Computing and Processing Correspondences with Functional Maps

SIGGRAPH 2017 course

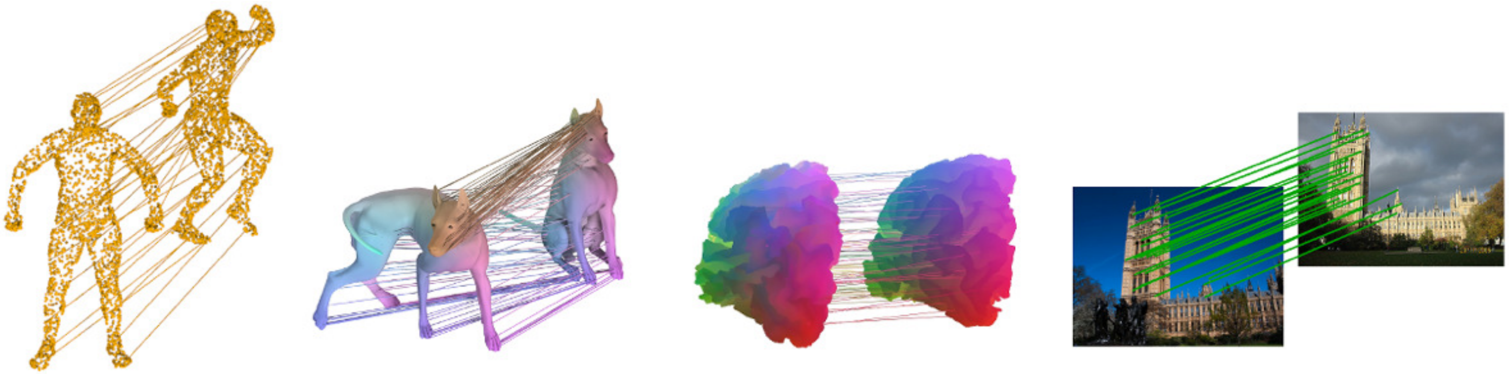
Maks Ovsjanikov, Etienne Corman, Michael Bronstein, Emanuele Rodolà, Mirela Ben-Chen, Leonidas Guibas, Frederic Chazal, Alex Bronstein

École Polytechnique Université Paris-Saclay
Università della Svizzera italiana
Technion Israel Institute of Technology
Stanford University
Inria Inventeurs du monde numérique

General Overview

Overall Objective:

Create tools for computing and analyzing *mappings* between geometric objects.



General Overview

Overall Objective:

Create tools for computing and analyzing *mappings* between geometric objects.

Rather than comparing *points* on objects it is often easier to compare *real-valued functions* defined on them. Such maps can be represented as matrices.

Course Overview

- Course Website:

  http://www.lix.polytechnique.fr/~maks/fmaps_SIG17_course/

  or http://bit.do/fmaps2017

- Course Notes:

  Linked from the website. Or use http://bit.do/fmaps2017_notes
  Attention: (significantly) more material than in the lectures

- Sample Code:

  See **Sample Code** link on the website.

  New: **demo code** for basic operations.

Course Schedule

- 2:00pm – 2:45pm *Introduction* (Maks)
  - Introduction to the functional maps framework.

- 2:50pm – 3:35pm *Computing Functional Maps* (Etienne)
  - Optimization methods for functional map estimation.

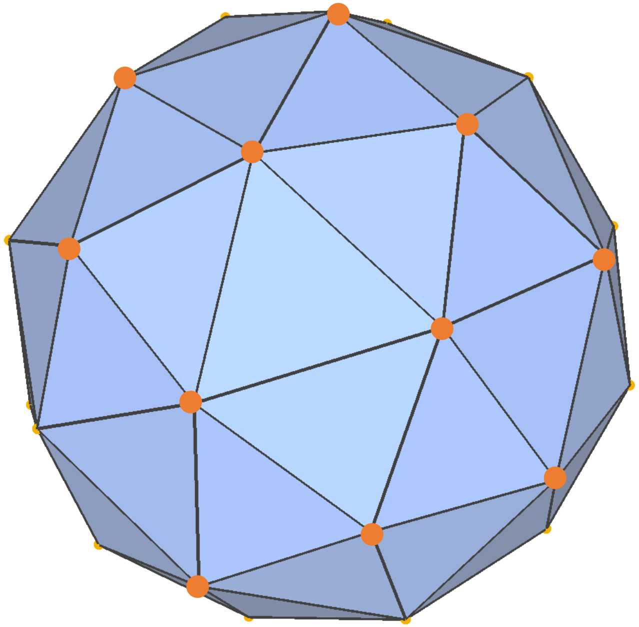
- 3:40pm - 4:25pm *Functional Vector Fields* (Miri)
  - From functional maps to tangent vector fields and back.

- 4:30pm - 5:15pm *Maps in Shape Collections* (Leo)
  - Networks of maps, shape variability, learning.

5:15pm - *Wrapup, Q&A* (all)

What is a Shape?

- Continuous: a surface embedded in 3D.
- Discrete: a graph embedded in 3D (triangle mesh).

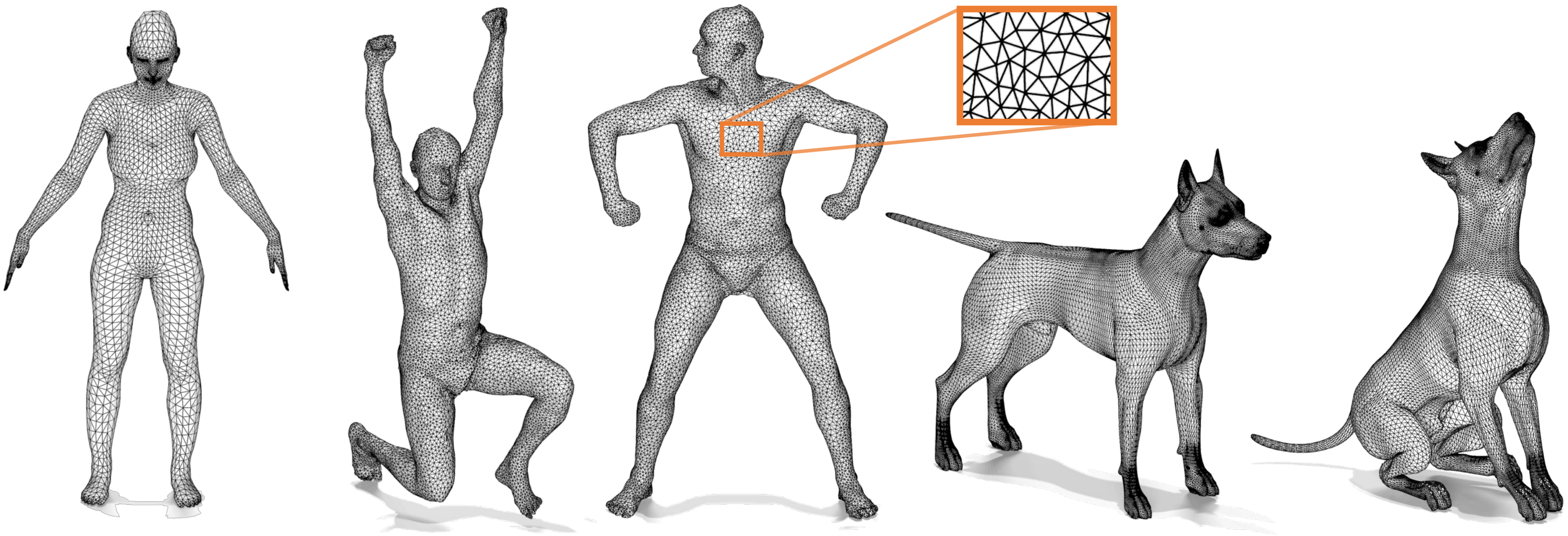


Common assumptions:

- Connected.
- Manifold.
- Without Boundary.

What is a Shape?

- Continuous: a surface embedded in 3D.
- Discrete: a graph embedded in 3D (triangle mesh).

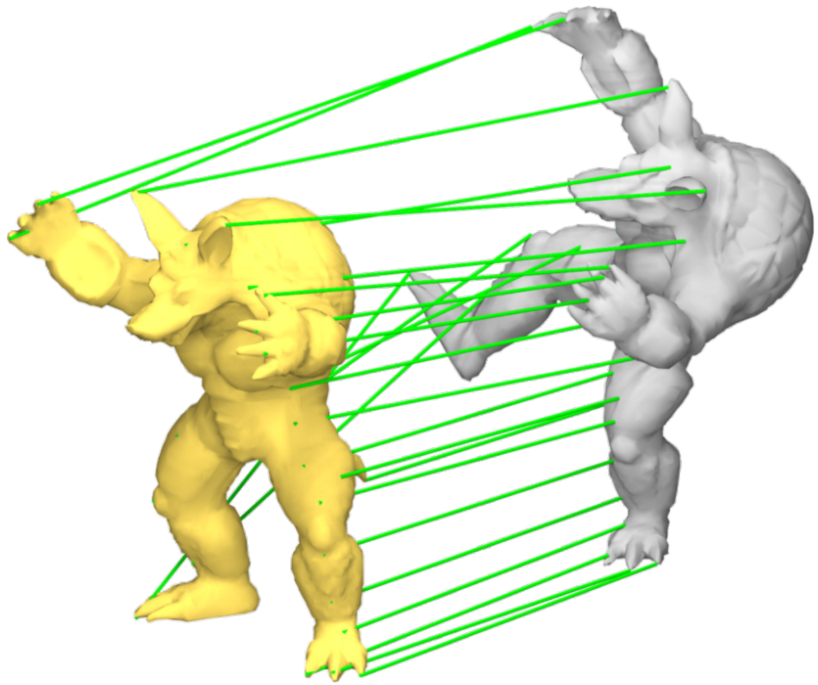


5k – 200k triangles

Shapes from the FAUST, SCAPE, and TOSCA datasets

Overall Goal

- Given two shapes, find **correspondences** between them.



- Finding the **best** map between a pair of shapes.

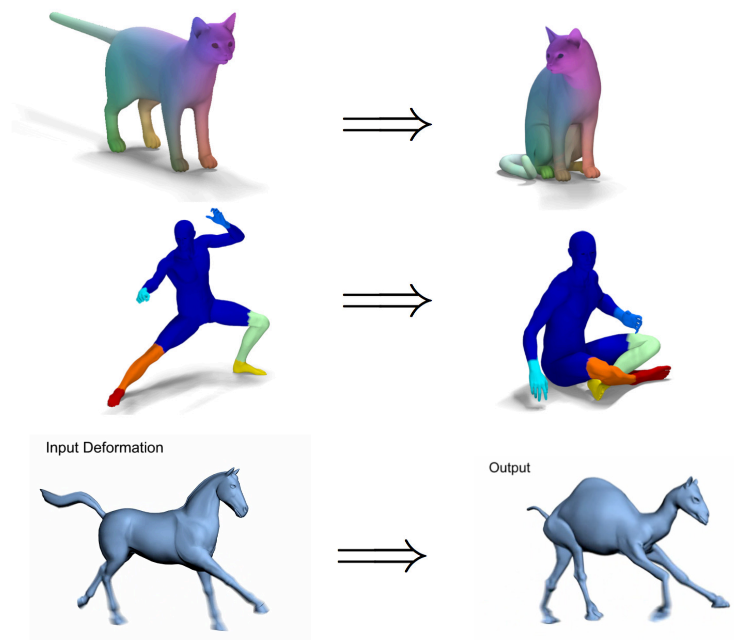
Why Shape Matching?

Given a correspondence, we can transfer:

texture and parametrization

segmentation and labels

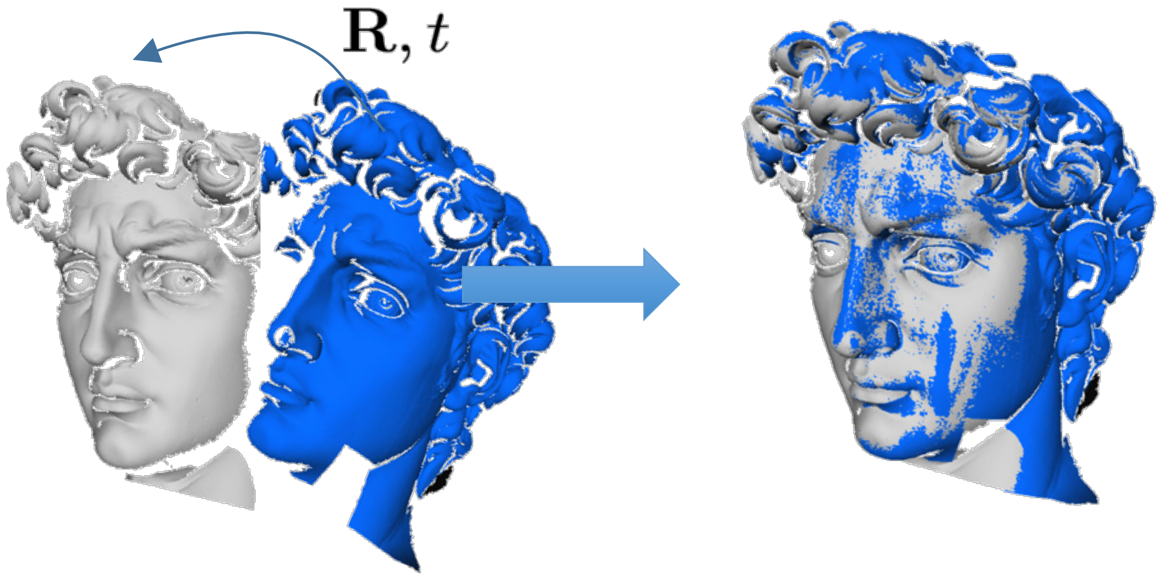
deformation



Sumner et al. ‘04.

Other applications: shape interpolation, reconstruction ...

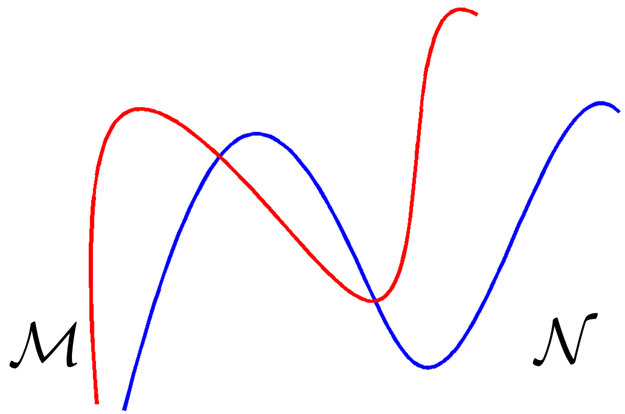
Rigid Shape Matching



- Given a pair of shapes, find the optimal *Rigid Alignment* between them.
- The unknowns are the rotation/translation parameters of the source onto the target shape.

Iterative Closest Point (ICP)

- Classical approach: iterate between finding correspondences and finding the transformation:



example in 2D

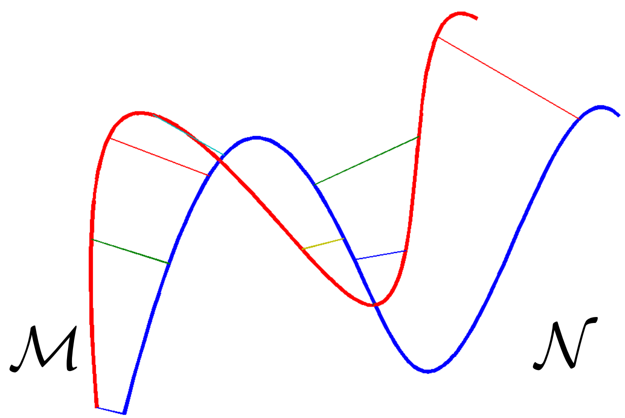
Given a pair of shapes, $\mathcal{M}$ and $\mathcal{N}$, iterate:

1. For each $x_i \in \mathcal{M}$ find **nearest** neighbor $y_i \in \mathcal{N}$.
2. Find optimal transformation R, t minimizing:

$$\arg\min_{R,t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Iterative Closest Point

- Classical approach: iterate between finding correspondences and finding the transformation:



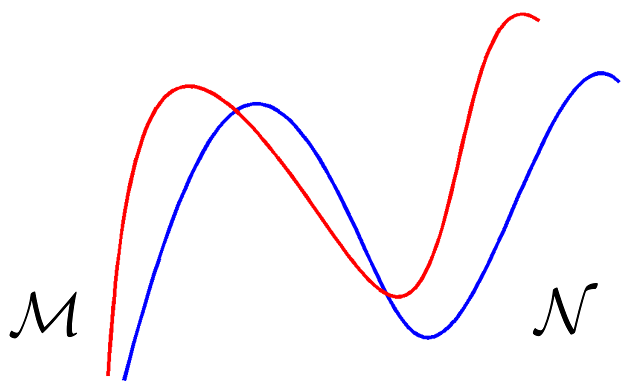
Given a pair of shapes, $\mathcal{M}$ and $\mathcal{N}$, iterate:

1. For each $x_i \in \mathcal{M}$ find **nearest** neighbor $y_i \in \mathcal{N}$.
2. Find optimal transformation R, t minimizing:

$$\arg\min_{R,t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Iterative Closest Point

- Classical approach: iterate between finding correspondences and finding the transformation:



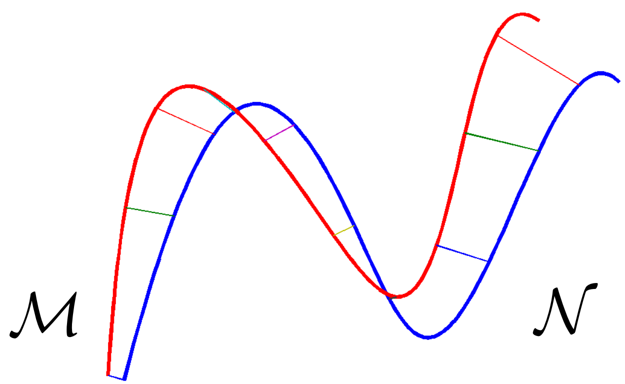
Given a pair of shapes, $\mathcal{M}$ and $\mathcal{N}$, iterate:

1. For each $x_i \in \mathcal{M}$ find **nearest** neighbor $y_i \in \mathcal{N}$.
2. Find optimal transformation R, t minimizing:

$$\arg\min_{R,t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Iterative Closest Point

- Classical approach: iterate between finding correspondences and finding the transformation:



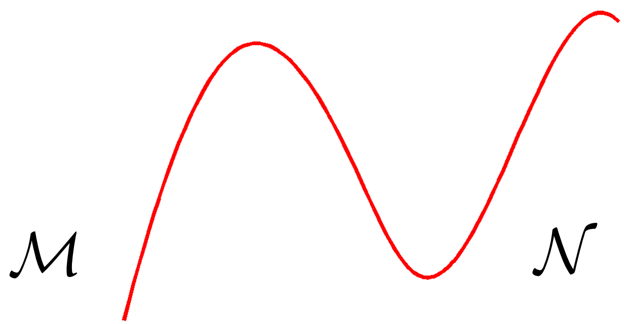
Given a pair of shapes, $\mathcal{M}$ and $\mathcal{N}$, iterate:

1. For each $x_i \in \mathcal{M}$ find **nearest** neighbor $y_i \in \mathcal{N}$.
2. Find optimal transformation R, t minimizing:

$$\arg\min_{R,t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Iterative Closest Point

- Classical approach: iterate between finding correspondences and finding the transformation:



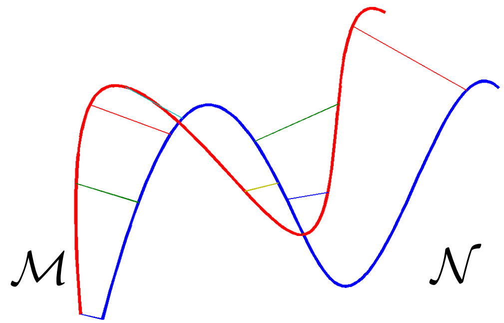
Given a pair of shapes, $\mathcal{M}$ and $\mathcal{N}$, iterate:

1. For each $x_i \in \mathcal{M}$ find **nearest** neighbor $y_i \in \mathcal{N}$.
2. Find optimal transformation R, t minimizing:

$$\arg\min_{R,t} \sum_i \|Rx_i + t - y_i\|_2^2$$

Iterative Closest Point

- Classical approach: iterate between finding correspondences and finding the transformation:



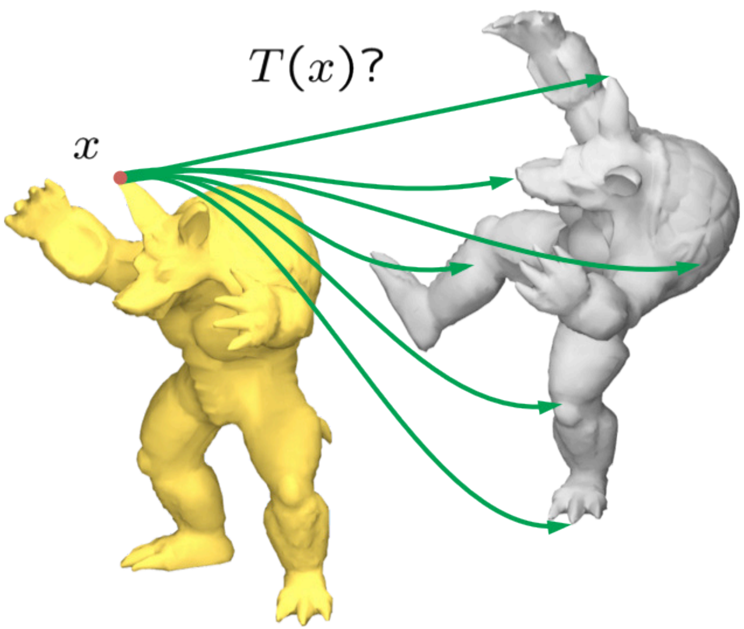
1. Finding nearest neighbors: can be done with space-partitioning data structures (e.g., KD-tree).
2. Finding the optimal transformation R, t minimizing:

$$\underset{R \in \mathrm{SO}(3),\ t \in \mathbb{R}^3}{\arg\min} \sum_i \|Rx_i + t - y_i\|_2^2$$

Can be done efficiently via SVD decomposition.

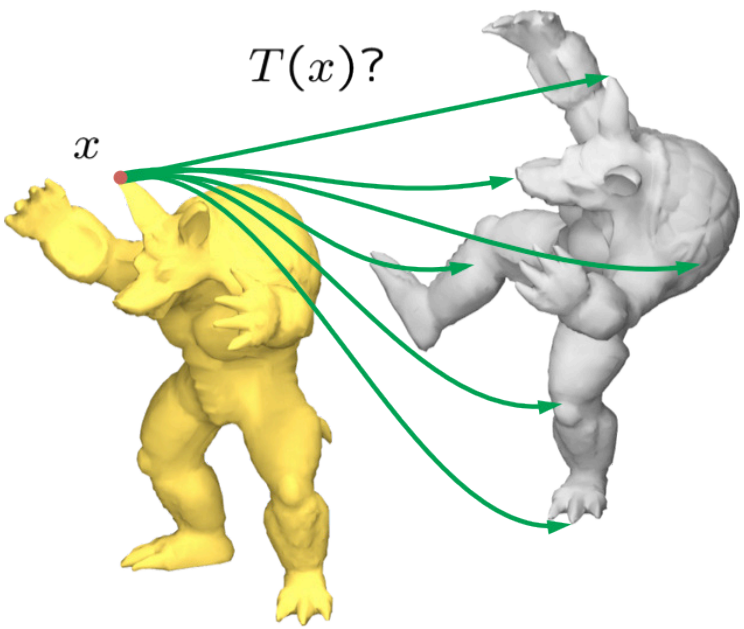
Arun et al., Least-Squares Fitting of Two 3-D Point Sets

Non-Rigid Shape Matching



Unlike rigid matching with rotation/translation, there is no compact representation to optimize for in non-rigid matching.

Non-Rigid Shape Matching



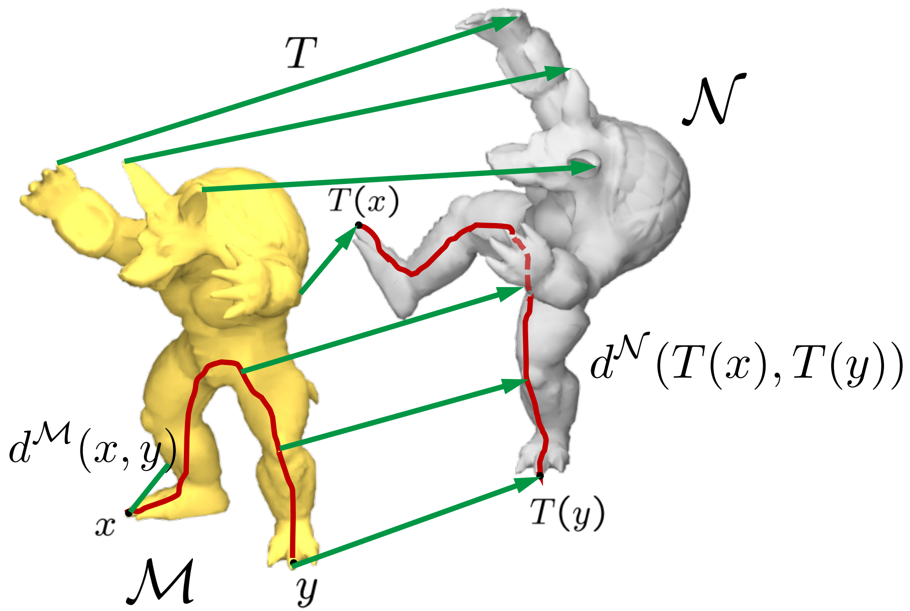
Main Questions:

- What does it mean for a correspondence to be "good"?
- How to compute it efficiently in practice?

Isometric Shape Matching

Deformation Model:

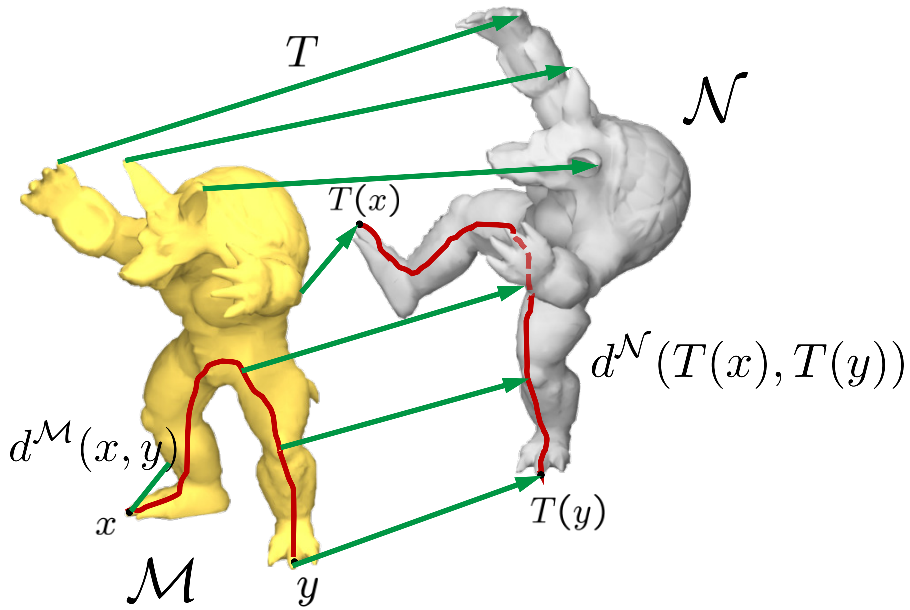
Good maps must preserve geodesic distances.



Geodesic: length of shortest path lying entirely on the surface.

Isometric Shape Matching

Approach:



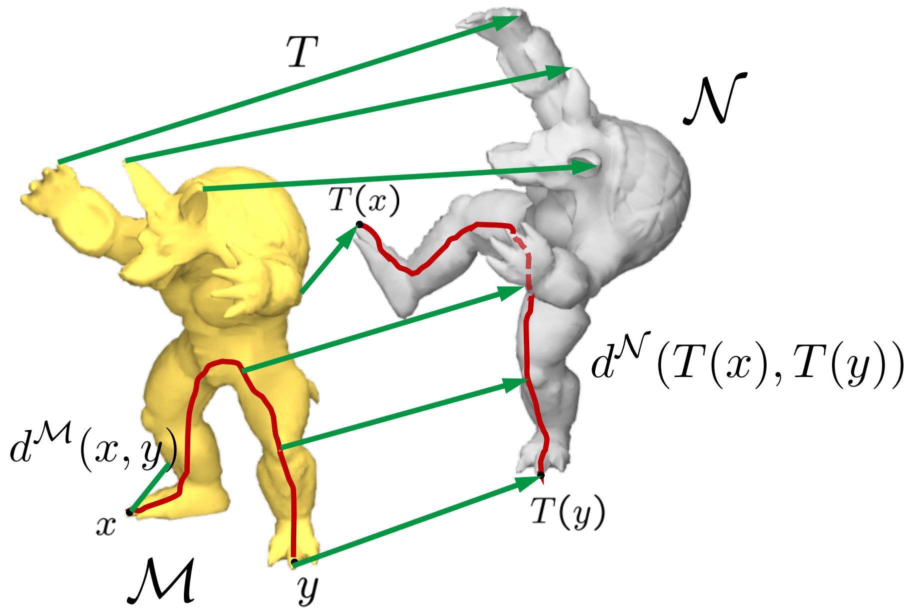
Find the point mapping by minimizing the distance distortion:

$$T_{\text{opt}} = \arg\min_{T} \sum_{x,y} \| d^{\mathcal{M}}(x, y) - d^{\mathcal{N}}(T(x), T(y)) \|$$

The unknowns are point correspondences.

Isometric Shape Matching

Approach:



Find the point mapping by minimizing the distance distortion:

$$T_{\text{opt}} = \arg\min_{T} \sum_{x,y} \| d^{\mathcal{M}}(x, y) - d^{\mathcal{N}}(T(x), T(y)) \|$$

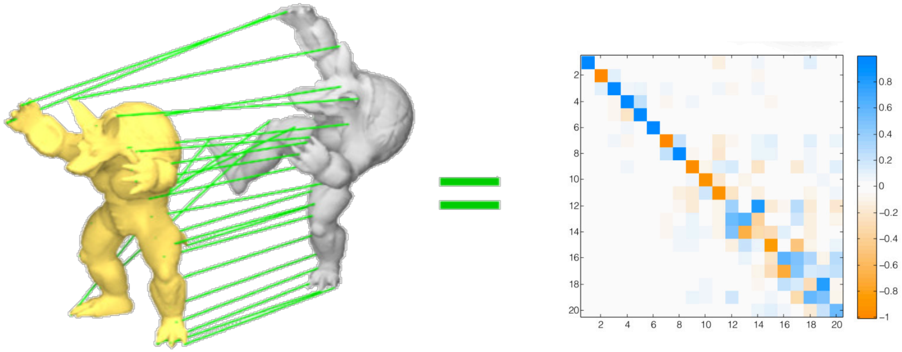
Problem:

The space of possible solutions is highly non-linear, non-convex.

Functional Map Representation

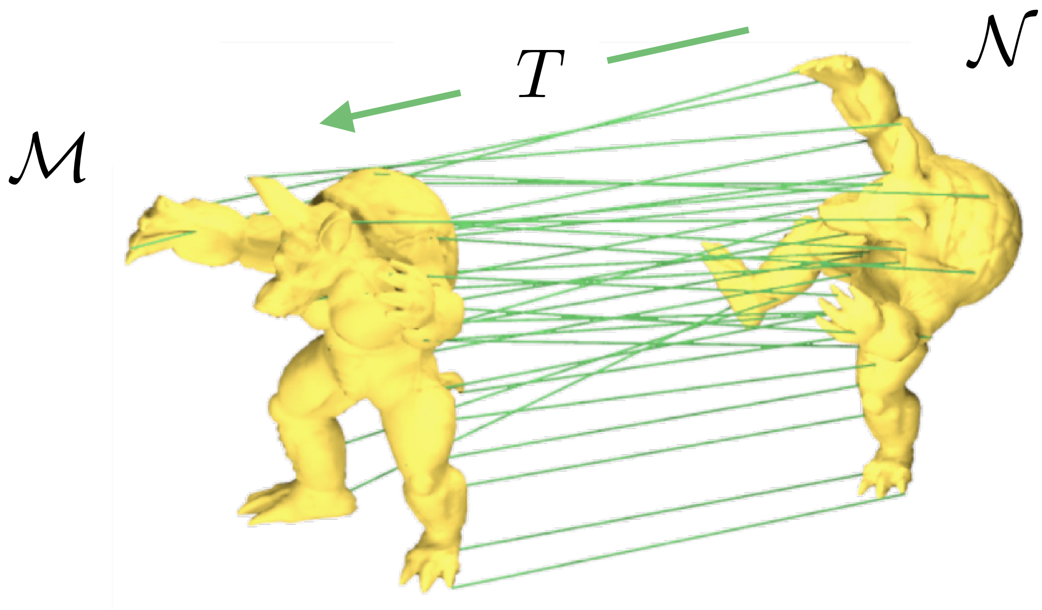
We would like to define a representation of shape maps that is more amenable to direct optimization.

1. A compact representation for “natural” maps.
2. Inherently global and multi-scale.
3. Handles uncertainty and ambiguity gracefully.
4. Allows efficient manipulations (averaging, composition).
5. Leads to simple (linear) optimization problems.



Functional Approach to Mappings

Given two shapes and a pointwise map $T : \mathcal{N} \to \mathcal{M}$



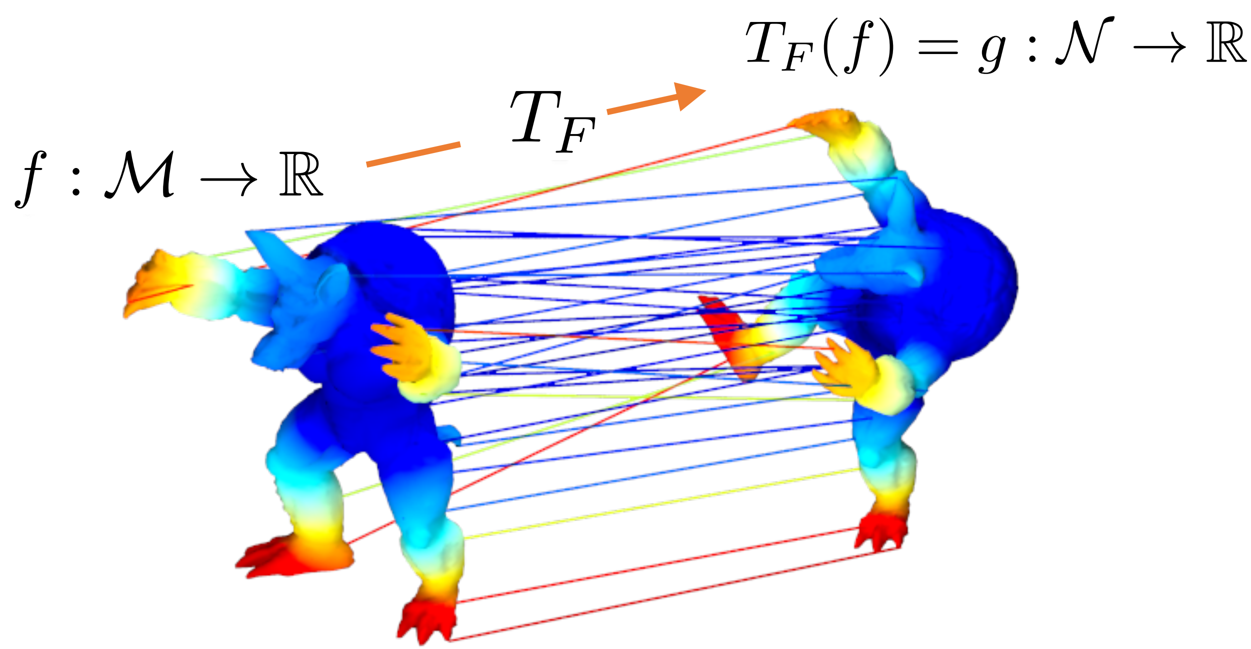
The map T induces a functional correspondence:

$$T_F(f) = g, \text{ where } g = f \circ T$$

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

Functional Approach to Mappings

Given two shapes and a pointwise map $T : \mathcal{N} \to \mathcal{M}$



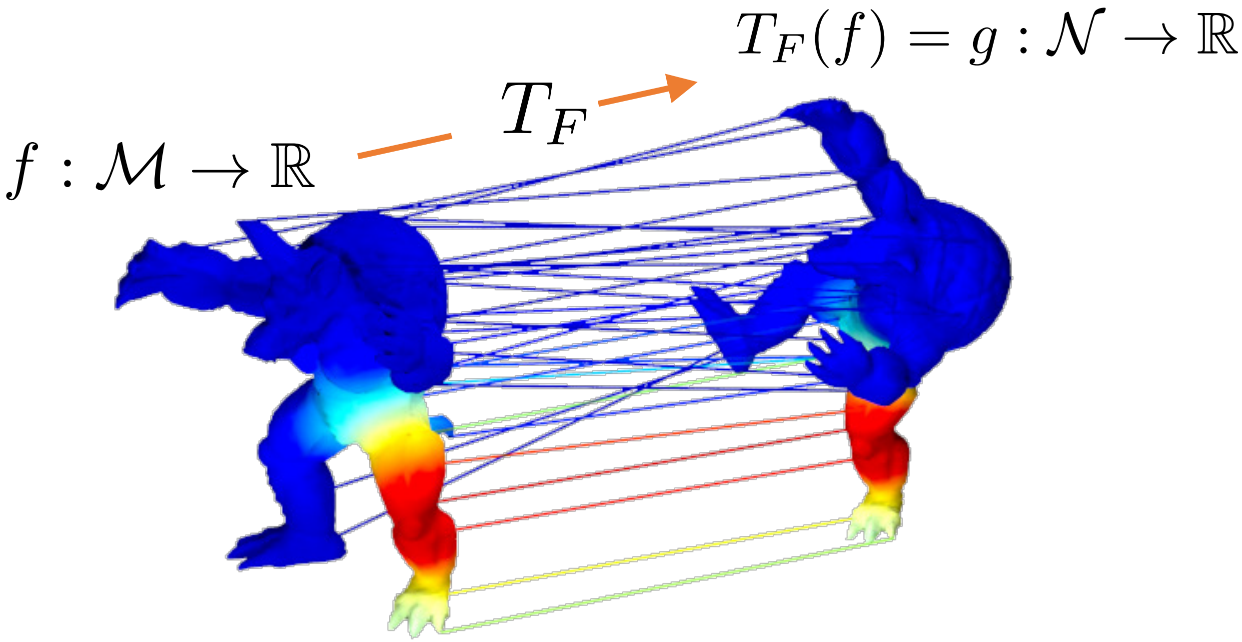
The map T induces a functional correspondence:

$$T_F(f) = g, \text{ where } g = f \circ T$$

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

Functional Approach to Mappings

Given two shapes and a pointwise map $T : \mathcal{N} \to \mathcal{M}$



The map T induces a functional correspondence:

$$T_F(f) = g, \text{ where } g = f \circ T$$

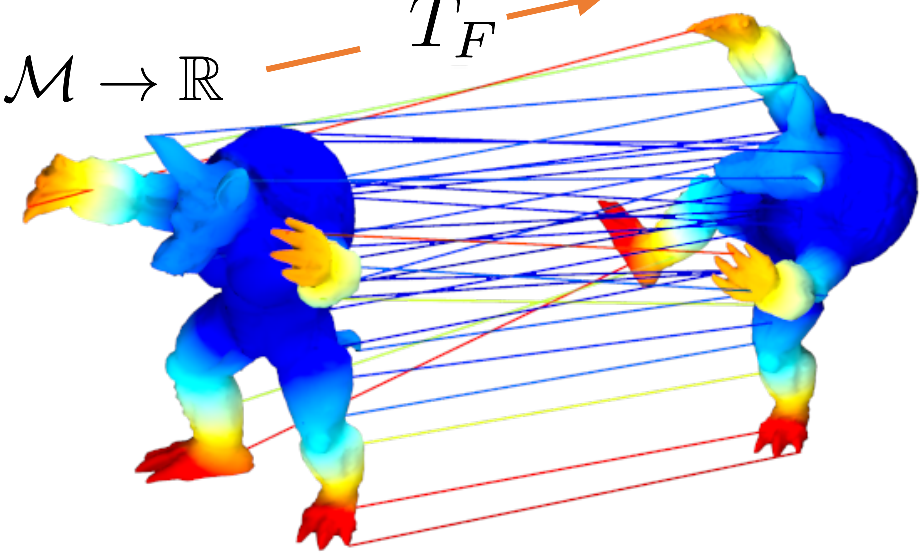
Functional maps: a flexible representation of maps between shapes, *O., Ben-Chen, Solomon, Butscher, Guibas*, SIGGRAPH 2012

Functional Approach to Mappings

Given two shapes and a pointwise map $T : \mathcal{N} \to \mathcal{M}$

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

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

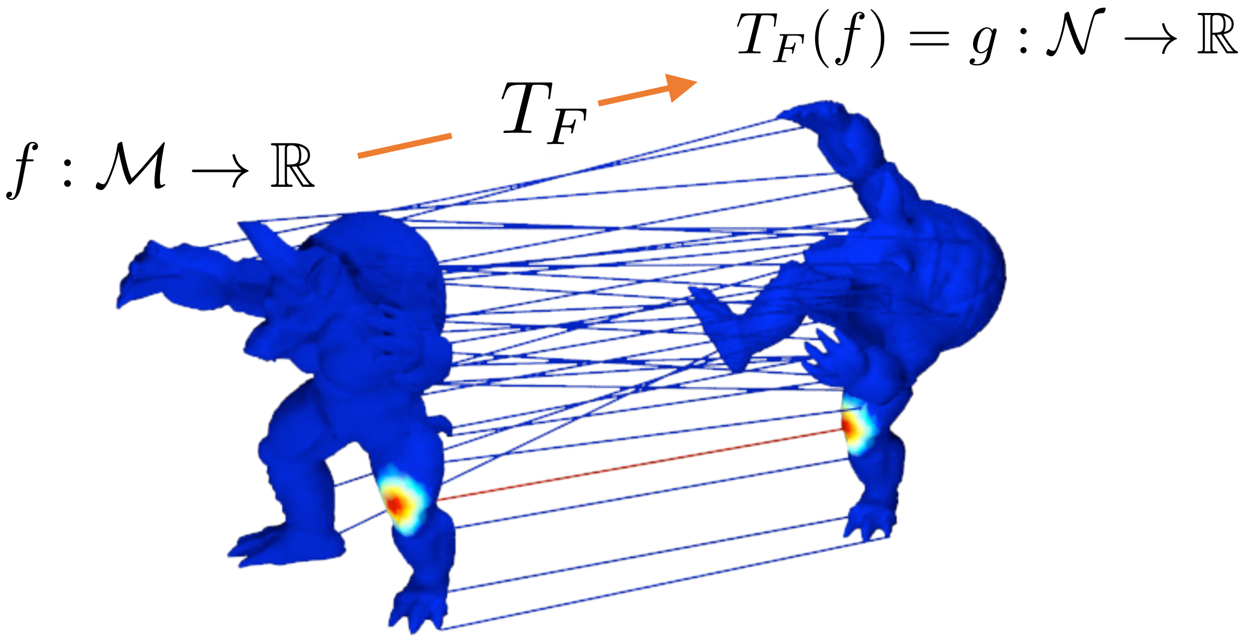


The induced functional correspondence is linear:

$$T_F(\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 T_F(f_1) + \alpha_2 T_F(f_2)$$

Functional Map Representation

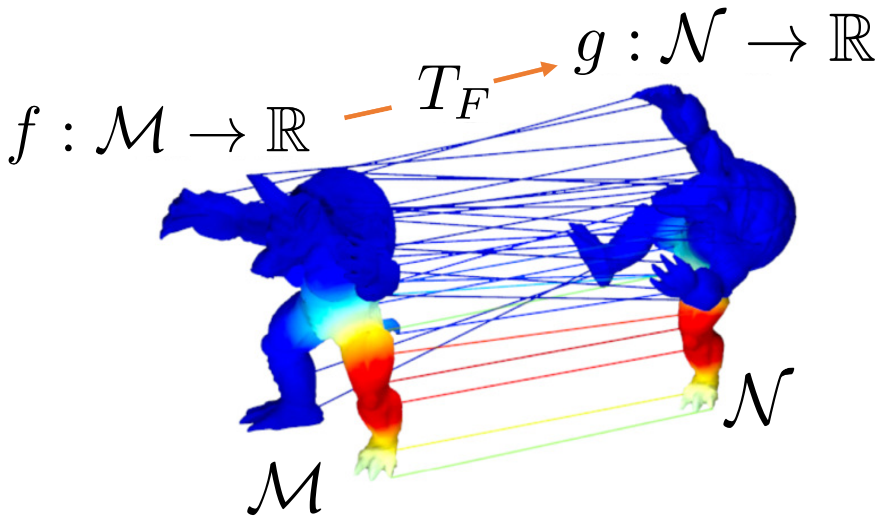
Given two shapes and a pointwise map $T : \mathcal{N} \to \mathcal{M}$



The induced functional correspondence is complete.

Observation

Assume that both: $f \in \mathcal{L}_2(\mathcal{M}), g \in \mathcal{L}_2(\mathcal{N})$



Express both f and $T_F(f)$ in terms of *basis functions:*

$$f = \sum_i a_i \phi_i^{\mathcal{M}} \qquad g = T_F(f) = \sum_j b_j \phi_j^{\mathcal{N}}$$

$\boxed{\text{Since } T_F \text{ is linear, there is a linear transformation from } \{a_i\} \text{ to } \{b_j\}.}$

Functional Map Representation

Choice of Basis:

Eigenfunctions of the Laplace-Beltrami operator:

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

- Generalization of *Fourier bases* to surfaces.

- Ordered by eigenvalues and provide a natural notion of *scale*.

$\lambda_0 = 0$ $\quad \lambda_1 = 2.6$ $\quad \lambda_2 = 3.4$ $\quad \lambda_3 = 5.1$ $\quad \lambda_4 = 7.6$

Functional Map Representation

Choice of Basis:

Eigenfunctions of the Laplace-Beltrami operator:

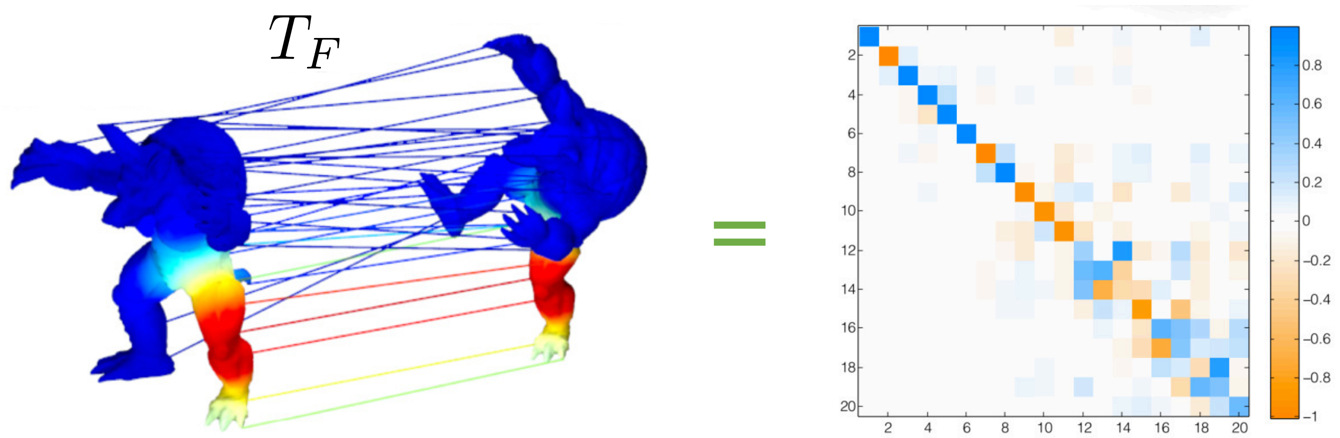
$$\Delta\phi_i = \lambda_i\phi_i$$

- Generalization of *Fourier bases* to surfaces.
- Ordered by eigenvalues and provide a natural notion of *scale*.
- Can be computed efficiently, with a sparse matrix eigensolver.

Functional Map Representation

Since the functional mapping T_F is linear:

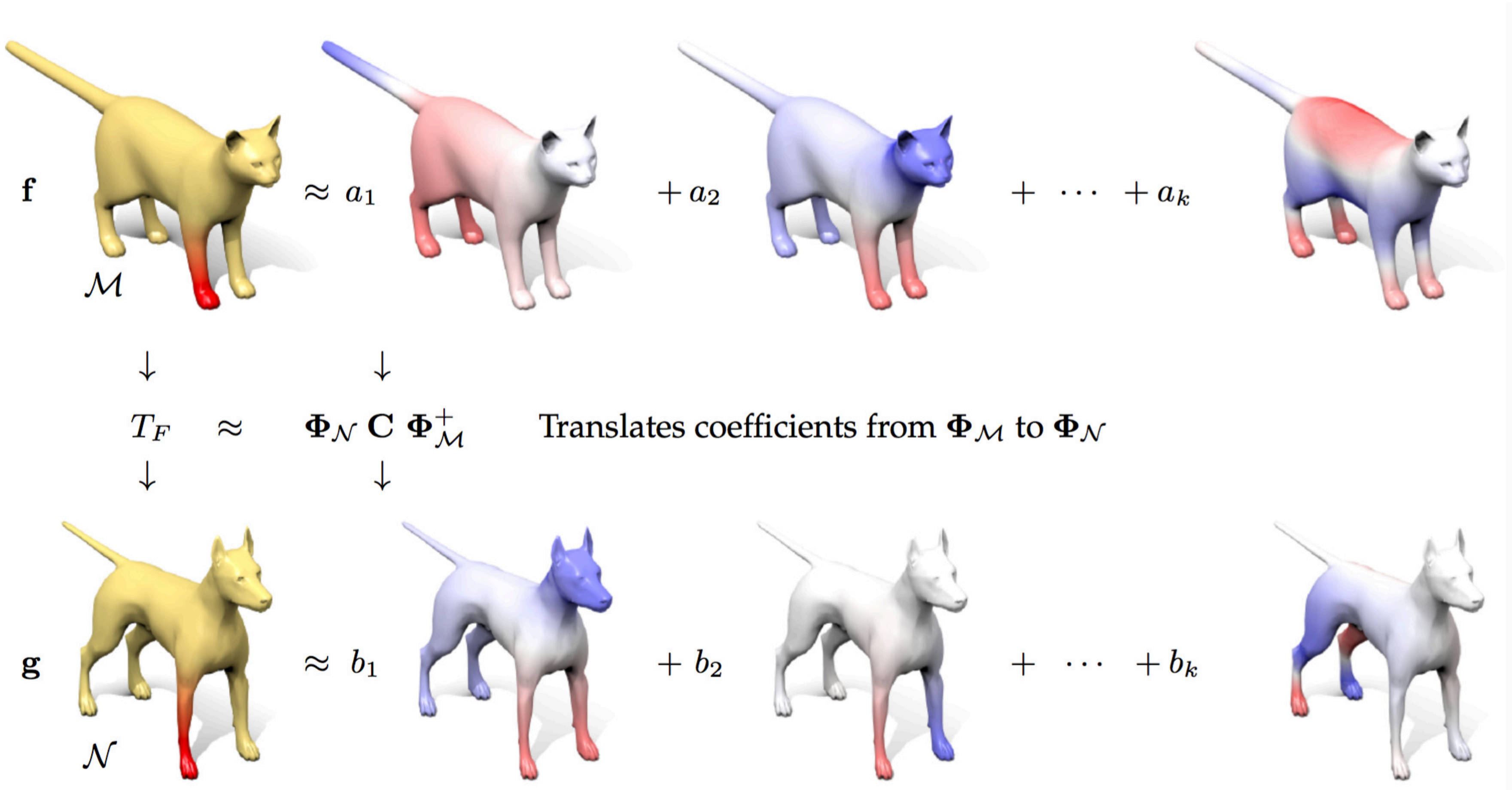
$$T_F(\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 T_F(f_1) + \alpha_2 T_F(f_2)$$



T_F can be represented as a matrix C, given a choice of basis for function spaces.

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

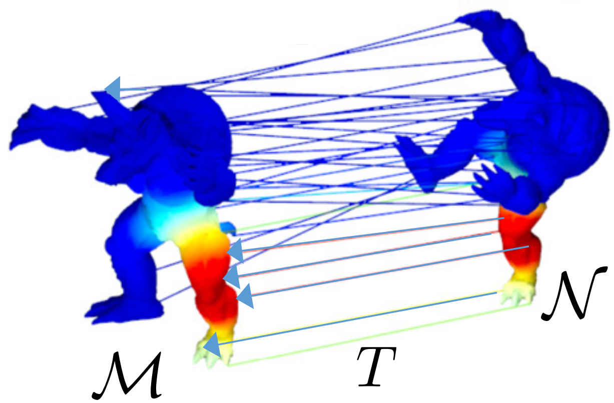
Functional Map Definition



Functional map:

matrix C that translates coefficients from $\Phi_{\mathcal{M}}$ to $\Phi_{\mathcal{N}}$.

Example 1



Given two shapes with $n_{\mathcal{M}}, n_{\mathcal{N}}$ points and a map: $T : \mathcal{N} \rightarrow \mathcal{M}$

$\mathbf{T} : n_{\mathcal{N}} \times n_{\mathcal{M}}$ matrix encoding the map T, one 1 per column with zeros everywhere else.

If functions are represented as vectors (in the hat basis), the functional map is given by matrix-vector product:

$$g = \mathbf{T}^T f \qquad C = \mathbf{T}^T$$

Example 2

Given two shapes with $n_{\mathcal{M}}, n_{\mathcal{N}}$ points and a map: $T : \mathcal{N} \to \mathcal{M}$

$\mathbf{T} : n_{\mathcal{N}} \times n_{\mathcal{M}}$ matrix encoding the map T, one 1 per column with zeros everywhere else.

If functions are represented in the reduced basis:

$\Phi_{\mathcal{M}} : n_{\mathcal{M}} \times k_{\mathcal{M}}$ matrix of the first $k_{\mathcal{M}}$ eigenfunctions of $\Delta_{\mathcal{M}}$ as columns.

$\Phi_{\mathcal{N}} : n_{\mathcal{N}} \times k_{\mathcal{N}}$ matrix of the first $k_{\mathcal{N}}$ eigenfunctions of $\Delta_{\mathcal{N}}$ as columns.

The functional map matrix:

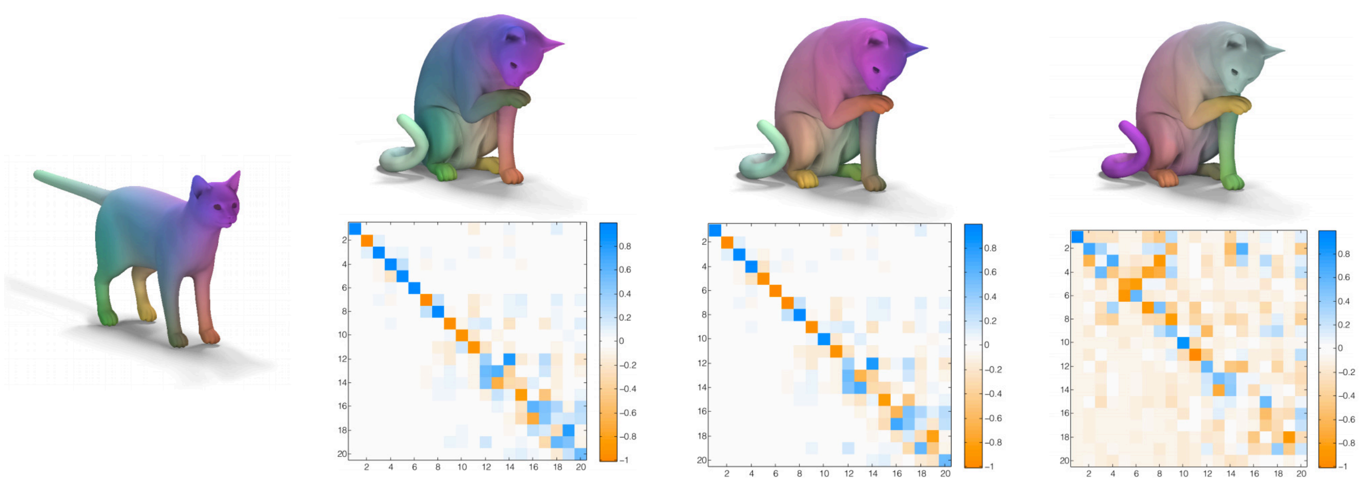
$C = \Phi_{\mathcal{N}}^{+} \mathbf{T}^T \Phi_{\mathcal{M}}$ $\quad$ $^{+}$: left pseudo-inverse.

$C = \Phi_{\mathcal{N}}^{T} \mathbf{T}^T \Phi_{\mathcal{M}}$ $\quad$ if $\quad \Phi_{\mathcal{N}}^{T} \Phi_{\mathcal{N}} = Id$

$C = \Phi_{\mathcal{N}}^{T} A_{\mathcal{N}} \mathbf{T}^T \Phi_{\mathcal{M}}$ $\quad$ if $\quad \Phi_{\mathcal{N}}^{T} A_{\mathcal{N}} \Phi_{\mathcal{N}} = Id$

Example Maps in a Reduced Basis

Triangle meshes with pre-computed pointwise maps



(a) source (b) ground-truth map (c) left-to-right map (d) head-to-tail map

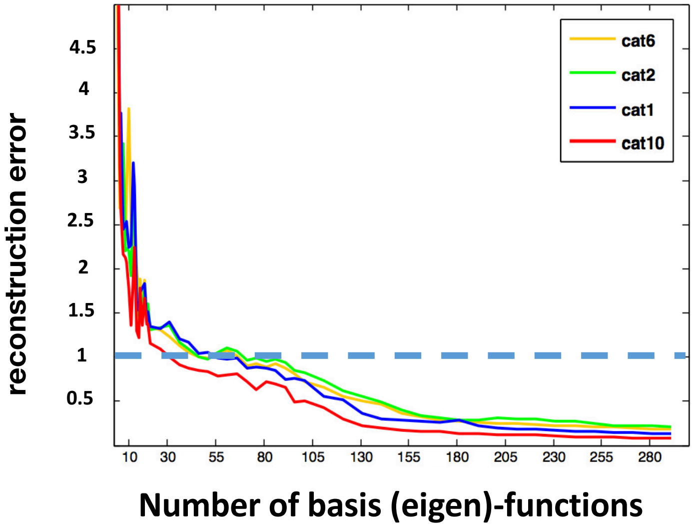
"Good" maps are close to being diagonal

Reconstructing from LB basis

Map reconstruction error using a fixed size matrix.

source Cat10 Cat1 Cat2 Cat6

27.9k vertices



Functional Map algebra

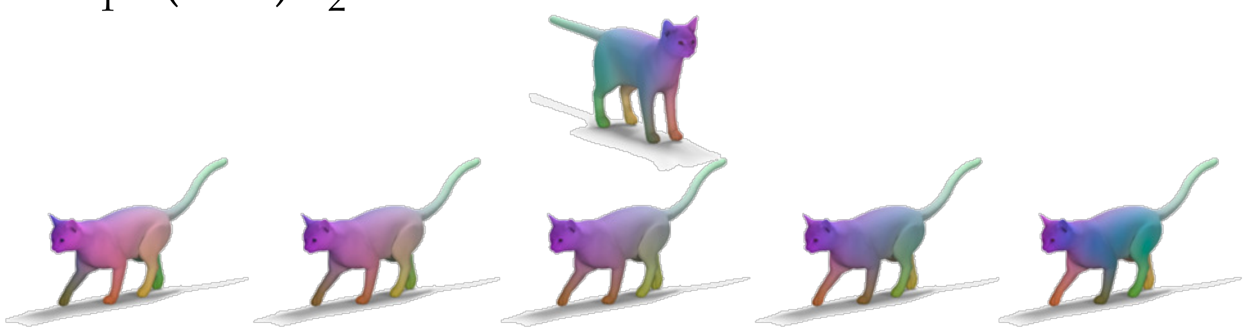
1. Map composition becomes matrix multiplication.

2. Map inversion is matrix inversion (in fact, transpose).

3. Algebraic operations on functional maps are possible.

E.g. interpolating between two maps with

$$C = \alpha C_1 + (1-\alpha) C_2.$$



(a) $\alpha = 0$ (b) $\alpha = 0.25$ (c) $\alpha = 0.5$ (d) $\alpha = 0.75$ (e) $\alpha = 1$

Shape Matching

In practice we do not know C. Given two objects our goal is to find the correspondence.

How can the functional representation help to compute the map in practice?

Matching via Function Preservation

Suppose we don't know C. However, we expect a pair of functions $f : \mathcal{M} \to \mathbb{R}$ and $g : \mathcal{N} \to \mathbb{R}$ to correspond. Then, C must be s.t.

$$C\mathbf{a} \approx \mathbf{b}$$

where $f = \sum_i a_i \phi_i^{\mathcal{M}}, \quad g = \sum_i b_i \phi_i^{\mathcal{N}}.$

Given enough $\{\mathbf{a}, \mathbf{b}\}$ pairs, we can recover C through *a linear least squares system*.

Map Constraints

Suppose we don't know C. However, we expect a pair of functions $f : \mathcal{M} \to \mathbb{R}$ and $g : \mathcal{N} \to \mathbb{R}$ to correspond. Then, C must be s.t.

$$C\mathbf{a} \approx \mathbf{b}$$

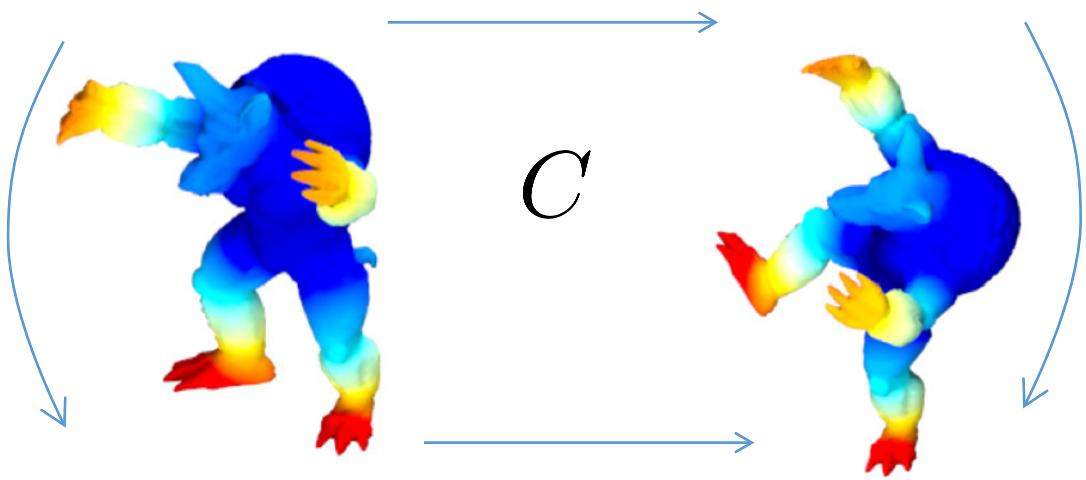
Function preservation constraint is general and includes:

- Attribute (e.g., color) preservation.
- Descriptor preservation (e.g. Gauss curvature).
- Landmark correspondences (e.g. distance to the point).
- Part correspondences (e.g. indicator function).

Commutativity Constraints

Regularizations:

Commutativity with other operators:



$$CS_{\mathcal{M}} = S_{\mathcal{N}}C$$

Note that the energy: $\|CS_{\mathcal{M}} - S_{\mathcal{N}}C\|_F^2$ is *quadratic* in C.

Regularization

Lemma 1:

The mapping is *isometric*, if and only if the functional map matrix commutes with the Laplacian:

$$C\Delta_{\mathcal{M}} = \Delta_{\mathcal{N}} C$$

Implies that exact isometries result in *diagonal functional maps*.

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

Regularization

Lemma 2:

The mapping is *locally volume preserving*, if and only if the functional map matrix is *orthonormal:*

$$C^T C = Id$$

Map-Based Exploration of Intrinsic Shape Differences and Variability, Rustamov et al., SIGGRAPH 2013

Regularization

Lemma 3:

If the mapping is *conformal* if and only if:

$$C^T \Delta_1 C = \Delta_2$$

Map-Based Exploration of Intrinsic Shape Differences and Variability, Rustamov et al., SIGGRAPH 2013

Basic Pipeline

Given a pair of shapes $\mathcal{M}, \mathcal{N}$:

1. Compute the first *k* (~80-100) eigenfunctions of the Laplace-Beltrami operator. Store them in matrices: $\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}$

2. Compute descriptor functions (e.g., Wave Kernel Signature) on $\mathcal{M}, \mathcal{N}$. Express them in $\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}$, as columns of : $\mathbf{A}, \mathbf{B}$

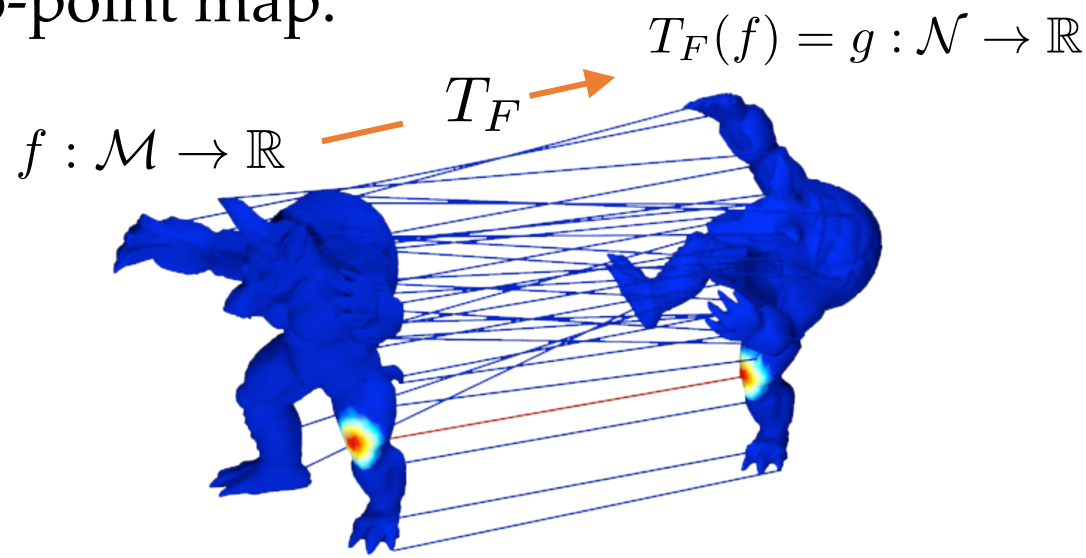
3. Solve $C_{\text{opt}} = \underset{C}{\arg\min} \|C\mathbf{A} - \mathbf{B}\|^2 + \|C\Delta_{\mathcal{M}} - \Delta_{\mathcal{N}} C\|^2$

   $\Delta_{\mathcal{M}}, \Delta_{\mathcal{N}}$: diagonal matrices of eigenvalues of LB operator

4. Convert the functional map C_{opt} to a point to point map T.

Conversion to point-to-point

Given a functional map C, we would like to convert to to a point-to-point map.

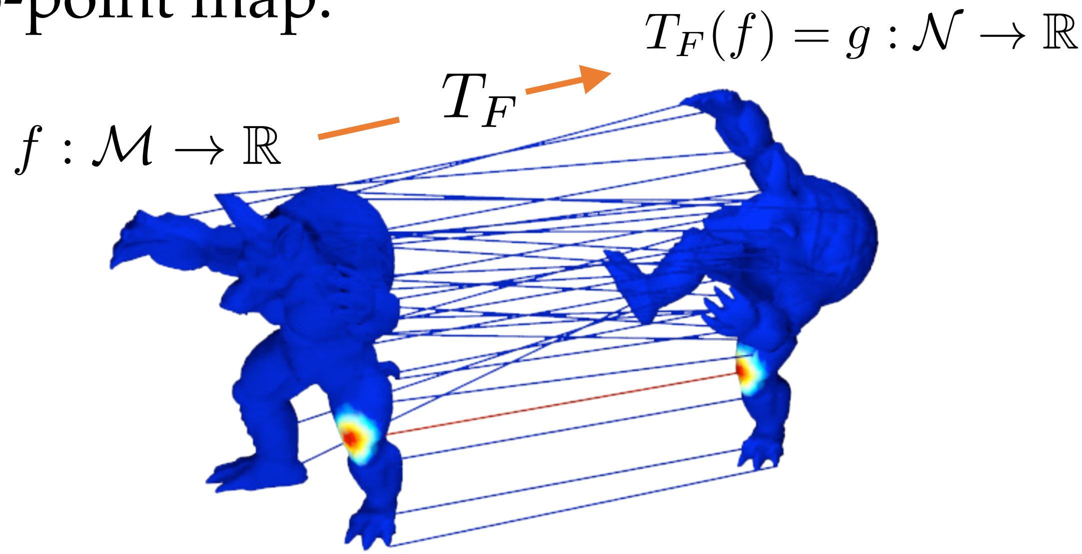


Option 1: declare $T(x) = \arg\max_{y} \Phi_{\mathcal{N}} C \delta_x$

Problems: high computational complexity $O(n_{\mathcal{M}} n_{\mathcal{N}})$, low accuracy.

Conversion to point-to-point

Given a functional map C, we would like to convert to to a point-to-point map.



Option 2: declare $T(x) = \arg\min_y \|\delta_y - C\delta_x\|_2$

Advantages: computational complexity $O(n_{\mathcal{M}} \log n_{\mathcal{N}})$, higher accuracy (e.g., works with the identity map).

Incorporating Orthonormality

In many practical situations we would expect a volume-preserving map, which implies:

$$C^T C = Id$$

Option: use post-processing to enforce this constraint.

Iterate:

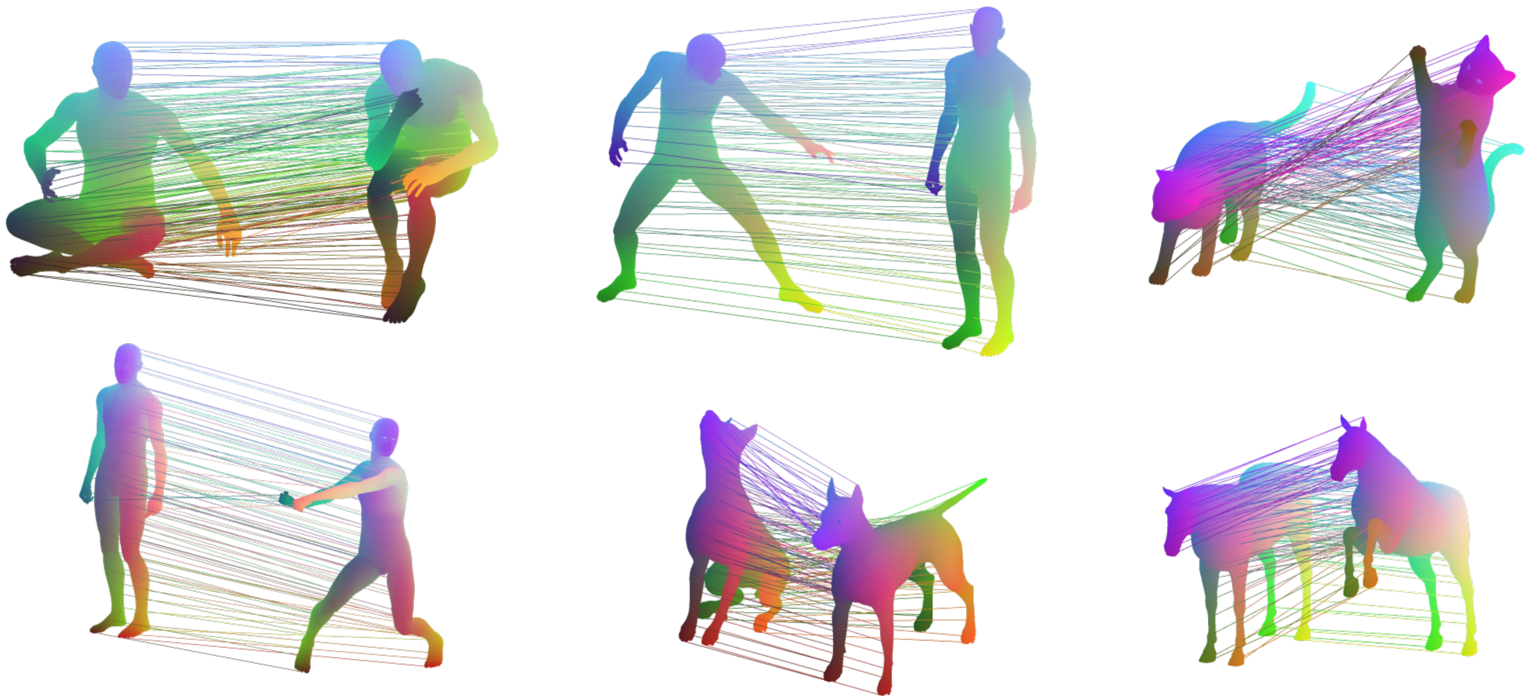
1. Compute the point-to-point map T.

2. Solve for the functional map: $\underset{C,\ \text{s.t. } C^T C = Id}{\arg\min} \sum_{x \in \mathcal{M}} \|C\delta_x - \delta_{T(x)}\|_2^2$

Exactly the same objective as ICP, but in higher dimension. Can use the same method!

Results

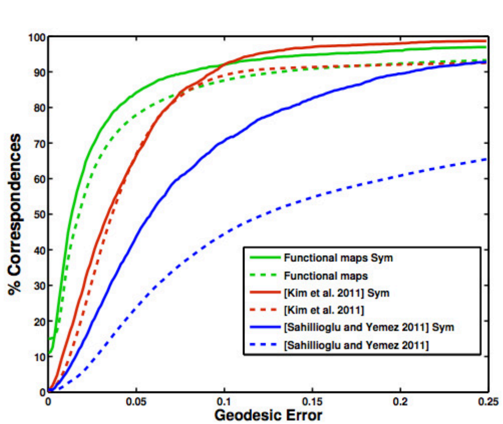
A very simple method that puts together many constraints and uses 100 basis functions gives reasonable results:



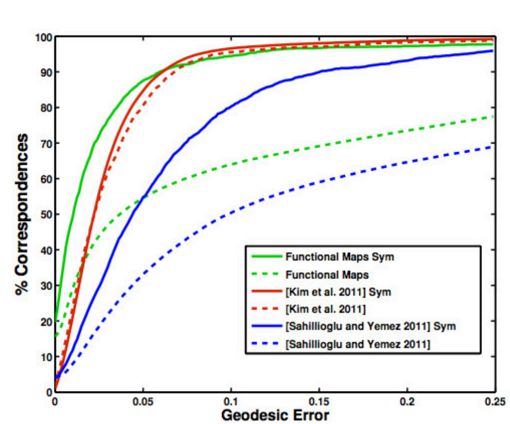
Functional maps: a flexible representation of maps between shapes, *O., Ben-Chen, Solomon, Butscher, Guibas*, SIGGRAPH 2012

Results

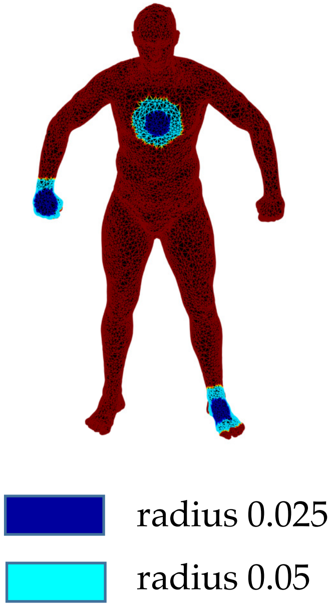
A very simple method that puts together many constraints and uses 100 basis functions gives reasonable results:



SCAPE



TOSCA



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

Segmentation Transfer without P2P

To transfer functions we do not need to convert functional to pointwise maps.

E.g. we can also transfer segmentations: for each segment, transfer its indicator function, and for each point pick the segment that gave the highest value.