} = \Psi^\dagger mat(\Pi)\Phi$$

FMaps = mapping Fourier



T



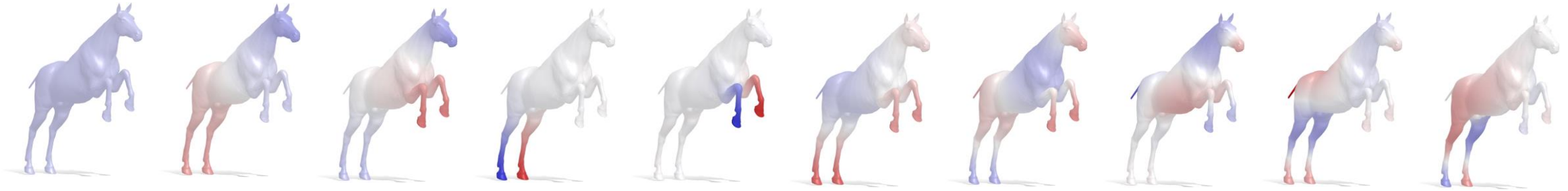
$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_k \end{bmatrix}$$

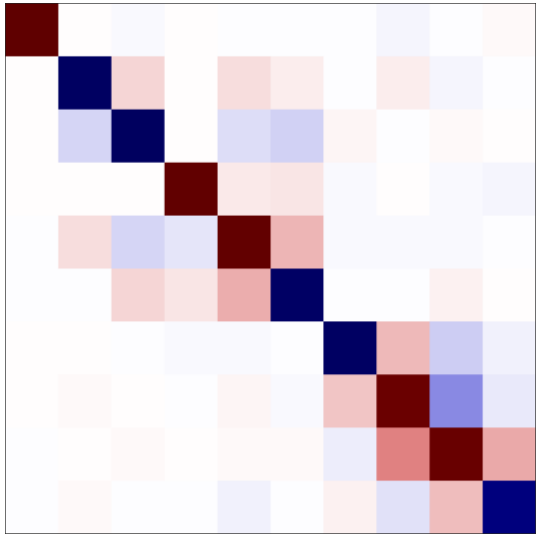
$\boldsymbol{C} \in \mathbb{R}^{h \times k}$ s.t.

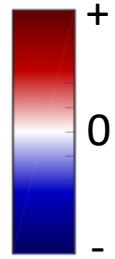
$$\boldsymbol{C\alpha} = \boldsymbol{\beta}$$

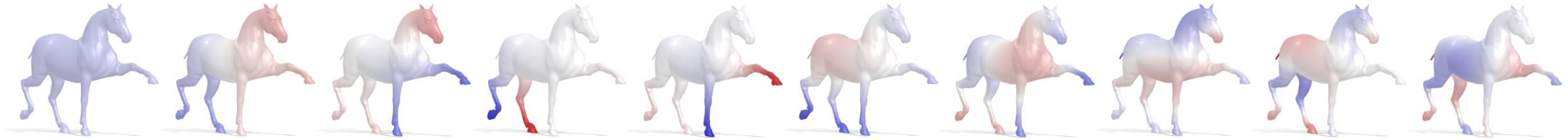
$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_h \end{bmatrix}$$

FMaps = change of the basis

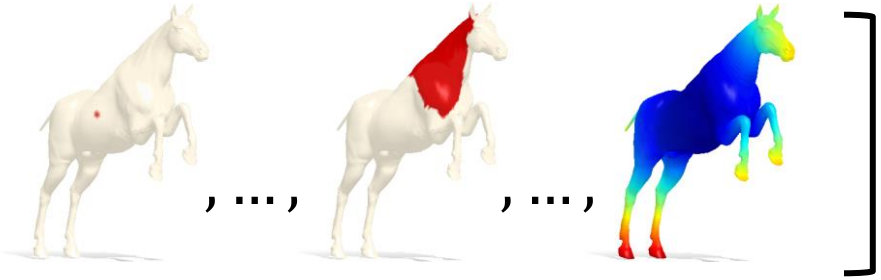




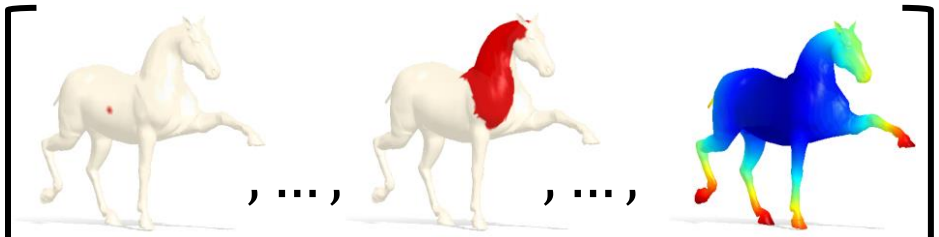




FMaps Optimization

$F = [$  $]$

f_1 (delta) f_l (region) f_q (descriptor)

$G = [$  $]$

g_1 (delta) g_l (region) g_q (descriptor)

$\hat{F} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_l, \ldots, \boldsymbol{\alpha}_q]$

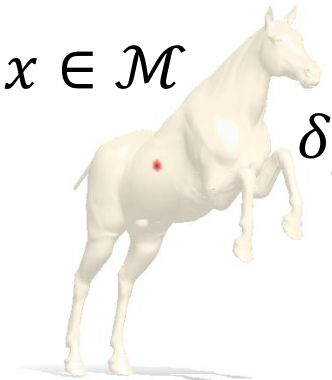
$\hat{G} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_l, \ldots, \boldsymbol{\beta}_q]$

The matrix C should align the coefficients of all the given probe functions

$$\boldsymbol{C} = \underset{\boldsymbol{C} \in \mathbb{R}^{h \times k}}{\operatorname{argmax}} \left\| \boldsymbol{C}\hat{F} - \hat{G} \right\|_2 + \mathcal{R}(\boldsymbol{C})$$

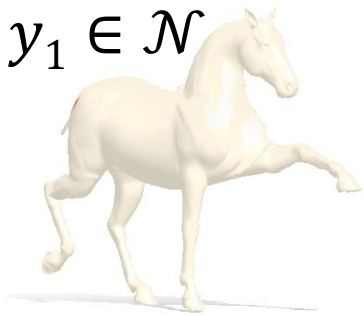
Matching from FMaps

$x \in \mathcal{M}$

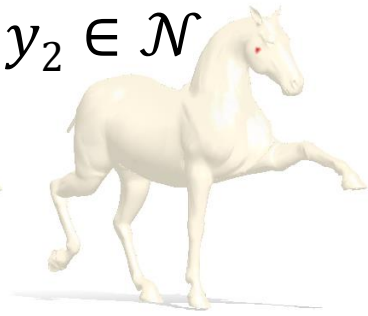


$$\delta_x(p) = \begin{cases} 1 \; if \; x = p \\ 0 \; if \; x \neq p \end{cases}$$

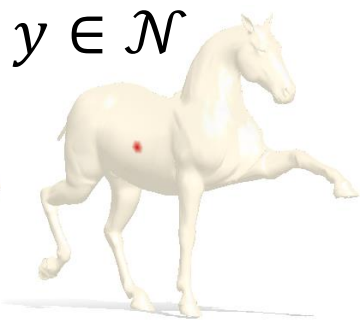
$y_1 \in \mathcal{N}$



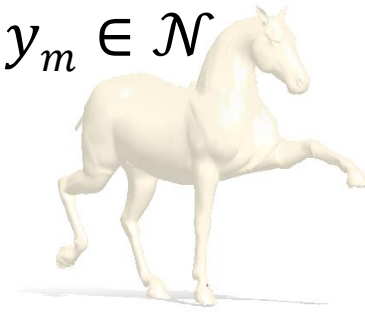
$y_2 \in \mathcal{N}$



$y \in \mathcal{N}$



$y_m \in \mathcal{N}$



$\delta_x \in F(\mathcal{M}, \mathbb{R})$

$\delta_{y_1}, \dots,$

$\delta_{y_2}, \dots,$

$\delta_y, \dots,$

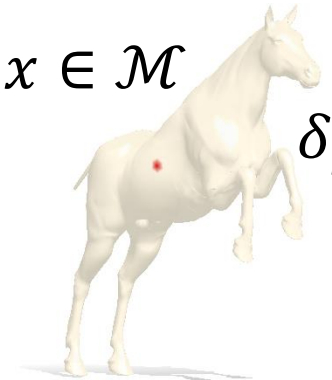
$\delta_{y_m} \in F(\mathcal{N}, \mathbb{R})$

$\boldsymbol{\alpha}_x$ — $\boldsymbol{C}$ maps $\boldsymbol{\alpha}_x$ in the space of coefficients of B → $\mathrm{B} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}, \dots, \boldsymbol{\beta}_m]$

$$y = \Pi \boldsymbol{C}(x) = \underset{y \in \mathcal{N}}{\operatorname{argmin}}(dist_{\mathbb{R}^h}(\boldsymbol{C}\boldsymbol{\alpha}_x - \boldsymbol{\beta}_y))$$

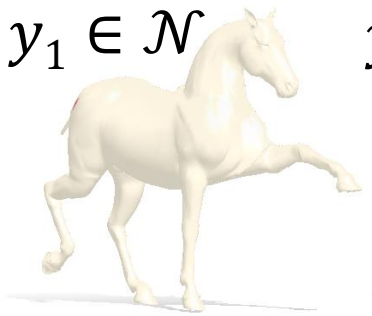
Matching from FMaps

$x \in \mathcal{M}$

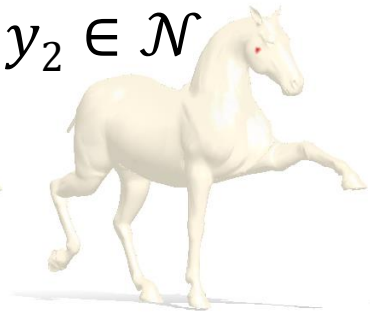


$$\delta_x(p) = \begin{cases} 1 \; if \; x = p \\ 0 \; if \; x \neq p \end{cases}$$

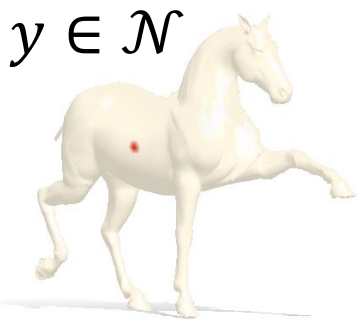
$y_1 \in \mathcal{N}$



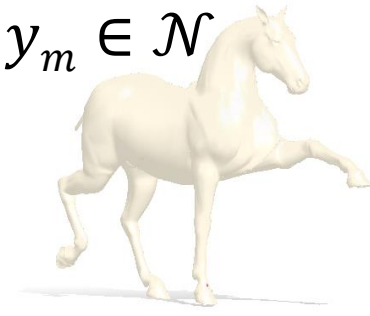
$y_2 \in \mathcal{N}$



$y \in \mathcal{N}$



$y_m \in \mathcal{N}$



$\delta_x \in F(\mathcal{M}, \mathbb{R})$

$\delta_{y_1}, \dots,$

$\delta_{y_2}, \dots,$

$\delta_y, \dots,$

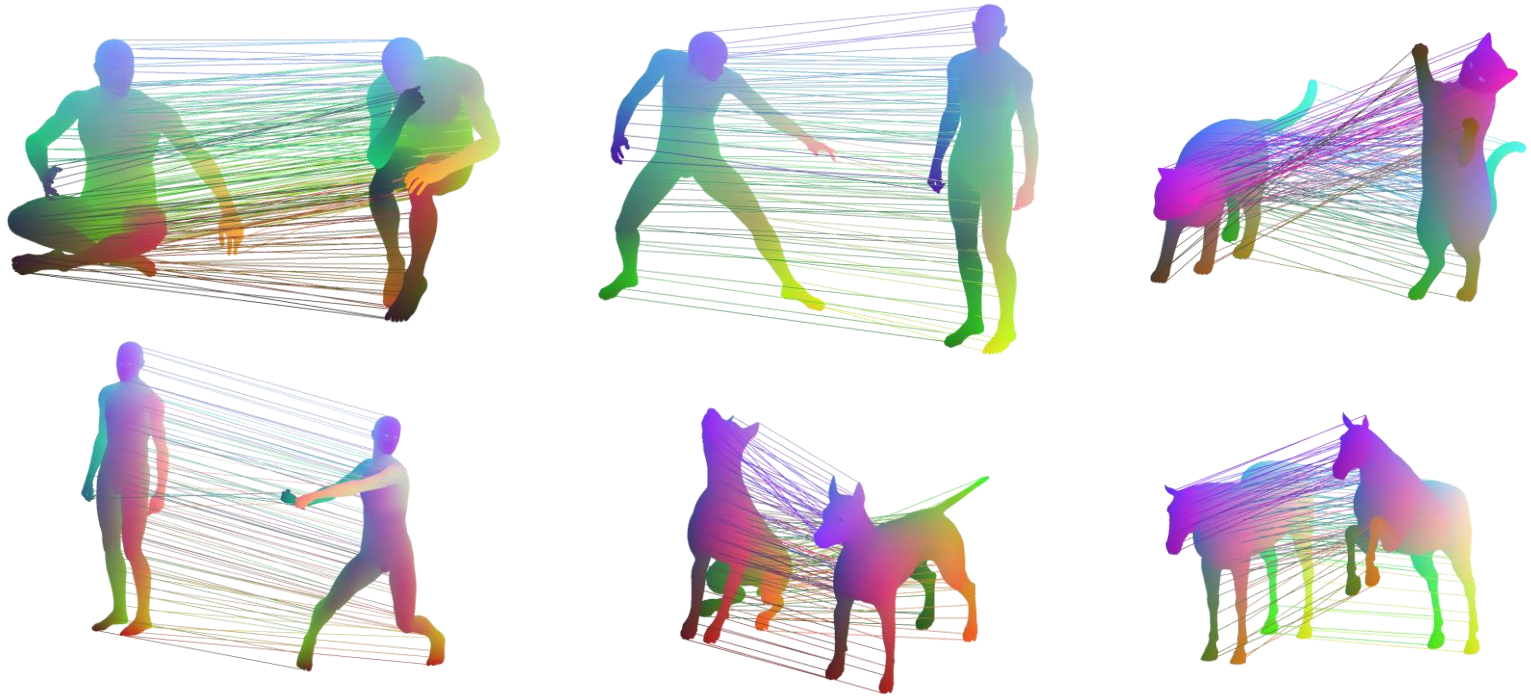
$\delta_{y_m} \in F(\mathcal{N}, \mathbb{R})$

$\boldsymbol{\alpha}_x$ — $\boldsymbol{C}$ maps $\boldsymbol{\alpha}_x$ in the space of coefficients of B → $\mathrm{B} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}, \dots, \boldsymbol{\beta}_m]$

$$y = \Pi \boldsymbol{C}(x) = \underset{y \in \mathcal{N}}{\operatorname{argmin}}(dist_{\mathbb{R}^h}(\boldsymbol{C}\Phi(x) - \Psi(y))) \text{ where } \Phi(x) = [\phi_1(x), \phi_2(x), \dots, \phi_k(x)]$$

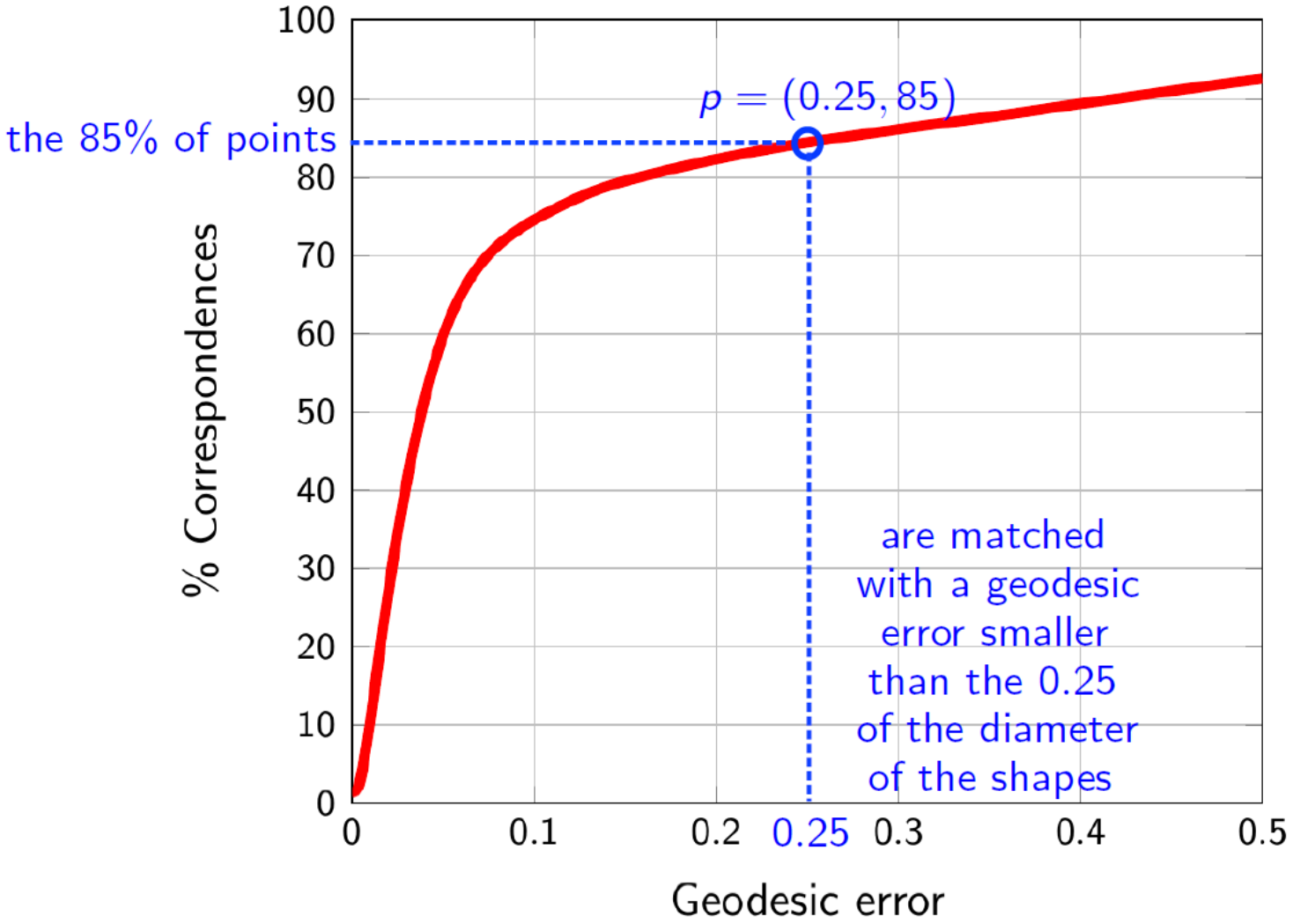
$$\Psi(y) = [\psi_1(y), \psi_2(y), \dots, \psi_h(y)]$$

Fmaps qualitative results



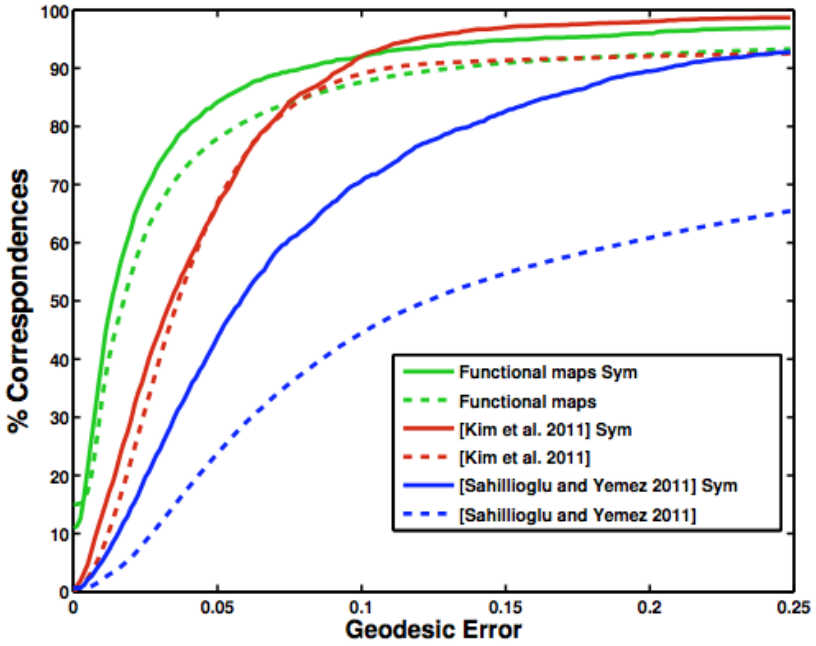
Slide credit M. *Ovsjanikov*

Standard evaluation

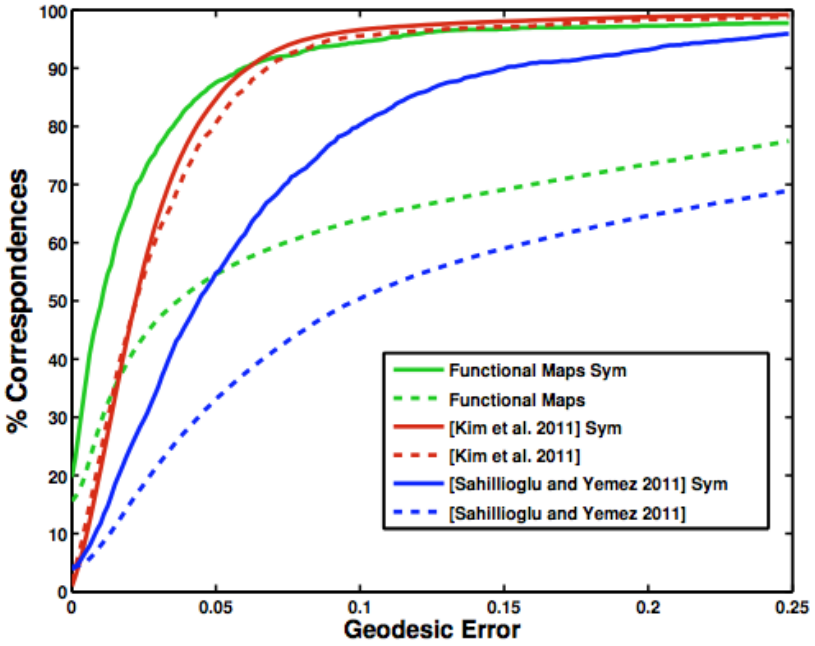


Blended intrinsic maps, V. *Kim, et al.*, 2011

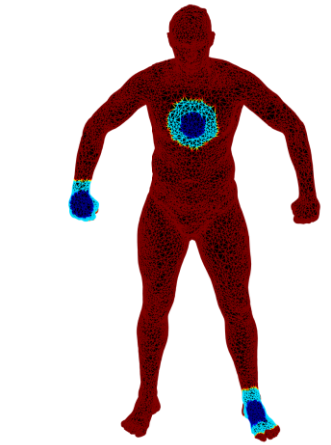
Fmaps quantitative results



SCAPE



TOSCA

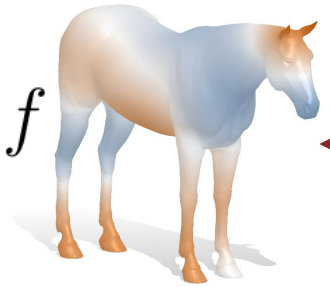


radius 0.025

radius 0.05

Slide credit M. *Ovsjanikov*

Question 2

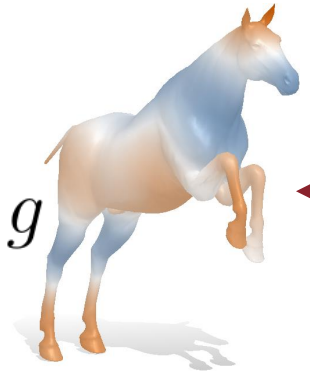


f

Fourier

$$[\alpha_1,\ \alpha_2,\ \alpha_3,\ \alpha_4,\ \alpha_5,\ \ldots\ \alpha_{29},\ \alpha_{30}]^\top = \boldsymbol{a}$$

What is the relation between the two set of coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$?

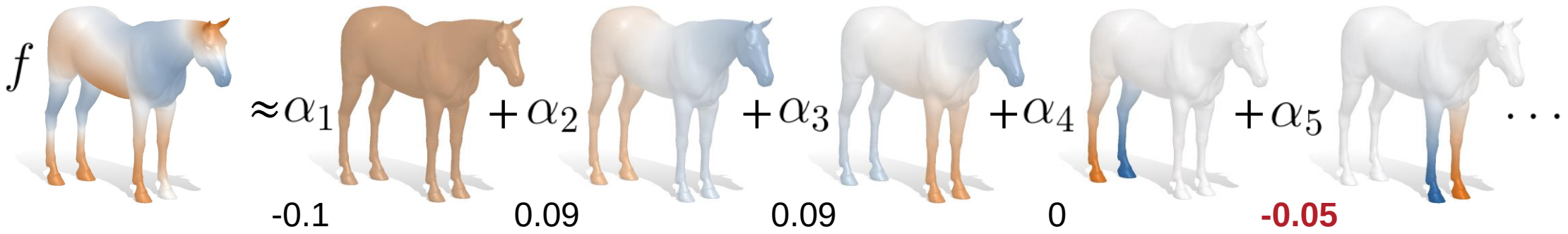


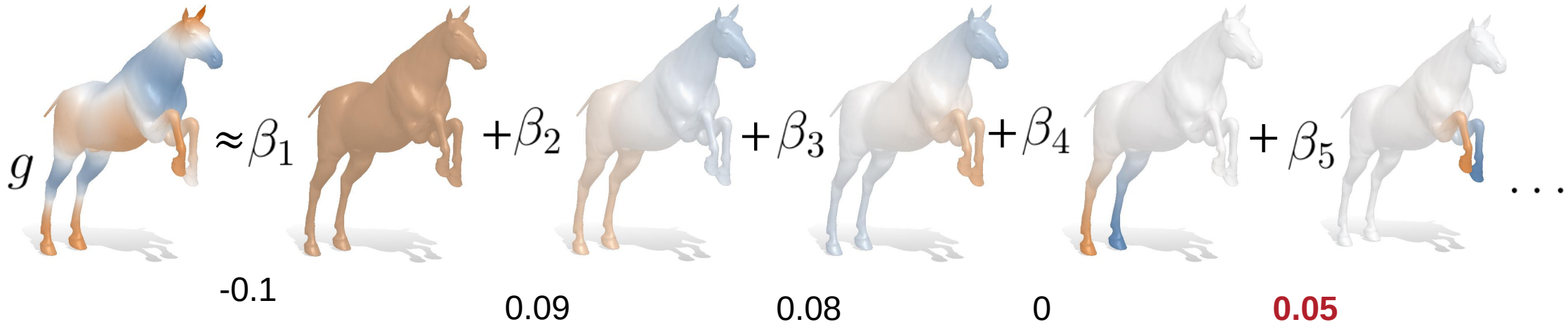
g

Fourier

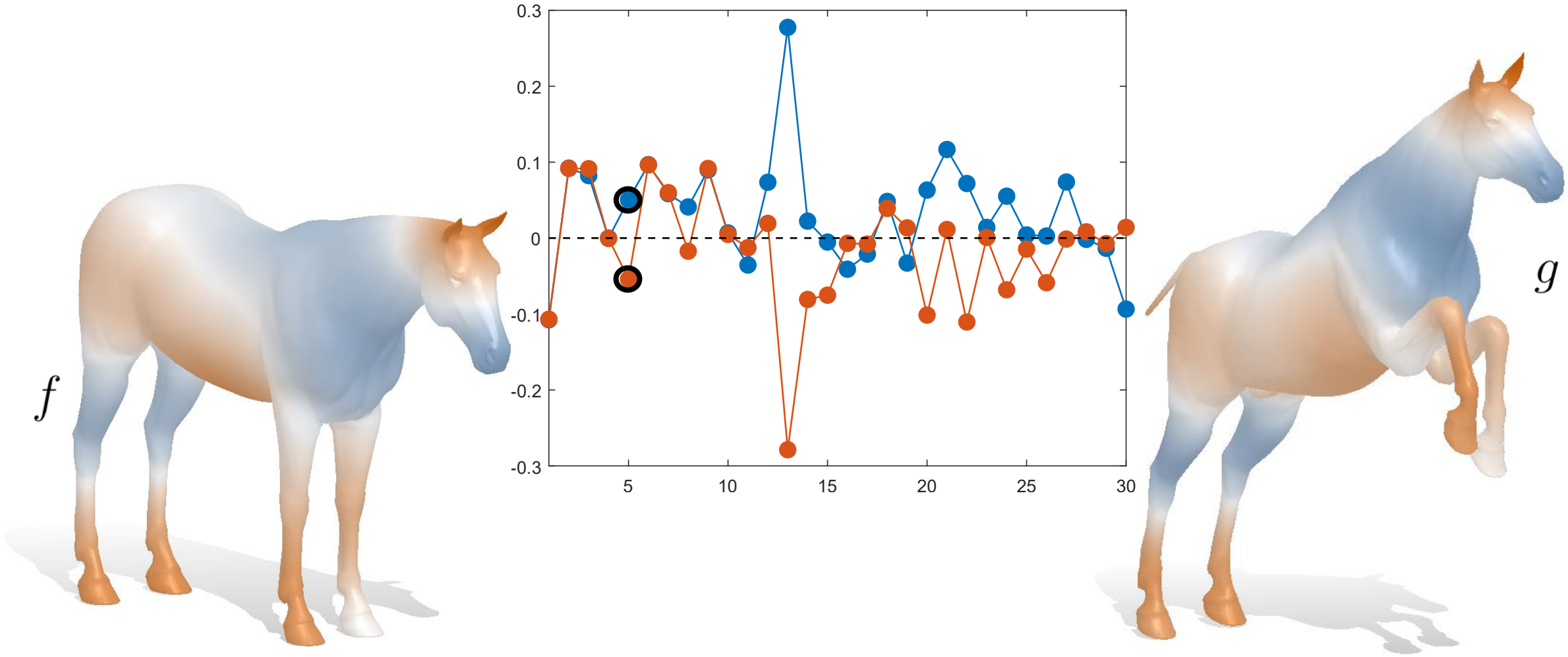
$$[\beta_1,\ \beta_2,\ \beta_3,\ \beta_4,\ \beta_5,\ \ldots\ \beta_{29},\ \beta_{30}]^\top = \boldsymbol{b}$$

Fourier analysis and synthesis

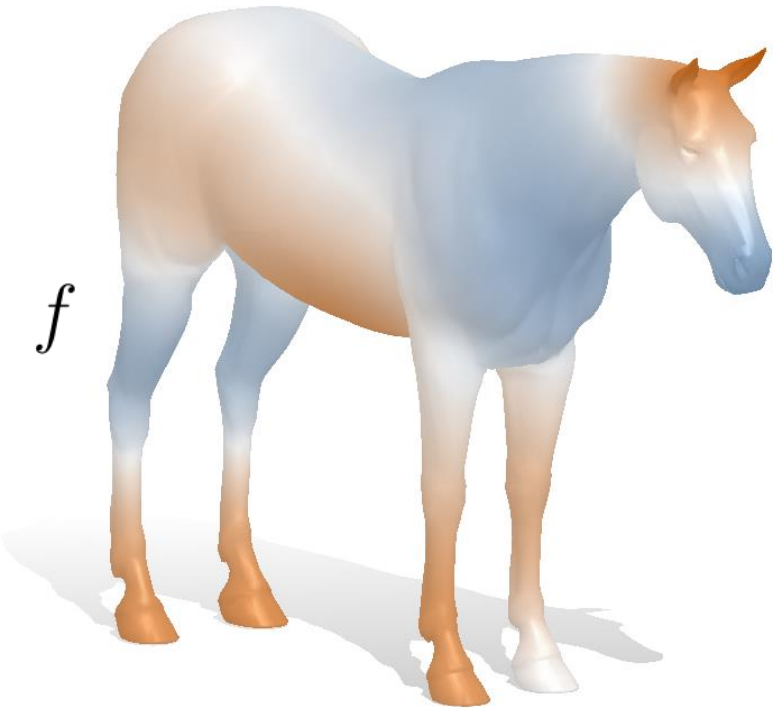




Functions on 2 different domains

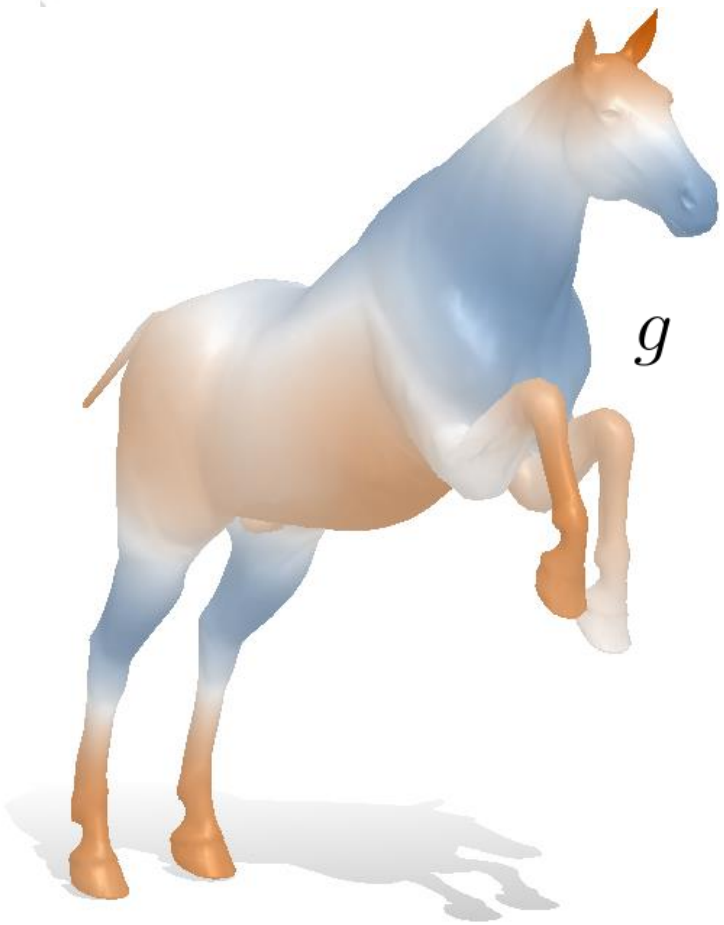


2 different set of coefficients

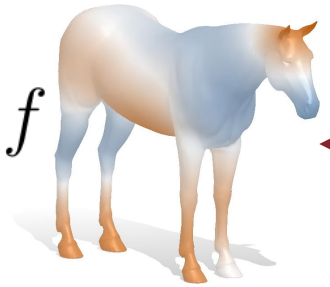


-0.1	α_1
0.09	α_2
0.09	α_3
0	α_4
-0.05	α_5
0.1	⋮
0.06	
⋮	

β_1	-0.1
β_2	0.09
β_3	0.08
β_4	0
β_5	0.05
⋮	0.1
	0.06
	⋮



How to compare them



f

Fourier

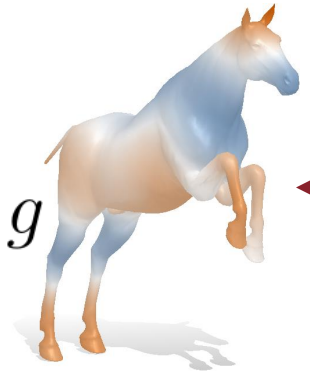
Analysis

$$[\alpha_1,\ \alpha_2,\ \alpha_3,\ \alpha_4,\ \alpha_5,\ \ldots\ \alpha_{29},\ \alpha_{30}]^\top = \boldsymbol{a}$$

Functional Map

C such that

$$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b}$$



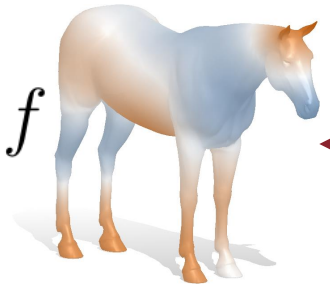
g

Fourier

Analysis

$$[\beta_1,\ \beta_2,\ \beta_3,\ \beta_4,\ \beta_5,\ \ldots\ \beta_{29},\ \beta_{30}]^\top = \boldsymbol{b}$$

Functions on 2 different domains

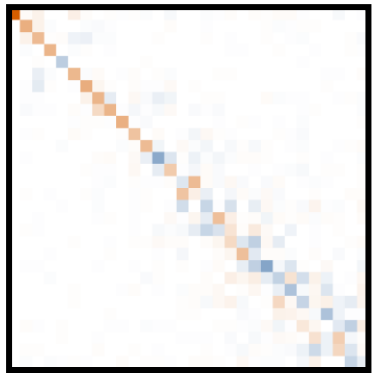


f

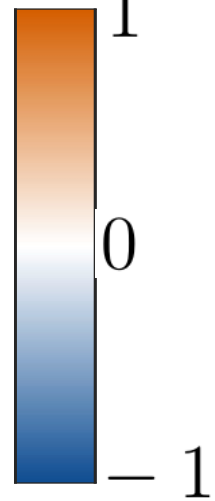
Fourier

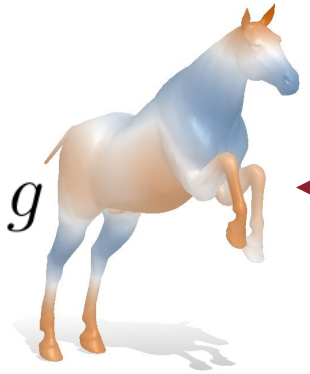
Analysis

$[\alpha_1,\ \alpha_2,\ \alpha_3,\ \alpha_4,\ \alpha_5,\ \ldots\ \alpha_{29},\ \alpha_{30}]^\top = \boldsymbol{a}$



$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{b}$





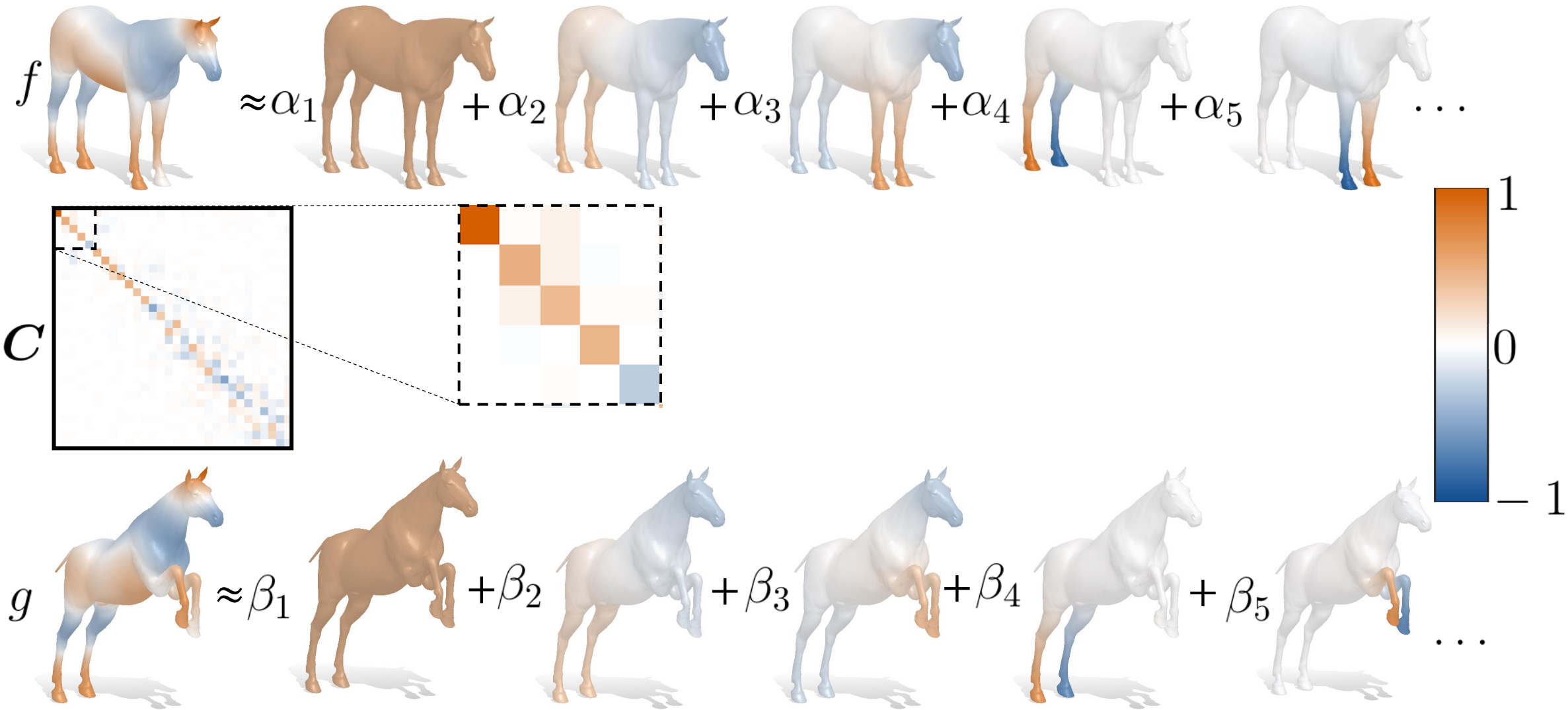
g

Fourier

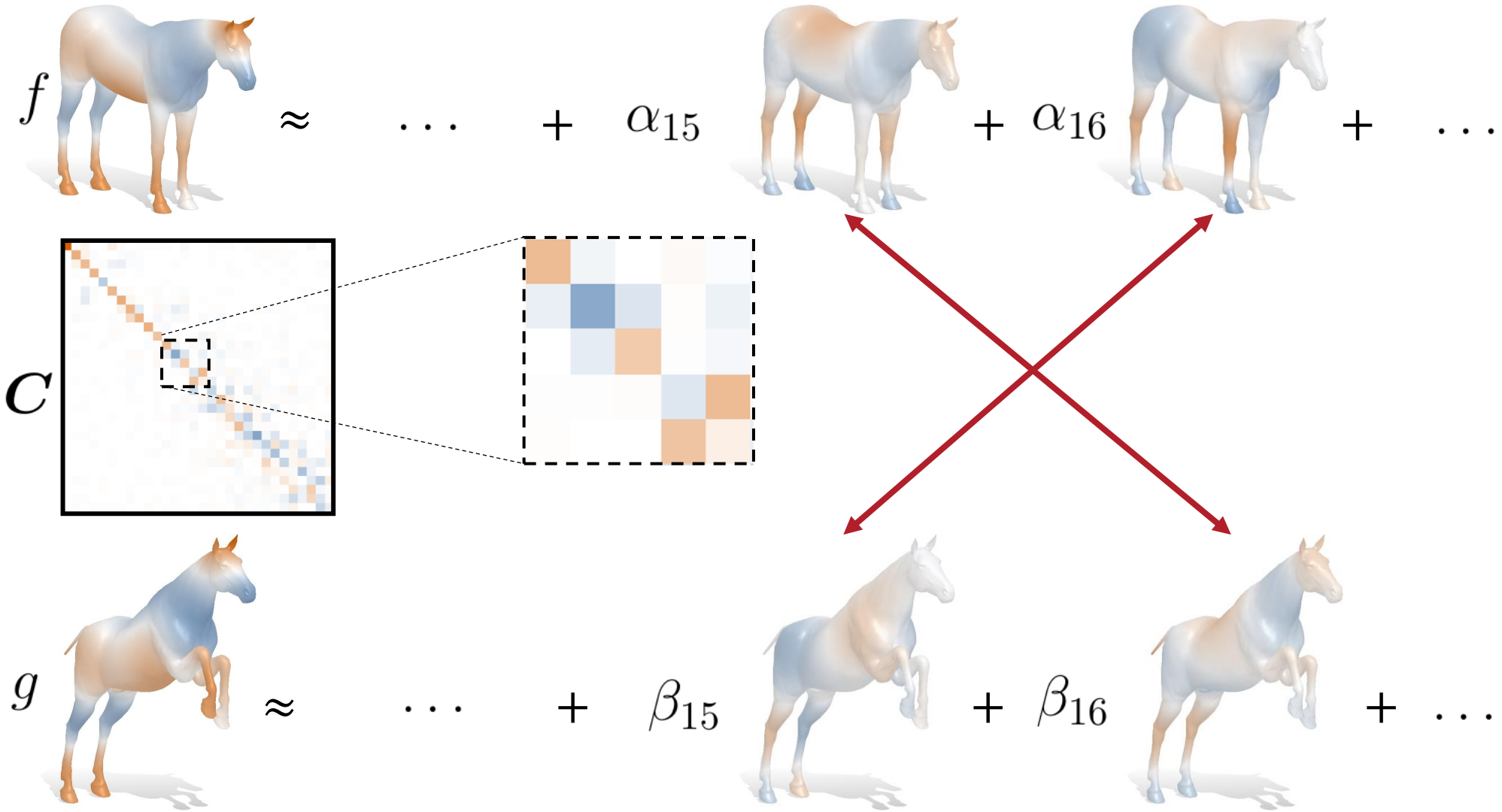
Analysis

$[\beta_1,\ \beta_2,\ \beta_3,\ \beta_4,\ \beta_5,\ \ldots\ \beta_{29},\ \beta_{30}]^\top = \boldsymbol{b}$

Functions on 2 different domains

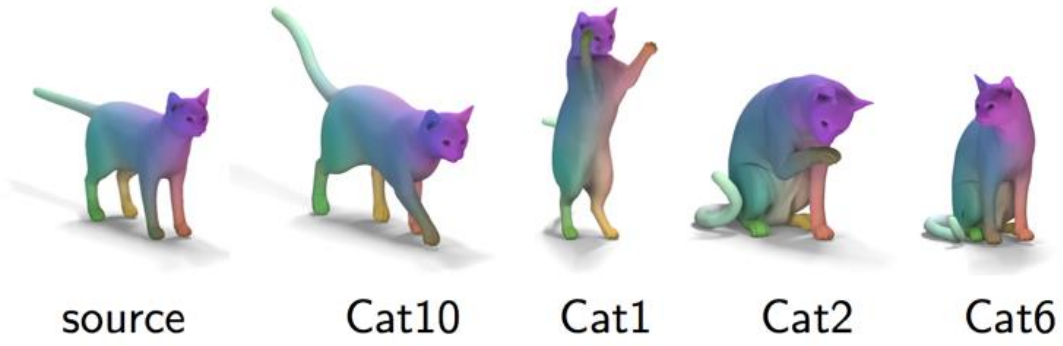


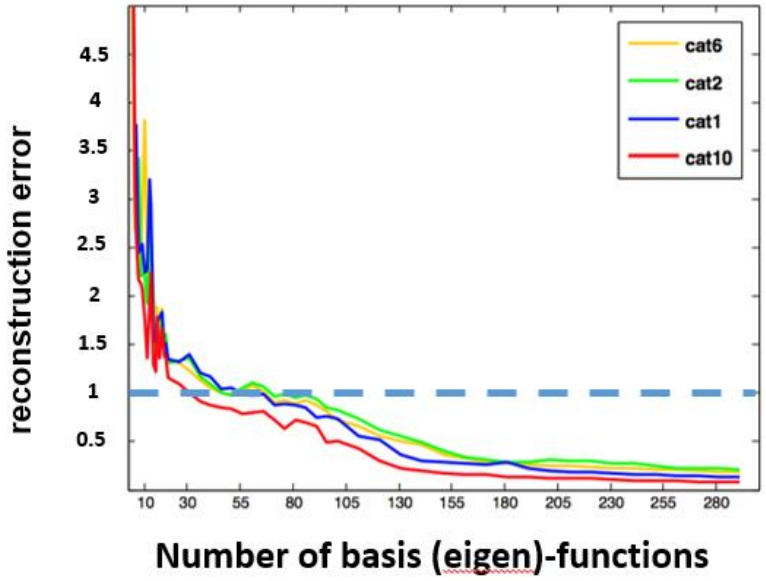
Functions on 2 different domains



Slide credit M. *Ovsjanikov*

Functional map and the size of the basis





Functional maps: a flexible representation of maps between shapes, *Ovsjanikov, Ben-Chen, Solomon, Butscher, Guibas*, SIGGRAPH 2012

Fmaps and the low-pass filter

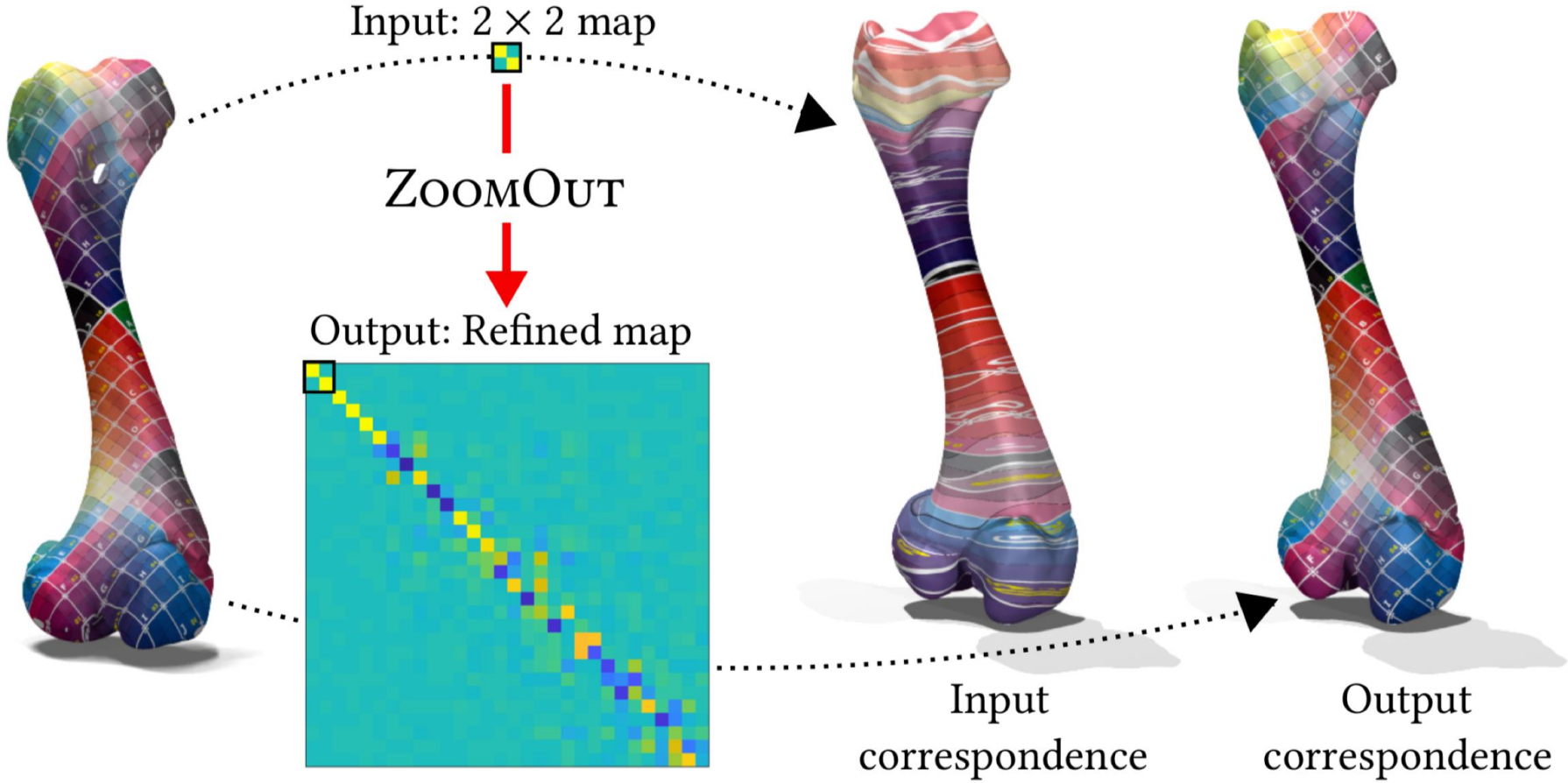
$\mathbf{60 \times 60}$

A trade-off

	+	−
Low-pass	Easy to optimize (fewer probe functions needed)	Low-pass representation (poor functional transfer)
High-frequencies	Better representation of details (good functional transfer)	Hard to optimize (more probe functions needed)

Exploiting the connection with point-to-point-map

Dataset provided by the Natural History Museum in Paris



ZoomOut: Spectral Upsampling for Efficient Shape Correspondence, *Melzi* et al., 2019

ZoomOut Idea

- Progressively registering the eigenfunctions
- Exploiting the connection between functional and point-to-point map

ZoomOut

- 5 lines of code
- Similar complexity to ICP

```
function [C,P]=ZoomOut(M,N,C,k_final)

for k=size(C,1):k_final-1
 x = knnsearch(N.Phi(:,1:k)*C',M.Phi(:,1:k));
 P = sparse(1:M.n,x,1,M.n,N.n);
 C = M.Phi(:,1:k+1)'*M.A*P*N.Phi(:,1:k+1);
end
```

ZoomOut: Spectral Upsampling for Efficient Shape Correspondence, *Melzi* et al., 2019

ZoomOut Algorithm

Slide credit J. *Ren*

$$\min_{C,\Pi} \|\Phi_1 C - \Phi_2(\Pi, :)\|_F^2$$

Algorithm

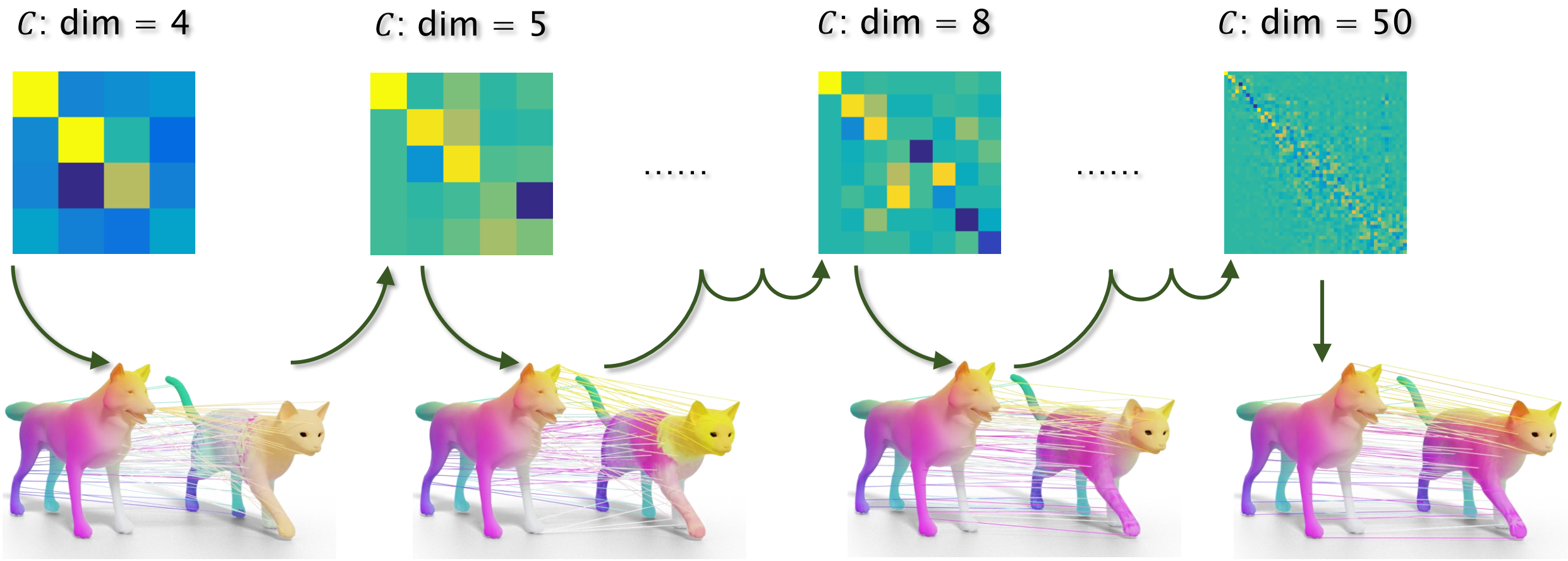
1. Input: an initial map Π and an integer k
2. Solve $C^k = \underset{C}{\operatorname{argmin}} \left\|\Phi_1^k C - \Phi_2^k(\Pi, :)\right\|_F^2$
3. Update $\Pi = \underset{\Pi}{\operatorname{argmin}} \left\|\Phi_1^k C^k - \Phi_2^k(\Pi, :)\right\|_F^2$
4. Update $k = k + 1$
5. Return to step 2.

Φ_i^k are the first k eigenfunctions of S_i

ZoomOut: Spectral Upsampling for Efficient Shape Correspondence, *Melzi* et al., 2019

ZoomOut a visualization

Slide credit J. *Ren*



ZoomOut: Spectral Upsampling for Efficient Shape Correspondence, *Melzi* et al., 2019

Now?

Several paper based on zoomout….

Consistent ZoomOut: Efficient Spectral Map Synchronization

MapTree: Recovering Multiple Solutions in the Space of Maps

Fast Sinkhorn Filters: Using Matrix Scaling for Non-Rigid Shape Correspondence with Functional Maps