252-0538-00L, Spring 2025

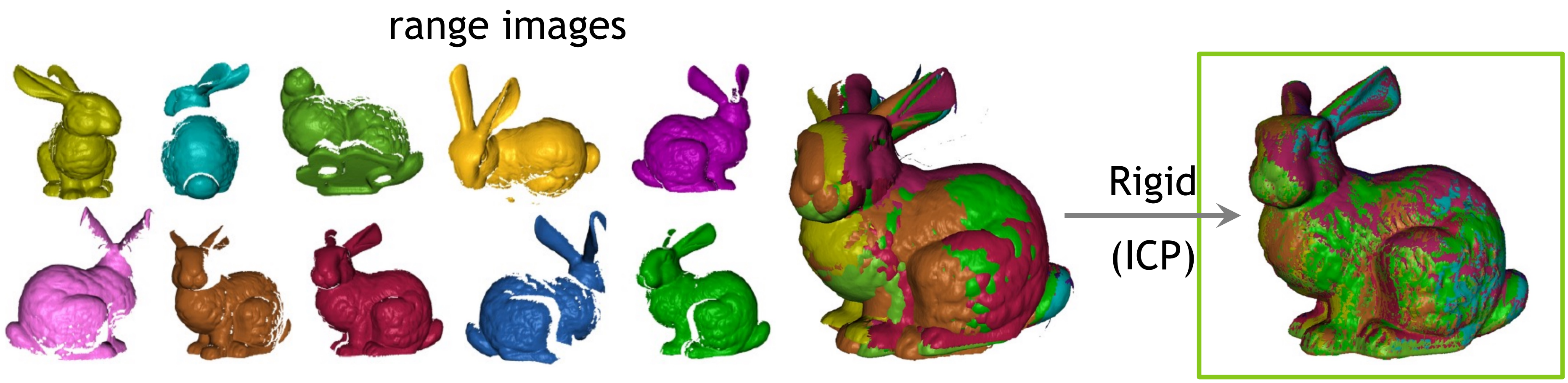
Shape Modeling and Geometry Processing

Inter-surface Mapping
Shape Matching
Functional Maps

INTERACTIVE GEOMETRY LAB

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Rigid Shape Matching

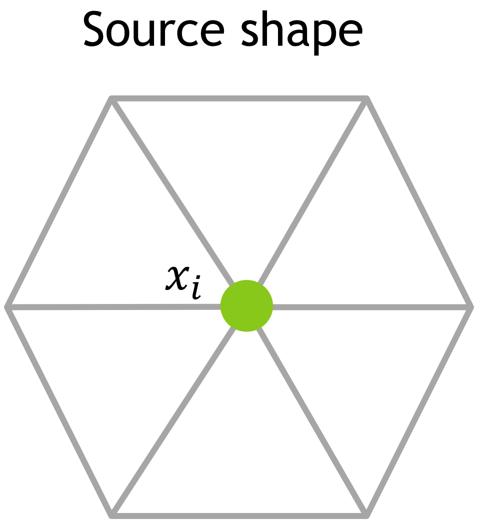


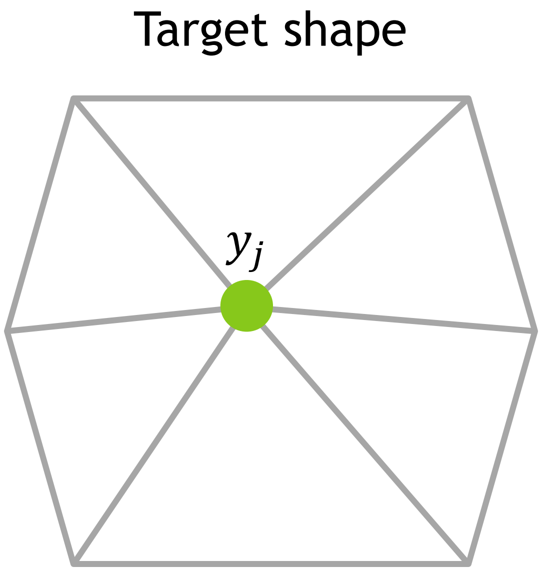
- Find the optimal rigid alignment between shapes
- Rigid alignment: rotation + translation (compact for optimization)

Non-Rigid Shape Matching

- No compact representation for non-rigid matching
- Find the map (correspondences) between two shapes directly

Shape Matching – what is a map?

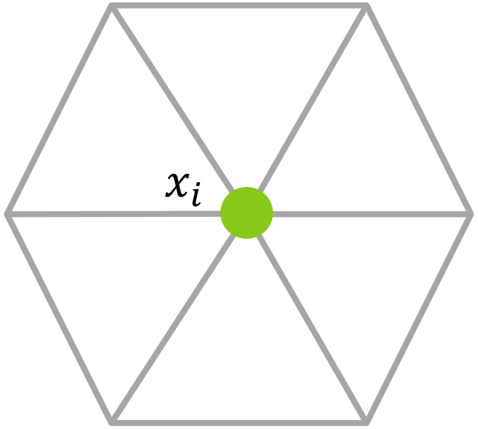




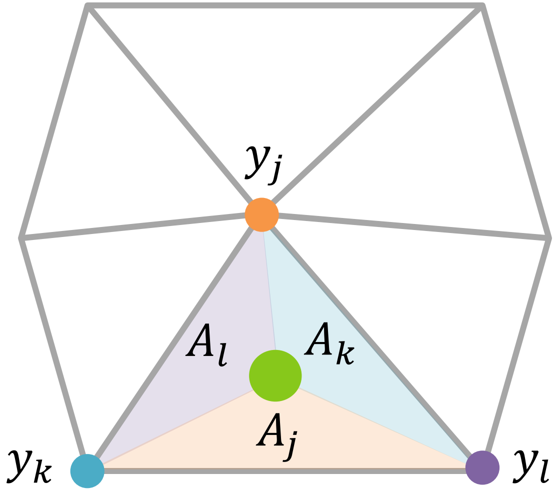
Vertex-to-vertex map: $\Pi(x_i) = y_j$

Shape Matching – what is a map?

Source shape



Target shape



Barycentric coordinate:

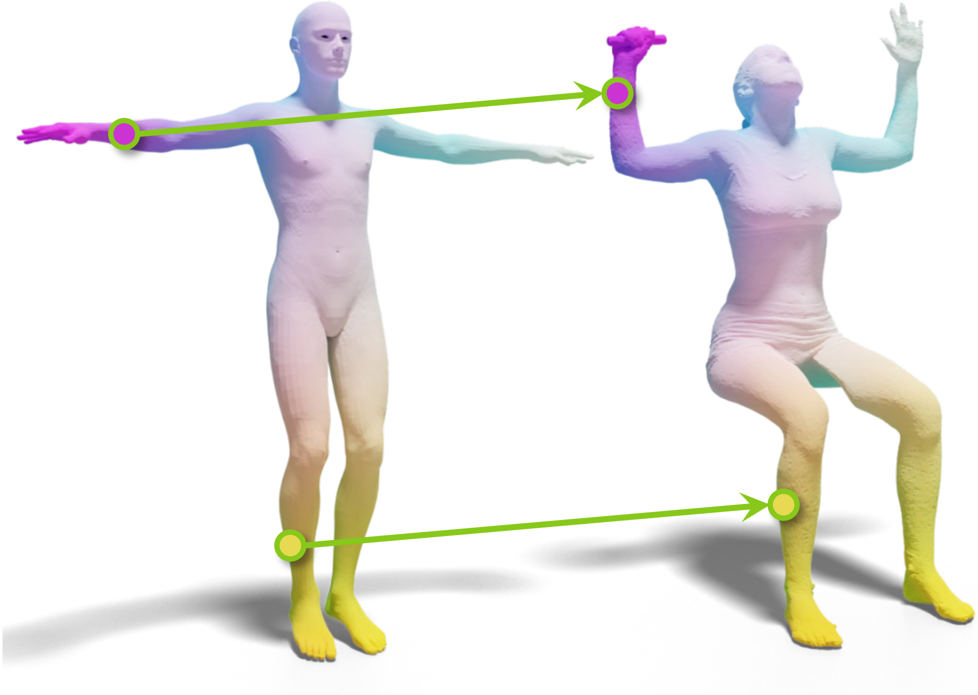
$$w_j = \frac{A_j}{A_j + A_l + A_k}$$

$$w_k = \frac{A_k}{A_j + A_l + A_k}$$

$$w_l = \frac{A_l}{A_j + A_l + A_k}$$

Vertex-to-point map: $\Pi(x_i) = w_j y_j + w_k y_k + w_l y_l$

Map Visualization



Points in correspondences
are assigned the same color

Shape Matching - Applications

- Motion transfer
  - Mocap captures the motion/expression of the actor
  - Motion/expression transferred to the Ape's model via correspondences

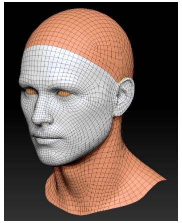
"The Hobbit"

"Dawn of the Planet of the Apes"

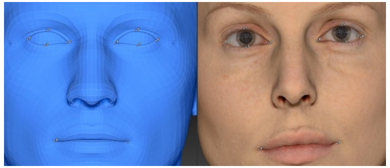
Shape Matching - Applications

Zwrap plugin for R3DS – Russian3DScanner

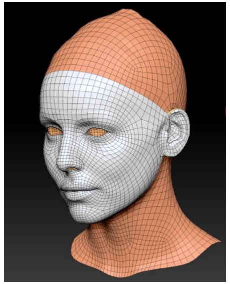
(left) template
(right) scan



Manually establish corres.



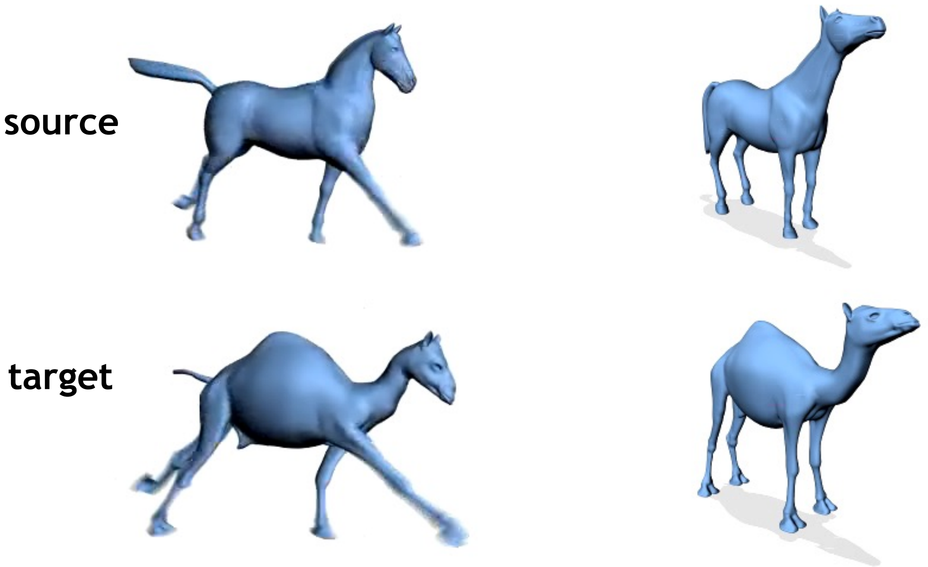
Wrapped results
- Topology from template
- Identity from scan



Shape Matching - Applications

- Motion transfer

- Motion in the source: S' = S + D
- Given the correspondences f between the source S and the target T
- Transferred to target: T' = T + f(D)

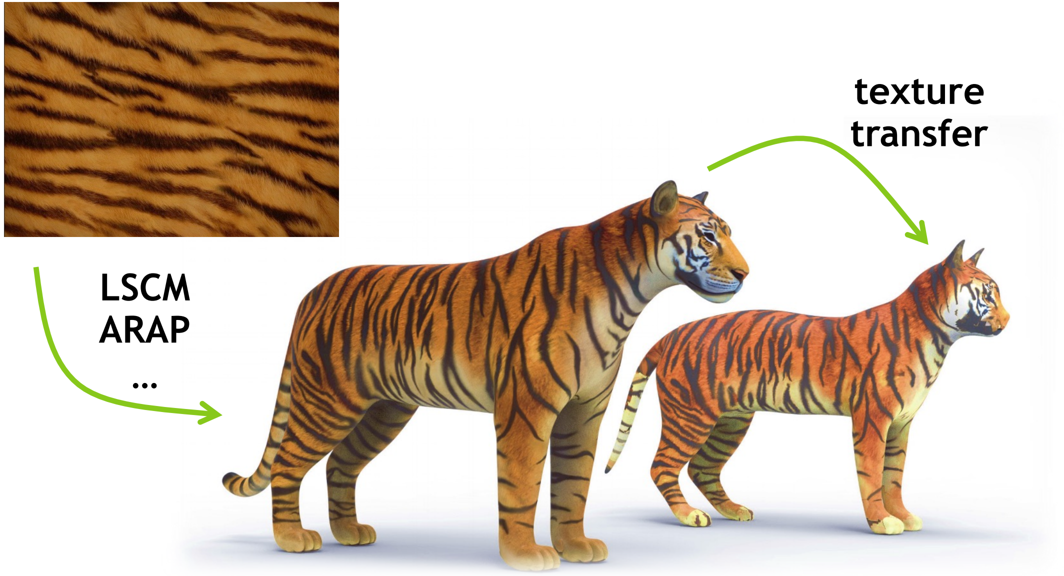


"Deformation Transfer for Triangle Meshes"
R. Sumner and J. Popovic, SIGGRAPH 2004

Shape Matching - Applications

- Texture transfer

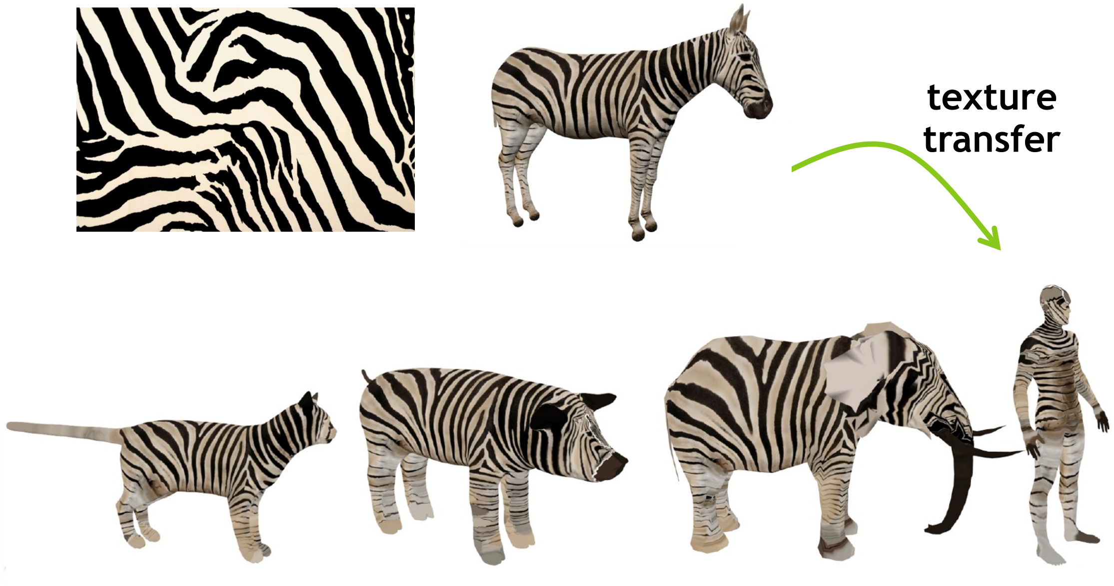
  - Paint texture on tiger shape
  - Transfer the texture to other shapes via correspondences



"Hierarchical Functional Map between Subdivision Surfaces"
M. Shoham, A. Vaxman, M. Ben-Chen, SGP2019

Shape Matching - Applications

- Texture transfer
  - Paint texture on zebra shape
  - Transfer the texture to other shapes via correspondences

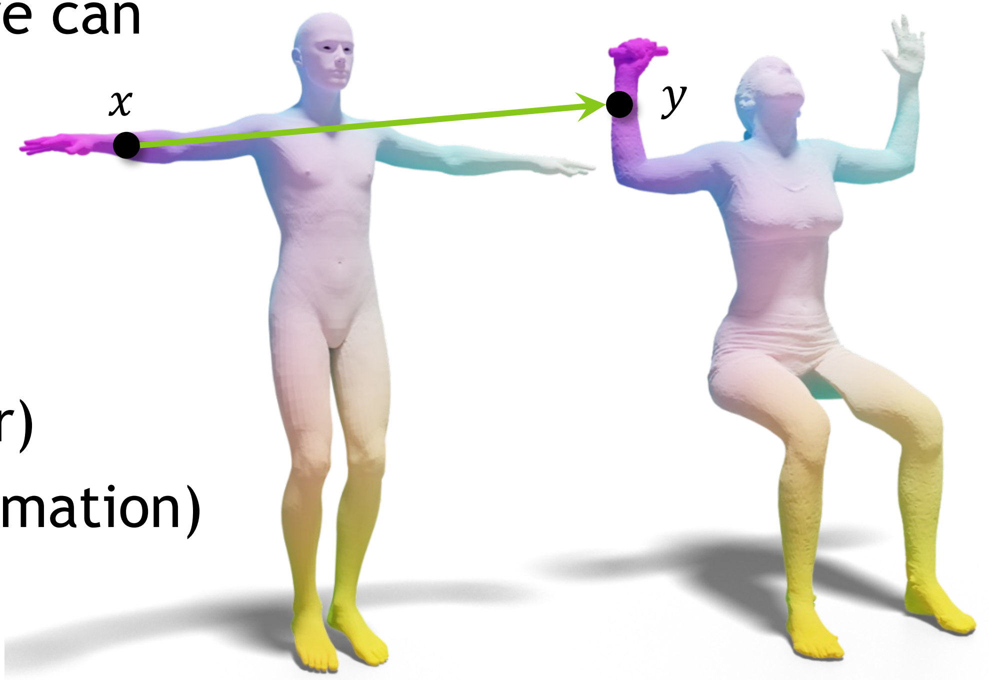


“Interactive Curve Constrained Functional Maps”
A.Gehre, M.Bronstein, L.Kobbelt, J. Solomon, SGP2018

Shape Matching - Applications

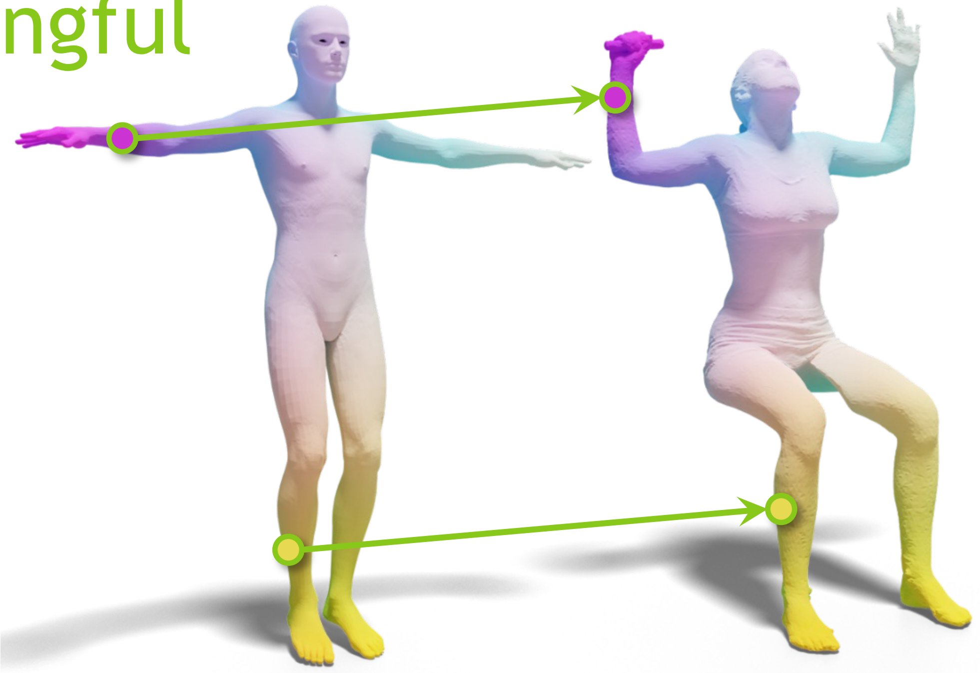
With given correspondences, we can transfer from x to y:

- uv-coordinate
- (R,G,B) color
- segmentation label
- motion (displacement vector)
- deformation (affine transformation)
- …



What is a good map?

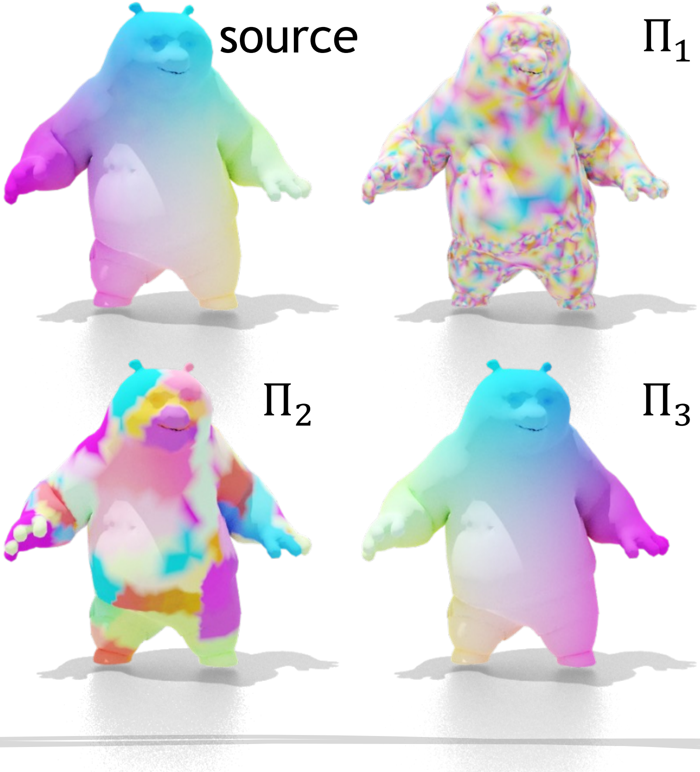
- Semantically meaningful
- Smooth
- Bijective
- Conformal
- ...



What is a good map?

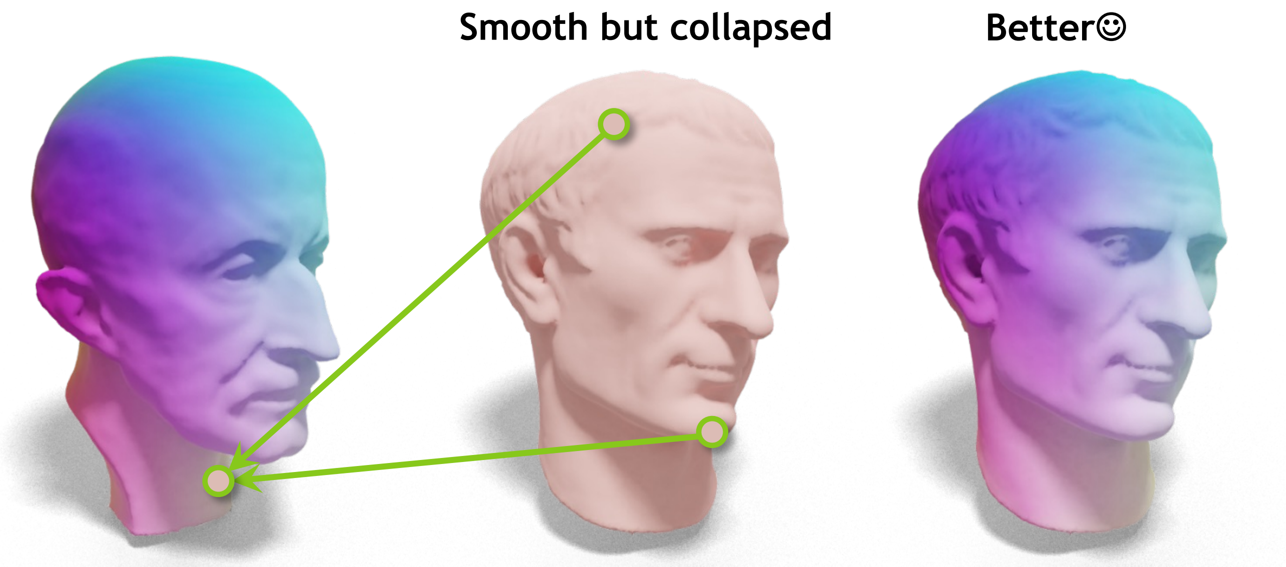
- Semantically meaningful
- Smooth
- Bijective
- Conformal
- ...

$\Pi_1 < \Pi_2 < \Pi_3$



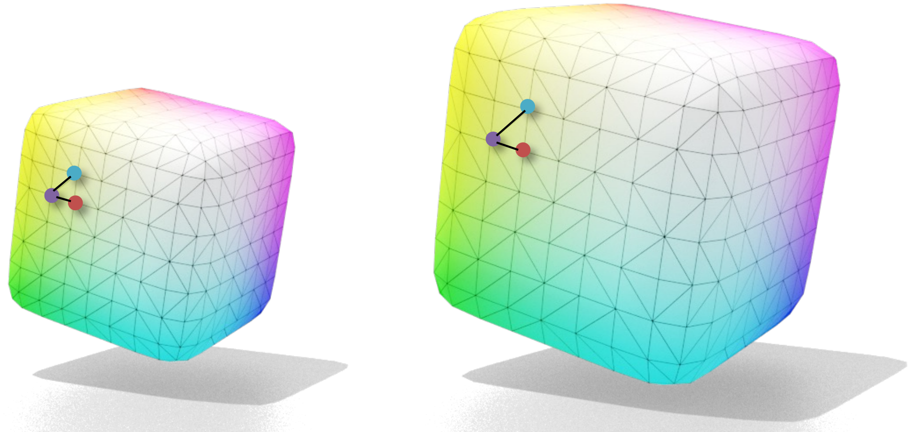
What is a good map?

- Semantically meaningful
- Smooth
- Bijective
- Conformal
- ...



What is a good map?

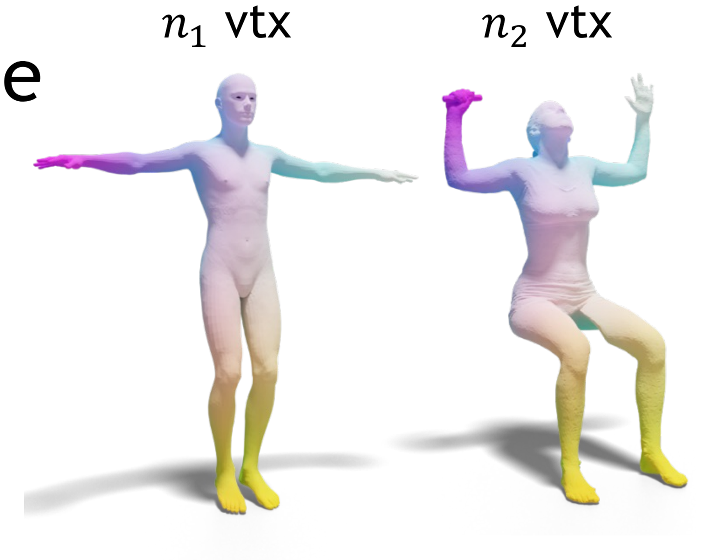
- Semantically meaningful
- Smooth
- Bijective
- Conformal
- ...



Recall the LSCM energy to measure angle-preservation

Challenges to find a good map

- Large search space
  - For each vertex on the male shape, it has n_2 choices
  - $n_2^{n_1}$ possible maps
  - $n > 10{,}000$



Challenges to find a good map

- Discrete search space
  - Recall LSCM for parameterization

$$\min_{U \in R^{2n}} \sum_{\text{triangles } T} A_T E_{\text{LSCM}}(\, J_T \,)$$

- J_T: Jacobian from the 3D triangle in the original shape to 2D triangle in the uv-coordinate
- Quadratic w.r.t. $U \in R^{2n}$, continuous space!

Challenges to find a good map

- Discrete search space
  - Try to generalize to shape matching

$$\min_{\Pi \in [1,\cdots,n_1]^{n_2}} \sum_{\text{triangles } T} A_T E_{\text{LSCM}}(J_T)$$

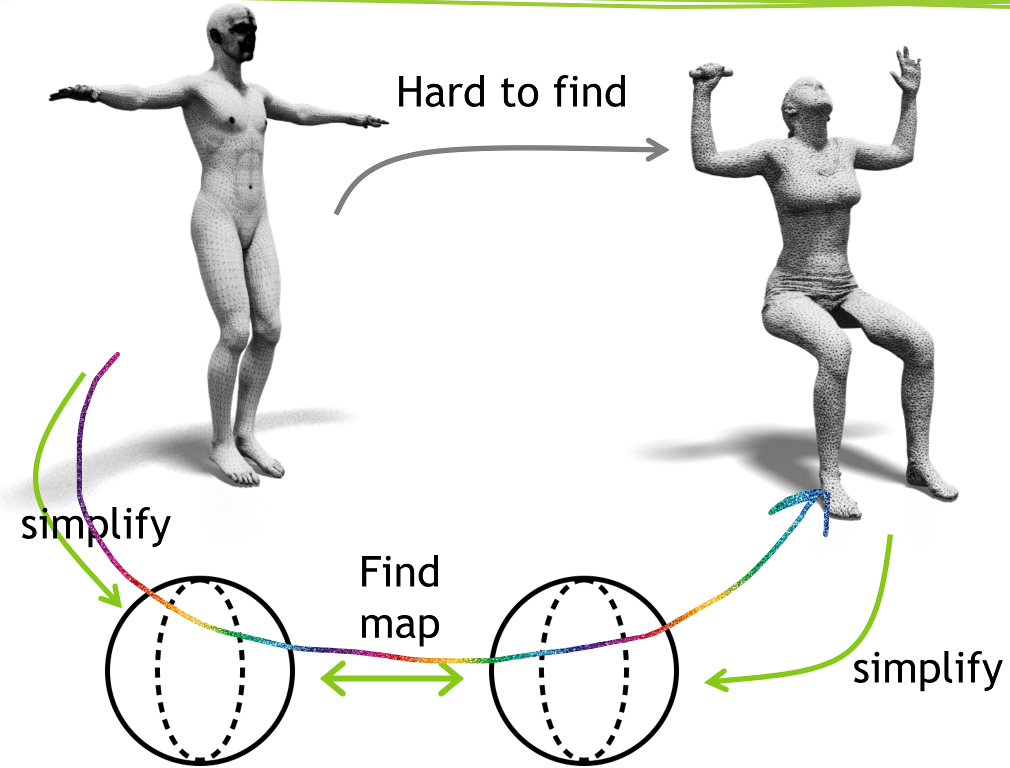
- J_T: Jacobian from the triangle in the source shape to the mapped triangle
- Discrete search space $\Pi \in [1,\cdots,n_1]^{n_2}$, gradient is not well-defined! Hard to optimize

Solutions

- Reduce search space size
  - Parameterization-based methods
- Find a continuous search space
  - Functional map-based methods

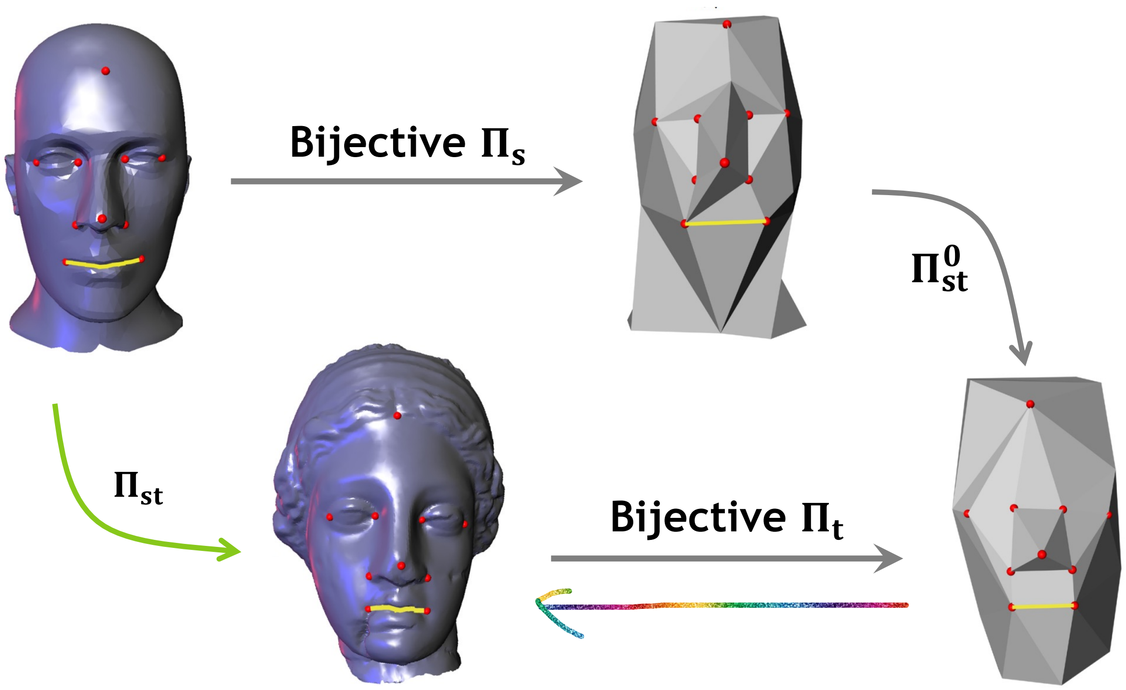
Parameterization-based methods

General Idea



- Map the complicated 3D shape to simpler domain
  - Sphere
  - Plane (square)
  - Simplified meshes…
- Find correspondences between the “simplified shapes”
- Propagate the correspondences back to original shapes (as map composition)

Multiresolution Mesh Morphing



$$\Pi_{st} = \Pi_t^{-1} \circ \Pi_{st}^0 \circ \Pi_s$$

- Q1: how to simplify shapes with bijective map?
- Q2: how to find correspondences at coarse level?

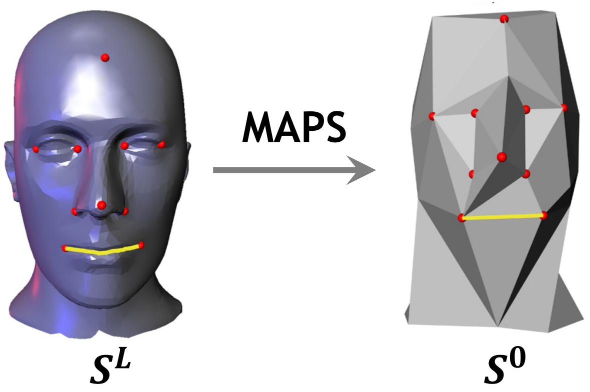
"Multiresolution Mesh Morphing"
A.Lee, D. Dobkin, W. Sweldens, P. Schroder, SIGGRAPH 1999

“MAPS”

Q1: how to simplify shapes with bijective map?

Key ideas:

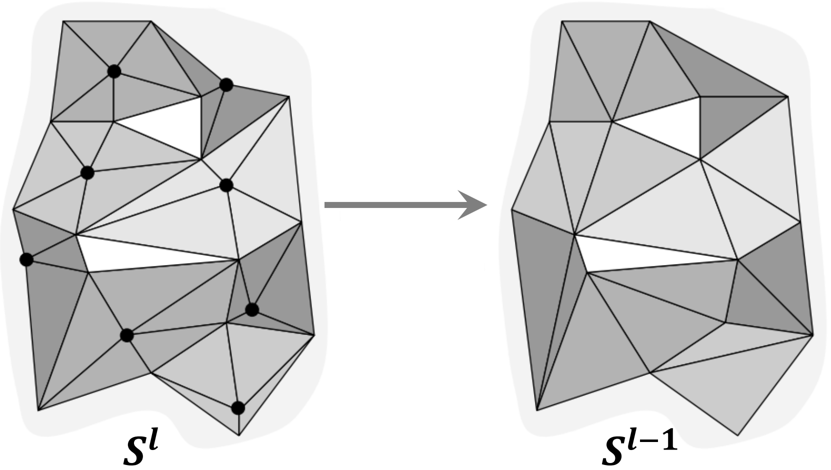
- Construct mesh hierarchy: $S^L \to \cdots \to S^l \to S^{l-1} \to \cdots \to S^0$
- $S^l \to S^{l-1}$:
  - Remove vertices
  - Fill holes
  - Establish bijective mapping



“Maps: Multiresolution Adaptive Parameterization of Surfaces”
A. Lee, W. Sweldens, P. Schroder, L. Cowsar, D. Dobkin, SIGGRAPH 1998

"MAPS" – vertex removal

$S^l \to S^{l-1}$: vertex removal

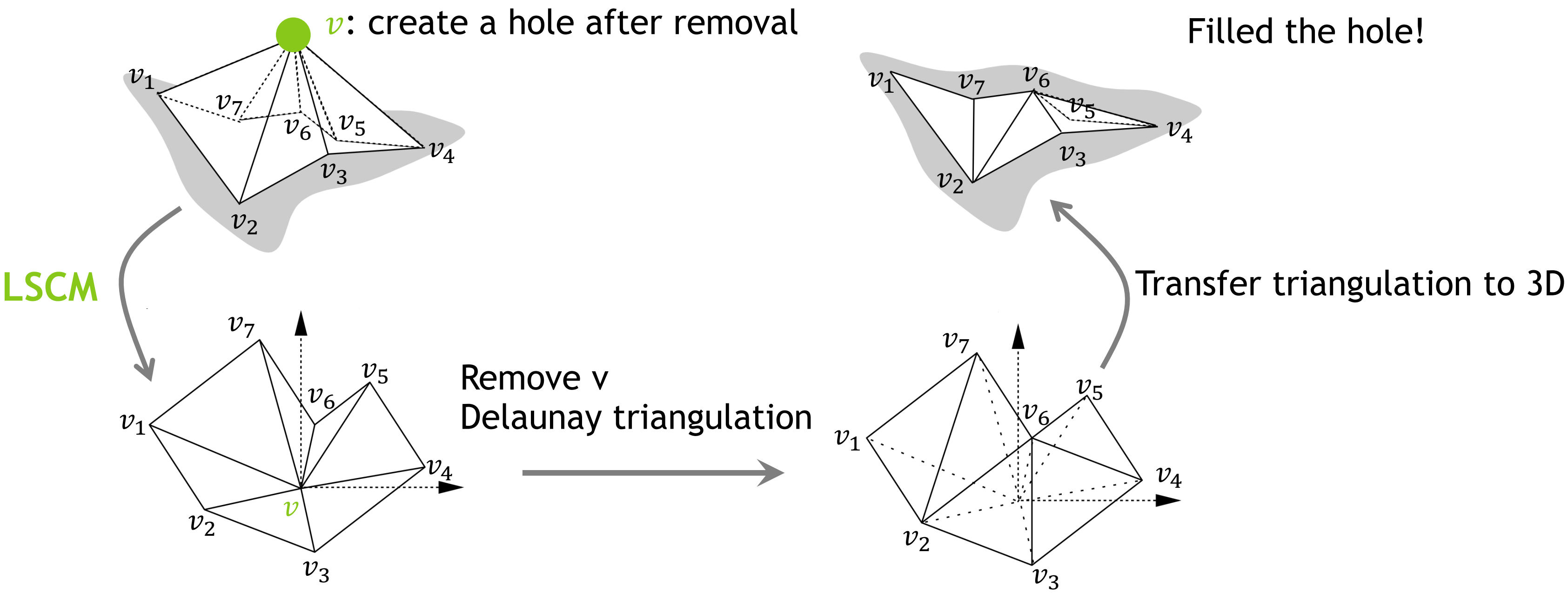


1. Initialize $V = V^l, A = [\,], B = [\,]$
2. Repeat until V is empty:
   1. Select one vertex from $v \in V$
   2. A.append(v), — Expand the maximally independent vtx set
      B.append($\mathcal{N}(v)$), — Mark the neighbor as non-removable
      V.pop($v \cup \mathcal{N}(v)$) — Update the search queue
3. $V^{l-1} \leftarrow V^l \backslash A$

"Maps: Multiresolution Adaptive Parameterization of Surfaces"
A. Lee, W. Sweldens, P. Schroder, L. Cowsar, D. Dobkin, SIGGRAPH 1998

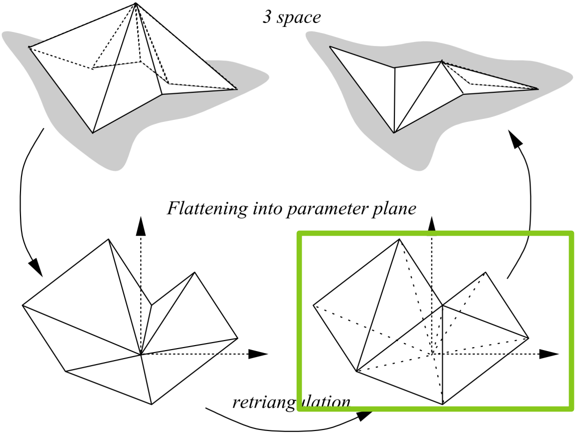
Note: (priority queue) vtx with a flat neighborhood will be selected first - recall Laplacian!

“MAPS”- flattening & retriangulation

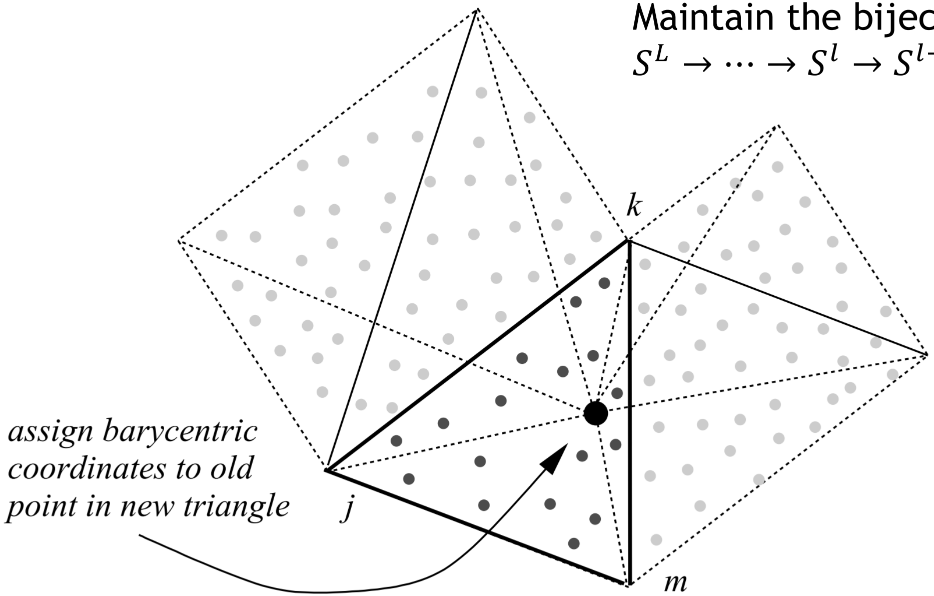


“Maps: Multiresolution Adaptive Parameterization of Surfaces”
A. Lee, W. Sweldens, P. Schroder, L. Cowsar, D. Dobkin, SIGGRAPH 1998

“MAPS”- flattening & retriangulation

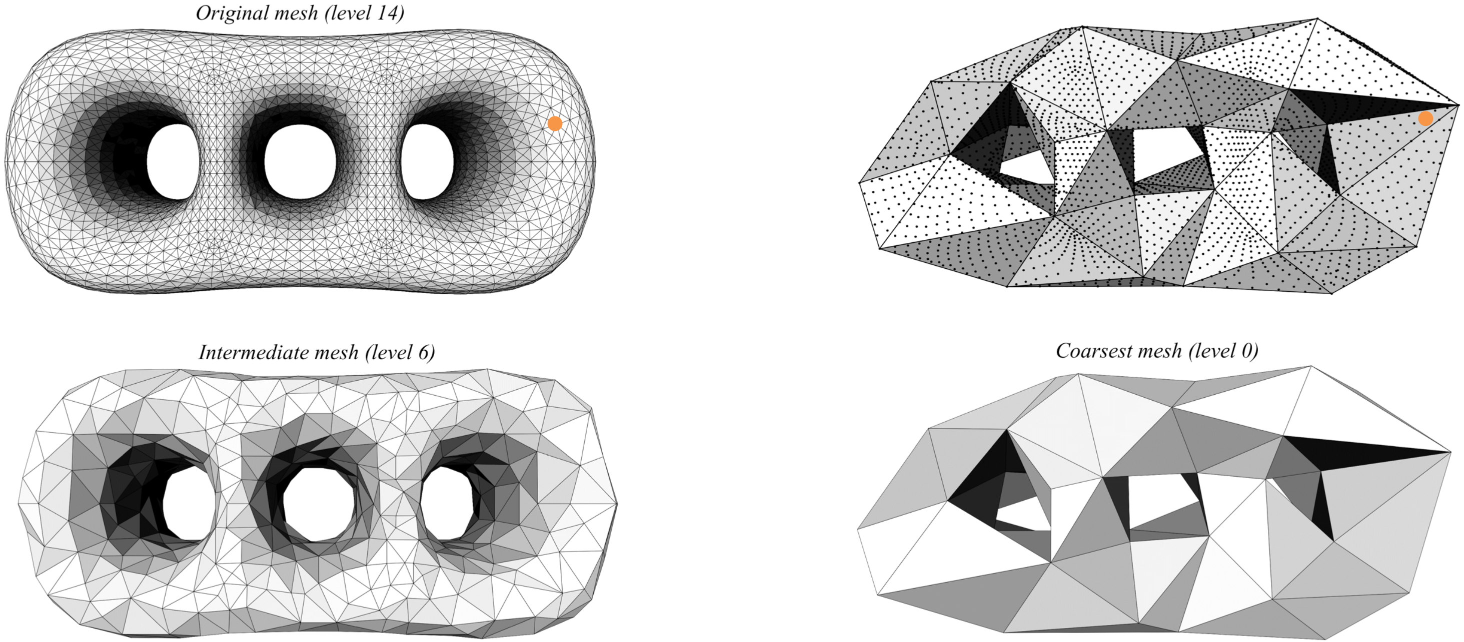


Maintain the bijective map from
$S^L \to \cdots \to S^l \to S^{l-1}$



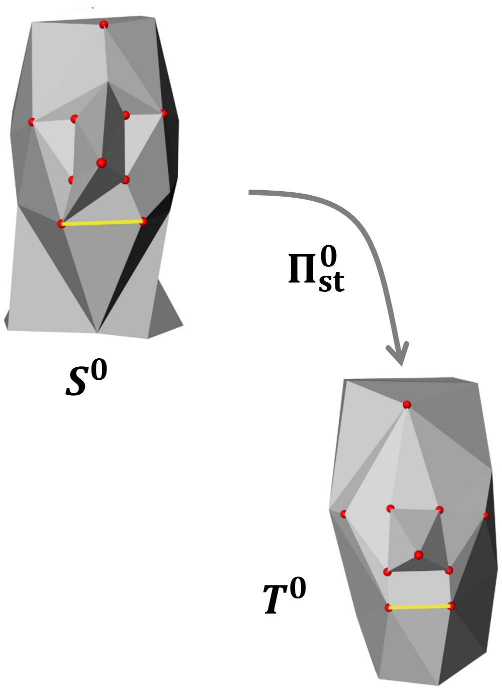
“Maps: Multiresolution Adaptive Parameterization of Surfaces”
A. Lee, W. Sweldens, P. Schroder, L. Cowsar, D. Dobkin, SIGGRAPH 1998

“MAPS”- flattening & retriangulation

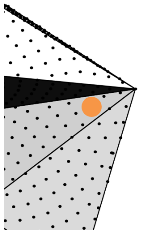


“Maps: Multiresolution Adaptive Parameterization of Surfaces”
A. Lee, W. Sweldens, P. Schroder, L. Cowsar, D. Dobkin, SIGGRAPH 1998

Parameterization-based methods

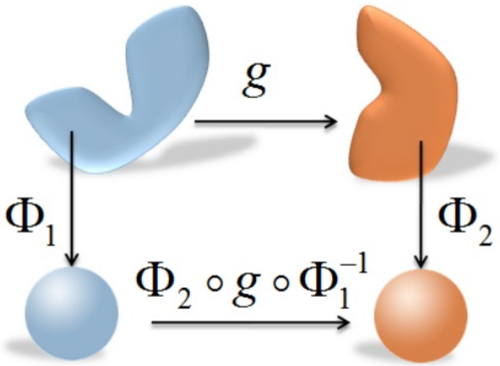


- Given corresponding landmarks (red points)
- Global alignment via landmarks (rigid - ICP)
- Vertex-to-point mapping from S^0 to T^0
  - For each vertex in S^0 , find its closest point in the closest triangle in T^0
  - The points (= vertices in the original shape) in the triangles of S^0 are map using barycentric coordinates



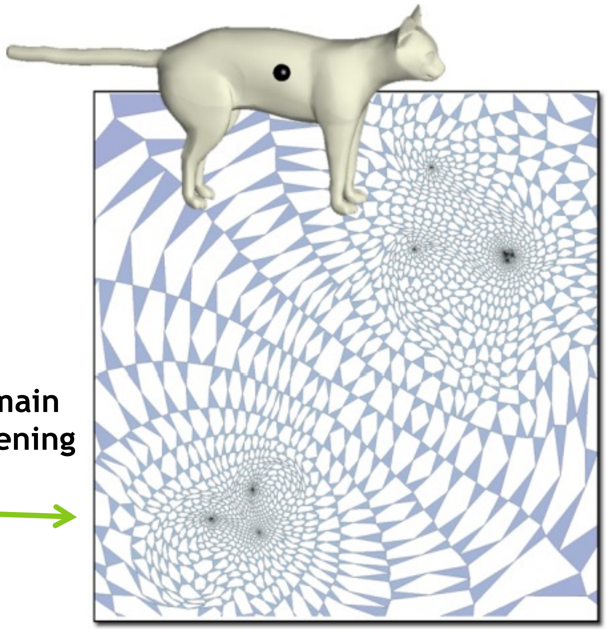
"Multiresolution Mesh Morphing"
A.Lee, D. Dobkin, W. Sweldens, P. Schroder, SIGGRAPH 1999

Mobius-Voting for surface correspondence



Mobius Transformation
& vote for correspondences

Canonical Domain
Mid-edge Flattening



"Mobius Voting for Surface Correspondence"
Y. Lipman, T. Funkhouser, SIGGRAPH 2009

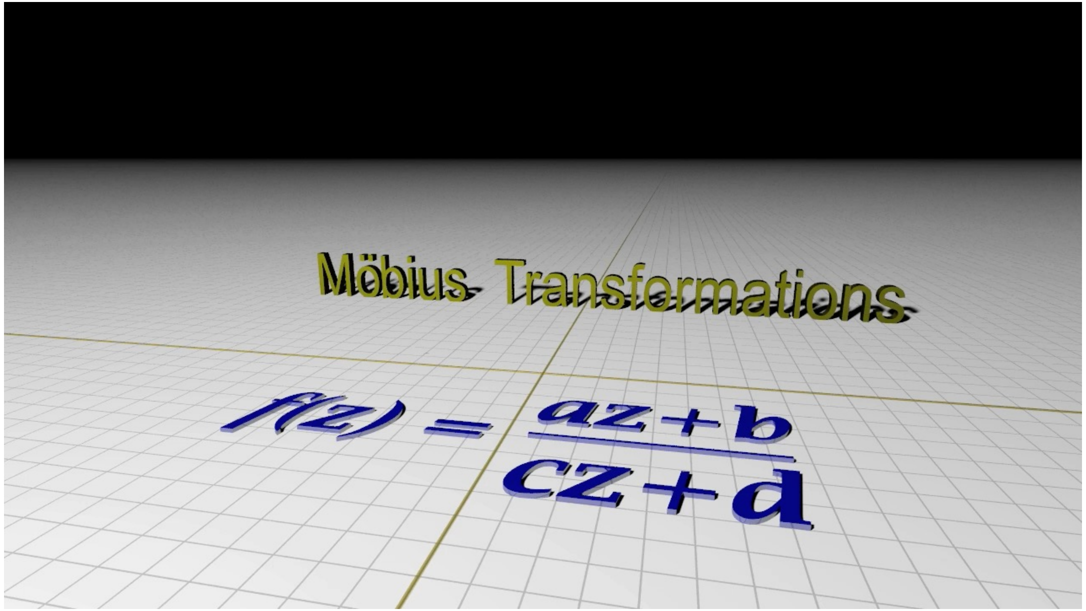
Mobius Transform: $f(z) = \frac{az+b}{cz+d}$

Mobius Transformation $f(z) = \frac{az+b}{cz+d}$

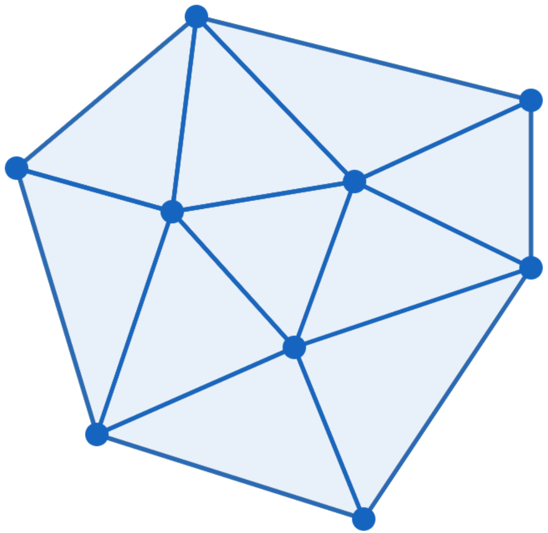
- Translation $f(z) = z + b$
- Rotation $f(z) = e^{i\theta} z$
- Scaling $f(z) = kz$
- Inversion $f(z) = \frac{1}{z}$
- …

In general: maps every line/circle to line or circle

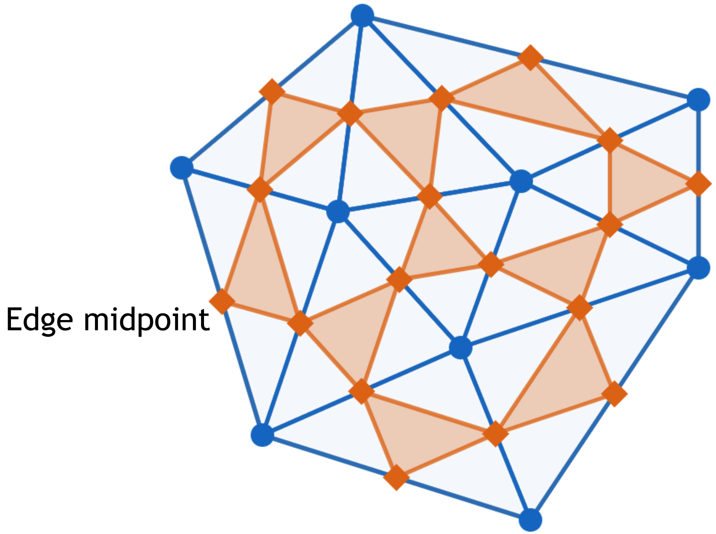
"Möbius Transformations Revealed [HD] - YouTube"



Mobius-Voting: Mid-Edge Mesh



Mesh = (● , △)

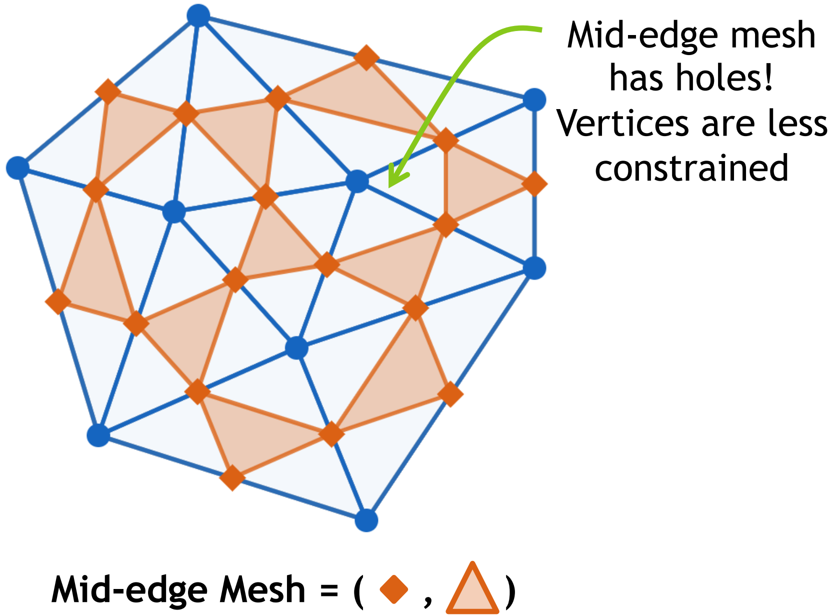


Mid-edge Mesh = (◆ , △)

"Mobius Voting for Surface Correspondence"
Y. Lipman, T. Funkhouser, SIGGRAPH 2009

Mobius-Voting: Mid-Edge Mesh

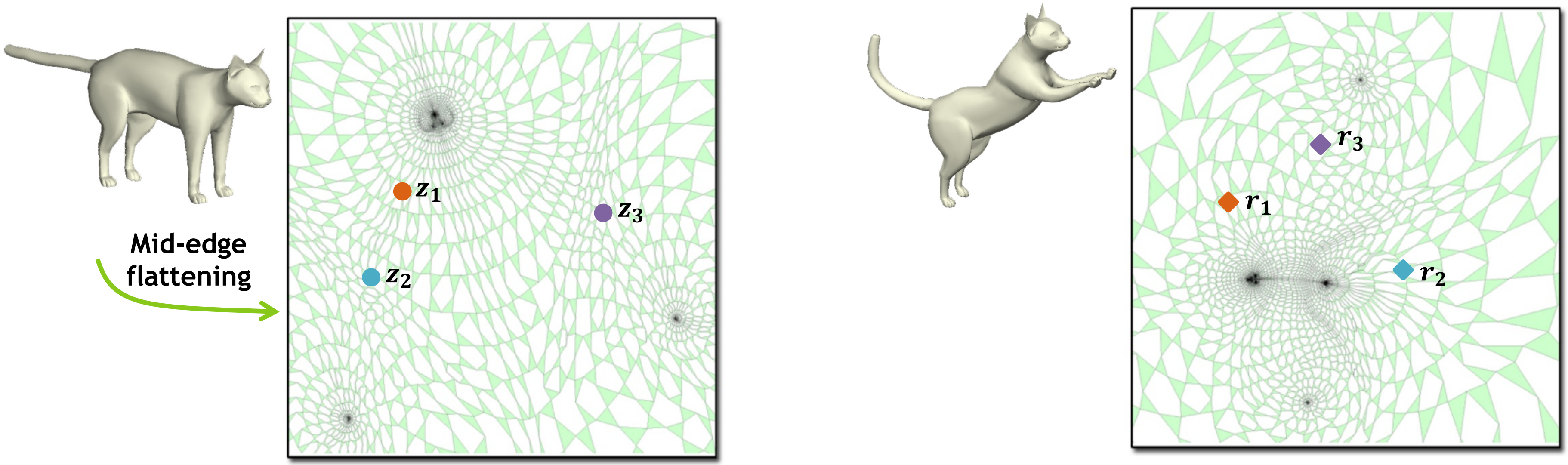
- For non-developable surface:
  $$E_{LSCM}(U) \neq 0 \; \forall \; U \neq \text{const.}$$
- i.e., any non-trivial uv-flattening has non-zero (discrete) conformal error
- Mid-edge mesh: can be flattened with zero (discrete) conformal error



Recall: zero conformal error means each face undergoes a similarity transformation

"Mobius Voting for Surface Correspondence"
Y. Lipman, T. Funkhouser, SIGGRAPH 2009

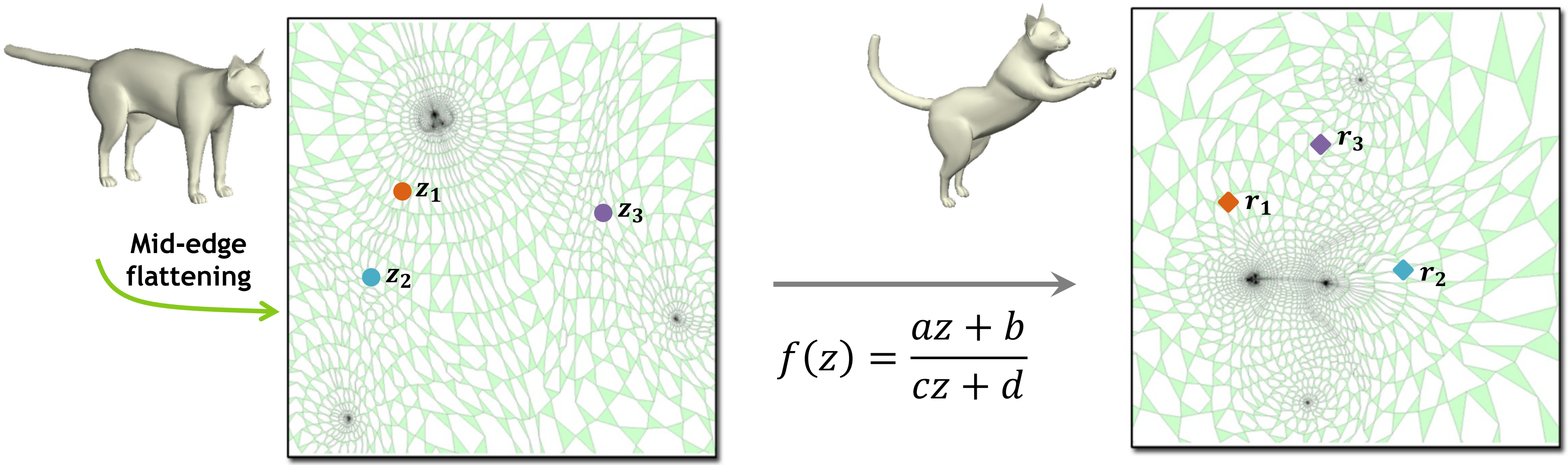
Mobius-Voting: solve transformation



We need 3 corresponding points in the canonical domain to solve for the Mobius transformation $f(z) = \frac{az+b}{cz+d}$

i.e., solve $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d})$ from $\boldsymbol{f}(\boldsymbol{z_i}) = \boldsymbol{r_i}, \boldsymbol{i} = \mathbf{1}, \mathbf{2}, \mathbf{3}$

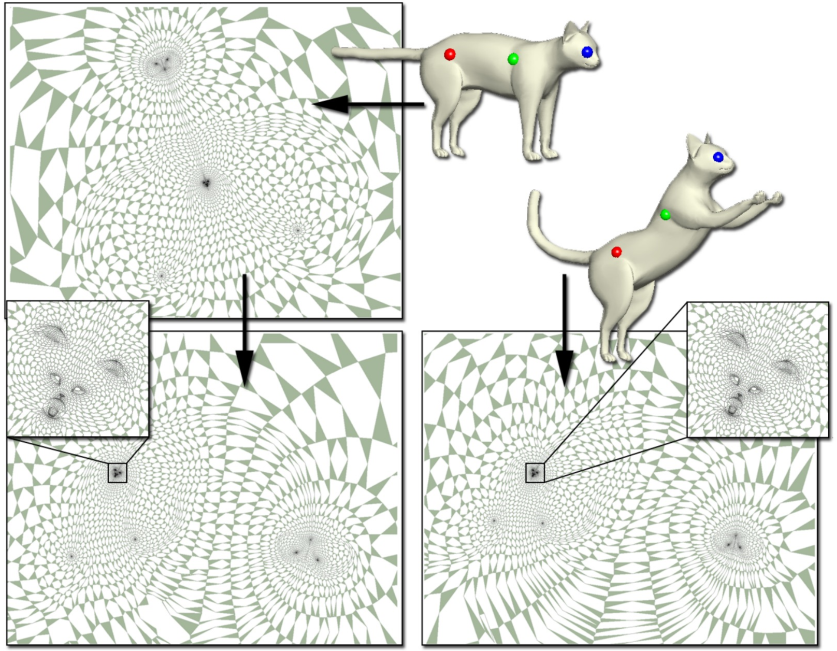
Mobius-Voting: solve transformation



$$f(z) = \frac{az + b}{cz + d}$$

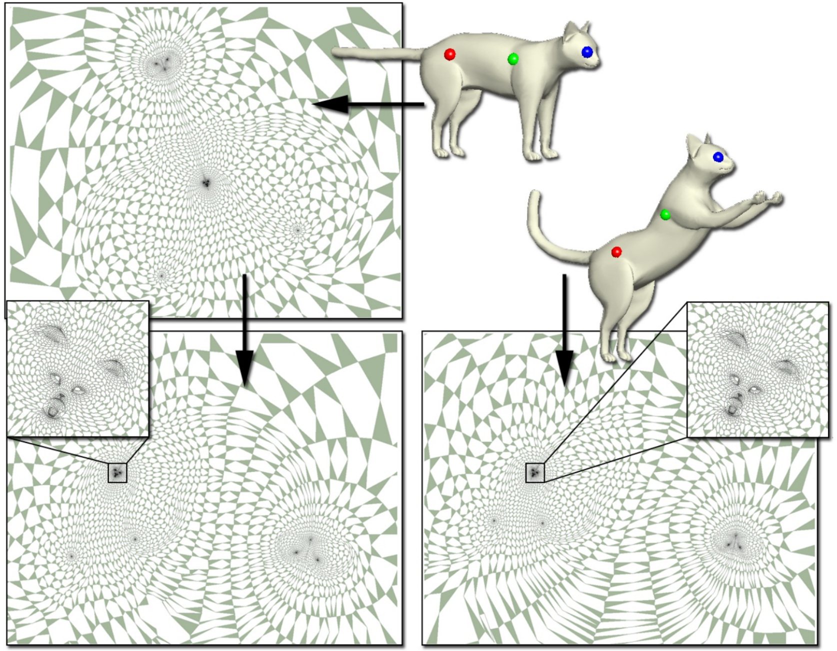
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} r_2 - r_3 & r_1 r_3 - r_1 r_2 \\ r_2 - r_1 & r_1 r_3 - r_3 r_2 \end{pmatrix}^{-1} \begin{pmatrix} z_2 - z_3 & z_1 z_3 - z_1 z_2 \\ z_2 - z_1 & z_1 z_3 - z_3 z_2 \end{pmatrix}$$

Mobius-Voting: solve transformation

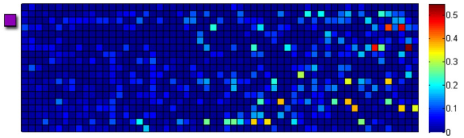


- If the three chosen points are in true correspondences, after applying the Mobius transformation (bottom row), the two flattenings look similar.
- I.e., only need to find 3 pairs of accurate correspondences, instead of N pairs.

Mobius-Voting: correspondences



1. Find correspondences in the canonical domain (complex plane)
2. Measure the distance between the corresponding points in the complex plane
3. The average error is used to score the 3 chosen pairs for Mobius transformation computation



Functional Map for Matching

Solutions

- Reduce search space size
  - Parameterization-based methods
- Find a continuous search space
  - Functional map-based methods
  - Instead of finding correspondences between vertices on shapes, try to find correspondences between functions defined on shapes.

Functional Map

- Function $f(\cdot)$: $x \rightarrow y = f(x)$
  - Maps a (high-dim) point to a point
  - E.g., $f(x) = x^2$

Functional Map

- Functional $F(\cdot): f \rightarrow g = F(f)$
  - Maps a function $f(\cdot)$ to another function $g(\cdot)$
  - E.g., $\big(F(f)\big)(x) = \int_{-\infty}^{x} f(t)dt$

Function defined on shape $f: R^3 \rightarrow R$

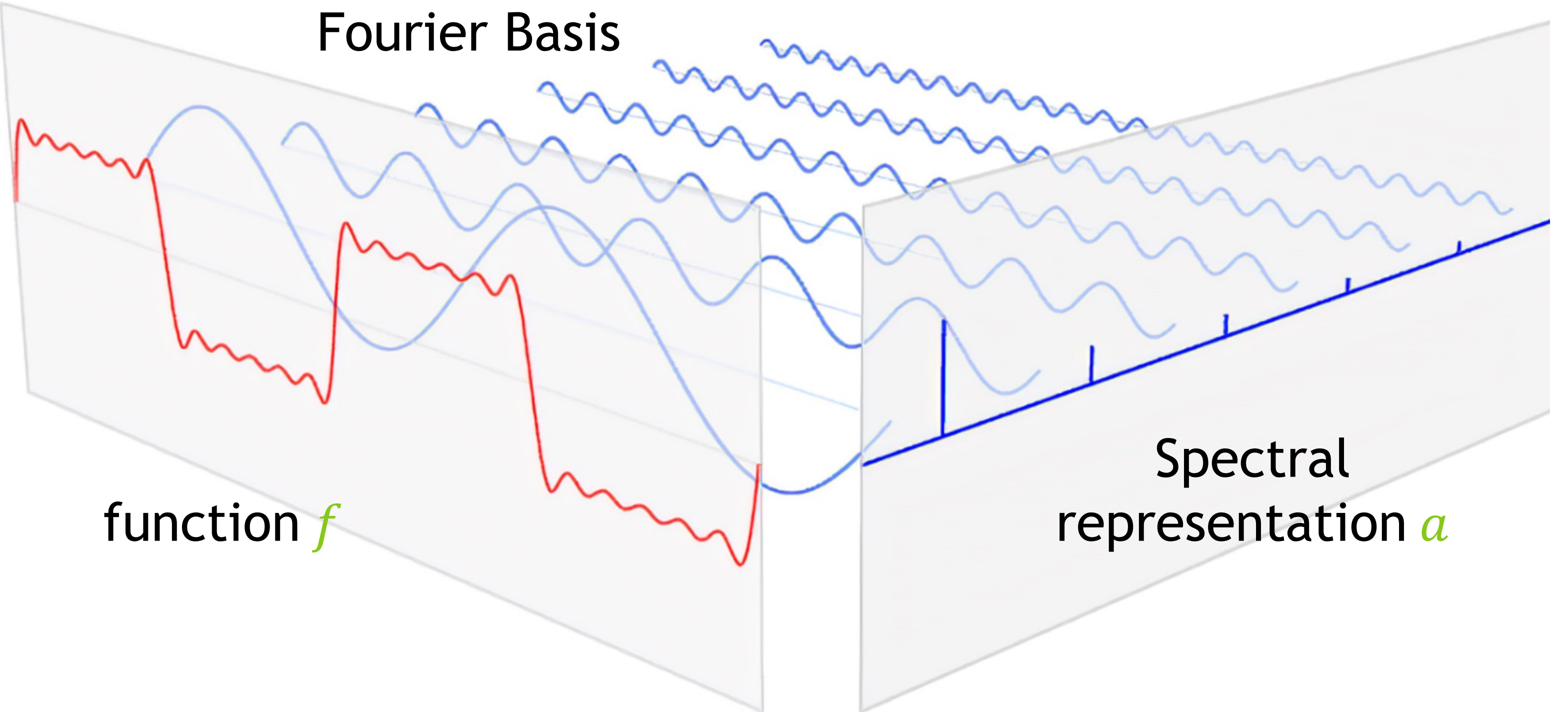
- Per-vertex function $f(v_i) = f_i$
- Piece-wise linear: for a point p in the triangle (v_i, v_j, v_k):

$$p = w_i v_i + w_j v_j + w_k v_k$$

- We can define

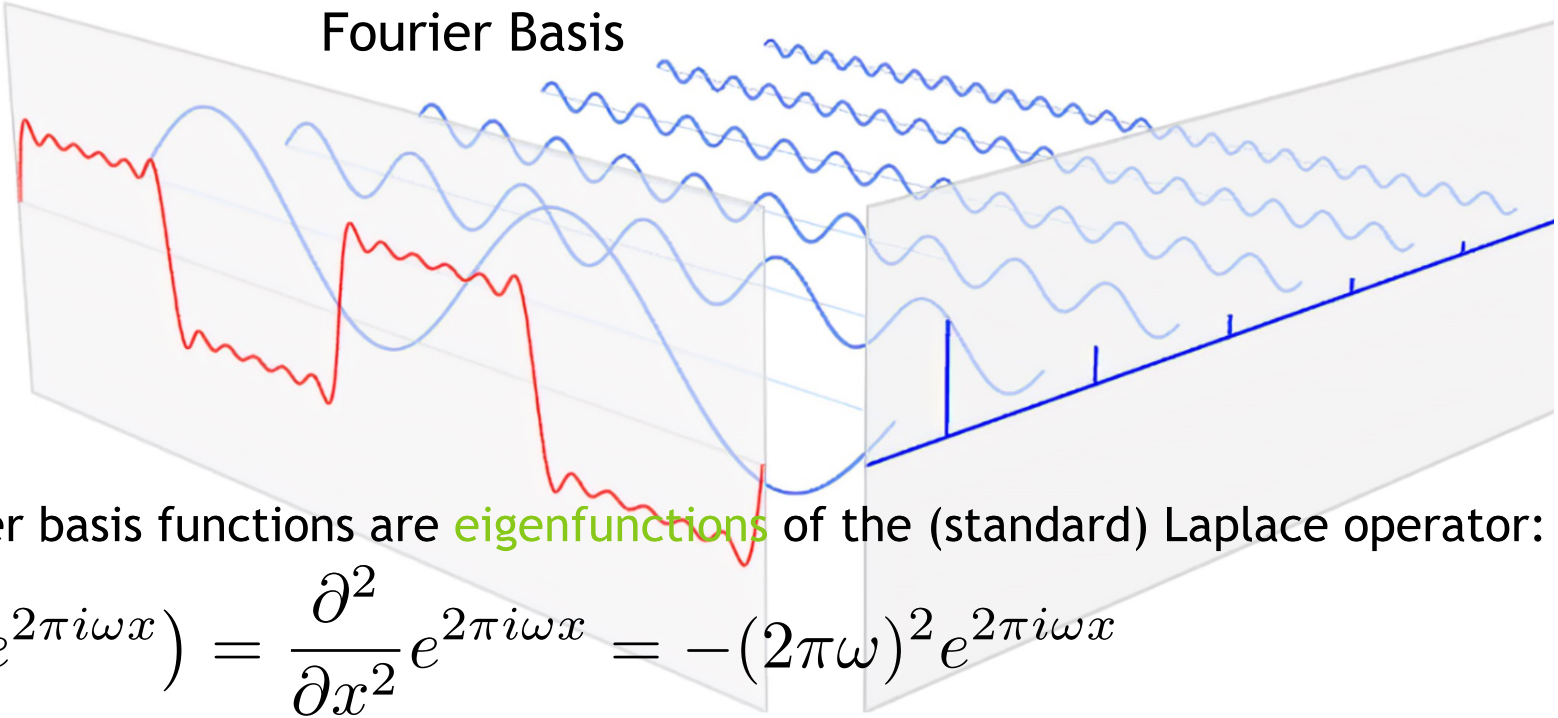
$$f(p) = w_i f(v_i) + w_j f(v_j) + w_k f(v_k)$$
$$= w_i f_i + w_j f_j + w_k f_k$$

Fourier Series



Fourier Series

Fourier Basis



Fourier basis functions are eigenfunctions of the (standard) Laplace operator:

$$\Delta\left(e^{2\pi i\omega x}\right) = \frac{\partial^2}{\partial x^2}e^{2\pi i\omega x} = -(2\pi\omega)^2 e^{2\pi i\omega x}$$

Laplace-Beltrami EigenBasis

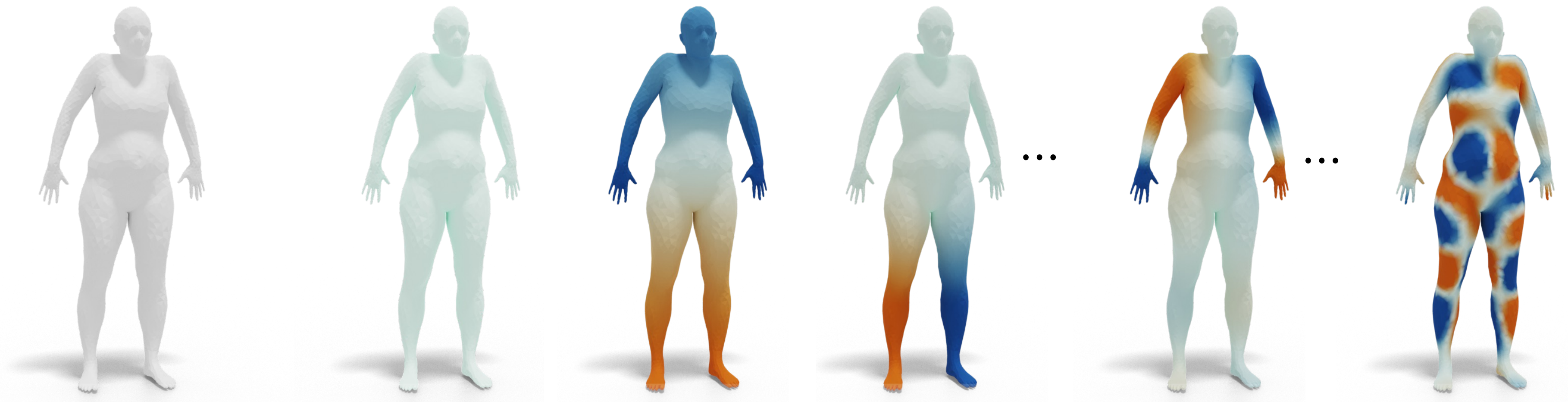
- Recall the cotangent Laplacian L
- Let's try to find its eigenvectors/eigenfunctions
- i.e., solve the Helmholtz equation

$$Lf = \lambda M f$$

- Note $L \in R^{n \times n}$, the eigenvector $f \in R^{n}$ therefore can be regarded as a basis function defined on the shape

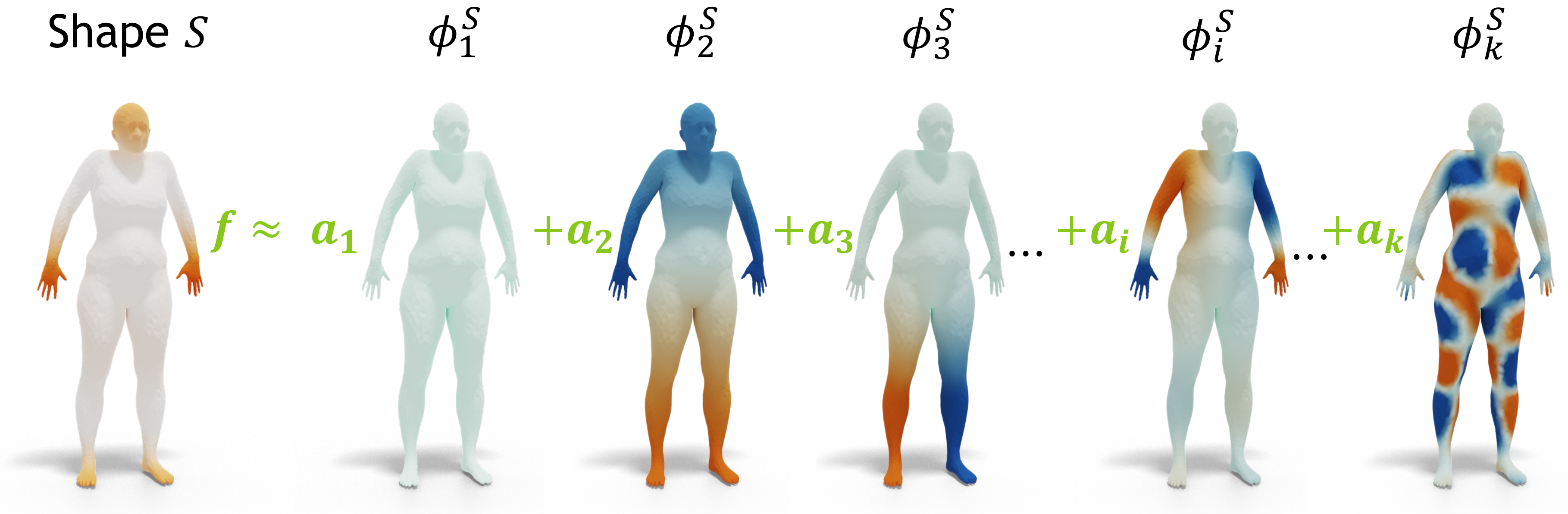
Laplace-Beltrami EigenBasis

Shape S $\quad \phi_1^S \quad \phi_2^S \quad \phi_3^S \quad \phi_i^S \quad \phi_k^S$



$$0 = \lambda_1^S \leq \lambda_2^S \leq \lambda_3^S \leq \cdots \lambda_i^S \cdots \leq \lambda_k^S$$

Laplace-Beltrami EigenBasis



function f

$$f \approx a_1\phi_1^S + a_2\phi_2^S + \cdots a_k\phi_k^S = \Phi^S a$$

Laplace-Beltrami EigenBasis

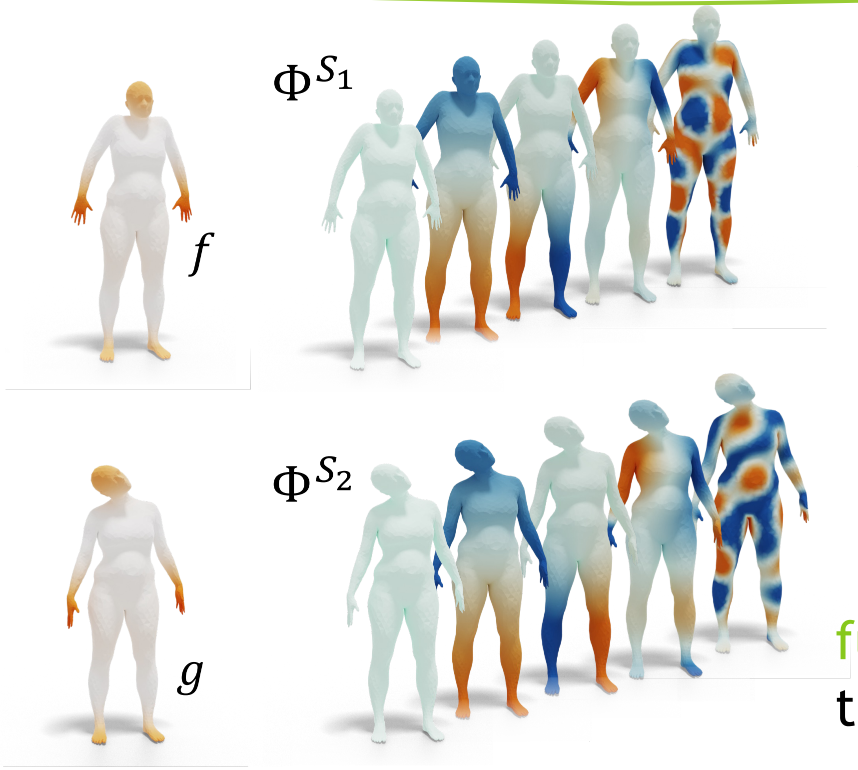
Shape S

function f

$$f \approx a_1\phi_1^S + a_2\phi_2^S + \cdots a_k\phi_k^S = \Phi^S a$$

It means, we can use a k-dim vector $a \in R^k$ to approximately represent the function $f \in R^n$, where $k \ll n$

Functional Map



$$f \approx \Phi^{S_1} a$$

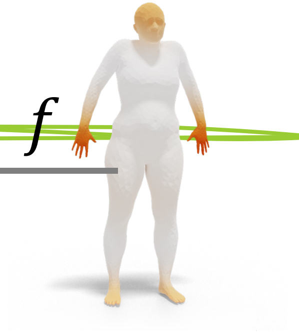
$$Ca = b$$

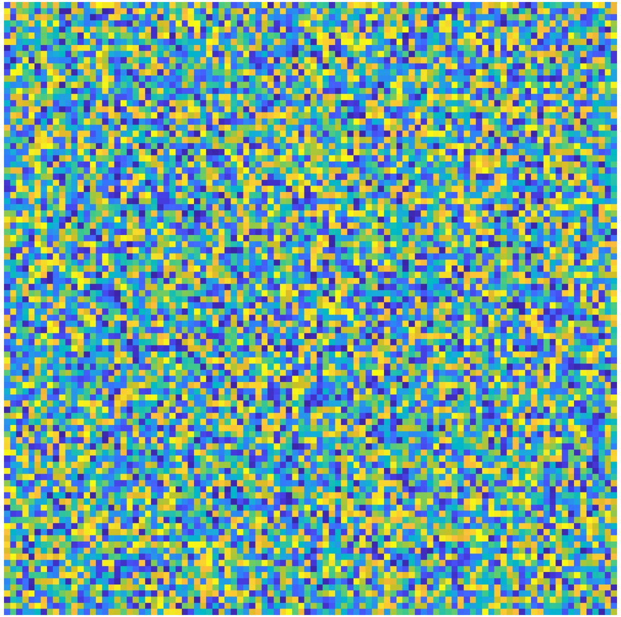
$$g \approx \Phi^{S_2} b$$

functional map: the matrix C that transports the coefficients from Φ^{S_1} to Φ^{S_2}

Functional Map

$a = (\Phi^{S_1})^{\dagger} f$





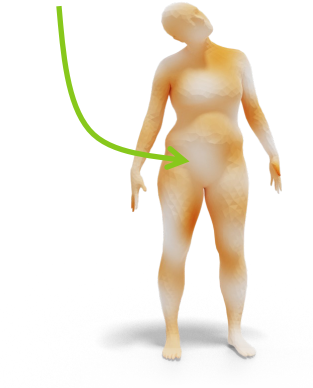
Functional map C

a

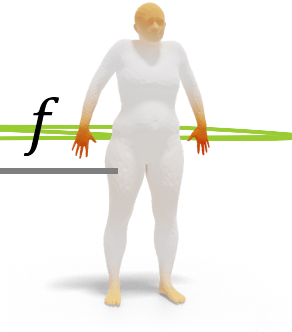
$=$

b

$\hat{g} = \Phi^{S_2} b$

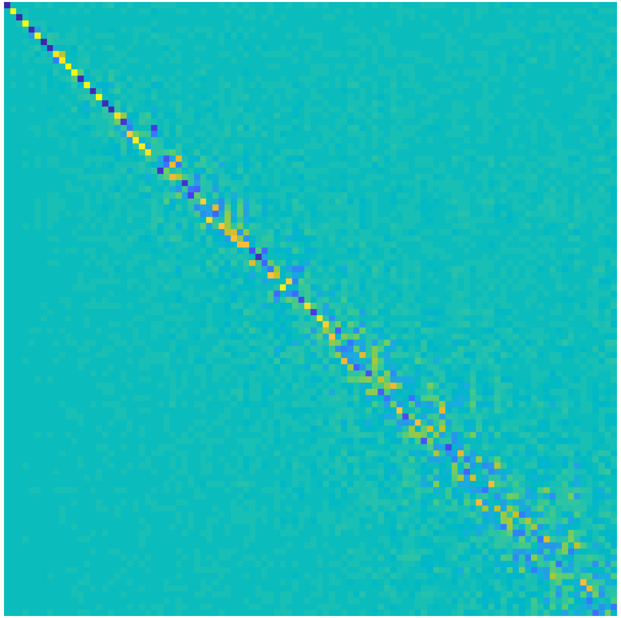


Functional Map



f

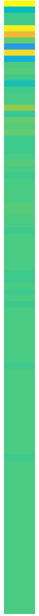
$a = (\Phi^{S_1})^{\dagger} f$



Functional map C

a

$=$



b

$\hat{g} = \Phi^{S_2} b$

Functional Map Pipeline

- Q1: How to find such a good functional map?
- Q2: How to recover a pointwise map from a functional map (matrix!)?

Functional Map Computation

$$C_{12}^{*} = \underset{C}{\operatorname{argmin}} \ \|CA \ - B\|_F^2$$

Descriptor preservation [OBCS*12]

$$+ w_1\|C\Delta_1 - \Delta_2 C\|_F^2$$

Laplacian commutativity [OBCS*12]

$$+ w_2\left\|C\Omega_1^{multi} - \Omega_2^{multi}C\right\|_F^2$$

Multiplicative operators [NO17]

$$+ w_3\left\|C\Omega_1^{orient} - \Omega_2^{orient}C\right\|_F^2$$

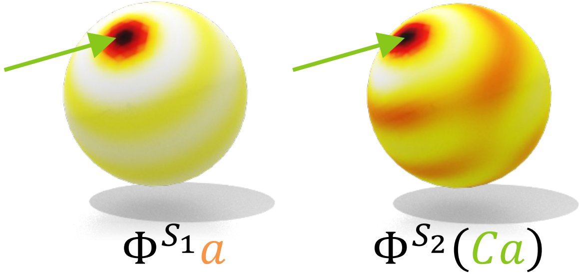
Orientation preservation [RPWO18]

$$+ \cdots$$

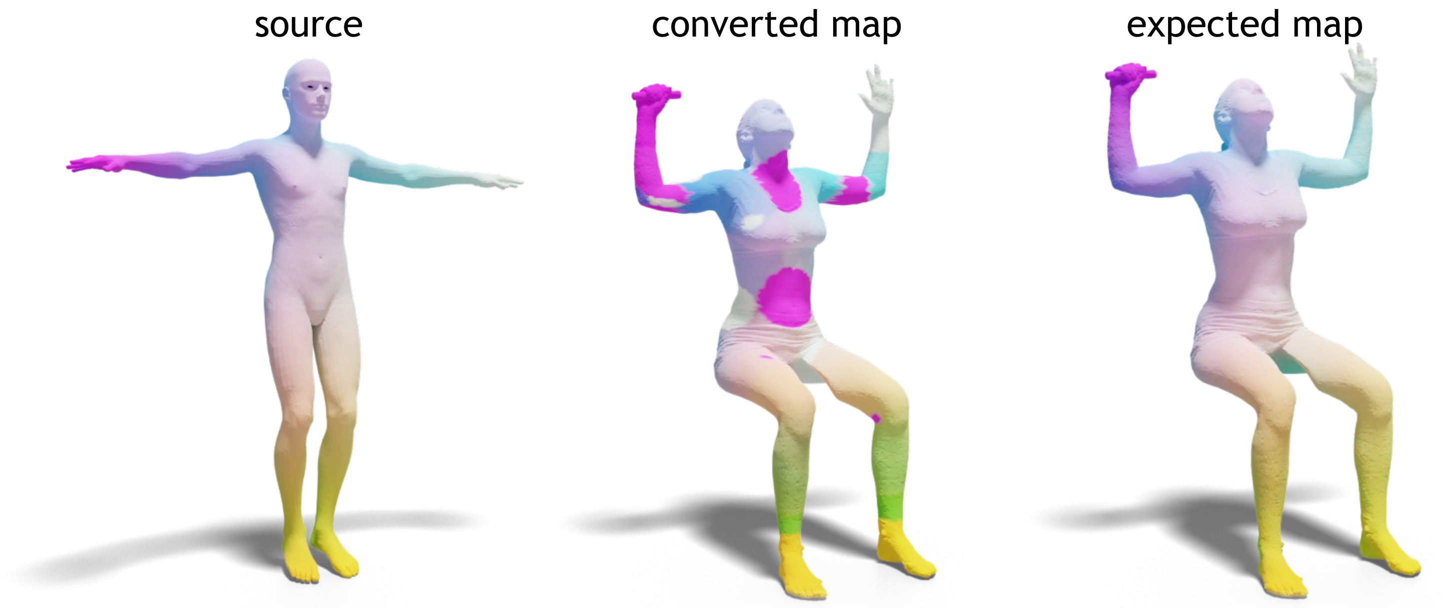
Use any quadratic solver to solve for C (the search space is $R^{k_1 \times k_2}$, continuous!)

Pointwise Map Conversion

- Given a good functional map C, how do we know where $v \in S_1$ is mapped to on S_2?
- Define a delta function $\delta(x) = 1$ if $x = v$, otherwise $\delta(x) = 0$
- $\delta(x)$ is a function defined on S_1 with spectral coefficients a
- Ca should gives the spectral coefficients on the target shape
- $g = \Phi^{S_2}(Ca)$ should be close to a delta function on S_2
- $\text{argmax}\, g$ gives the correspondence to v

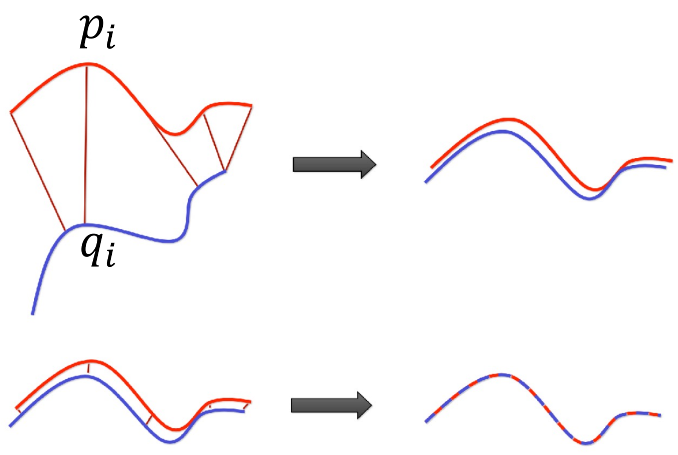


Map Refinement



- Functional map computed in the truncated spectral basis
- Converted map can be noisy → need postprocessing (i.e., map refinement)

ICP: Iterative Closest Point



Basic Algorithm:

1. Find corresponding points (p_i, q_i) via nearest neighbor searching
2. Find the best rigid alignment by minimizing:

$$E(R,t) = \sum_{i=1}^{n} \|(Rp_i + t) - q_i\|^2$$

3. Apply (R, t) to the source shape, go back to step 1

ICP: Iterative Closest Point

Rewrite ICP:

$$E(R, \Pi_{12}) = \sum_{p \in S_1} \left\| R X_1(p) - X_2\big(\Pi_{12}(p)\big) \right\|^2$$

1. Solve $\text{argmin}_{\Pi_{12}} \; E(\, \Pi_{12} \mid \; R \;\;)$
2. Solve $\text{argmin}_{\;R} \;\; E(\;\; R \;\; \mid \Pi_{12})$
3. go back to step 1

$X_i(p)$: homogeneous coordinate $(x, y, z, 1)^T$ of vertex p in shape S_i, Π_{12} is a pointwise map from S_1 to S_2

Basic Algorithm:

1. Find corresponding points (p_i, q_i) via nearest neighbor searching
2. Find the best rigid alignment by minimizing:

$$E(R, t) = \sum_{i=1}^{n} \|(Rp_i + t) - q_i\|^2$$

3. Apply (R, t) to the source shape, go back to step 1

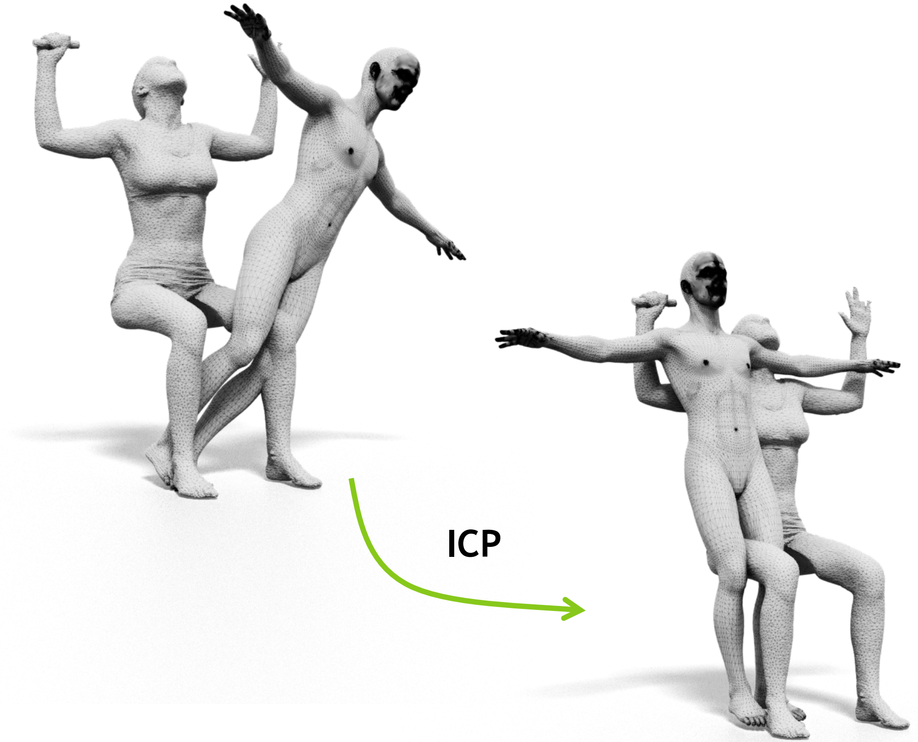
ICP: Iterative Closest Point

Rewrite ICP:

$$E(R, \Pi_{12}) = \sum_{p \in S_1} \left\| R X_1(p) - X_2\big(\Pi_{12}(p)\big) \right\|^2$$

1. Solve $\text{argmin}_{\Pi_{12}}\ E(\ \Pi_{12} \mid\ R\ \ )$
2. Solve $\text{argmin}_{R}\ \ E(\ \ R\ \ \mid \Pi_{12})$
3. go back to step 1

$X_i(p)$: homogeneous coordinate $(x, y, z, 1)^T$ of vertex p in shape S_i, Π_{12} is a pointwise map from S_1 to S_2



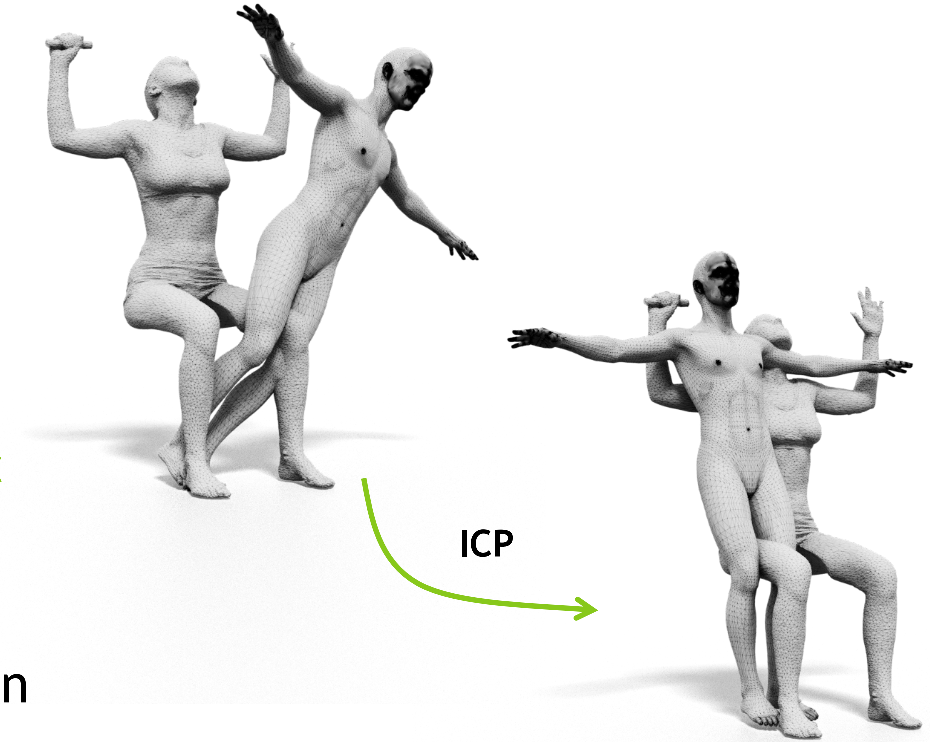
ICP: Iterative Closest Point

Why ICP fails in non-rigid matching:

$$E(R, \Pi_{12}) = \sum_{p \in S_1} \left\| R X_1(p) - X_2(\Pi_{12}(p)) \right\|^2$$

- nn-search in spatial domain to establish correspondences
- Vertex positions X_1, X_2 are extrinsic features

Solution: find intrinsic features to align



Spectral ICP

Spatial ICP:

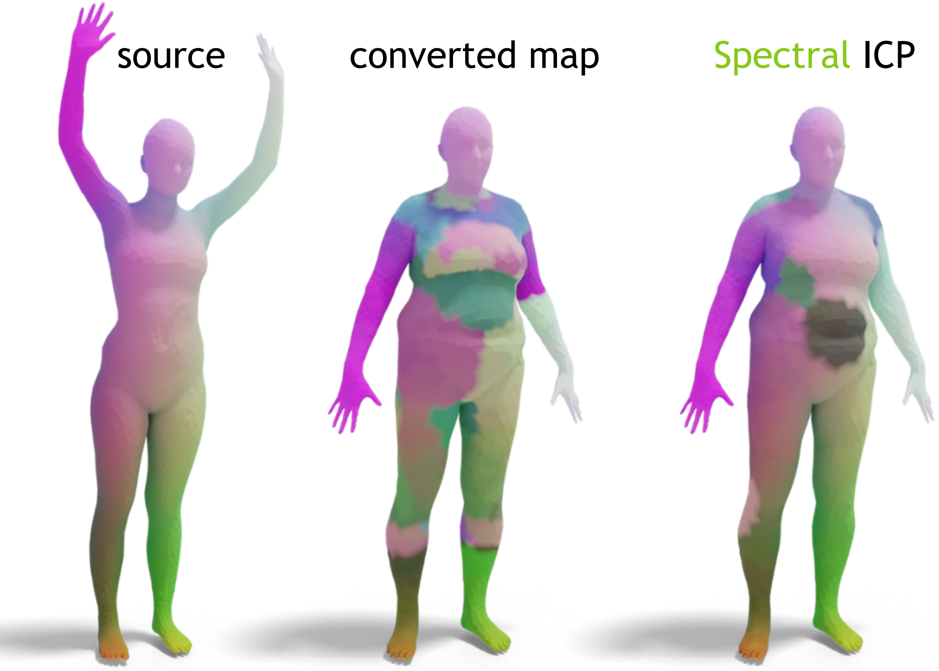
$$E(R, \Pi_{12}) = \sum_{p \in S_1} \left\| R X_1(p) - X_2\big(\Pi_{12}(p)\big) \right\|^2$$

Spectral ICP:

$$E(C, \Pi_{12}) = \sum_{p \in S_1} \left\| C \Phi_1(p) - \Phi_2\big(\Pi_{12}(p)\big) \right\|^2$$

- nn-search in spectral domain to establish correspondences
- Laplace-Beltrami EigenBasis Φ_1, Φ_2 are intrinsic features
- C: high-dimensional rotation to align the spectral domain - functional map!

Spectral ICP



Spectral ICP:

$$E(C, \Pi_{12}) = \sum_{p \in S_1} \left\| C\Phi_1(p) - \Phi_2\left(\Pi_{12}(p)\right) \right\|^2$$

- Φ_i usually has a size of $50 \sim 500$
- C then has a size of $50^2 \sim 500^2$
- linear system can be under-determined
- Can easily get trapped into local minima

ZoomOut: progressive upsampling

Spectral ICP:

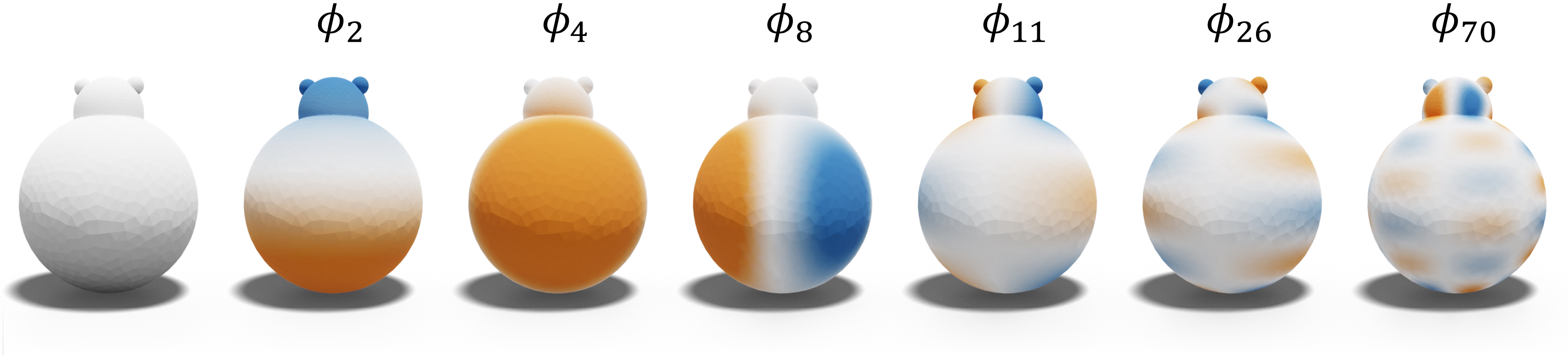
$$E(C, \Pi_{12}) = \sum_{p \in S_1} \left\| C\Phi_1(p) - \Phi_2\big(\Pi_{12}(p)\big) \right\|^2$$

- The functional map C tries to align the high-dimensional feature (spectral) space of the two shapes
- Align in a progressive manner

ZoomOut: progressive upsampling

Global structure → Local structure

ZoomOut: progressive upsampling



Low frequency → High frequency

ZoomOut: progressive upsampling

ZoomOut algorithm for minimizing Spectral ICP:

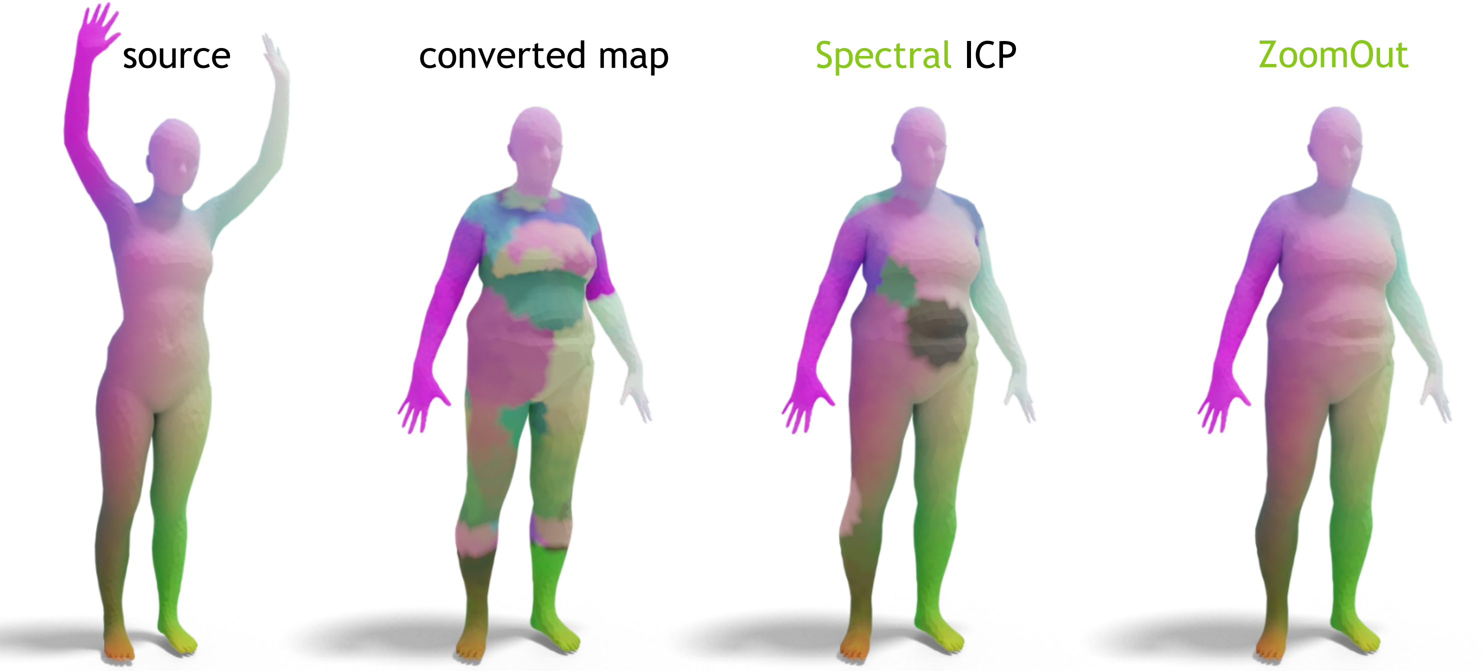
$$E(C, \Pi_{12}) = \sum_{p \in S_1} \left\| C\Phi_1(p) - \Phi_2(\Pi_{12}(p)) \right\|^2$$

1. Initialize Π_{12} and k, where $\Phi_i^{(k)}$ stores the first k Eigen-basis
2. Solve for $C^{(k)} = \underset{C}{\operatorname{argmin}} \sum_{p \in S_1} \left\| C\Phi_1^{(k)}(p) - \Phi_2^{(k)}(\Pi_{12}(p)) \right\|^2$
3. Solve for $\Pi_{12} = \underset{\Pi_{12}}{\operatorname{argmin}} \sum_{p \in S_1} \left\| C^{(k)}\Phi_1^{(k)}(p) - \Phi_2^{(k)}(\Pi_{12}(p)) \right\|^2$
4. $k \leftarrow k + 1$, go to step 2

ZoomOut: progressive upsampling

$C^{(k)}$ with increasing k

Spectral ICP v.s. ZoomOut



Summary

- Two parameterization-based methods which reduce the search space
  - "MAPS": simplify mesh while maintain the bijective map
  - "Mobius Voting": find 3 correspondences to compute the Mobius transformation
- Functional map pipeline which solves the matching problem in a continuous search space
  - Spectral ICP
  - ZoomOut

Thank You

INTERACTIVE GEOMETRY LAB

April 15, 2025

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich