

Geometric Registration for Deformable Shapes

1.1 Introduction

Overview • Data Sources and Applications • Problem Statement

31st Annual Conference of the
European Association for Computer Graphics

europa
graphics 2010

Overview

Presenters



Will Chang

University of California at
San Diego, USA

wychang@cs.ucsd.edu



Hao Li

ETH Zürich, EPFL Lausanne
Switzerland

hao@inf.ethz.ch



Niloy Mitra

KAUST, Saudi Arabia
IIT Delhi, India

niloy@cse.iitd.ernet.in



Mark Pauly

EPFL Lausanne
Switzerland

mark.pauly@epfl.ch



Michael Wand

Saarland University,
MPI Informatik, Germany

mwand@mpi-inf.mpg.de

Tutorial Outline

Overview

- **Part I:** Introduction (1h)
- **Part II:** Local Registration (1.5h)
- **Part III:** Global Matching (2h)
- **Part IV:** Animation Reconstruction (1.25h)
- Conclusions and Wrap up (0.25h)

Part I: Introduction

Introduction (Michael)

- Problem statement and motivation
- Example data sets and applications

Differential geometry and deformation modeling (Mark)

- Differential geometry Background
- Brief introduction to deformation modeling

Kinematic 4D surfaces (Niloy)

- Rigid motion in space-time
- Kinematic 4D surfaces

Part II: Local Registration

ICP and of rigid motions (Niloy)

- Rigid ICP, geometric optimization perspective
- Dynamic geometry registration (Intro)

Deformable Registration (Michael)

- A variational model for deformable shape matching
- Variants of deformable ICP

Subspace Deformation, Robust Registration (Hao)

- Subspace deformations / deformation graphs
- Robust local matching

Part III: Global Matching

Features (Will)

- Key point detection and feature descriptors

Isometric Matching and Quadratic Assignment (Michael)

- Extrinsic vs. intrinsic geometry
- Global matching techniques with example algorithms

Advanced Global Matching (Will)

- Global registration algorithms

Probabilistic Techniques (Michael)

- Ransac and forward search

Articulated Registration (Will)

- Articulated registration with graph cuts

Part IV: Animation Reconstruction

Dynamic Geometry Registration (Niloy)

- Multi-piece alignment

Deformable Reconstruction (Michael)

- Basic numerical algorithm
- Urshape/Deformation Factorization

Improved Algorithm (Hao)

- Efficient implementation
- Detail transfer

Part V: Conclusions and Wrap-up

Conclusions and Wrap-up (Mark)

- Conclusions
- Future work and open problems

After every part:

- Q&A session with all speakers
- Feel free to ask questions at any time

Problem Statement and Motivation

Deformable Shape Matching

What is the problem?

Settings:

- We have two or more shapes
- The same object, but deformed



Data courtesy of C. Stoll, MPI Informatik

Deformable Shape Matching

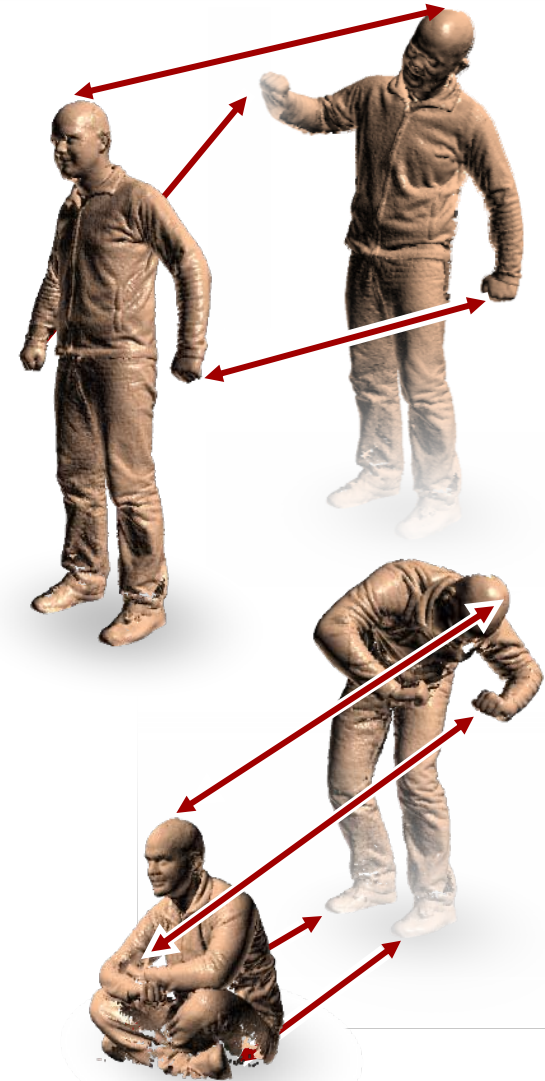
What is the problem?

Settings:

- We have two or more shapes
- The same object, but deformed

Question:

- What points correspond?



Data courtesy of C. Stoll, MPI Informatik

Applications

Why is this an interesting problem?

Building Block:

- Correspondences are a building block for higher level geometry processing algorithms

Example Applications:

- Scanner data registration
- Animation reconstruction & 3D video
- Statistical shape analysis (shape spaces)

Applications

Why is this an interesting problem?

Building Block:

- Correspondences are a building block for higher level geometry processing algorithms

Example Applications:

- Scanner data registration
- Animation reconstruction & 3D video
- Statistical shape analysis (shape spaces)

Deformable Scan Registration

Scan registration

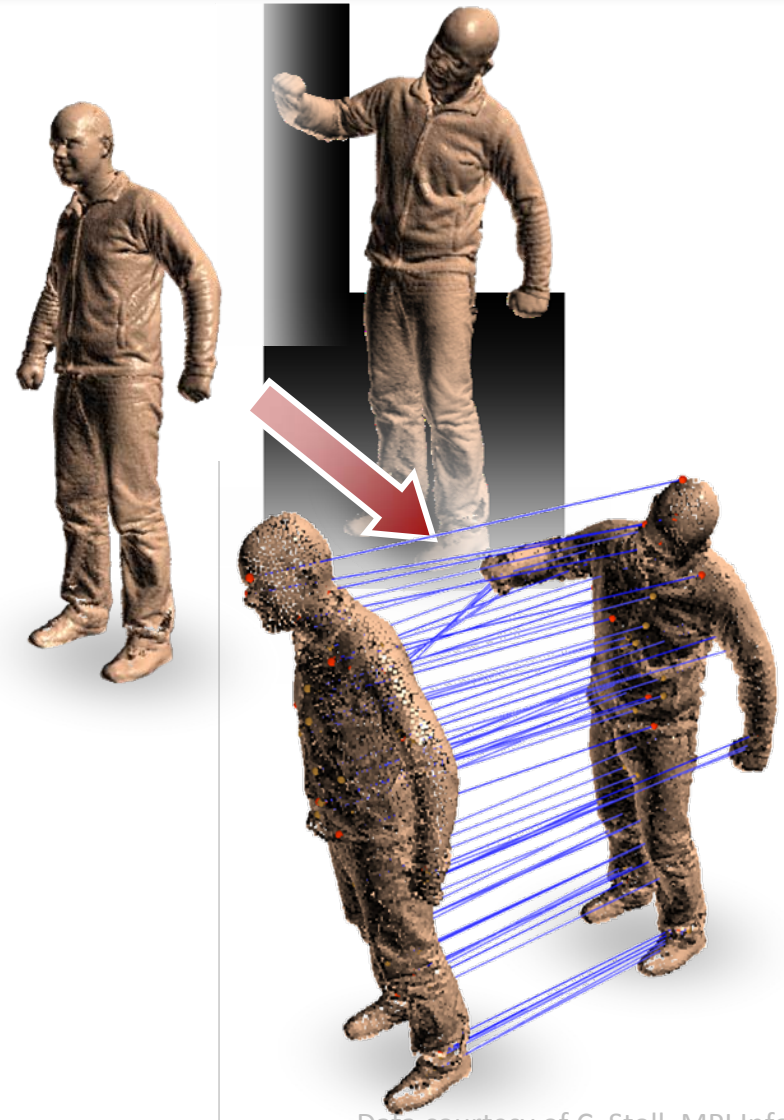
- Rigid registration is standard

Why deformation?

- Scanner miscalibrations
 - Sometimes unavoidable, esp. for large acquisition volumes
- Scanned Object might be deformable
 - Elastic / plastic objects
- In particular: Scanning people, animals
 - Need multiple scans
 - Impossible to maintain constant pose

Example: Full Body Scanner

Full Body Scanning



Data courtesy of C. Stoll, MPI Informatik

Applications

Why is this an interesting problem?

Building Block:

- Correspondences are a building block for higher level geometry processing algorithms

Example Applications:

- Scanner data registration
- Animation reconstruction & 3D video
- Statistical shape analysis (shape spaces)

3D Animation Scanner

New technology

- 3D animation scanners
- Record 3D video
- Active research area

Ultimate goal

- 3D movie making
- New creative perspectives



Photo: P. Jenke, WSI/GRIS Tübingen

Structured Light Scanners



**space-time
stereo**

courtesy of James Davis,
UC Santa Cruz



**color-coded
structured light**

courtesy of Phil Fong,
Stanford University



**motion compensated
structured light**

courtesy of Sören König,
TU Dresden

Passive Multi-Camera Acquisition



**segmentation &
belief propagation**

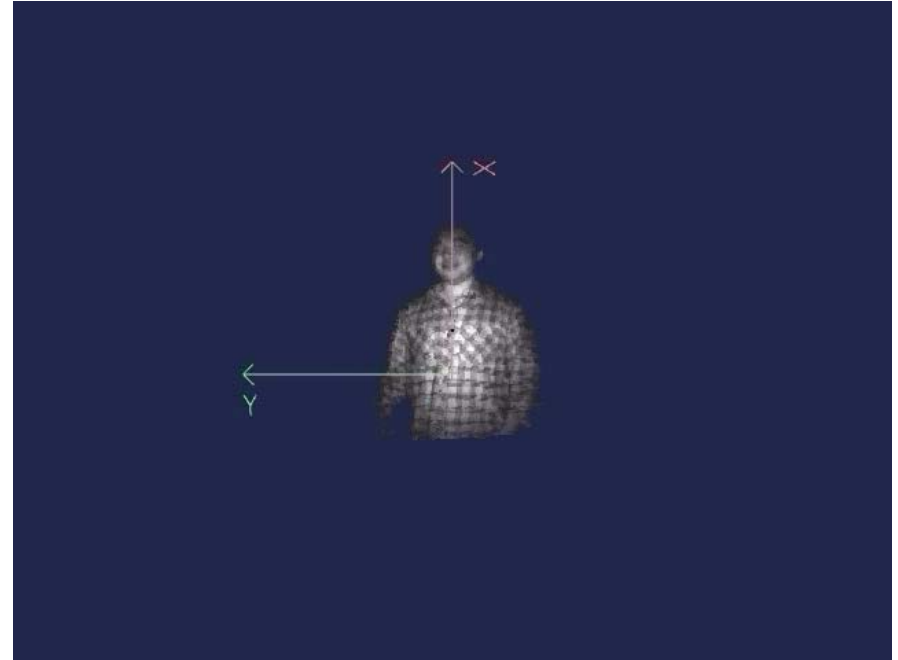
[Zitnick et al. 2004]
Microsoft Research



**photo-consistent
space carving**

Christian Theobald
MPI-Informatik

Time-of-Flight / PMD Devices

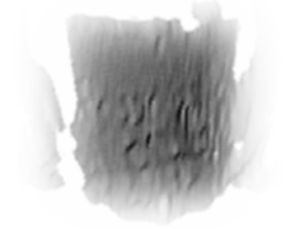


Swiss Ranger Time-of-flight camera

Animation Reconstruction

Problems

- Noisy data
- Incomplete data (acquisition holes)
- No correspondences



noise



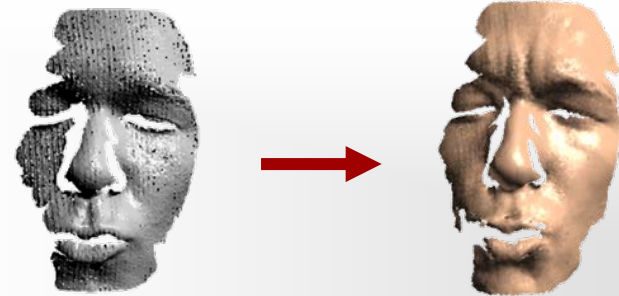
holes



missing correspondences

Animation Reconstruction

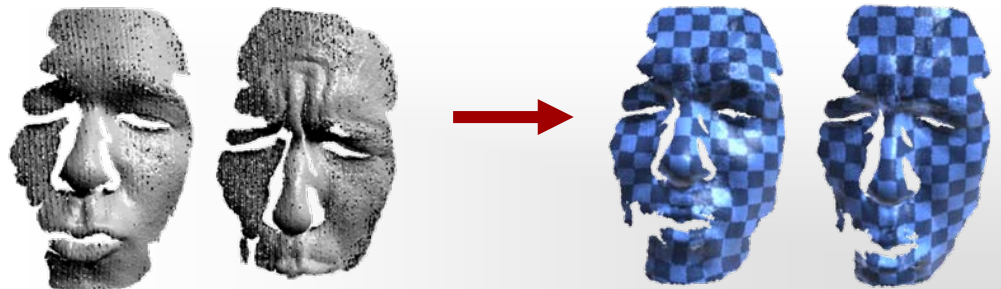
Remove noise, outliers



Fill-in holes
(from all frames)



Dense correspondences



Applications

Why is this an interesting problem?

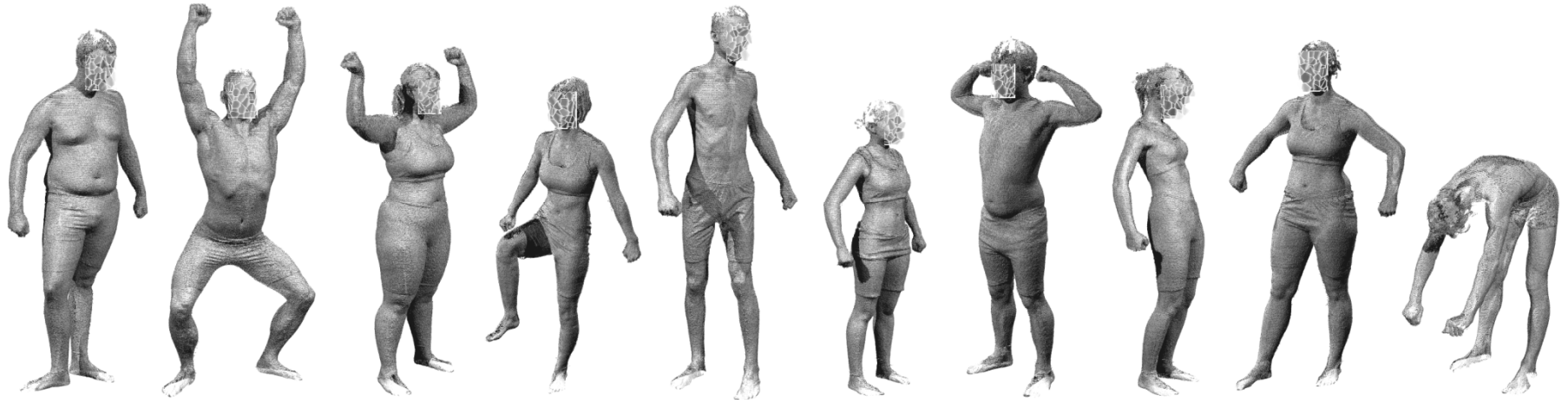
Building Block:

- Correspondences are a building block for higher level geometry processing algorithms

Example Applications:

- Scanner data registration
- Animation reconstruction & 3D video
- Statistical shape analysis (shape spaces)

Statistical Shape Spaces



Courtesy of N. Hassler, MPI Informatik

Morphable Shape Models

- Scan a large number of individuals
 - Different pose
 - Different people
- Compute correspondences
- Build shape statistics (PCA, non-linear embedding)

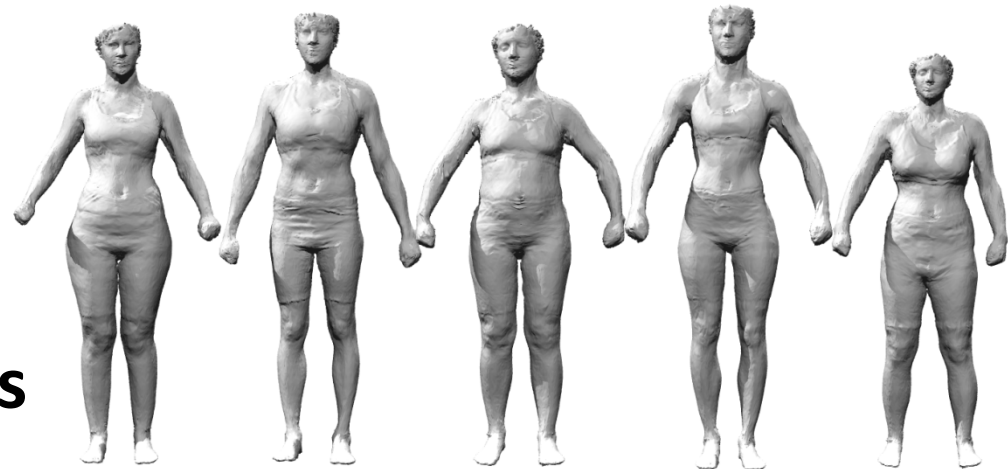
Statistical Shape Spaces

Numerous Applications:

- Fitting to ambiguous data (prior knowledge)
- Constraint-based editing
- Recognition, classification, regression



Courtesy of N. Hassler, MPI Informatik



Courtesy of N. Hassler, MPI Informatik

**Building such models
requires correspondences**

Data Characteristics

Scanner Data – Challenges

“Real world data” is more challenging

- 3D Scanners have artifacts

Rules of thumb:

- The faster the worse (real time vs. static scans)
- Active techniques are more accurate
(passive stereo is more difficult than laser triangulation)
- There is more than just “Gaussian noise” ...

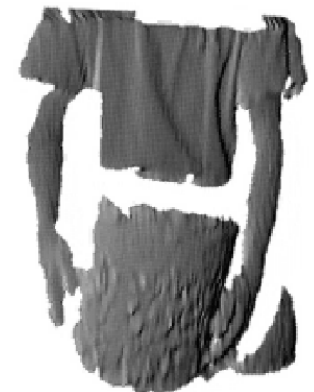
Challenges

“Noise”

- “Standard” noise types:
 - Gaussian noise (analog signal processing)
 - Quantization noise
- More problematic: Structured noise
 - Structured noise (spatio-temporally correlated)
 - Structured outliers
 - Reflective / transparent surfaces
- Incomplete Acquisition
 - Missing parts
 - Topological noise



Courtesy of J. Davis, UCSC

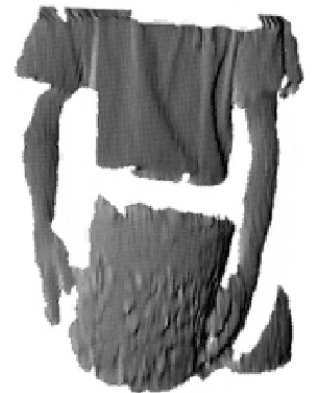
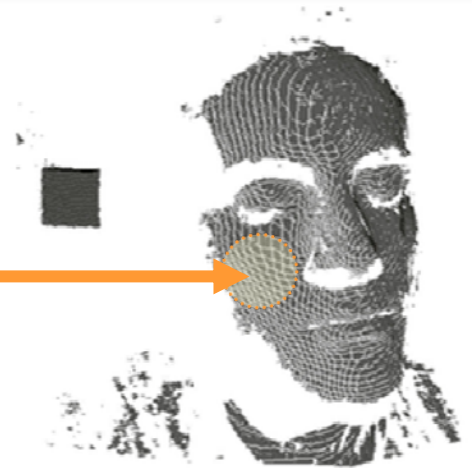


Courtesy of P. Phong, Stanford University

Challenges

“Noise”

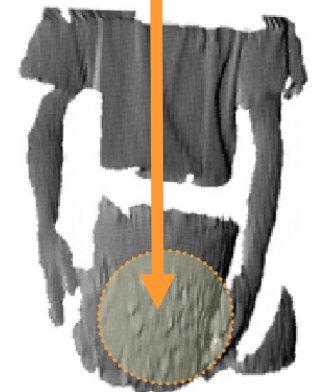
- “Standard” noise types:
 - Gaussian noise (analog signal processing)
 - Quantization noise
- More problematic: Structured noise
 - Structured noise (spatio-temporally correlated)
 - Structured outliers
 - Reflective / transparent surfaces
- Incomplete Acquisition
 - Missing parts
 - Topological noise



Challenges

“Noise”

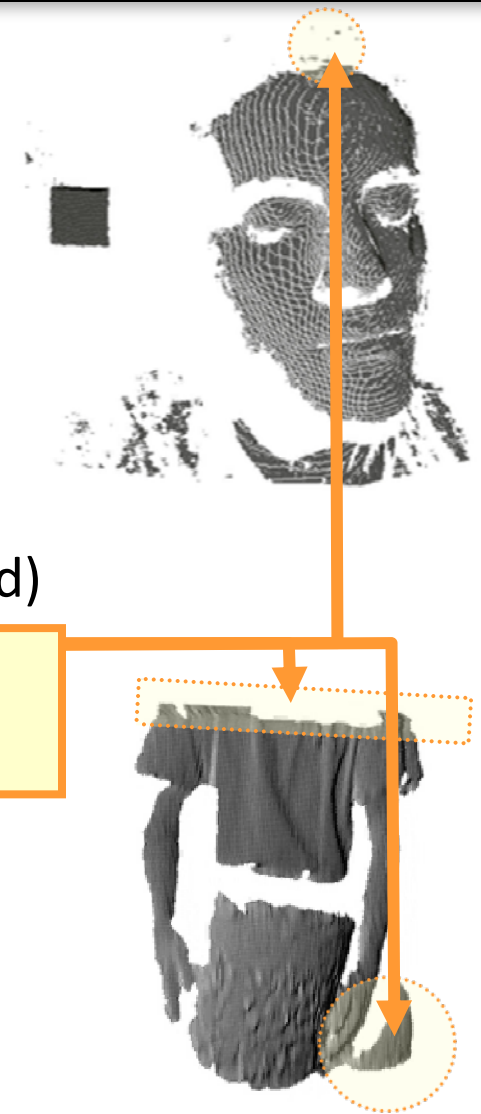
- “Standard” noise types:
 - Gaussian noise (analog signal processing)
 - Quantization noise
- More problematic
 - Structured noise (spatio-temporally correlated)
 - Structured outliers
 - Reflective / transparent surfaces
- Incomplete Acquisition
 - Missing parts
 - Topological noise



Challenges

“Noise”

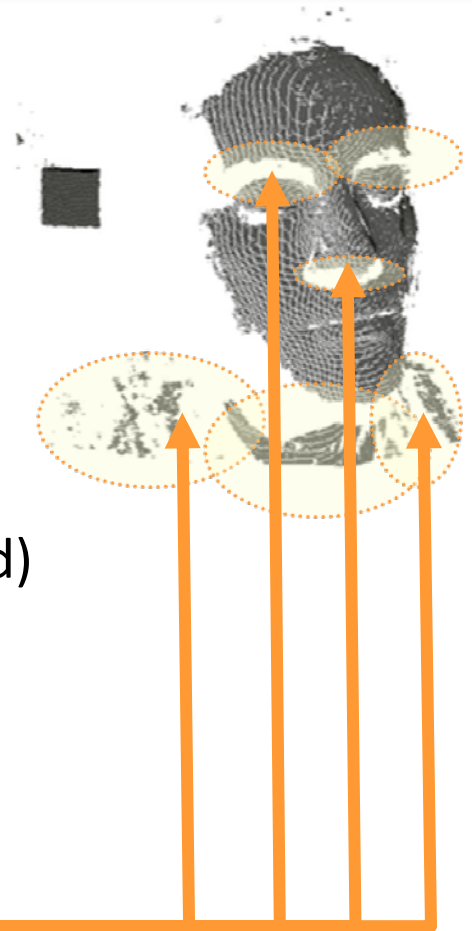
- “Standard” noise types:
 - Gaussian noise (analog signal processing)
 - Quantization noise
- More problematic
 - Structured noise (spatio-temporally correlated)
 - Structured outliers
 - Reflective / transparent surfaces
- Incomplete Acquisition
 - Missing parts
 - Topological noise



Challenges

“Noise”

- “Standard” noise types:
 - Gaussian noise (analog signal processing)
 - Quantization noise
- More problematic
 - Structured noise (spatio-temporally correlated)
 - Structured outliers
 - Reflective / transparent surfaces
- Incomplete Acquisition
 - Missing parts
 - Topological noise



Outlook

This Tutorial

Different aspects of the problem:

- Shape deformation and matching
 - How to *quantify deformation*?
 - How to *define deformable shape matching*?
- Local matching
 - Known initialization
- Global matching
 - No initialization
- Animation Reconstruction
 - Matching temporal sequences of scans

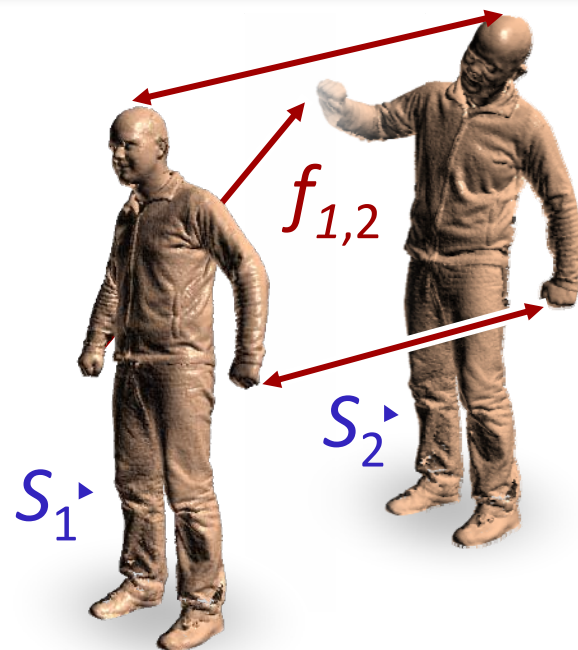
Problem Statement

Given:

- Two surfaces $S_1, S_2 \subseteq \mathbb{R}^3$
- Discretization:
 - Point clouds $S = \{s_1, \dots, s_n\}, s_i \in \mathbb{R}^3$ or
 - Triangle meshes

We are looking for:

- A deformation function $f_{1,2}: S_1 \rightarrow \mathbb{R}^3$ that brings S_1 close to S_2



Problem Statement

We are looking for:

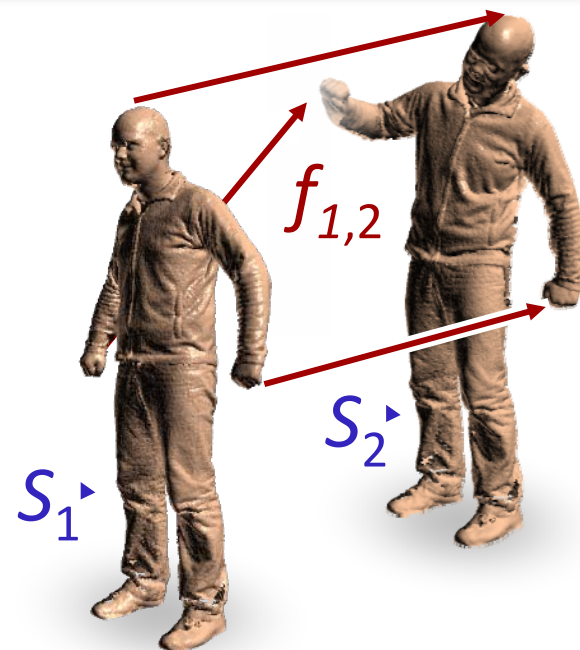
- A deformation function $f_{1,2}: S_1 \rightarrow \mathbb{R}^3$ that brings S_1 close to S_2

Open Questions:

- What does “close” mean?
- What properties should f have?

Next part:

- We will now look at these questions more in detail



Differential Geometry

of Curves & Surfaces (Overview)

Part I: Curves

Parametric Curves

Parametric Curves:

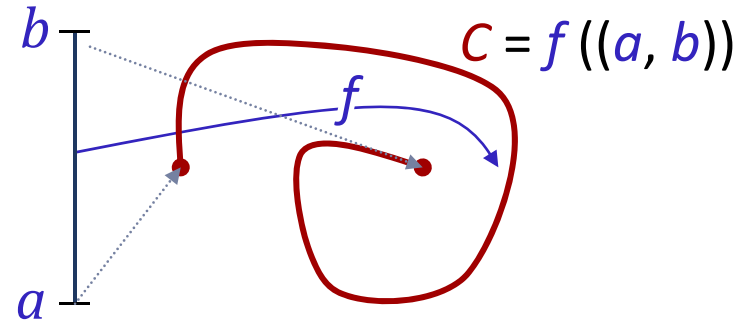
- A differentiable function

$$f: (a, b) \rightarrow \mathbb{R}^n$$

describes a *parametric curve*

$$C = f((a, b)), C \subseteq \mathbb{R}^n.$$

- The parametrization is called *regular* if $f'(t) \neq 0$ for all t .
- If $\|f'(t)\| \equiv 1$ for all t , f is called a *unit-speed parametrization* of the curve C .



Length of a Curve

The length of a curve:

- The length of a regular curve C is defined as:

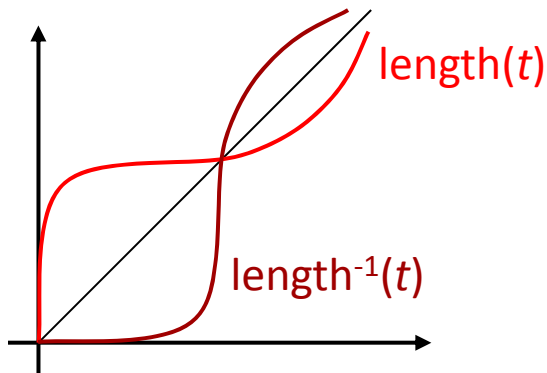
$$\text{length}(C) = \int_a^b \|f'(t)\| dt$$

- This definition is independent of the parametrization (integral transformation theorem).
- Alternatively, the length of the curve can be defined as $\text{length}(C) = |b - a|$ for a unit-speed parametrization $C = f((a, b))$; this obviously yields the same result.

Reparametrization

Enforcing unit-speed parametrization:

- Assume: $\|f'(t)\| \neq 0$ for all t .
- We have:
$$\text{length}(C) = \int_a^b \|f'(t)\| dt \quad (\text{invertible, because } f'(t) > 0)$$
- Concatenating $f \circ \text{length}^{-1}(C)$ yields a unit-speed parametrization of the curve



Tangents

Unit Tangents:

- The unit tangent vector at $x \in (a, b)$ is given by:

$$\text{tangent}(t) = \frac{f'(t)}{\|f'(t)\|}$$

- For curves $C \subseteq \mathbb{R}^2$, the unit normal vector of the curve is defined as:

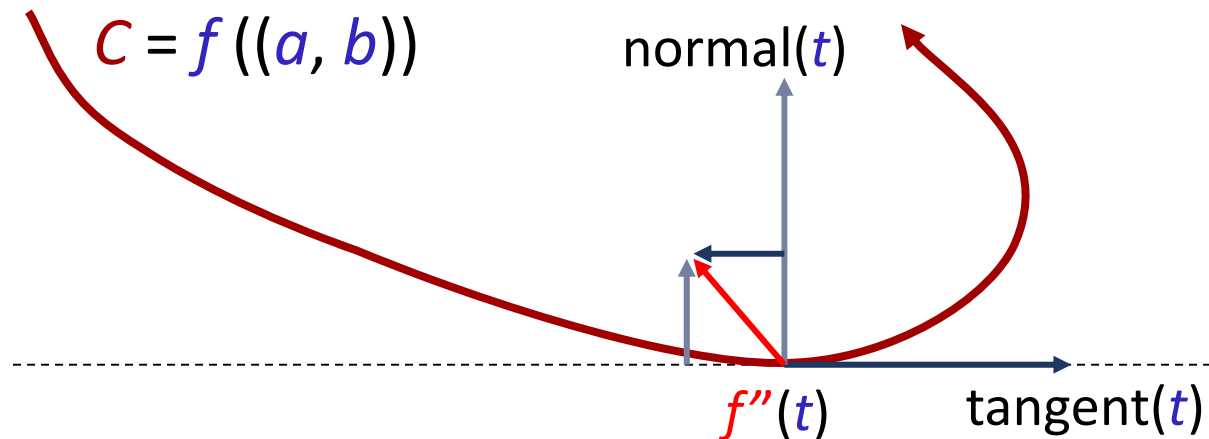
$$\text{normal}(t) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \frac{f'(t)}{\|f'(t)\|}$$

Curvature

Curvature:

- First derivatives show curve direction / speed of movement.
- Curvature is encoded in 2nd order information.
- Why not just use f'' ?
- Two problems:
 - Depends on parametrization (different velocity yields different results)
 - Have to distinguish between acceleration in tangential and non-tangential directions.

Curvature & 2nd Derivatives



Definition of curvature

- We want only the non-tangential component of f'' .
- Braking / accelerating does not matter for curvature of the traced out curve C .
- Need to normalize speed.

Curvature

Curvature of a Curve $\mathbf{C} \in \mathbb{R}^2$:

$$\kappa^2(t) = \frac{\left\langle f''(t), \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} f'(t) \right\rangle}{\|f'(t)\|^3}$$

- Normalization factor:

- Divide by $\|f'\|$ to obtain unit tangent vector

- Divide again twice to normalize f''

- Taylor expansion / chain rule:

$$f(\lambda t) = f(t_0) + \lambda f'(t_0)(t - t_0) + \frac{1}{2} \lambda^2 f''(t)(t - t_0)^2 + O(t^3)$$

- Second derivative scales quadratically with speed

Unit-speed parametrization

Unit-speed parametrization:

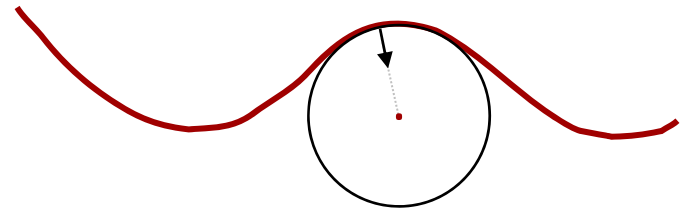
- Assume a unit-speed parametrization, i.e. $\|f'\| \equiv 1$.
- Then, κ^2 simplifies to:

$$\kappa^2(t) = \|f''(t)\|$$

Radius of Curvature

Easy to see:

- Curvature of a circle is constant, $\kappa^2 \equiv \pm 1/r$ (r = radius).
(see problem sets)
- Accordingly: Define radius of curvature as $1/\kappa^2$.
- Osculating circle:
 - Radius: $1/\kappa^2$
 - Center: $f(t) + \frac{1}{\kappa^2} \mathit{normal}(t)$



Theorems

Definition:

- Rigid motion: $\mathbf{x} \rightarrow \mathbf{Ax} + \mathbf{b}$ with orthogonal \mathbf{A}
 - Orientation preserving (no mirroring) if $\det(\mathbf{A}) = +1$
 - Mirroring leads to $\det(\mathbf{A}) = -1$

Theorems for plane curves:

- Curvature is invariant under rigid motion
 - Absolute value is invariant
 - Signed value is invariant for orientation preserving rigid motion
- Two unit speed parameterized curves with identical signed curvature function differ only in a orientation preserving rigid motion.

Space Curves

General case: Curvature of a Curve $C \subseteq \mathbb{R}^n$

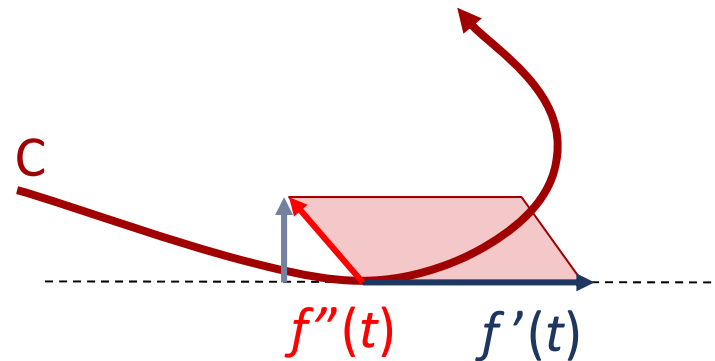
- *W.l.o.g.*: Assume we are given a unit-speed parametrization f of C
- The *curvature* of C at parameter value t is defined as:

$$\kappa(t) = \|f''(t)\|$$

- For a general, regular curve $C \subseteq \mathbb{S}^3$ (any regular parametrization):

$$\kappa(t) = \frac{\|f'(t) \times f''(t)\|}{\|f'(t)\|^3}$$

- General curvature is unsigned



Torsion

Characteristics of Space Curves in \mathbb{R}^3 :

- Curvature not sufficient
- Curve may “bend” in space
- Curvature is a 2nd order property
- 2nd order curves are always flat
 - Quadratic curves are specified by 3 points in space, which always lie in a plane
 - Cannot capture out-of-plane bends
- Missing property: Torsion

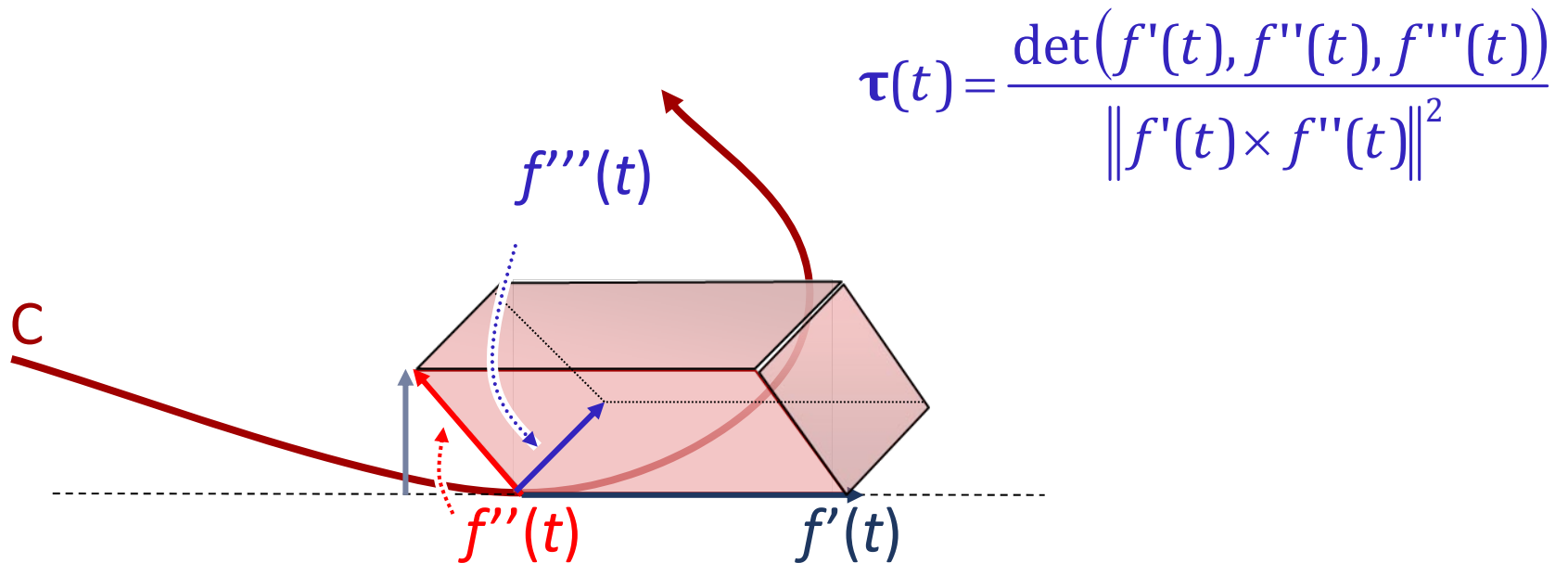
Torsion

Definition:

- Let f be a regular parametrization of a curve $C \subseteq \mathbb{R}^3$ with non-zero curvature
- The torsion of f at t is defined as

$$\tau(t) = \frac{f'(t) \times f''(t) \cdot f'''(t)}{\|f'(t) \times f''(t)\|^2} = \frac{\det(f'(t), f''(t), f'''(t))}{\|f'(t) \times f''(t)\|^2}$$

Illustration



Theorem

Fundamental Theorem of Space Curves

- Two unit speed parameterized curves $C \subseteq \mathbb{R}^3$ with identical, positive curvature and identical torsion are identical up to a rigid motion.

Part II: Surfaces

Parametric Patches

Parametric Surface Patches:

A smoothly differentiable function

$$f: \mathbb{R}^2 \supseteq \Omega \rightarrow \mathbb{R}^n$$

describes a *parametric surface patch*

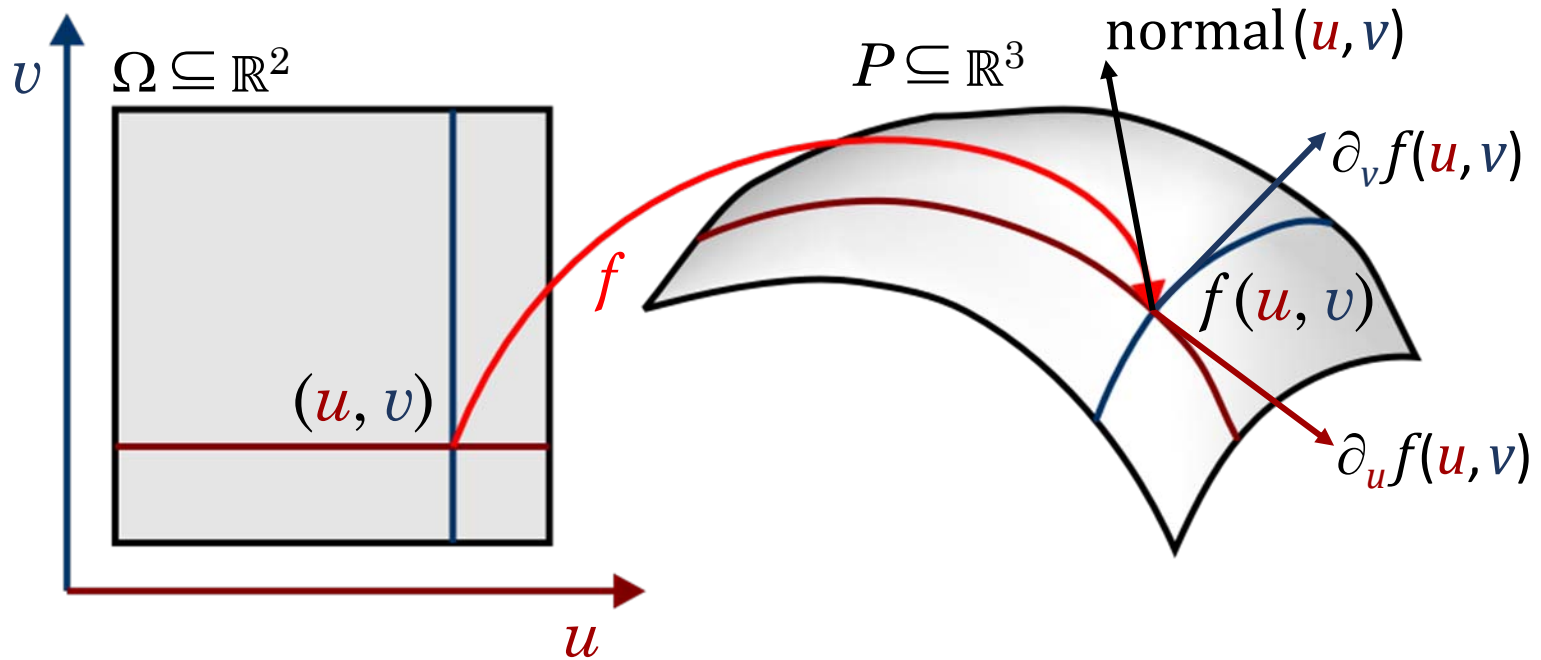
$$P = f(\Omega), P \subseteq \mathbb{R}^n.$$

Parametric Patches

Parametric Surface Patches:

- The vectors $\text{tangent}_{\mathbf{x}_0}(r) = \frac{d}{dt} f(\mathbf{x}_0 + t\mathbf{r}) = \nabla_{\mathbf{r}} f(\mathbf{x}_0)$ are *tangent vectors* of the surface. In particular, there are canonical tangents $\partial_u f(u, v)$, $\partial_v f(u, v)$ in principal parameter directions.
- *Regular parametrization*: $\partial_u f$, $\partial_v f$ linearly independent.
- For a regularly parametrized patch in \mathbb{R}^3 , the unit normal vector is given by:
$$\text{normal}(u, v) = \frac{\partial_u f(u, v) \times \partial_v f(u, v)}{\|\partial_u f(u, v) \times \partial_v f(u, v)\|}$$

Illustration



Tangents

Computing Tangents:

- General tangents can be computed from principal tangents:

$$\text{tangent}_{\mathbf{x}_0}(\mathbf{r}) = \nabla f(\mathbf{x}_0)\mathbf{r} = \begin{pmatrix} | & | \\ \partial_u f(\mathbf{x}_0) & \partial_v f(\mathbf{x}_0) \\ | & | \end{pmatrix} \begin{pmatrix} r_u \\ r_v \end{pmatrix}$$

Surface Area

Surface Area:

- Computation is simple
- For a patch $f: \mathbb{R}^2 \supseteq \Omega \rightarrow \mathbb{R}^n$, integrate over a constant function (one everywhere) over the surface area:
- Then just apply integral transformation theorem:

$$\text{area}(P) = \int_{\Omega} \|\partial_u f(\mathbf{x}) \times \partial_v f(\mathbf{x})\| d\mathbf{x}, \quad x = \begin{pmatrix} u \\ v \end{pmatrix}$$

Fundamental Forms

Fundamental Forms:

- Describe the local parametrized surface
- Measure...
 - ...distortion of length (first fundamental form)
 - ...surface curvature (second fundamental form)
- Parametrization independent surface curvature measures will be derived from this

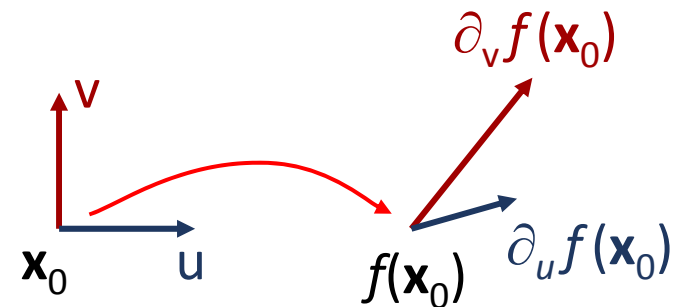
First Fundamental Form

First Fundamental Form

- Also known as *metric tensor*.
- Given a regular parametric patch $f: \mathbb{R}^2 \supseteq \Omega \rightarrow \mathbb{R}^3$.
- f will distort angles and distances
- We will look at a local first order Taylor approximation to measure the effect:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

- Length changes become visible in the scalar product...



First Fundamental Form

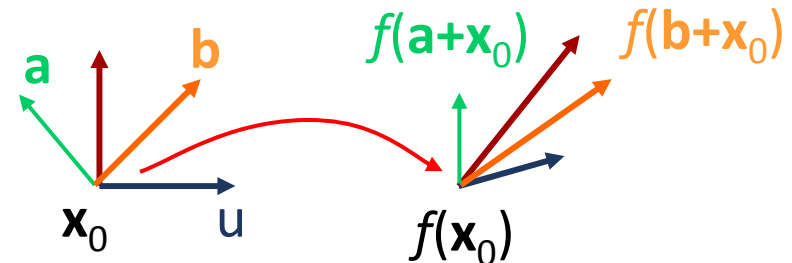
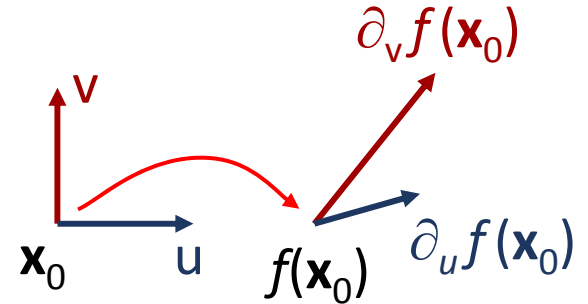
First Fundamental Form

- First order Taylor approximation:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

- Scalar product of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$:

$$\begin{aligned} \langle f(\mathbf{x} + \mathbf{a}) - f(\mathbf{x}), f(\mathbf{x} + \mathbf{b}) - f(\mathbf{x}) \rangle &\approx \langle \nabla f(\mathbf{x}_0)\mathbf{a}, \nabla f(\mathbf{x}_0)\mathbf{b} \rangle \\ &= \mathbf{a}^T \underbrace{(\nabla f(\mathbf{x}_0)^T \nabla f(\mathbf{x}_0))}_{\text{first fundamental form}} \mathbf{b} \end{aligned}$$



First Fundamental Form

First Fundamental Form

- The first fundamental form can be written as a 2×2 matrix:

$$\left(\nabla f^T \nabla f\right) = \begin{pmatrix} \partial_u f \partial_u f & \partial_u f \partial_v f \\ \partial_u f \partial_v f & \partial_v f \partial_v f \end{pmatrix} =: \begin{pmatrix} E & F \\ F & G \end{pmatrix} \quad \mathbf{I}(\mathbf{x}, \mathbf{y}) := \mathbf{x}^T \left(\nabla f^T \nabla f\right) \mathbf{y}$$

- The matrix is symmetric and positive definite (for a regular parametrization)
- Defines a *generalized scalar product* that measures lengths and angles *on the surface*.

Second Fundamental Form

Problems:

- The first fundamental form measures length changes only
- A cylinder looks like a flat sheet in this view
- We need a tool to measure curvature of a surface as well
- Again, we will need second order information
(any first order approximation is inherently flat)

Second Fundamental Form

Definition:

- Given a regular parametric patch $f: \mathbb{R}^2 \supseteq \Omega \rightarrow \mathbb{R}^3$.
- The *second fundamental form* (also known as *shape operator*, or *curvature tensor*) is the matrix:

$$S(\mathbf{x}_0) = \begin{pmatrix} \partial_{uu} f(\mathbf{x}_0) \cdot \mathbf{n} & \partial_{uv} f(\mathbf{x}_0) \cdot \mathbf{n} \\ \partial_{uv} f(\mathbf{x}_0) \cdot \mathbf{n} & \partial_{vv} f(\mathbf{x}_0) \cdot \mathbf{n} \end{pmatrix}$$

- Notation:

$$\mathbf{II}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \begin{pmatrix} \partial_{uu} f(\mathbf{x}_0) \cdot \mathbf{n} & \partial_{uv} f(\mathbf{x}_0) \cdot \mathbf{n} \\ \partial_{uv} f(\mathbf{x}_0) \cdot \mathbf{n} & \partial_{vv} f(\mathbf{x}_0) \cdot \mathbf{n} \end{pmatrix} \mathbf{y}$$

Second Fundamental Form

Basic Idea:

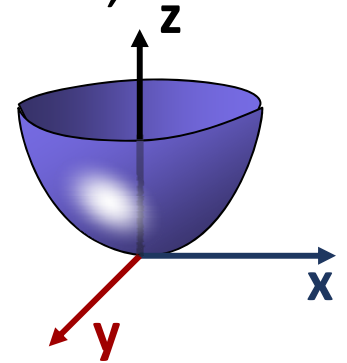
- Compute second derivative vectors
- Project in normal direction (remove tangential acceleration)

Alternative Computation

Alternative Formulation (Gauss):

- Local height field parameterization $f(\mathbf{x}, \mathbf{y}) = z$
- Orthonormal \mathbf{x}, \mathbf{y} coordinates *tangential* to surface, \mathbf{z} in normal direction, origin at zero
- 2nd order Taylor representation:

$$f(\mathbf{x}) \approx \frac{1}{2} \underbrace{f''(\mathbf{x})\mathbf{x}^2}_{=ex^2 + 2fxy + gy^2} + \underbrace{f'(\mathbf{x})\mathbf{x} + f(0)}_0$$



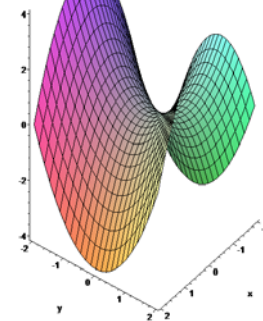
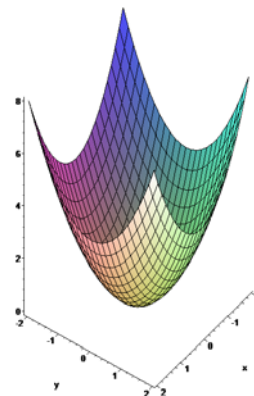
- Second fundamental form: Matrix of second derivatives

$$\begin{pmatrix} \partial_{xx} f & \partial_{xy} f \\ \partial_{xy} f & \partial_{yy} f \end{pmatrix} =: \begin{pmatrix} e & f \\ f & g \end{pmatrix}$$

Basic Idea

In other words:

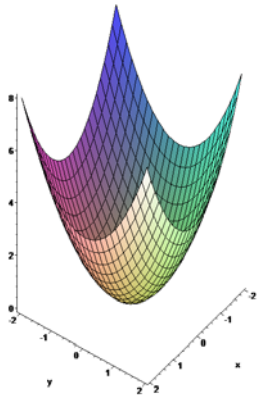
- The first fundamental form is the linear part (squared) of local Taylor approximation.
- The second fundamental form is the quadratic part of a local quadratic approximation of the heightfield
- The matrix is symmetric. So next thing to try is eigenanalysis, of course...



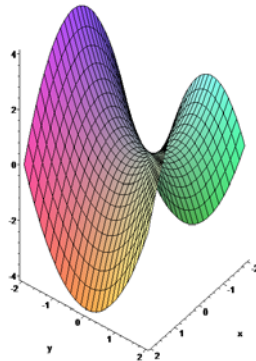
Principal Curvature

Eigenanalysis:

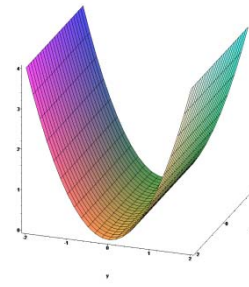
- The eigenvalues of the shape operator for an orthonormal tangent basis are called *principal curvatures* κ_1, κ_2 .
- The corresponding eigenvectors (which are orthogonal) are called *principal directions of curvature*.
- Again, we get different cases...:



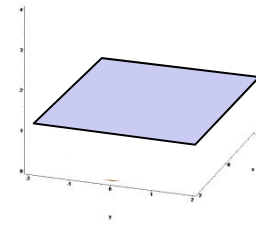
$$\kappa_i > 0$$



$$\kappa_0 > 0, \kappa_1 < 0$$



$$\kappa_0 = 0, \kappa_1 > 0$$



$$\kappa_0 = 0, \kappa_1 = 0$$

...

Normal Curvature

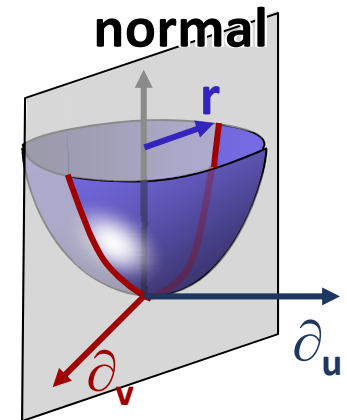
Definition:

- The *normal curvature* $\mathbf{k}(\mathbf{r})$ in direction \mathbf{r} for a unit length direction vector \mathbf{r} at parameter position \mathbf{x}_0 is given by:

$$\mathbf{k}_{\mathbf{x}_0}(\mathbf{r}) = \mathbf{II}_{\mathbf{x}_0}(\mathbf{r}, \mathbf{r}) = \mathbf{r}^T \mathbf{S}(\mathbf{x}_0) \mathbf{r}$$

Relation to Curvature of Plane Curves:

- Intersect the surface locally with plane spanned by **normal** and \mathbf{r} through point \mathbf{x}_0 .
- The curvature of the curve at \mathbf{x}_0 is equal to the normal curvature up to its sign.



Principal Curvatures

Relation to principal curvature:

- The maximum principal curvature κ_1 is the maximum of the normal curvature
- The minimum principal curvature κ_2 is the minimum of the normal curvature

Gaussian & Mean Curvature

More Definitions:

- The Gaussian curvature K is the product of the principal curvatures: $K = \kappa_1 \kappa_2$
- The mean curvature H is the average: $H = 0.5 \cdot (\kappa_1 + \kappa_2)$

Theorems:

- $K(\mathbf{x}_0) = \det(S(x_0)) = \frac{eg - f^2}{EG - F^2}$
- $H(\mathbf{x}_0) = \frac{1}{2} \text{tr}(S(x_0)) = \frac{eG - 2fF + gE}{2(EG - F^2)}$

Global Properties

Definition:

- An *isometry* is a mapping between surfaces that preserves distances on the surface (geodesics)
- A *developable surface* is a surface with Gaussian curvature zero everywhere (i.e. no curvature in at least one direction)
 - Examples: Cylinder, Cone, Plane
- A developable surface can be locally mapped to a plane isometrically (flattening out, unroll).

Theorema Egregium

Theorema egregium (Gauss):

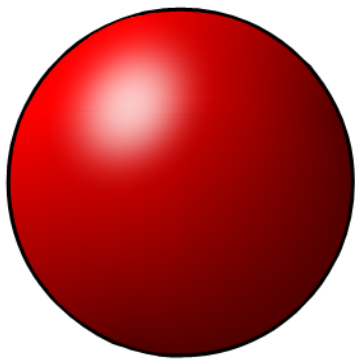
- Any isometric mapping preserves Gaussian curvature, i.e. Gaussian curvature is invariant under isometric maps (“intrinsic surface property”)
- Consequence: The earth (\approx sphere) cannot be mapped in an exactly length preserving way.

Gauss Bonnet Theorem

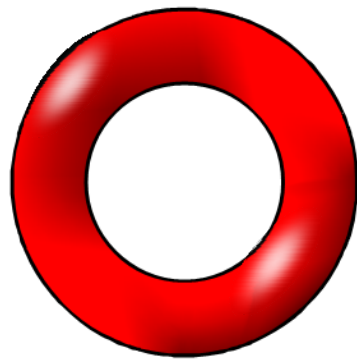
Gauss Bonnet Theorem:

For a compact, orientable surface without boundary in \mathbb{R}^3 , the area integral of the mean curvature is related to the genus g of the surface:

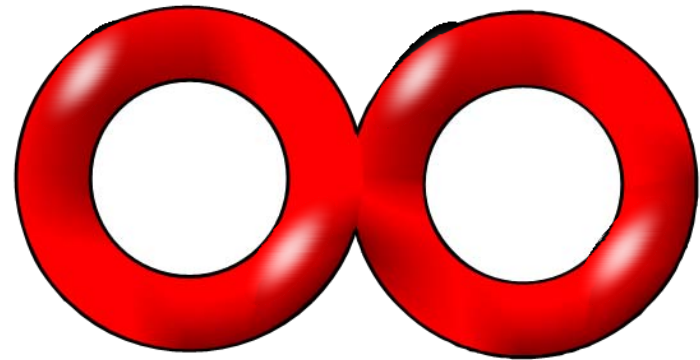
$$\int_S H(x) dx = 4\pi(1 - g)$$



$g = 0$



$g = 1$



$g = 2$

...

Fundamental Theorem of Surfaces

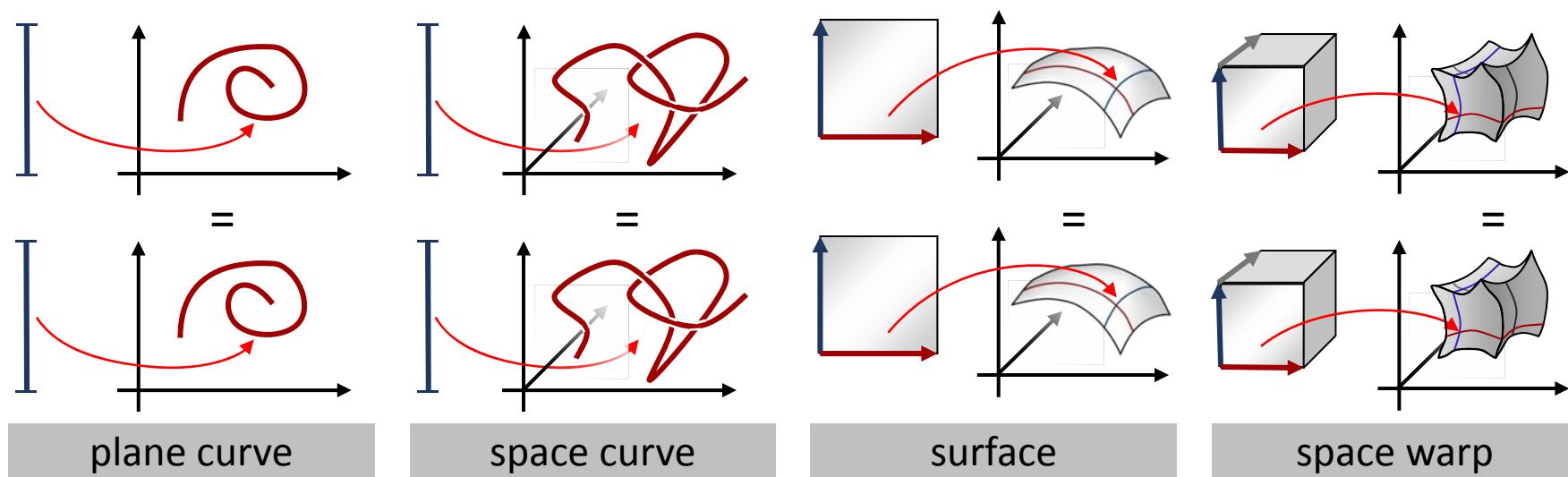
Theorem:

- Given two parametric patches in \mathbb{R}^3 defined on the same domain Ω .
- Assume that the first and second fundamental form are identical.
- Then there exists a rigid motion that maps one surface to the other.

Summary

Objects are the same up to a rigid motion, if...:

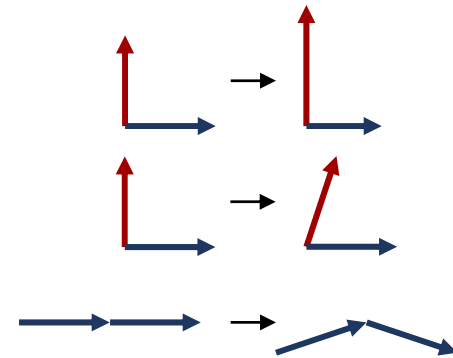
- Curves $\mathbb{R} \rightarrow \mathbb{R}^2$: Same *speed*, same *curvature*
- Curves $\mathbb{R} \rightarrow \mathbb{R}^3$: Same *speed*, same *curvature*, *torsion*
- Surfaces $\mathbb{R}^2 \rightarrow \mathbb{R}^3$: Same *first* & *second* fundamental form
- Volumetric Objects $\mathbb{R}^3 \rightarrow \mathbb{R}^3$: Same *first* fundamental form



Deformation Models

What if this does not hold?

- Deviation in fundamental forms is a measure of deformation
- Example: Surfaces
 - Diagonals of $\mathbf{I}_1 - \mathbf{I}_2$: **scaling** (stretching)
 - Off-diagonals of $\mathbf{I}_1 - \mathbf{I}_2$: **sheering**
 - Elements of $\mathbf{II}_1 - \mathbf{II}_2$: **bending**
- This is the basis of *deformation models*.



Reference: D. Terzopoulos, J. Platt, A. Barr, K. Fleischer: Elastically Deformable Models. In: *Siggraph '87 Conference Proceedings (Computer Graphics 21(4))*, 1987.

Geometric Registration for Deformable Shapes

1.3 4D Kinematic Surfaces

31st Annual Conference of the
European Association for Computer Graphics

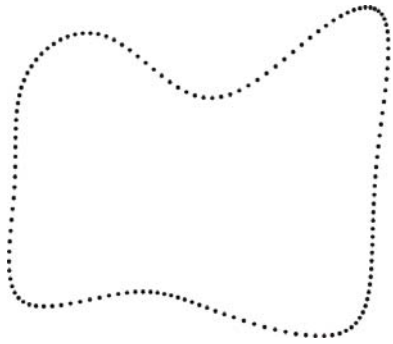
europa
graphics 2010

Rigid Motion

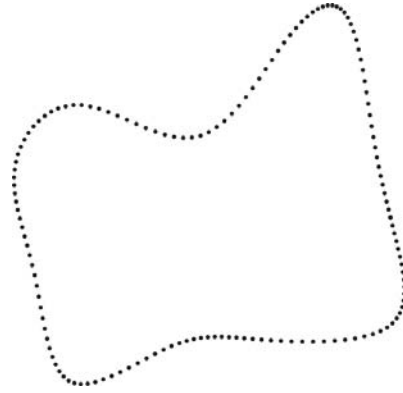
Scanning Moving Objects

Space-time Surface

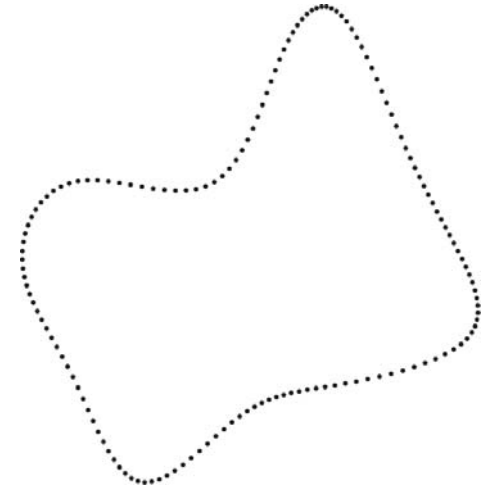
Time Ordered Scans



t^j



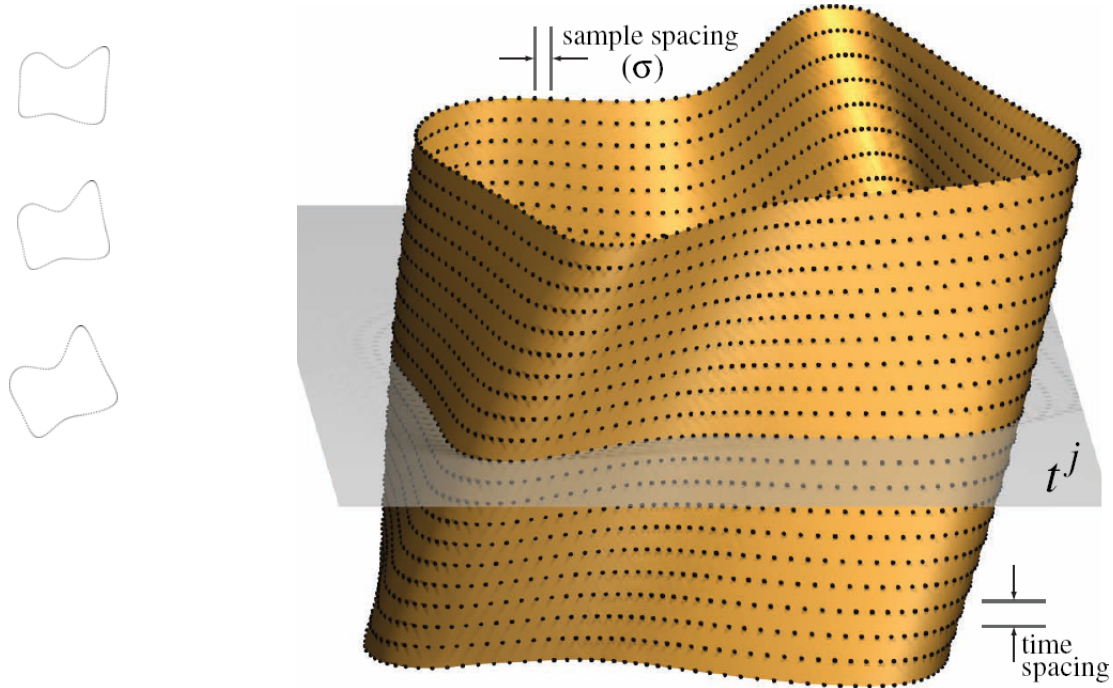
t^{j+1}



t^{j+2}

$$\tilde{P}^j \equiv \{\tilde{\mathbf{p}}_i^j\} := \{(\mathbf{p}_i^j, t^j), \mathbf{p}_i^j \in \mathbb{R}^d, t^j \in \mathbb{R}\}$$

Space-time Surface



Kinematic Surfaces

Geometric Registration for Deformable Shapes

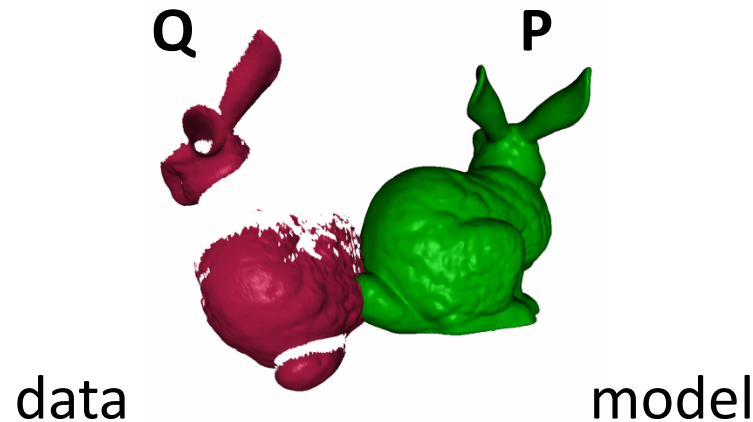
2.1 ICP + Tangent Space optimization for Rigid Motions

Rigid Motion

Registration Problem

Given

Two point cloud data sets \mathbf{P} (*model*) and \mathbf{Q} (*data*) sampled from surfaces $\Phi_{\mathbf{p}}$ and $\Phi_{\mathbf{Q}}$ respectively.



Assume $\Phi_{\mathbf{Q}}$ is a part of $\Phi_{\mathbf{p}}$.

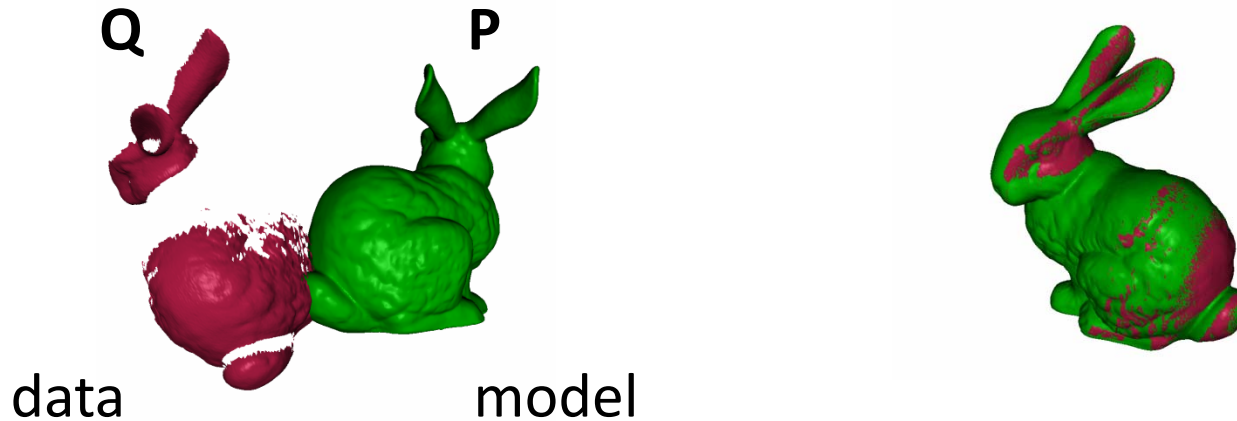
Registration Problem

Given

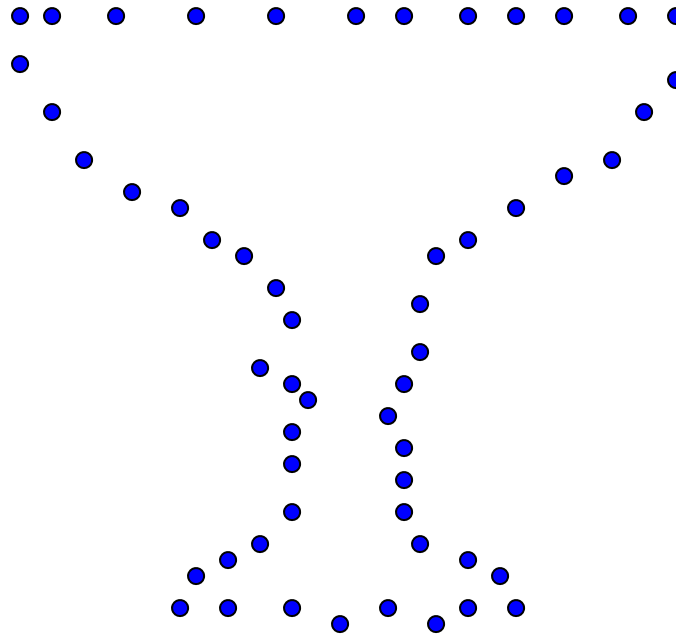
Two point cloud data sets P and Q .

Goal

Register Q against P by minimizing the squared distance between the underlying surfaces using only *rigid transforms*.

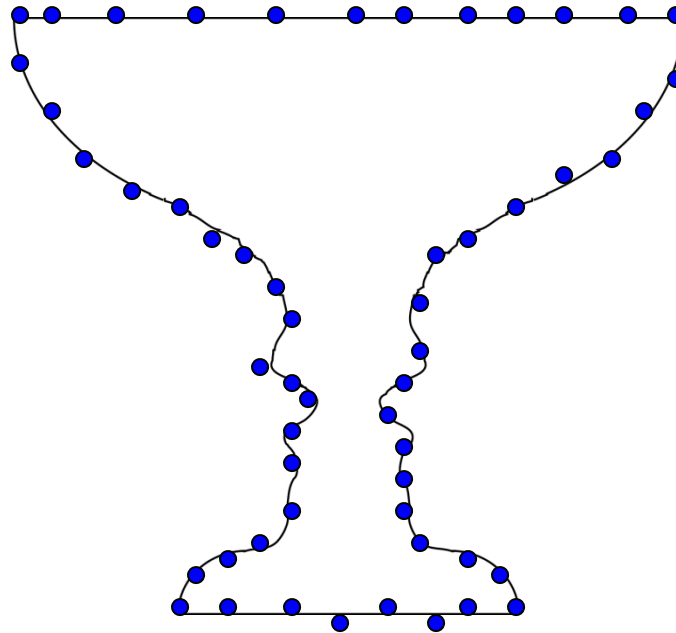


Notations



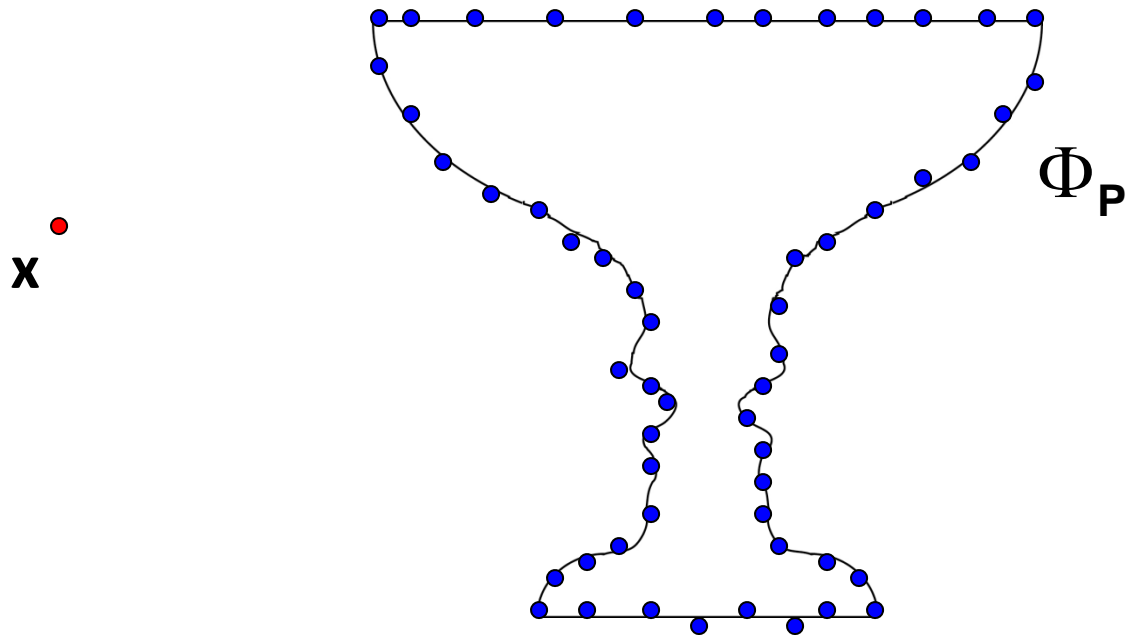
$$\mathbf{P} = \{\mathbf{p}_i\}$$

Notations

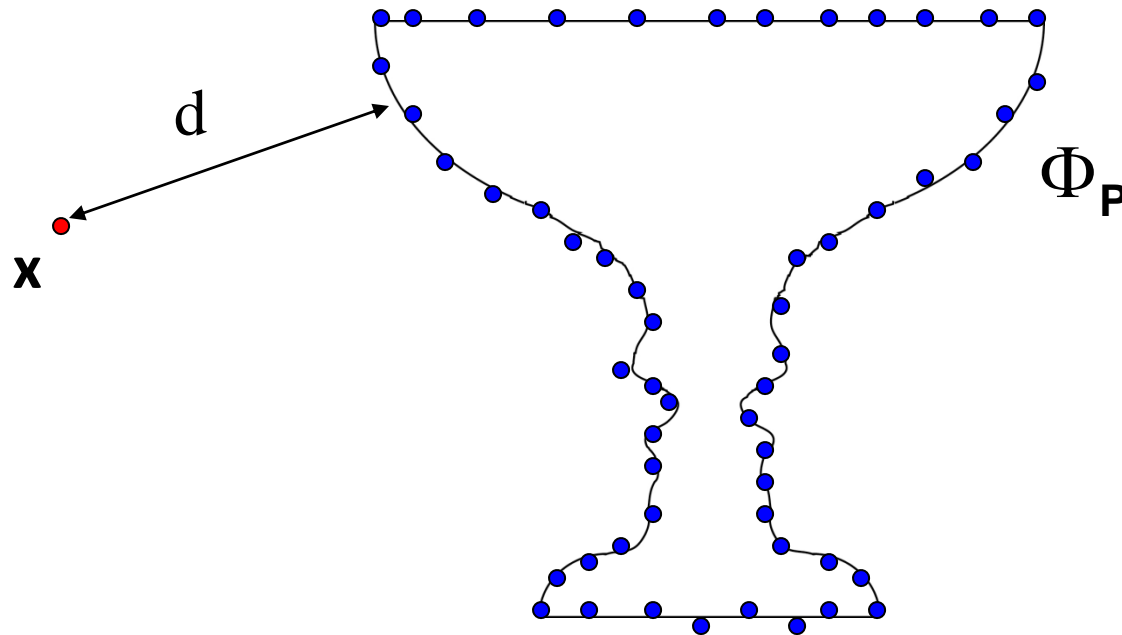


Φ_P

Squared Distance Function (F)



Squared Distance Function (F)



$$\mathbf{F}(\mathbf{x}, \Phi_P) = d^2$$

Registration Problem

Rigid transform α that takes points $\mathbf{q}_i \rightarrow \alpha(\mathbf{q}_i)$.

Our goal is to solve for,

$$\min_{\alpha} \sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{F}(\alpha(\mathbf{q}_i), \Phi_{\mathbf{P}})$$

An optimization problem in the squared distance field of \mathbf{P} , the model PCD.

Registration Problem

α = rotation (**R**) + translation(**t**)

Our goal is to solve for,

$$\min_{\mathbf{R}, \mathbf{t}} \sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{F}(\mathbf{R}\mathbf{q}_i + \mathbf{t}, \Phi_{\mathbf{P}})$$

Optimize for **R** and **t**.

Overview of Our Approach

Construct approximate $F^+(\mathbf{x}, \Phi_p)$ such that, $F^+(\mathbf{x}, \Phi_p) \approx F(\mathbf{x}, \Phi_p)$ to second order.

Linearize α .

Solve

$$\min_{R, t} \sum_{\mathbf{q}_i \in Q} F^+(\mathbf{R}\mathbf{q}_i + \mathbf{t}, \Phi_p)$$

to get a linear system.

Apply α to data PCD (Q) and iterate.

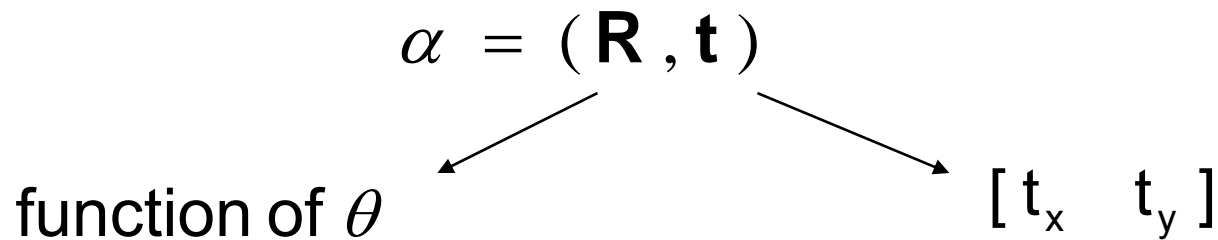
Registration in 2D

- Quadratic Approximant

$$\begin{aligned}\mathbf{F}^+(\mathbf{x}) &= Ax^2 + Bxy + Cy^2 + Dx + Ey + F \\ &= [x \ y \ 1] \mathbf{Q}_x [x \ y \ 1]^T\end{aligned}$$

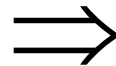
Registration in 2D

- Quadratic Approximant $\mathbf{F}^+(\mathbf{x}) = [\mathbf{x} \ y \ 1] \mathbf{Q}_x [\mathbf{x} \ y \ 1]^T$
- Linearize rigid transform



$$\sin(\theta) \approx \theta$$

$$\cos(\theta) \approx 1$$



$$x \rightarrow x - \theta y + \mathbf{t}_x$$

$$y \rightarrow \theta x + y + \mathbf{t}_y$$

Registration in 2D

- Quadratic Approximant $\mathbf{F}^+(\mathbf{x}) = [\mathbf{x} \ y \ 1] \mathbf{Q}_x [\mathbf{x} \ y \ 1]^T$
- Linearize rigid transform

$$\mathbf{x} \rightarrow \mathbf{x} - \theta \mathbf{y} + \mathbf{t}_x$$

$$\mathbf{y} \rightarrow \theta \mathbf{x} + \mathbf{y} + \mathbf{t}_y$$

- Residual error

$$\sum_{\mathbf{q}_i \in \mathbf{Q}} \mathbf{F}^+(\mathbf{R}\mathbf{q}_i + \mathbf{t}, \Phi_{\mathbf{P}}) = \varepsilon(\theta, \mathbf{t}_x, \mathbf{t}_y)$$

Registration in 2D

- Minimize residual error $\varepsilon(\theta, t_x, t_y)$

$$\begin{bmatrix} \mathbf{M}_1 \end{bmatrix} \begin{bmatrix} \theta \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} \mathbf{M}_2 \end{bmatrix}$$

Depends on \mathbf{F}^+ and data PCD (\mathbf{Q}).

Registration in 2D

- Minimize residual error $\varepsilon(\theta, t_x, t_y)$
- Solve for \mathbf{R} and \mathbf{t} .
- Apply a fraction of the computed motion
 - \mathbf{F}^+ valid locally
 - Step size determined by *Armijo condition*
 - Fractional transforms [Alexa et al. 2002]

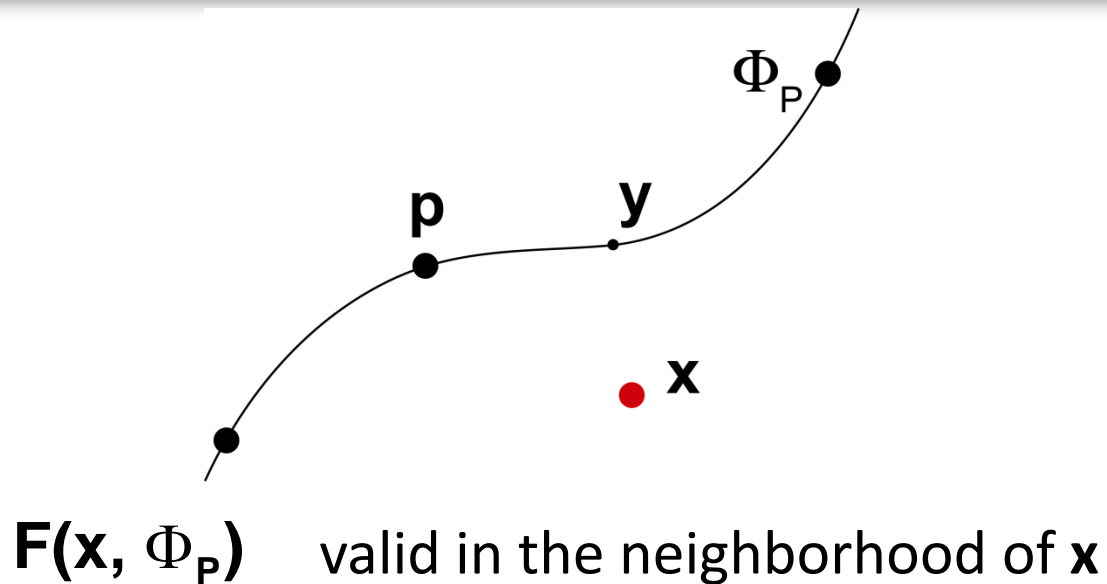
Registration in 3D

- Quadratic Approximant $\mathbf{F}^+(\mathbf{x}) = [x \ y \ z \ 1] \mathbf{Q}_x [x \ y \ z \ 1]^T$
- Linearize rigid transform
- Residual error

$$\varepsilon(\alpha, \beta, \gamma, t_x, t_y, t_z)$$

Minimize to get a linear system

Approximate Squared Distance



Two methods for estimating \mathbf{F}

1. d2Tree based computation
2. On-demand computation

$F(x, \Phi_p)$ using d2Tree

A kd-tree like data structure for storing approximants of the squared distance function.

Each cell (c) stores a quadratic approximant as a matrix Q_c .

Efficient to query.

[Leopoldseder et al. 2003]

$F(x, \Phi_p)$ using d2Tree

A kd-tree like data structure for storing approximants of the squared distance function.

Each cell (c) stores a quadratic approximant as a matrix Q_c .

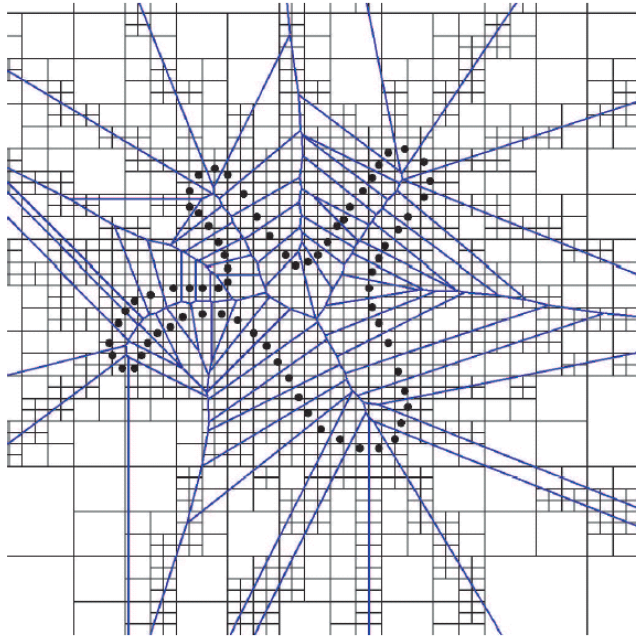
Efficient to query.

Simple bottom-up construction

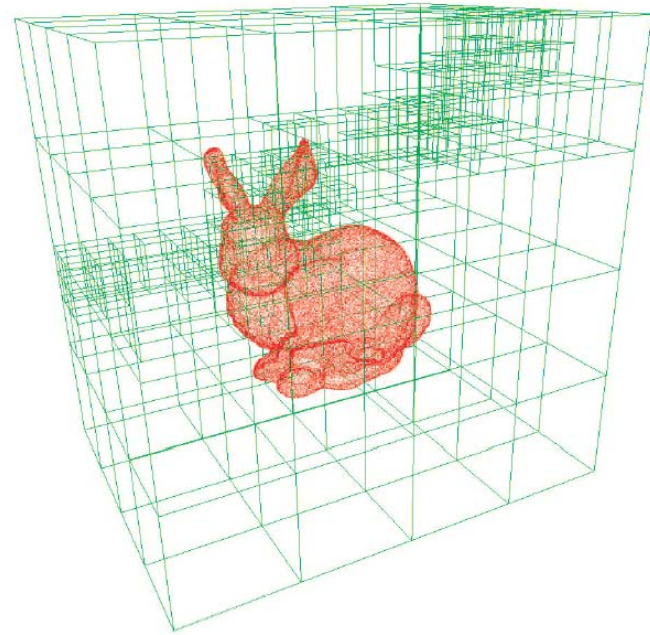
Pre-computed for a given PCD.

Closest point information implicitly embedded in the squared distance function.

Example d2trees



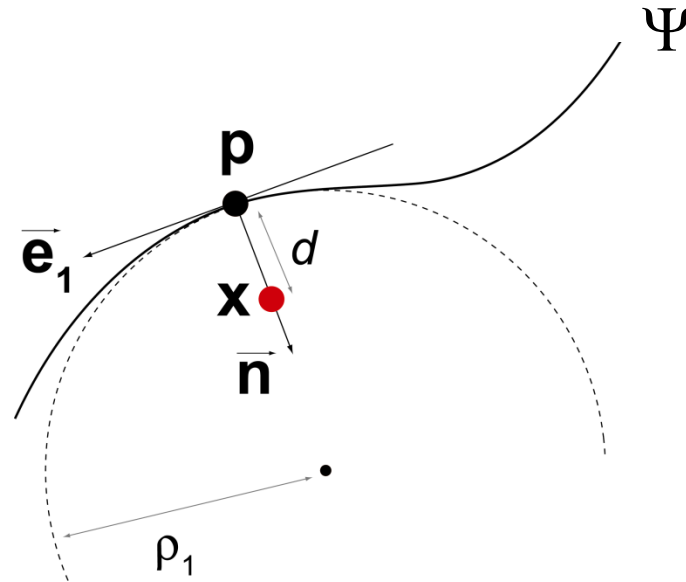
2D



3D

Approximate Squared Distance

For a curve Ψ ,



$$\mathbf{F}(\mathbf{x}, \Psi) = \frac{d}{d-\rho_1} \mathbf{x}_1^2 + \mathbf{x}_2^2 = \delta_1 \mathbf{x}_1^2 + \mathbf{x}_2^2$$

[Pottmann and Hofer 2003]

Approximate Squared Distance

For a curve Ψ ,

$$\mathbf{F}(\mathbf{x}, \Psi) = \frac{d}{d-\rho_1} \mathbf{x}_1^2 + \mathbf{x}_2^2 = \delta_1 \mathbf{x}_1^2 + \mathbf{x}_2^2$$

For a surface Φ ,

$$\mathbf{F}(\mathbf{x}, \Phi) = \frac{d}{d-\rho_1} \mathbf{x}_1^2 + \frac{d}{d-\rho_2} \mathbf{x}_2^2 + \mathbf{x}_3^2 = \delta_1 \mathbf{x}_1^2 + \delta_2 \mathbf{x}_2^2 + \mathbf{x}_3^2$$

[Pottmann and Hofer 2003]

On-demand Computation

Given a PCD, at each point \mathbf{p} we pre-compute,

- a local frame
 - normal ($\vec{\mathbf{n}}$)
 - principal direction of curvatures ($\vec{\mathbf{e}}_1$ and $\vec{\mathbf{e}}_2$)
- radii of principal curvature (ρ_1 and ρ_2)

On-demand Computation

Given a PCD, at each point \mathbf{p} we pre-compute,

- a local frame
 - normal ($\vec{\mathbf{n}}$)
 - principal direction of curvatures ($\vec{\mathbf{e}}_1$ and $\vec{\mathbf{e}}_2$)
- radii of principal curvature (ρ_1 and ρ_2)

Estimated from a PCD using local analysis

- *covariance analysis* for local frame
- *quadric fitting* for principal curvatures

On-demand Computation

Given a point \mathbf{x} ,

nearest neighbor (\mathbf{p}) computed using approximate
nearest neighbor (ANN) data structure

$$\mathbf{F}^+(\mathbf{x}, \Phi_{\mathbf{p}}) = \delta_1 (\vec{\mathbf{e}}_1 \cdot (\mathbf{x} - \mathbf{p}))^2 + \delta_2 (\vec{\mathbf{e}}_2 \cdot (\mathbf{x} - \mathbf{p}))^2 + (\vec{\mathbf{n}} \cdot (\mathbf{x} - \mathbf{p}))^2$$

$$\text{where } \delta_j = \begin{cases} d/(d-\rho_j) & \text{if } d < 0 \\ 0 & \text{otherwise.} \end{cases}$$

Iterated Closest Point (ICP)

Iterate

1. Find correspondence between **P** and **Q**.
 - closest point (point-to-point).
 - tangent plane of closest point (point-to-plane).
2. Solve for the best rigid transform given the correspondence.

ICP in Our Framework

- Point-to-point ICP (good for large d)

$$F(\mathbf{x}, \Phi_{\mathbf{p}}) = (\mathbf{x} - \mathbf{p})^2 \quad \Rightarrow \quad \delta_j = 1$$

- Point-to-plane ICP (good for small d)

$$F(\mathbf{x}, \Phi_{\mathbf{p}}) = (\vec{\mathbf{n}} \cdot (\mathbf{x} - \mathbf{p}))^2 \quad \Rightarrow \quad \delta_j = 0$$

Convergence Properties

Gradient decent over the error landscape

Gauss -Newton Iteration

Zero residue problem (model and data PCD-s match)

Quadratic Convergence

For fractional steps, Armijo condition used

Damped Gauss-Newton Iteration

Linear convergence

- can be improved by quadratic motion approximation (not currently used)

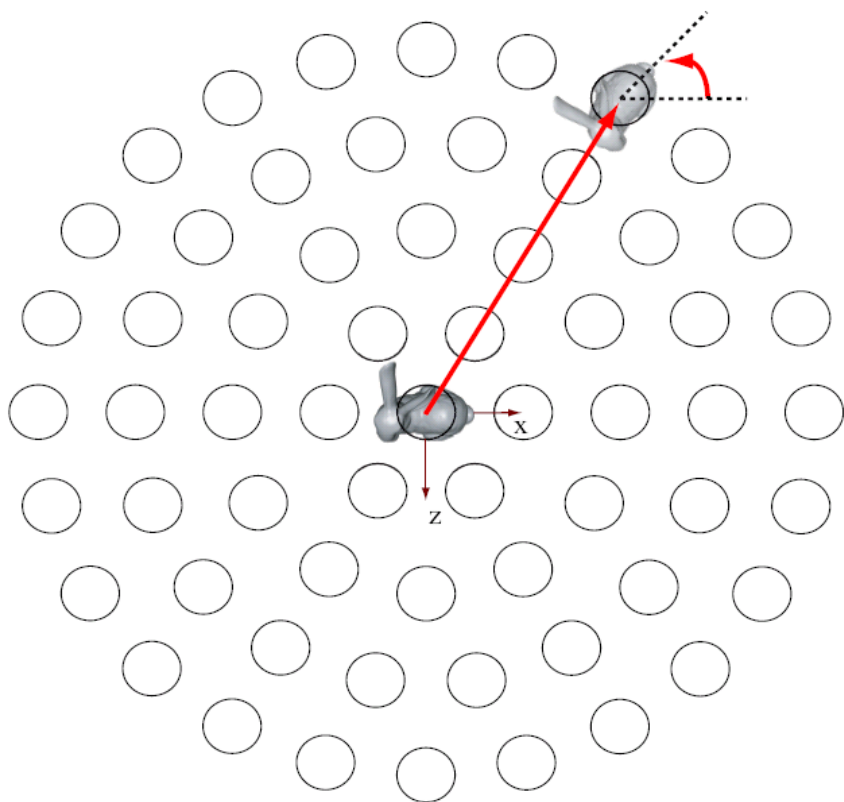
Convergence Funnel

Set of all initial poses of the data PCD with respect to the model PCD that is successfully aligned using the algorithm.

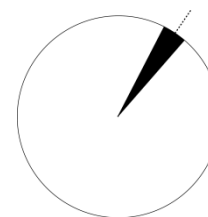
Desirable properties

- broad
- stable

Convergence Funnel



Translation in x-z plane.
Rotation about y-axis.

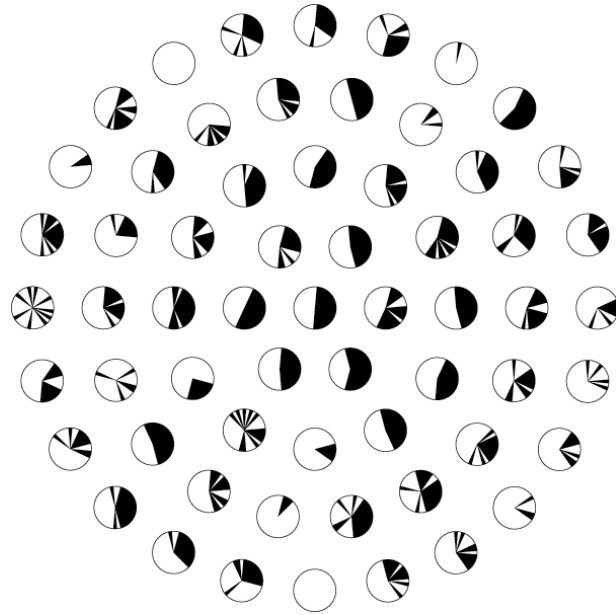


Converges

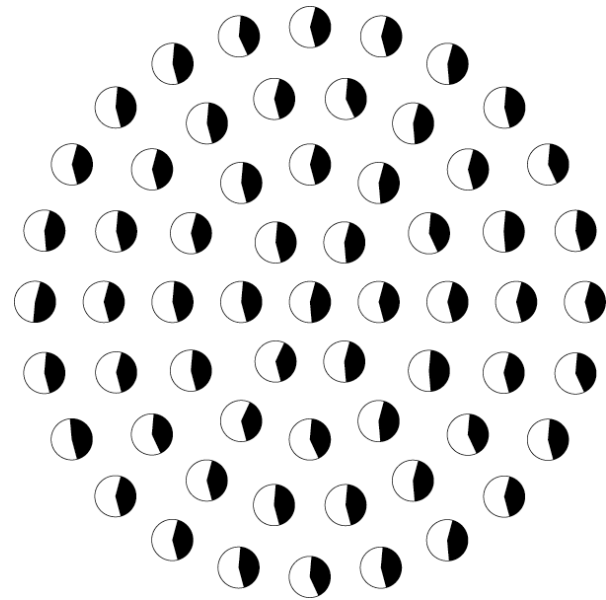


Does not converge

Convergence Funnel

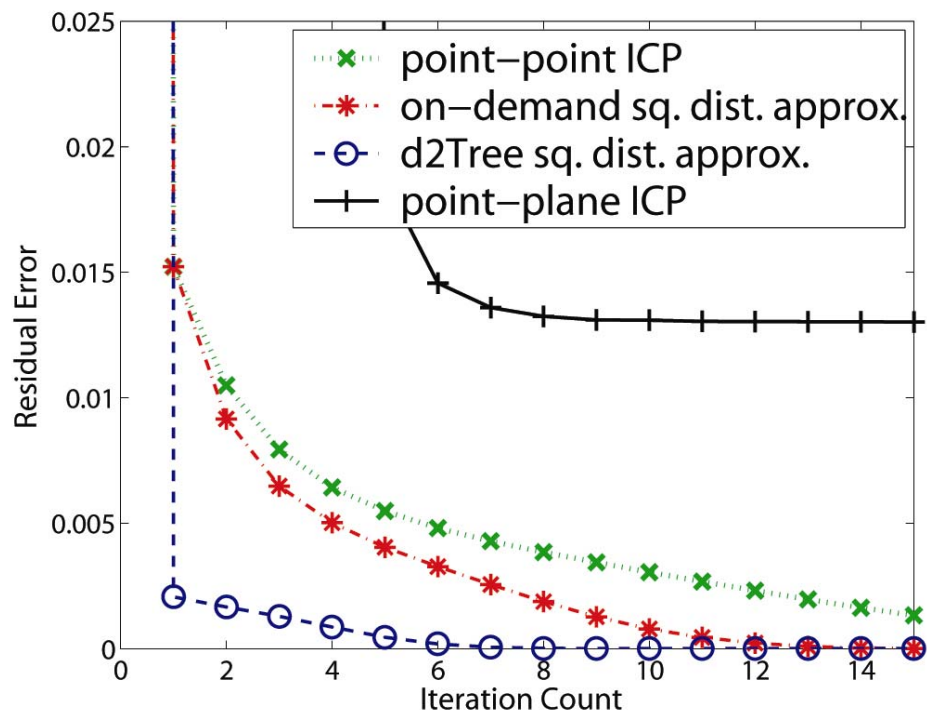


Plane-to-plane ICP



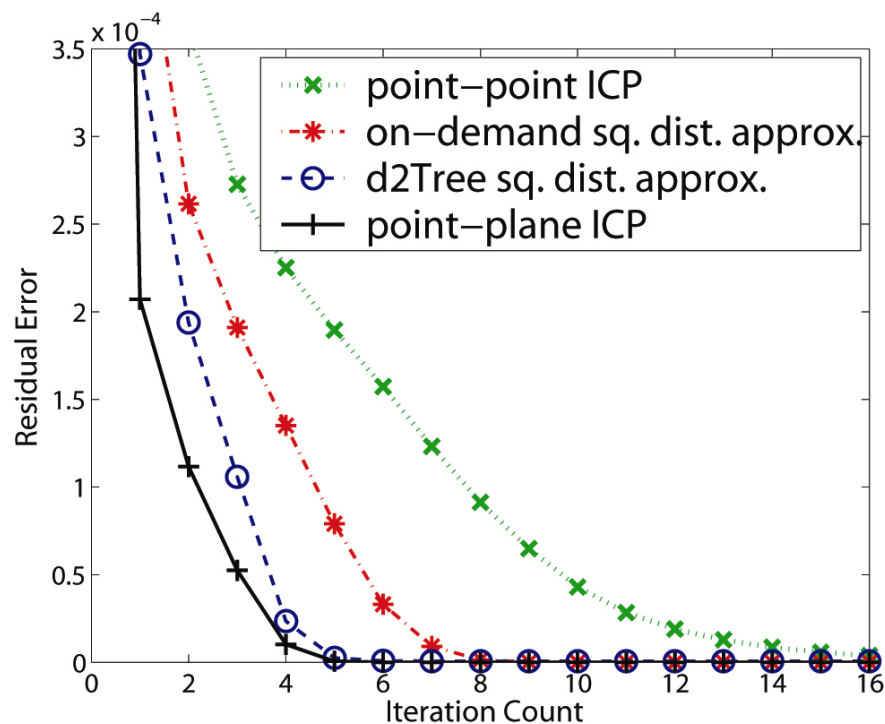
Our algorithm

Convergence Rate I



Bad Initial Alignment

Convergence Rate II



Good Initial Alignment

Partial Alignment



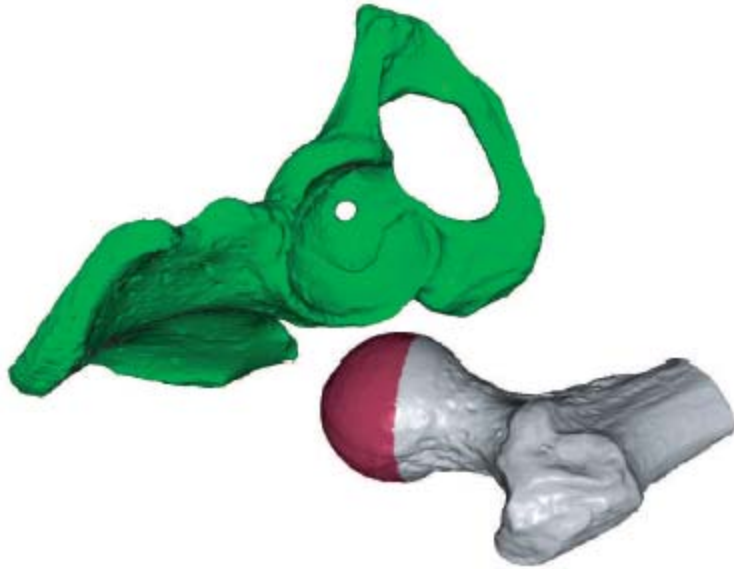
Starting Position

Partial Alignment



After 6 iterations

Partial Alignment



Different sampling density



After 6 iterations

Closed Form Solution

How to Establish Correspondence?

When Objects are *Almost* Aligned

ICP

Convergence Funnel

Convergence Rate

Improvements

Tangent Space

Instantaneous Formulation

Geometric Registration for Deformable Shapes

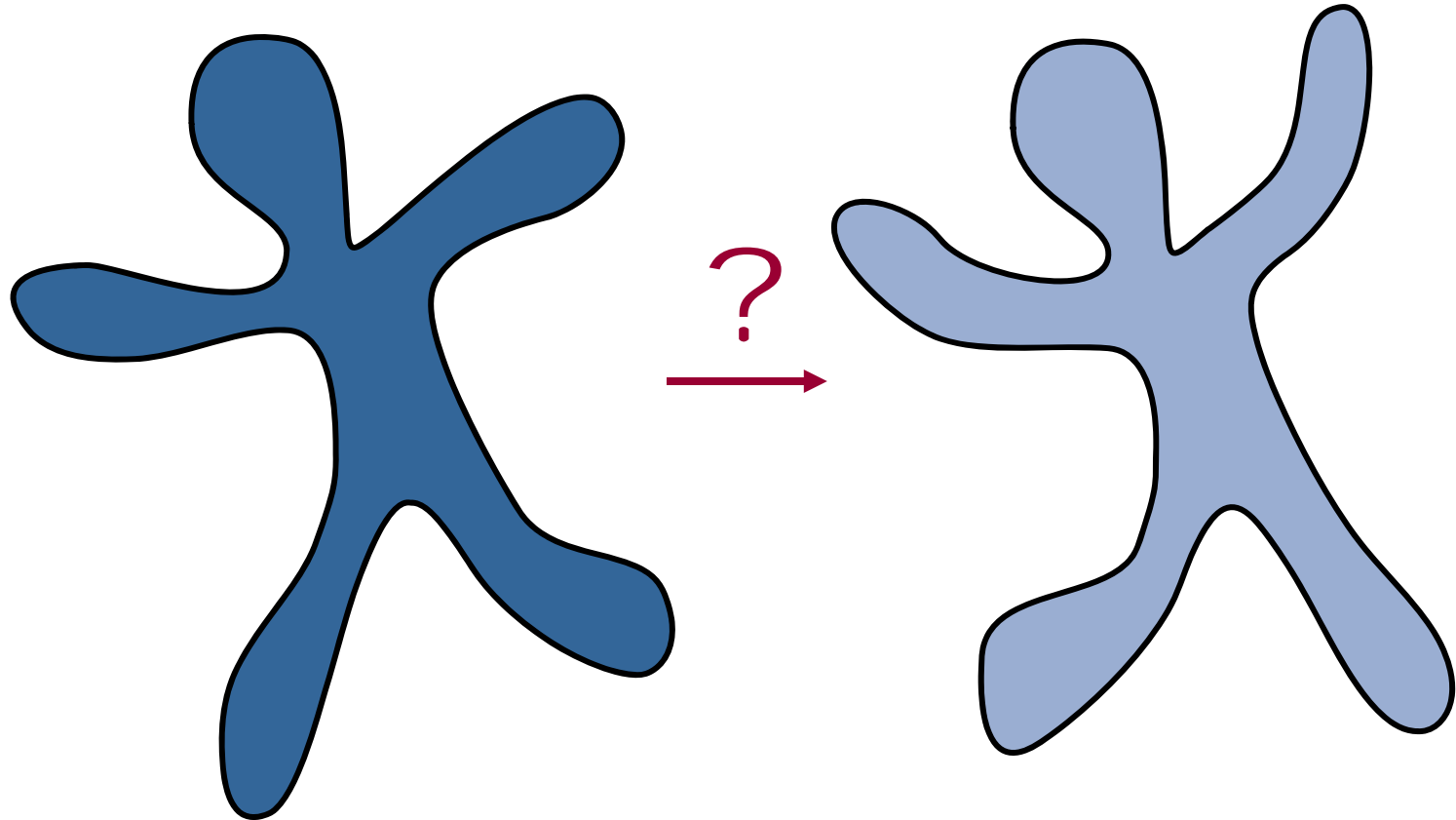
2.2 Deformable Registration

Variational Model • Deformable ICP

Variational Model

What is deformable shape matching?

Example



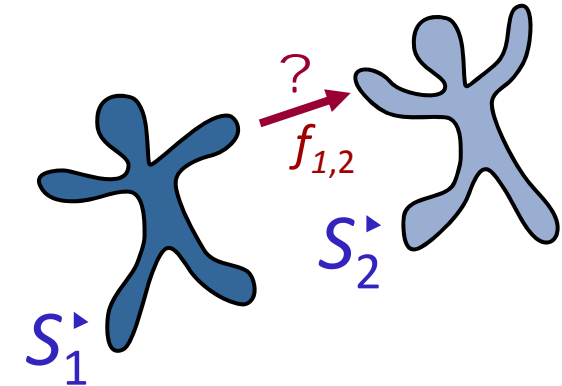
What are the Correspondences?

What are we looking for?

Problem Statement:

Given:

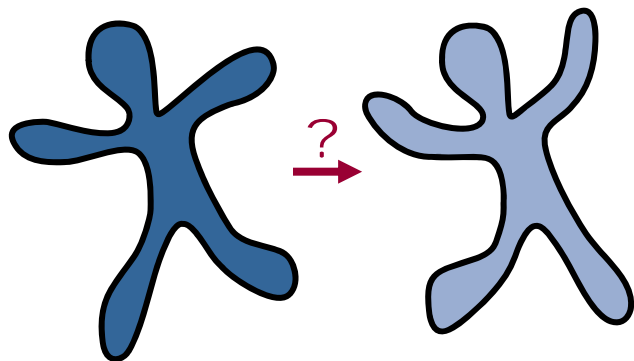
- Two surfaces $S_1, S_2 \subseteq \mathbb{R}^3$



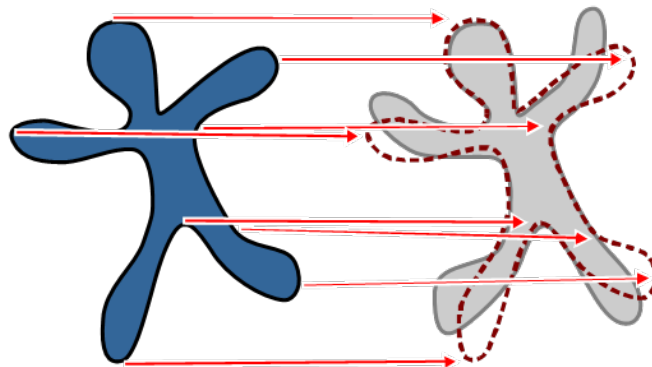
We are looking for:

- A *reasonable* deformation function $f_{1,2}: S_1 \rightarrow \mathbb{R}^3$ that brings S_1 close to S_2

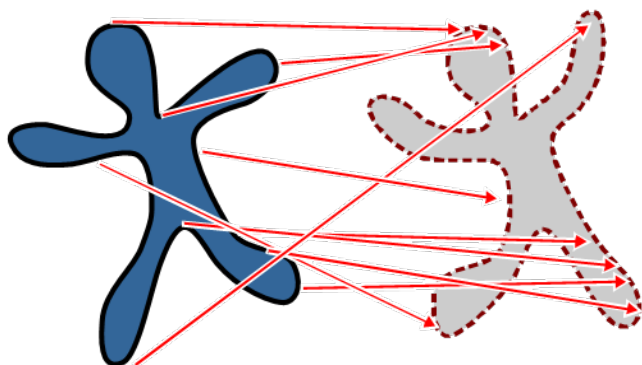
Example



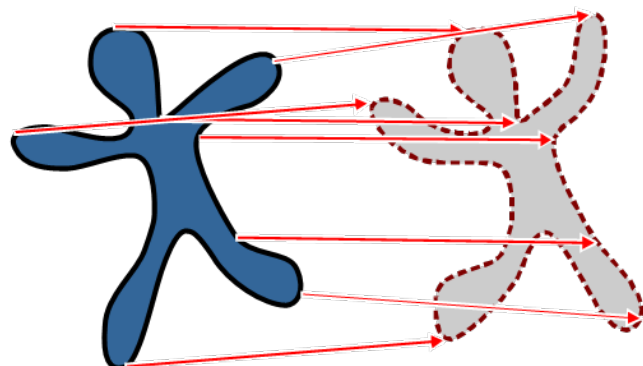
Correspondences?



X no shape match



X too much deformation

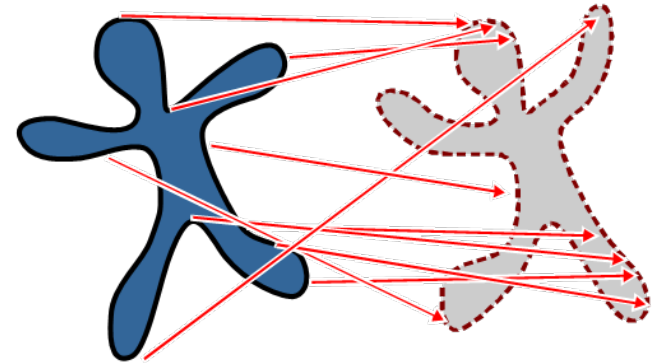


✓ optimum

This is a Trade-Off

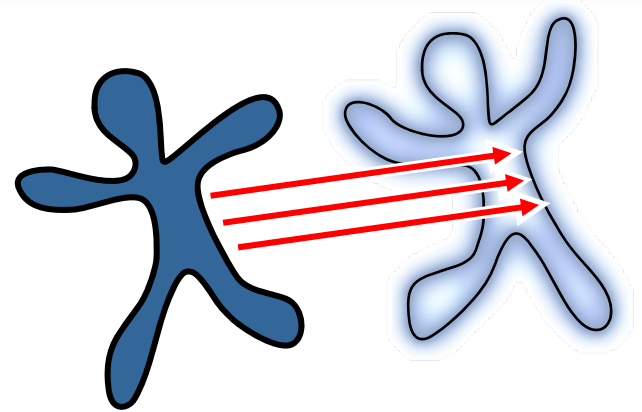
Deformable Shape Matching is a Trade-Off:

- We can match any two shapes using a weird deformation field
- We need to trade off
 - Shape matching (close to data)
 - Regularity of the deformation field (reasonable match)

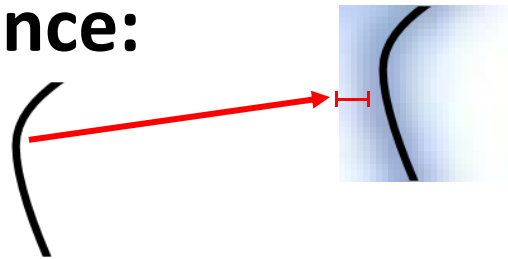


Variational Model

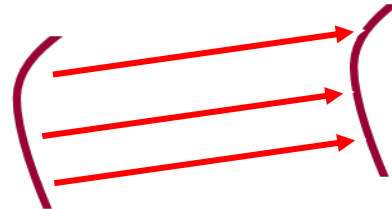
Components:



Matching Distance:



Deformation / rigidity:

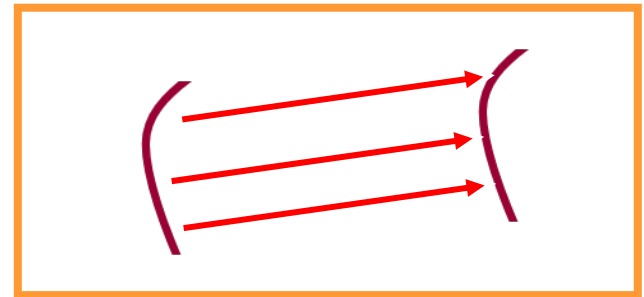
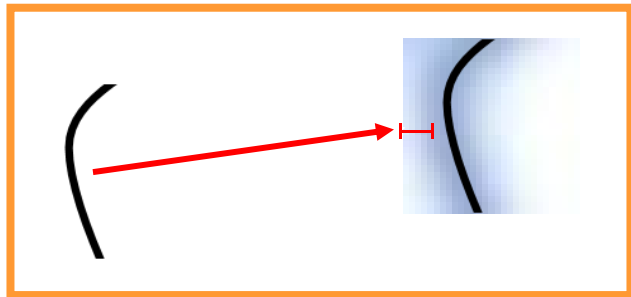


Variational Model

Variational Problem:

- Formulate as an energy minimization problem:

$$E(f) = E^{(match)}(f) + E^{(regularizer)}(f)$$



Part 1: Shape Matching

Assume:

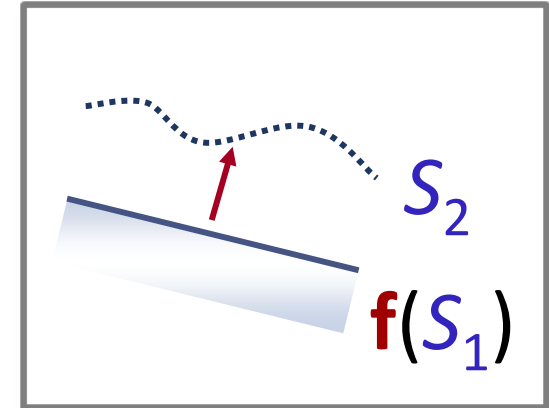
- Objective Function:

$$E^{(match)}(f) = \text{dist}(f(S_1), S_2)$$

- Example: least squares distance

$$E^{(match)}(f) = \int_{x_1 \in S_1} \text{dist}(\mathbf{x}_1, S_2)^2 d\mathbf{x}_1$$

- Other distance measures:
Hausdorff distance, L_p -distances, etc.
- L_2 measure is frequently used (models Gaussian noise)



Point Cloud Matching

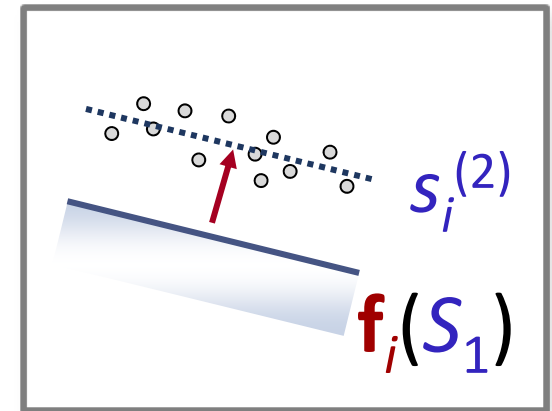
Implementation example: Scan matching

- Given: S_1, S_2 as point clouds
 - $S_1 = \{\mathbf{s}_1^{(1)}, \dots, \mathbf{s}_n^{(1)}\}$
 - $S_2 = \{\mathbf{s}_1^{(2)}, \dots, \mathbf{s}_m^{(2)}\}$

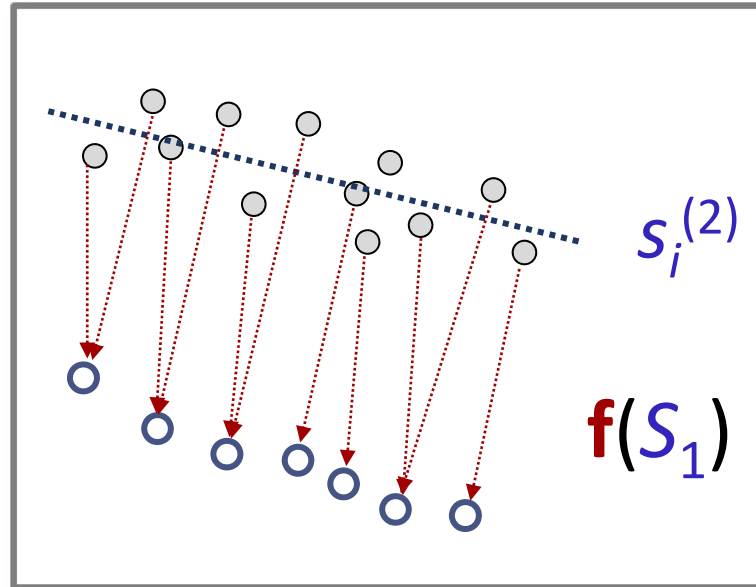
- Energy function:

$$E^{(match)}(f) = \frac{|S_1|}{m} \sum_{i=1}^m \text{dist}(S_1, \mathbf{s}_i^{(2)})^2$$

- How to measure $\text{dist}(S_1, \mathbf{x})$?
 - Estimate distance to a point sampled surface



Surface approximation

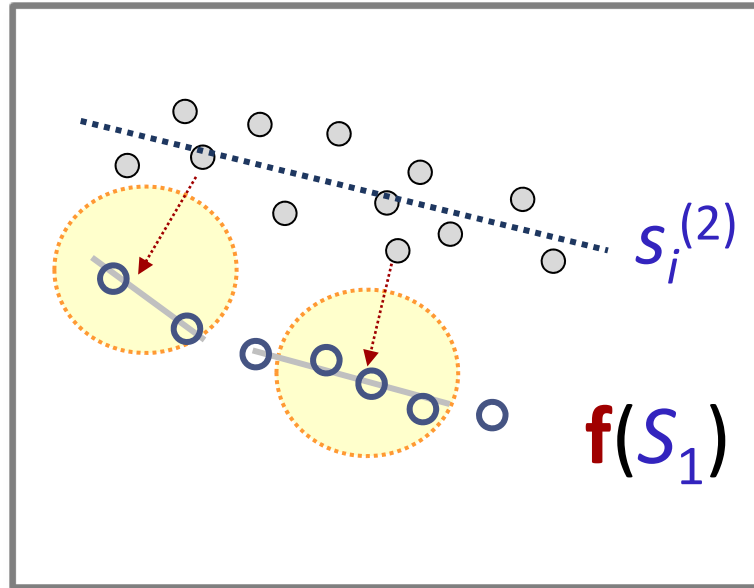


Solution #1: Closest point matching

- “Point-to-point” energy

$$E^{(match)}(f) = \frac{|S_1|}{m} \sum_{i=1}^m \text{dist}\left(s_i^{(2)}, NN_{in S_1}(s_i^{(2)})\right)^2$$

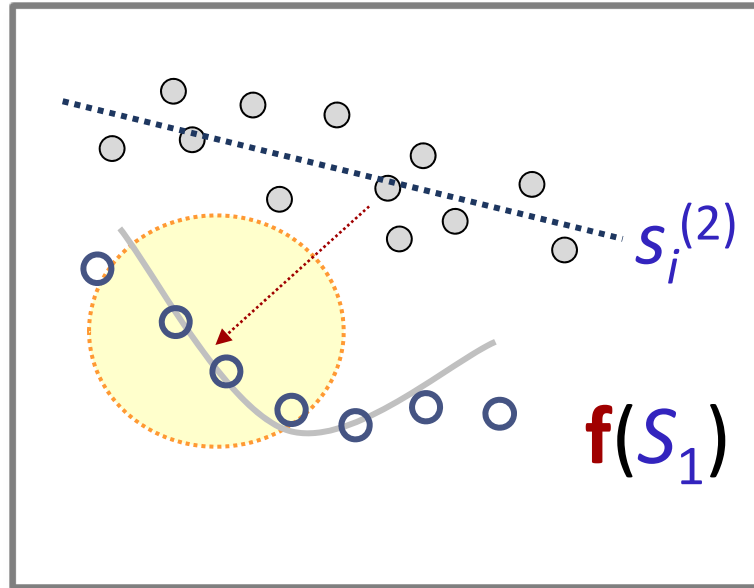
Surface approximation



Solution #2: Linear approximation

- “Point-to-plane” energy
- Fit plane to k -nearest neighbors
- k proportional to noise level, typically $k \approx 6 \dots 20$

Surface approximation



Solution #3: Higher order approximation

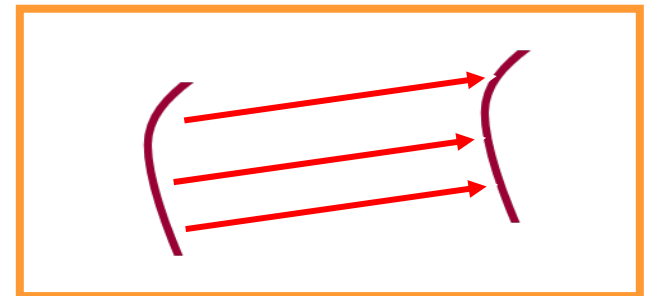
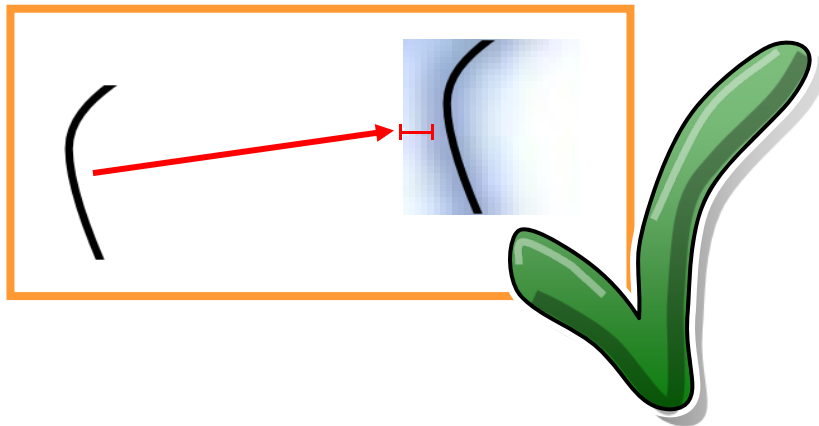
- Higher order fitting (e.g. quadratic)
 - Moving least squares
- Rarely used: No close form solution for distance
- Point-to-plane recommended in practice

Variational Model

Variational Problem:

- Formulate as an energy minimization problem:

$$E(f) = E^{(match)}(f) + E^{(regularizer)}(f)$$

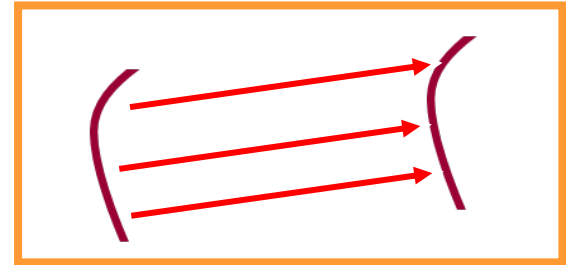


Part II: Deformation Model

What is a “nice” deformation field?

- Isometric energies
 - Extrinsic (“volumetric deformation”)
 - Intrinsic (“as-isometric-as possible embedding”)
- Elastic energy
 - Physical model: The model is made of rubber
- Thin-plate splines
 - Allowing strong deformations, but keep shape
- Approximations
 - Laplacian surface deformation

$$E^{(regularizer)}(f)$$



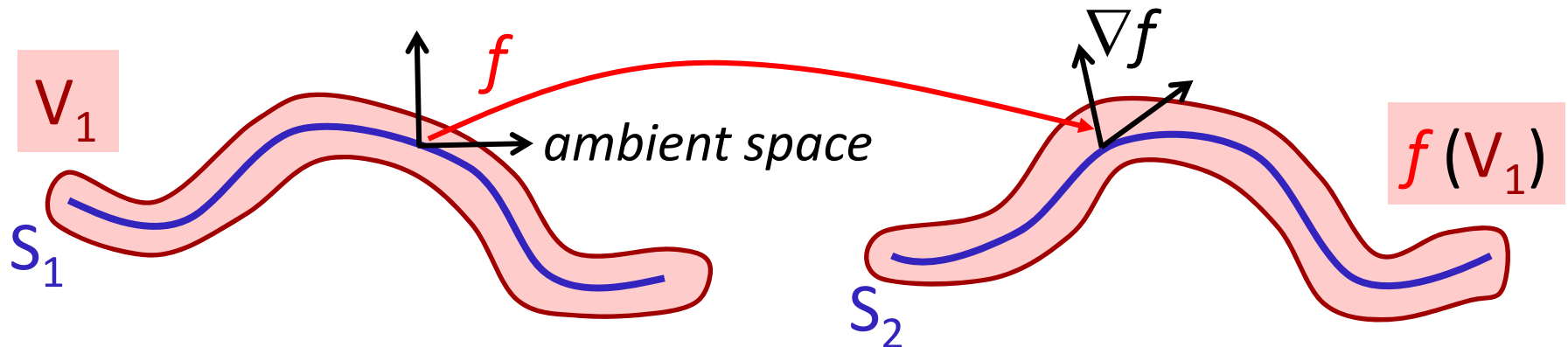
Volume Model

Extrinsic Volumetric “As-Rigid-As Possible”

- Embed source surface S_1 in volume
- f should preserve 3×3 metric tensor (least squares)

$$E^{(\text{regularizer})}(f) = \int_{V_1} \left[\nabla f \nabla f^T - \mathbf{I} \right]^2 dx$$

metric tensor (\mathbb{R}^3)



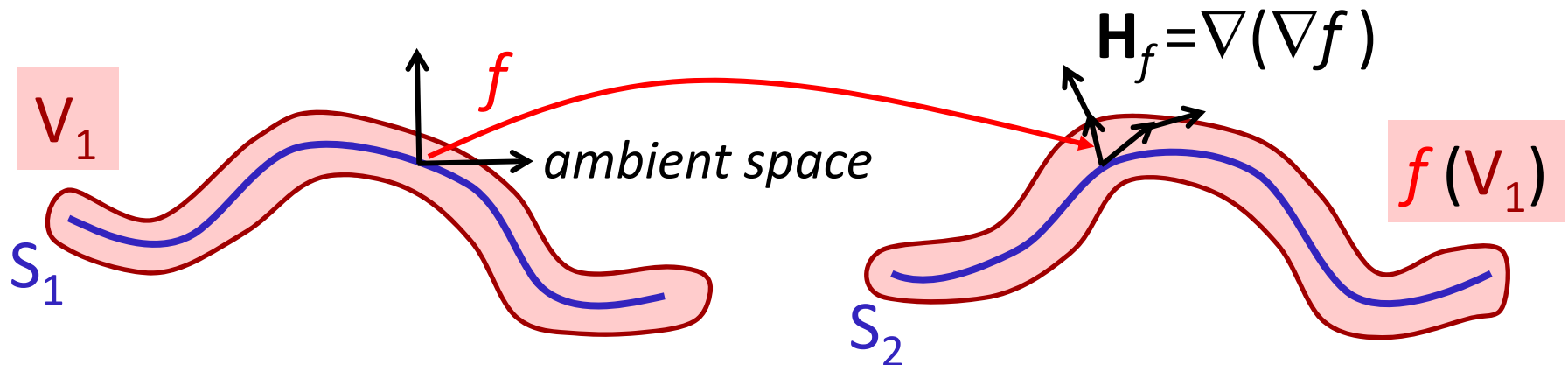
Volume Model

Variant: Thin-Plate-Splines

- Use regularizer that penalizes curved deformation

$$E^{(\text{regularizer})}(f) = \int_{V_1} H_f(x)^2 dx$$

second derivative (\mathbb{R}^3)



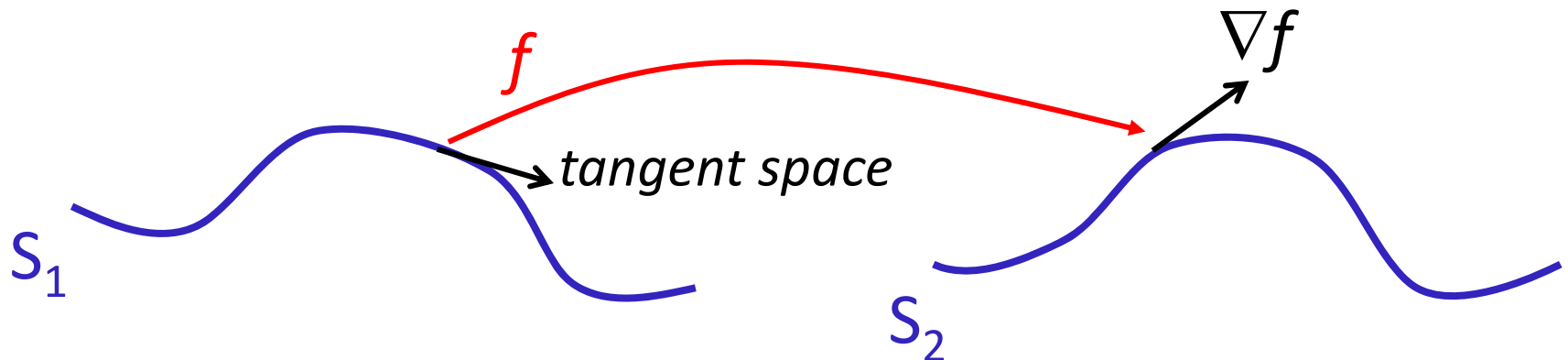
Isometric Regularizer

Intrinsic Matching (2-Manifold)

- Target shape is given and *complete*
- Isometric embedding

$$E^{(regularizer)}(f) = \int_{S_1} \left[\boxed{\nabla f \nabla f^T} - \mathbf{I} \right]^2 dx$$

metric tensor (S_1 , intrinsic)



Elastic Regularizer

Elastic Regularizer

- Differential geometry point of view
 - Preserve metric tensor (least squares)
 - Preserve curvature tensor (least squares)
 - „Thin-shell model“
- Complicated to implement
- Usually approximated
 - Volumetric shells (as shown before)
 - Other approximation (next slide)

Example Implementation

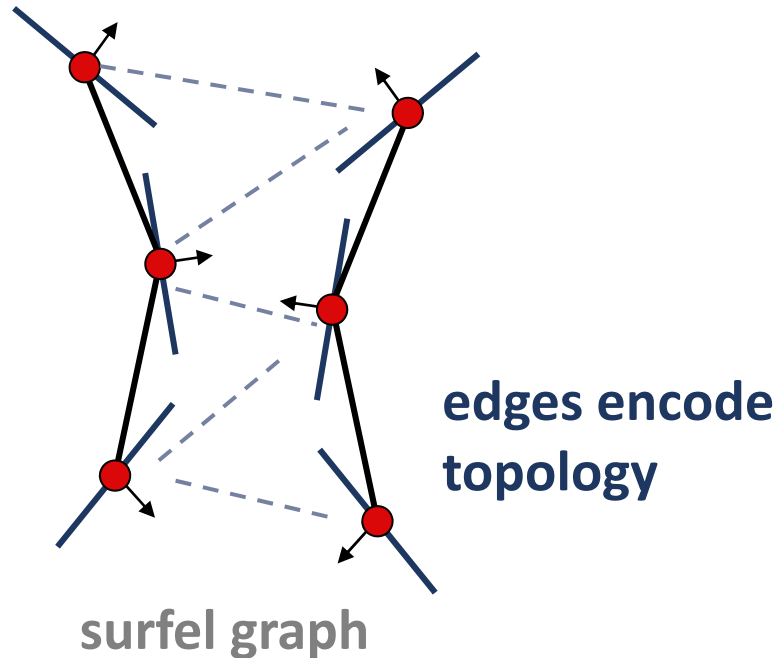
Example: elastic / as-rigid-as possible model

- Idee: associate local *rigid* transformation with surface points
- Optimize simultaneously with deformed surface
- Transformation is *implicitly defined* by deformed surface (*and vice versa*)

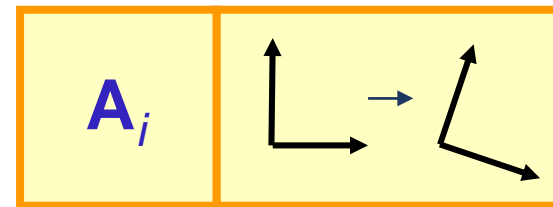
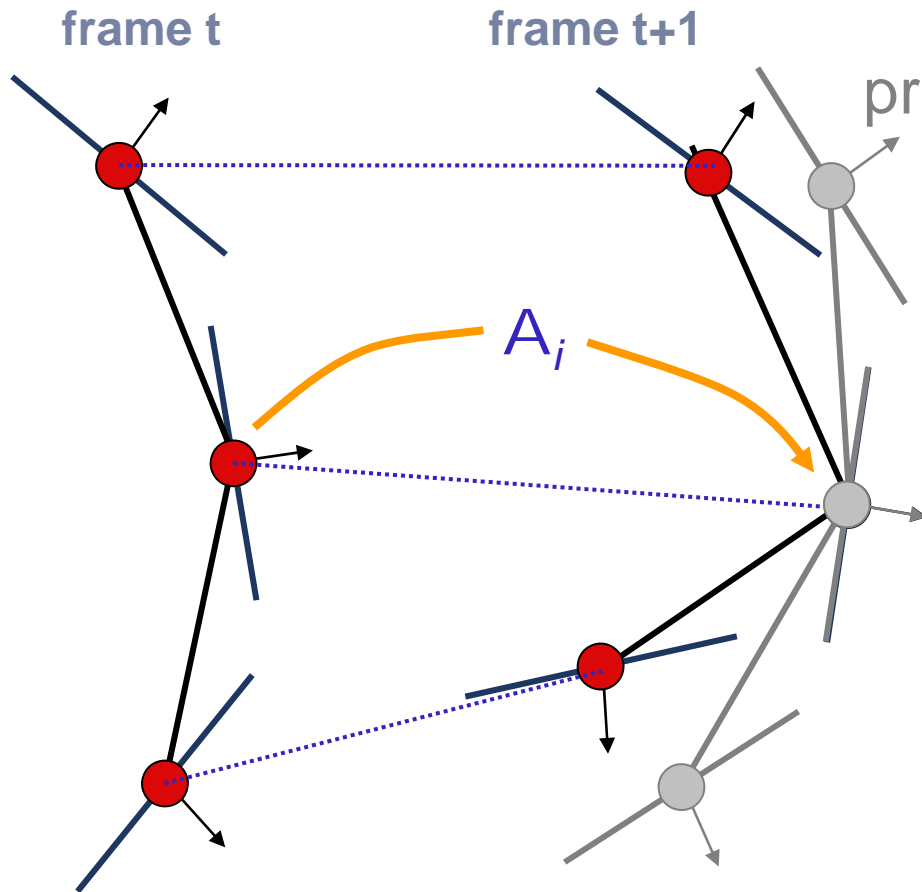
Parameterization

Parameterization of S_1

- Surfel graph
- This could be a mesh, but does not need to



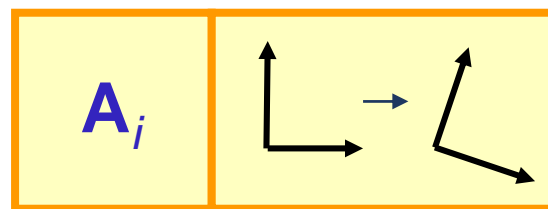
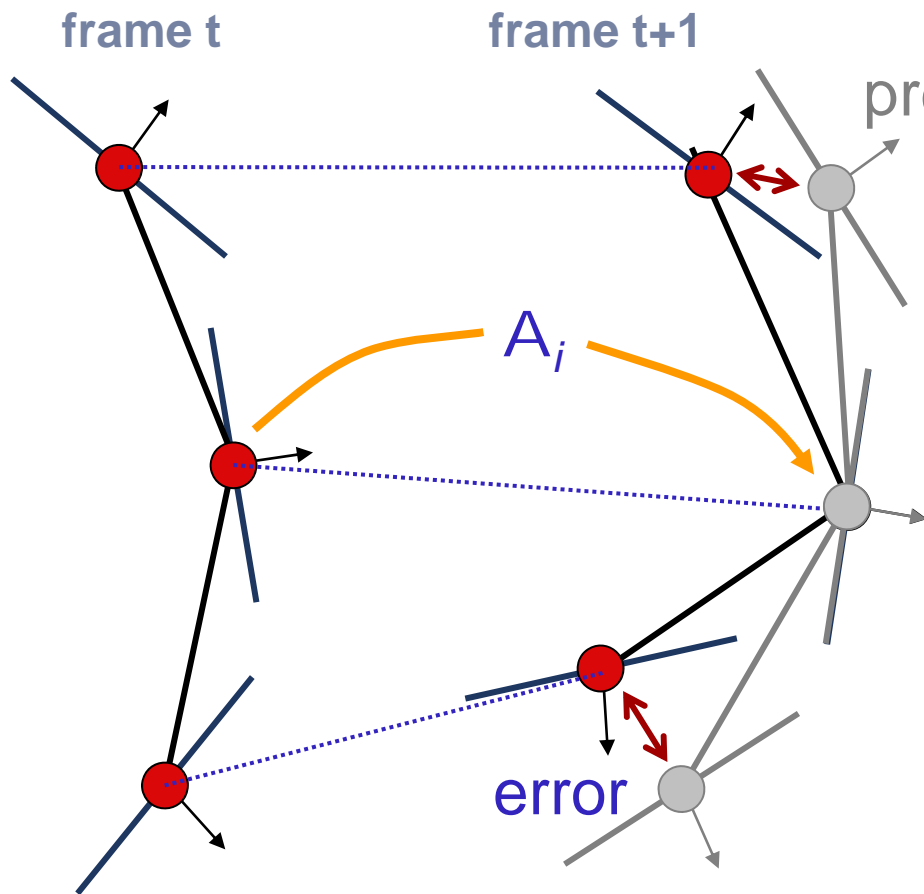
Deformation



Orthonormal Matrix A_i

per surfel (neighborhood),
latent variable

Deformation



Orthonormal Matrix A_i

per surfel (neighborhood),
latent variable

$$E^{(regularizer)} = \sum_{surfels} \sum_{neighbors} \left[A_i^t (\mathbf{s}_i^{(t)} - \mathbf{s}_{i_j}^{(t)}) - (\mathbf{s}_i^{(t+1)} - \mathbf{s}_{i_j}^{(t+1)}) \right]^2$$

Unconstrained Optimization

Orthonormal matrices

- Local, 1st order, non-degenerate parametrization:

$$\mathbf{C}_{\mathbf{x}_i}^{(t)} = \begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -\gamma & 0 \end{pmatrix} \quad \begin{aligned} \mathbf{A}_i &= \mathbf{A}_0 \exp(\mathbf{C}_{\mathbf{x}_i}) \\ &\doteq \mathbf{A}_0 (I + \mathbf{C}_{\mathbf{x}_i}^{(t)}) \end{aligned}$$

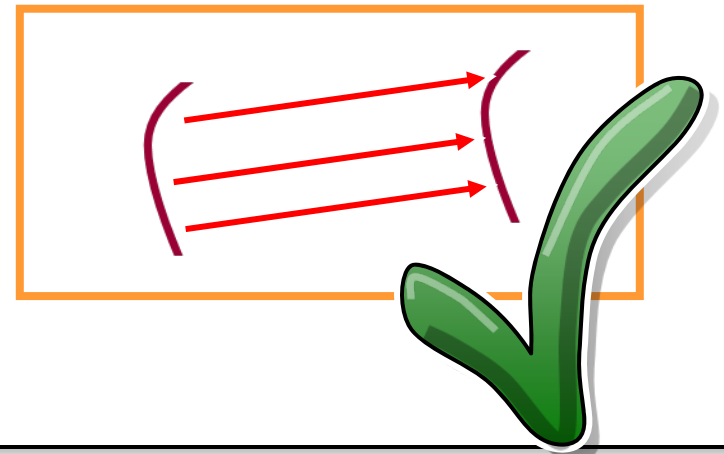
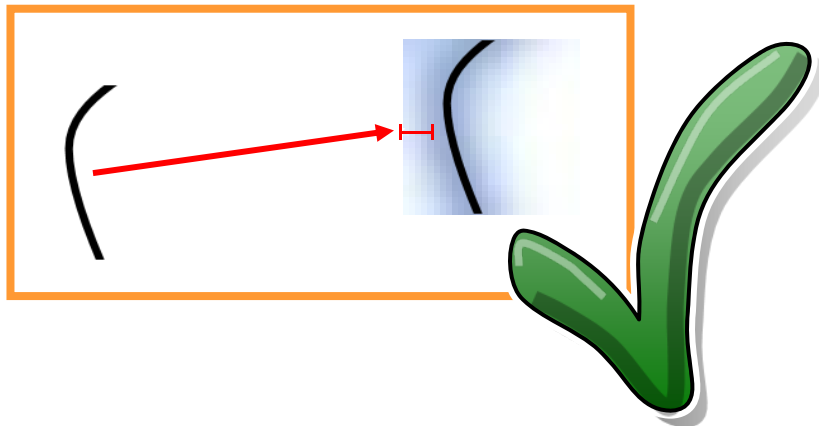
- Optimize parameters α , β , γ , then recompute \mathbf{A}_0
- Compute initial estimate using [*Horn 87*]

Variational Model

Variational Problem:

- Formulate as an energy minimization problem:

$$E(f) = E^{(match)}(f) + E^{(regularizer)}(f)$$



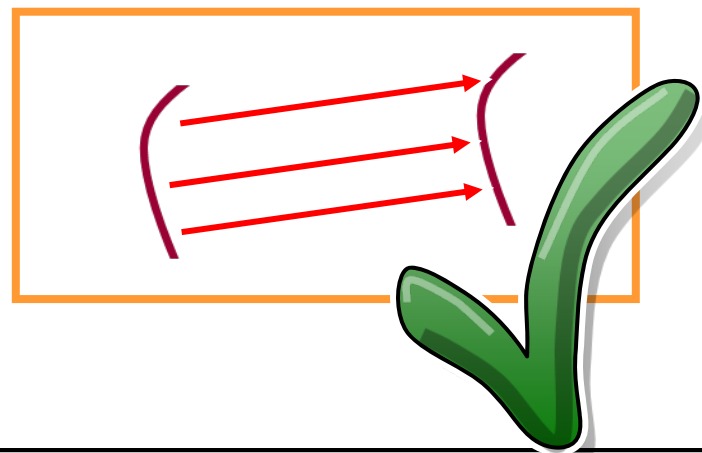
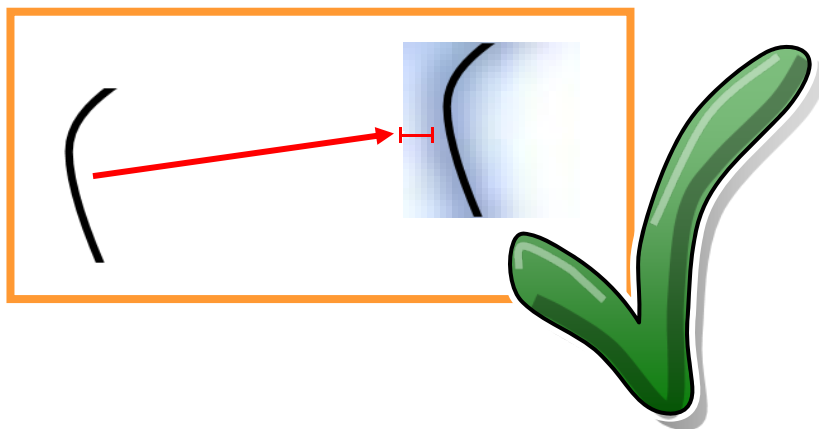
Deformable ICP

Deformable ICP

How to build a deformable ICP algorithm

- Pick a surface distance measure
- Pick an deformation model / regularizer

$$E(f) = E^{(match)}(f) + E^{(regularizer)}(f)$$



Deformable ICP

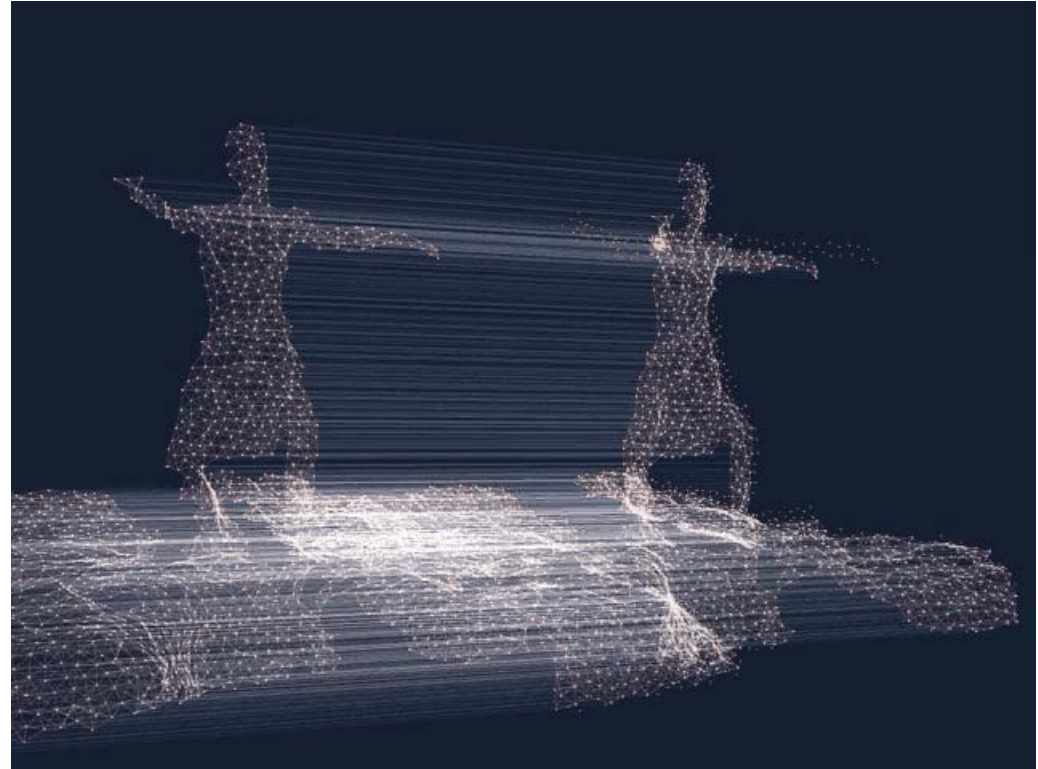
How to build a deformable ICP algorithm

- Pick a surface distance measure
- Pick an deformation model / regularizer
- Initialize $f(S_1)$ with S_2
- Pick a non-linear optimization algorithm
 - Gradient decent (easy, but bad performance)
 - Preconditioned conjugate gradients (better)
 - Newton or Gauss Newton (recommended, but more work)
 - Always use analytical derivatives!
- Run optimization

Example

Example

- Elastic model
- Local rigid coordinate frames
- Align $A \rightarrow B$, $B \rightarrow A$



Robust Local Registration



ETH Zurich / EPFL

Robust **Local** Registration

Hao Li



ETH Zurich / EPFL



Pairwise Non-Rigid Registration

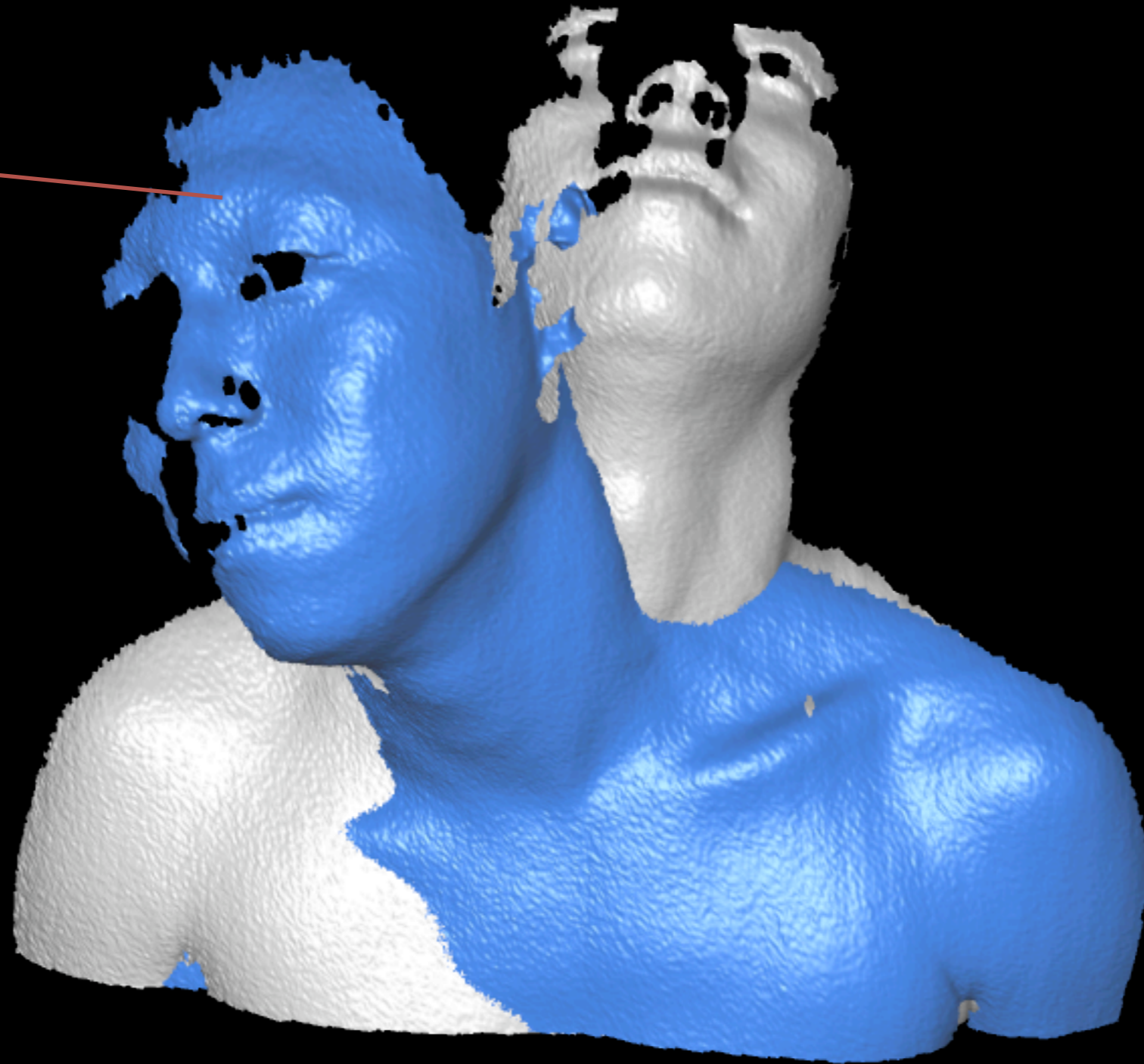


Initial Alignment



Initial Alignment

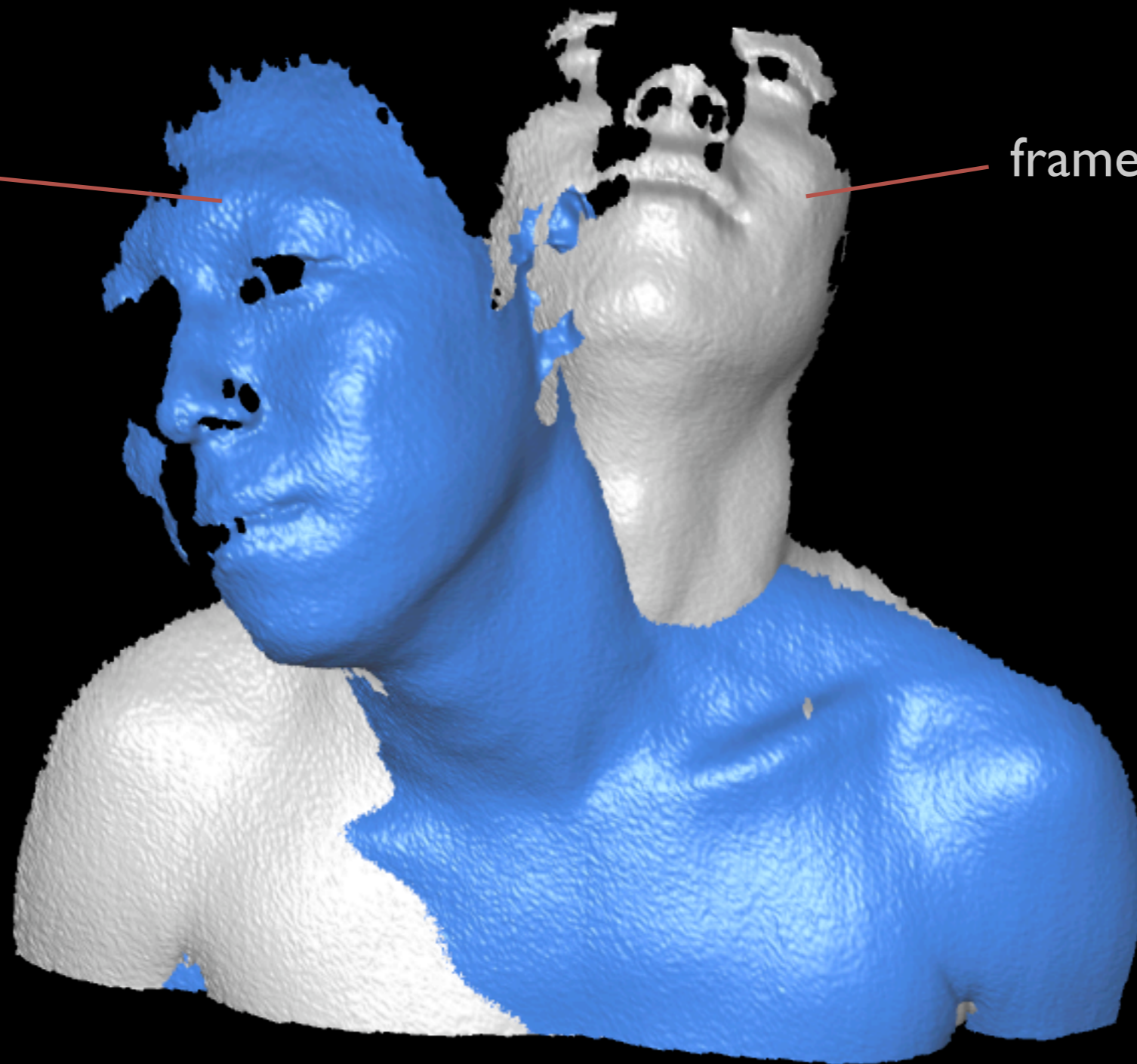
frame n



Initial Alignment

frame n

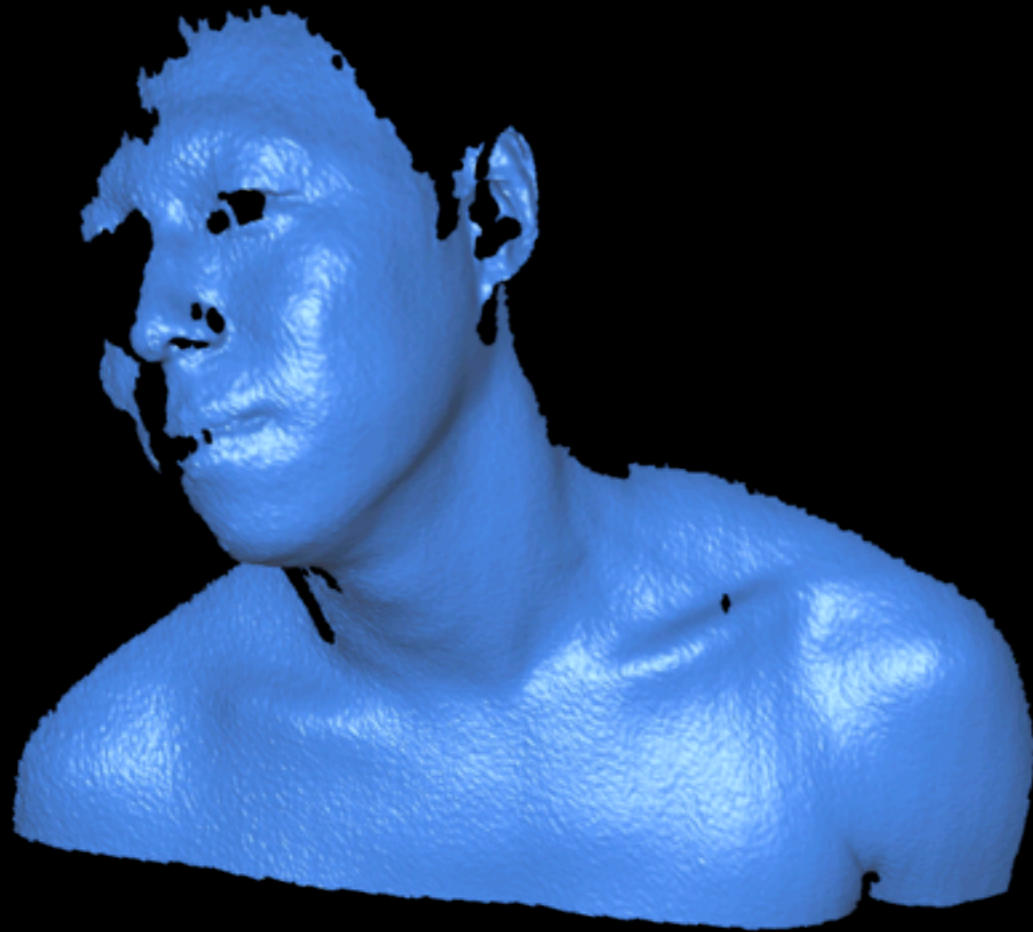
frame $n+1$



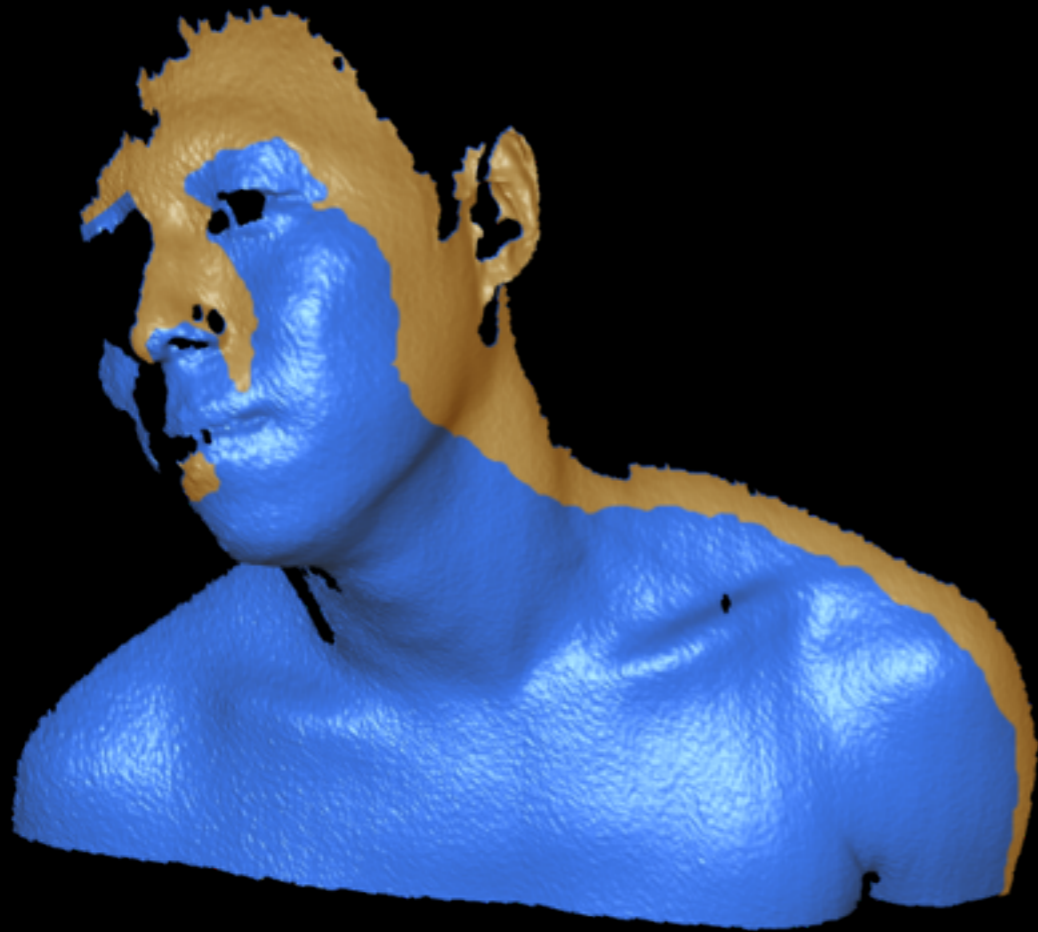
Source and Target



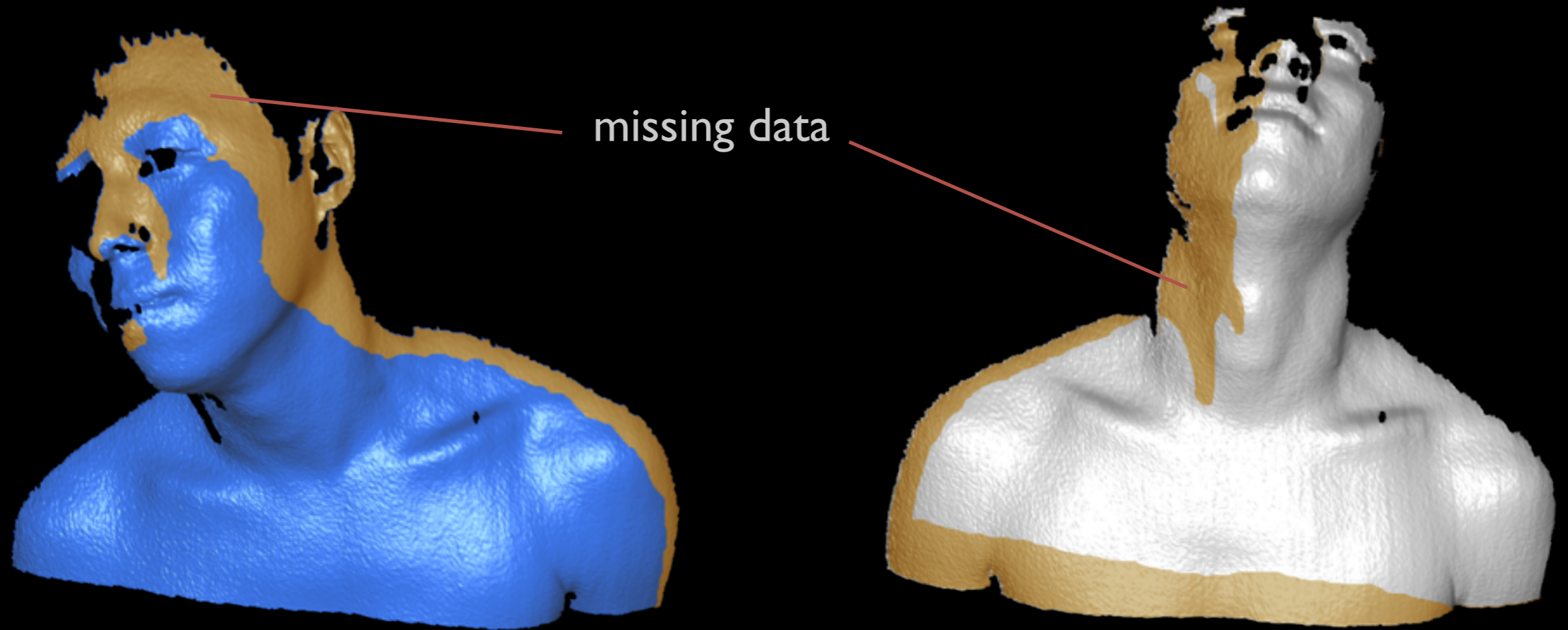
Source and Target



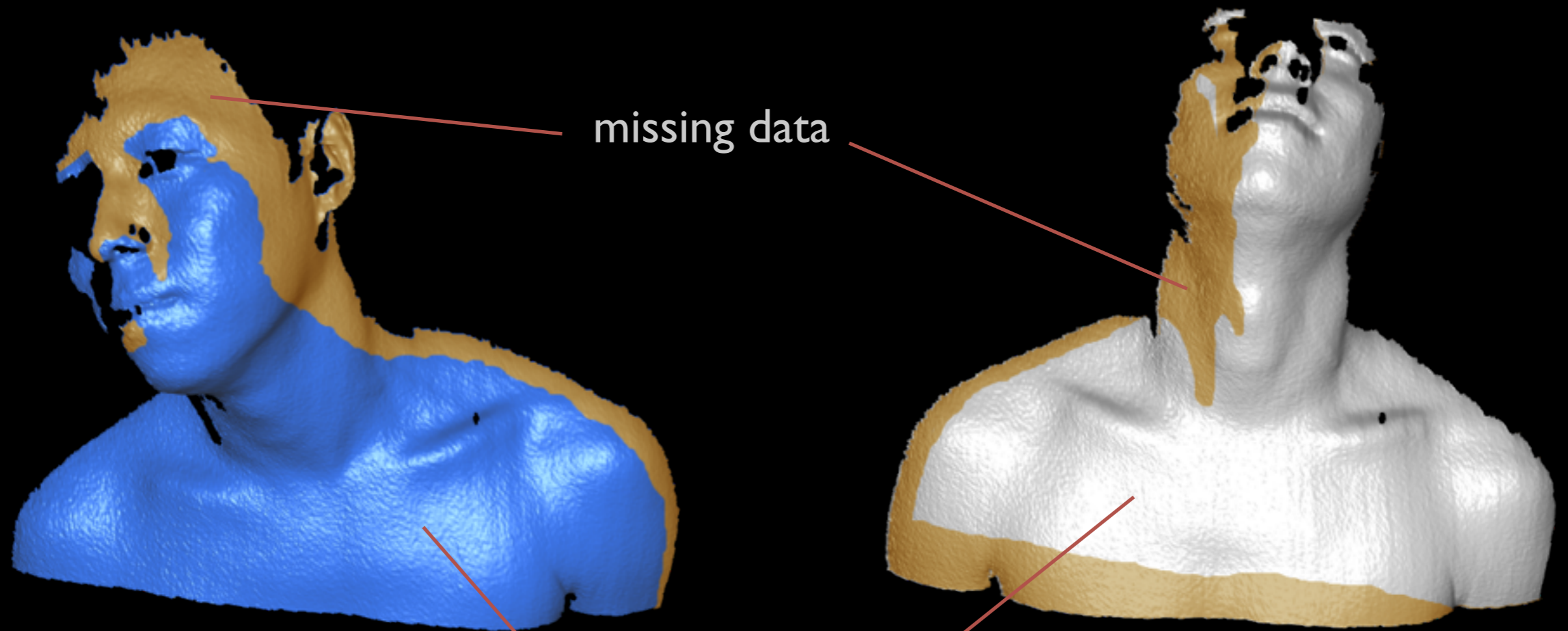
Deformation and Occlusion



Deformation and Occlusion



Deformation and Occlusion

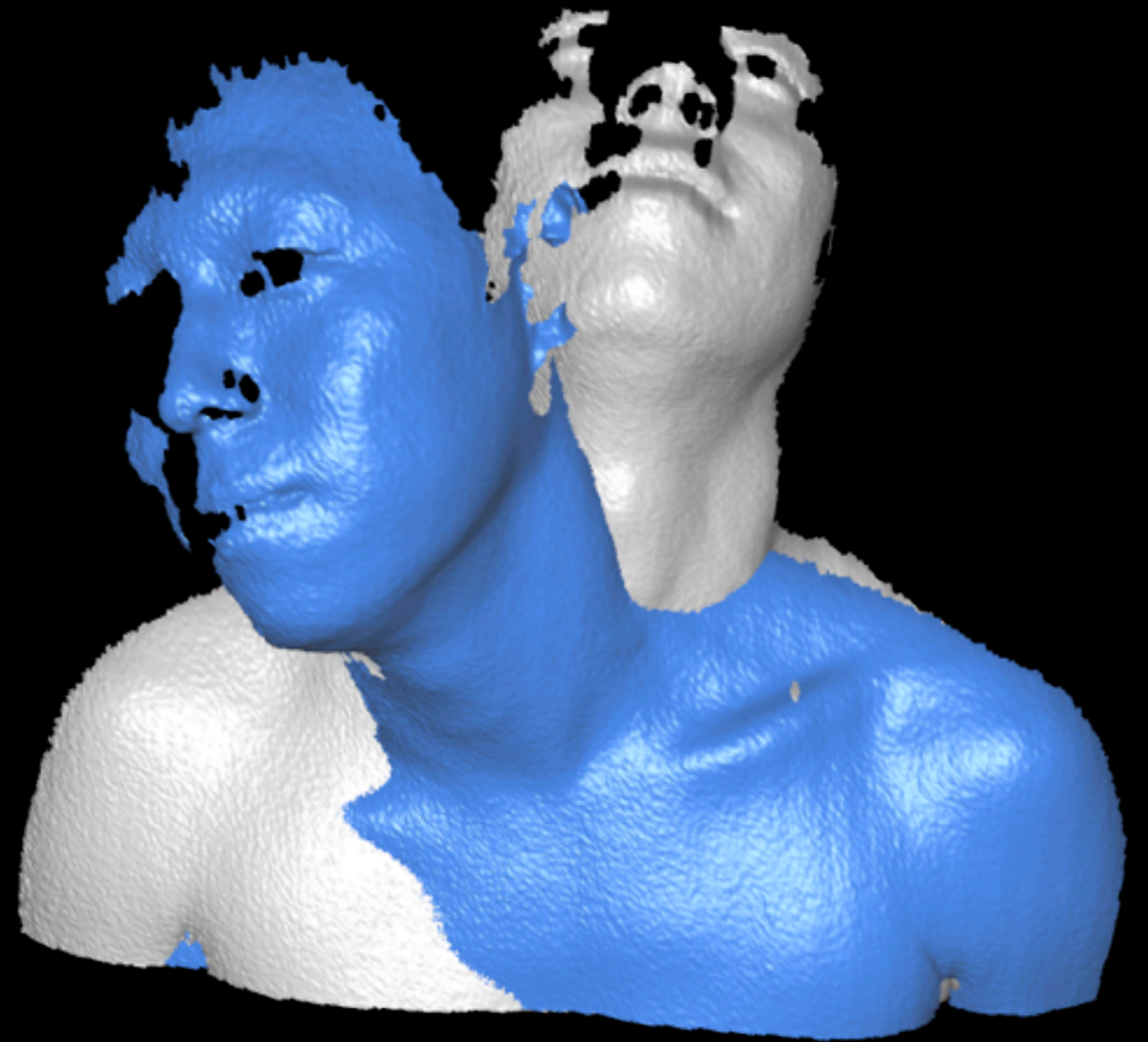


missing data

overlapping regions



No **Explicit** Prior Knowledge



No **Explicit** Prior Knowledge

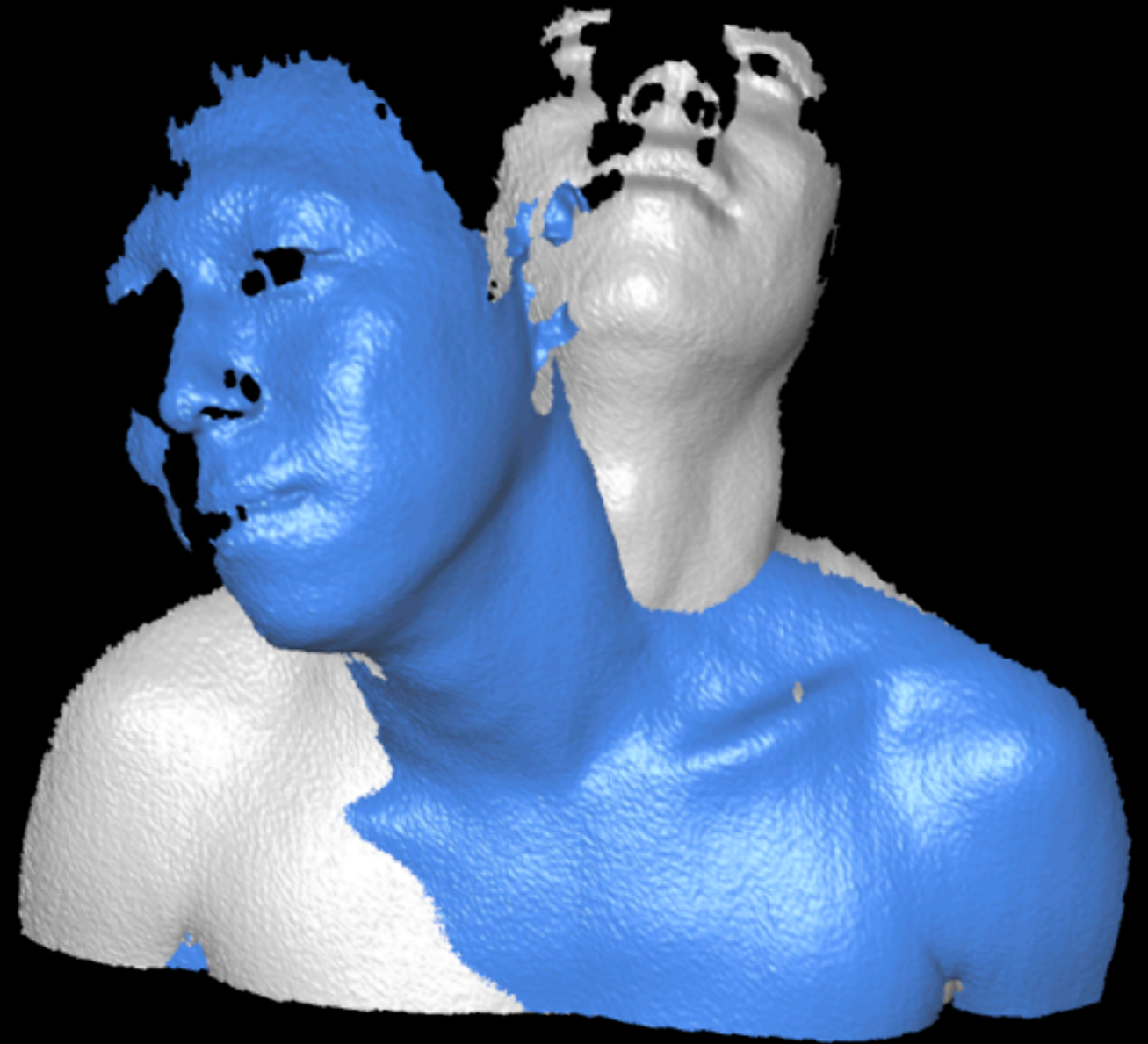
no **knowledge** about



No **Explicit** Prior Knowledge

no knowledge about

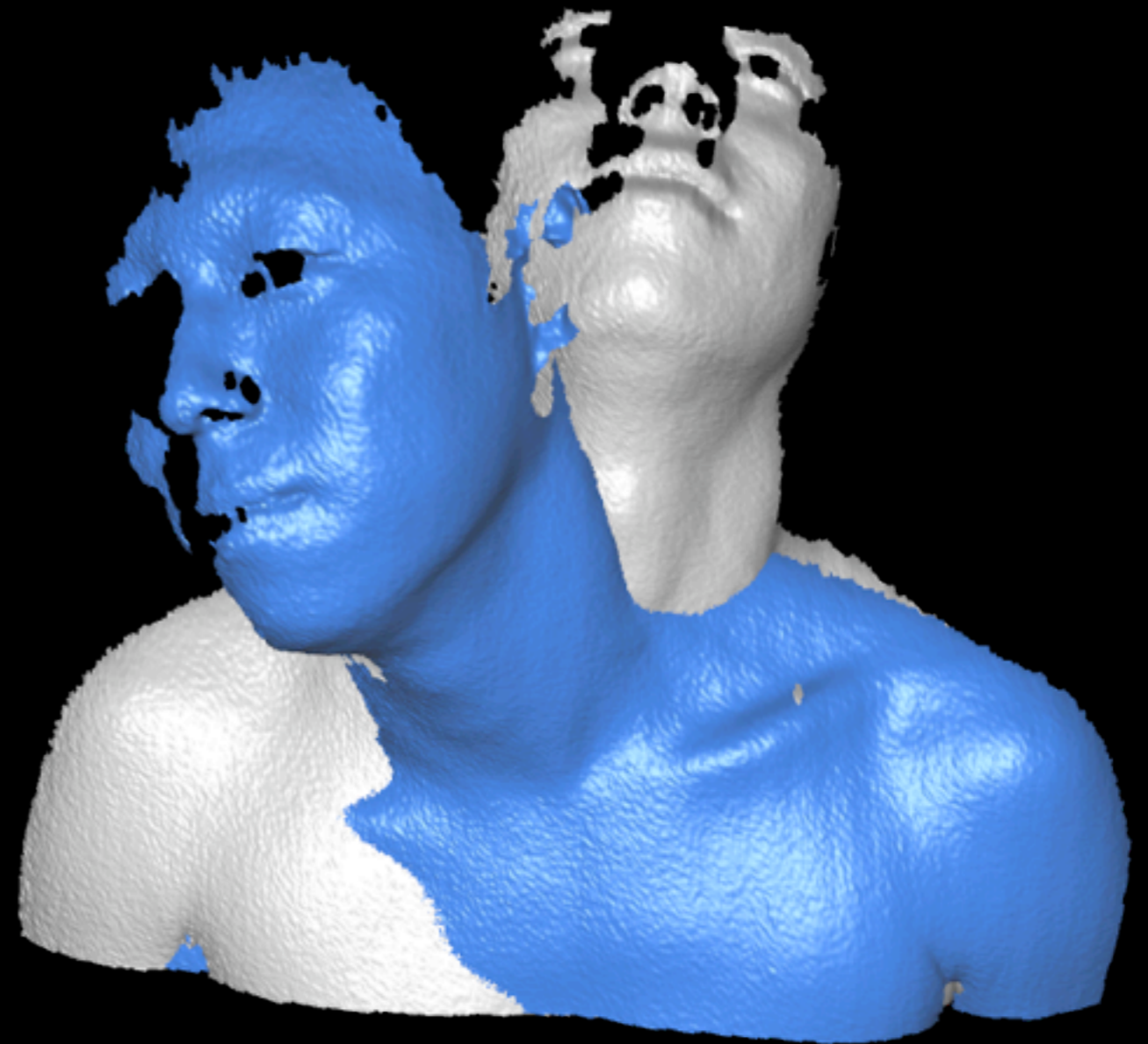
- full 3-D model



No **Explicit** Prior Knowledge

no knowledge about

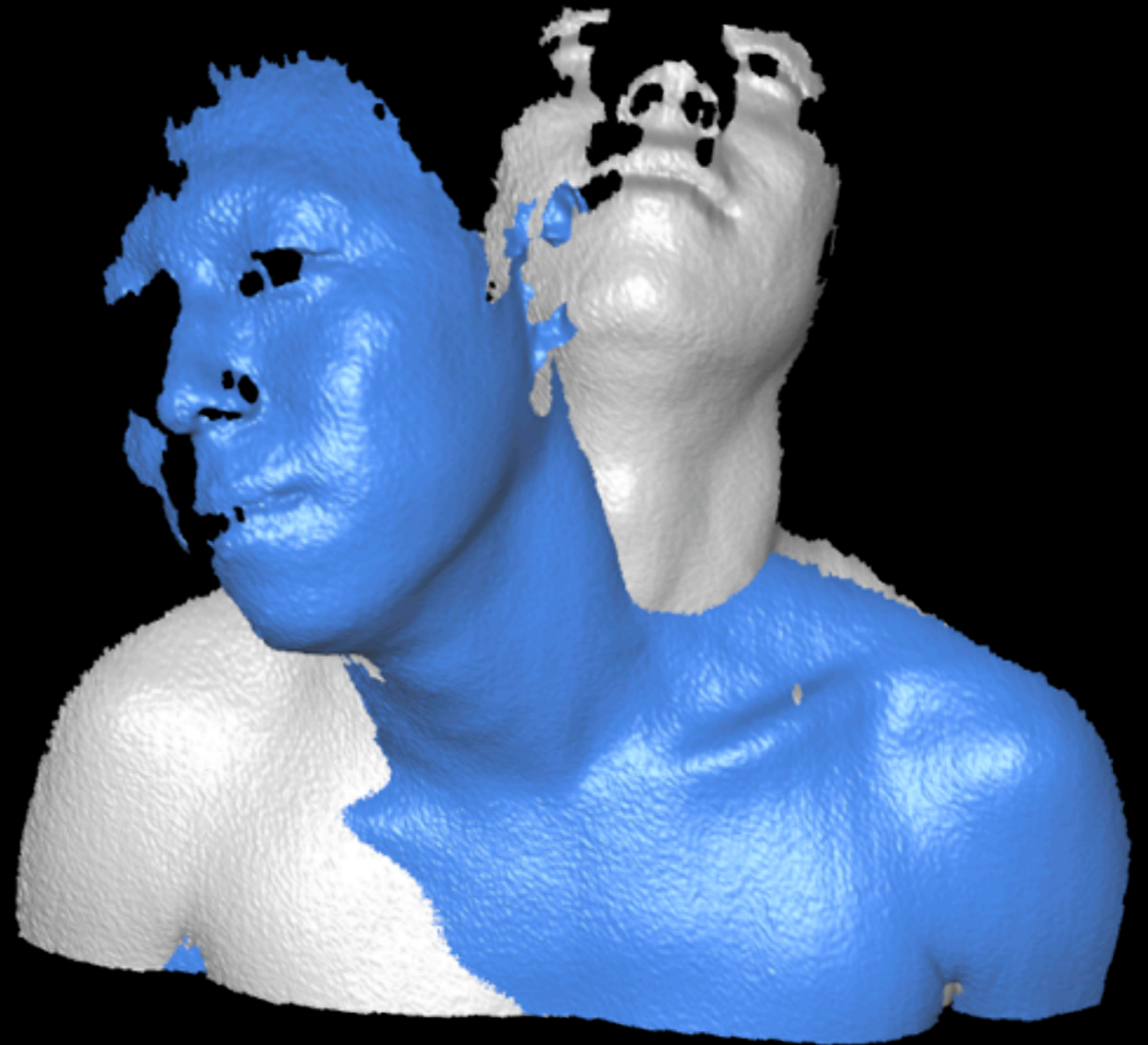
- full 3-D model
- correspondences



No **Explicit** Prior Knowledge

no knowledge about

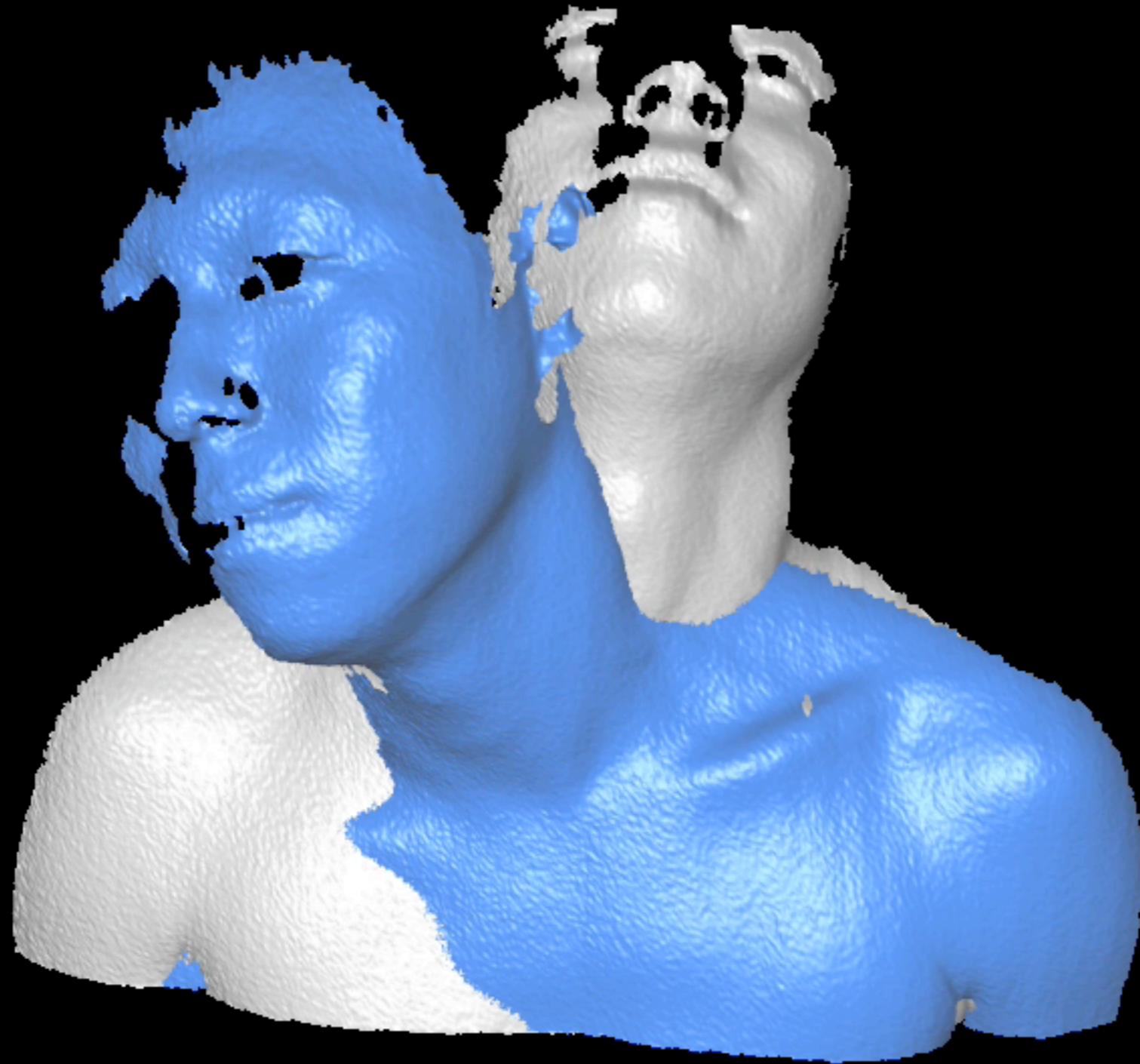
- full 3-D model
- correspondences
- regions of overlap



Goal: Automatic Local Registration



Goal: Automatic Local Registration



Ingredients for Non-Rigid Registration



Ingredients for Non-Rigid Registration

source

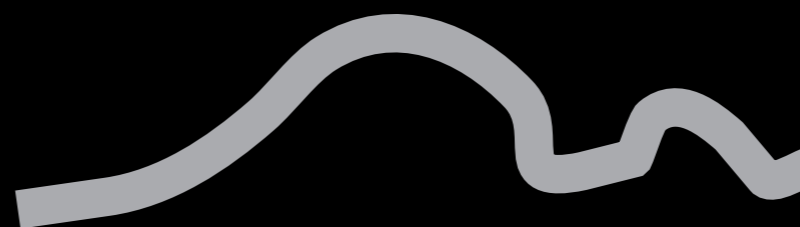
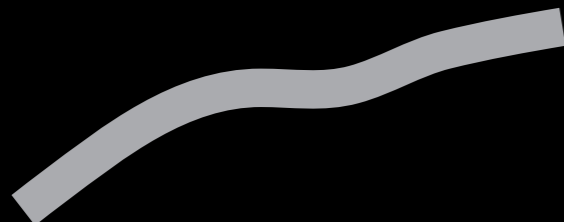


Ingredients for Non-Rigid Registration

source



target

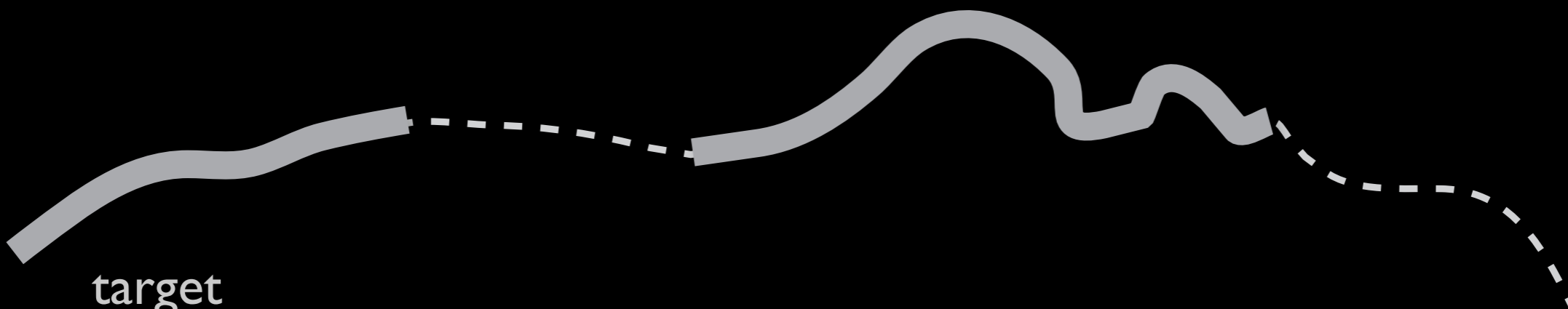


Ingredients for Non-Rigid Registration

source



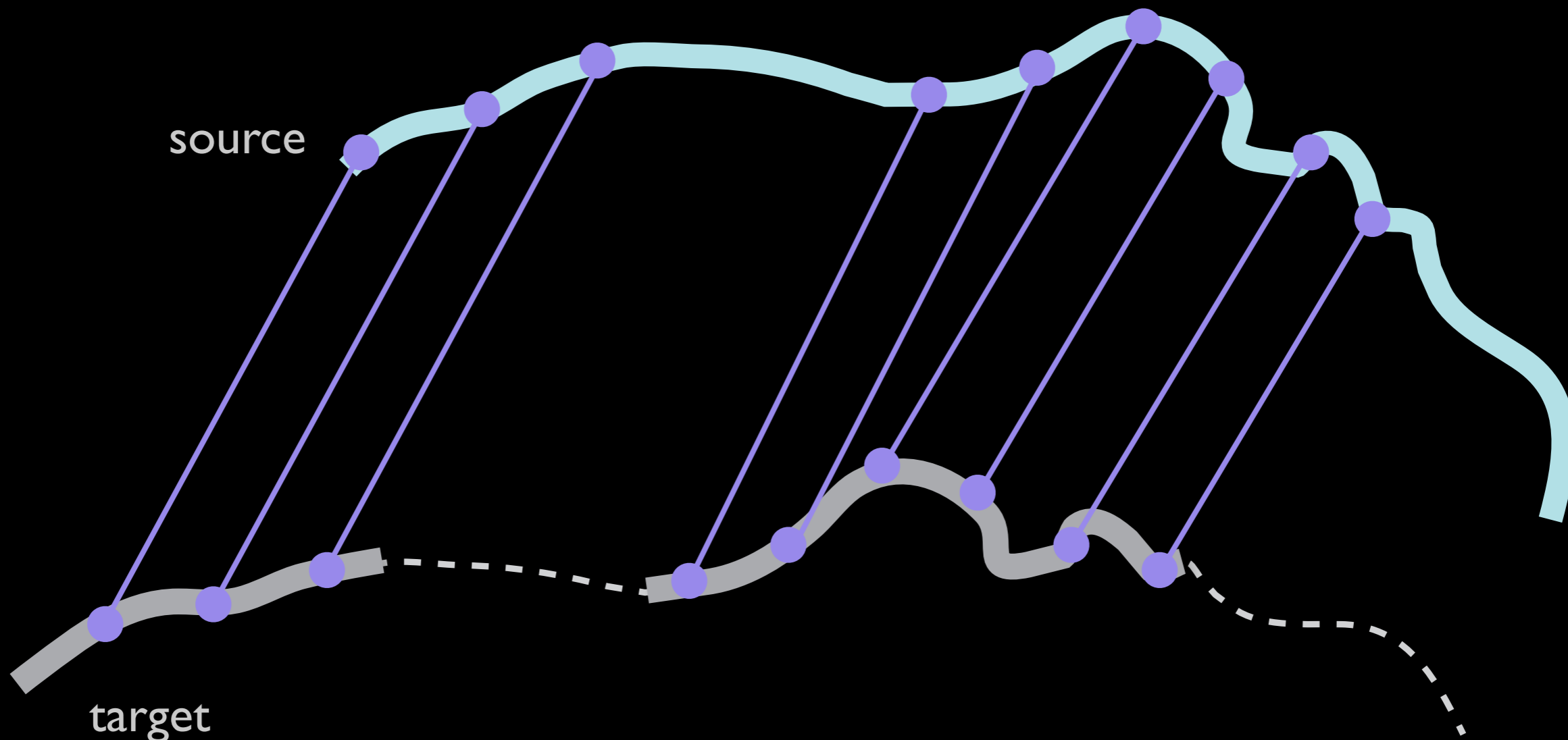
target



partial overlap



Ingredients for Non-Rigid Registration

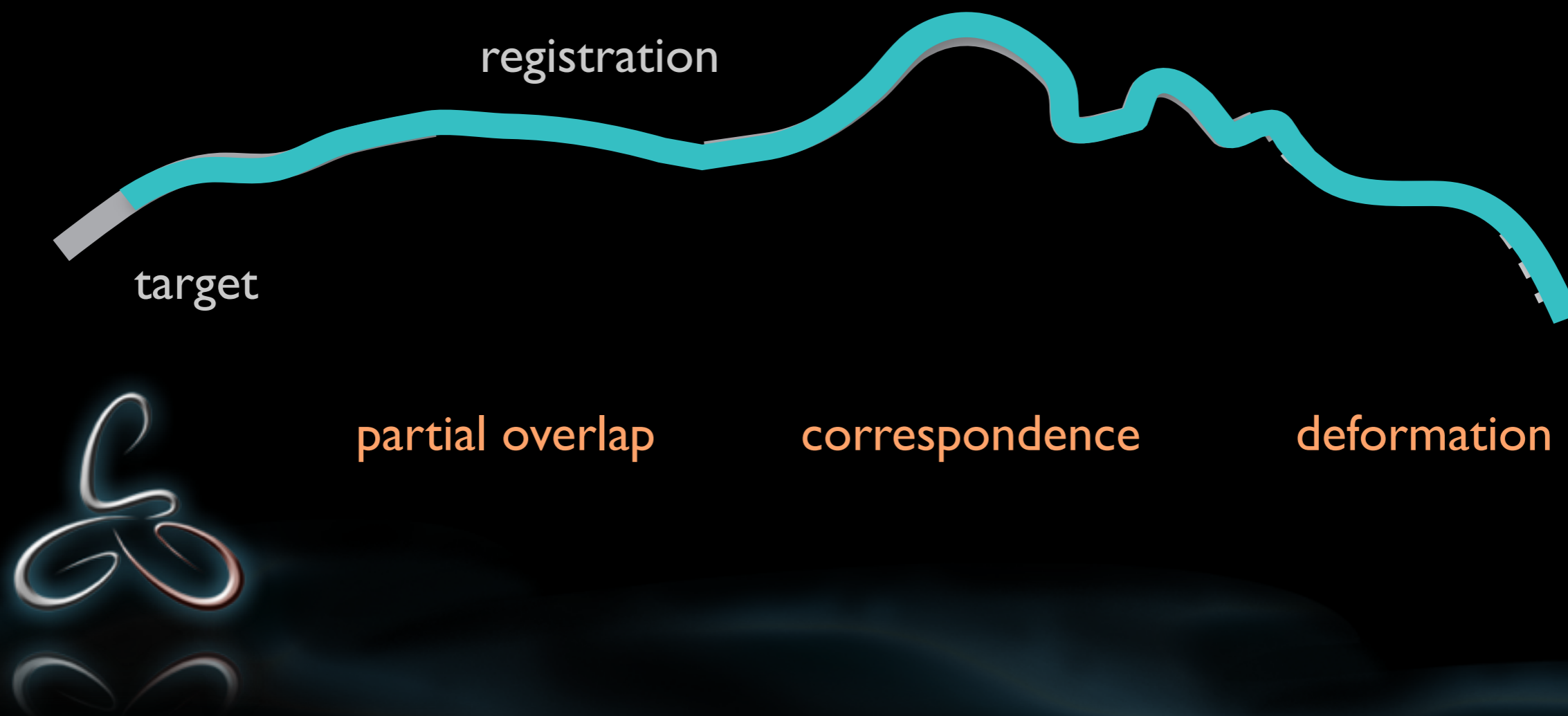


partial overlap

correspondence



Ingredients for Non-Rigid Registration



Chicken & Egg Dillema



Chicken & Egg Dillema

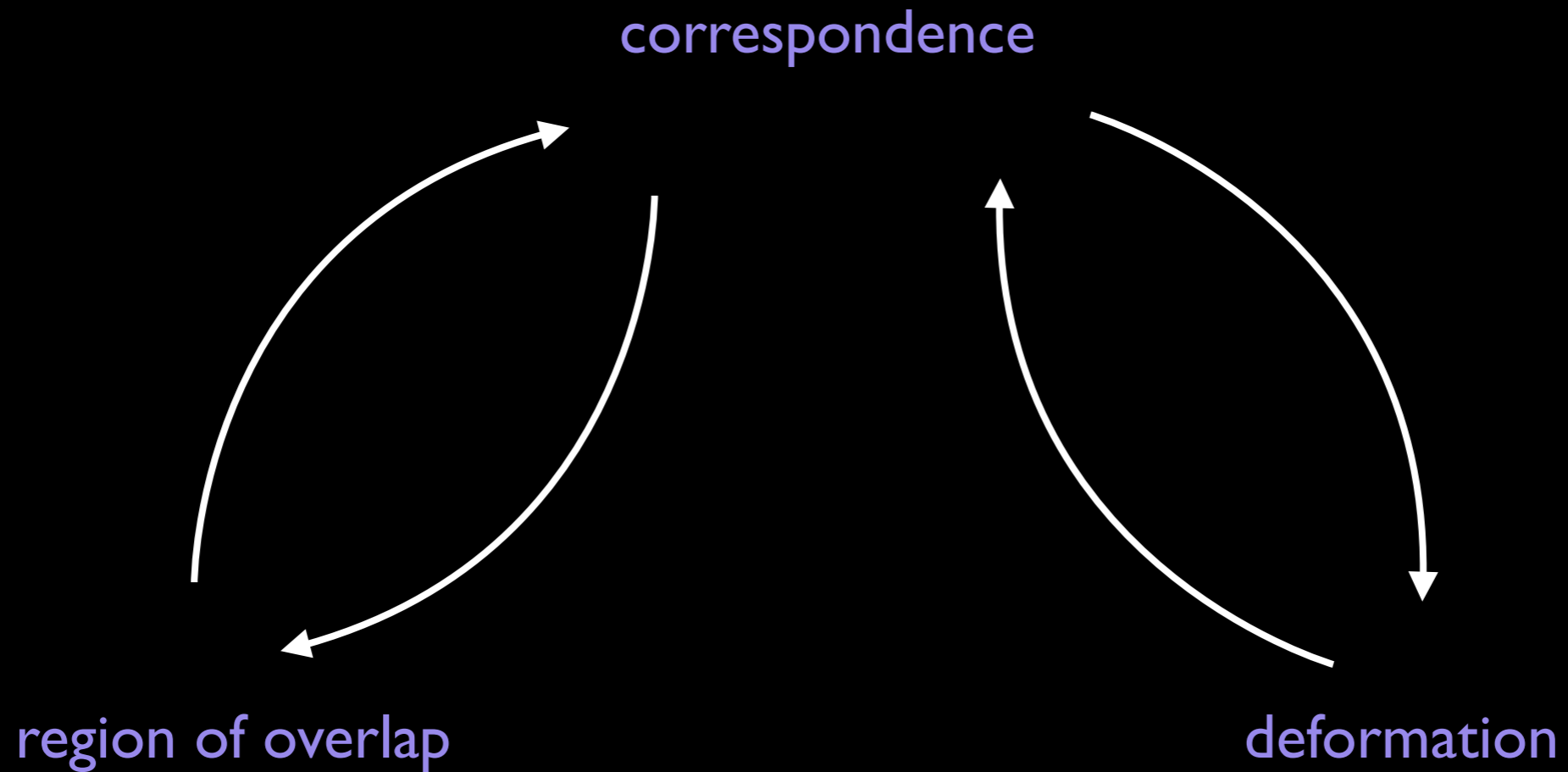
correspondence

region of overlap

deformation



Chicken & Egg Dillema



Chicken & Egg Dillema

correspondence

solve within a single
optimization problem

region of overlap

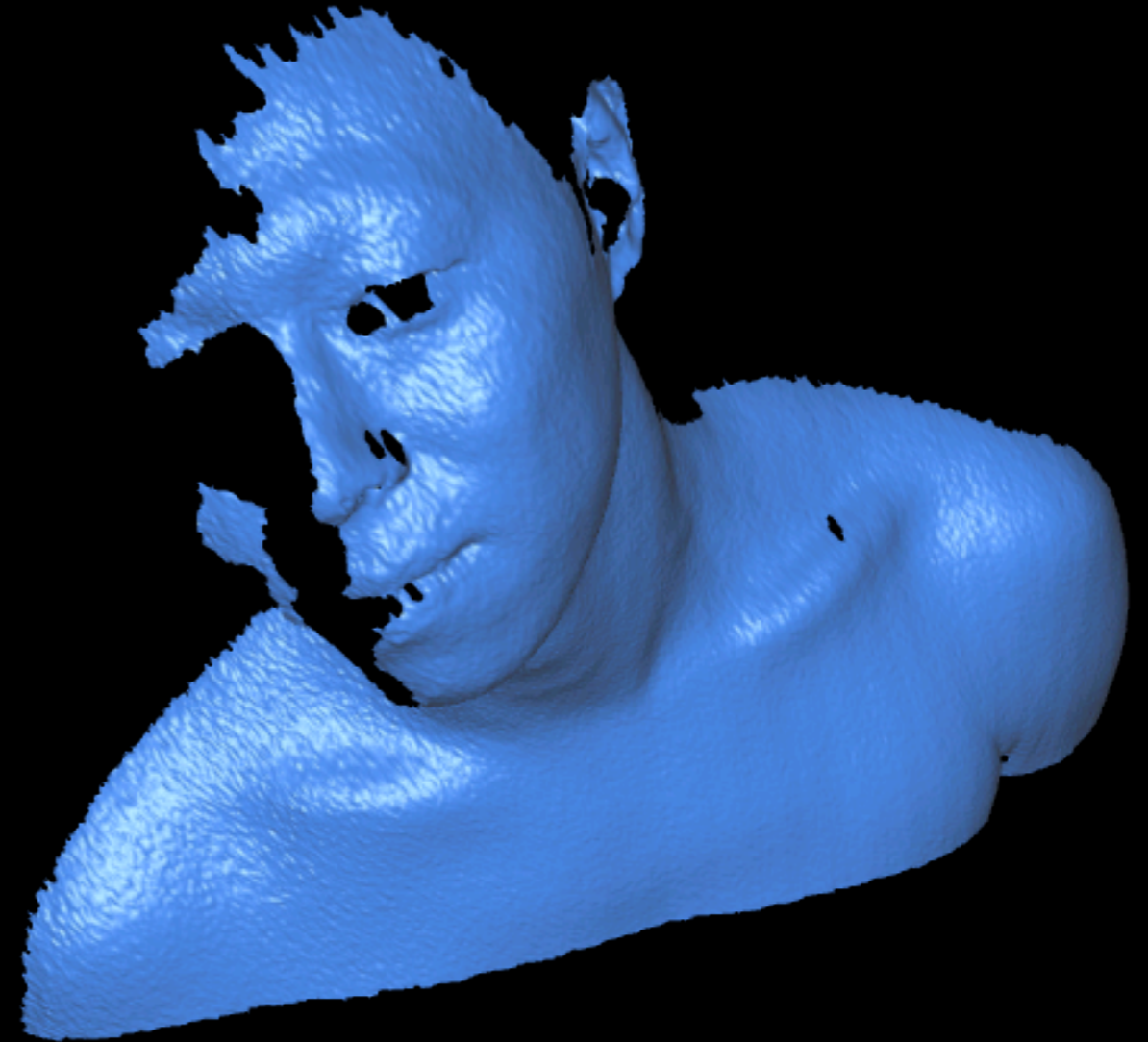
deformation



Embedded Deformation



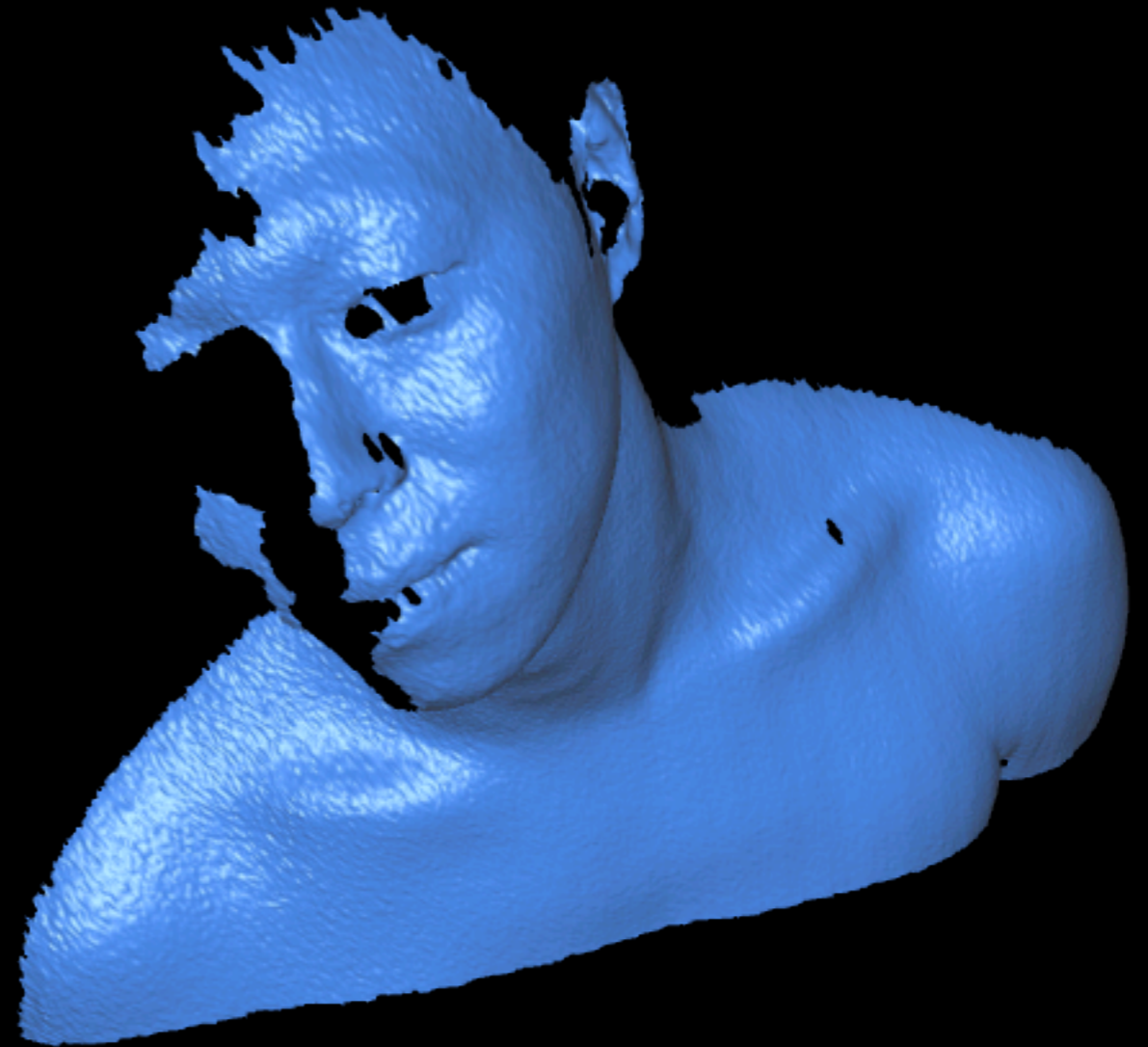
Deformation Model



Deformation Model

Embedded Deformation

[Sumner et al. '07]

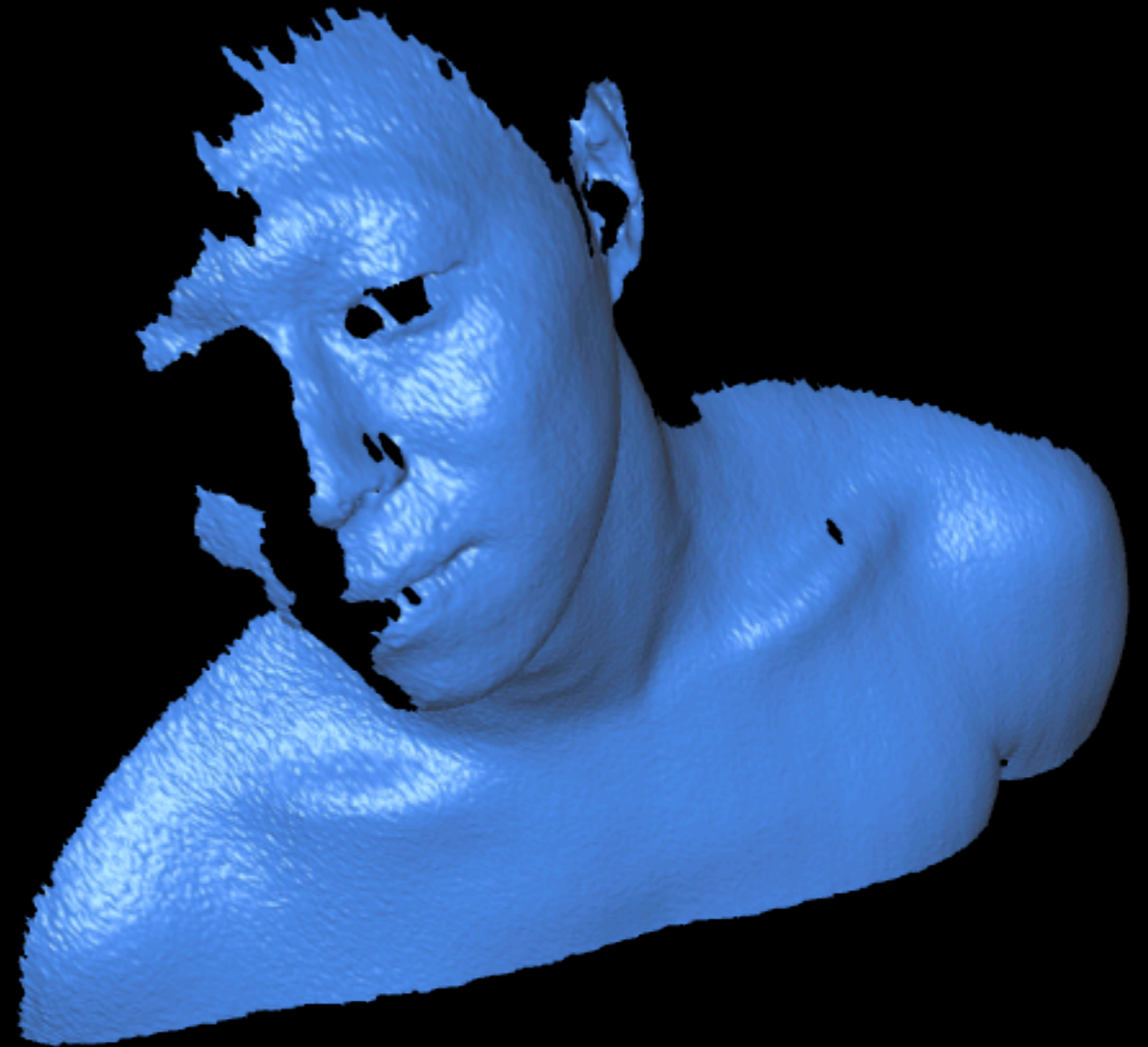


Deformation Model

Embedded Deformation

[Sumner et al. '07]

- efficiency

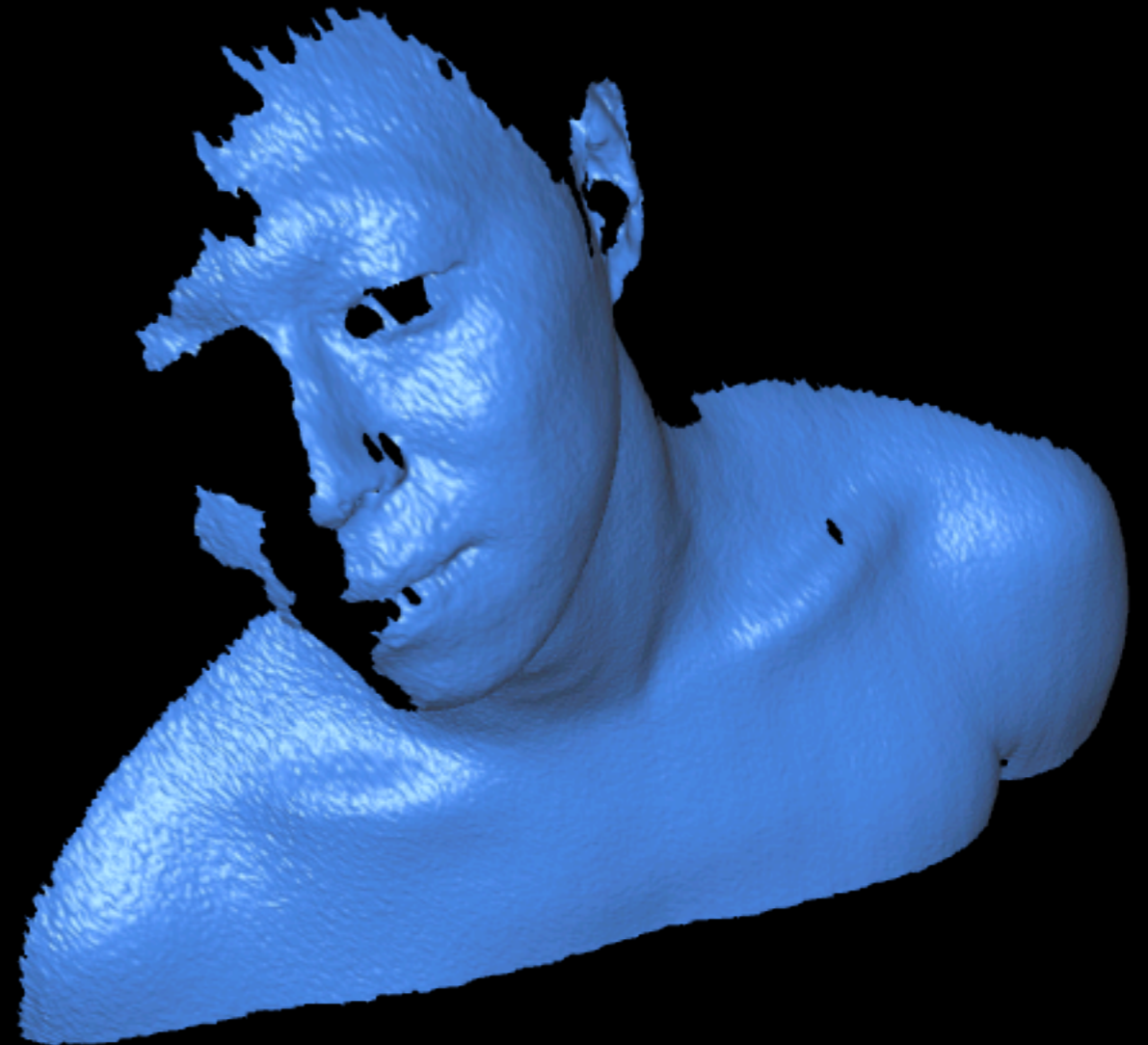


Deformation Model

Embedded Deformation

[Sumner et al. '07]

- efficiency
- generality

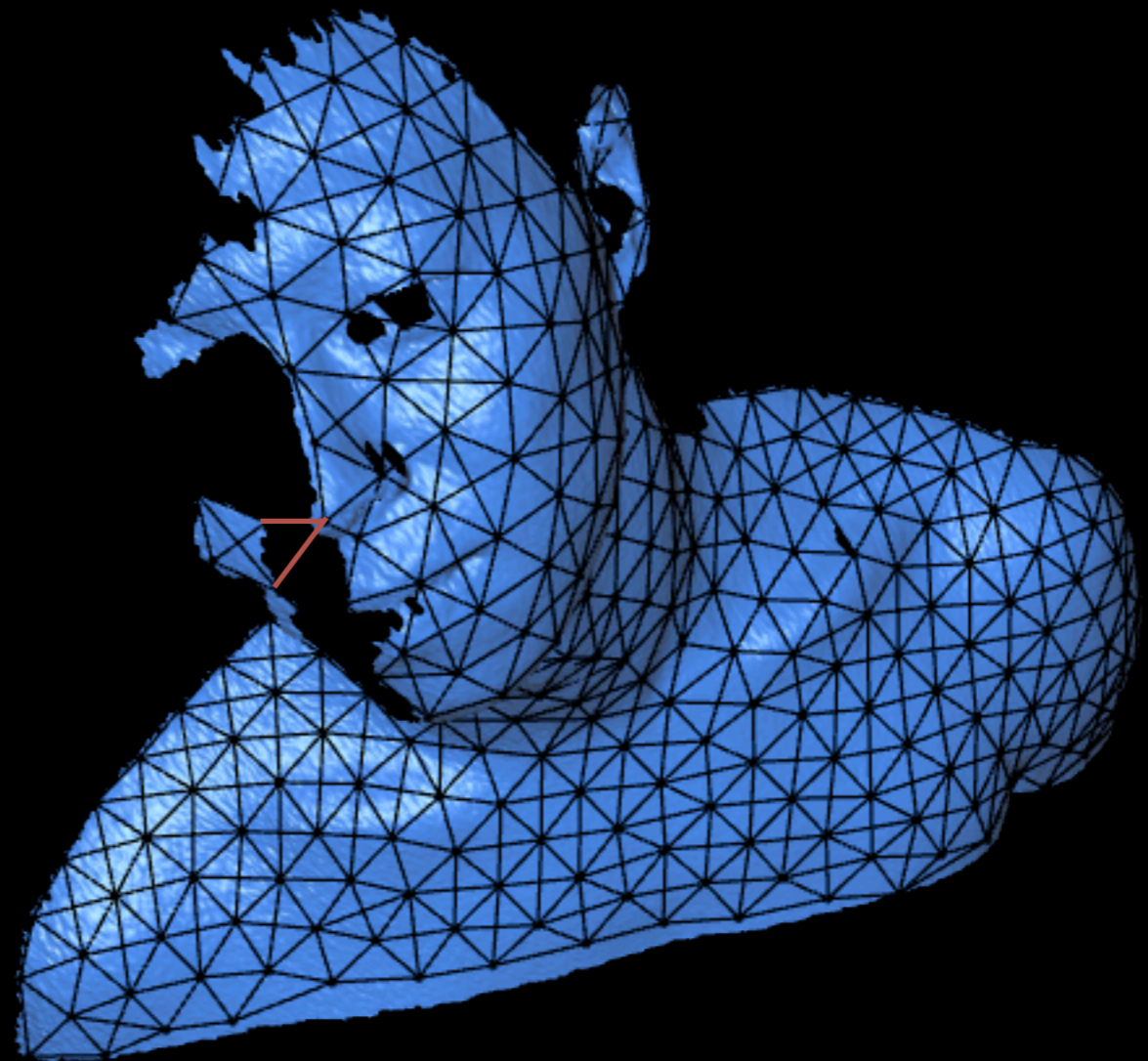


Deformation Model

Embedded Deformation

[Sumner et al. '07]

- efficiency
- generality

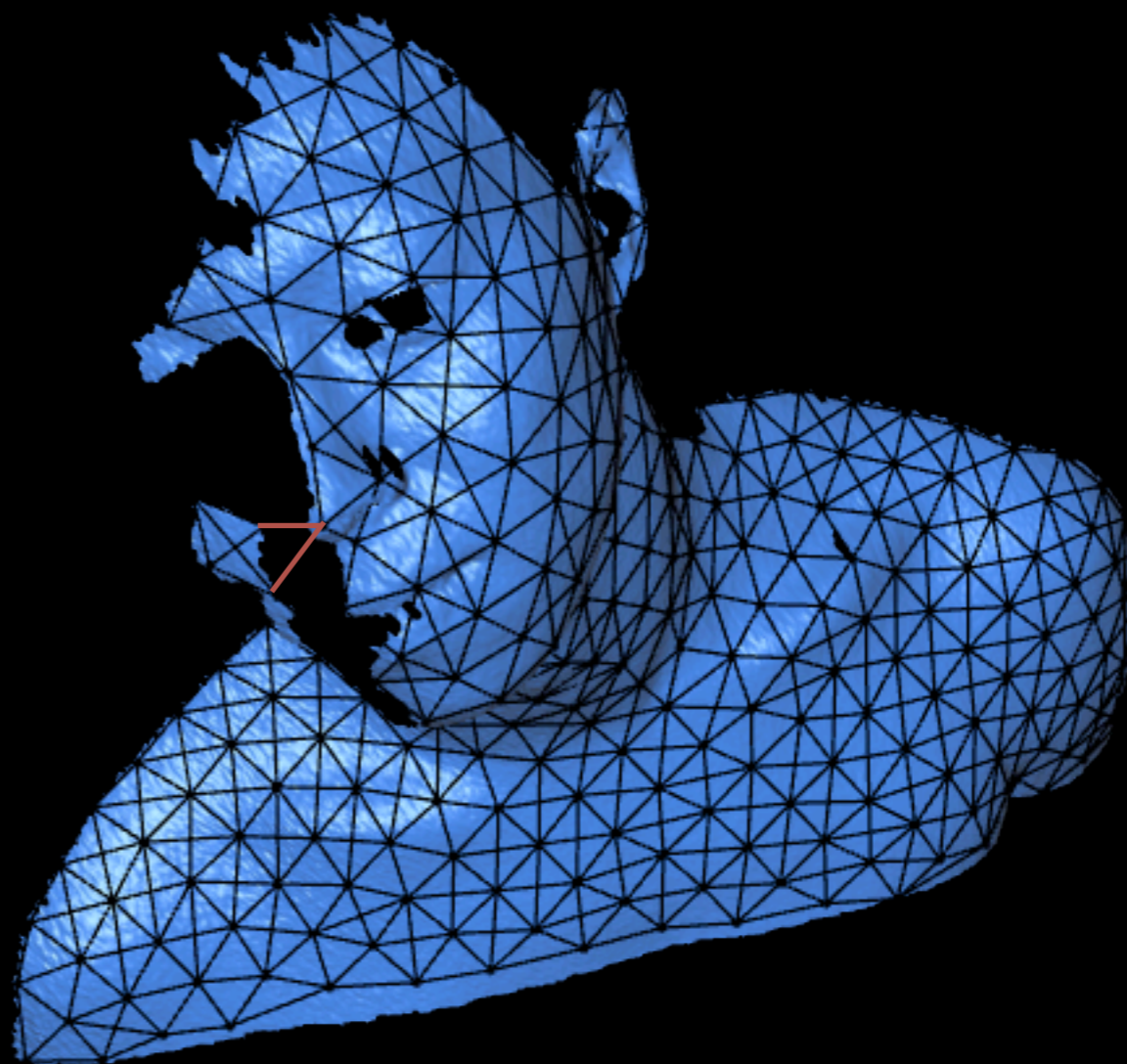


Deformation Model

Embedded Deformation

[Sumner et al. '07]

- efficiency
- generality
- natural deformation

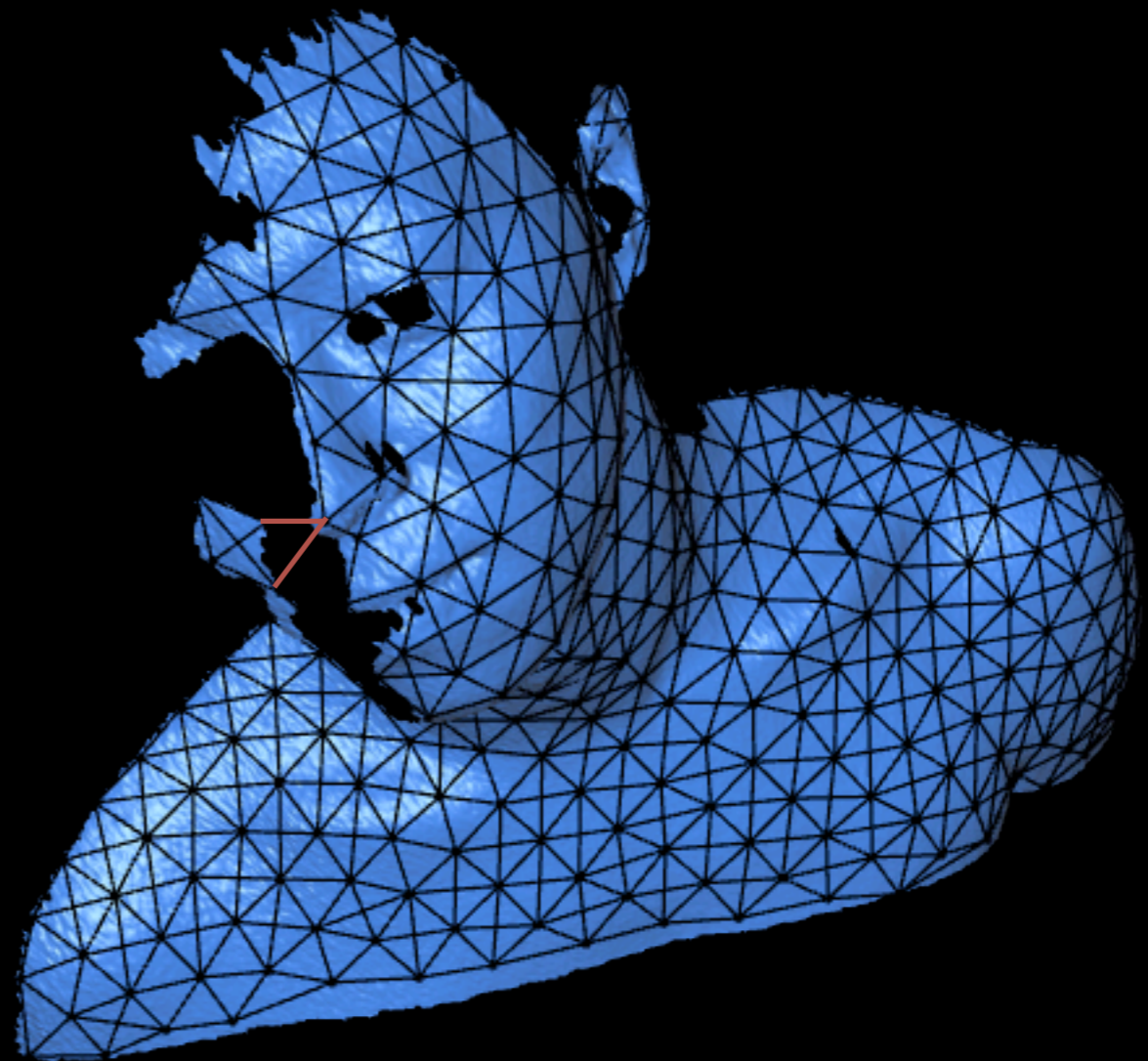


Deformation Model

Embedded Deformation

[Sumner et al. '07]

- efficiency
- generality
- natural deformation
- detail preservation

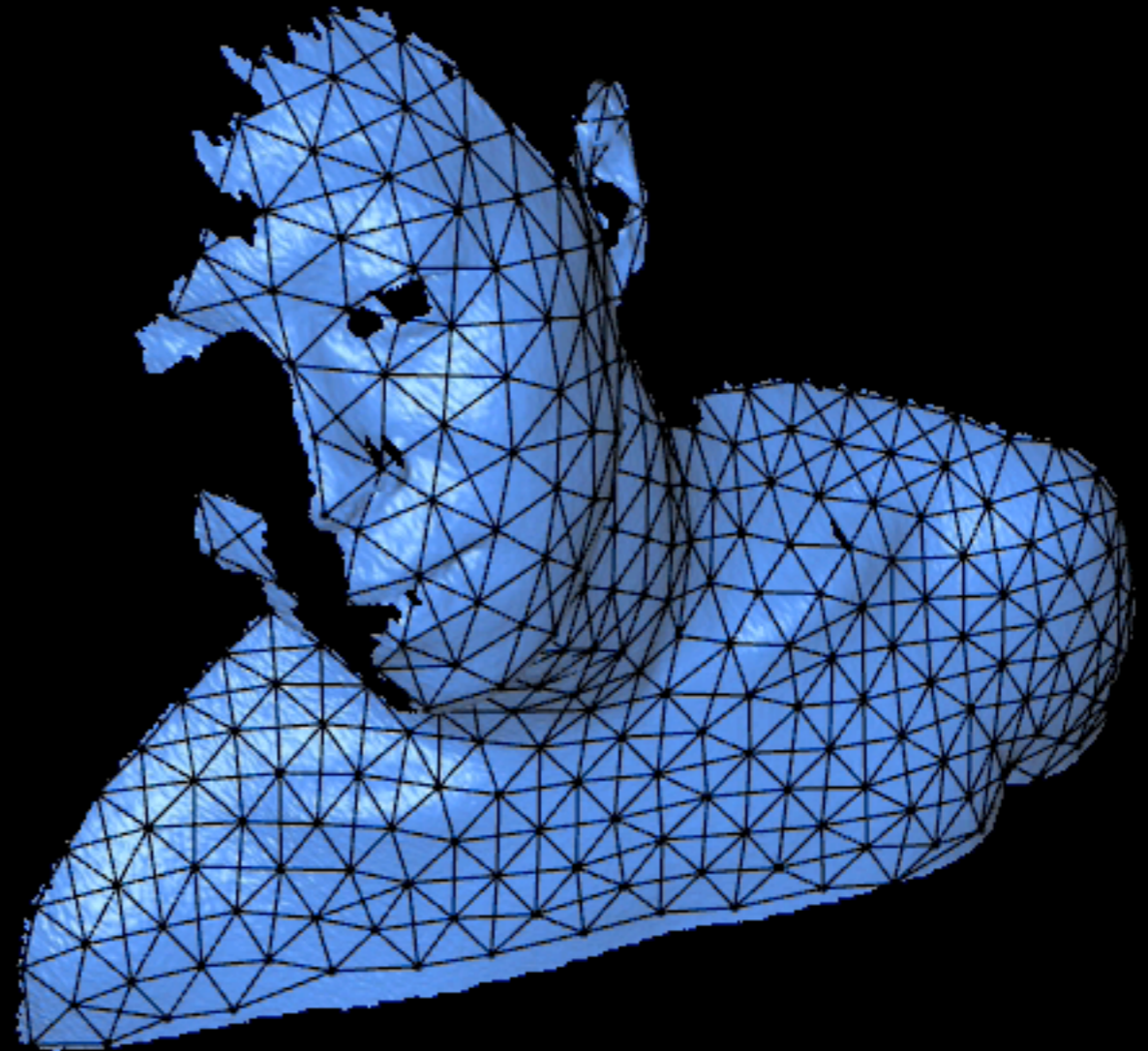


Deformation Model

Embedded Deformation

[Sumner et al. '07]

- efficiency
- generality
- natural deformation
- detail preservation

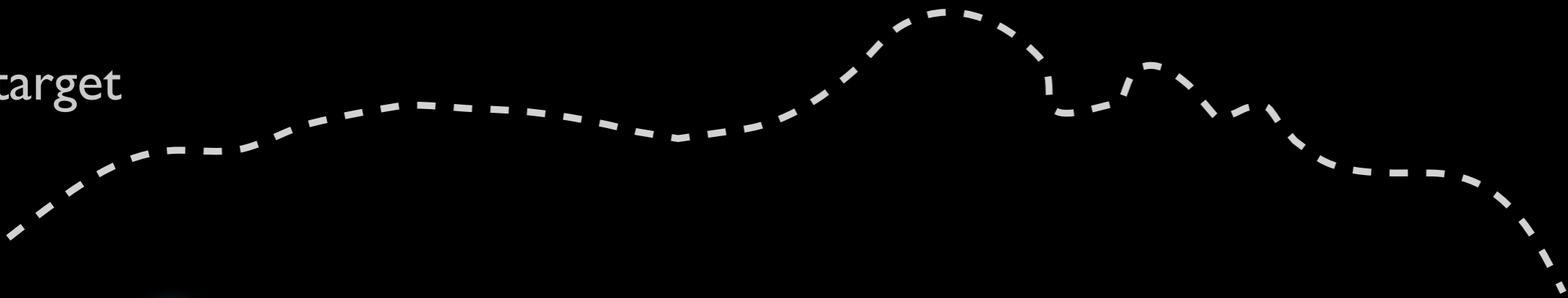


Deformation Model

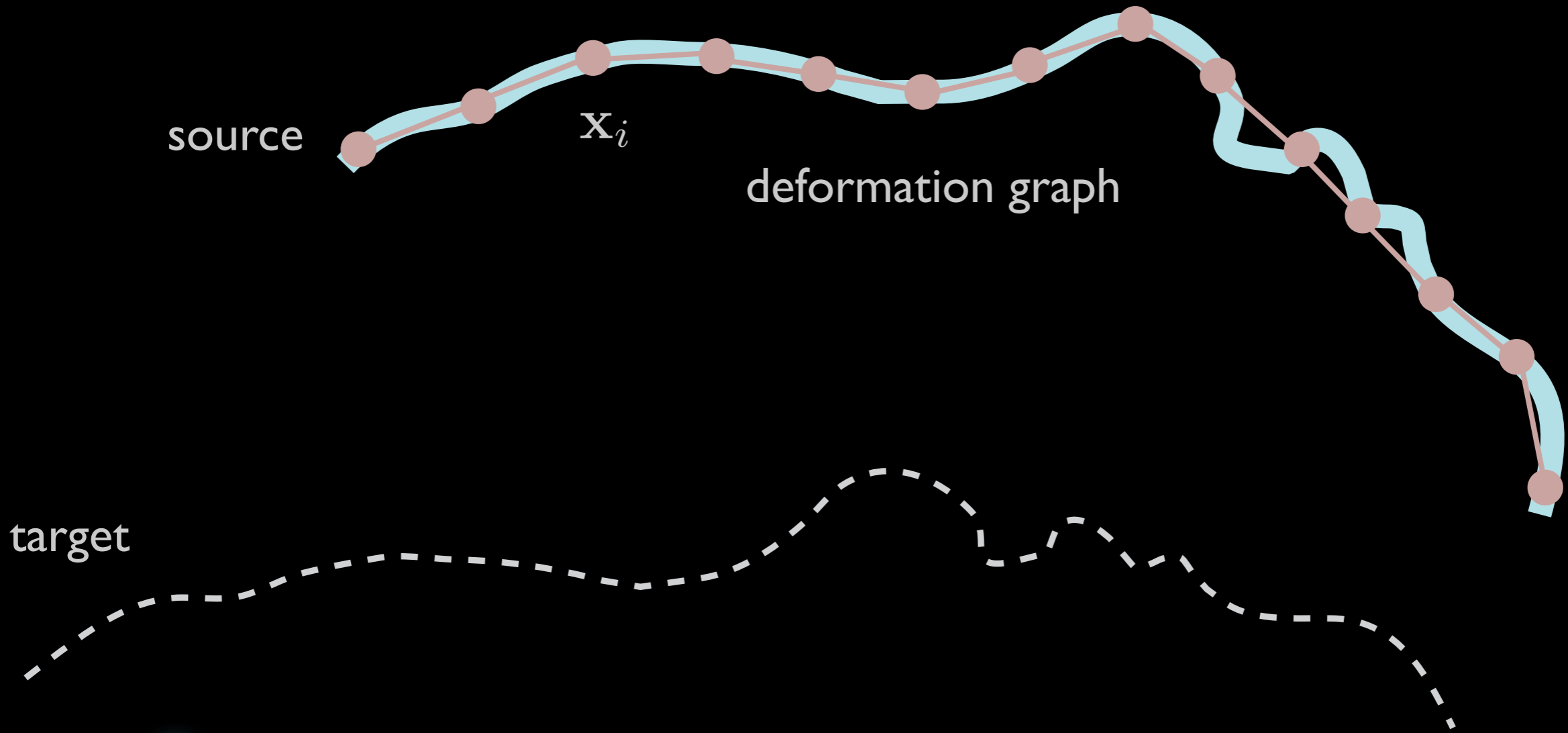
source



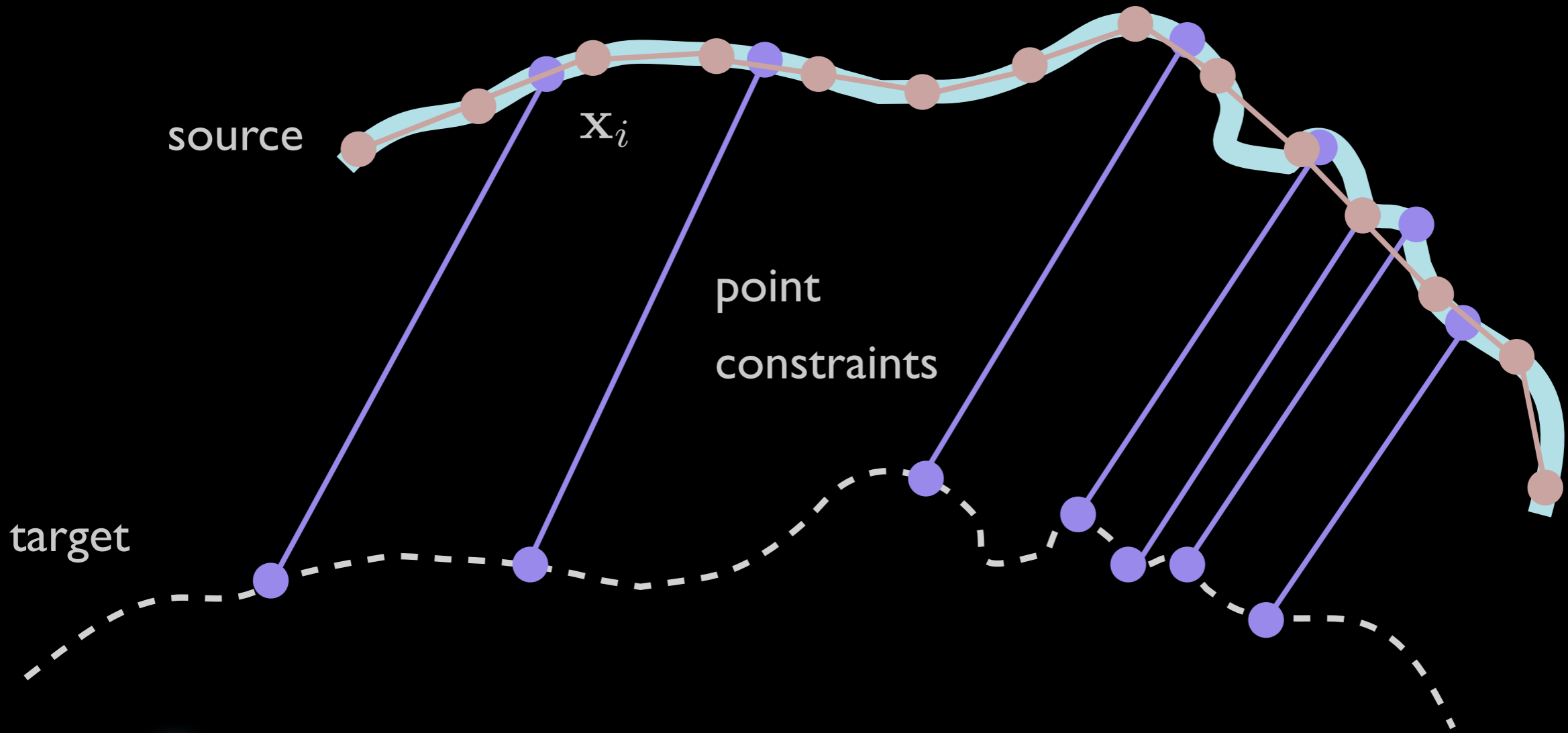
target



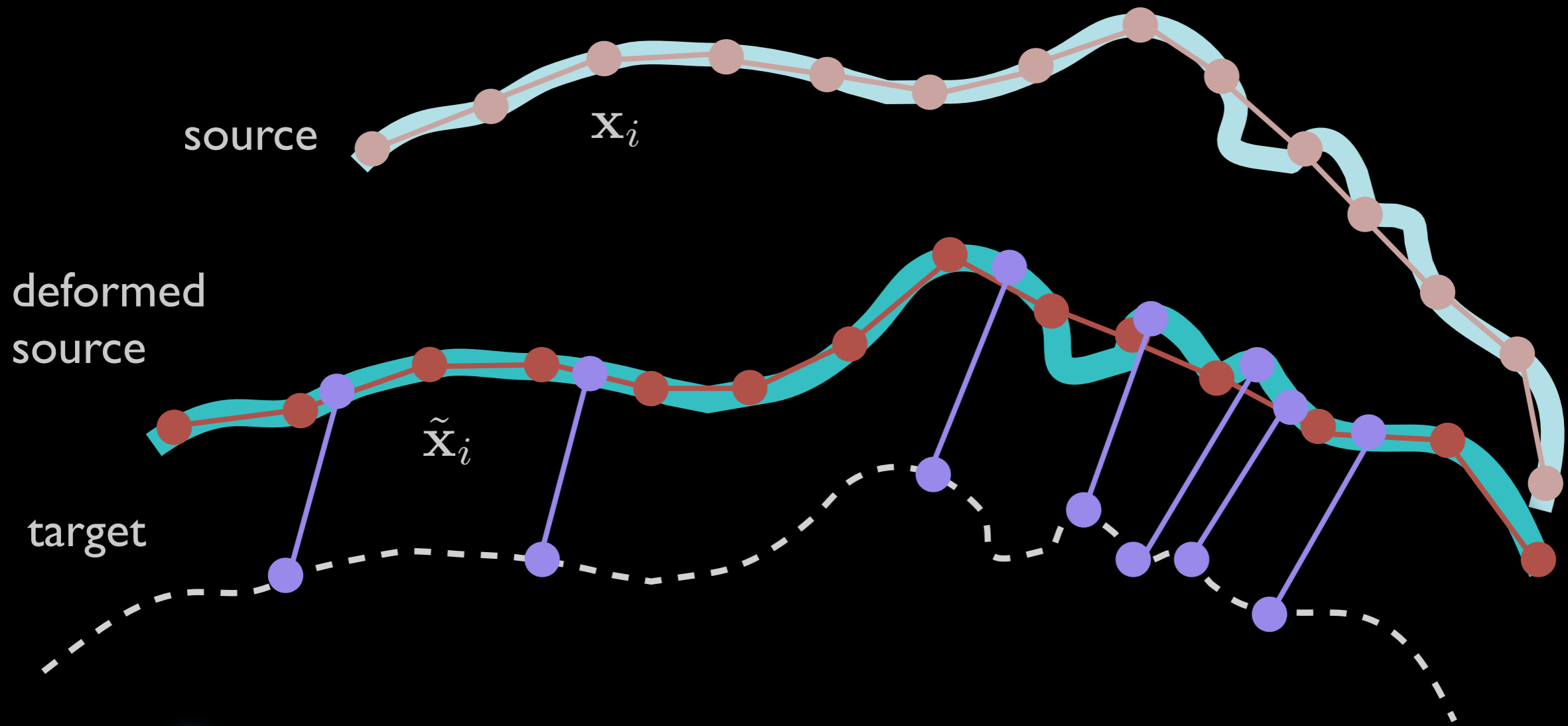
Embedded Deformation



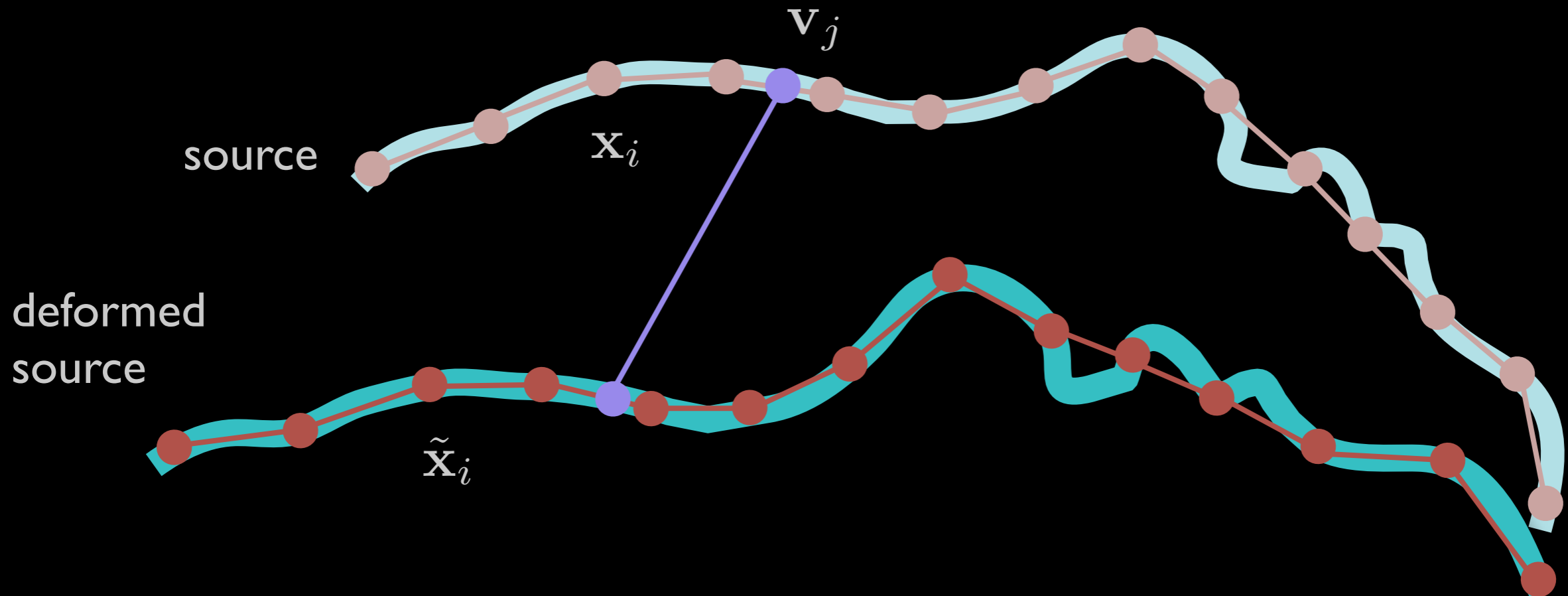
Embedded Deformation



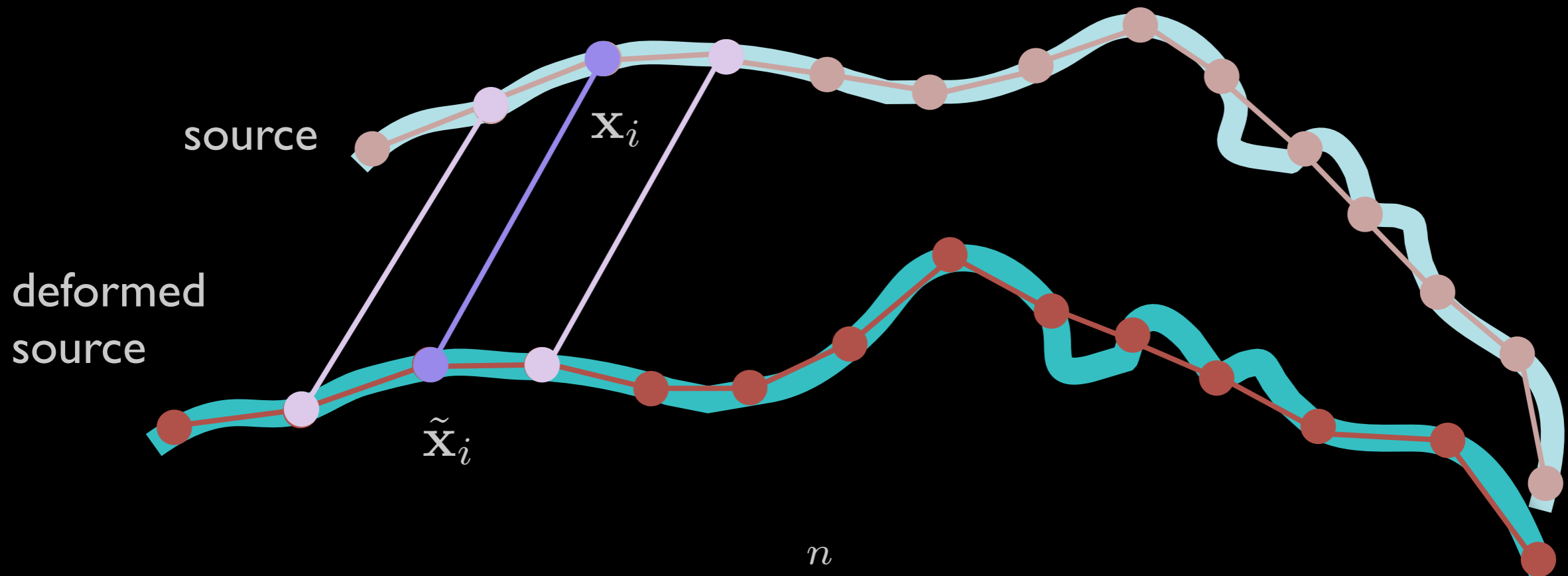
Embedded Deformation



Embedded Deformation



Embedded Deformation

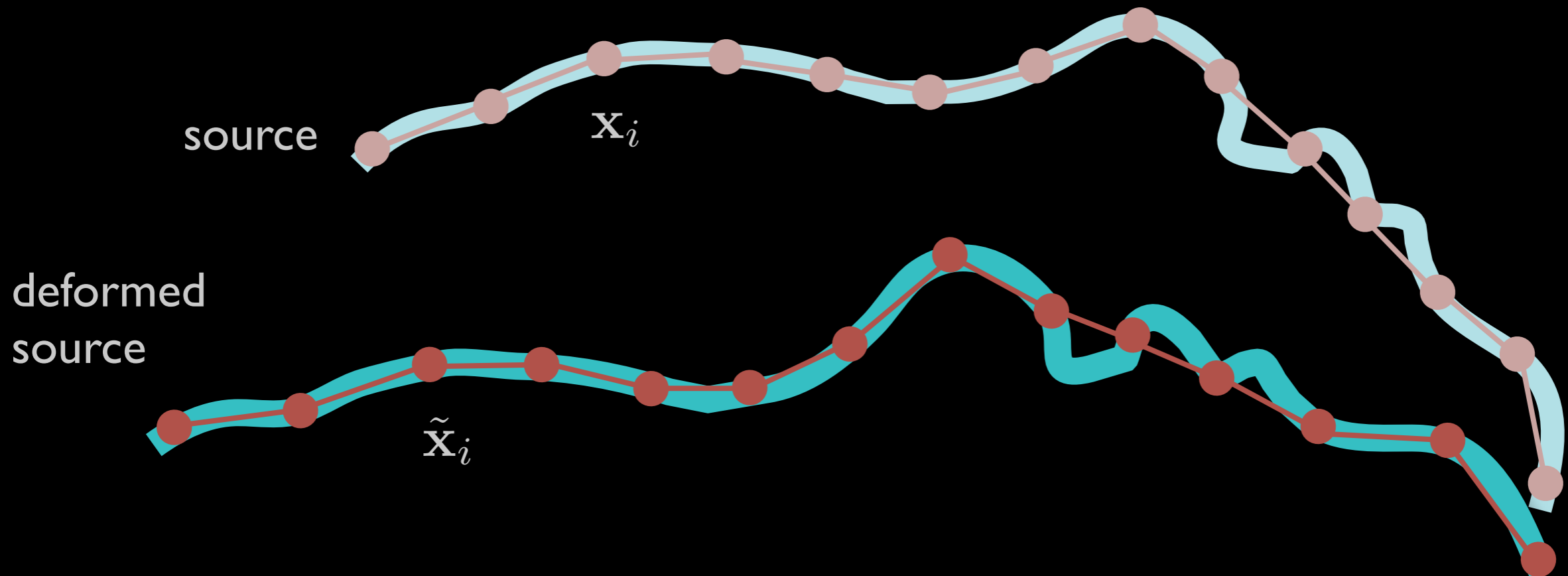


$$\tilde{\mathbf{v}}_j = \Phi_{\text{affine}}(\mathbf{v}_j) = \sum_{i=1}^n w_i(\mathbf{v}_j) [A_i(\mathbf{v}_j - \mathbf{x}_i) + \mathbf{x}_i + \mathbf{b}_i]$$



$$E_{\text{rigid}} = \sum_{\mathbf{x}_i} ((\mathbf{a}_1^\top \mathbf{a}_2)^2 + (\mathbf{a}_1^\top \mathbf{a}_3)^2 + (\mathbf{a}_2^\top \mathbf{a}_3)^2 \\ + (1 - \mathbf{a}_1^\top \mathbf{a}_1)^2 + (1 - \mathbf{a}_2^\top \mathbf{a}_2)^2 + (1 - \mathbf{a}_3^\top \mathbf{a}_3)^2)$$

Embedded Deformation



$$E_{\text{smooth}} = \sum_{\mathbf{x}_i} \sum_{\mathbf{x}_j} w_i(\mathbf{x}_j) \|A_i(\mathbf{x}_j - \mathbf{x}_i) + \mathbf{x}_i + \mathbf{b}_i - (\mathbf{x}_j + \mathbf{b}_j)\|_2^2$$

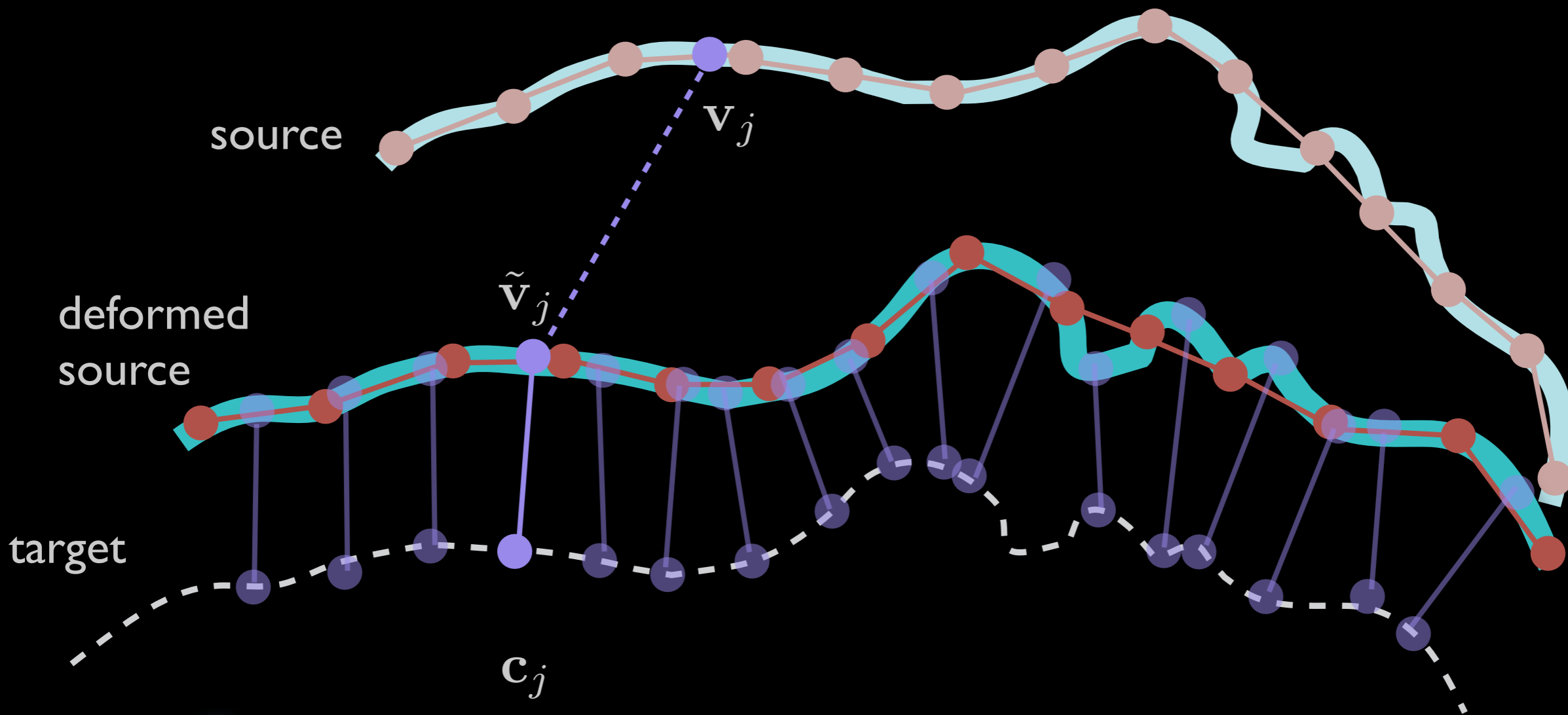


E_{rigid} E_{smooth}

Global Optimal Correspondence Optimization



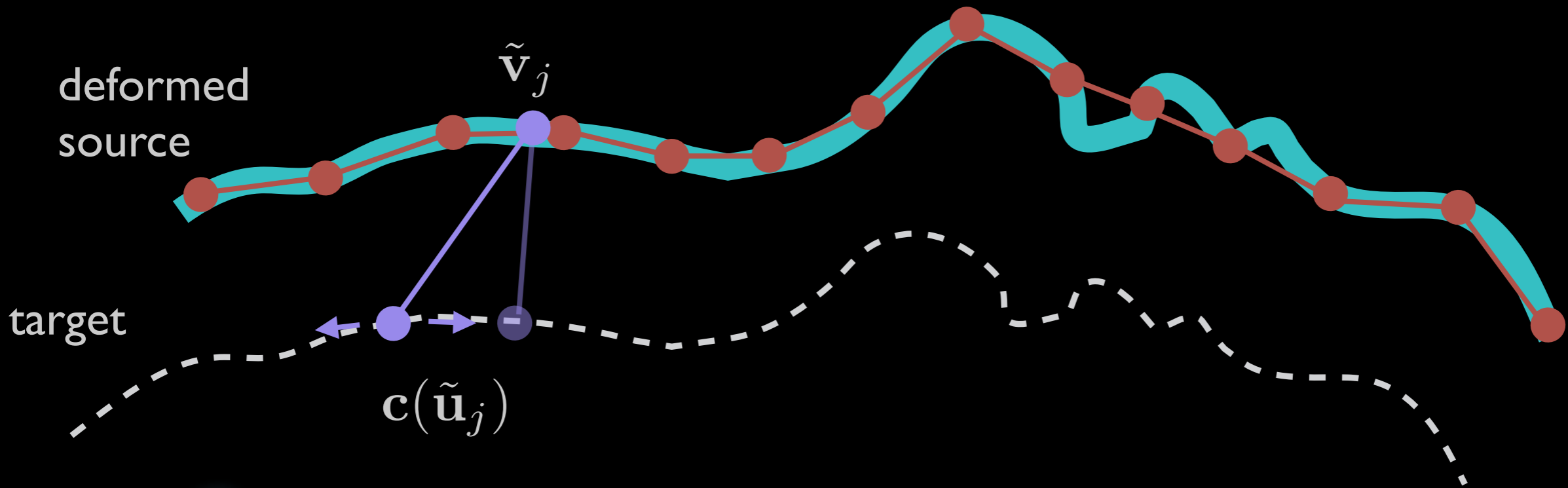
Minimize Alignment Error



Correspondences as **Unknowns**

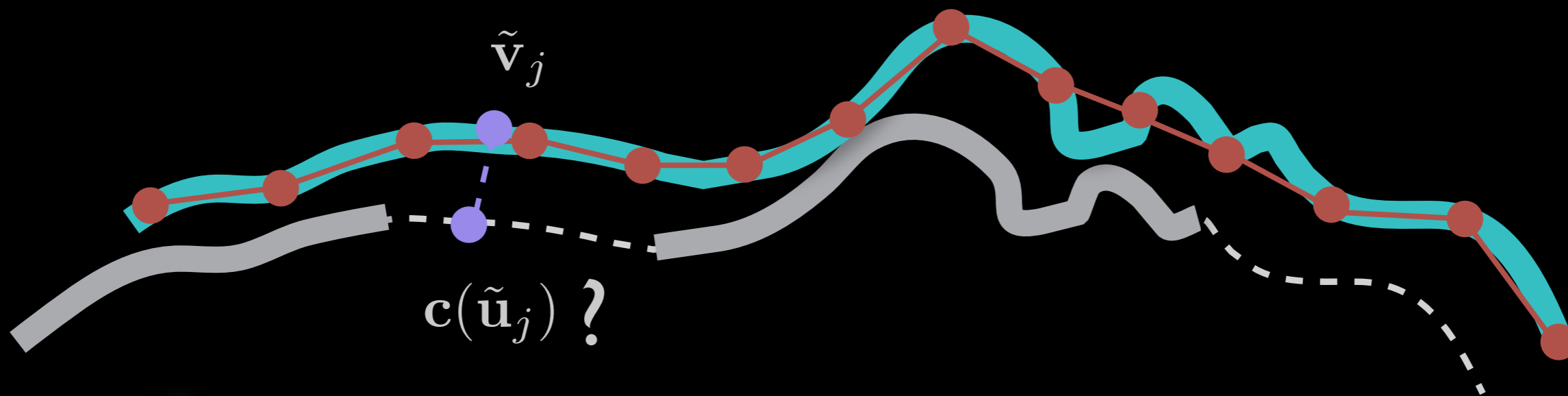
$$\mathbf{c}(\tilde{\mathbf{u}}_j)$$

$$\tilde{\mathbf{u}}_j = (\tilde{u}_j, \tilde{v}_j) \text{ optimization variable}$$



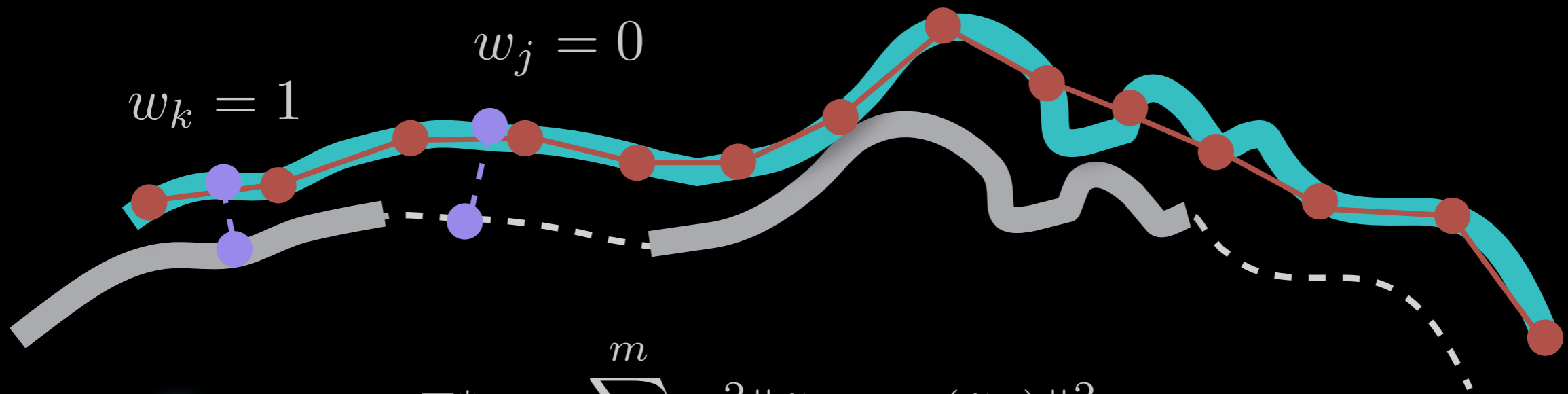
$$E_{\text{fit}} = \sum_{j=1}^m \|\tilde{\mathbf{v}}_j - \mathbf{c}(\tilde{\mathbf{u}}_j)\|_2^2$$

Partial Data



$$E_{\text{fit}} = \sum_{j=1}^m \|\tilde{\mathbf{v}}_j - \mathbf{c}(\tilde{\mathbf{u}}_j)\|_2^2$$

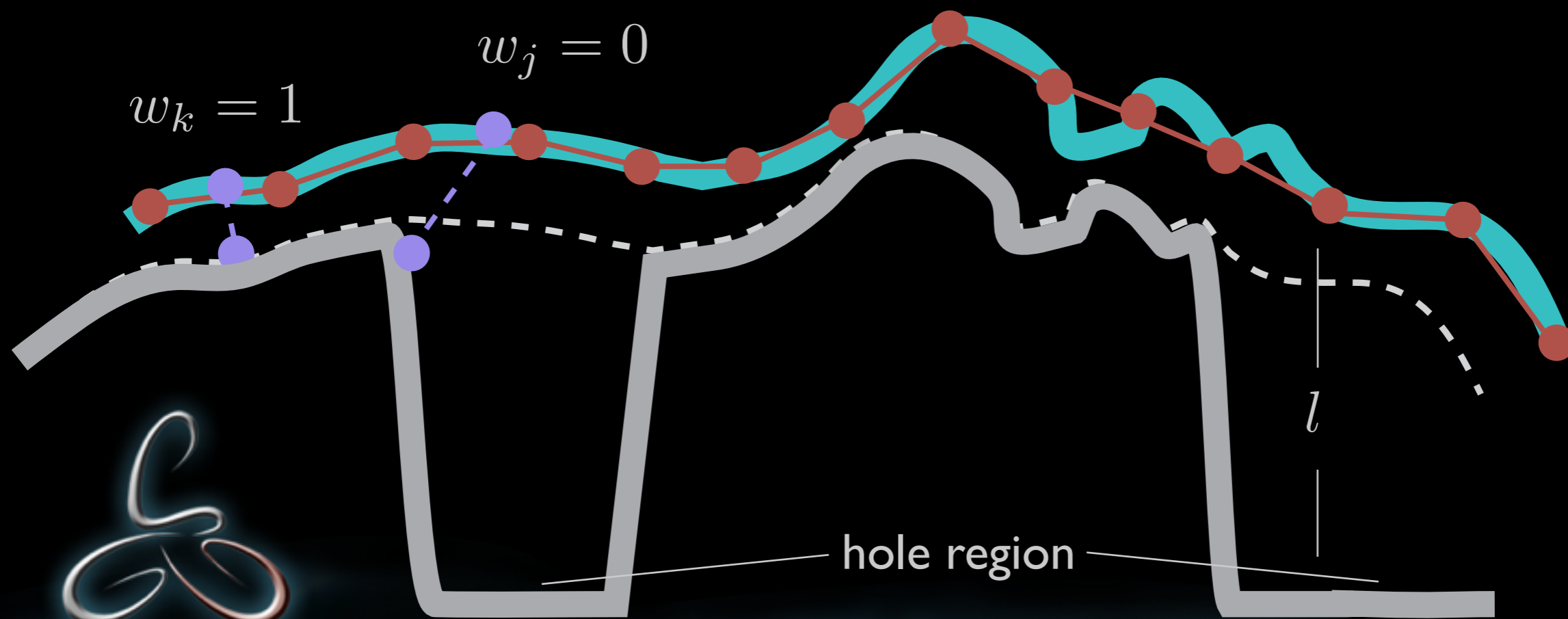
Confidence Weights



$$E_{\text{fit}}^* = \sum_{i=1}^m w_j^2 \|\tilde{\mathbf{v}}_j - \mathbf{c}(\tilde{\mathbf{u}}_j)\|_2^2$$

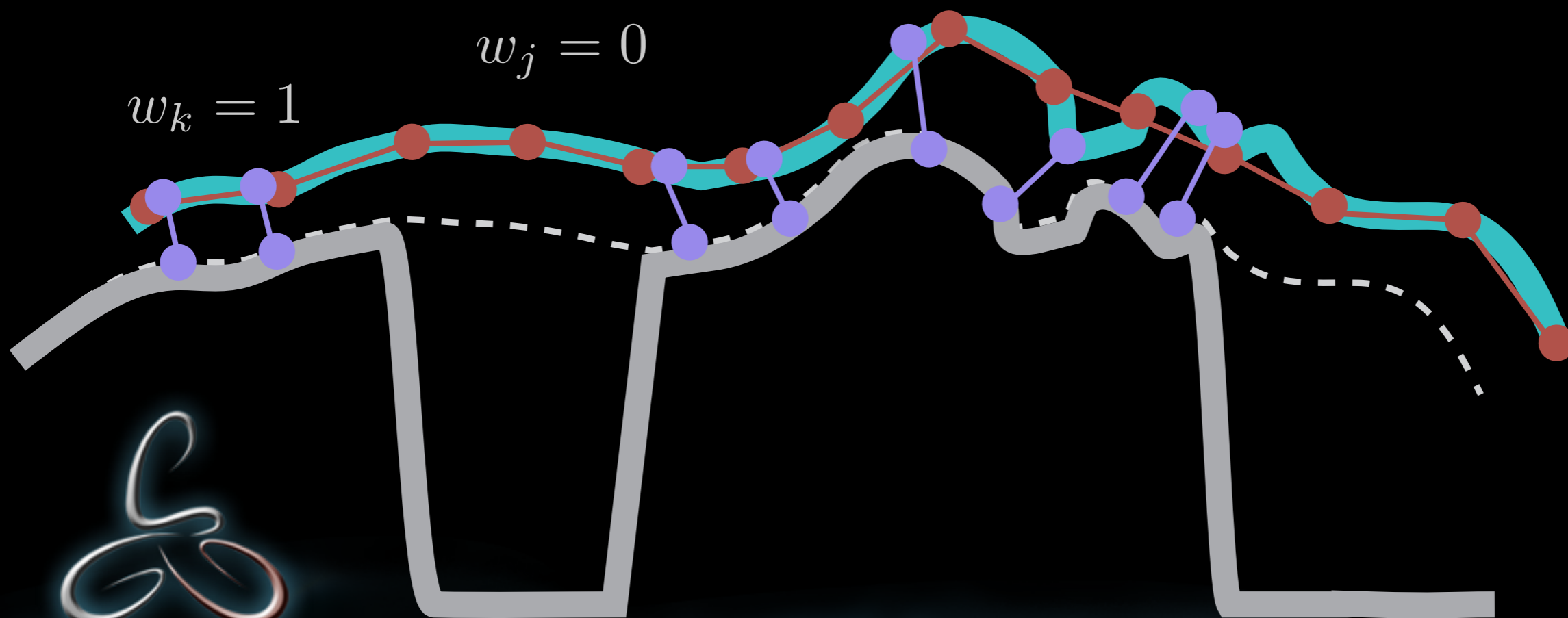


Continuous Representation



$$E_{\text{fit}}^* = \sum_{i=1}^m w_j^2 \|\tilde{\mathbf{v}}_j - \mathbf{c}(\tilde{\mathbf{u}}_j)\|_2^2$$

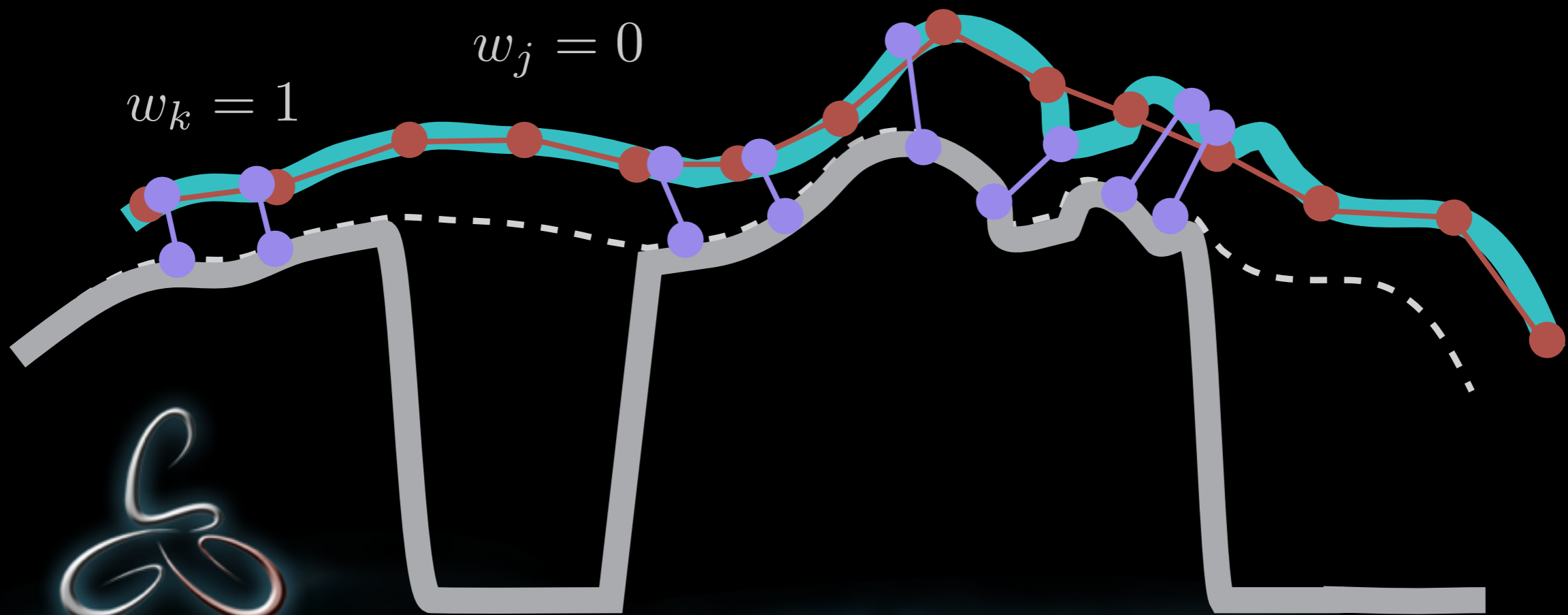
$$E_{\text{conf}} = \sum_{j=1}^m (1 - w_j^2)^2$$





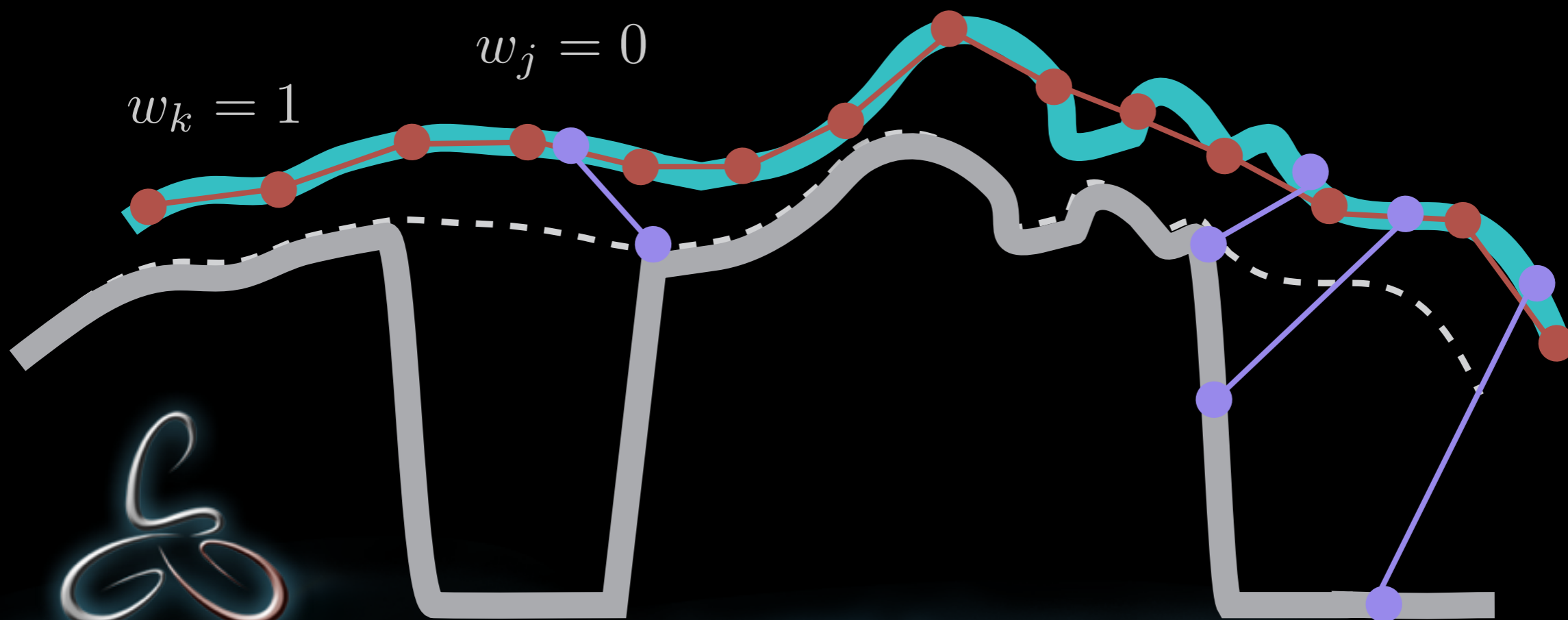
$$E_{\text{fit}}^* = \sum_{i=1}^m w_j^2 \|\tilde{\mathbf{v}}_j - \mathbf{c}(\tilde{\mathbf{u}}_j)\|_2^2$$

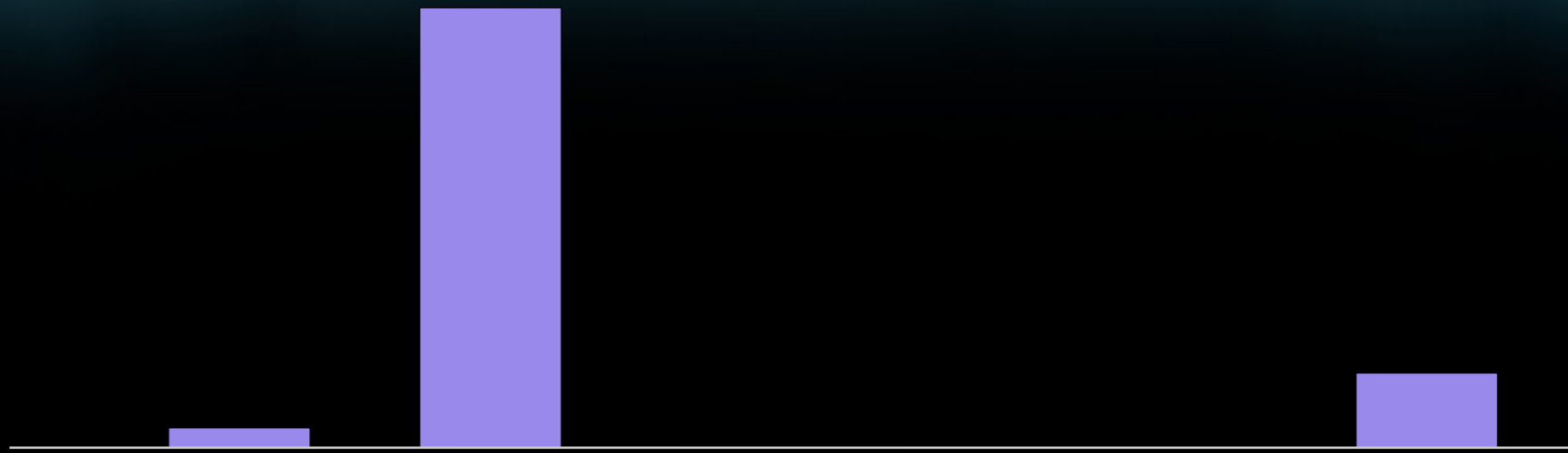
$$E_{\text{conf}} = \sum_{j=1}^m (1 - w_j^2)^2$$



$$E_{\text{fit}}^* = \sum_{i=1}^m w_j^2 \|\tilde{\mathbf{v}}_j - \mathbf{c}(\tilde{\mathbf{u}}_j)\|_2^2$$

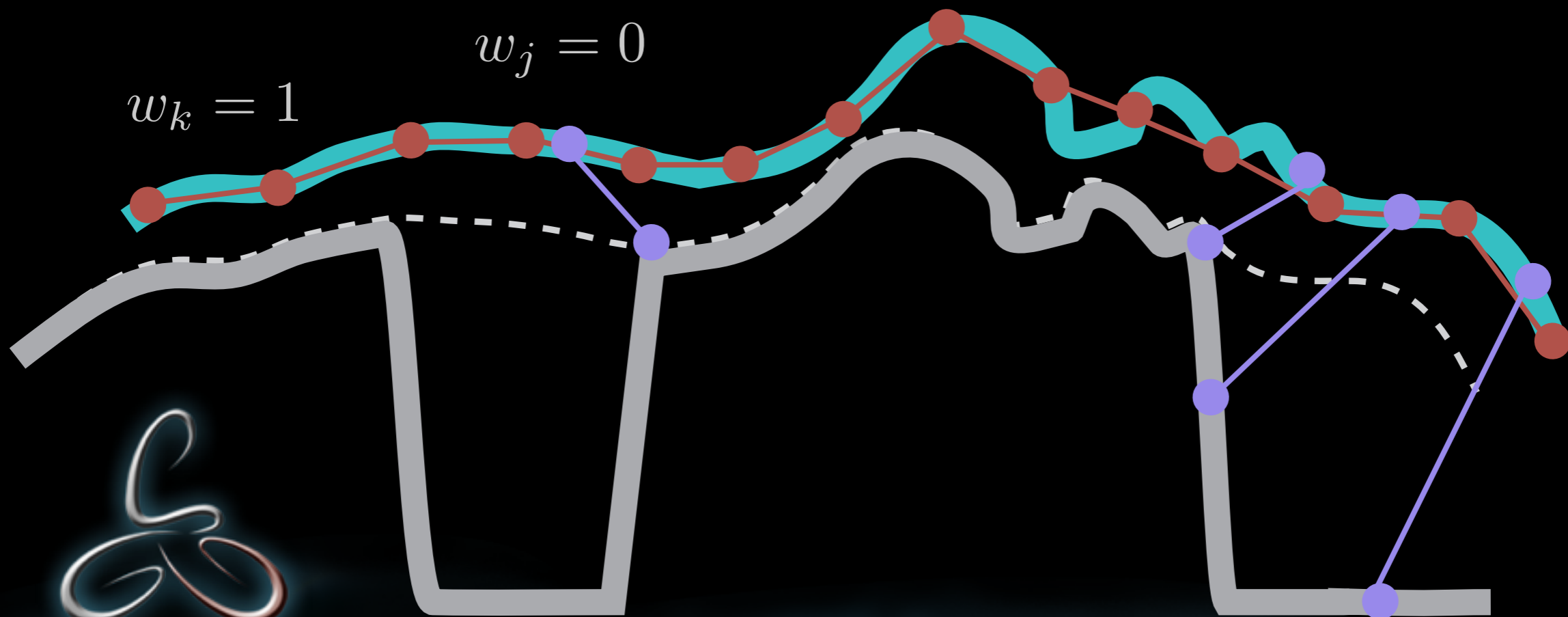
$$E_{\text{conf}} = \sum_{j=1}^m (1 - w_j^2)^2$$



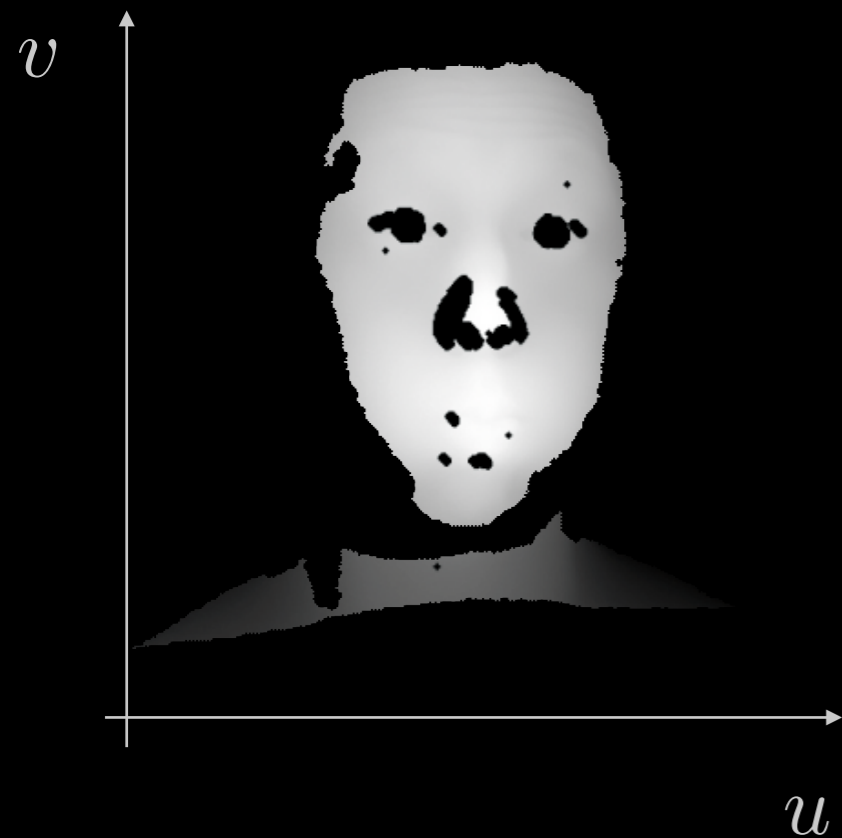


$$E_{\text{fit}}^* = \sum_{i=1}^m w_j^2 \|\tilde{\mathbf{v}}_j - \mathbf{c}(\tilde{\mathbf{u}}_j)\|_2^2$$

$$E_{\text{conf}} = \sum_{j=1}^m (1 - w_j^2)^2$$



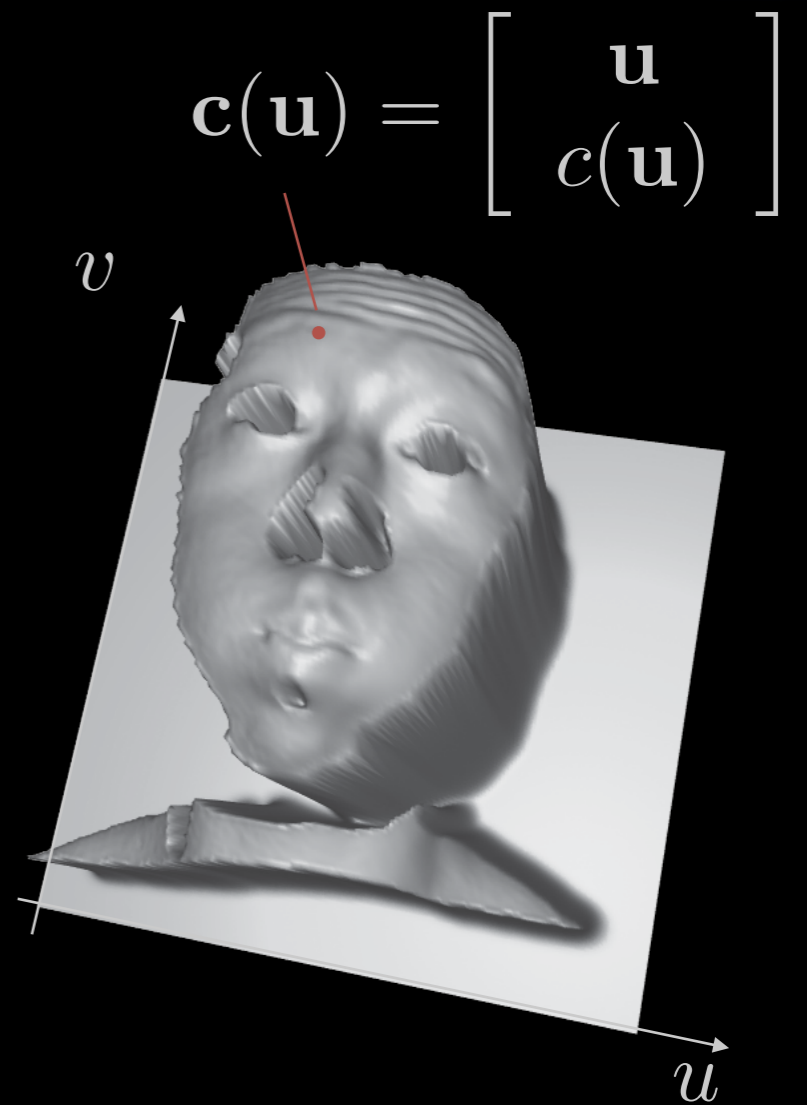
Depth-Scan Parameterization



raw depth map



depth scan



weighted least squares approximation



Optimization

$$E_{\text{tot}} = \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}} + \alpha_{\text{fit}} E_{\text{fit}}^* + \alpha_{\text{conf}} E_{\text{conf}}$$

- Minimize deformation energy
- Minimize alignment error
- Maximize regions of overlap



Optimization

$$E_{\text{tot}} = \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}} + \alpha_{\text{fit}} E_{\text{fit}}^* + \alpha_{\text{conf}} E_{\text{conf}}$$

- Minimize deformation energy
- Minimize alignment error
- Maximize regions of overlap



Regularization Relaxation

$$E_{\text{tot}} = \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}} + \alpha_{\text{fit}} E_{\text{fit}}^* + \alpha_{\text{conf}} E_{\text{conf}}$$

$$\alpha_{\text{rigid}} = 1000 \rightarrow 1 \qquad \alpha_{\text{fit}} = 0.1$$

$$\alpha_{\text{smooth}} = 100 \rightarrow 0.1 \qquad \alpha_{\text{conf}} = 100 \rightarrow 1$$

stiffness reduction

confidence adaptation



Results



Synthetic Model

Elephant (329 nodes, 21k vertices)

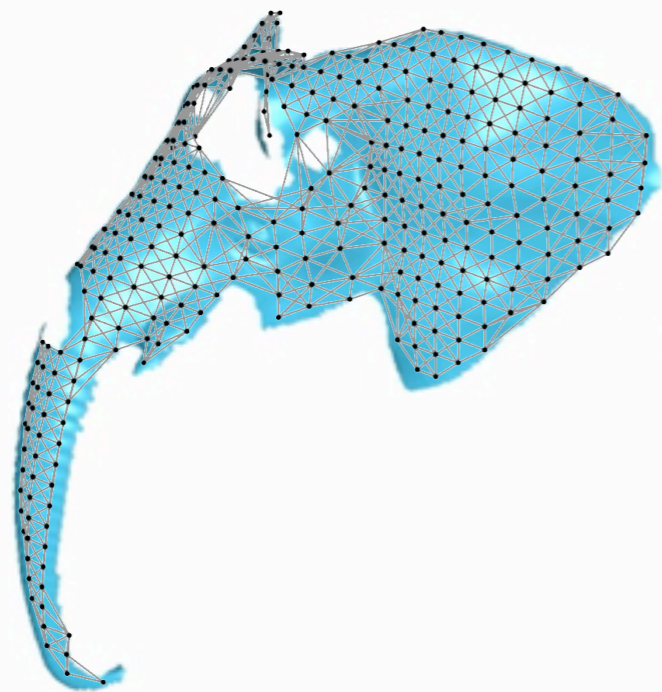


Source



Synthetic Model

Elephant (329 nodes, 21k vertices)

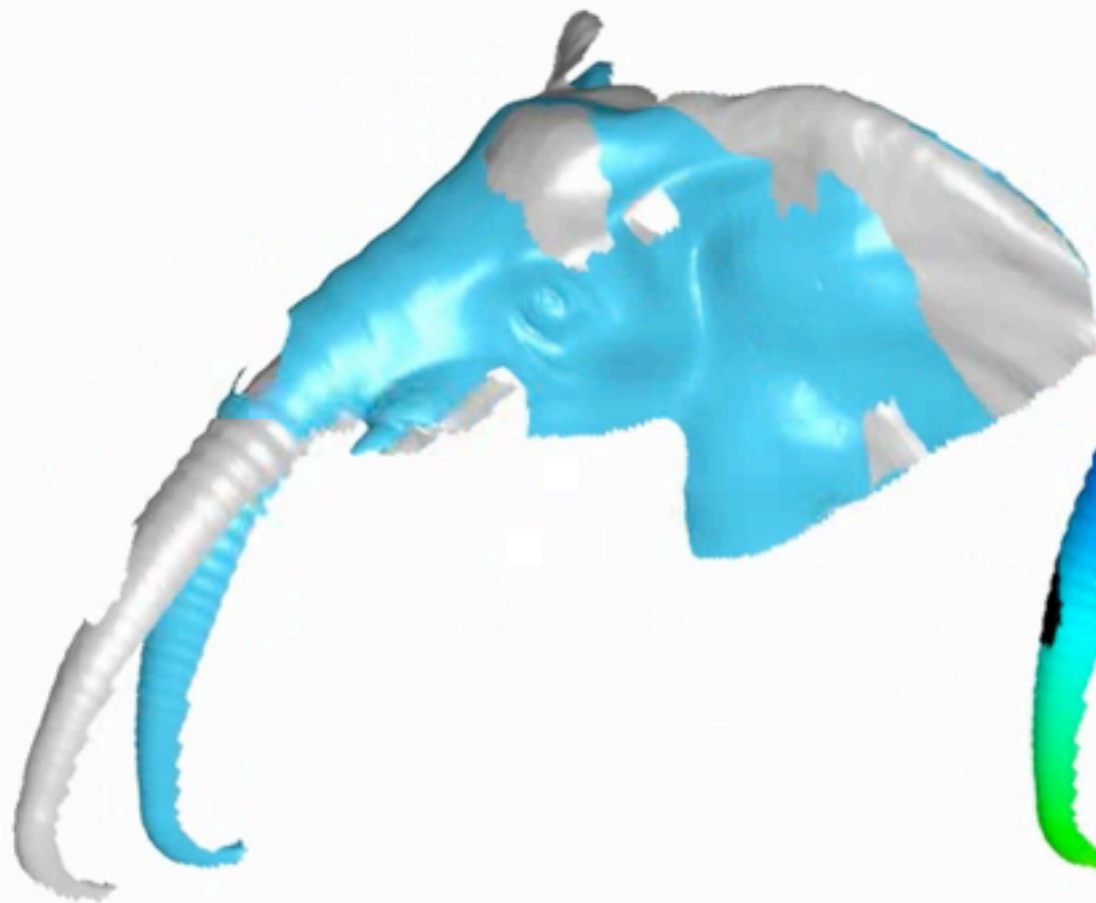


Source

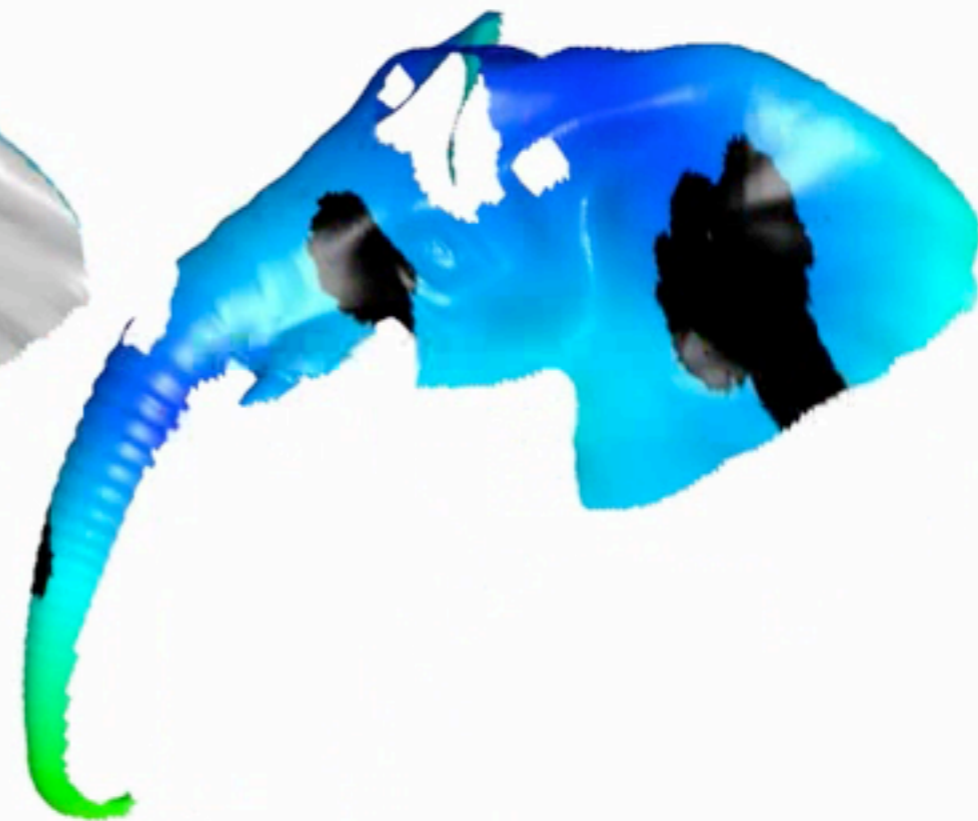


Comparison to Ground-Truth

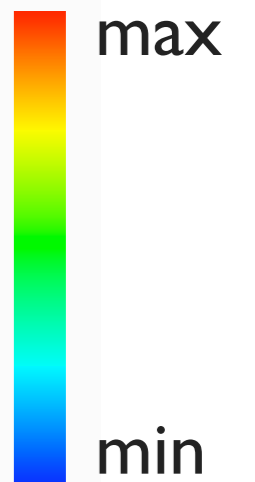
21K vertices 329 nodes



Registration

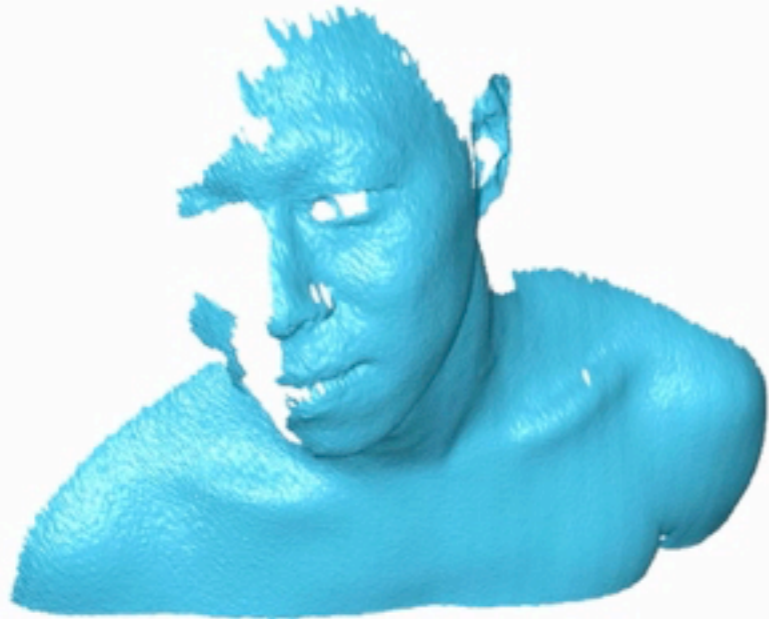


Correspondence Error



Real Scans

120 K vertices 336 nodes

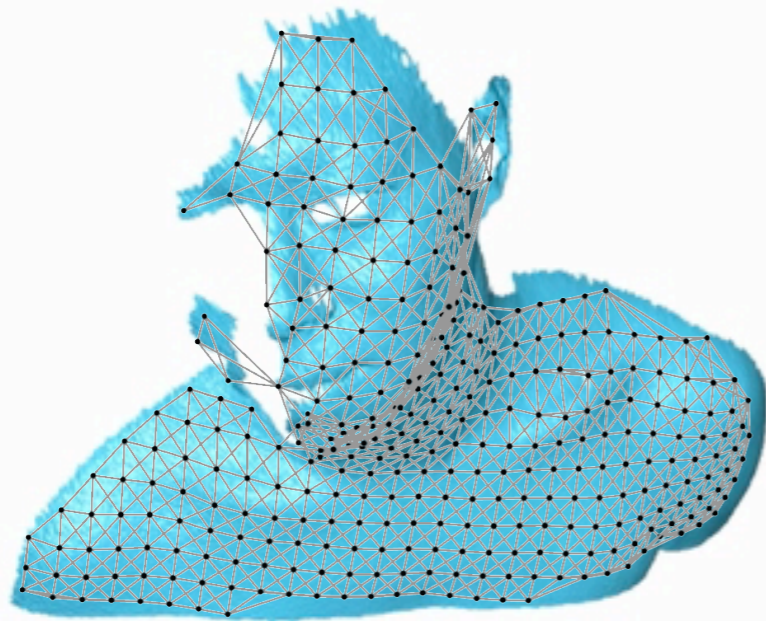


Source



Real Scans

120 K vertices 336 nodes



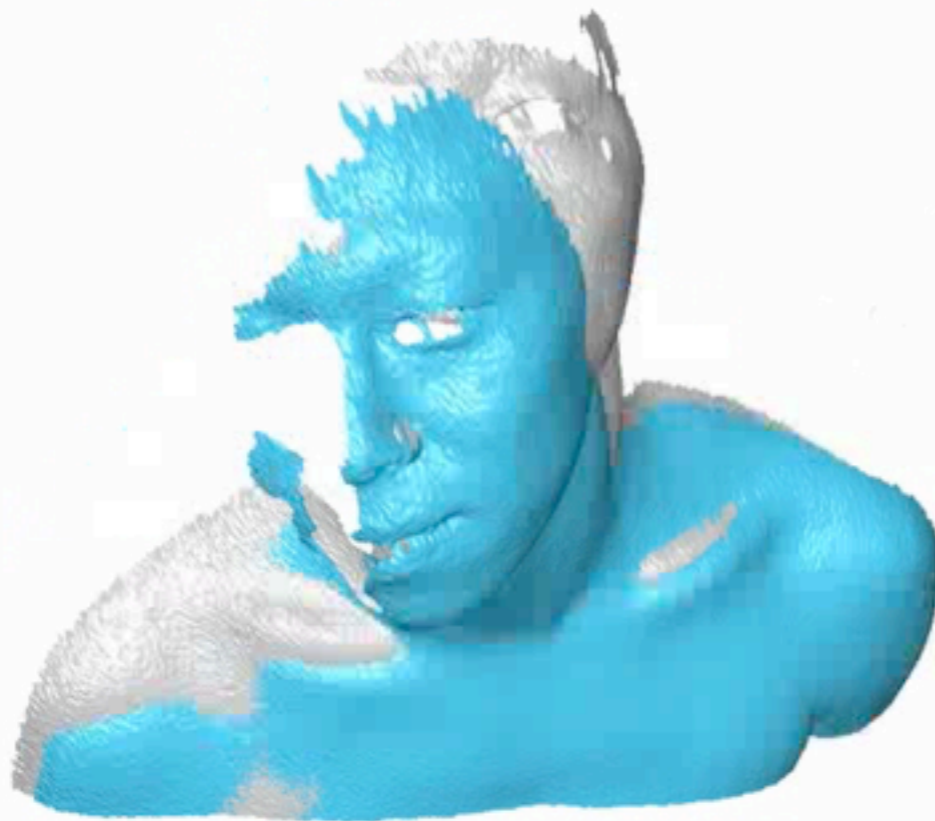
Source



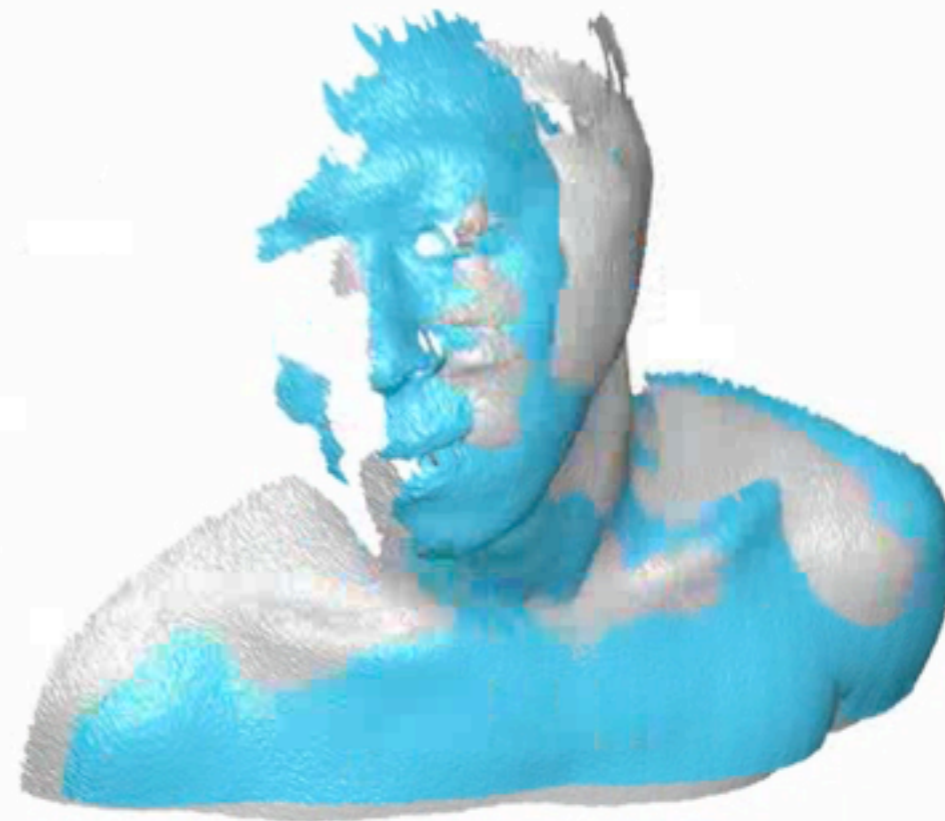
Optimization

219 iterations

2 min 19 s



Initial Alignment



Registration



Energy Term Visualization



E_{conf}



E_{fit}



E_{smooth}

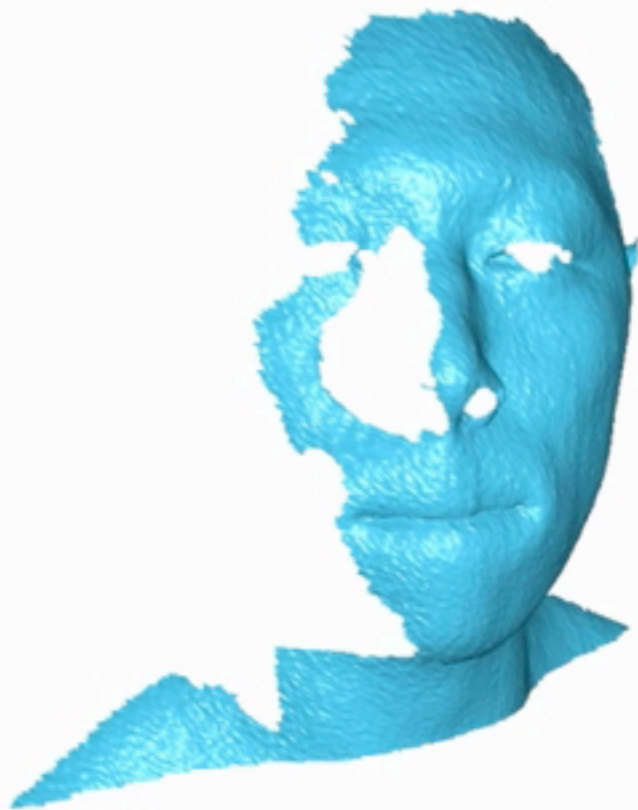


E_{rigid}



Deformation

44 K vertices 798 nodes

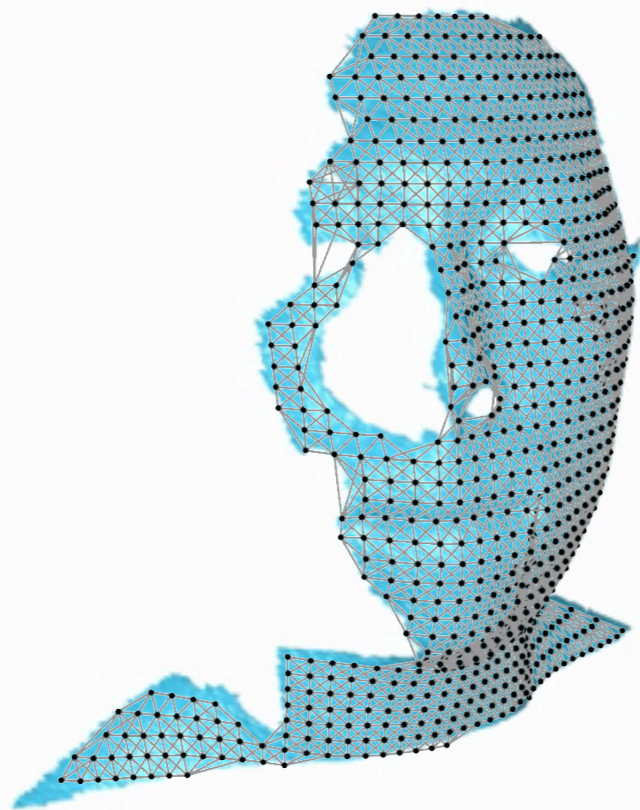


Source



Deformation

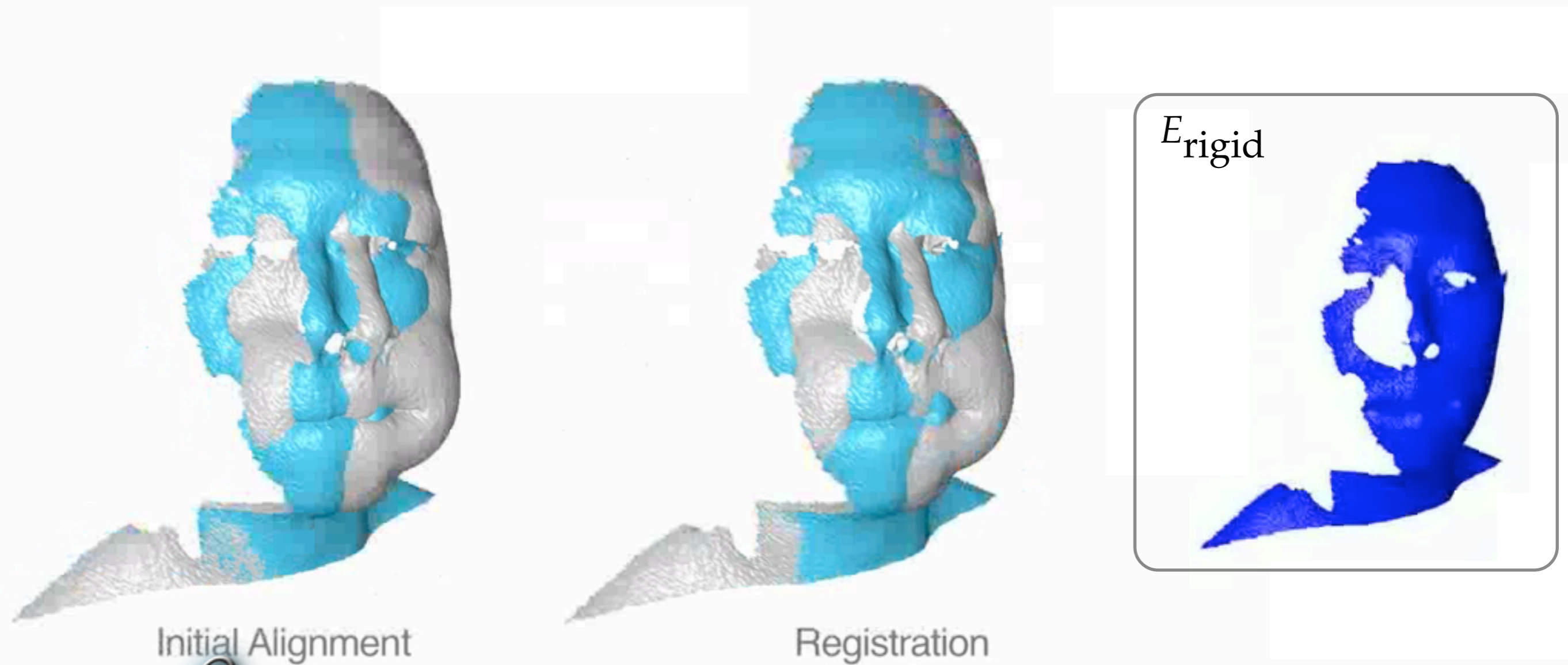
44 K vertices 798 nodes



Source



Optimization

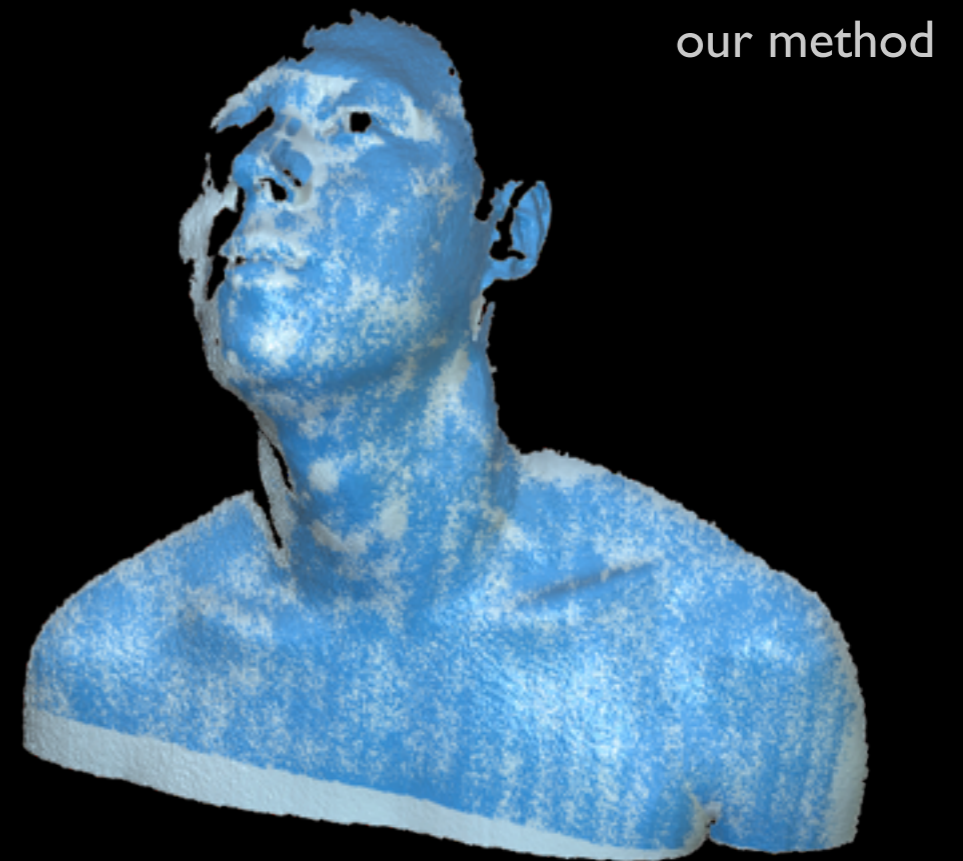
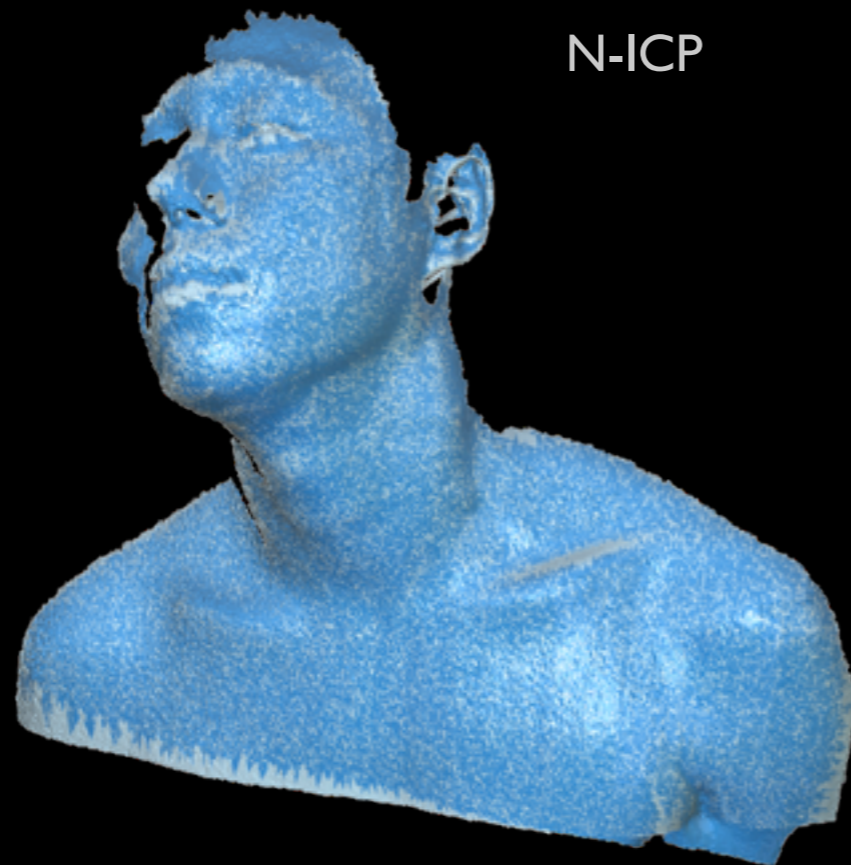


Comparison to Previous Techniques on Non-Rigid ICP



Comparison with other N-ICP

[Pauly et al. '05] [Pottmann et al. '06]

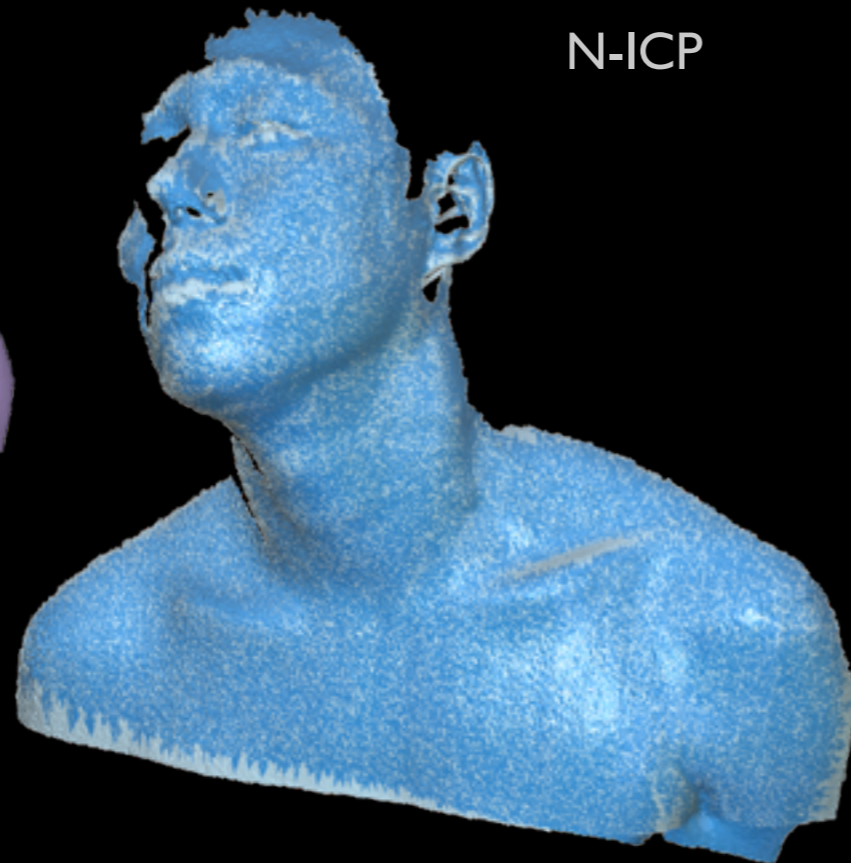


Comparison with other N-ICP

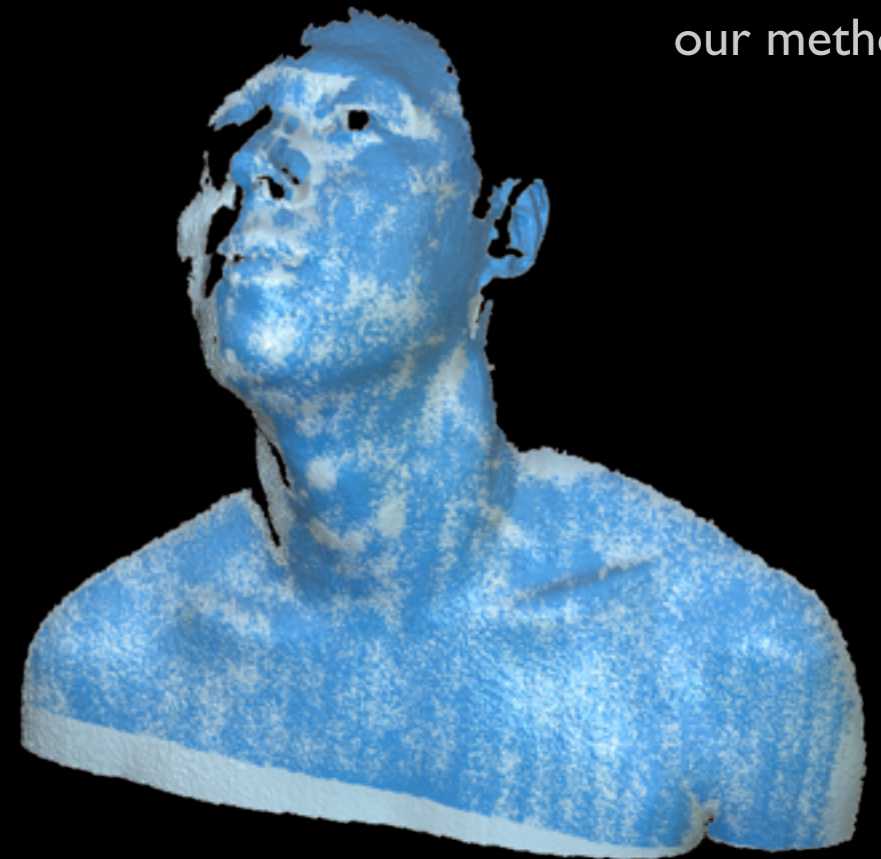
[Pauly et al. '05] [Pottmann et al. '06]



texturing



N-ICP



our method

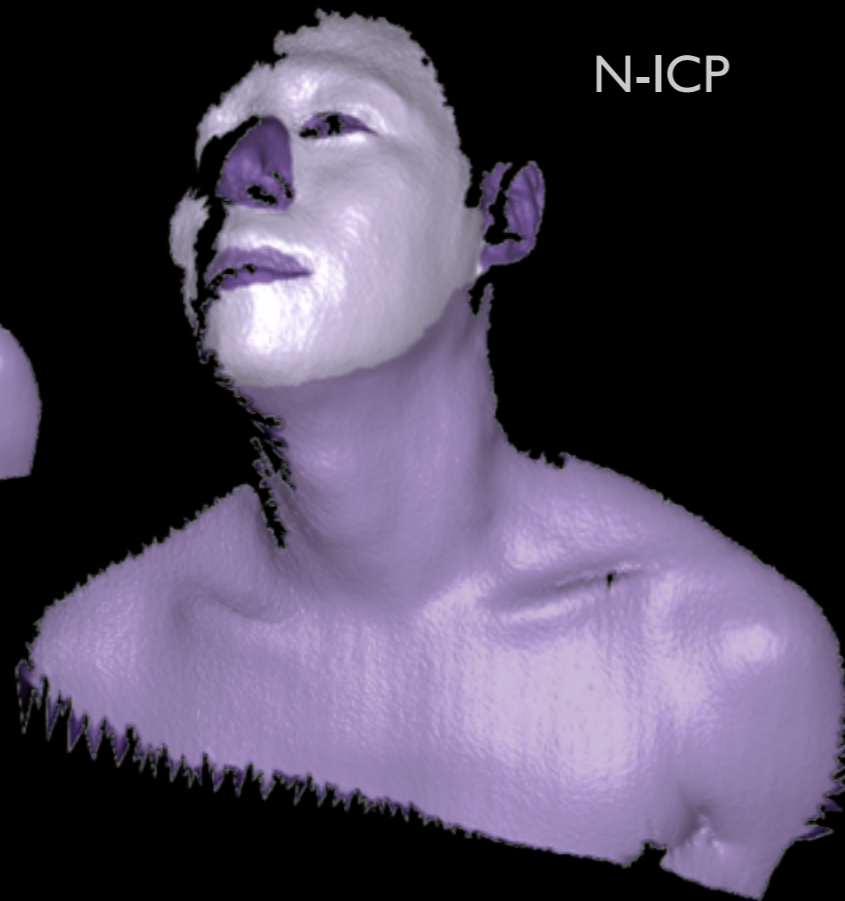


Comparison with other N-ICP

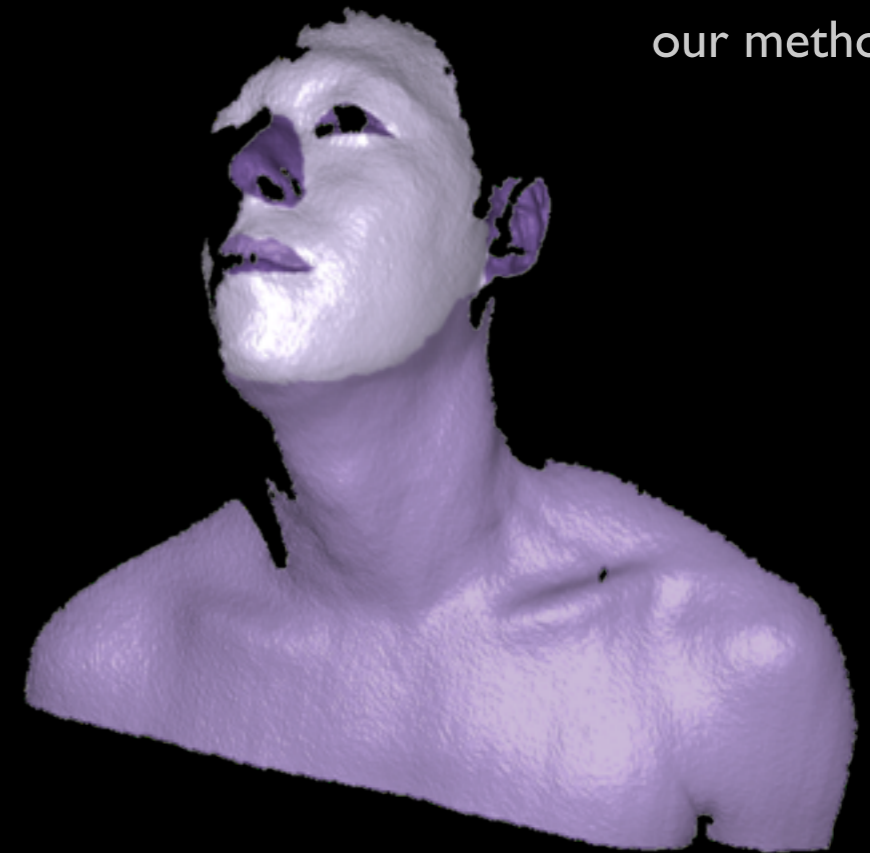
[Pauly et al. '05] [Pottmann et al. '06]



texturing



N-ICP



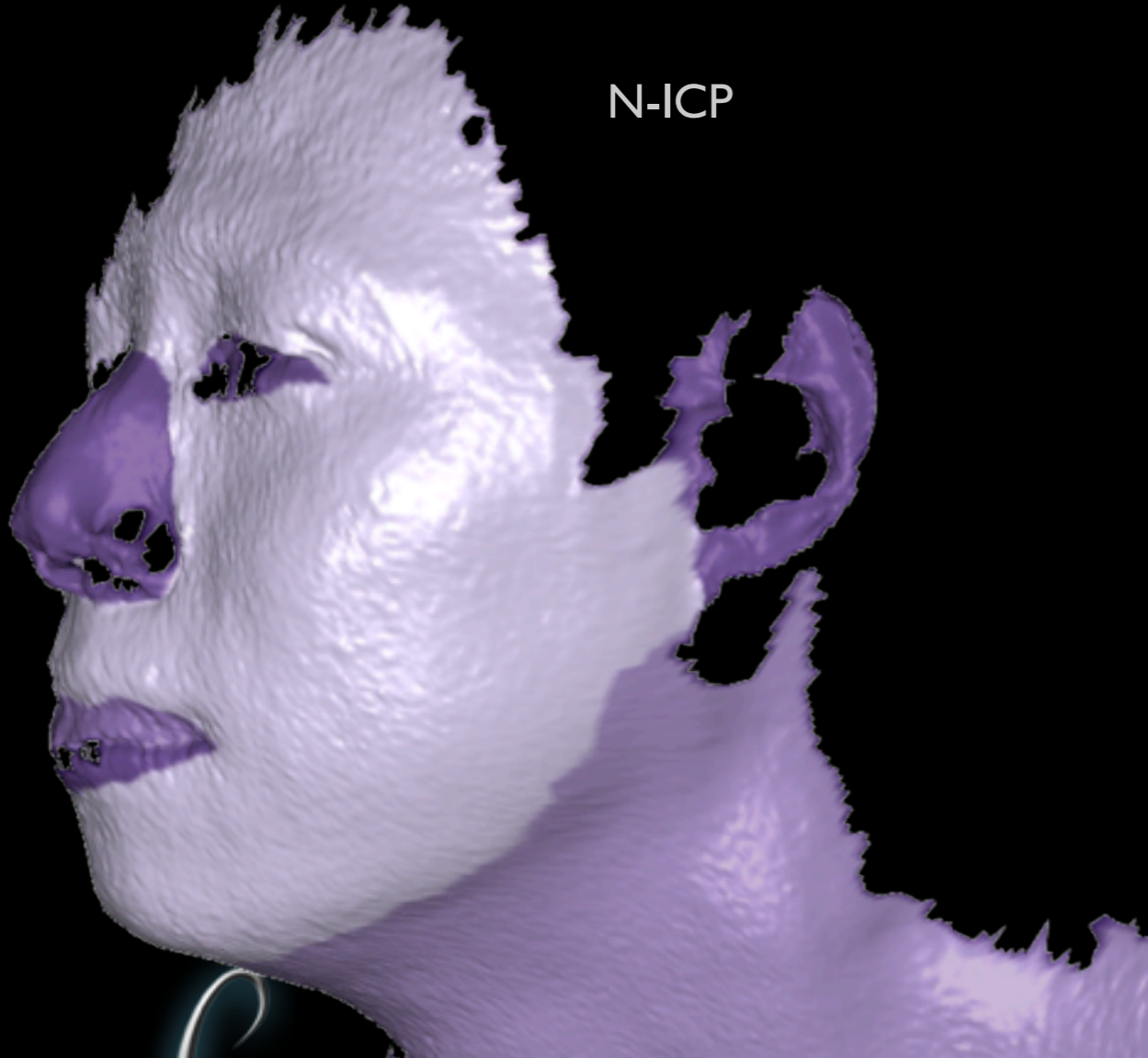
our method



Comparison with Non-Rigid ICP

[Pauly et al. '05] [Pottmann et al. '06]

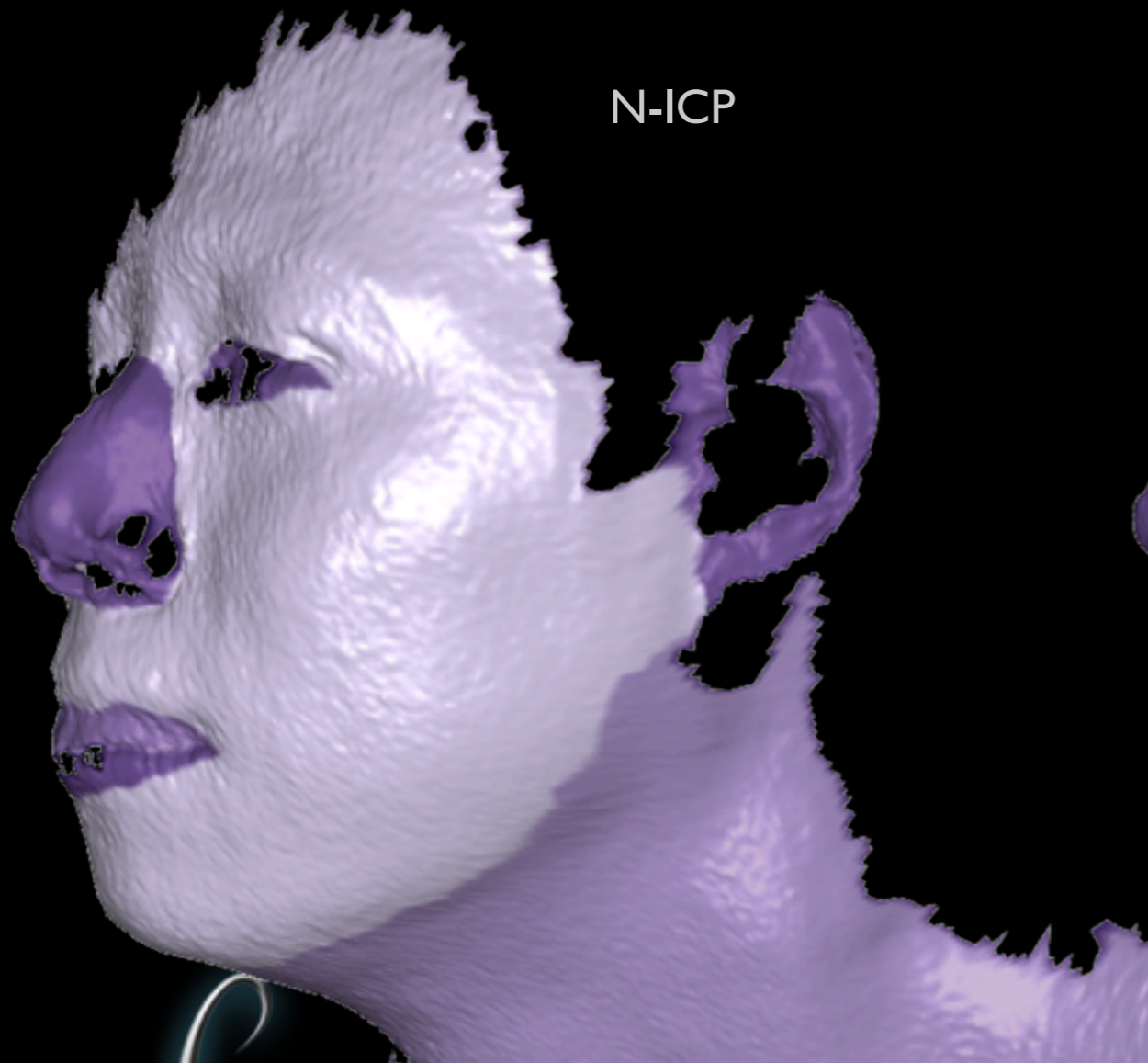
N-ICP



Comparison with Non-Rigid ICP

[Pauly et al. '05] [Pottmann et al. '06]

N-ICP



our method

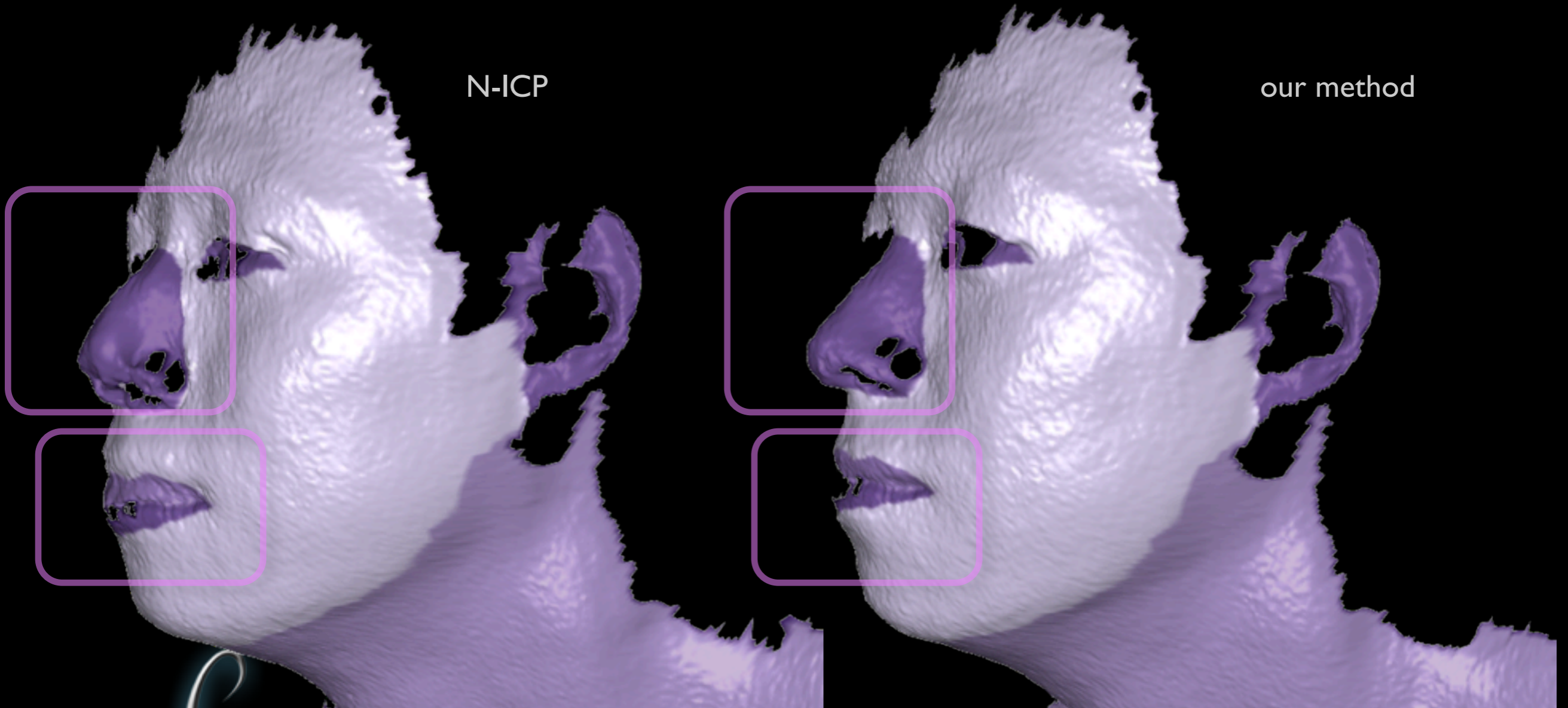


Comparison with Non-Rigid ICP

[Pauly et al. '05] [Pottmann et al. '06]

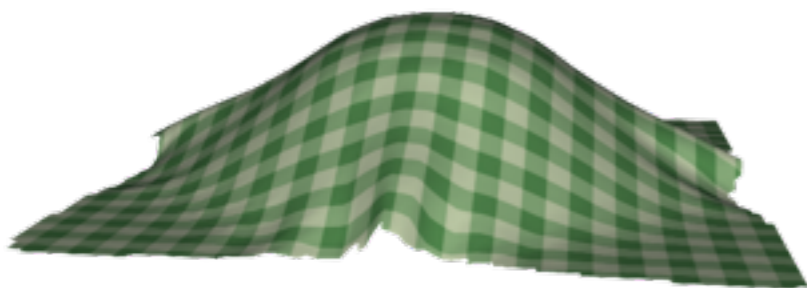
N-ICP

our method



Depth-Scan of a Draping Table Cloth

source

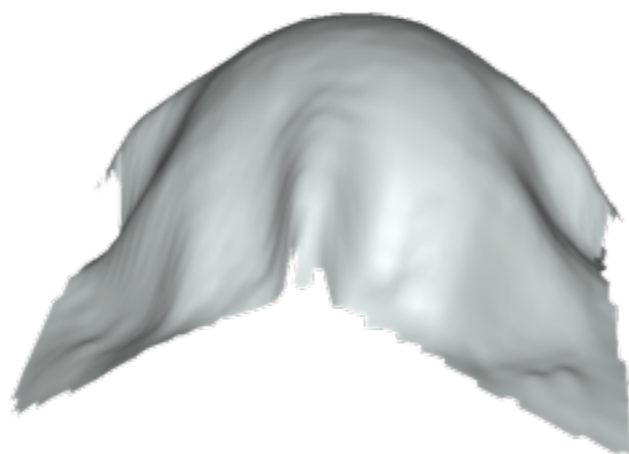


Depth-Scan of a Draping Table Cloth

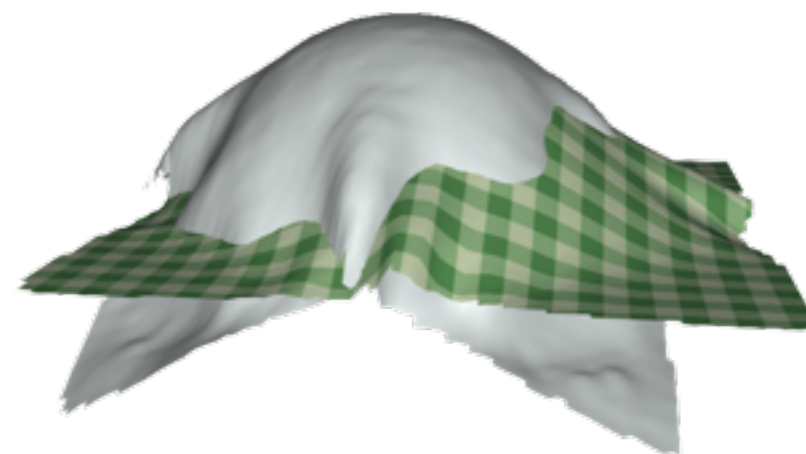
source



target



initial alignment



Suitable for Isometric Deformations

N-ICP

our method



Suitable for **Isometric Deformations**

N-ICP



our method



Suitable for Isometric Deformations

N-ICP

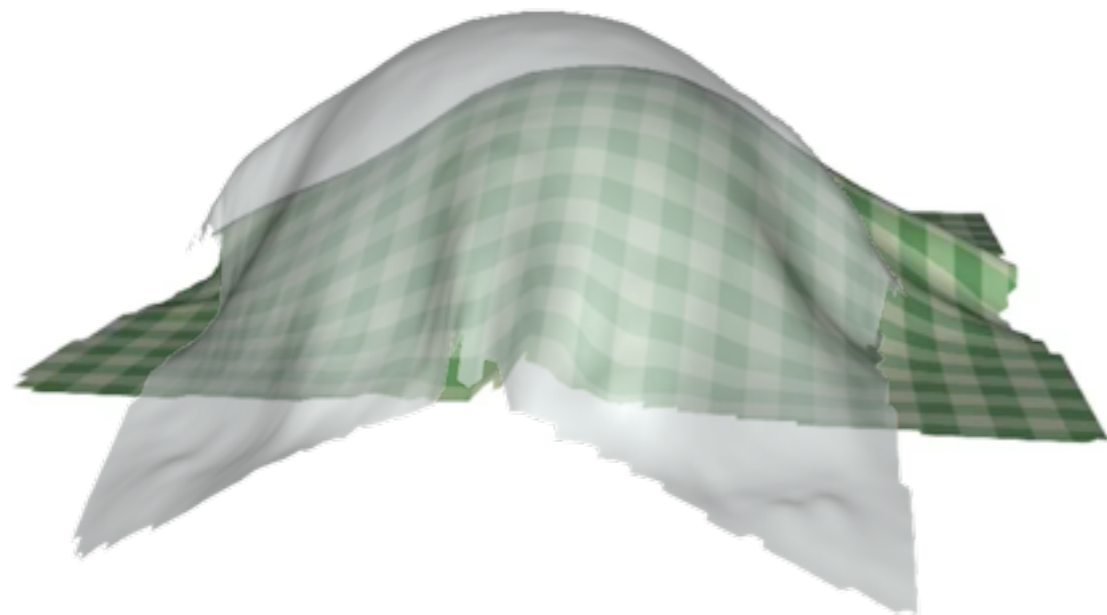


our method

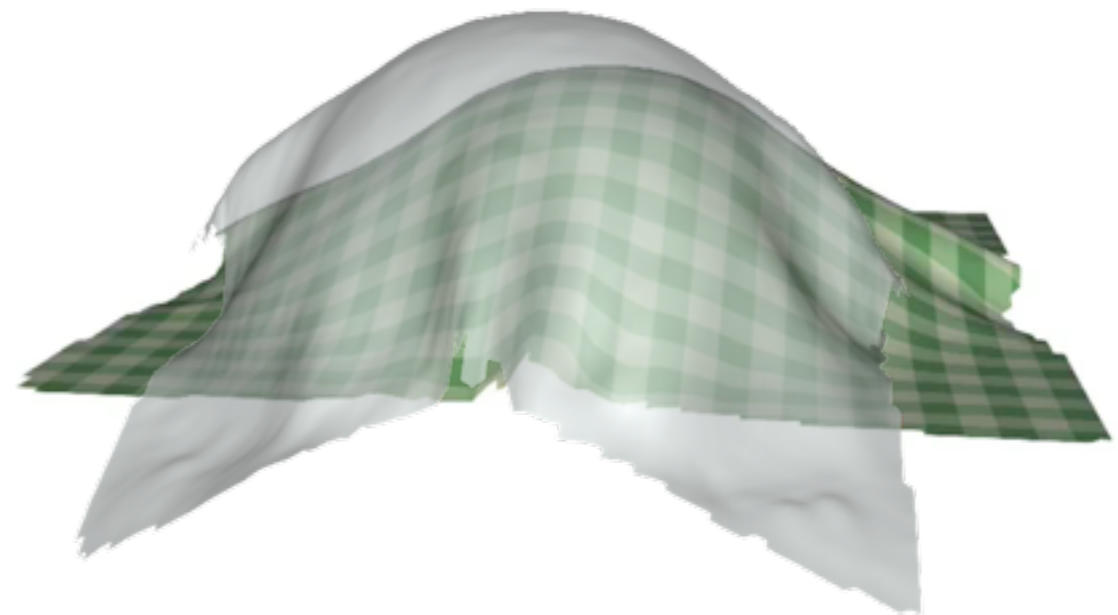


Suitable for **Isometric Deformations**

N-ICP



our method

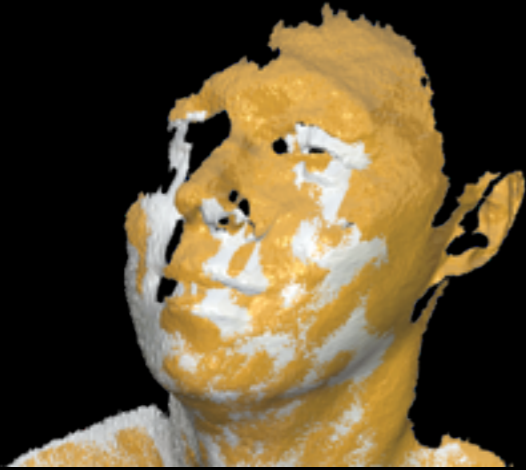


Limitations



$c(u, v)$

v

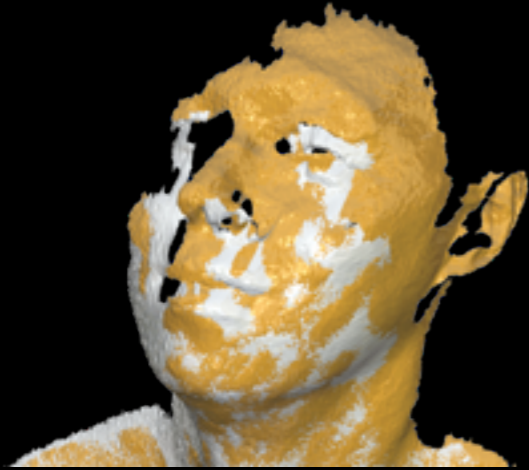


u

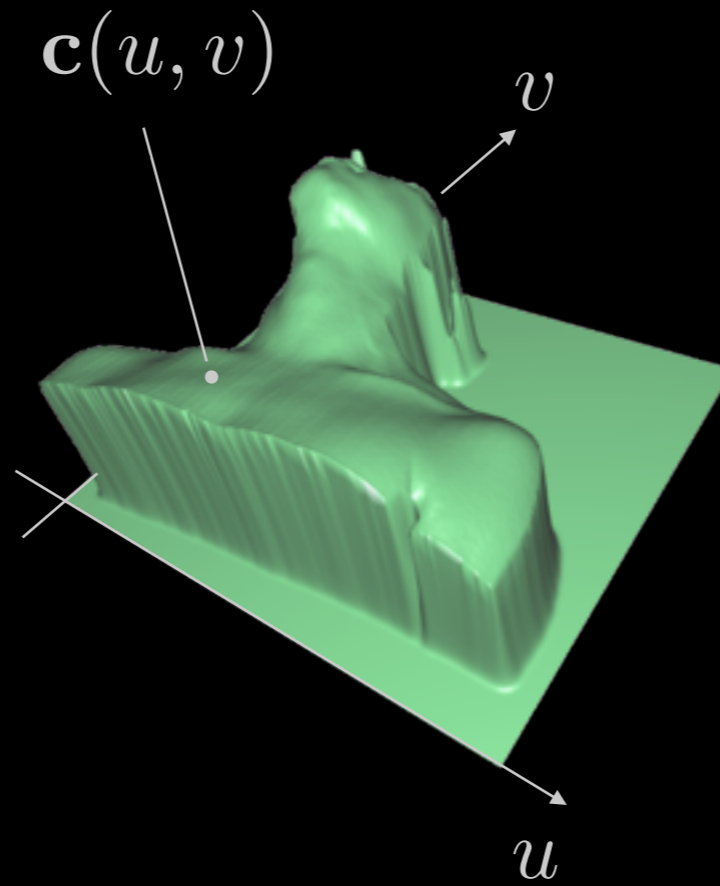
local minima



Limitations



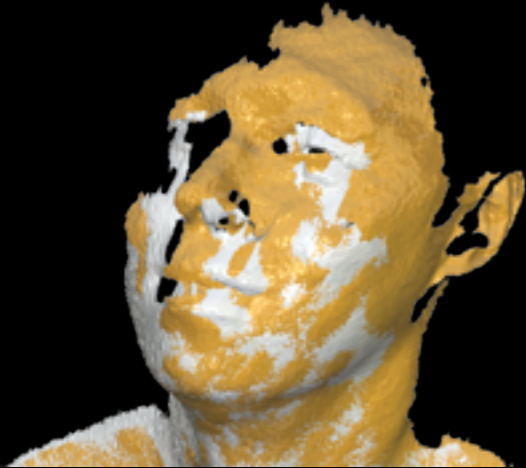
local minima



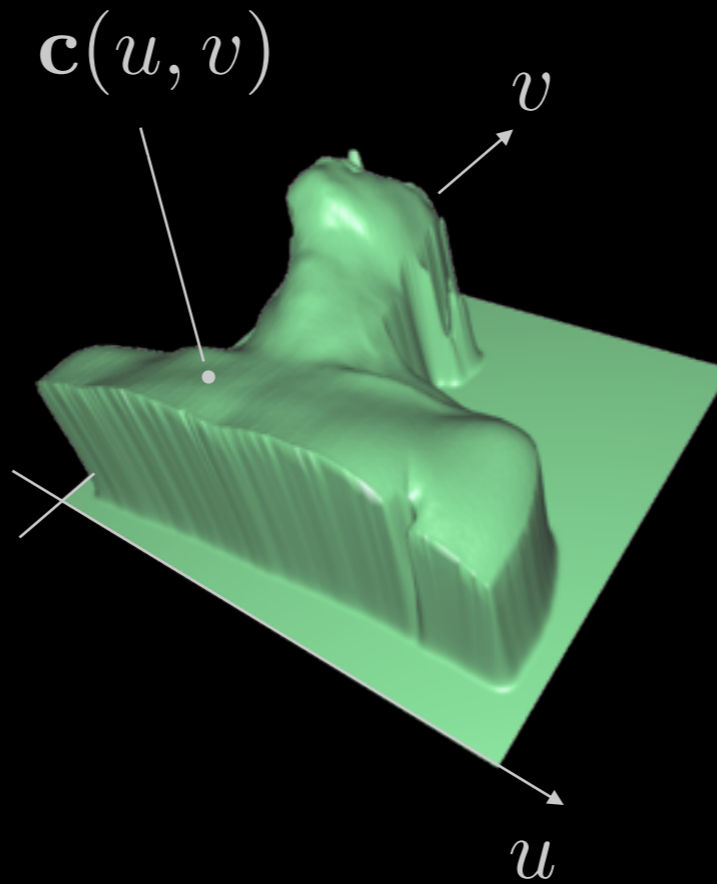
parameterization



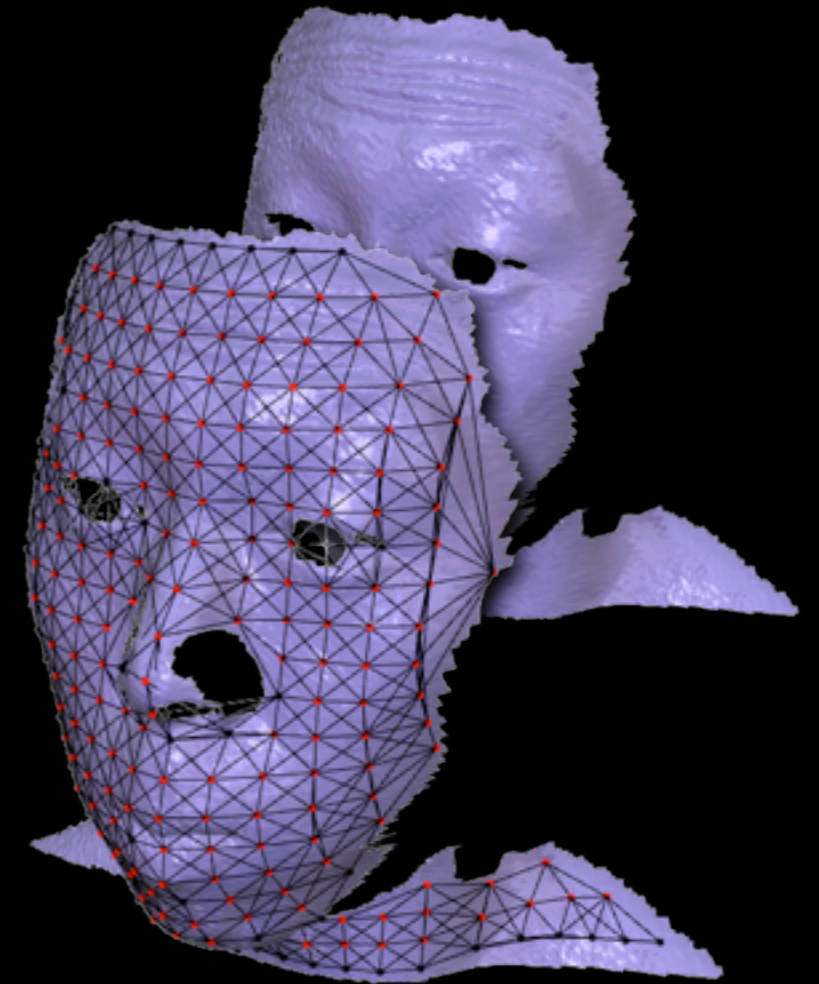
Limitations



local minima



parameterization



small features

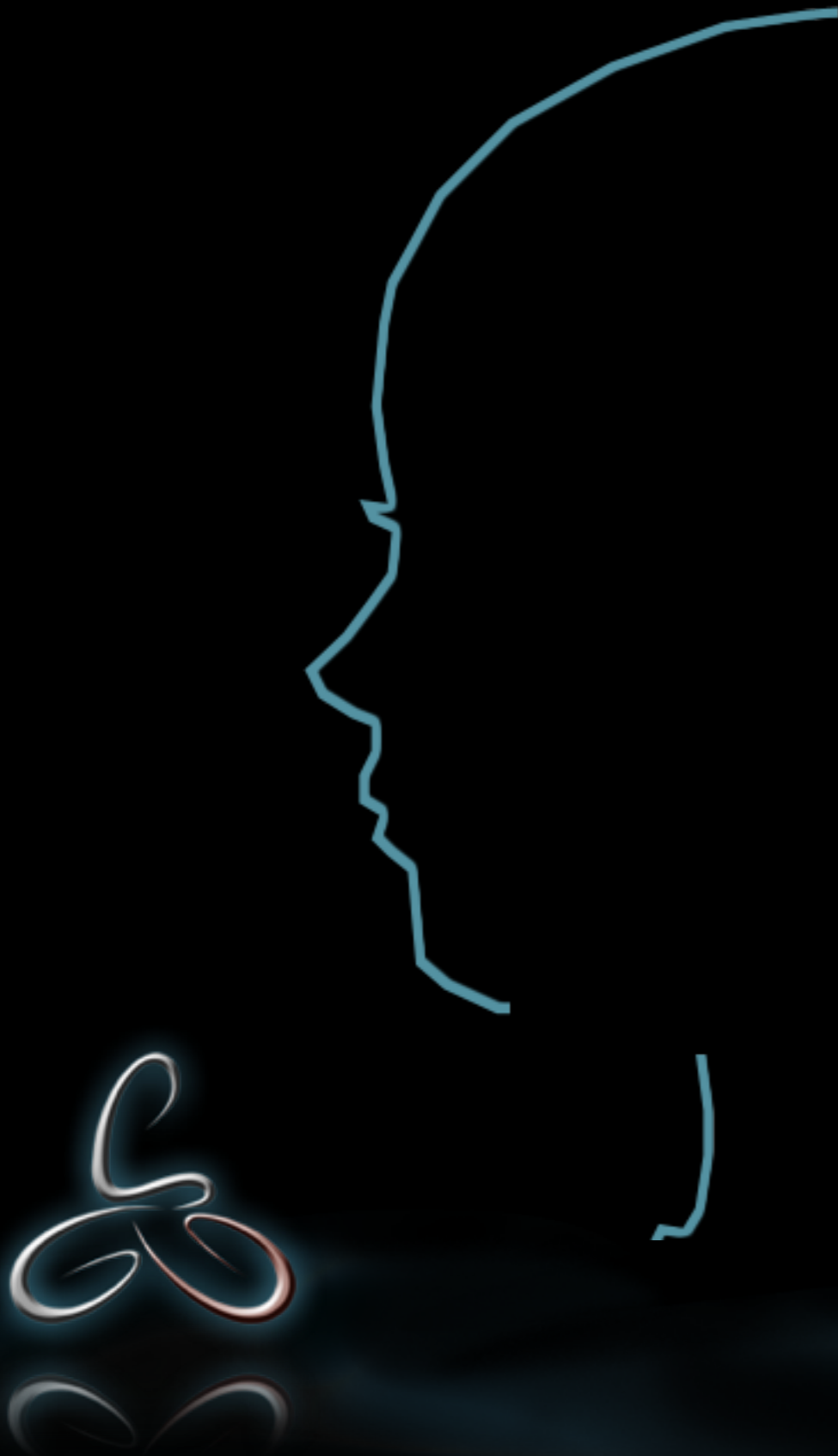


Robust De-Coupling



Robust Non-Rigid ICP

Non-Rigid ICP



Robust Non-Rigid ICP

Non-Rigid ICP



Robust Non-Rigid ICP

Non-Rigid ICP



Robust Non-Rigid ICP

Non-Rigid ICP

Sampling



Robust Non-Rigid ICP

Non-Rigid ICP

Sampling



Robust Non-Rigid ICP



Non-Rigid ICP

Sampling



Closest Point

Robust Non-Rigid ICP



Non-Rigid ICP

Sampling



Closest Point

Robust Non-Rigid ICP



Non-Rigid ICP

Sampling



Closest Point

Robust Non-Rigid ICP



Non-Rigid ICP

Sampling

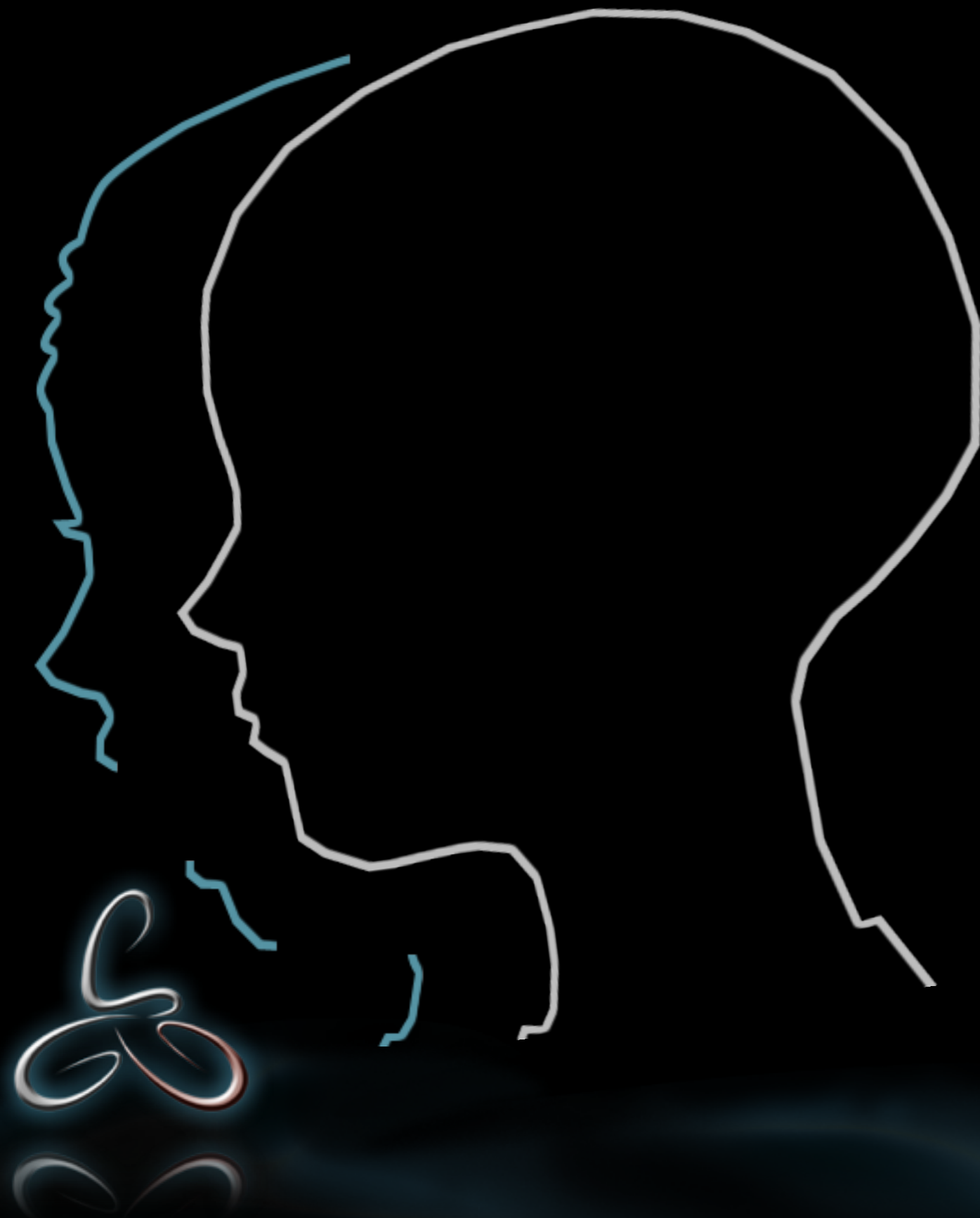


Closest Point



Deformation

Robust Non-Rigid ICP



Non-Rigid ICP

Sampling

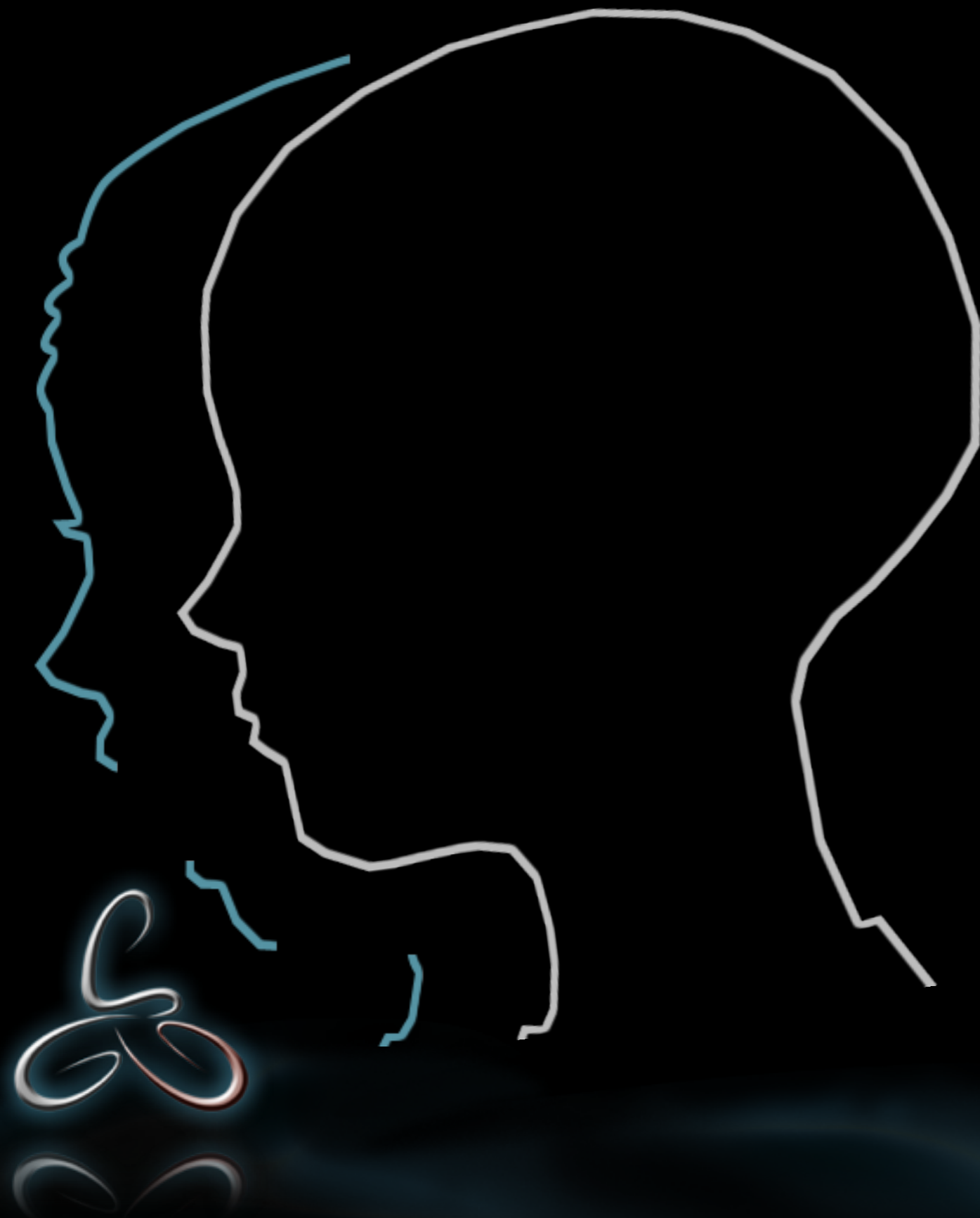


Closest Point

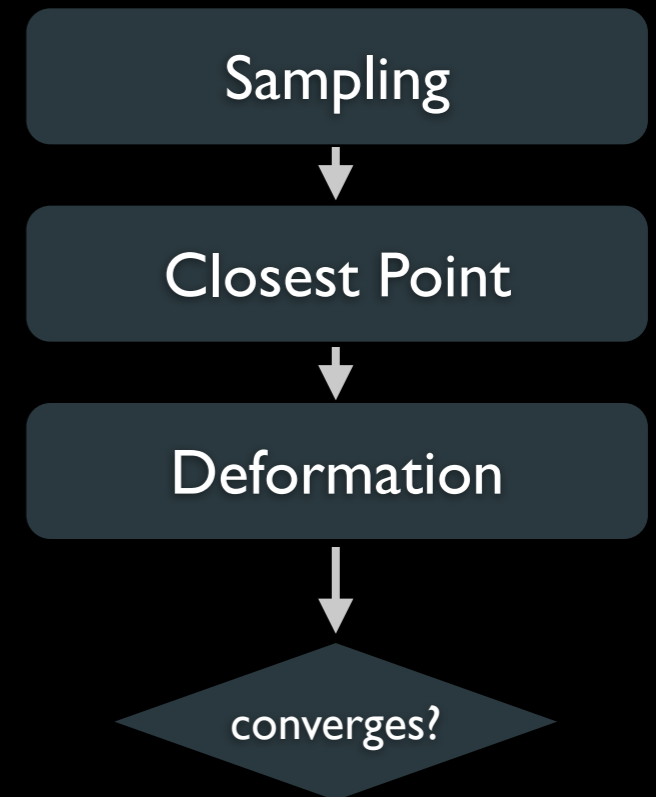


Deformation

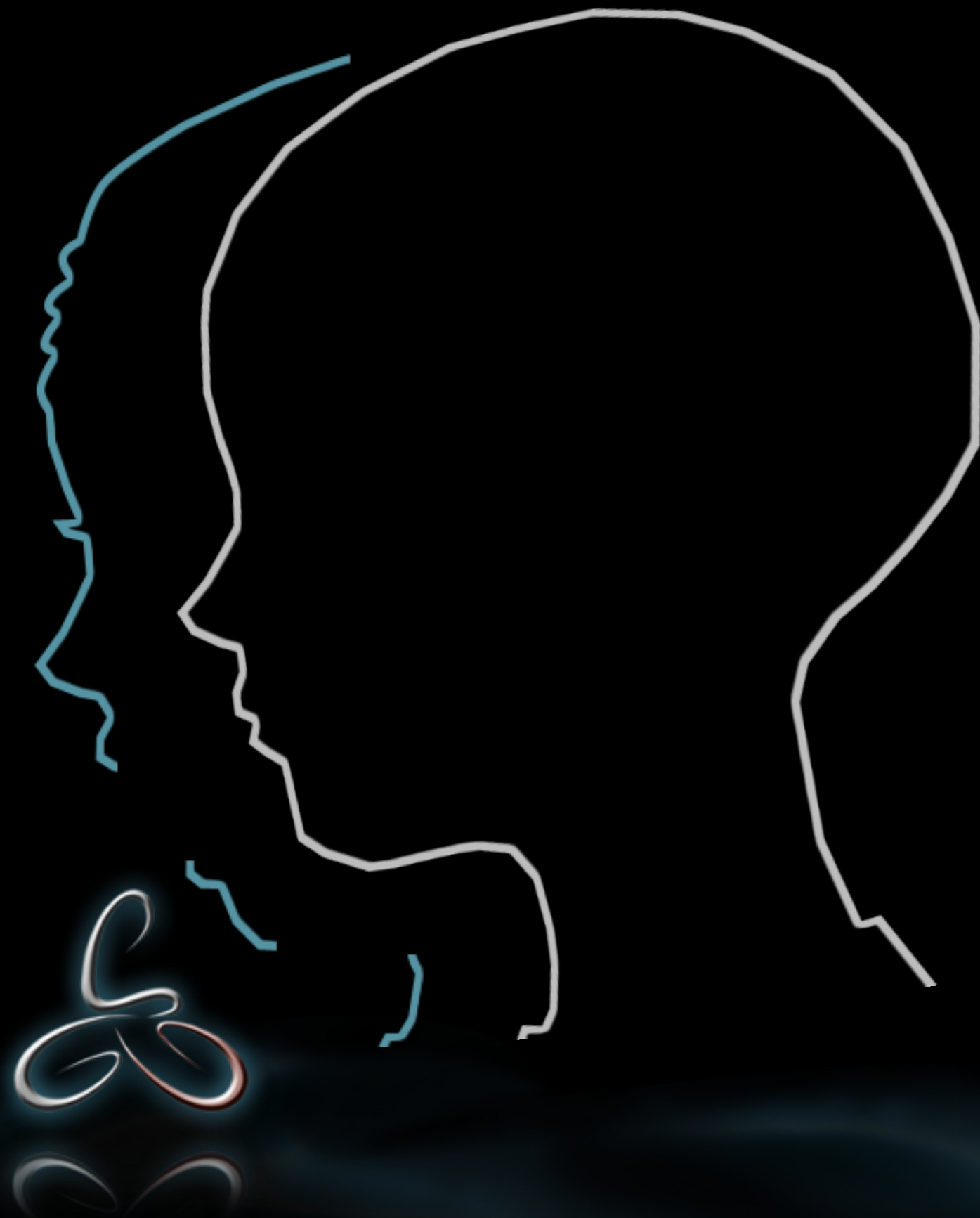
Robust Non-Rigid ICP



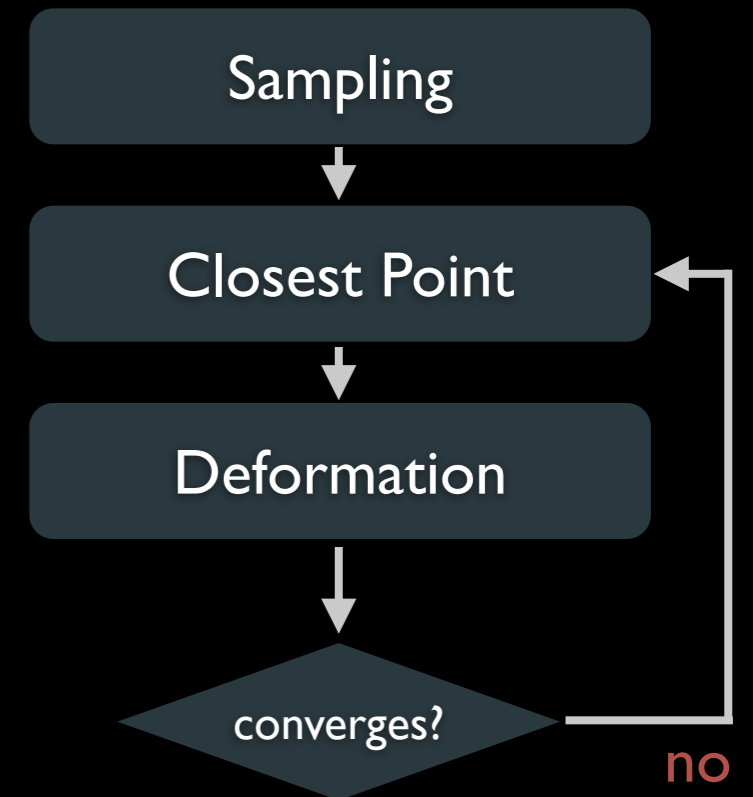
Non-Rigid ICP



Robust Non-Rigid ICP



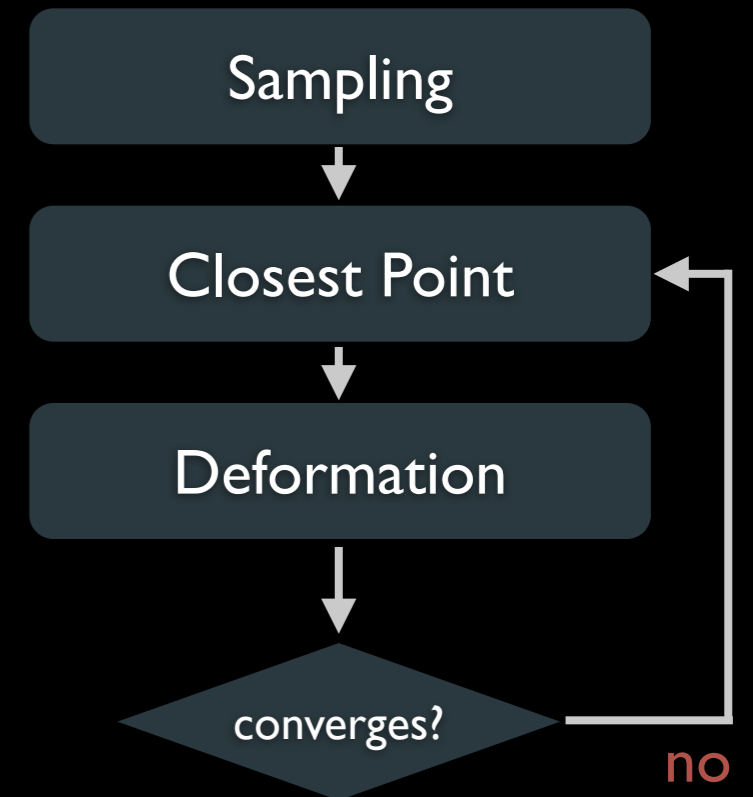
Non-Rigid ICP



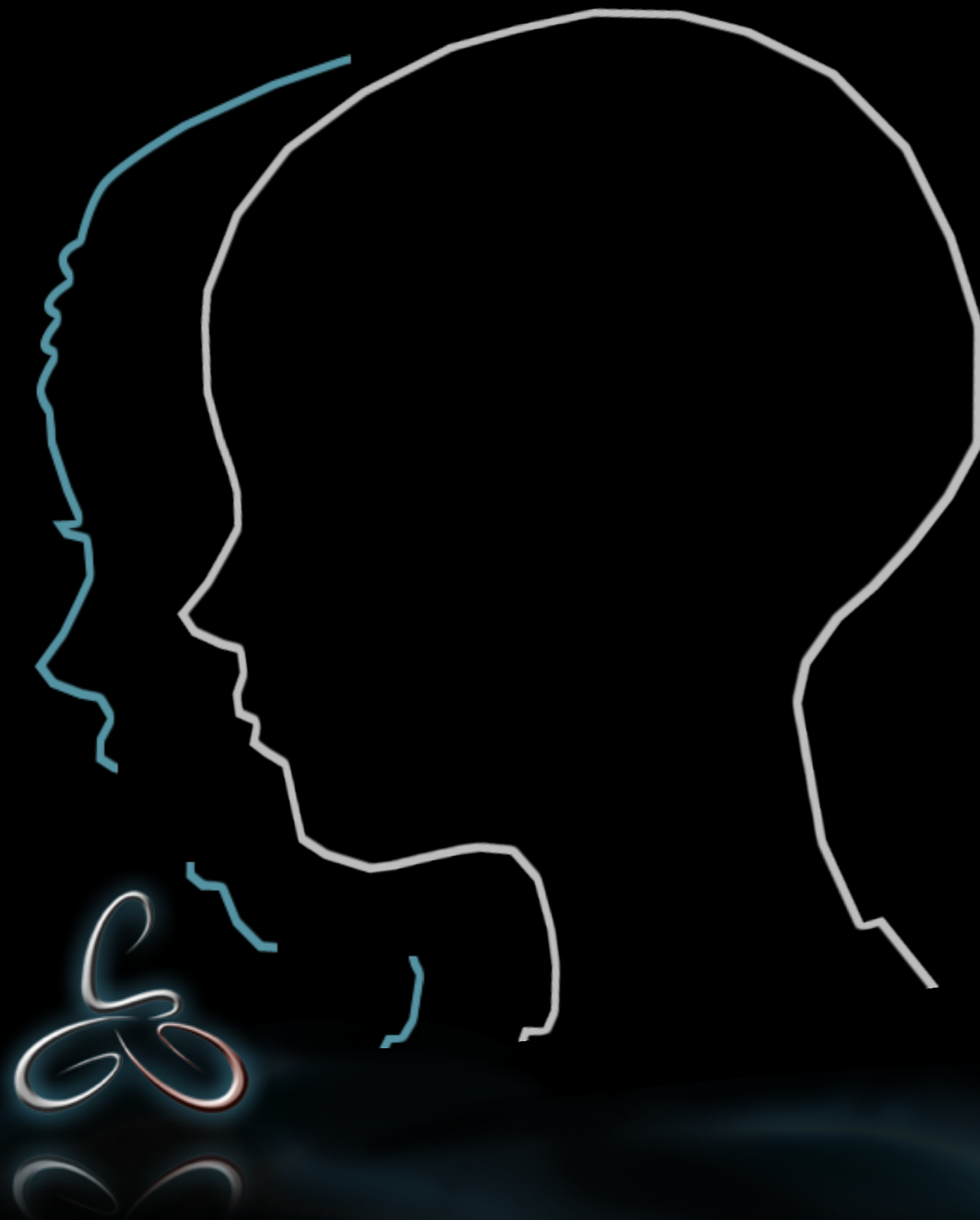
Robust Non-Rigid ICP



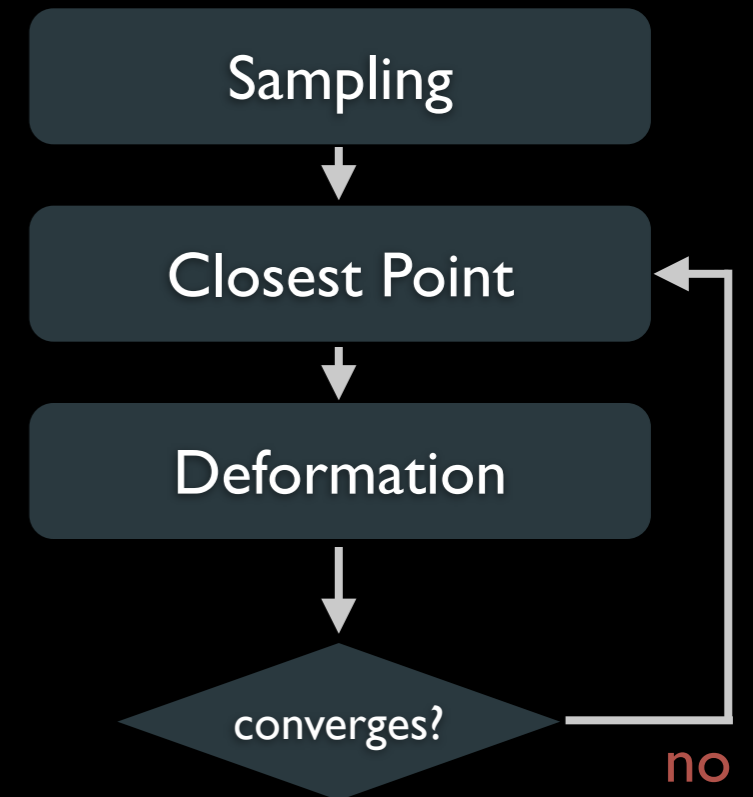
Non-Rigid ICP



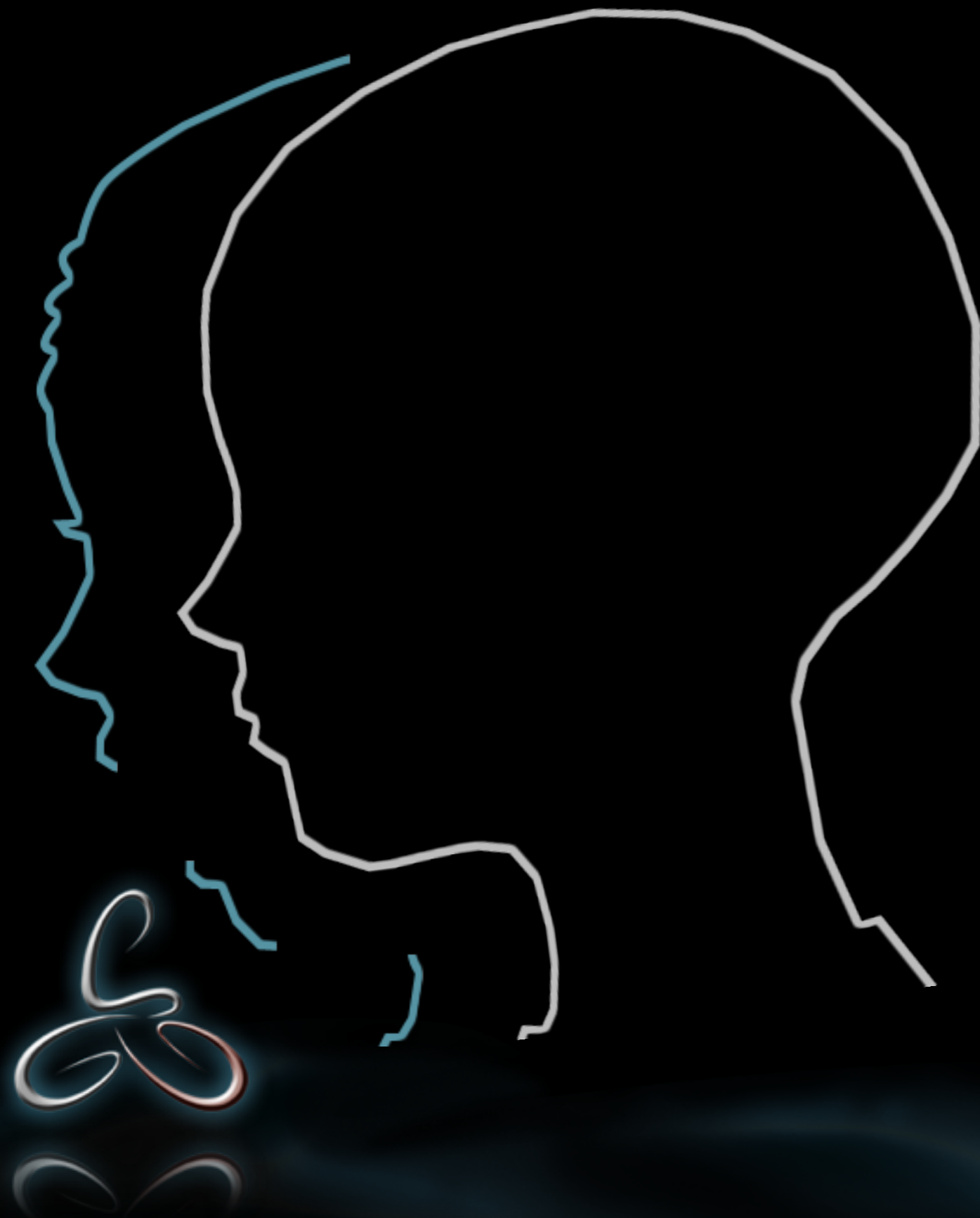
Robust Non-Rigid ICP



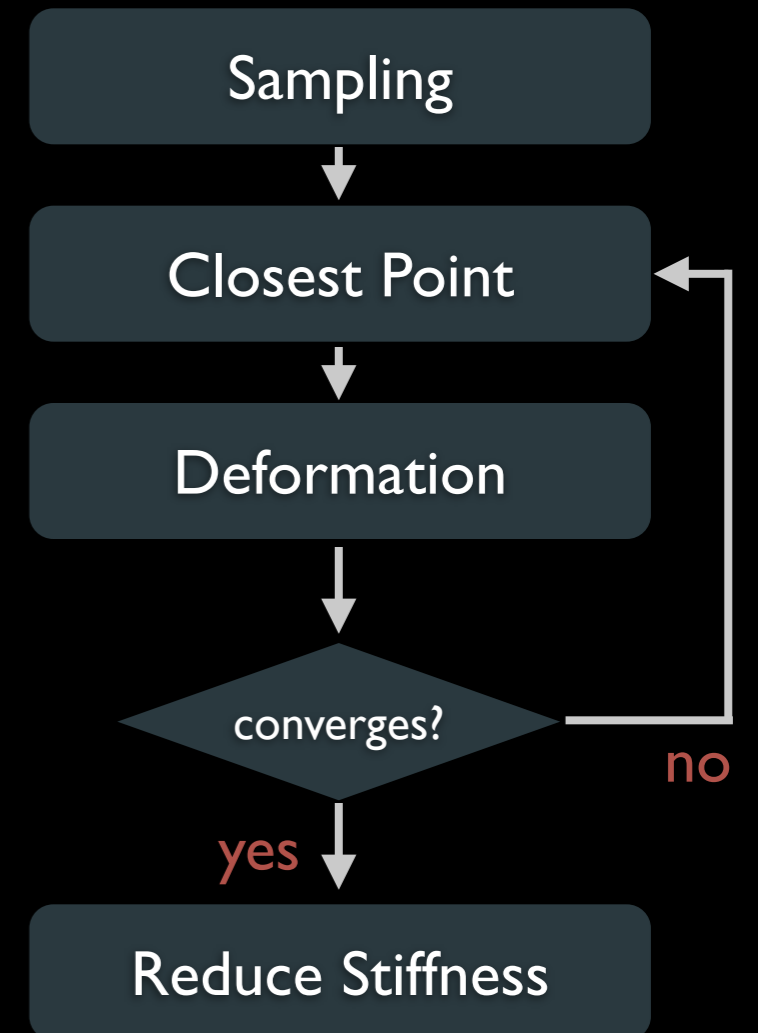
Non-Rigid ICP



Robust Non-Rigid ICP



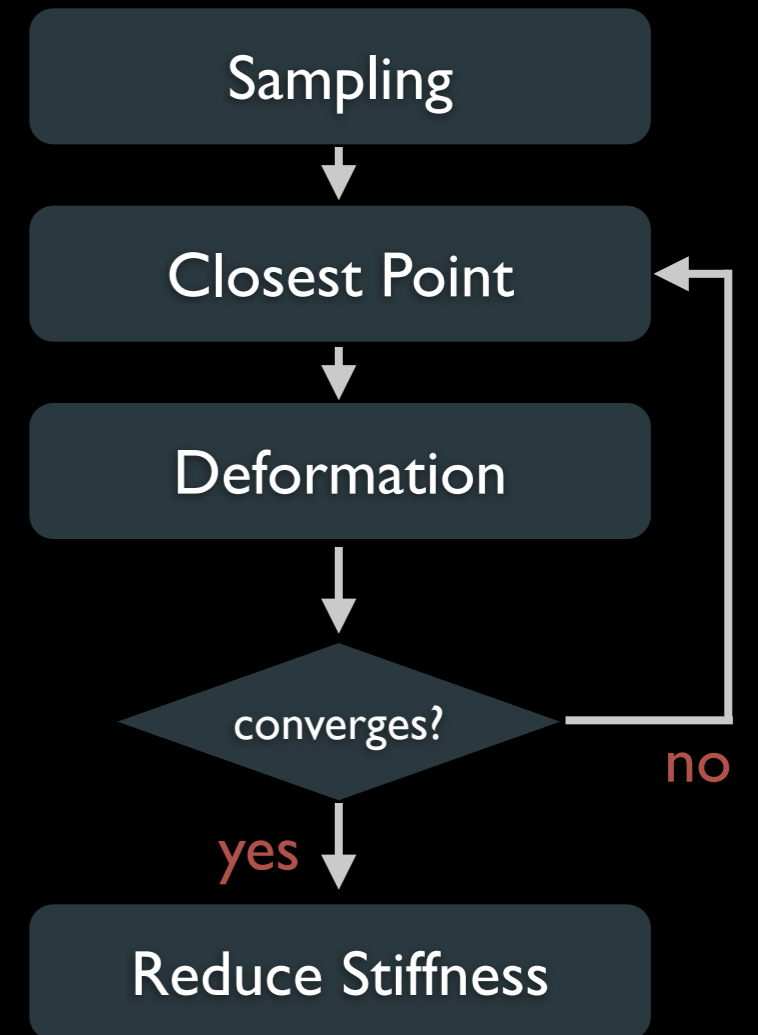
Non-Rigid ICP



Robust Non-Rigid ICP



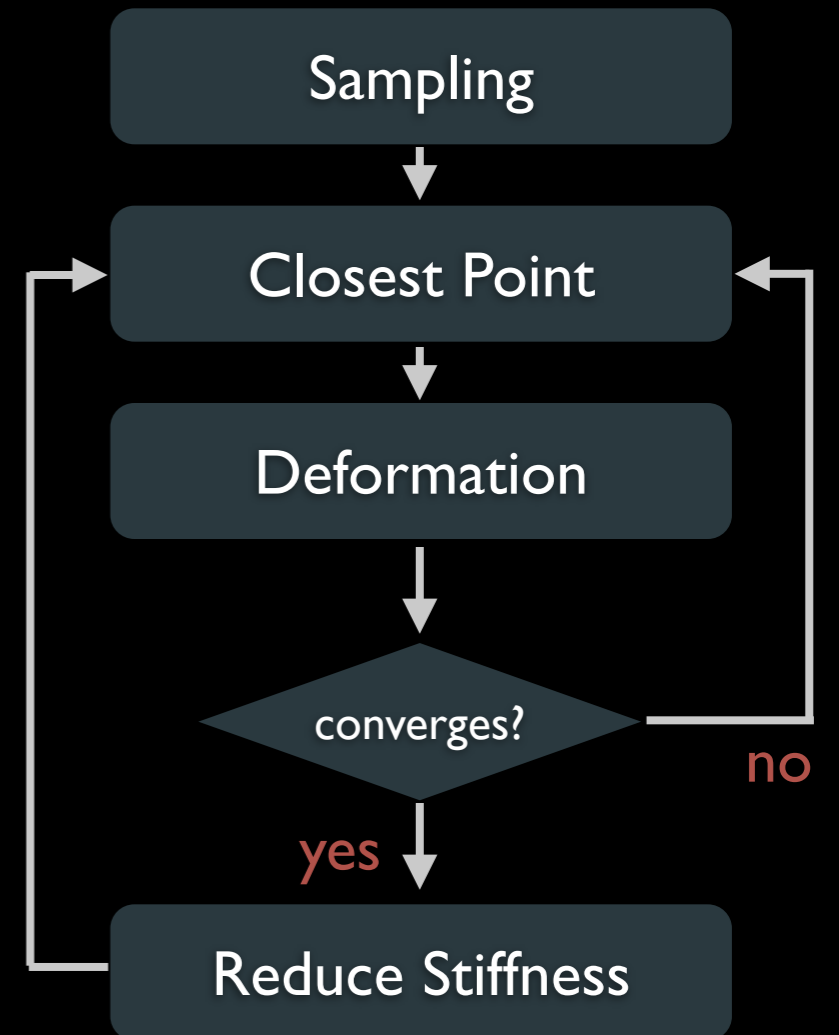
Non-Rigid ICP



Robust Non-Rigid ICP



Non-Rigid ICP

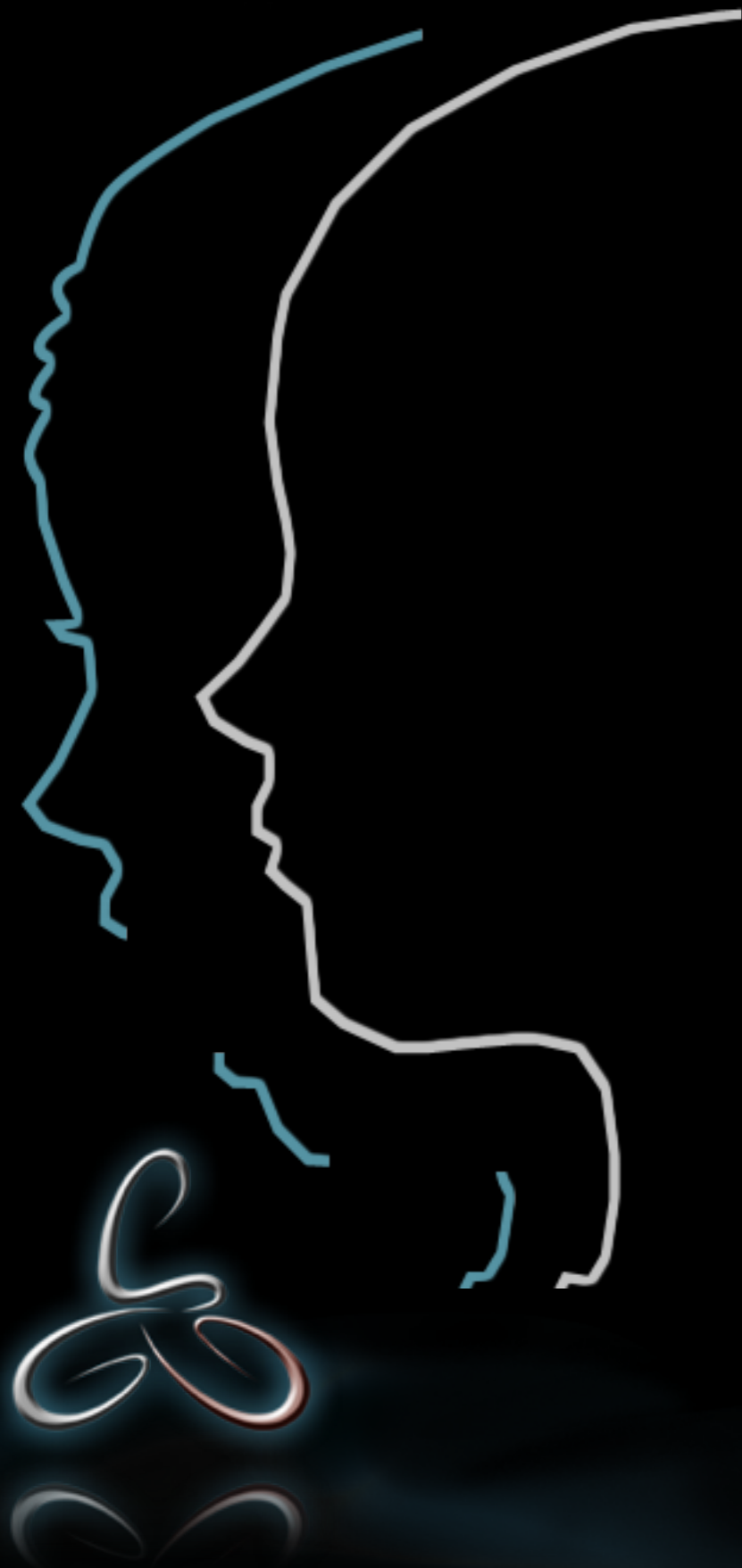


Alignment Error Minimization



Extension of [Li et al. '08]

Alignment Error Minimization



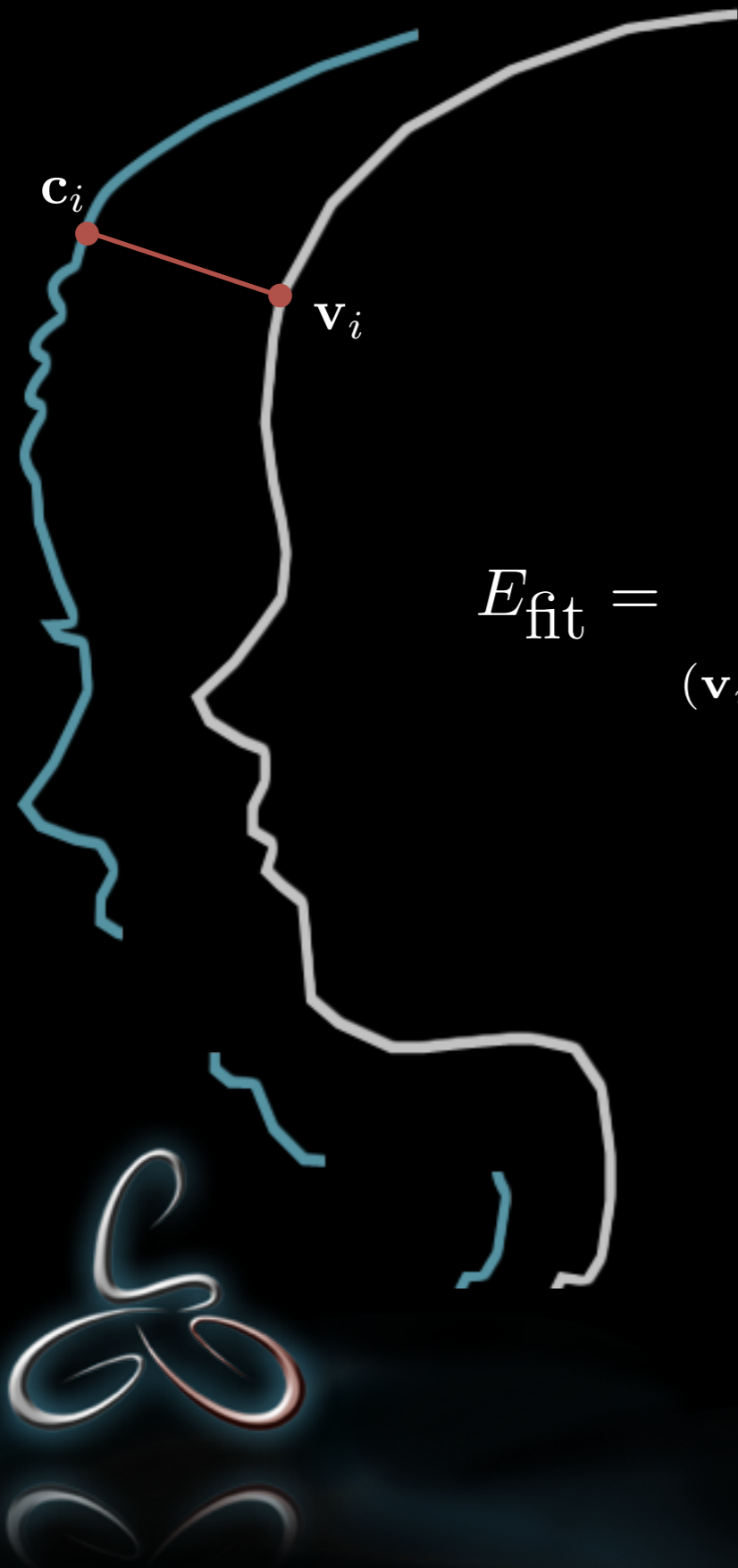
Extension of [Li et al. '08]

Alignment Error Minimization



Extension of [Li et al. '08]

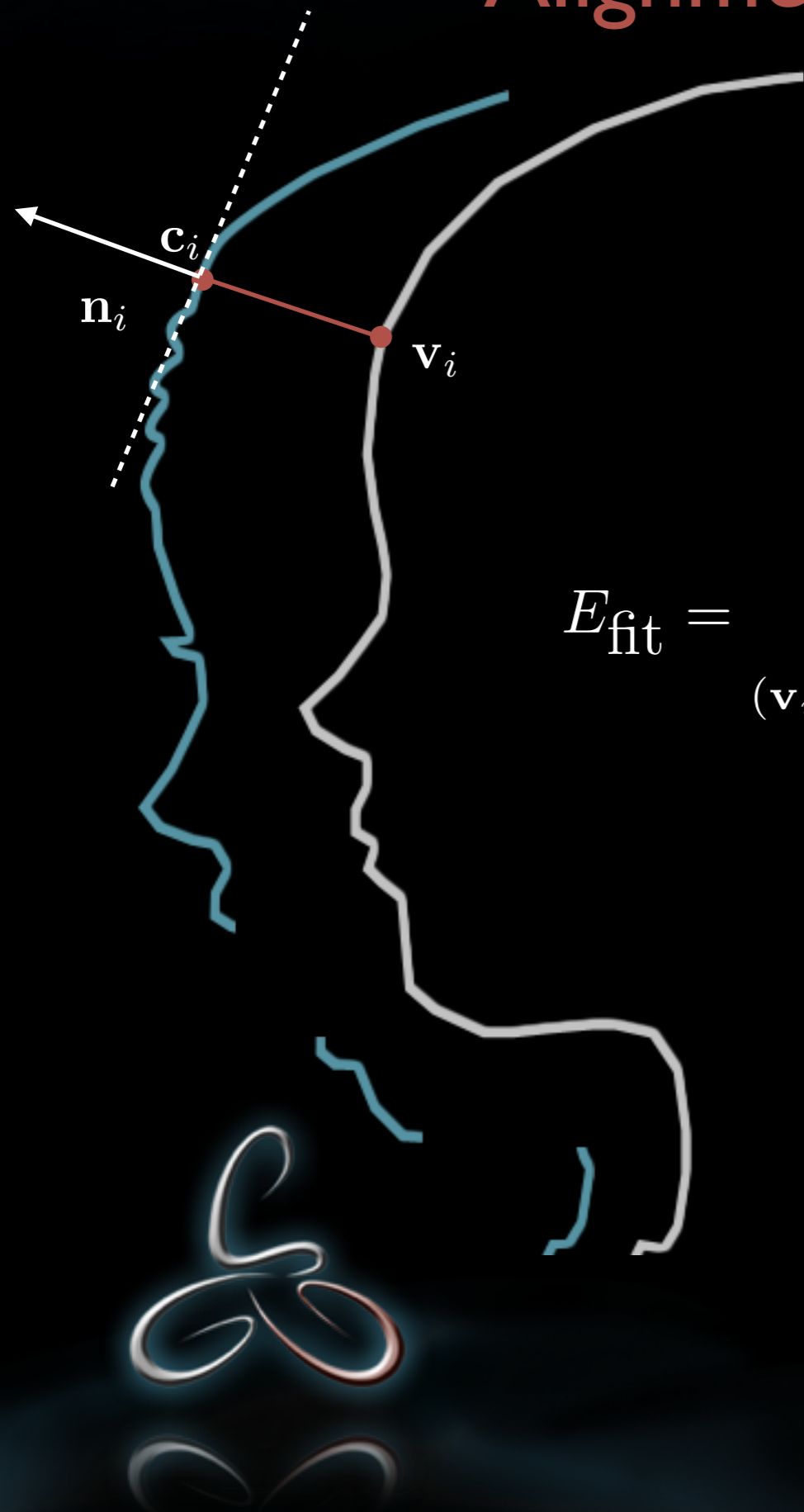
Alignment Error Minimization



Point-to-point

$$E_{\text{fit}} = \sum_{(\mathbf{v}_i, \mathbf{c}_i) \in \mathcal{C}} \alpha_{\text{point}} \|\mathbf{v}_i - \mathbf{c}_i\|^2$$

Alignment Error Minimization



Point-to-point

Point-to-plane

$$E_{\text{fit}} = \sum_{(\mathbf{v}_i, \mathbf{c}_i) \in \mathcal{C}} \alpha_{\text{point}} \|\mathbf{v}_i - \mathbf{c}_i\|^2 + \alpha_{\text{plane}} |\mathbf{n}_i^\top (\mathbf{v}_i - \mathbf{c}_i)|^2$$

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

Maximize
Rigidity



Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

Maximize
Consistency



Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}} \quad \text{Stiffness}$$



Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

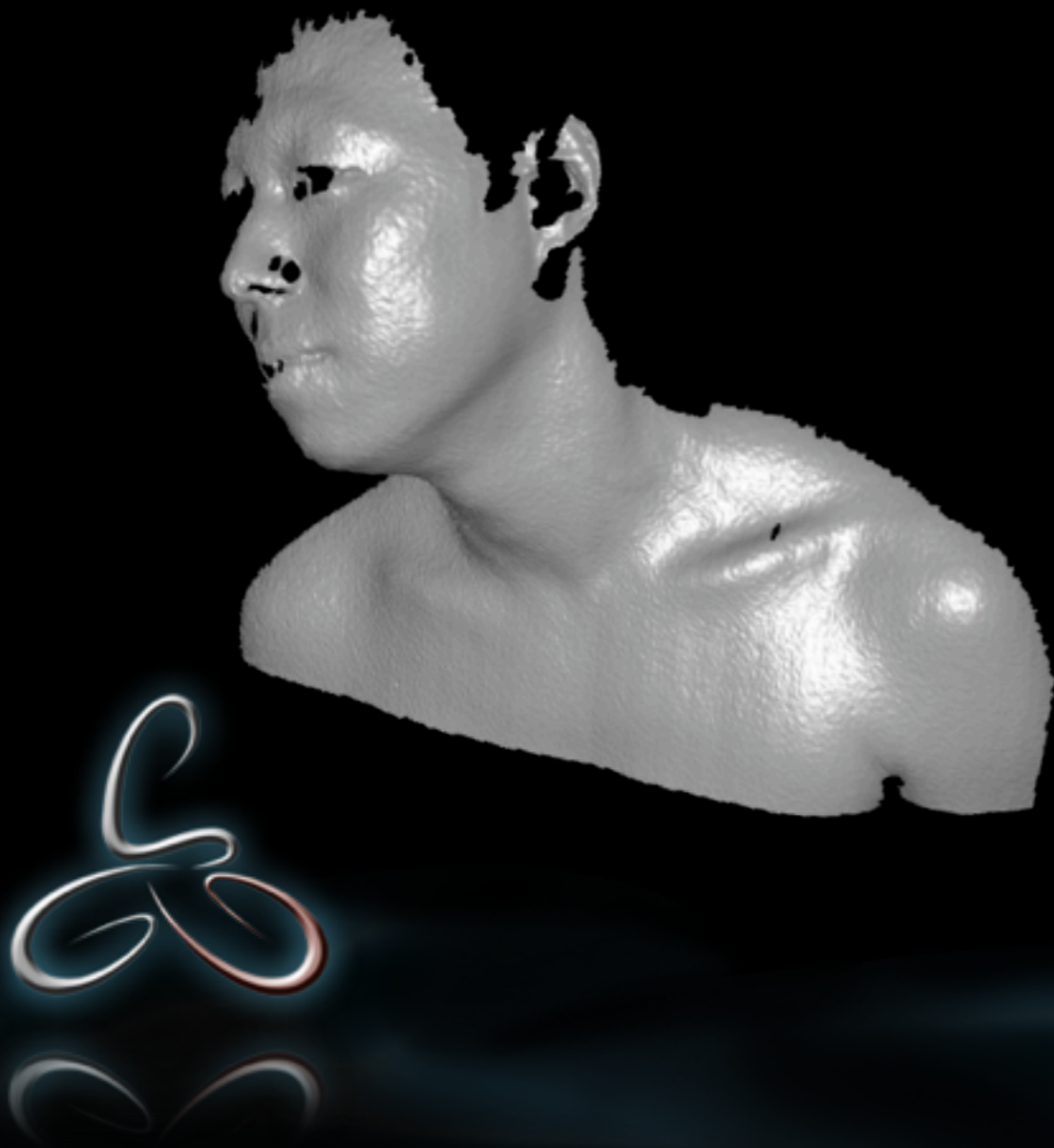


Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

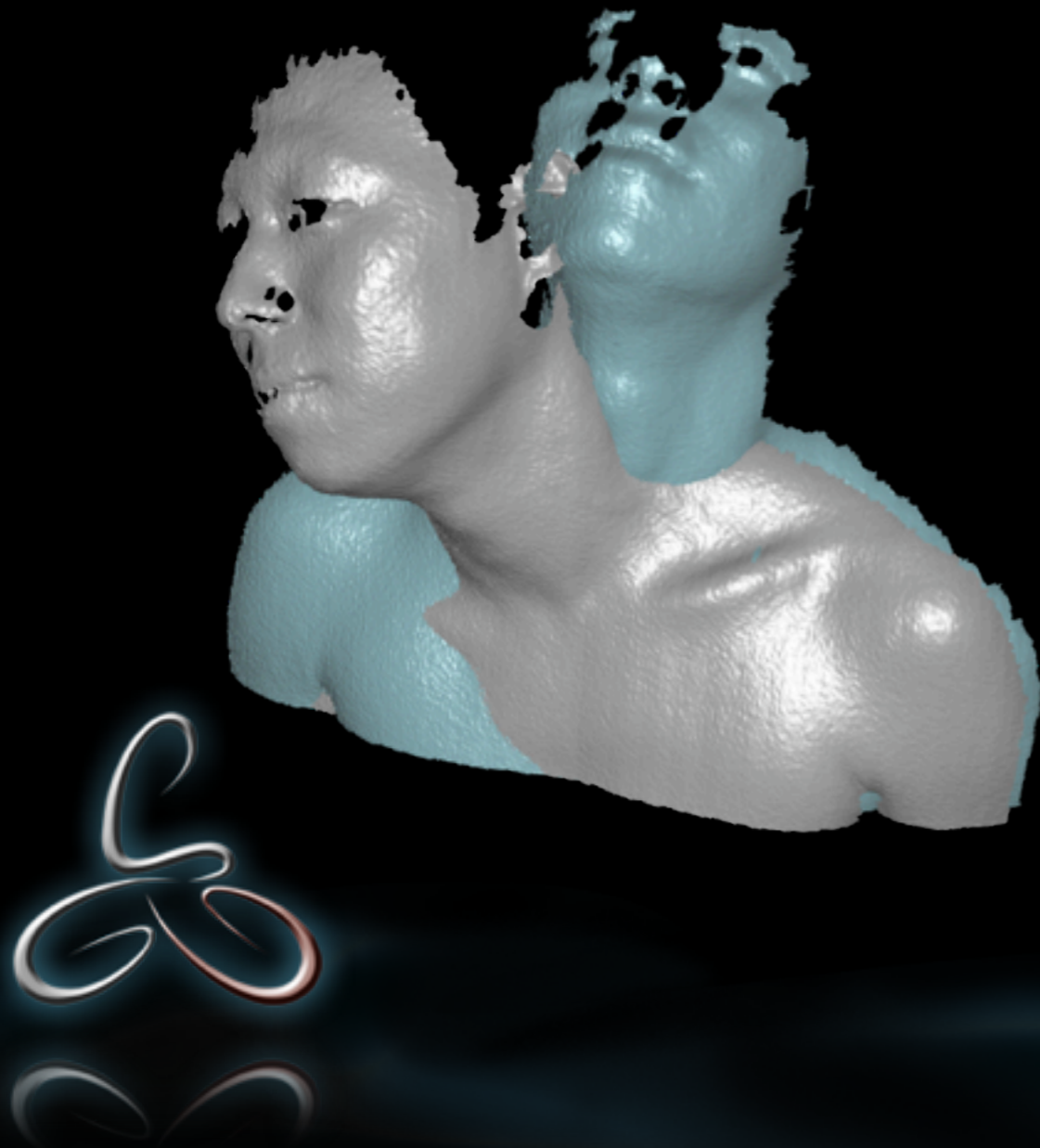


Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

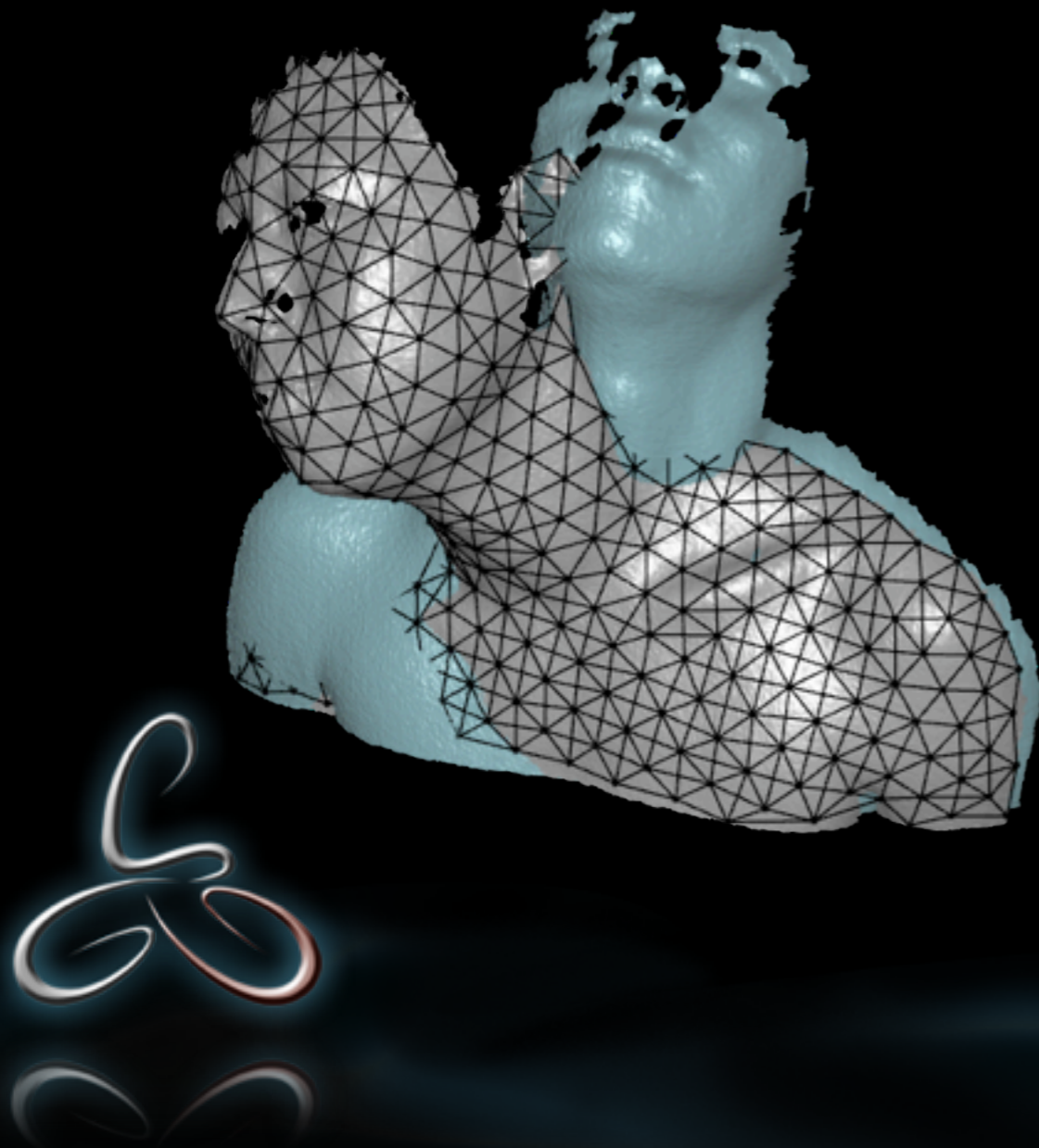


Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

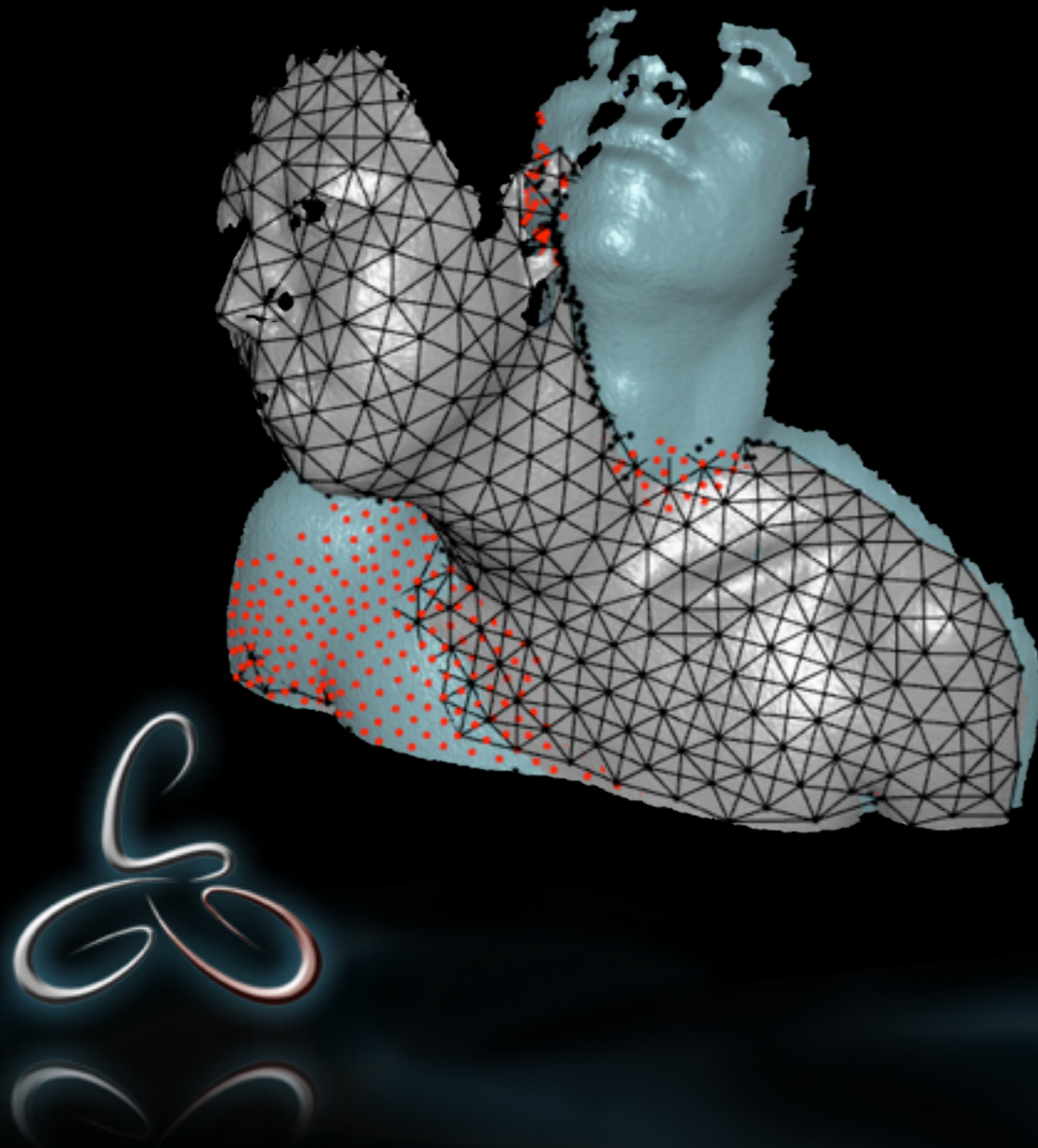


Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

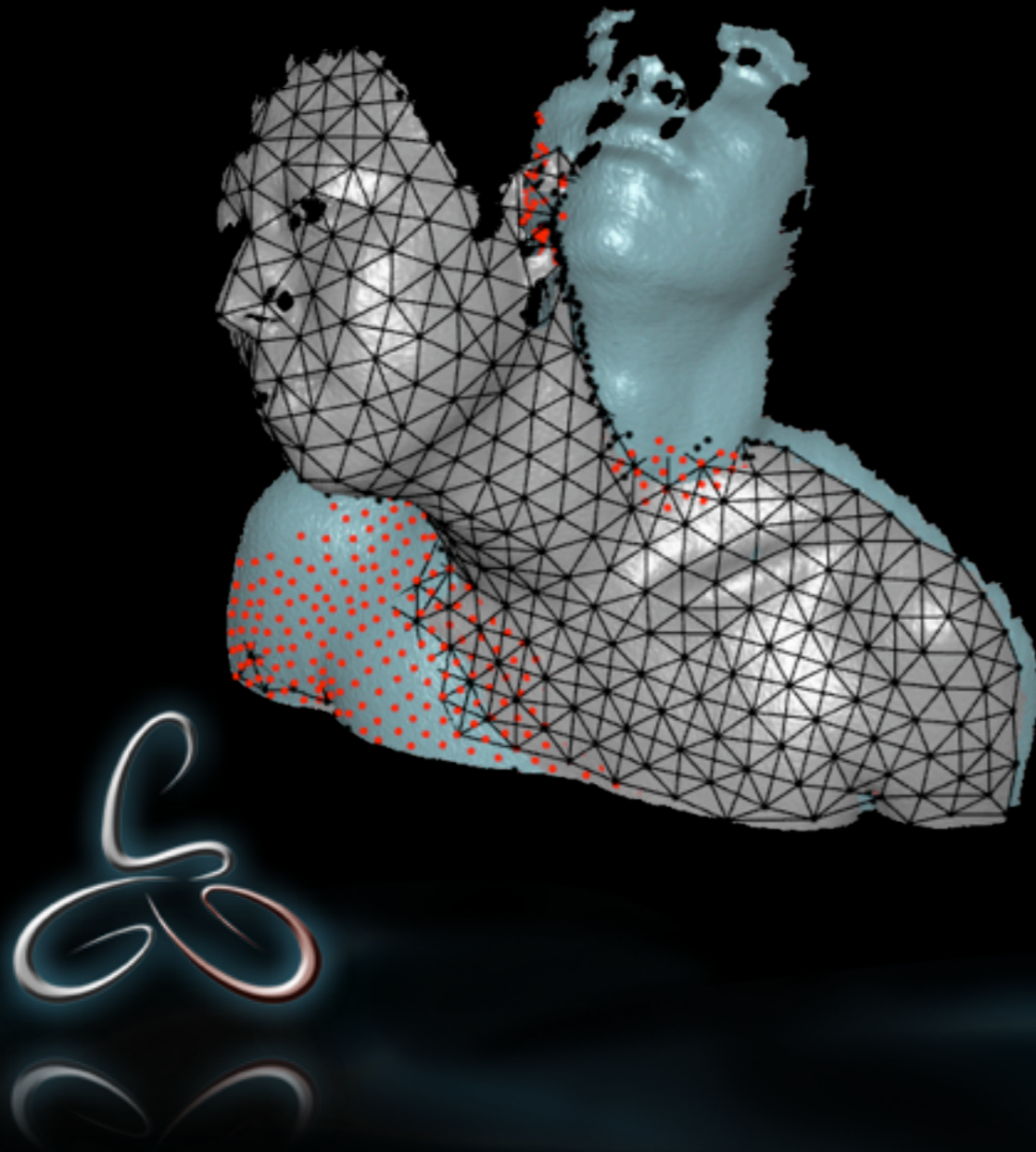


Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:



Extension of [Li et al. '08]

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate



Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate

Too many nodes:

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate

Too many nodes:

- inefficient

Optimization

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate

Too many nodes:

- inefficient
- less robust

www.hao-li.com



www.hao-li.com



Global Matching

Part I: Introduction to geometric key point detection and feature descriptors

The story so far

Problem statement

- Given pair of shapes/scans, produce an alignment

Local registration

- Solves for an alignment assuming that pose is similar or motion is small between shapes / scans
- Like “tracking” of motion in this respect

In this session: Global matching

What is Global Matching?

Problem statement

- Find the globally optimal alignment between a pair of shapes
- Search space = set of all possible correspondences
- Same sense as local minimum vs global minimum in optimization

- Don't get confused with **global registration**
 - “Global registration” is commonly used to refer to aligning *multiple scans* together to make a single shape

Local vs Global

Local Registration

- Search in space of *transformations*, minimize alignment energy
- Relatively small search space... relatively easy

vs.

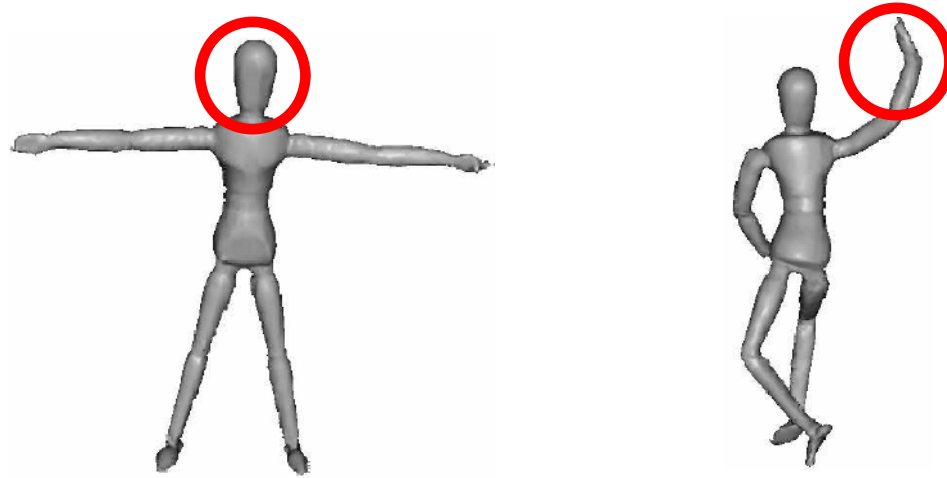
Global Registration

- Search in the space of *all possible correspondences*, minimize alignment energy
- Incredibly large search space... nearly impossible?

➔ **Features to the rescue!**

Our eyes recognize features

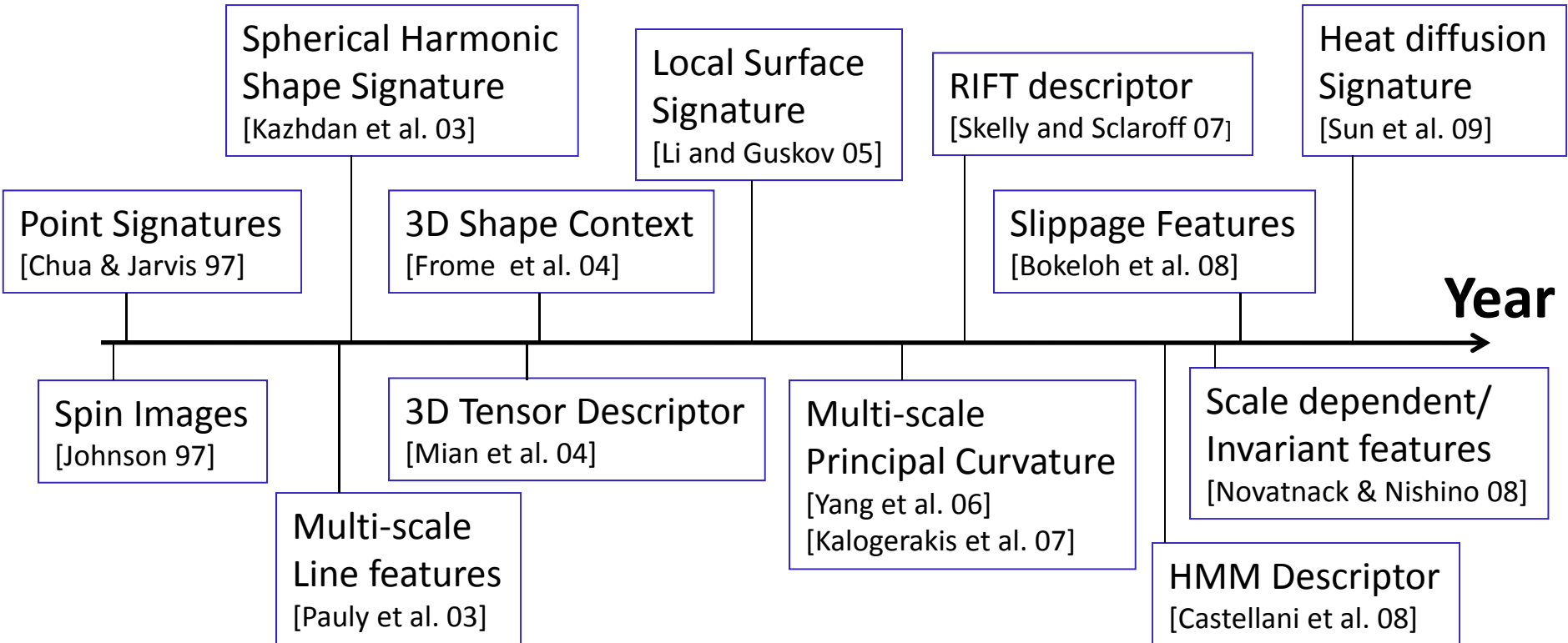
Face \neq Arm



- Why? It looks different!
- Can dramatically reduce space of possible solutions
- How can we directly compare the geometric content to recognize similarity/dissimilarity?

Types of features

Welcome to the world of feature descriptors..



- Many more exist... possibly with different objectives
 - ex) Matching whole shape vs. local patches

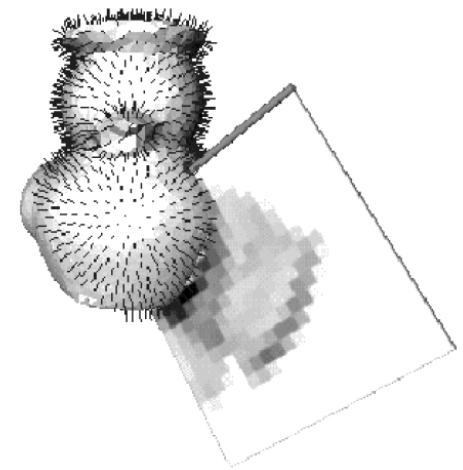
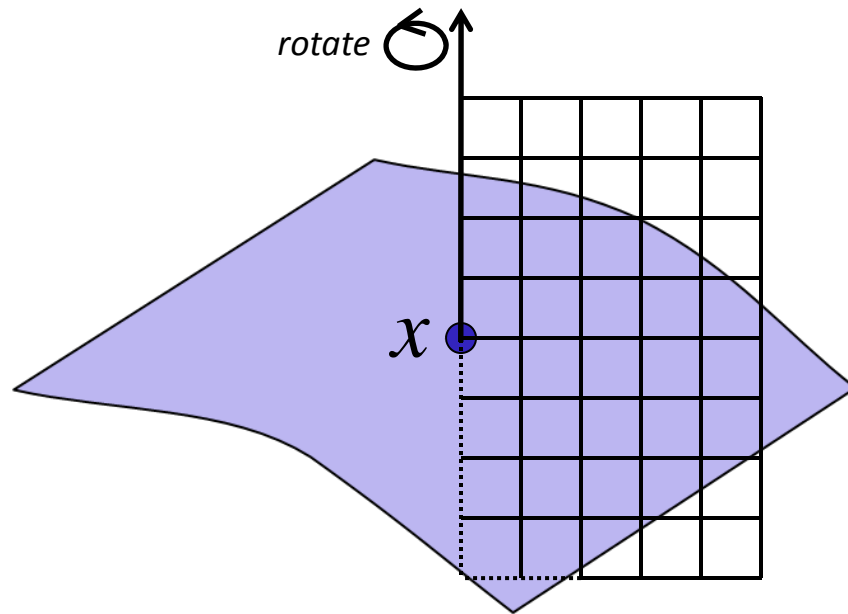
An Example: Spin Images

One of the earliest feature descriptors

- Established, simple, well analyzed
- Clearly illustrates the process of how this type of recognition works
- Also illustrates potential problems & drawbacks common to any type of feature descriptor

Spin Image Construction

- Converts a local patch of geometry into an image, which we can directly compare to determine similarity

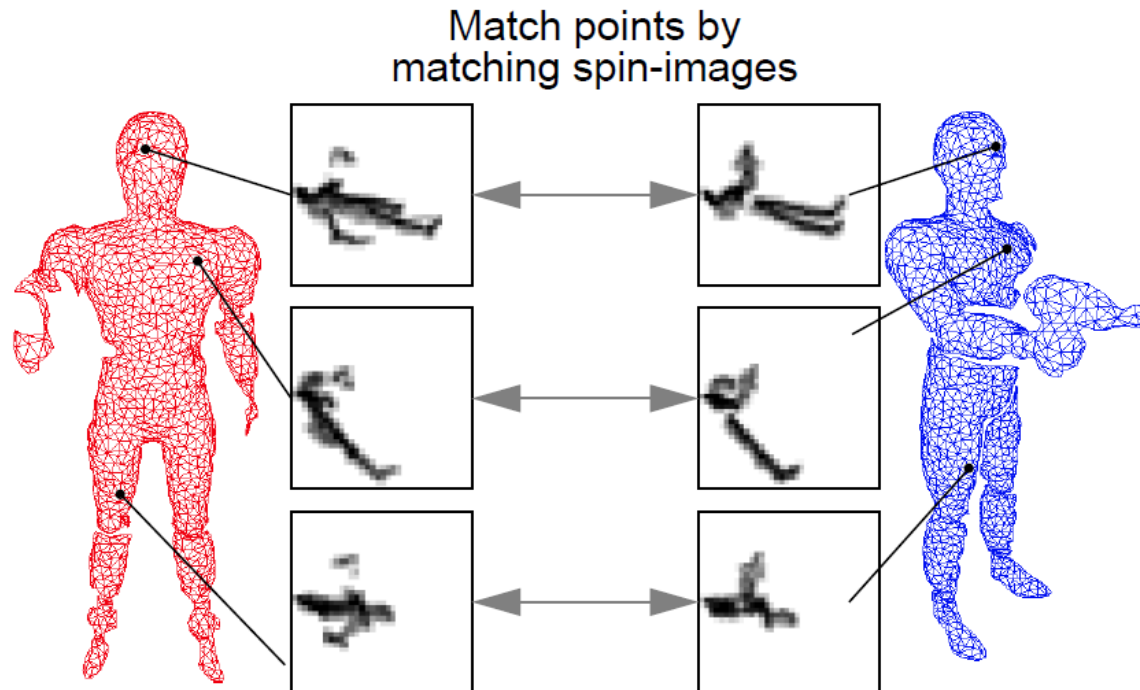


Images from [Johnson 97]

Spin Image Matching

Compare images directly to obtain similarity score

- Linear correlation coefficient \rightarrow Similarity measure
- Compute only in “overlap”: when both bins have a value

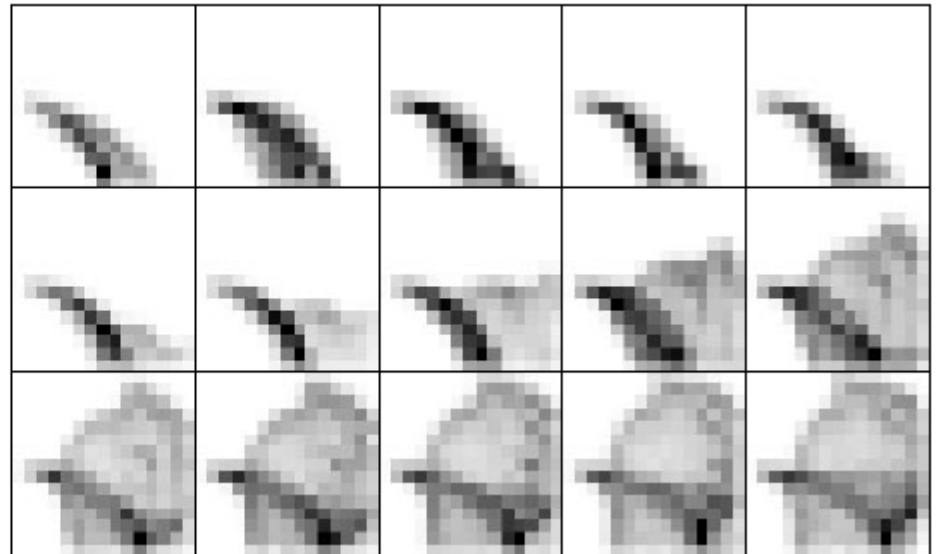
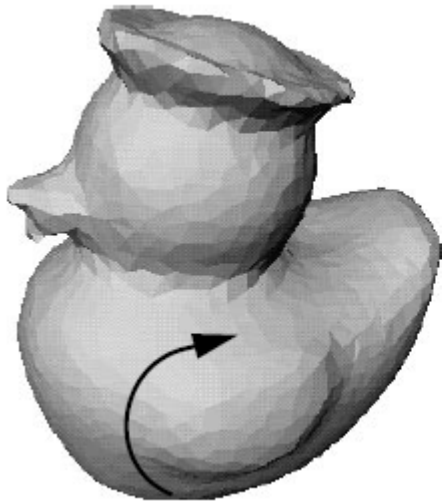


Images from [Johnson 97]

Compressing Spin Images

Spin images from the same model are similar

- Reduce redundancy with PCA compression
- Save space and matching time



Images from [Johnson 97]

Problem #1: False positive/negative

False positive

- Saying that two points match when in fact they don't

False negative

- Saying that two points don't match when in fact they do

Aka “noise” or “outliers”

- Occurs with any type of descriptor

Problem #2: Parameter Selection

Examples of parameters in spin images

- Bin size
- Image width
- Support angle
- Mesh resolution

How to pick the best parameters?

- Fortunately well analyzed for spin images
- Not so well-analyzed for others

Problem #3: Non-unique patches

What to do in flat/spherical/cylindrical regions?

- In this case, the region is not “unique” or distinctive
- Doesn't make sense to compare such regions..
- Or does it?
 - Increasing the scale/support
- Multi-scale features, select scale automatically
- “Global” features – ex) heat diffusion signature

Conclusion

Feature descriptors

- Very useful for narrowing down search space
- Does not solve the problem completely
- Additional optimization in the (reduced) search space is needed → explored in the next few talks!

Geometric Registration for Deformable Shapes

3.2 Isometric Matching and Quadratic Assignment

Quadratic Assignment · Spectral Matching · MRF Model

Overview and Motivation

Global Isometric Matching

Goal

- We want to compute correspondences between deformable shape
- *Global algorithm*, no initialization

Global Isometric Matching

Approach & Problems

- Consistency criterion: global isometry

Problem

- How to find globally consistent matches?

Model

- Quadratic assignment problem
 - General QA-problem is NP-hard
 - But it turns out: solution can usually be computed in polynomial time (using a randomized technique, later)

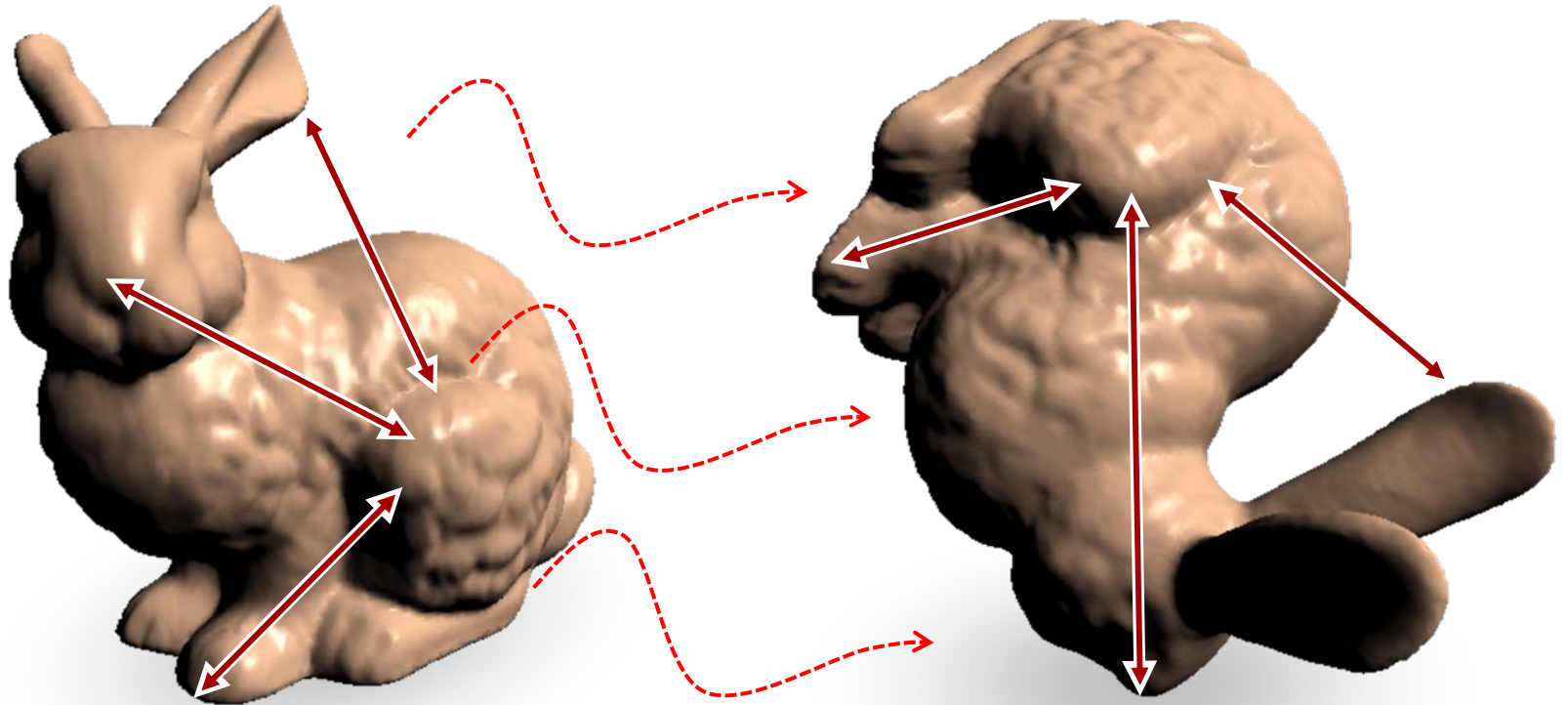
Isometric Matching

(vs. extrinsic matching)

Invariants

Rigid Matching

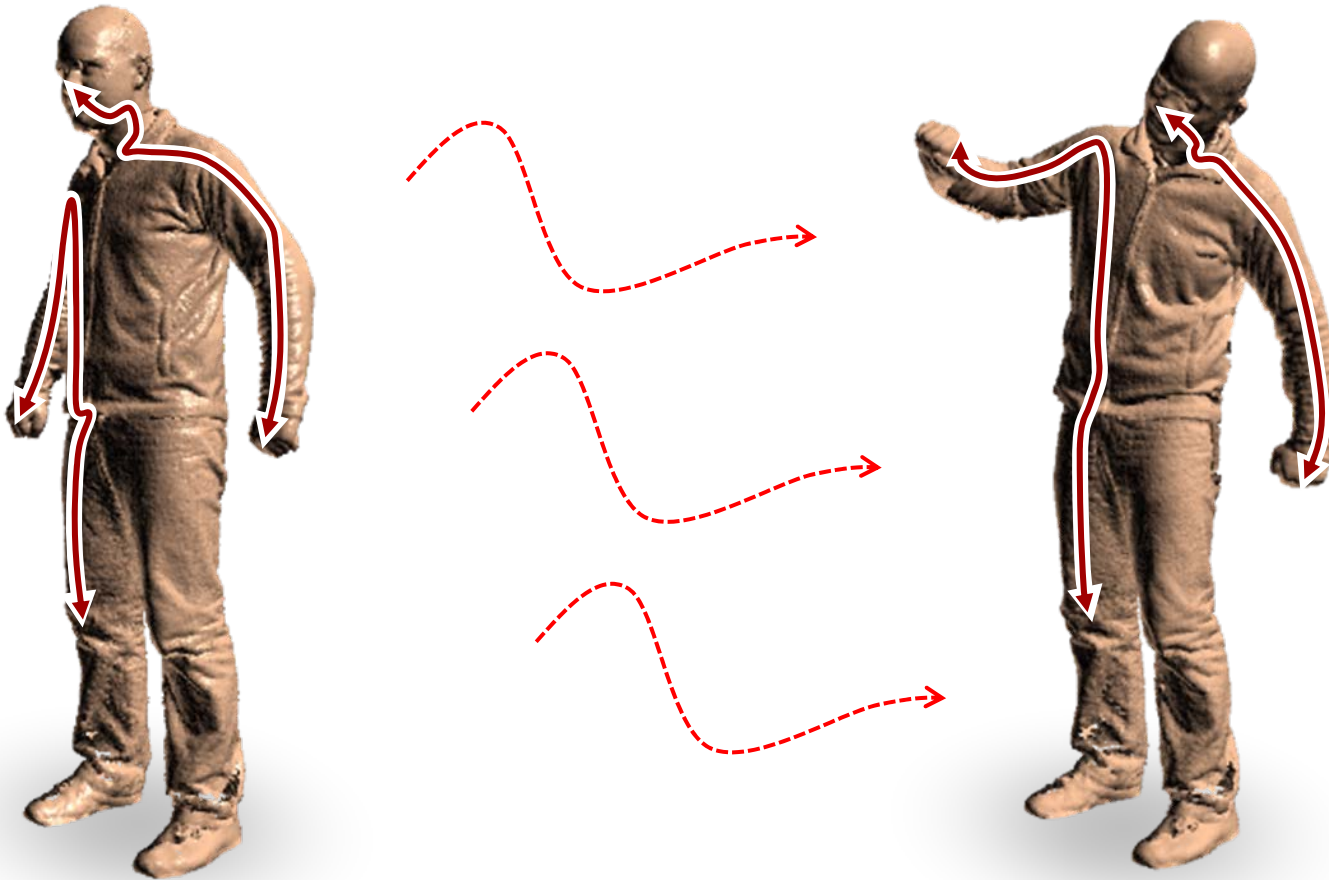
- Invariants: All Euclidean distances are preserved



Invariants

Intrinsic Matching

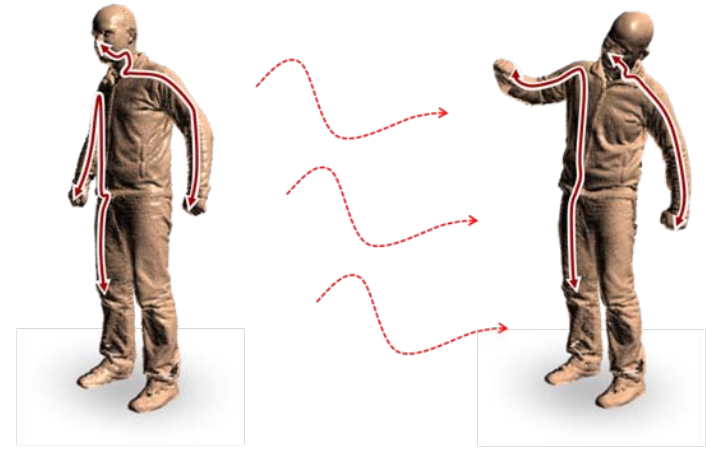
- Invariants: All geodesic distances are preserved



Invariants

Intrinsic Matching

- Preservation of geodesic distances („intrinsic distances“)
- Approximation
 - Cloth is almost unstretchable
 - Skin does not stretch a lot
 - Most live objects show approximately isometric surfaces
- Accepted model for deformable shape matching
 - In cases where one subject is presented in different poses
 - Across different subjects: Other assumptions necessary
 - Then: global matching is an open problem



Feature Based Matching

Quadratic Assignment Model

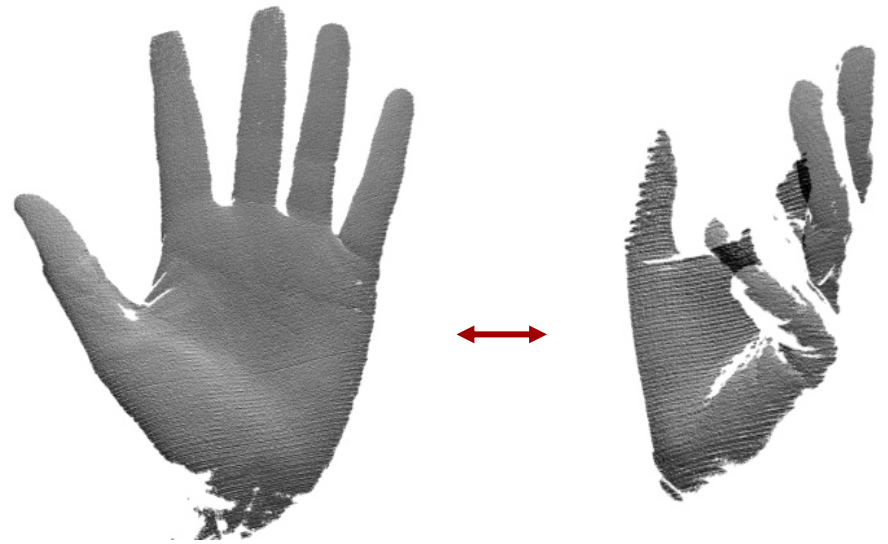
Problem Statement

Deformable Matching

- Two shapes: original, deformed
- How to establish correspondences?
- Looking for global optimum
 - Arbitrary pose

Assumption

- Approximately isometric deformation

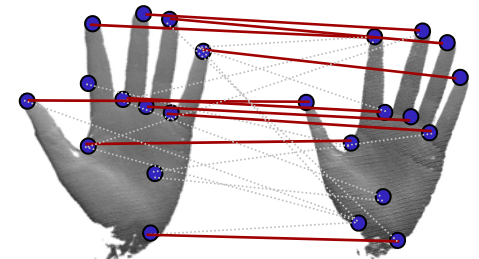
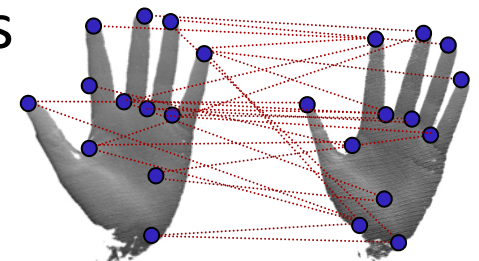
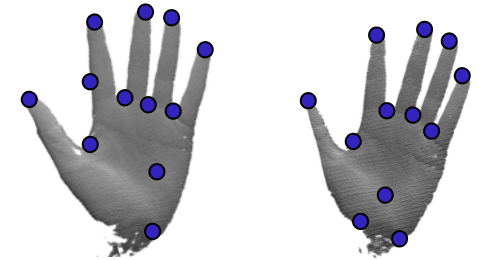


[data set: S. König, TU Dresden]

Algorithm

Feature-Matching

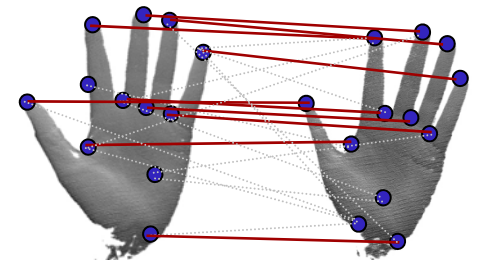
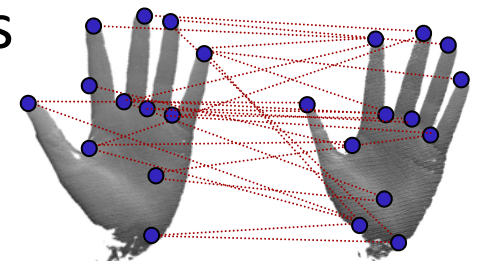
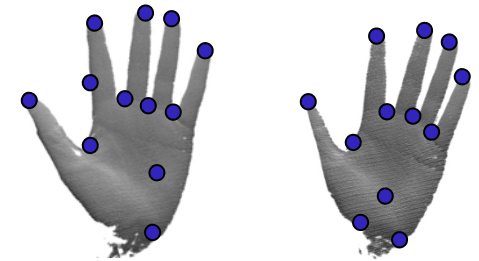
- Detect feature points
- Local matching: potential correspondences
- Global filtering: correct subset



Algorithm

Feature-Matching

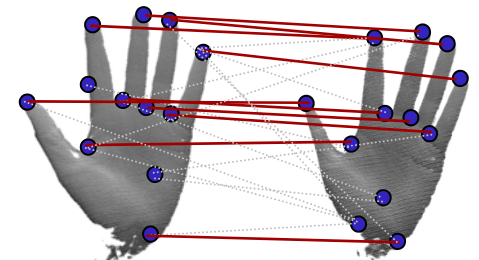
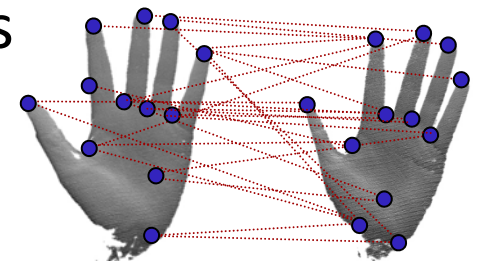
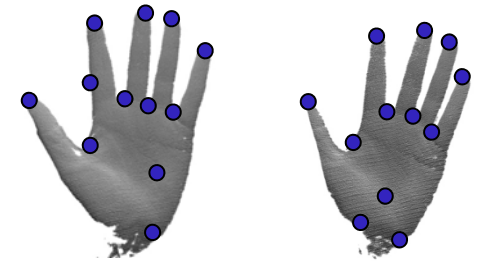
- Detect feature points
 - Locally unique points
 - Such as: maxima of Gaussian curvature
 - E.g.: Geometric MLS-SIFT Features
- Local matching: potential correspondences
- Global filtering: correct subset



Algorithm

Feature-Matching

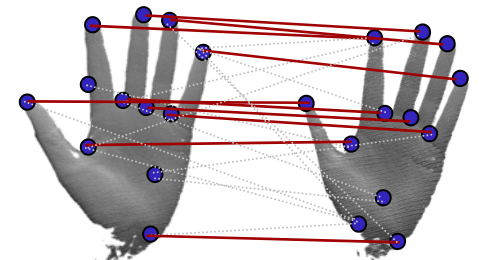
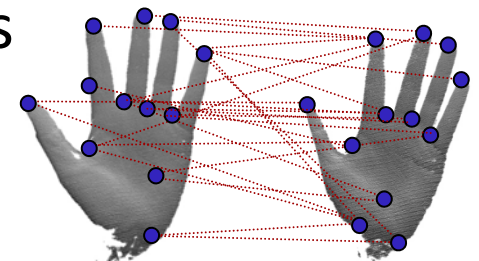
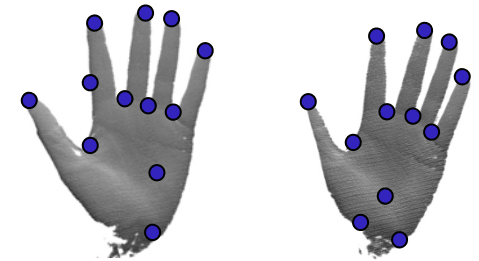
- Detect feature points
 - Locally unique points
 - Such as: maxima of Gaussian curvature
 - E.g.: Geometric MLS-SIFT Features
- Local matching: potential correspondences
 - Descriptors
 - E.g. curvature histograms
- Global filtering: correct subset



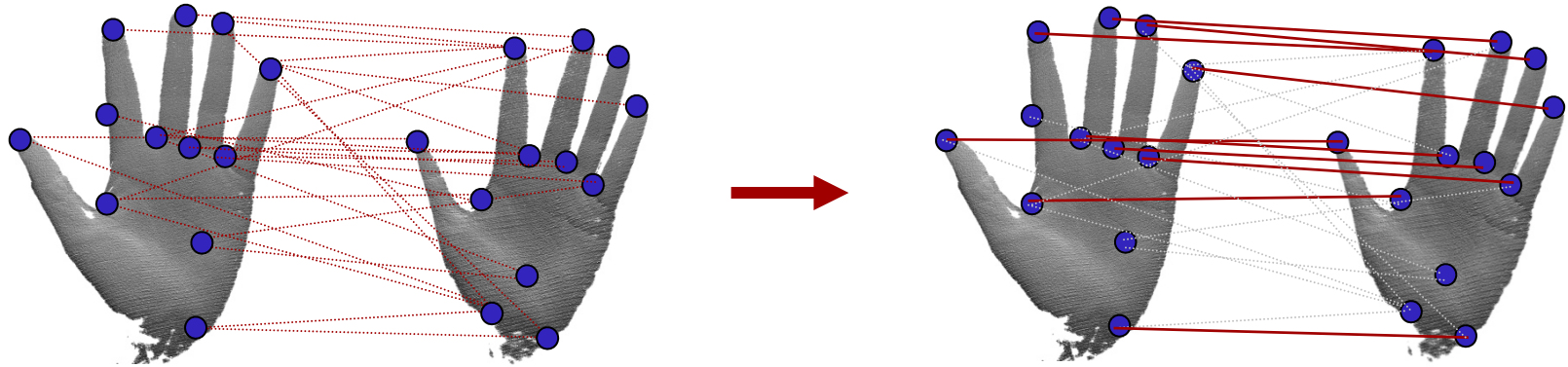
Algorithm

Feature-Matching

- Detect feature points
 - Locally unique points
 - Such as: maxima of Gaussian curvature
 - E.g.: Geometric MLS-SIFT Features
- Local matching: potential correspondences
 - Descriptors
 - E.g. curvature histograms
- Global filtering: correct subset
 - Quadratic assignment
 - Spectral relaxation [Leordeanu et al. 05]
 - RANSAC



Quadratic Assignment



Most difficult part: Global filtering

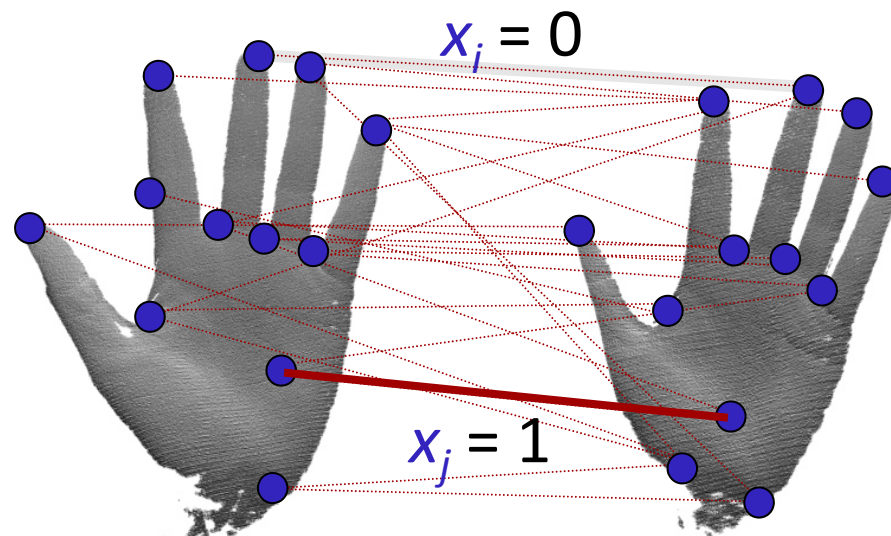
- Find a consistent subset
- Pairwise consistency:
 - Correspondence pair must preserve intrinsic distance
- Maximize number of pairwise consistent pairs
 - Quadratic assignment (in general: NP-hard)

Quadratic Assignment Model

Quadratic Assignment

- n potential correspondences
- Each one can be turned on or off
- Label with variables x_i
- Compatibility score:

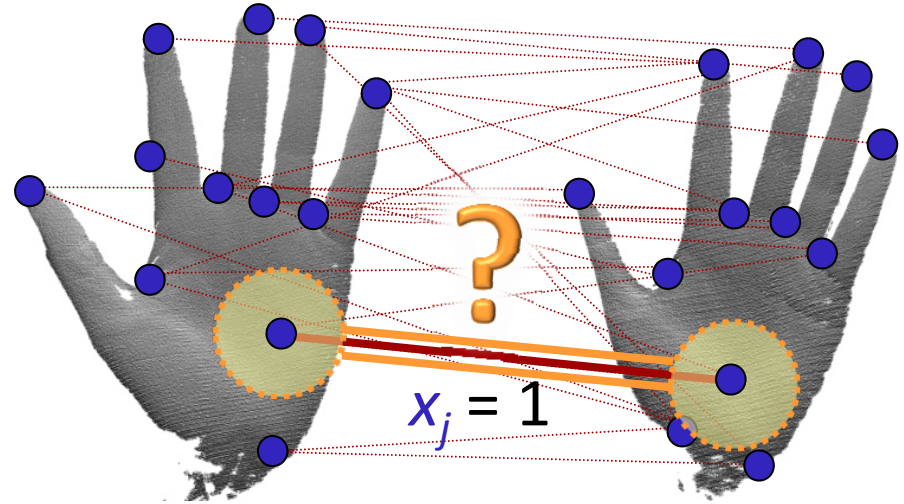
$$P^{(match)}(x_1, \dots, x_n) = \prod_{i=1}^n P_i^{(single)} \prod_{i,j=1}^n P_{i,j}^{(compatible)}, x_i \in \{0,1\}$$



Quadratic Assignment Model

Quadratic Assignment

- Compatibility score:
 - **Singeltons:**
Descriptor match

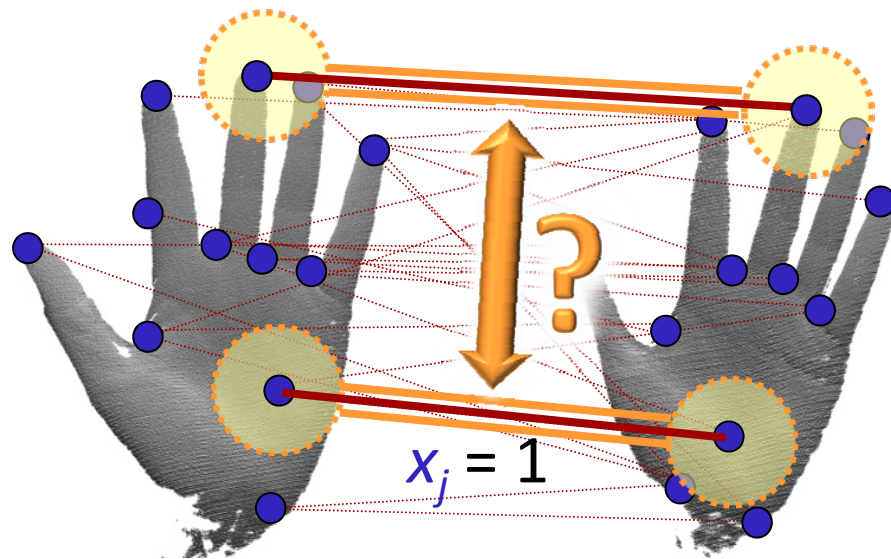


$$P^{(match)}(x_1, \dots, x_n) = \prod_{i=1}^n P_i^{(single)} \prod_{i,j=1}^n P_{i,j}^{(compatible)}, x_i \in \{0,1\}$$

Quadratic Assignment Model

Quadratic Assignment

- Compatibility score:
 - **Singeltons:**
Descriptor match
 - **Doubles:**
Compatibility



$$P^{(match)}(x_1, \dots, x_n) = \prod_{i=1}^n P_i^{(single)} \prod_{i,j=1}^n P_{i,j}^{(compatible)}, x_i \in \{0,1\}$$

Quadratic Assignment Model

Quadratic Assignment

- Matrix notation:

$$\begin{aligned} P^{(match)}(x_1, \dots, x_n) &= \prod_{i=1}^n P_i^{(single)} \prod_{i,j=1}^n P_{i,j}^{(compatible)} \\ &= \mathbf{x}\mathbf{s} + \mathbf{x}^T \mathbf{D}\mathbf{x} \end{aligned}$$

- Quadratic scores are encoded in Matrix **D**
- Linear scores are encoded in Vector **s**
- Task: find optimal binary vector **x**

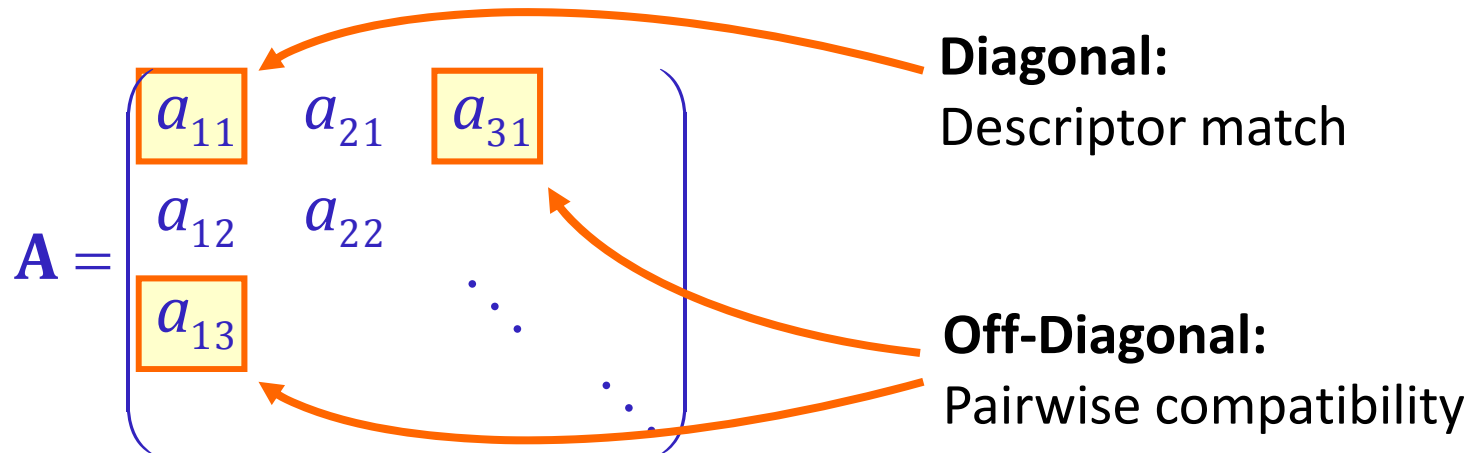
Spectral Matching

Approximate Quadratic Assignment

Spectral Matching

Simple & Effective Approximation:

- Spectral matching [Leordeanu & Hebert 05]
- Form compatibility matrix:



All entries within [0..1]
= [no match...perfect match]

Spectral Matching

Approximate largest clique:

- Compute eigenvector with largest eigenvalue
- Maximizes Rayleigh quotient:

$$\operatorname{arg\,max} \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\|\mathbf{x}\|^2}$$

- “Best yield” for bounded norm
 - The more consistent pairs (rows of 1s), the better
 - Approximates largest clique
- Implementation
 - For example: power iteration

Spectral Matching

Postprocessing

- Greedy quantization
 - Select largest remaining entry, set it to 1
 - Set all entries to 0 that are not pairwise consistent with current set
 - Iterate until all entries are quantized

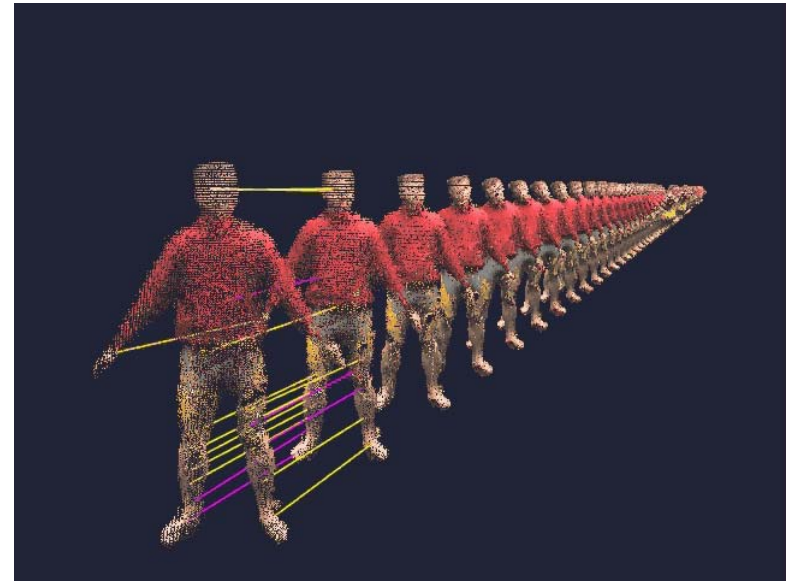
In practice...

- This algorithm turns out to work quite well.
- Very easy to implement
- Limited to (approx.) quadratic assignment model

Spectral Matching Example

Application to Animations

- **Feature points:**
Geometric MLS-SIFT features [Li et al. 2005]
- **Descriptors:**
Curvature & color ring histograms
- **Global Filtering:**
Spectral matching
- **Pairwise animation matching:**
Low precision passive stereo data



Data courtesy of C. Theobald, MPI Informatik

Markov Random Field Model


Probabilistic Interpretation

Direct MRF Approach

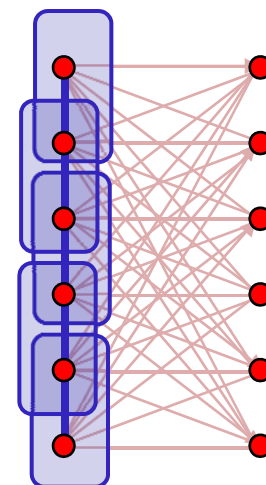
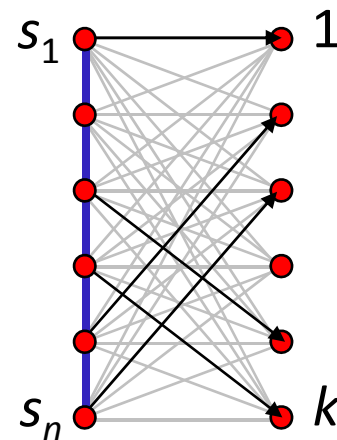
Bayesian interpretation

- Probability Space
 - $\Omega = \{f : (s_1 \dots s_n) \rightarrow \{1, \dots, k\}^n\}$
 - Exponential size!
- Markov-Random Field / graphical model
- Distribution:

$$P(f) = \frac{1}{Z} \left[\prod_{i=1}^n P^{(D)}(\mathbf{s}_i, f(\mathbf{s}_i)) \right] \left[\prod_{(i,j) \in G} P^{(S)}(\mathbf{s}_i, \mathbf{s}_j, f(\mathbf{s}_i), f(\mathbf{s}_j)) \right]$$



match local shape preserve local distance

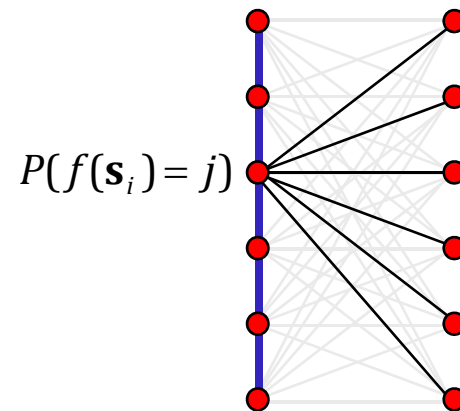


Direct MRF Approach

Solution

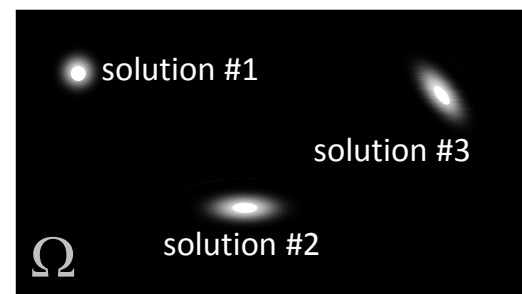
- Posterior distribution is *exponential*
- Instead, we compute marginals:
“Average” of all solutions

$$P(f(\mathbf{s}_i) = j) = \sum_{i_1=1}^k \dots \sum_{i_n=1}^k P(f = (i_1, \dots, j, \dots, i_n))$$



Postprocessing:

- Extract solutions
- Few solutions in a very large space

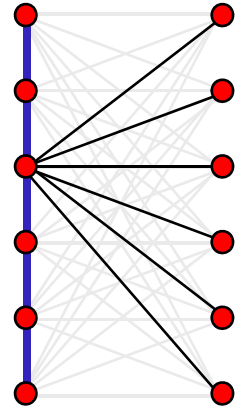


Direct MRF Approach

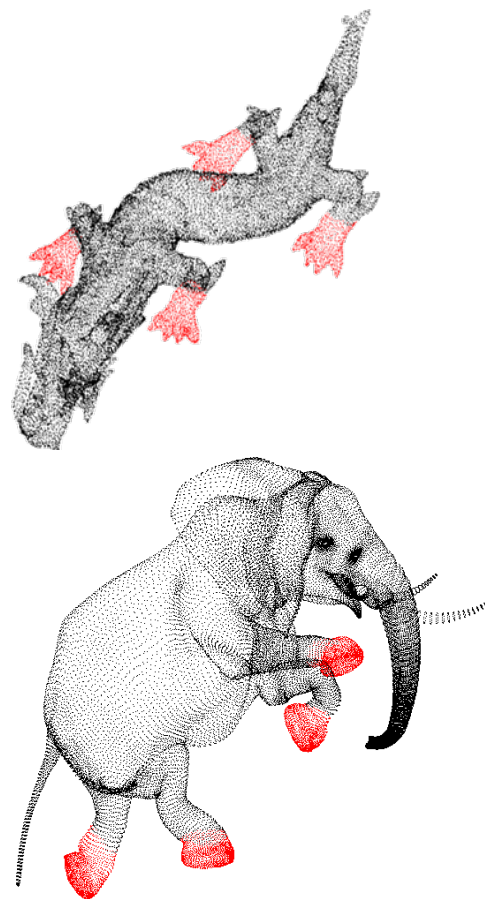
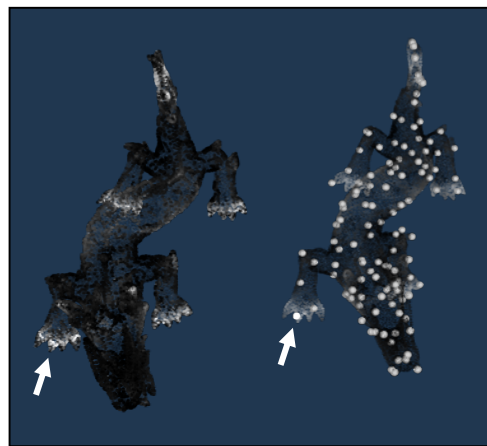
Inference

$$P(f(\mathbf{s}_i) = j) = \sum_{i_1=1}^k \dots \sum_{i_n=1}^k P(f = (i_1, \dots, j, \dots, i_n))$$

- Representation is polynomial, but computation is still NP hard
- Heuristic approximation: *Loopy belief propagation*
- Works well in practice



Example Result



Self-matching: Deformable Symmetries

Advanced Global Matching

Correlated correspondences [ASP*04]

A complete registration system [HAW*08]

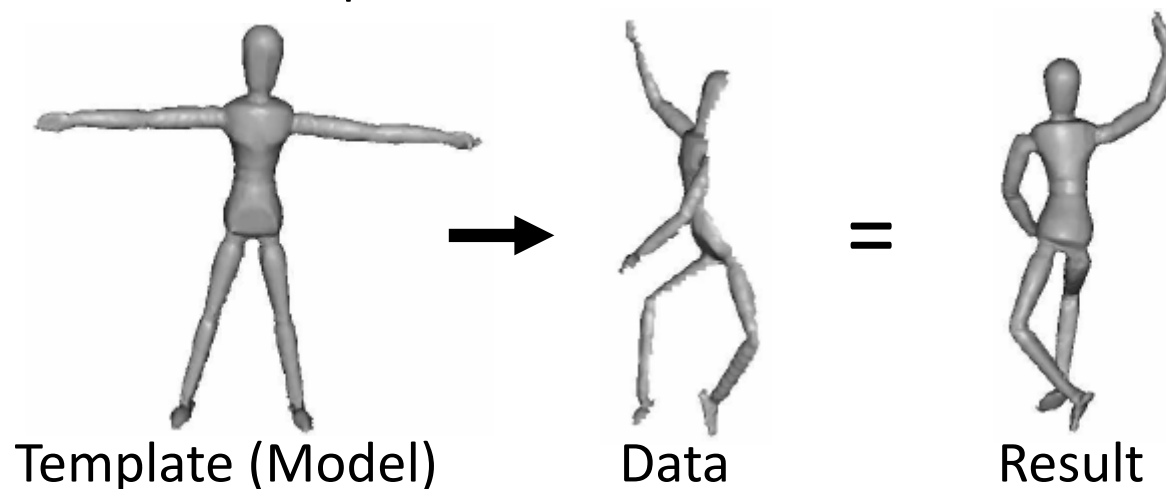
In this session...

Advanced Global Matching

- **How it's the same as last session**
- **How it's different**
- **What I'm going to talk about (overview)**
- **Figures in the next slides are from the respective papers**

Correlated correspondences

- Correspondence between data and model meshes
- Model mesh is a template; i.e. data is a subset of model



- Not a registration method; just computes corresponding points between data/model meshes
 - Non-rigid ICP [Hanhel et al. 2003] (using the outputted correspondences) used to actually generate the registration results seen in the paper

Basic Approach

Search (in the set of all possible correspondences) for the best correspondence set that “makes sense”

- Define what you mean by “makes sense”?
 - Minimize the amount of deformation induced by the corresp.
 - Preserve the same geodesic distances in model and data
 - Corresponding points have same feature descriptor values

Mathematical Technique

A joint probability model encodes this “makes sense” criteria

- Define a “probability” of each correspondence set between data/model meshes
- Find the correspondence with the highest probability using Loopy Belief Propagation (LBP) [Yedidia et al. 2003]

2 main components (next parts of the talk)

- Probability model
- Optimization

Joint Probability model

Given a correspondence, measure how much it “makes sense”

- A. Minimize the amount of deformation induced by the corresp.
- B. Preserve the same geodesic distances in model and data
- C. Corresponding points have same feature descriptor values

A, B are probabilities involving pairs of correspondences

- Call this a “pairwise potential”
- Represents prior knowledge of what “makes sense”

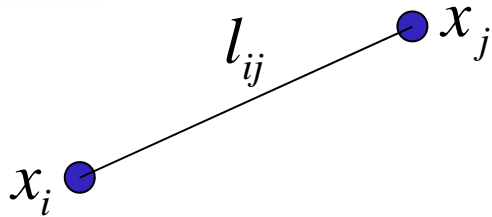
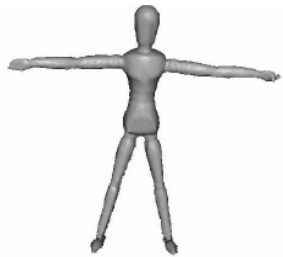
C involves only a single correspondence

- A “pointwise potential”

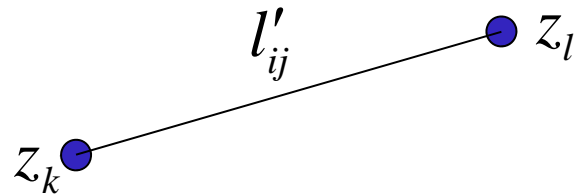
Deformation potential

Penalize unnatural deformations

- Edges lengths should stay the same $l_{ij} \approx l'_{ij}$



In **model** mesh

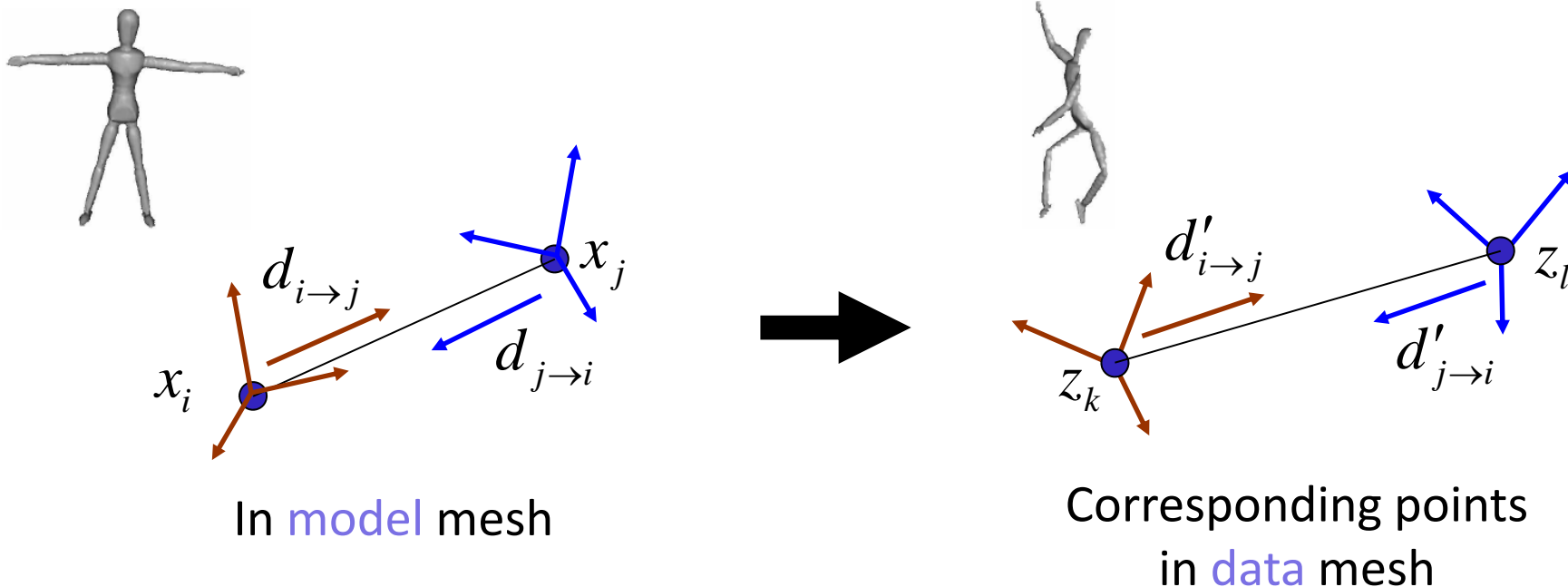


Corresponding points
in **data** mesh

Deformation potential

Penalize unnatural deformations

- Edges should twist little as possible $d_{i \rightarrow j} \approx d'_{i \rightarrow j}$, $d_{j \rightarrow i} \approx d'_{j \rightarrow i}$
- $d_{i \rightarrow j}$ Is the direction from x_i to x_j in x_i 's coord system



Encoding the preference

- Zero-mean Gaussian noise model for length and twists
- Define potential ψ_d for each edge (z_k, z_l) in the data mesh
 - (c_k, c_l) are “correspondence variables” indicating what is the corresponding point in the model mesh for z_k, z_l respectively

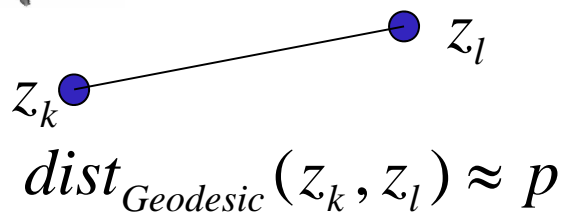
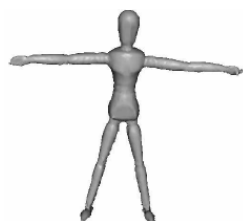
$$\psi_d(c_k = i, c_l = j) = G(l'_{ij} | l_{ij})G(d'_{i \rightarrow j} | d_{i \rightarrow j})G(d'_{j \rightarrow i} | d_{j \rightarrow i})$$

- Caveat: additional rotation needed to measure twist
 - For each possibility of $c_k = i$ precompute aligning rotation matrices via rigid ICP on surrounding local patch
 - Expand corresp. variables to be site/rotation pairs

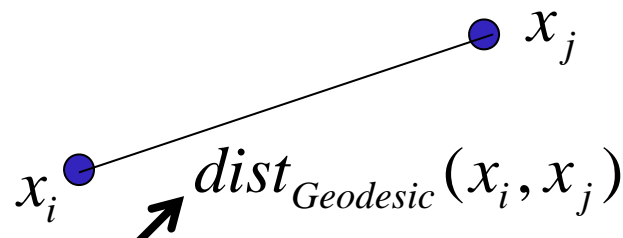
Geodesic distance potential

Penalize large changes in geodesic distance

- Geodesically **nearby** points should stay **nearby**
 - Enforced for each edge in the data mesh



Adjacent points
in **data** mesh



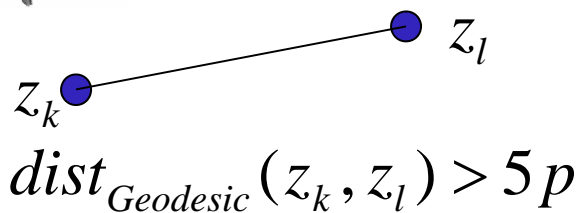
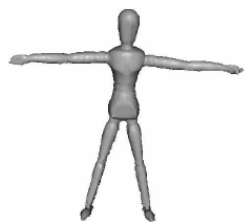
If $> 3.5p \rightarrow$ prob assigned 0
otherwise \rightarrow prob assigned 1

Corresponding points
in **model** mesh

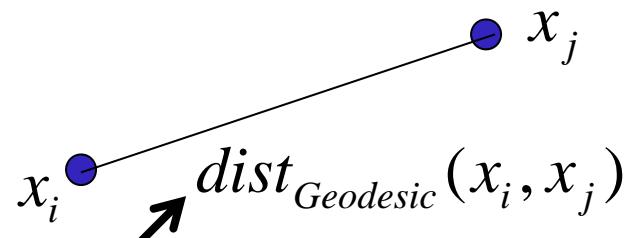
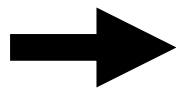
Geodesic distance potential

Penalize large changes in geodesic distance

- Geodesically **far** points should stay **far** away
 - Enforced for each pair of points in the data mesh whose geodesic distance is $> 5p$



Adjacent points
in **data** mesh



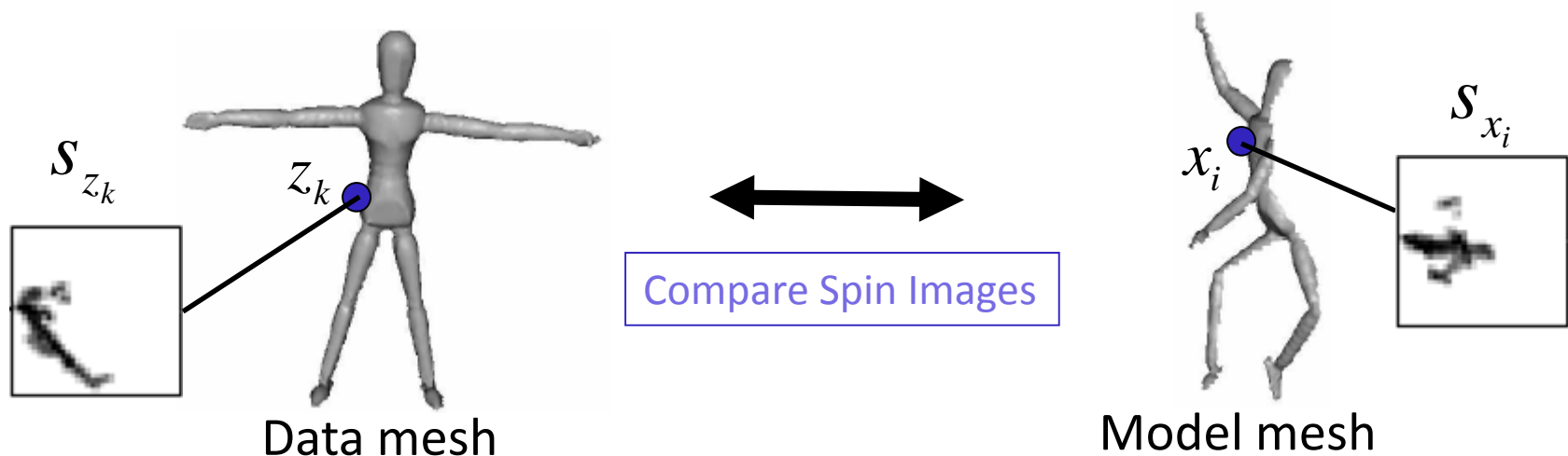
If $< 2p \rightarrow$ prob assigned 0
otherwise \rightarrow prob assigned 1

Corresponding points
in **model** mesh

Local surface signature potential

Spin images gives matching score for each individual correspondence

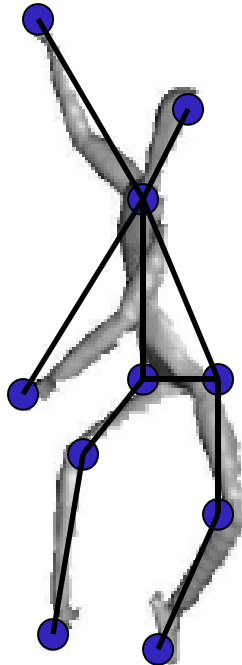
- Compute spin images & compress using PCA
→ gives **surface signature** s_{x_i} at each point x_i
- Discrepancy between s_{z_k} (data) and s_{x_i} (model)
- Zero-mean Gaussian noise model



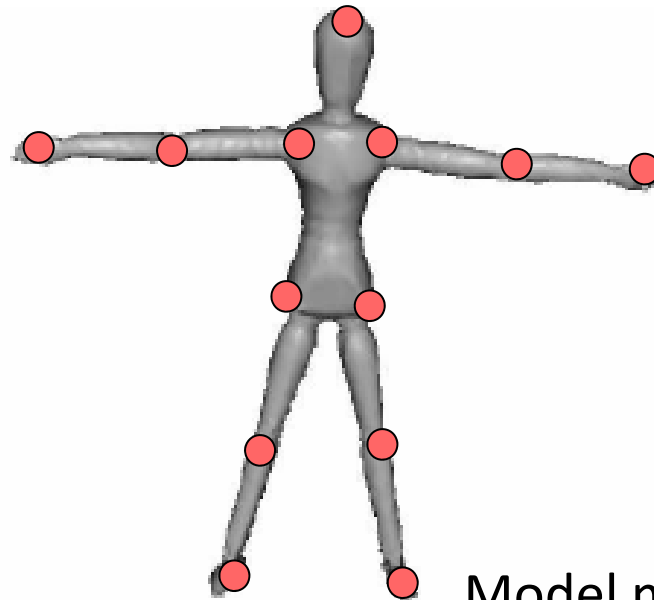
Model summary

Get Pairwise Markov Random Field (MRF)

- Pointwise potential for each pt in data
- Pairwise potential for each edge in data
 - Far geodesic potentials for each pair of points $> 5p$ apart



Data mesh



Model mesh

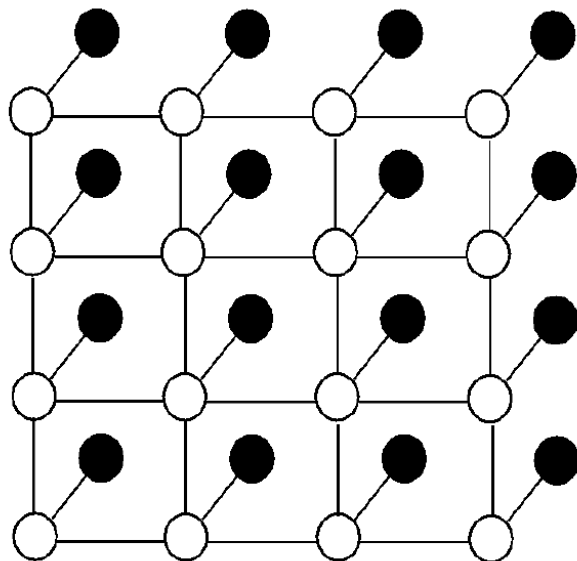
Quick intro: Markov Random Fields

Joint probability function visualized by a graph

- Prob. = Product of the potentials at all edges

● — ○ $\psi_k(c_k)$ ← (ex) Surface signature potential

○ — ○ $\psi_{kl}(c_k, c_l)$ ← (ex) Deformation, geodesic distance potential



$$P(\{c\}) = \frac{1}{Z} \prod_{k,l} \psi_{kl}(c_k, c_l) \prod_k \psi_k(c_k)$$

● “Observed” nodes

○ “Hidden” nodes

Loopy Belief Propagation (LBP)

**Usual way to compute marginal probabilities
(tabulate and sum up) takes exponential time**

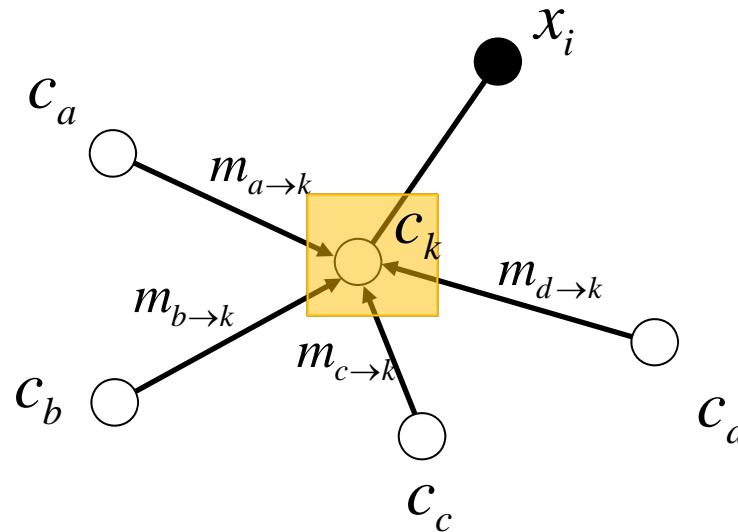
- BP is a dynamic programming approach to efficiently compute marginal probabilities
- Exact for tree MRFs, approximate for general MRFs

Loopy Belief Propagation (LBP)

Basic idea

- Marginals at node proportional to product of pointwise potential and **incoming messages**

$$b_k(c_k) = \phi_k(c_k) \prod_{l \in N(k)} m_{l \rightarrow k}(c_k)$$

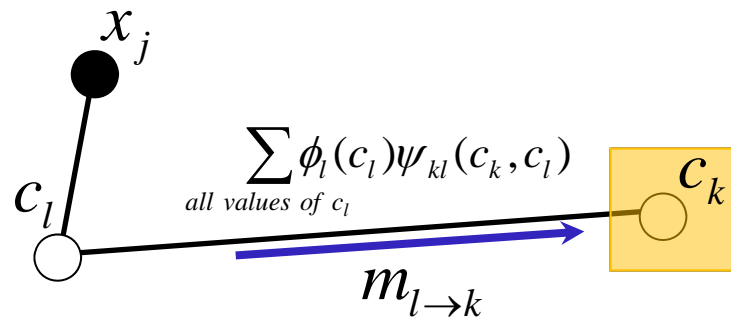


Loopy Belief Propagation (LBP)

Basic idea

- Compute these messages (at each edge) and we are done

$$m_{l \rightarrow k}(c_k) \leftarrow \sum_{\text{all values of } c_l} \phi_l(c_l) \psi_{kl}(c_k, c_l) \prod_{q \in N(l) \setminus k} m_{q \rightarrow l}(c_l)$$

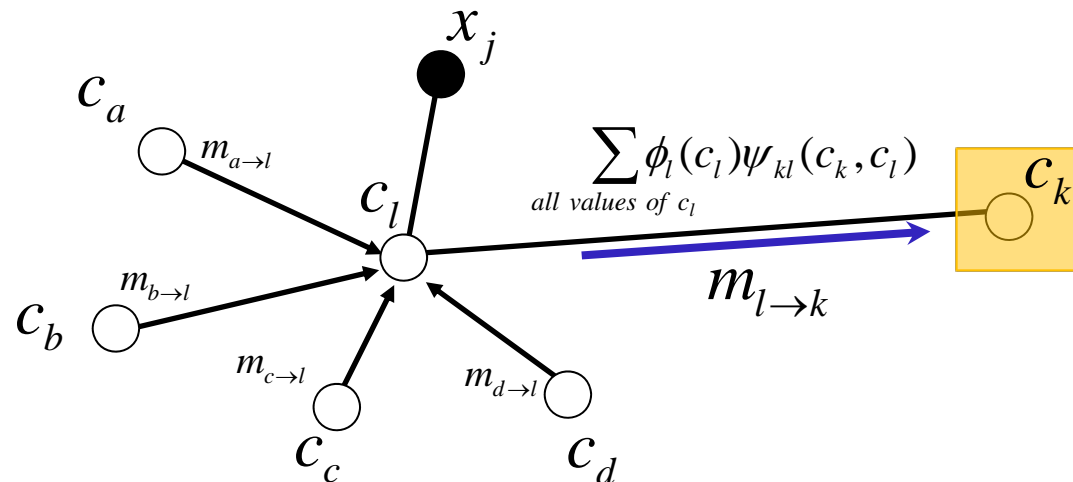


Loopy Belief Propagation (LBP)

Basic idea

- Compute these messages (at each edge) and we are done

$$m_{l \rightarrow k}(c_k) \leftarrow \sum_{\text{all values of } c_l} \phi_l(c_l) \psi_{kl}(c_k, c_l) \prod_{q \in N(l) \setminus k} m_{q \rightarrow l}(c_l)$$

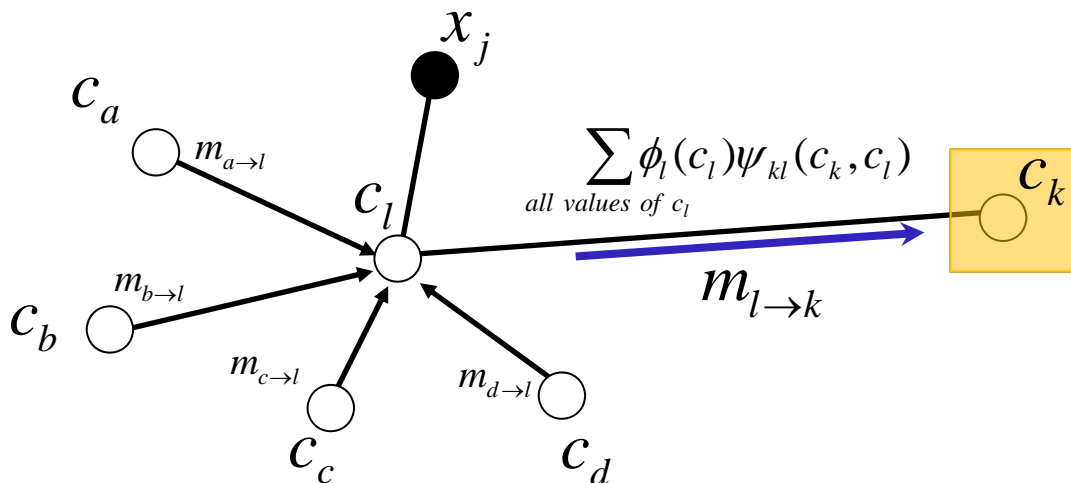


Loopy Belief Propagation (LBP)

Basic idea

- Compute these messages (at each edge) and we are done

$$m_{l \rightarrow k}(c_k) \leftarrow \sum_{\text{all values of } c_l} \phi_l(c_l) \psi_{kl}(c_k, c_l) \prod_{q \in N(l) \setminus k} m_{q \rightarrow l}(c_l)$$

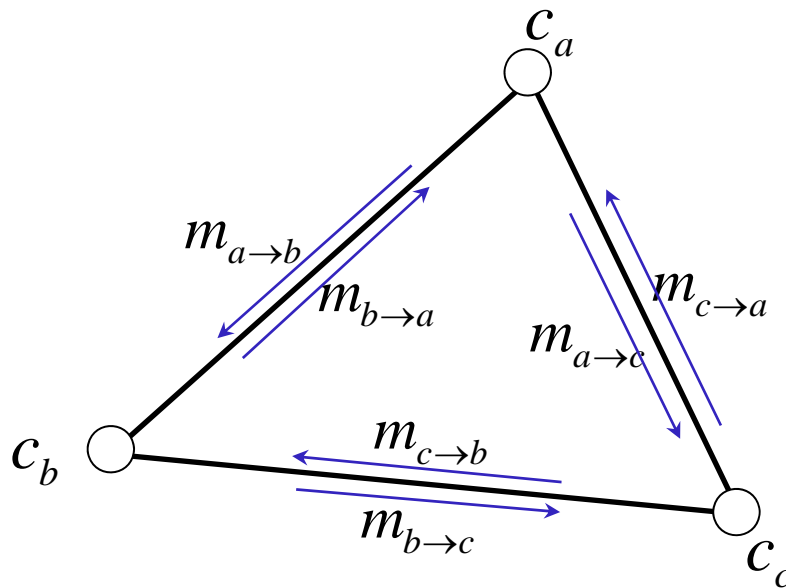


- Recursive formulation
- Start at ends and work your way towards the rest

Loopy Belief Propagation (LBP)

Loops: iterate until messages converge

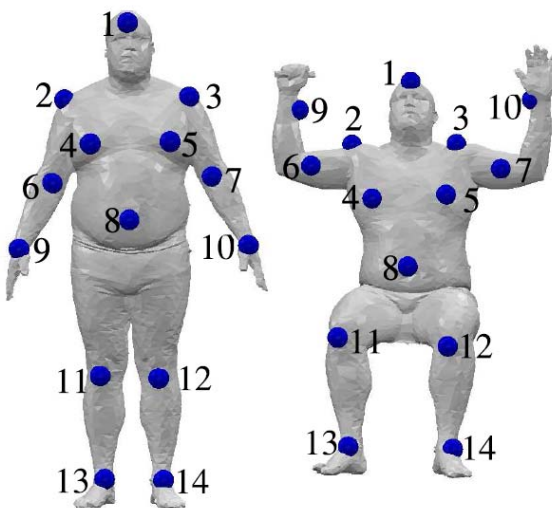
- Start with initial values (ex: $\sum_{\text{all values of } c_a} \phi_a(c_a) \psi_{ab}(c_a, c_b)$)
- Apply message update rule until convergence



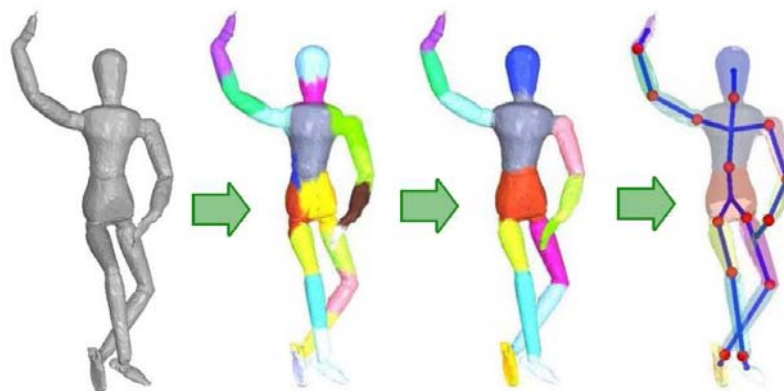
- Convergence not guaranteed, but works well in practice

Results & Applications

- Efficient, coarse-to-fine implementation
- Xeon 2.4 GHz CPU, 1.5 mins for arm, 10 mins for puppet



Correspondences on human body models



Finding articulated parts



Interpolation between poses

Conclusion

Correlated correspondence

- Robust method for matching correspondences
- Measure how much the correspondence “makes sense”
- Probability model → optimized using LBP
- Requires a template
 - If model is incomplete, then there is no “correct” corresponding point to assign

Questions?

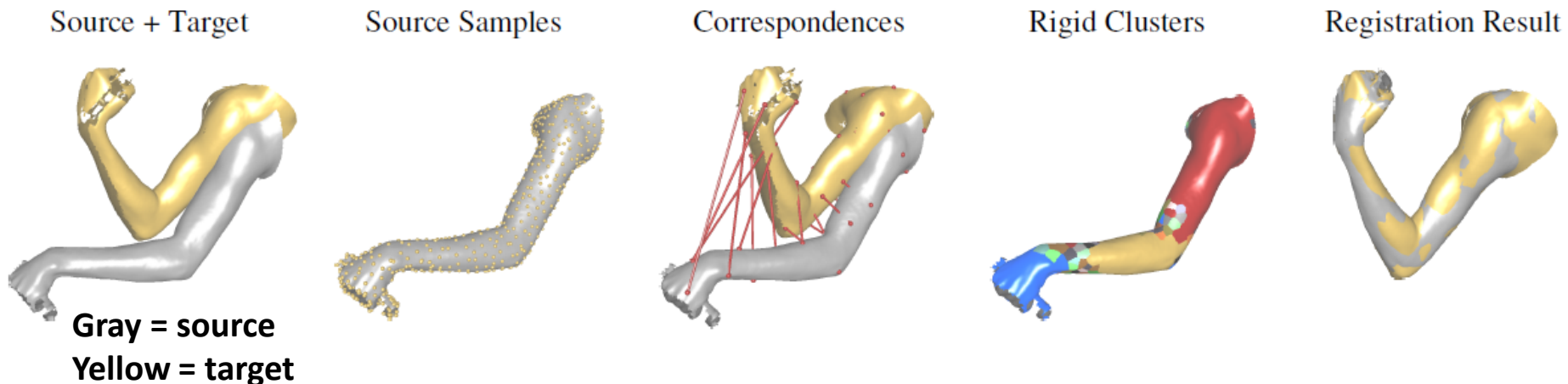
Next topic: HAW*08

An application to the spectral matching method of last session

- A good illustration of how a matching method fits into a real registration pipeline

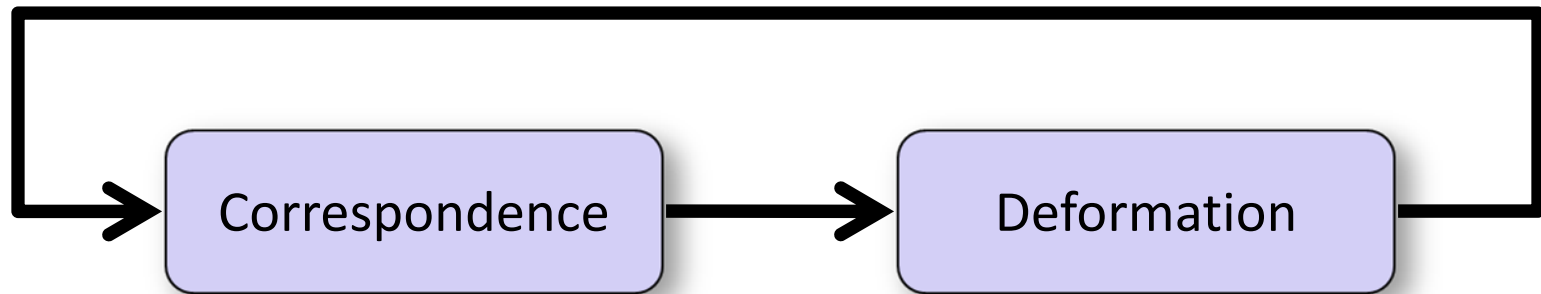
A pairwise method

- Deform the source shape to match the target shape



Overview

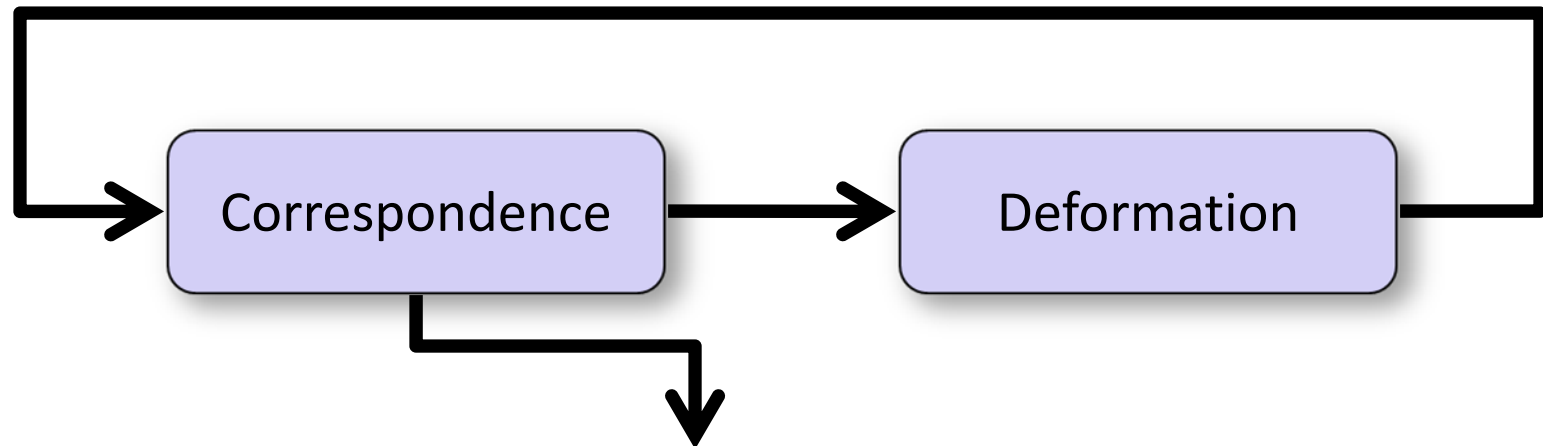
Performs both correspondence and deformation



- Correspondences based on **improving closest points**
- After finding correspondences, **deform** to move shapes closer together
- Re-take correspondences from the deformed position
- Deform again, and repeat until convergence

Overview

Performs both correspondence and deformation

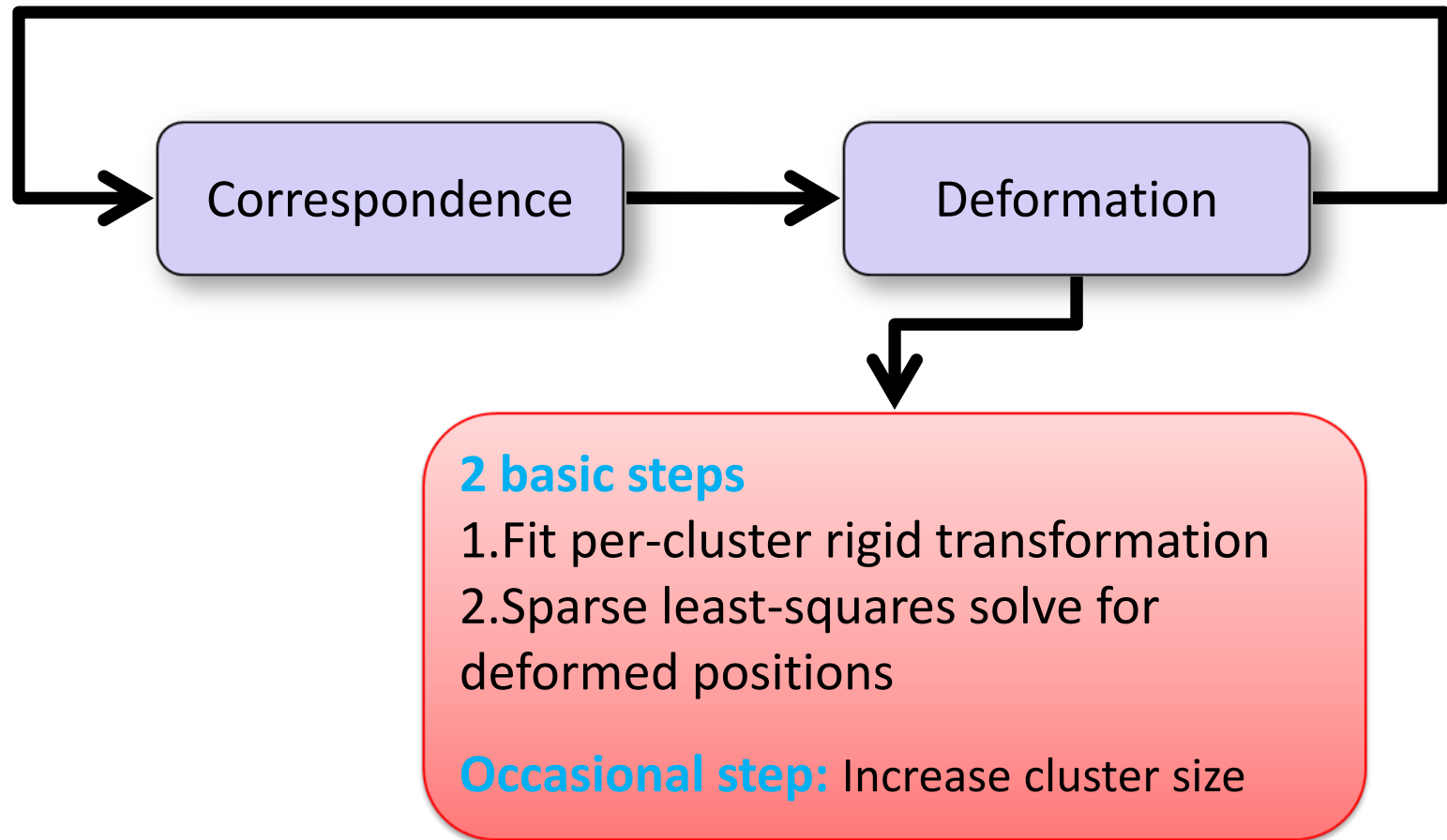


5 basic steps

1. Closest points
2. Improve by feature matching
3. Filter by spectral matching
4. Expand sparse set
5. Fine-tune target locations

Overview

Performs both correspondence and deformation



Detailed Overview

Sampling

- Whole process works with reduced sample set

Correspondence & Deformation

- Examine each step in more detail

Discussion

- Discuss pros/cons

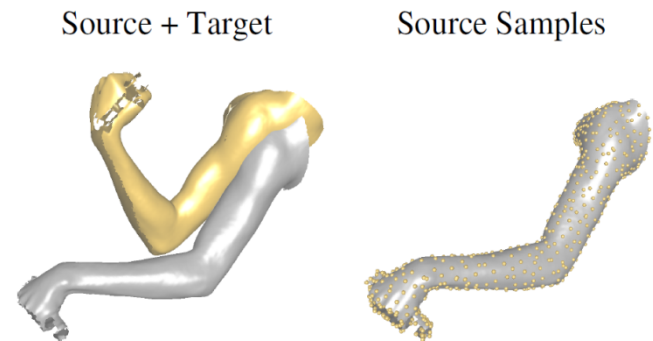
Sample for robustness & efficiency

Coarse to fine approach

- Use uniform subsampling of the surface and its normals
- Improve efficiency, can improve robustness to local minima

Let's make it more concrete

- Sample set denoted S_i
- In correspondence: for each S_i , find corresponding target points t_i
- In deformation: given t_i , find deformed sample positions S'_i that match t_i while preserving local shape detail



Correspondence Step #1

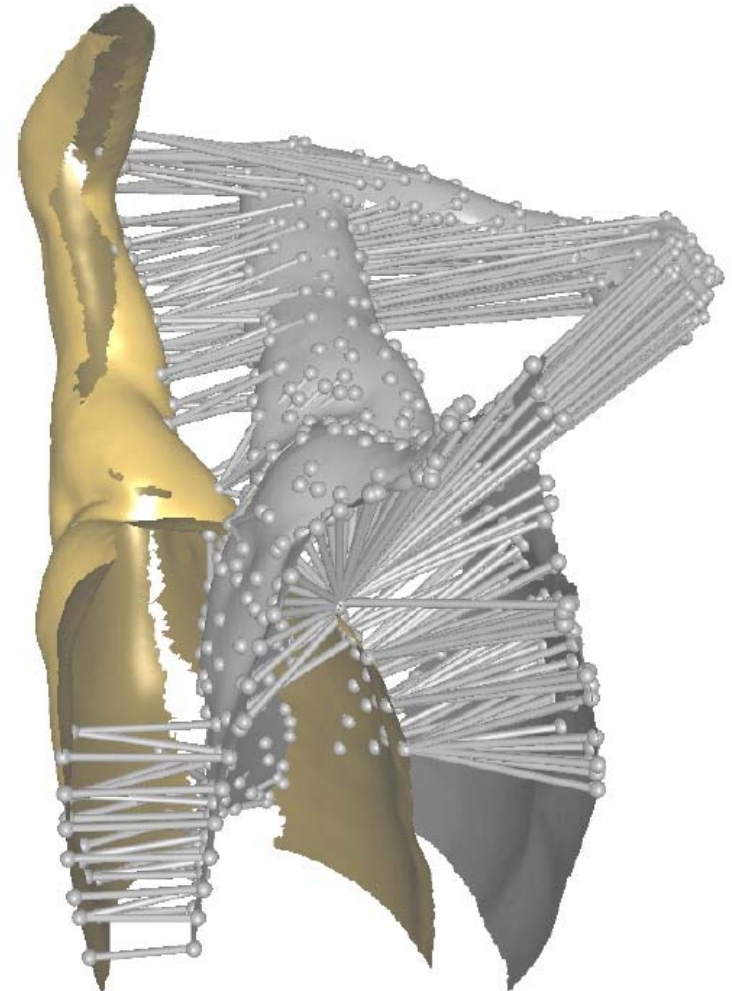
Find closest points

- For each source sample, find the closest target sample
 - s = sample point on source
 - t = sample point on target

$$\arg \min_{t \in \hat{T}} \|s - t\|^2$$

- Usually pretty bad

Target (yellow) ← Source (gray)



Closest point correspondences

Correspondence Step #2

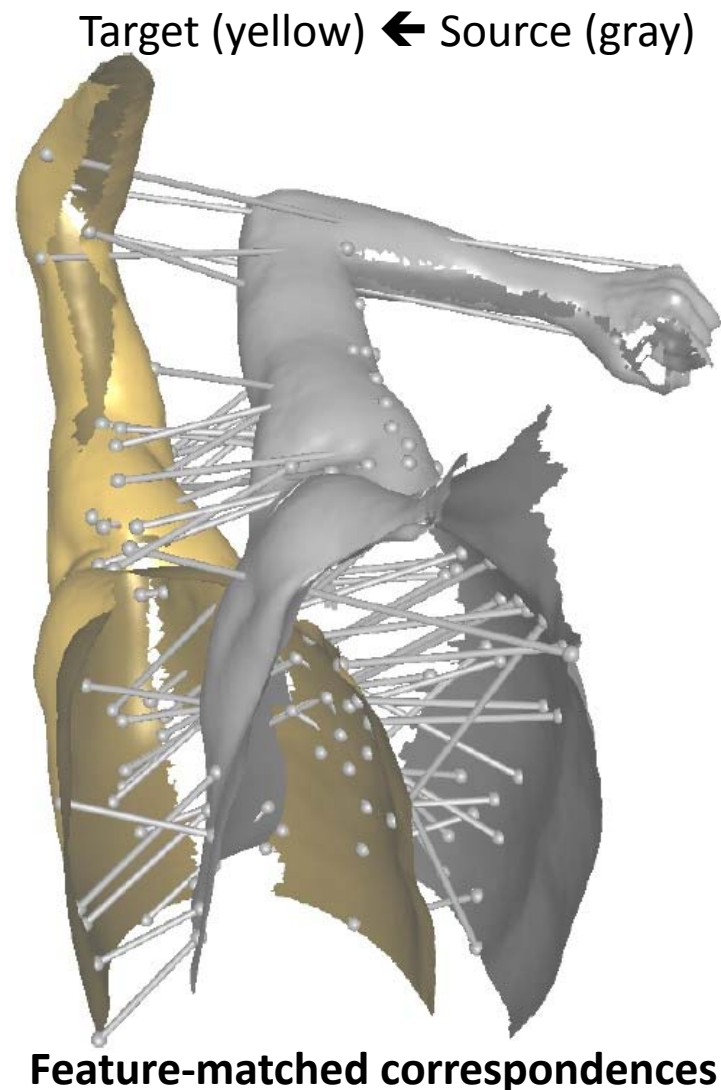
Improve by feature matching

- **Search** target's neighbors to see if there's better feature match, **replace** target

- Let $f(s)$ be feature value of s

$$t \leftarrow \arg \min_{t' \in N(t)} \|f(s) - f(t')\|^2$$

- Iterate until we stop moving
- If we move too much, discard correspondence
- **Much better, but still outliers**



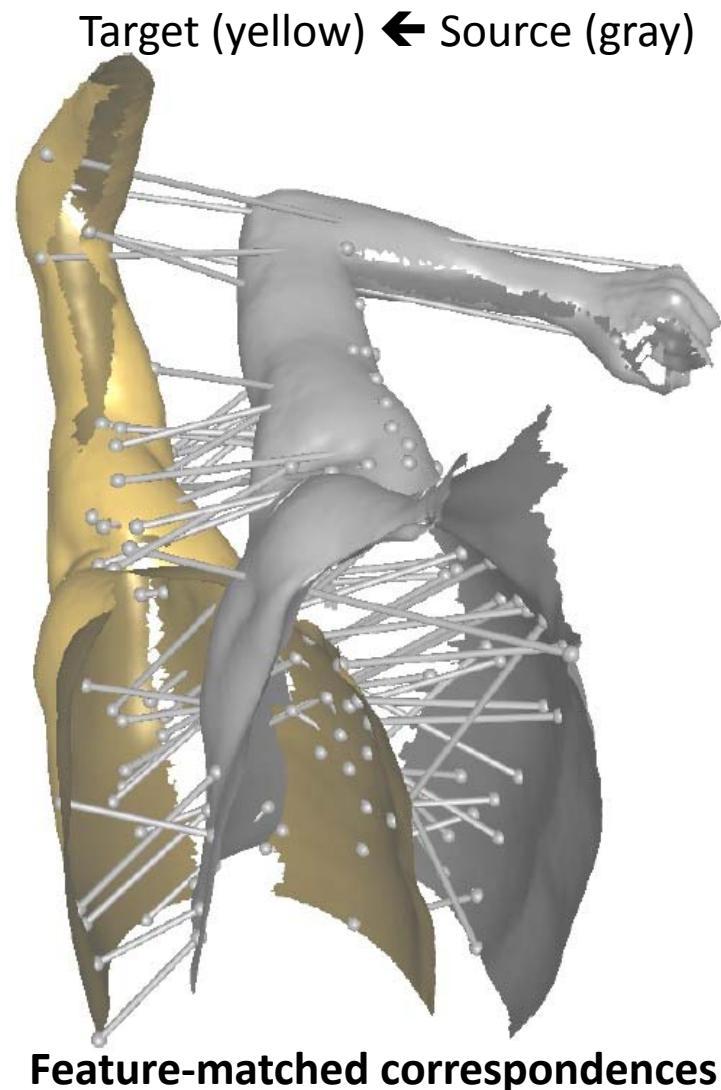
Correspondence Step #3

Filter by spectral matching

- *(First some preprocessing)*
- Construct k -nn graph on both src & tgt sample set ($k = 15$)
- Length of shortest path on graph gives approx. geodesic distances on src & tgt

$$d_g(s_i, s_j) \quad d_g(t_i, t_j)$$

- Goal is to filter these -----> and keep a subset which is **geodesically consistent**



Correspondence Step #3

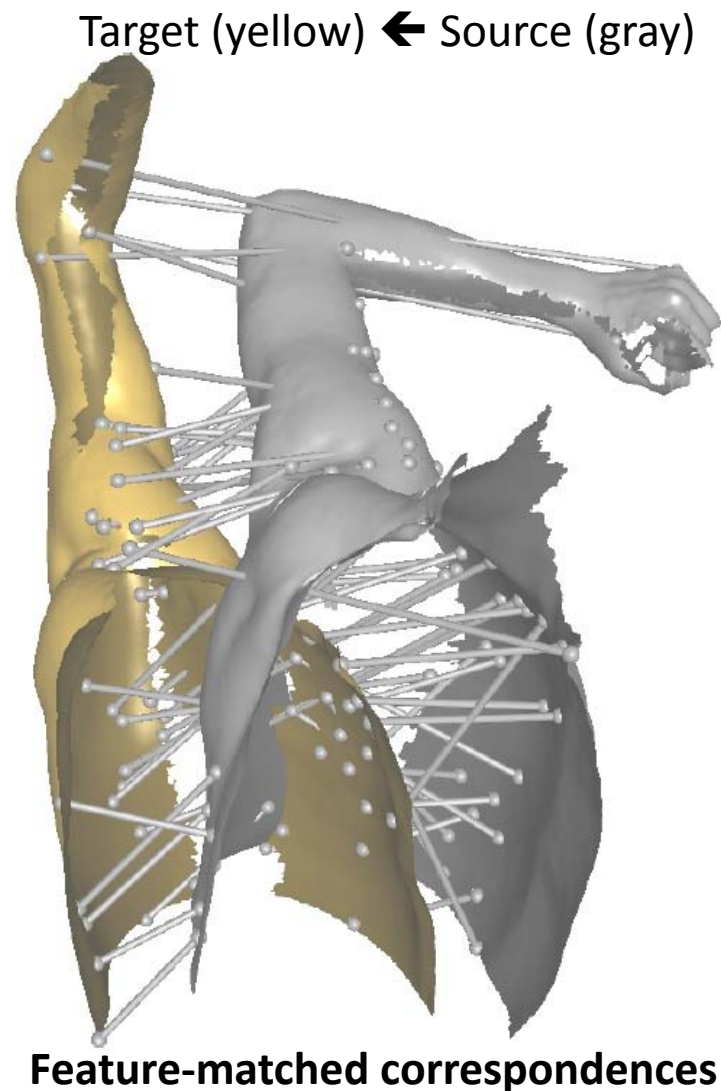
Filter by spectral matching

- Construct affinity matrix M using these shortest path distances
- Consistency term & matrix

$$c_{ij} = \min\left\{\frac{d_g(s_i, s_j)}{d_g(t_i, t_j)}, \frac{d_g(t_i, t_j)}{d_g(s_i, s_j)}\right\}, \quad c_{ii} = 1$$

$$M_{ij} = \begin{cases} \left(\frac{c_{ij} - c_0}{1 - c_0}\right)^2 & c_{ij} > c_0, \\ 0 & \text{otherwise,} \end{cases}$$

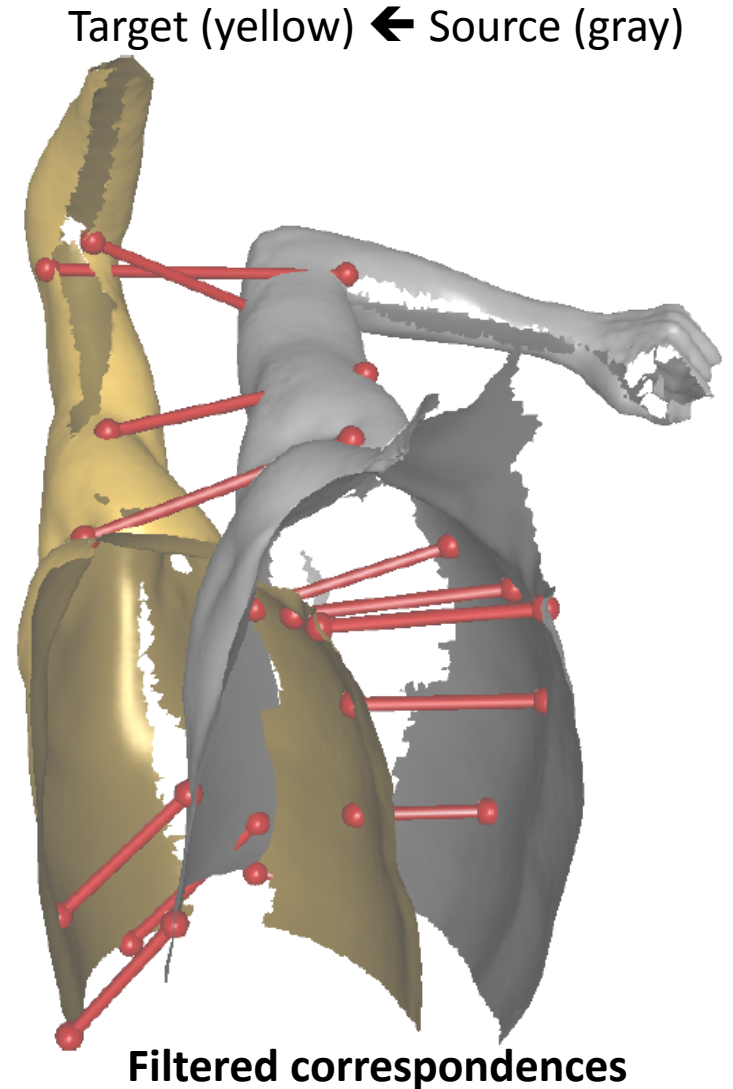
- Threshold $c_0 = 0.7$ gives how much error in consistency we are willing to accept



Correspondence Step #3

Filter by spectral matching

- Apply spectral matching: find eigenvector with largest eigenvalue \rightarrow score for each correspondence
- Iteratively add corresp. with largest score while consistency with the rest is above c_0
- Gives **kernel** correspondences
- **Filtered matches usually sparse**



Correspondence Step #4

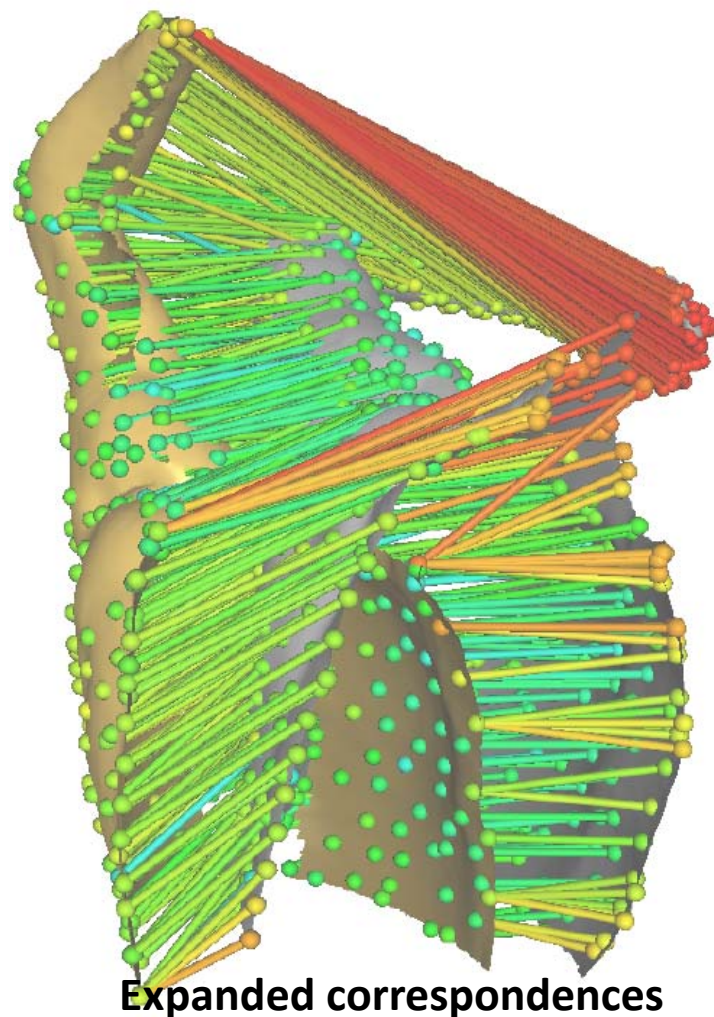
Expand sparse set

- Lots of samples have no target position
- For these, find best target position that respects geodesic distances to kernel set

$$\mathbf{t}_i = \arg \min_{\mathbf{t} \in N_g(\mathbf{t}_j, \bar{T})} e_K(\mathbf{s}_i, \mathbf{t})$$

$$e_K(\mathbf{s}, \mathbf{t}) = \sum_{(\mathbf{s}_k, \mathbf{t}_k) \in K} \left[d_g(\mathbf{s}, \mathbf{s}_k) - d_g(\mathbf{t}, \mathbf{t}_k) \right]^2$$

Target (yellow) ← Source (gray)



Correspondence Step #4

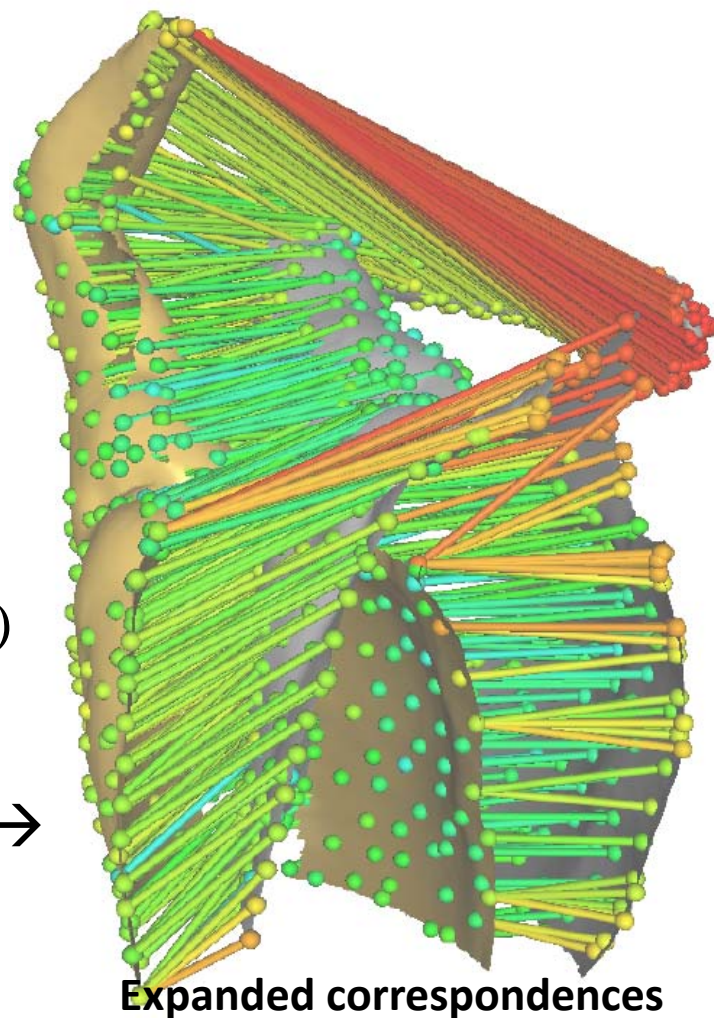
Expand sparse set

- Lots of samples have no target position
- Compute confidence weight based only how well it respects geodesic distances to kernel set

$$w_i = \exp\left(-\frac{e_K(\mathbf{s}_i, \mathbf{t}_i)}{2e}\right) \quad e = \frac{1}{|K|} \sum_{(\mathbf{s}_k, \mathbf{t}_k) \in K} e_K(\mathbf{s}_k, \mathbf{t}_k)$$

Red = not consistent ---→
Blue = very consistent

Target (yellow) ← Source (gray)

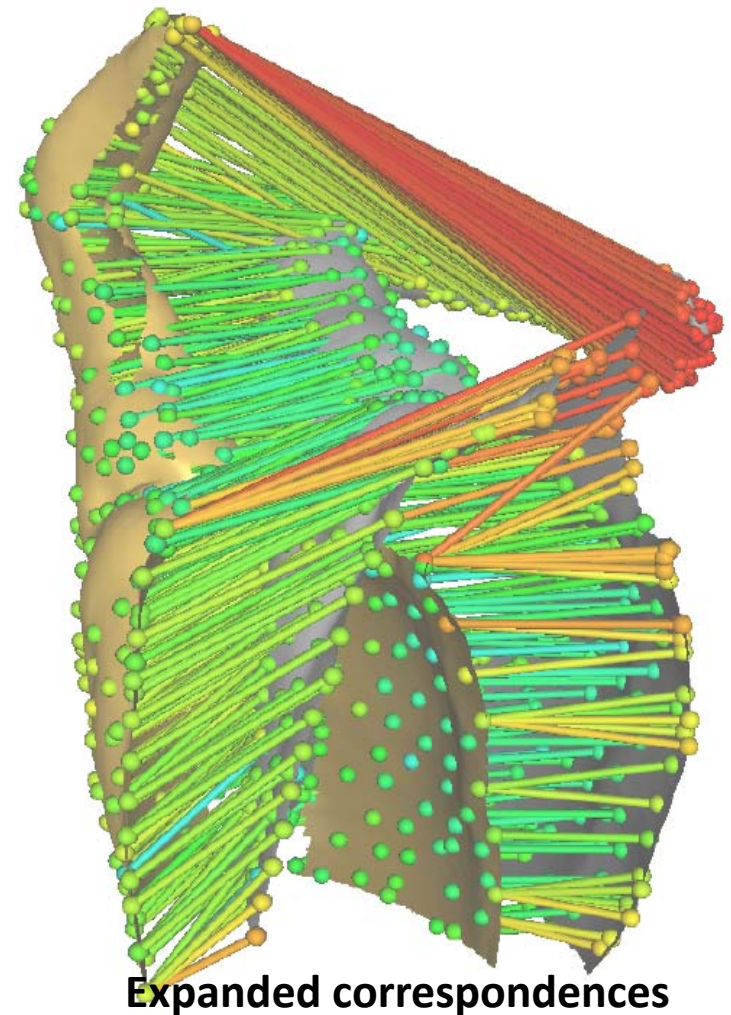


Correspondence Step #5

Fine-tuning

- So far, target points restricted to be points in target **samples**
- **Not accurate when shapes are close together**
- Relax this restriction and let target points become any point in the original point cloud
- Replace target sample with a closer neighbor in the original point cloud

Target (yellow) ← Source (gray)



Deformation

Solved by energy minimization (least squares)

- Last step gave target positions t_i
- Now find deformed sample positions s'_i that match target positions t_i

Two basic criteria:

- Match correspondences: s_i should be close to t_i
- Shape should preserve detail (as-rigid-as-possible)
- Combine to give energy term:

$$E = \lambda_{corr} E_{corr} + \lambda_{rigid} E_{rigid}$$

Correspondence matching term

Combination of point-to-point ($\alpha=0.6$) and point-to-plane ($\beta=0.4$) metrics

- Weighted by confidence weight w_i of the target position

$$E_{corr} = \sum_{\mathbf{s}_i \in \bar{S}} w_i \left[\alpha \|\mathbf{s}'_i - \mathbf{t}_i\|^2 + \beta ((\mathbf{s}'_i - \mathbf{t}_i)^T \mathbf{n}_i)^2 \right]$$

The diagram illustrates the decomposition of the correspondence matching term E_{corr} . The equation is shown with arrows pointing from the two terms inside the brackets to labels in rounded rectangular boxes below. The first term, $\alpha \|\mathbf{s}'_i - \mathbf{t}_i\|^2$, is labeled "Point-to-point". The second term, $\beta ((\mathbf{s}'_i - \mathbf{t}_i)^T \mathbf{n}_i)^2$, is labeled "Point-to-plane". Additionally, an arrow points from the confidence weight w_i in the summation to the text "Weighted by confidence weight w_i of the target position" in the list above.

Point-to-point

Point-to-plane

Shape preservation term

Deformed positions should preserve shape detail

- Form an extended cluster \tilde{C}_k for each sample point: the sample itself and its neighbors
- For each \tilde{C}_k find the rigid transformation (R,T) from sample positions to their deformed locations

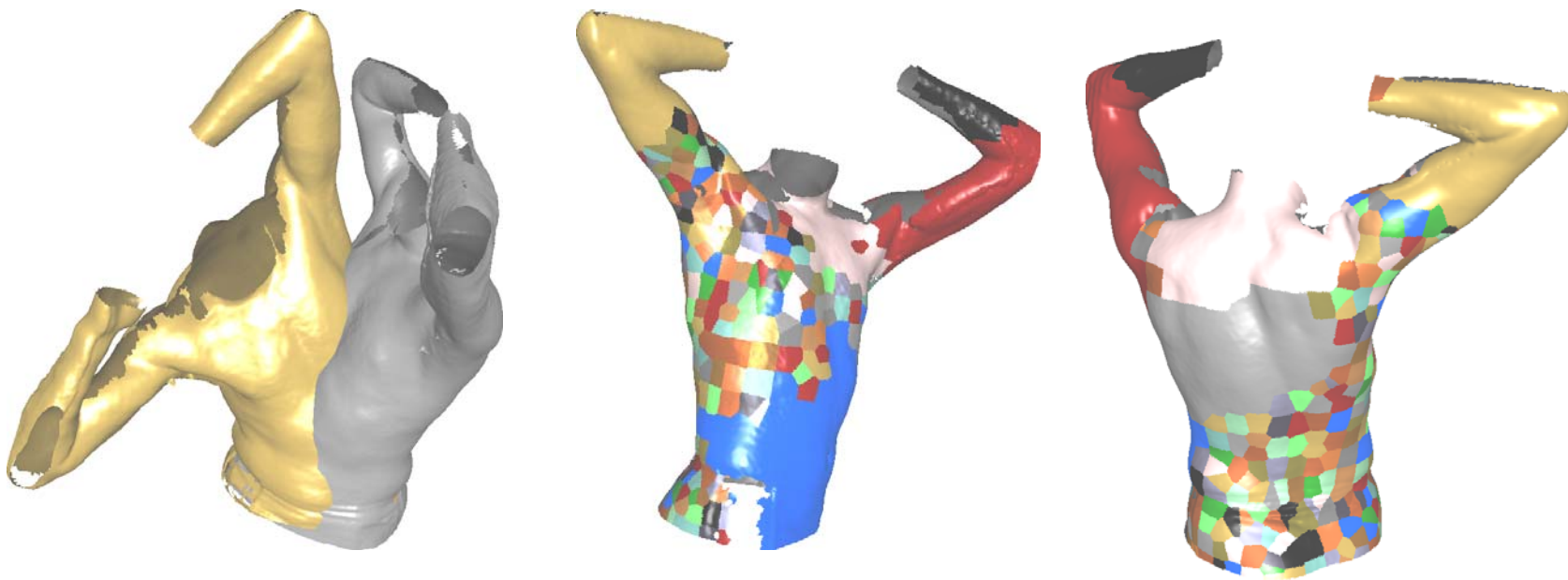
$$E_k = \sum_{s_i \in \tilde{C}_k} \left\| \mathbf{R}_k \mathbf{s}_i + \mathbf{T}_k - \mathbf{s}'_i \right\|^2$$

- When solving for \mathbf{s}'_i , constrain them to move rigidly according to each cluster that it's associated with

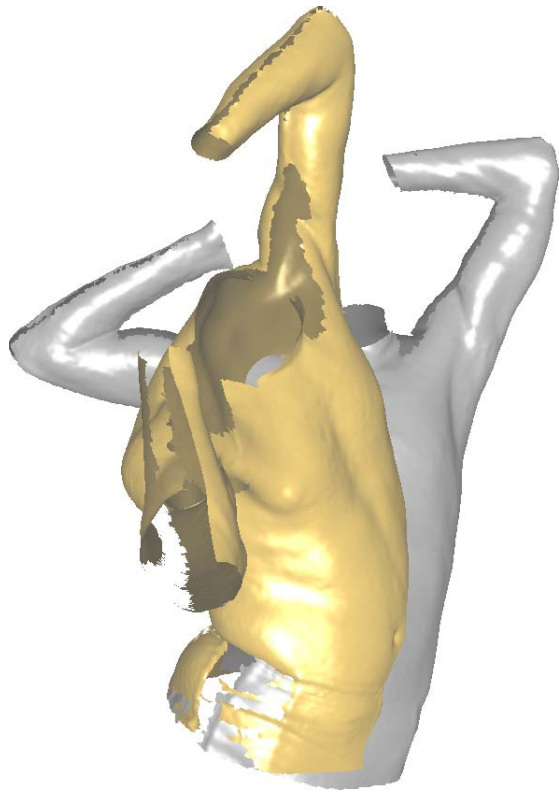
$$E_{\text{rigid}} = \sum_k E_k = \sum_k \sum_{s_i \in \tilde{C}_k} \left\| \mathbf{R}_k \mathbf{s}_i + \mathbf{T}_k - \mathbf{s}'_i \right\|^2$$

Clusters for local rigidity

- Initially each cluster contains a single sample point
- Every 10 iterations (of correspondence & deformation), combine clusters that have similar rigid transformations (forming larger rigid parts)



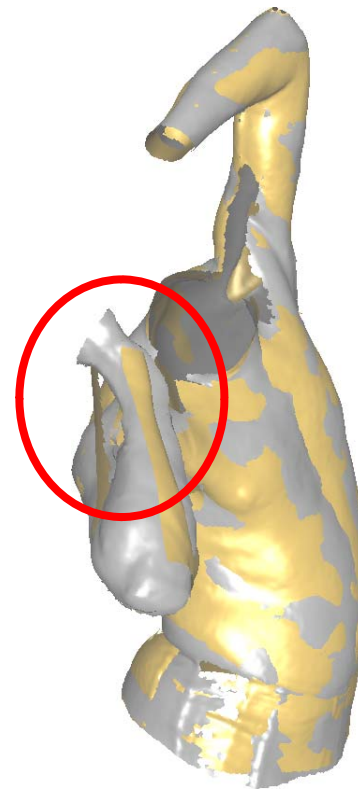
Advantages of features & clustering



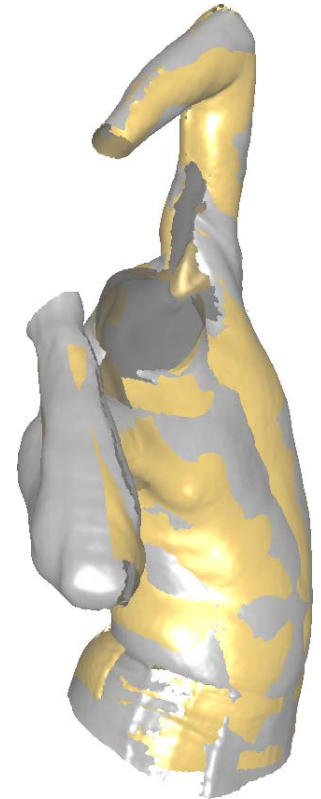
Source + Target



Without Features

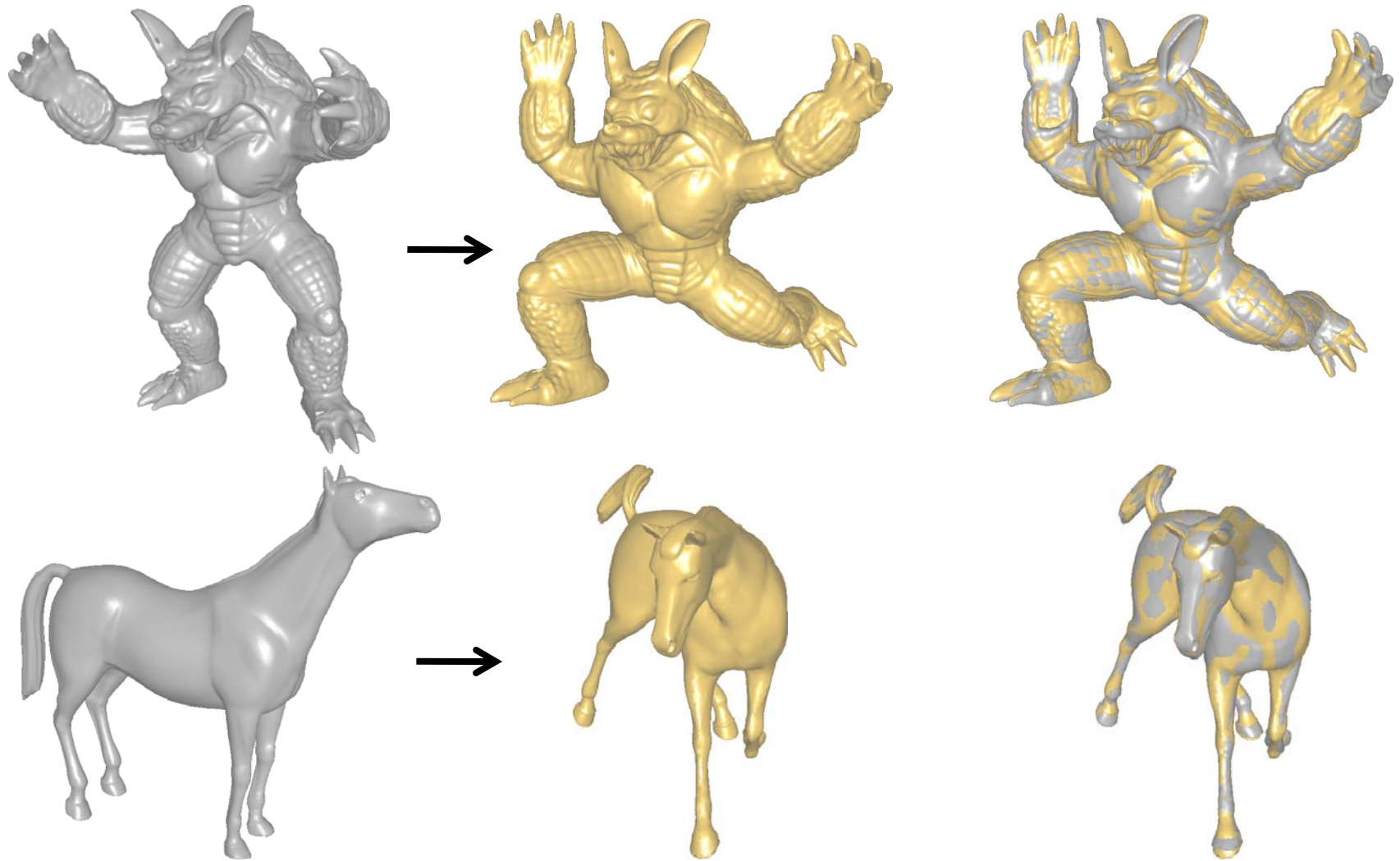


Without Clustering

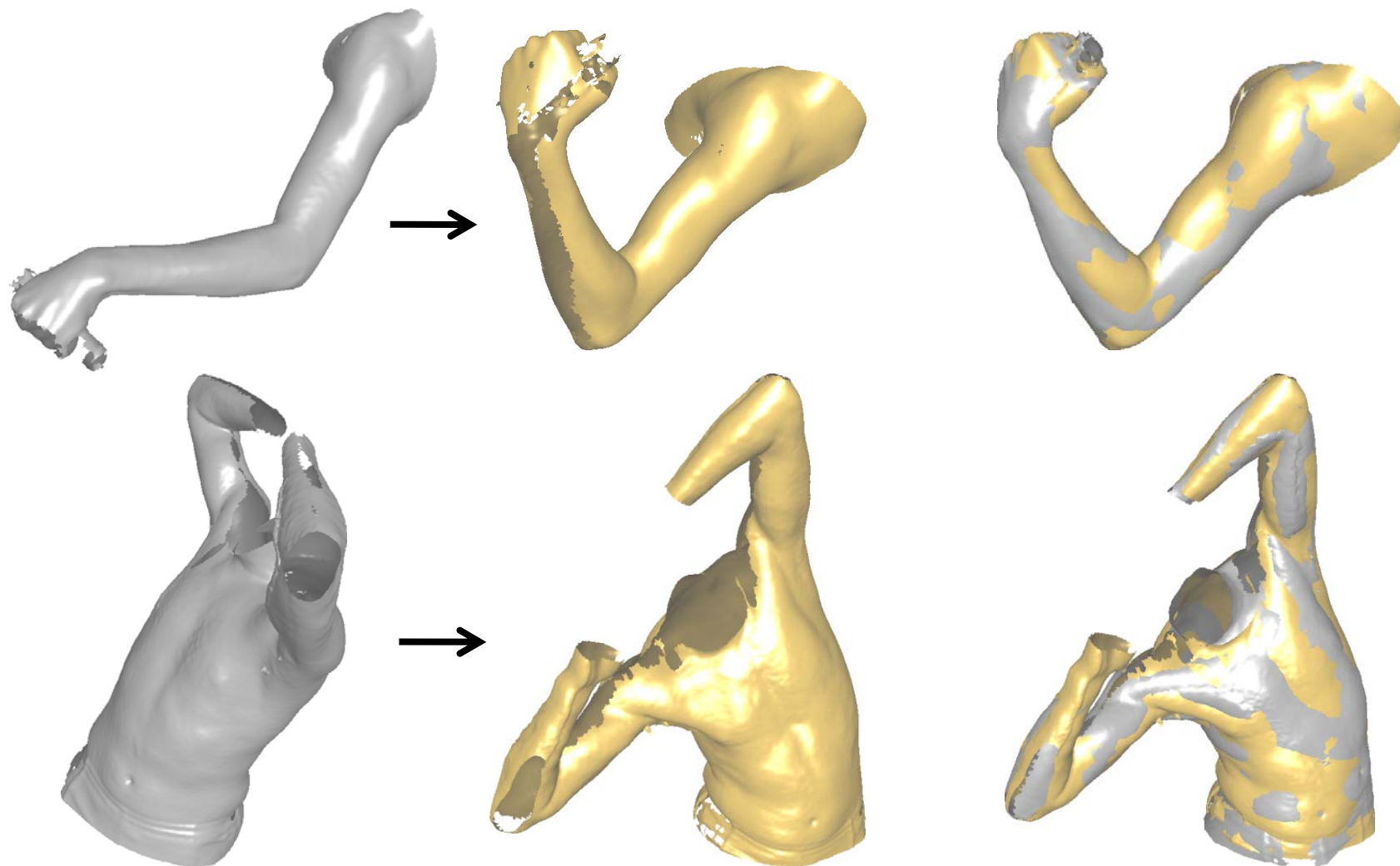


With Both

Results



Results



Results

Efficient, robust method



| data set | #poses | #pairs | $ \mathcal{S} $ | $ \hat{\mathcal{S}} $ | pre time | reg time |
|-----------|--------|--------|-----------------|-----------------------|----------|----------|
| Horse | 10 | 45 | 80k | 2500 | 7.4s | 13.6s |
| Armadillo | 12 | 66 | 332k | 2500 | 7.6s | 14.8s |
| Arms | 36 | 630 | 80k | 600 | 2.1s | 1.1s |
| Shoulder | 33 | 528 | 117k | 800 | 3.4s | 1.9s |
| Torso | 27 | 231 | 325k | 1100 | 4.5s | 4.5s |

Conclusion

Non-rigid registration under isometric deformations

- Improve closest point correspondences using features and spectral matching
- Deform shape while preserving local rigidity of clusters
- Iteratively estimate correspondences and deformation until convergence
- Robust, efficient method
- Relies on geodesic distances (problematic when holes are too large)

Questions?

Geometric Registration for Deformable Shapes

3.4 Probabilistic Techniques

RANSAC • Forward Search • Efficiency Guarantees

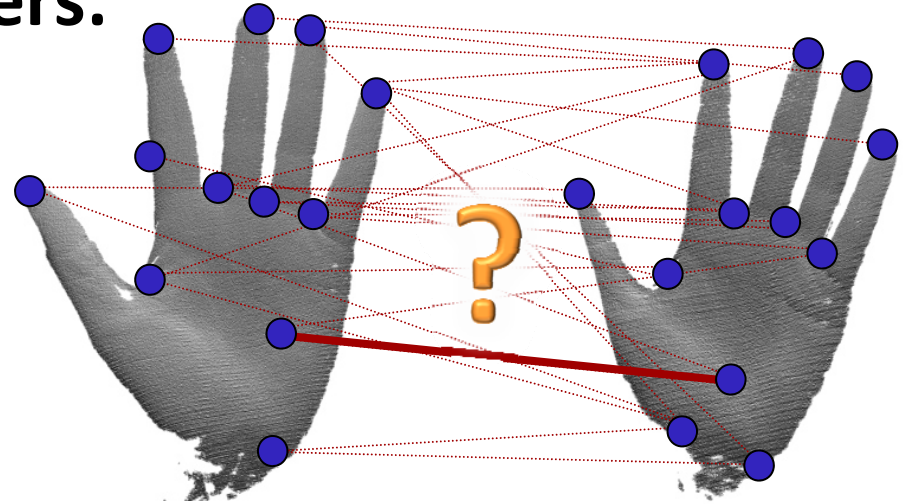
Ransac and Forward Search

The Basic Idea

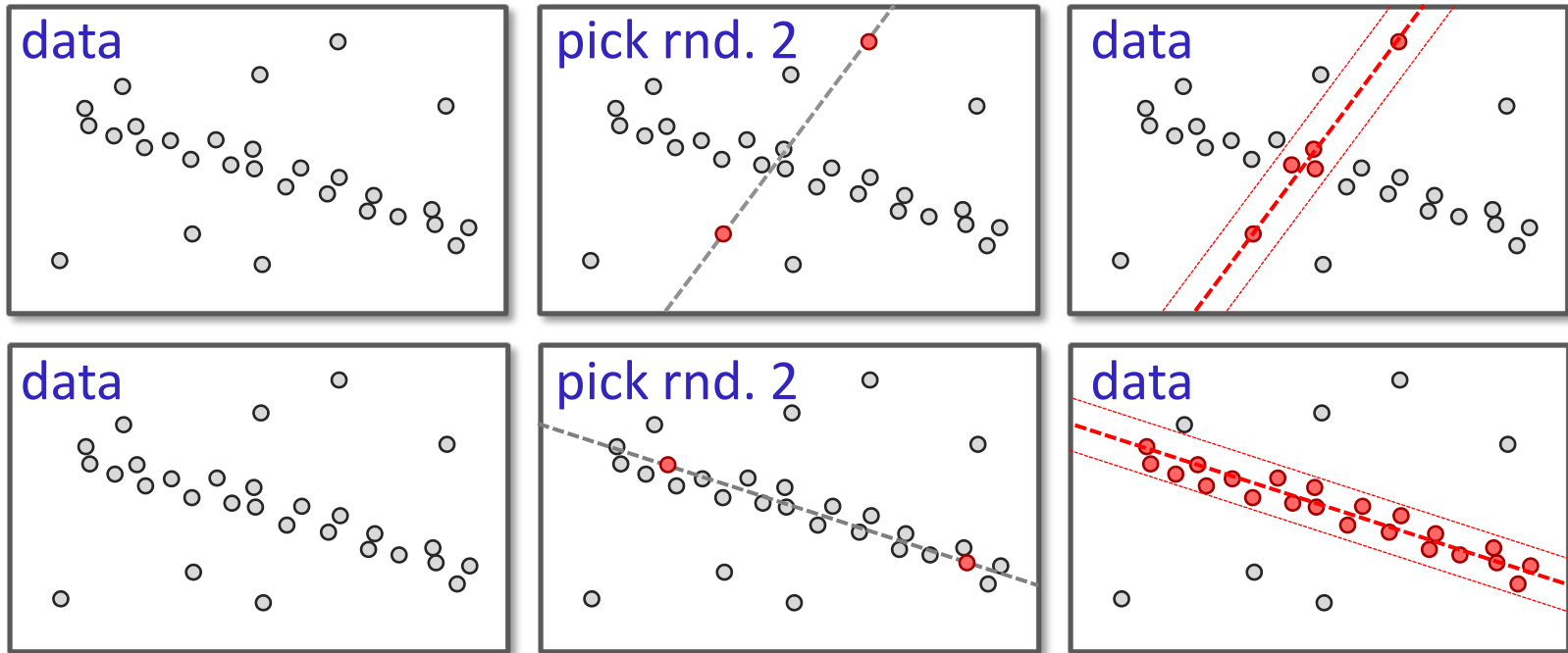
Random Sampling Algorithms

Estimation subject to outliers:

- We have candidate correspondences
- But most of them are bad
- Standard vision problem
- Standard tools:
Ransac & forward search



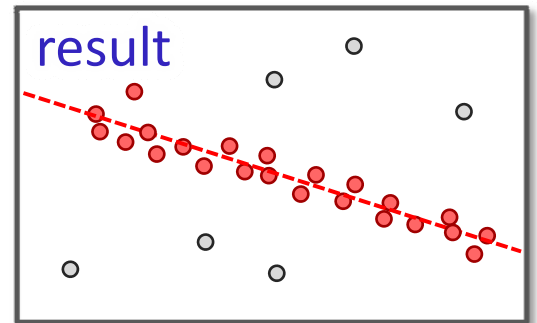
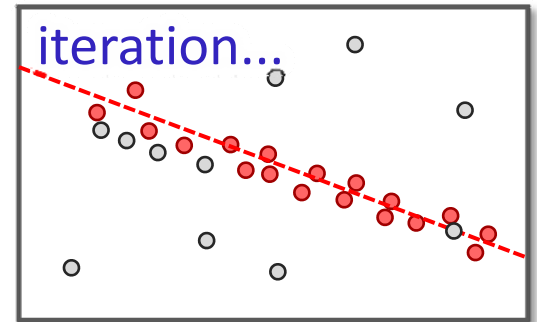
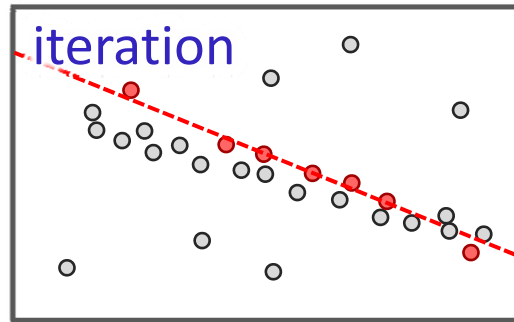
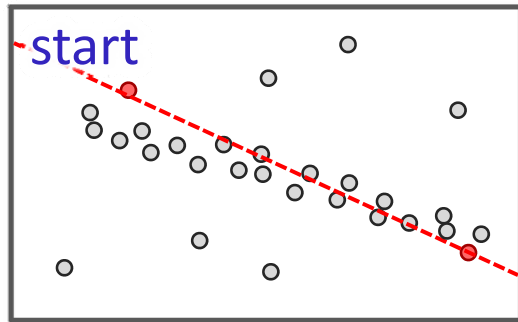
RANSAC



„Standard“ RANSAC line fitting example:

- Randomly pick two points
- Verify how many others fit
- Repeat many times and pick the best one (most matches)

Forward Search



Forward Search:

- Ransac variant
- Like ransac, but refine model by „growing“
- Pick best match, then recalculate
- Repeat until threshold is reached

Ransac-Based Correspondence Estimation

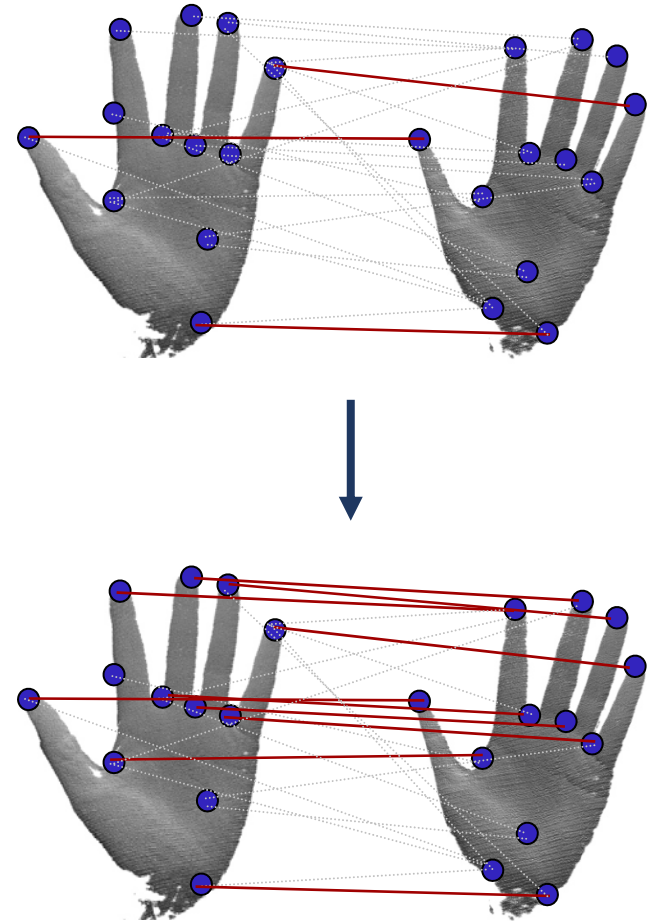
RANSAC Algorithm

RANSAC Idea

- Starting correspondence
- Add more that are consistent
 - Preserve intrinsic distances
- Importance sampling algorithm

Advantages

- Efficient (small initial set)
- General (arbitrary criteria)



Ransac Details

Algorithm: Simple Idea

- Select correspondences with probability proportional to their plausibility
- First correspondence: Descriptors
- Second: Preserve distance (distribution peaks)
- Third: Preserve distance (even fewer choices)
- ...
- Rapidly becomes deterministic
- Repeat multiple times (typ.: 100x)
 - Choose the largest solution (largest #correspondences)

Ransac Details

Provably Efficient:

- Optimal solution in expected $O(n^3 \log n)$ for n candidate correspondences and sphere topology
- Much faster in practice (using descriptors)

Flexible:

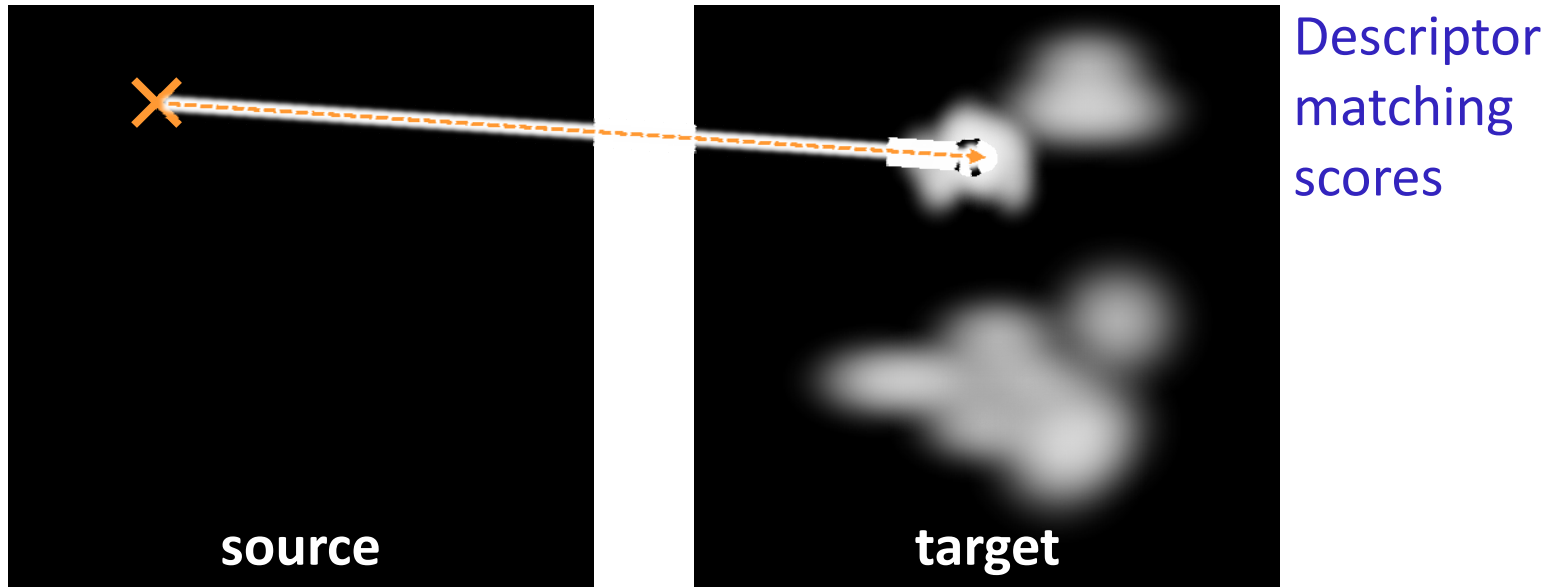
- In later iterations (> 3 correspondences), allow for outlier geodesics
- Can handle topological noise

Forward Search Algorithm

Refined Version: Forward Search

- Add correspondences incrementally
- Compute match probabilities given the information already decided on
- Iterate until no more matches can be found that meet a certain error threshold
- Outer Loop:
 - Iterate the algorithm with random choices
 - Pick the best (i.e., largest) solution

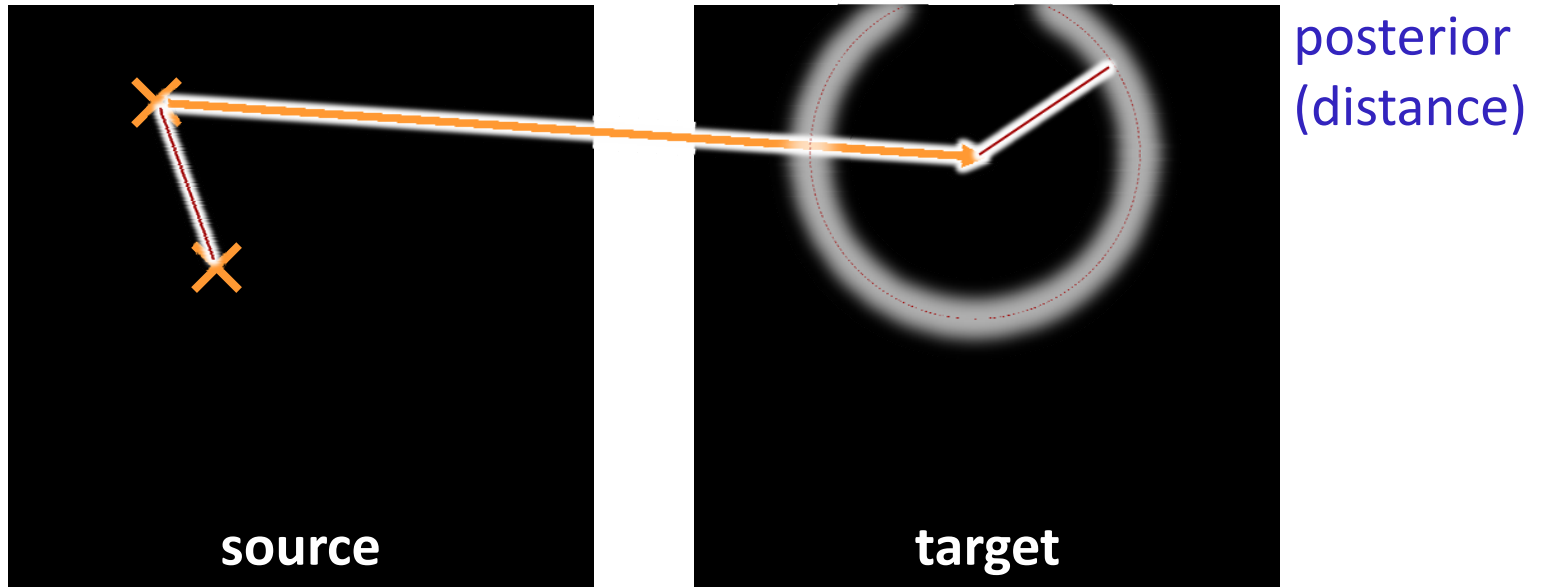
Forward Search Algorithm



Step 1:

- Start with one correspondence
 - Target side importance sampling: prefer good descriptor matches
 - Optional source side imp. sampl: prefer unique descriptors

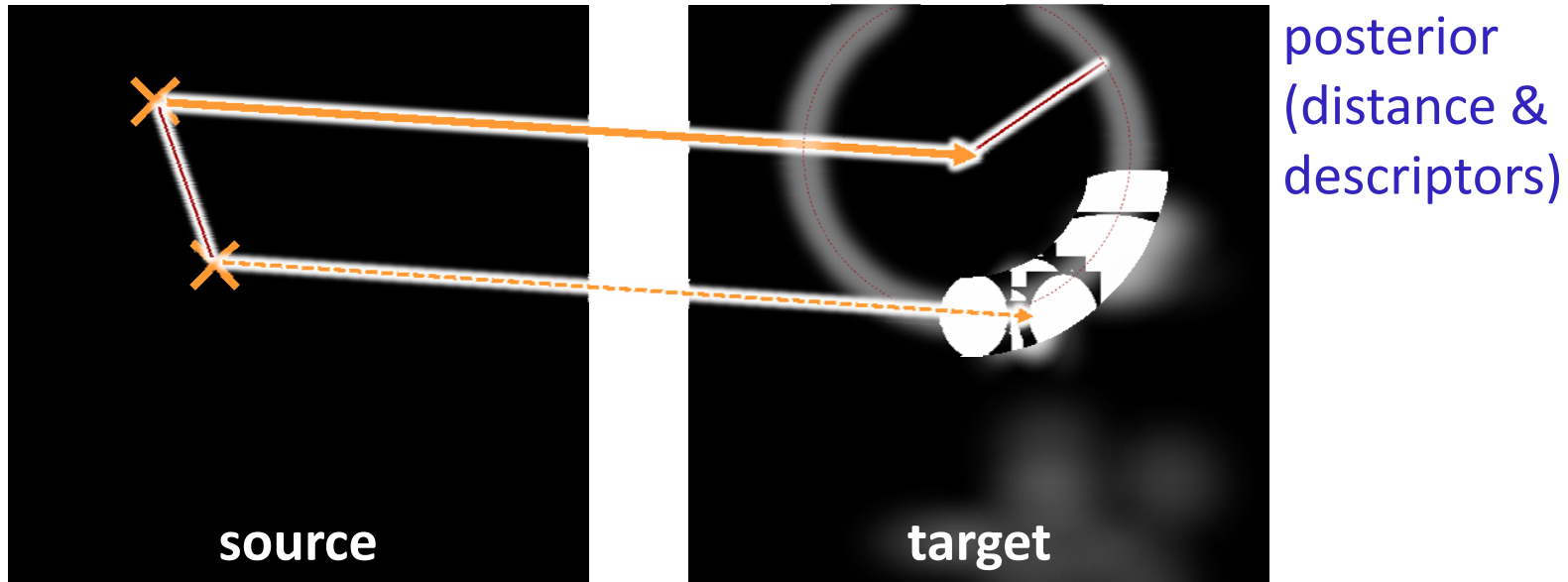
Forward Search Algorithm



Step 2:

- Compute „posterior“ incorporating geodesic distance
 - Target side importance sampling:
sample according to descriptor match \times distance score
 - Again: optional source side imp. sampl: prefer unique descriptors

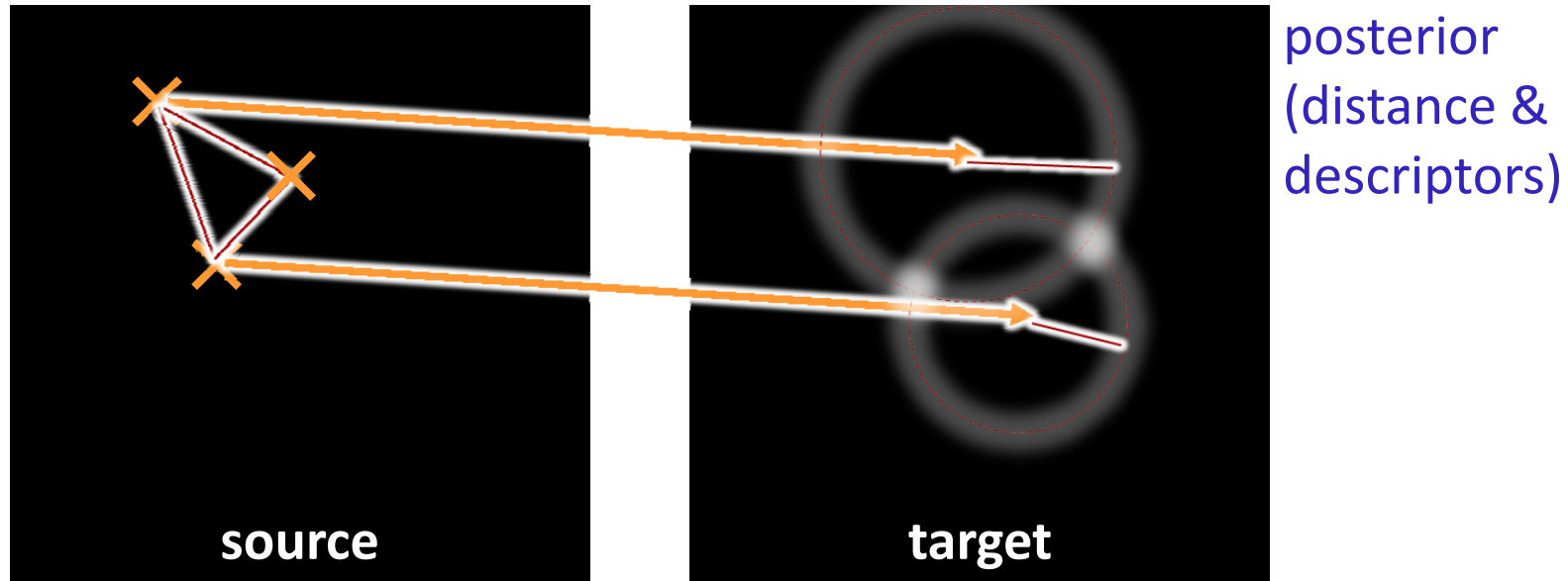
Forward Search Algorithm



Step 2:

- Compute „posterior“ incorporating geodesic distance
 - Target side importance sampling:
sample according to descriptor match \times distance score
 - Again: optional source side imp. sampl: prefer unique descriptors

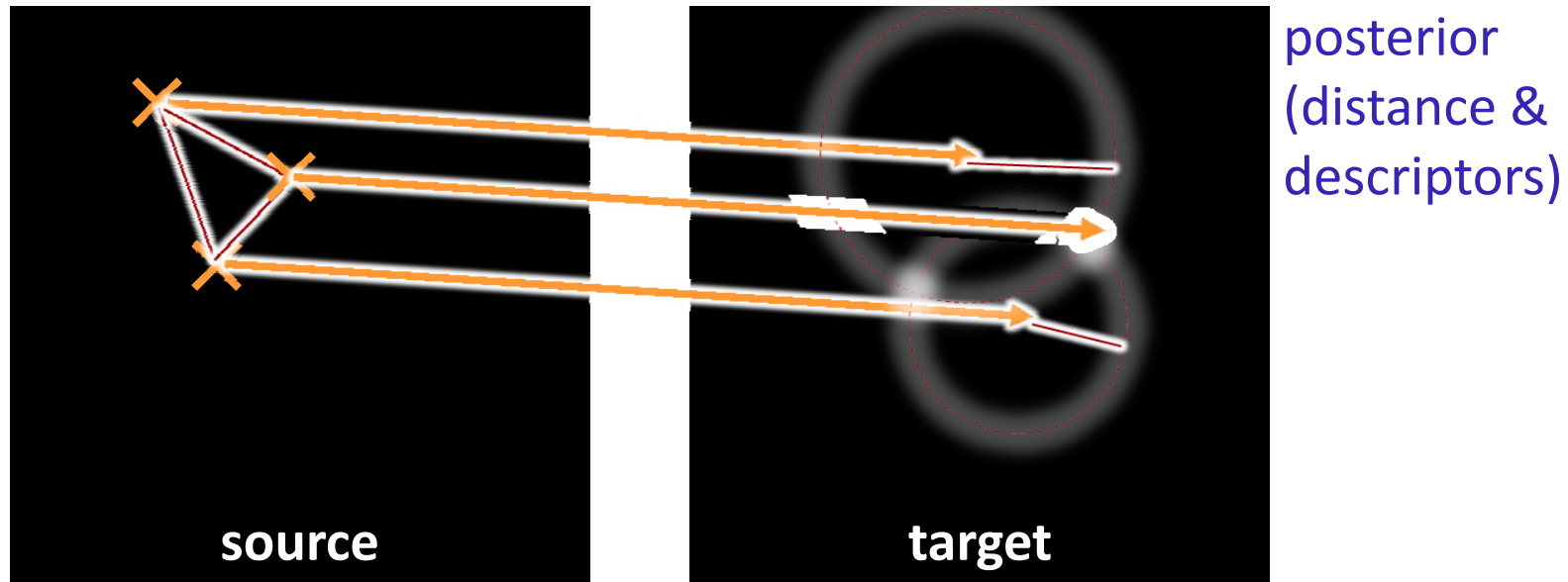
Forward Search Algorithm



Step 3:

- Same as step 2, continue sampling...

Forward Search Algorithm



Step 3:

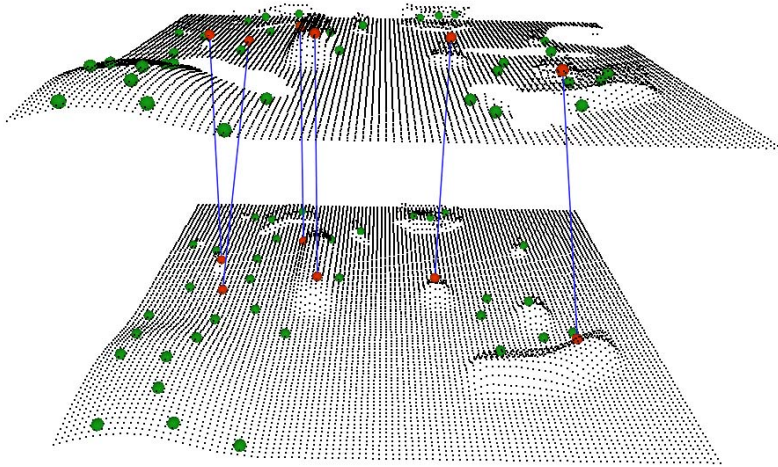
- Same as step 2, continue sampling...

Another View

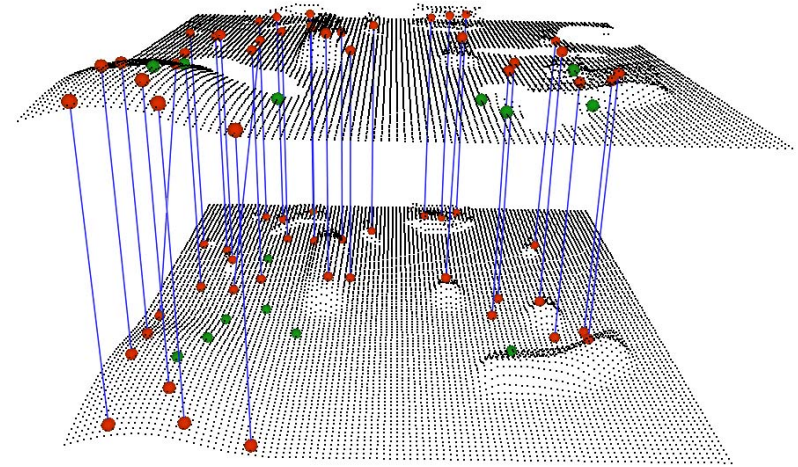
Landmark Coordinates

- Distance to already established points give a charting of the manifold
- How many correspondences are necessary to find a unique map?

Results: Topological Noise

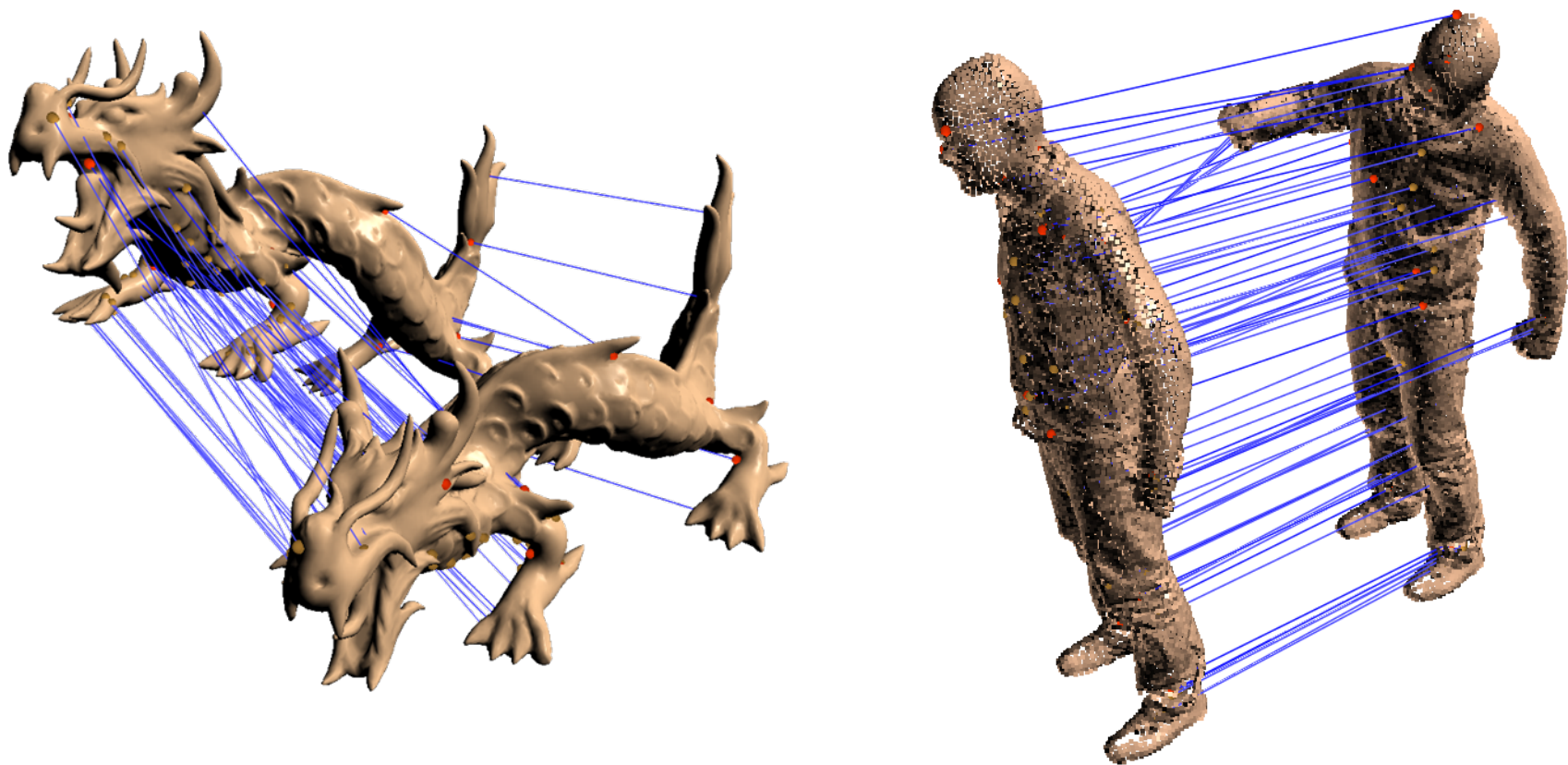


Spectral Quadratic Assignment
[Leordeanu et al. 05]



Ransac Algorithm
[Tevs et al. 09]

Results



[data sets: Stanford 3D Scanning Repository / Carsten Stoll]

Complexity

How expensive is all of this?

Cost analysis:

- How many rounds of sampling are necessary?

Constraints [Lipman et al. 2009]:

- Assume disc or sphere topology
- An isometric mapping is in particular a conformal mapping
- A conformal mapping is determined by 3 point-to-point correspondences

How expensive is it..?

First correspondence:

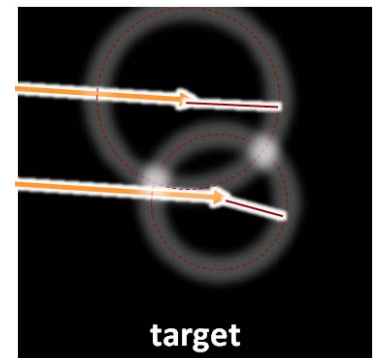
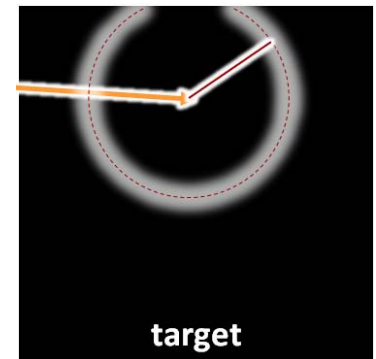
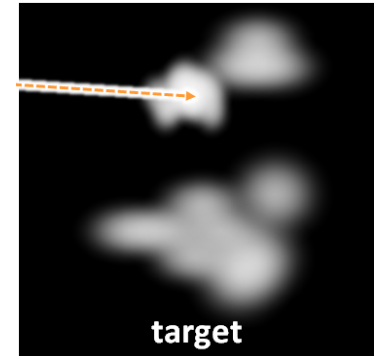
- Worst case: n trials (n feature points)
- In practice: $k \ll n$ good descriptor matches (typically $k \approx 5-20$)

Second correspondence:

- Worst case: n trials, expected: \sqrt{n} trials
- In practice: very few (due to descriptor matching, maybe 1-3)

Last match:

- At most two matches



Costs...

Overall costs:

- Worst case: $O(n^2)$ matches to explore
- Typical: $O(n^{1.5})$ matches to explore

Randomization:

- Exploring m items costs expected $O(m \log m)$ trials
- Worst case bound of $O(n^2 \log n)$ trials
- Asymptotically sharp: $O(c)$ -times more trials for shrinking failure probability to $O(\exp(-c^2))$

Costs...

Surface discretization:

- Assume ε -sampling of the manifold (no features): $O(\varepsilon^{-2})$ sample points
- Worst case $O(\varepsilon^{-4} \log \varepsilon^{-1})$ sample correspondences for finding a match with accuracy ε .
- Expected: $O(\varepsilon^{-3} \log \varepsilon^{-1})$.

In practice:

- Importance sampling by descriptors is very effective
- Typically: Good results after 100 iterations

General Case

Numerical errors:

- Noise surfaces, imprecise features: reflected in probability maps (we know how little we might know)

Topological noise:

- Use robust constraint potentials
- For example: account for 5 best matches only

Topologically complex cases:

- No analysis beyond disc/spherical topology
- However: the algorithm will work in the general case (potentially, at additional costs)

Articulated Registration

Graph cuts and piecewise-rigid registration [CZ08]

Articulated registration [CZ09]

Implementation issues and alternatives

Overview

Examine recent articulated registration methods

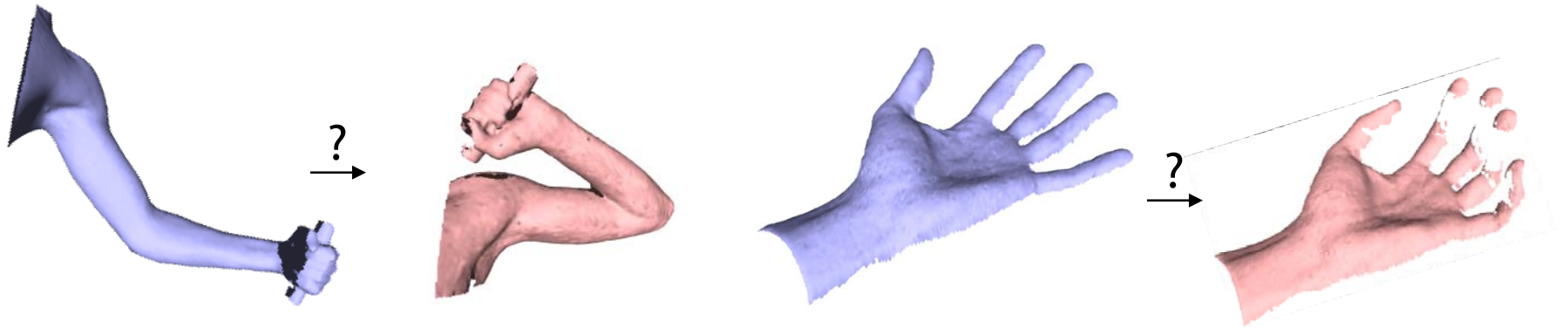
- [CZ08]: Global method able to deal with partial data
- [CZ09]: Local method to determine a deformable model, similar to multi-part ICP but with joint constraints

Articulation assumption simplifies non-rigid registration

Problem Statement

Solve pairwise registration problem

- Develop robust method independent of initial pose
- Do not require markers, template, or user-painted segmentation

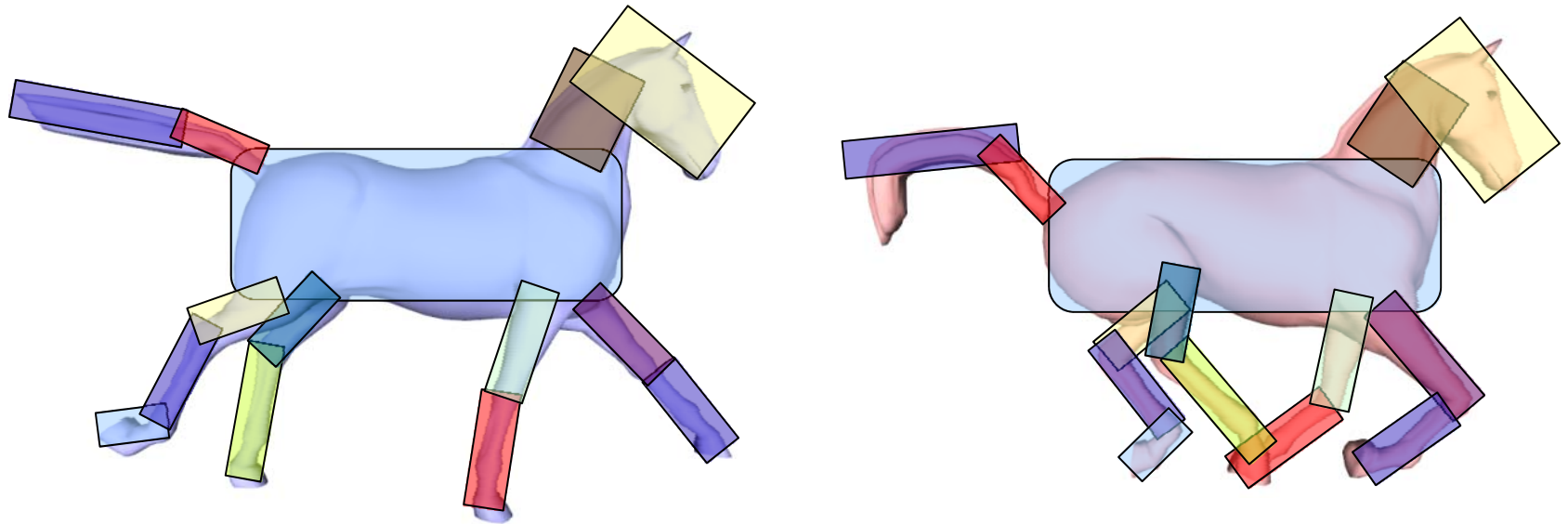


[CZ08] Algorithm Overview

Articulated motion \rightarrow small set of transformations

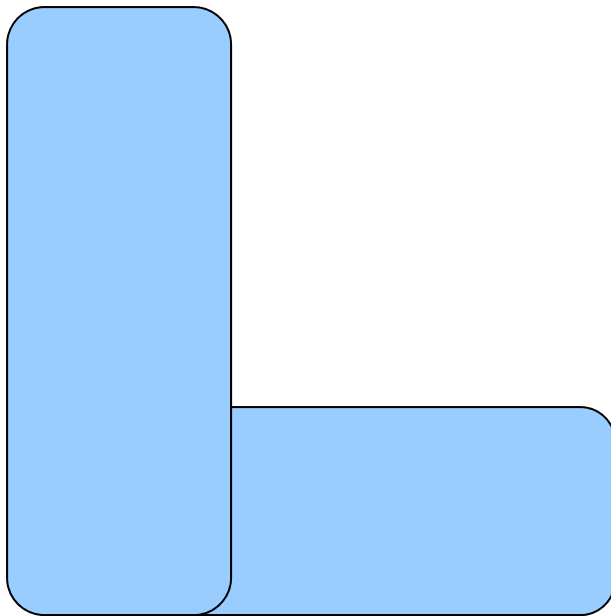
Predetermine a set of transformations describing the motion

Optimize assignment of transformations to the points

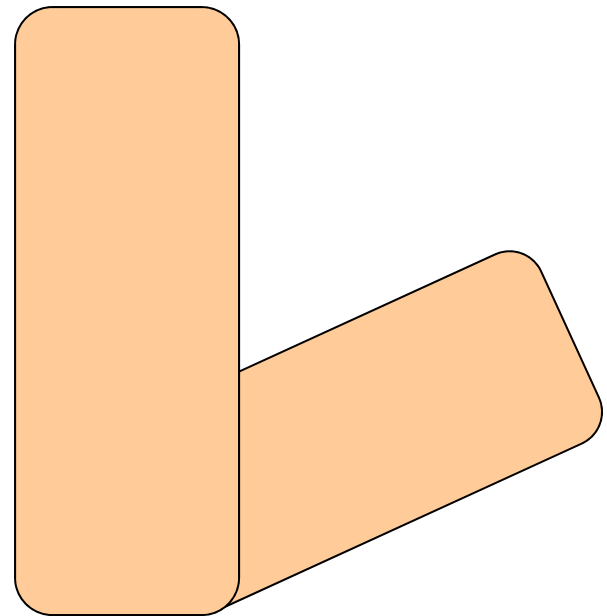


Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target



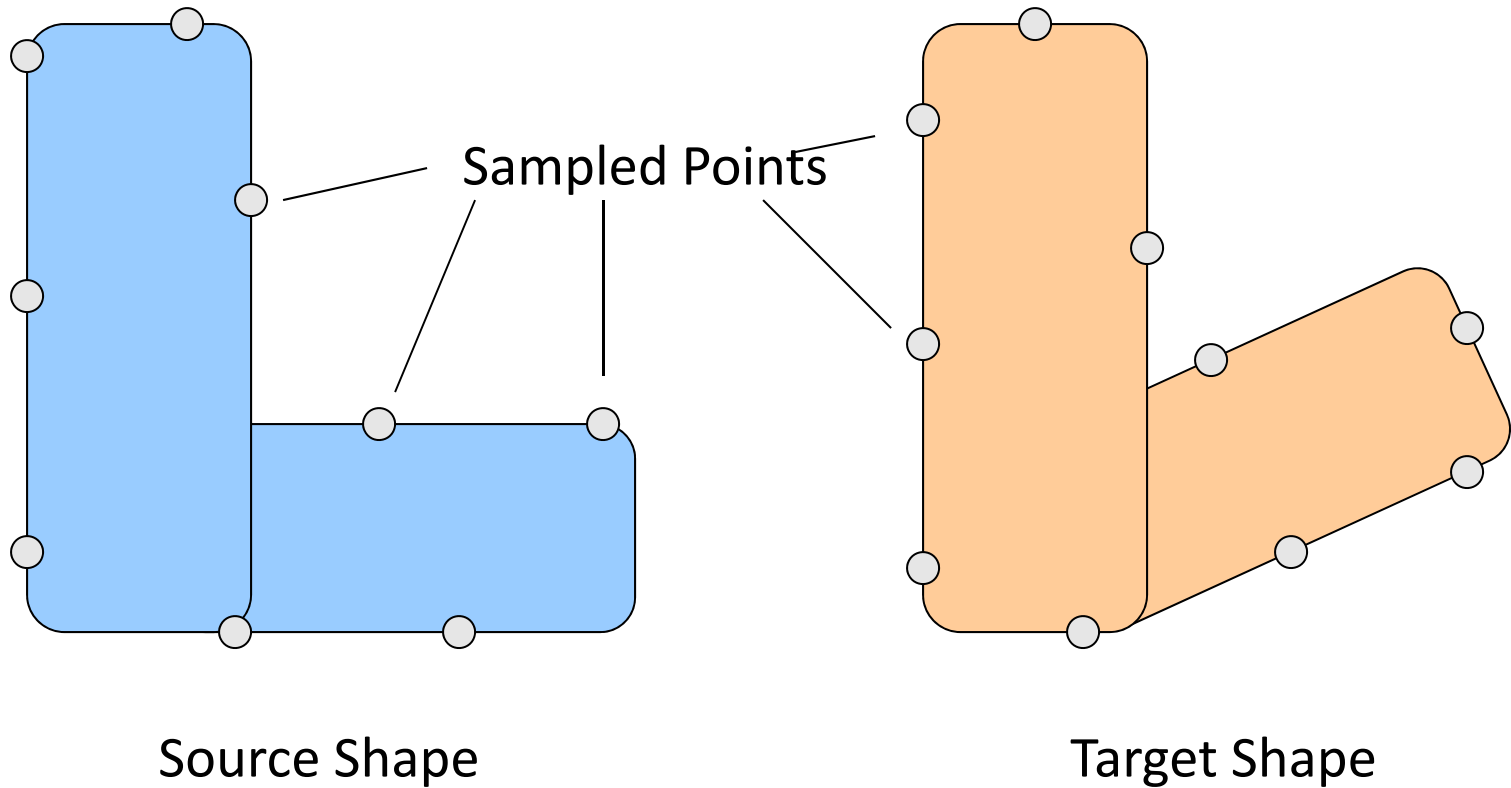
Source Shape



Target Shape

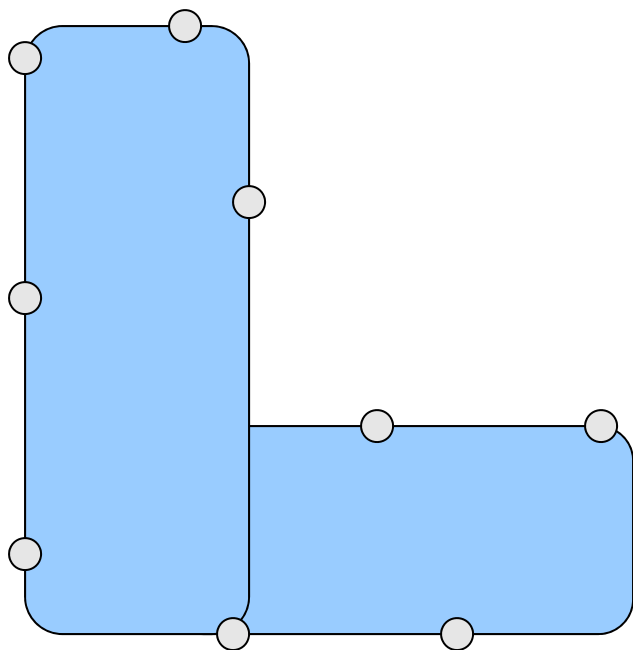
Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target

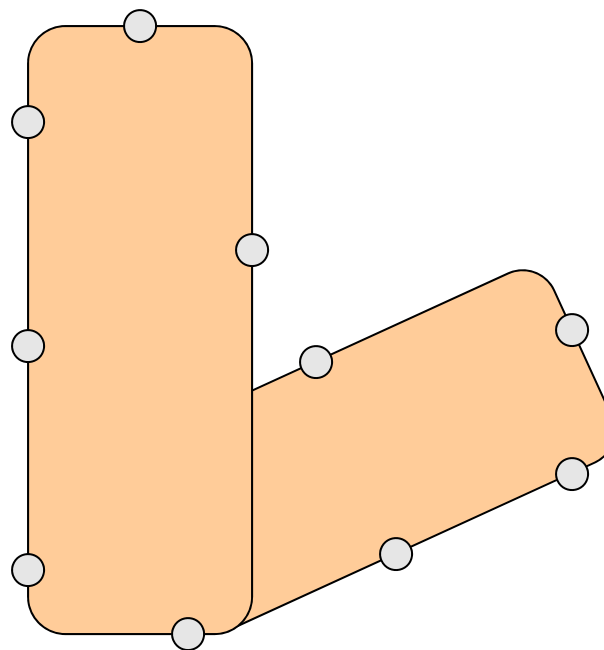


Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target



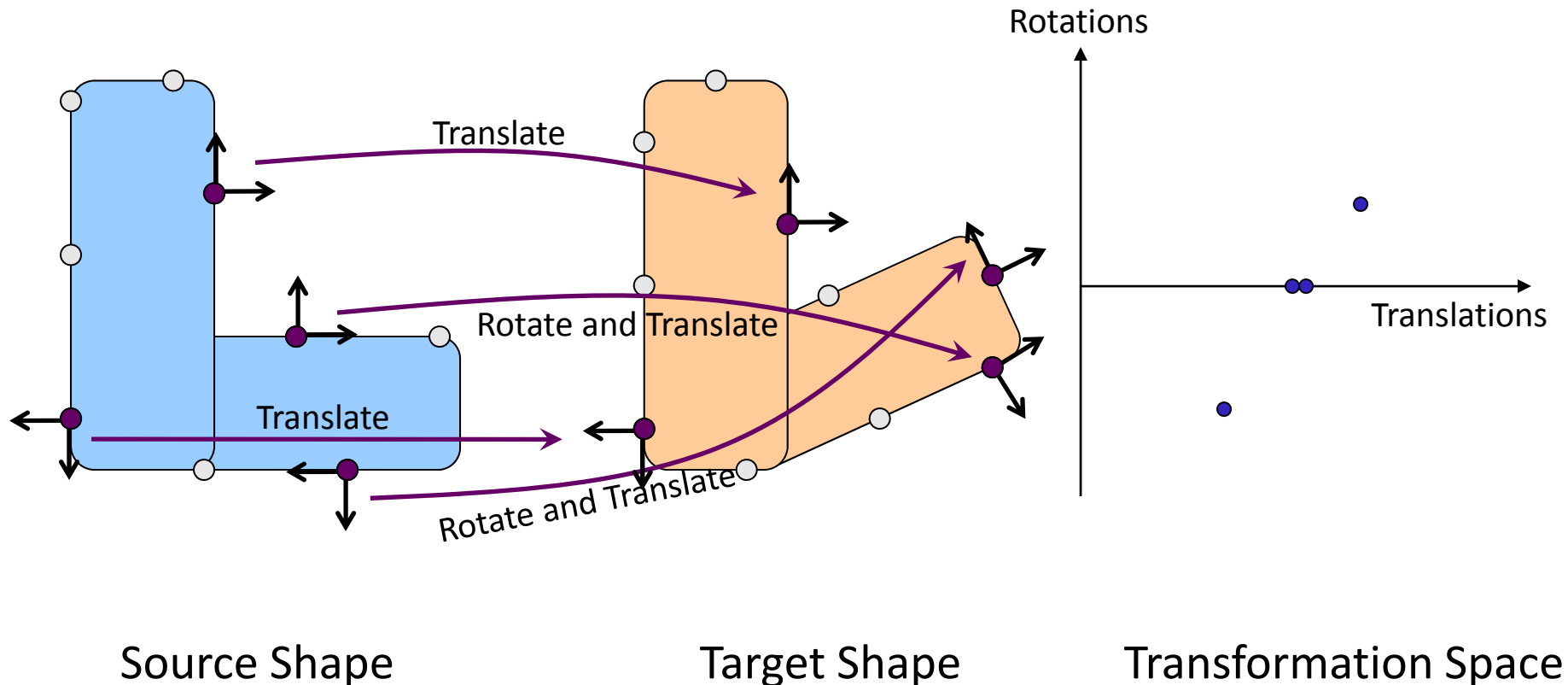
Source Shape



Target Shape

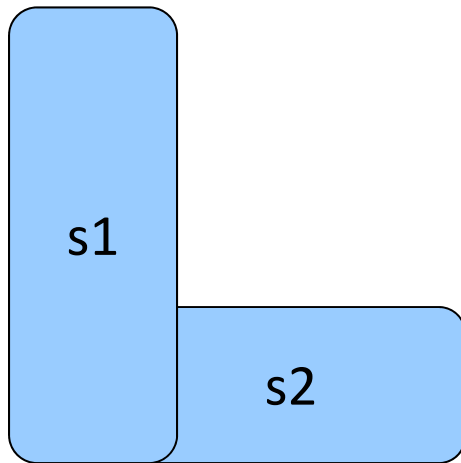
Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target

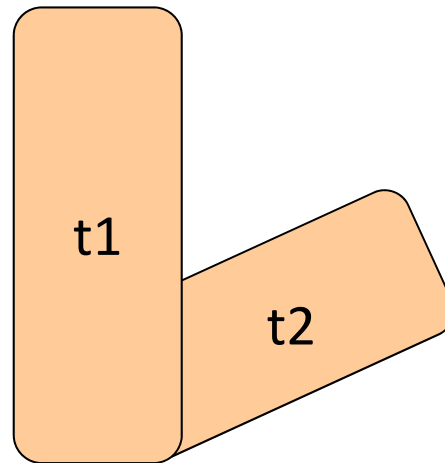


Motion Sampling Illustration

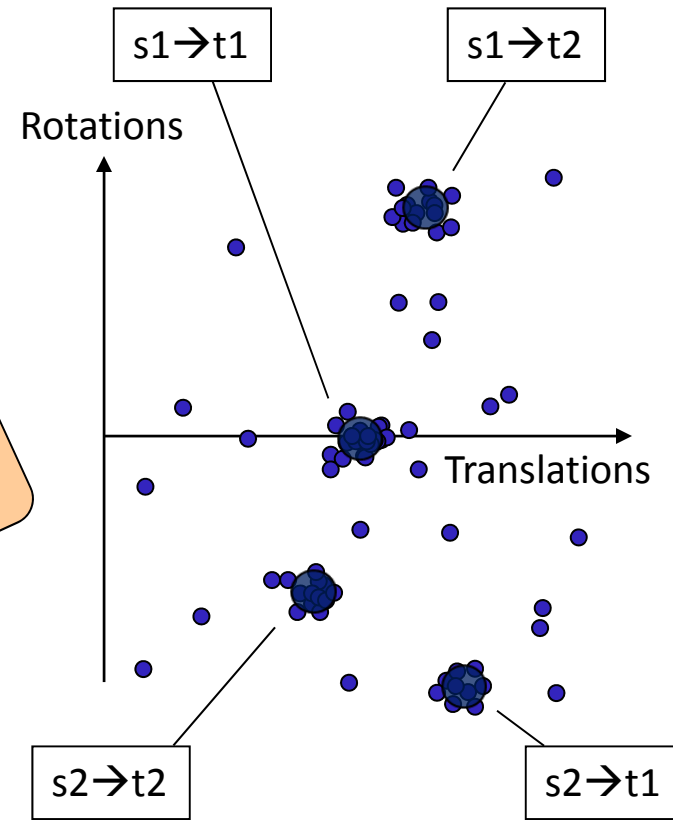
Find transformations that move parts of the source to parts of the target



Source Shape



Target Shape



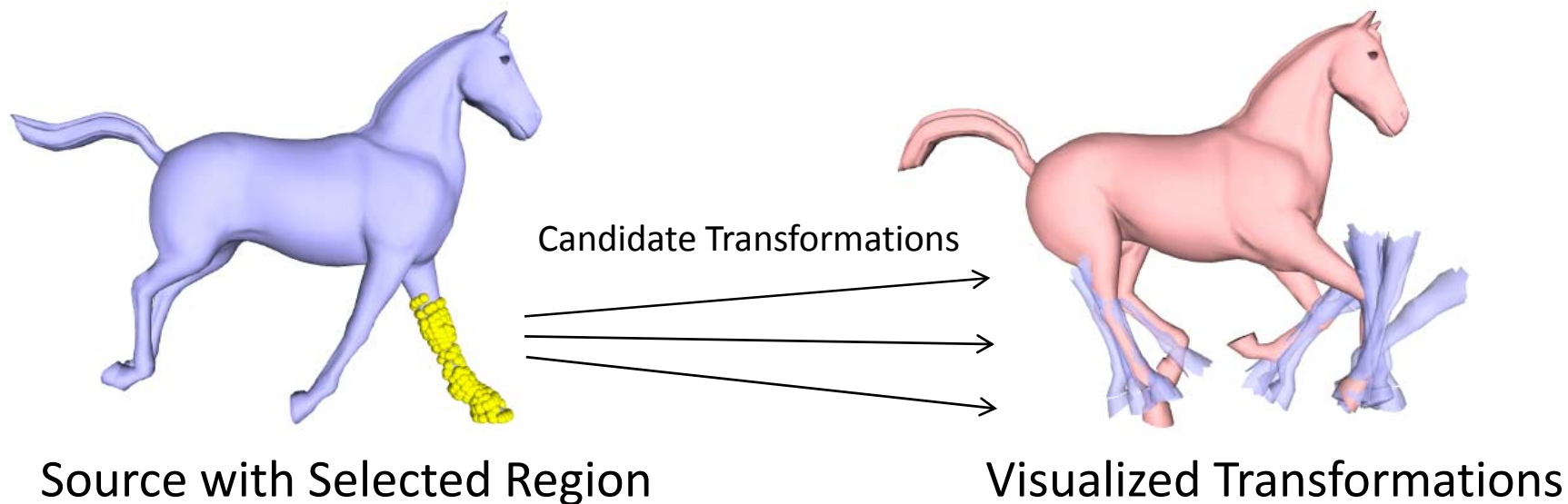
Transformation Space

Limitations of Motion Sampling

Final Output: finite set of rigid transformations

If there are multiple similar parts

- Does not figure out the correct part
- Disambiguate in the optimization step



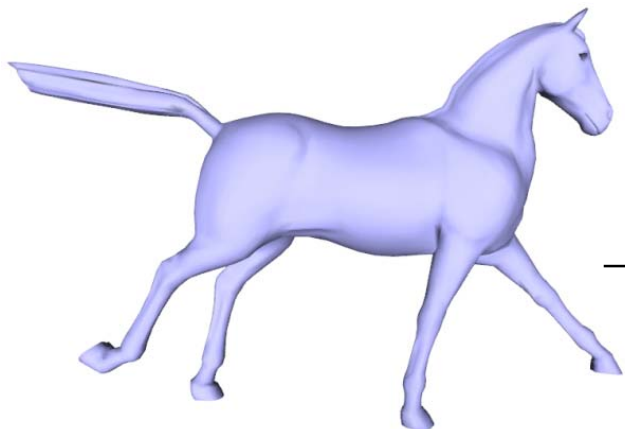
Global Motion Optimization

Optimize an *assignment* from a finite set of transformations

argmin **Data Cost** + **Smoothness Cost**

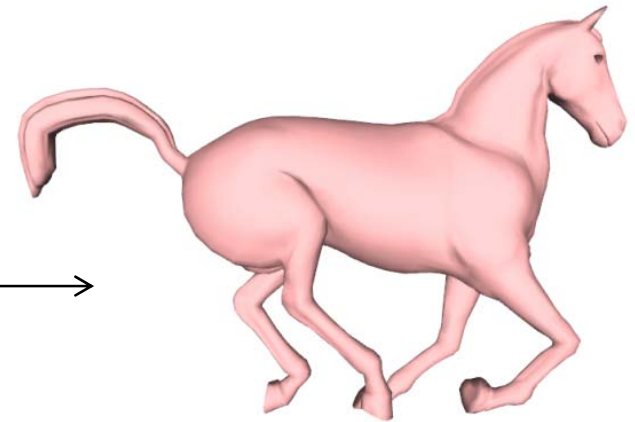
*Assignment from
a set of transformations*

A discrete labelling problem → **Graph Cuts for optimization**



Source Shape \mathcal{P}

Transformations
from finite set



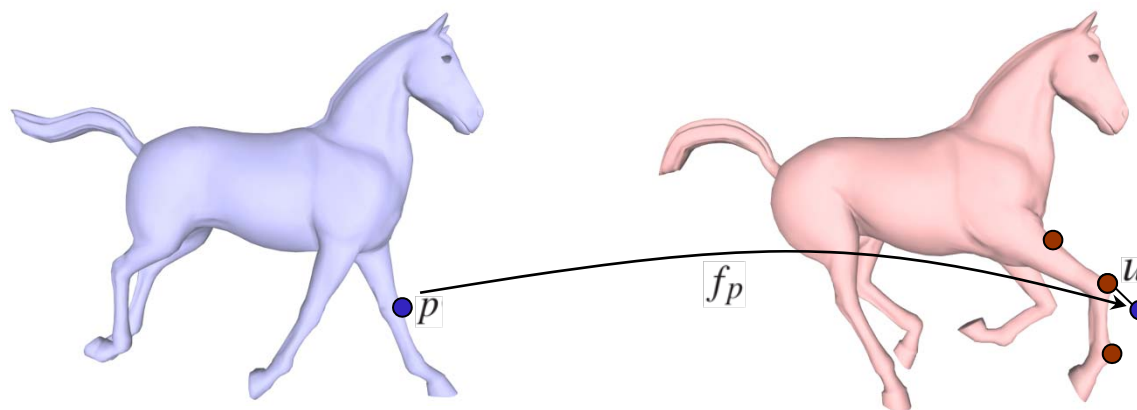
Target Shape \mathcal{U}

Data Term

Move all points as close as possible to the target

How to measure distance to target?

- Apply selected transformation f_p to all p $f_p(p)$
- Measure distance to closest point u target

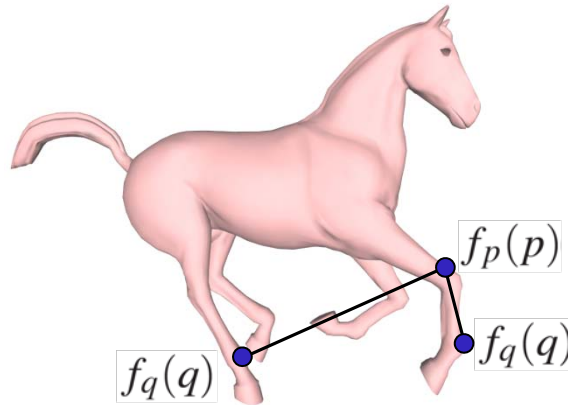
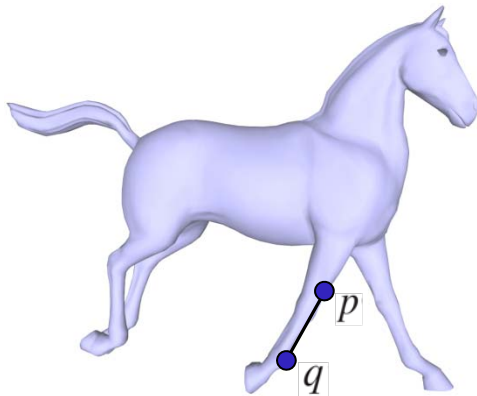


Smoothness Term

Preserve edge length between neighboring points

$$V(p, q, f_p, f_q) = \left| \underbrace{\|p - q\|}_{\text{Original Length}} - \underbrace{\|f_p(p) - f_q(q)\|}_{\text{Transformed Length}} \right|$$

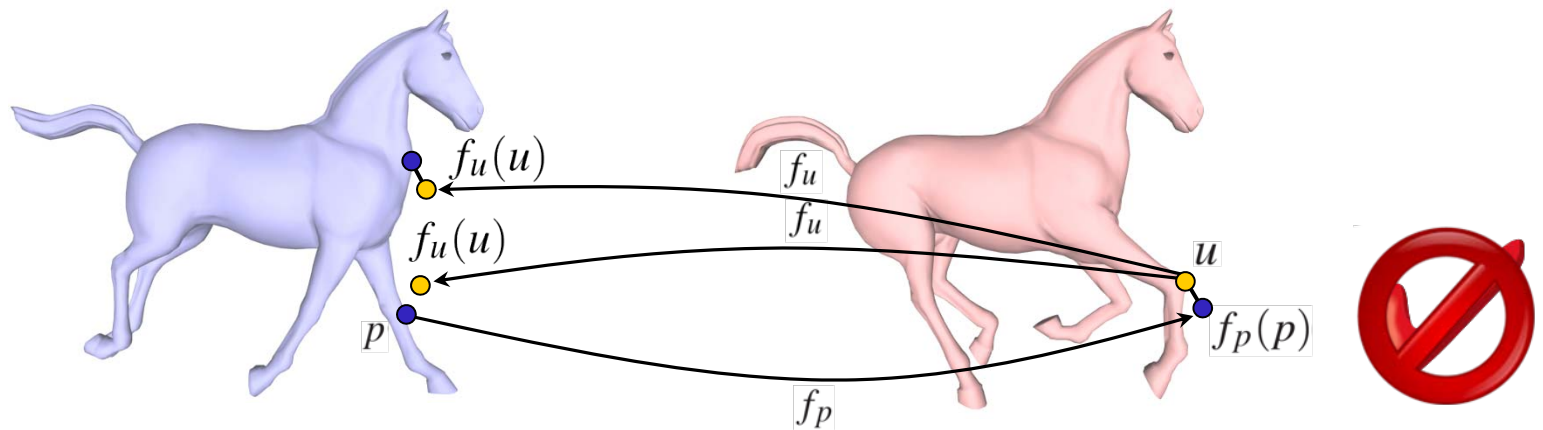
Disambiguates multiple possible mappings



Symmetric Cost Function

Swapping source / target can give different results

- Optimize assignment in both meshes (forward & backward)
- Enforce consistent assignment: penalty when $f_p \neq f_u$



$f_p \neq f_u$ No Penalty

Optimization Using Graph Cuts

argmin
*Assignment from a set
of transformations*

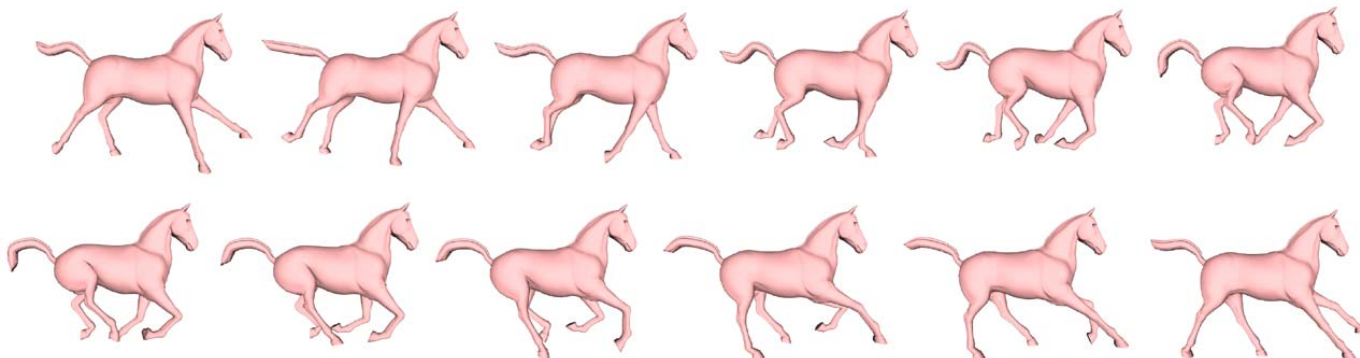
$$\text{Data}_{\text{Source}} + \text{Smoothness}_{\text{Source}} + \text{Data}_{\text{Target}} + \text{Smoothness}_{\text{Target}} + \text{Symmetric Consistency}_{\text{Source \& Target}}$$

- **Data** and **smoothness** terms apply to both shapes
- Additional **symmetric consistency** term
- Weights to control relative influence of each term
- Use “graph cuts” to optimize assignment
 - [Boykov, Veksler & Zabih PAMI '01]

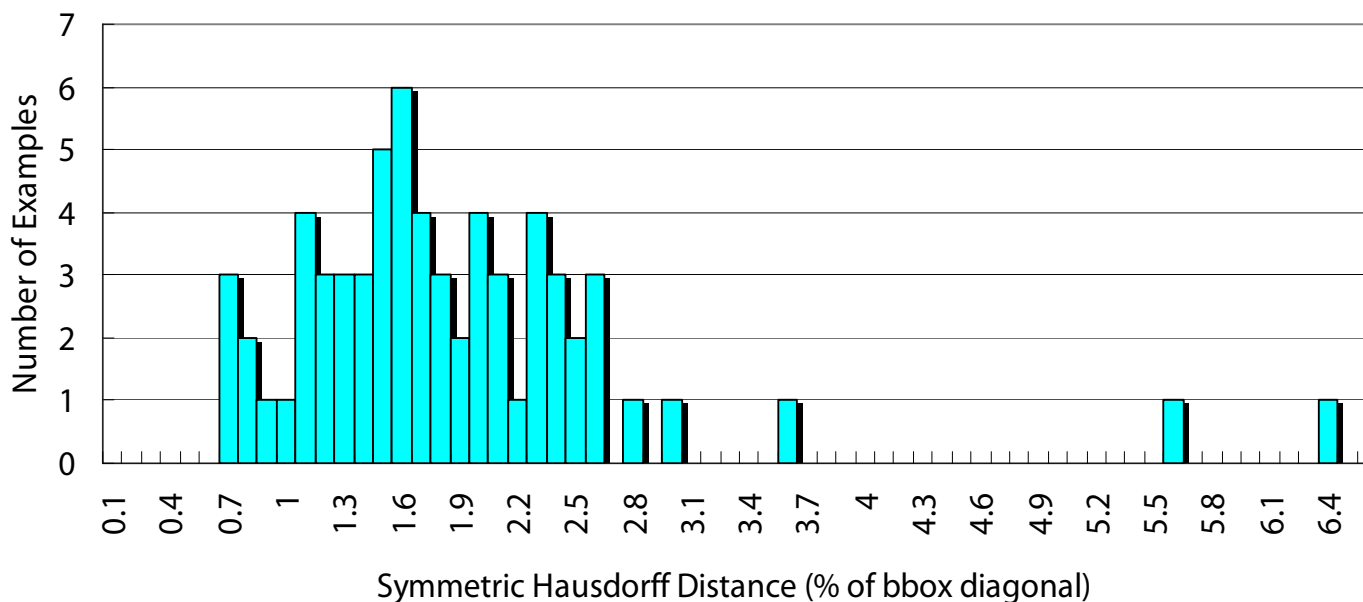
Horse Dataset Results

12 poses of galloping horse: total of 66 pairs. correct

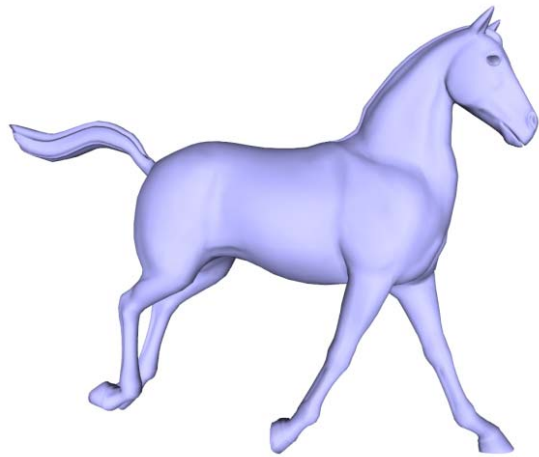
leg 1



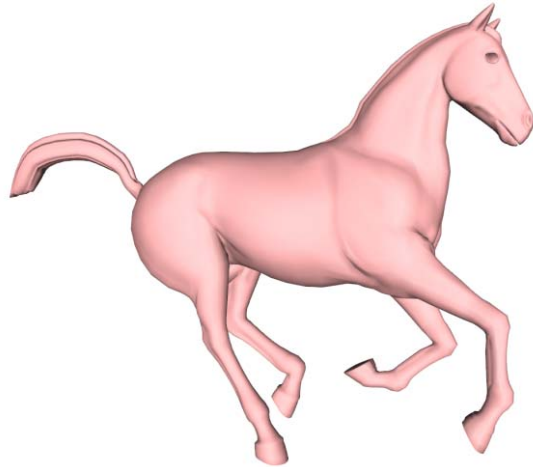
Histogram of Error in Galloping Horse Dataset (minimum over 3 trials)



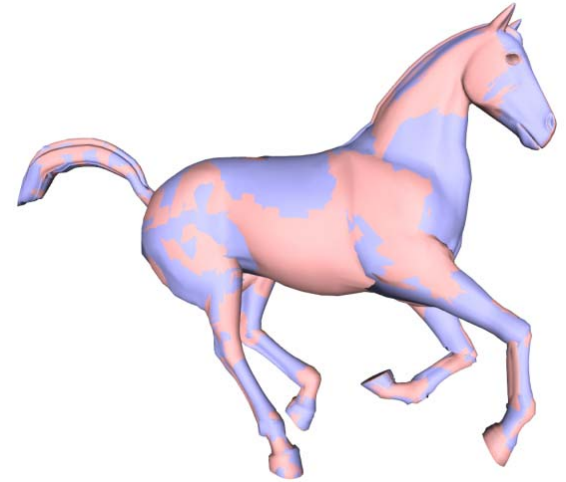
Synthetic Dataset Example



Source



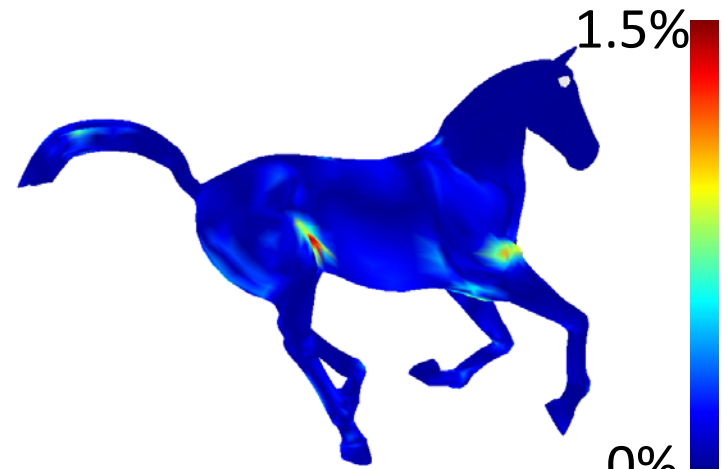
Target



Aligned Result

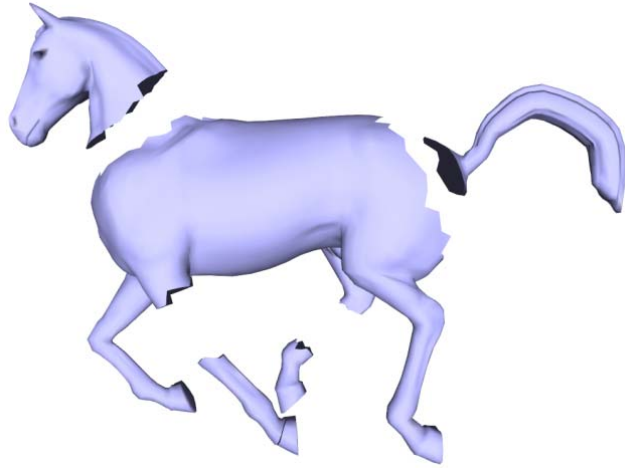


Motion Segmentation (from Graph Cuts)

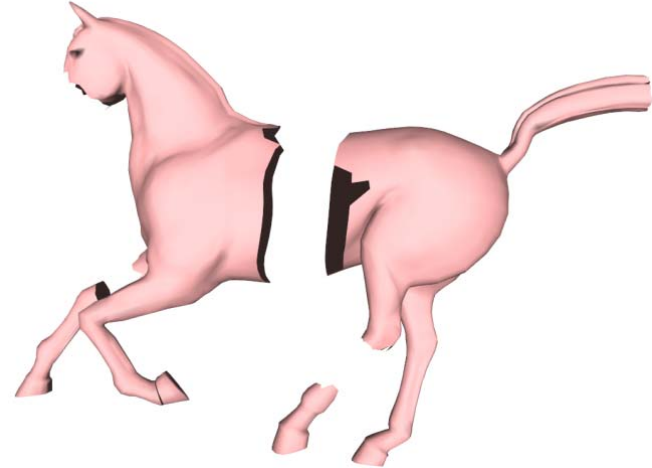


Registration Error

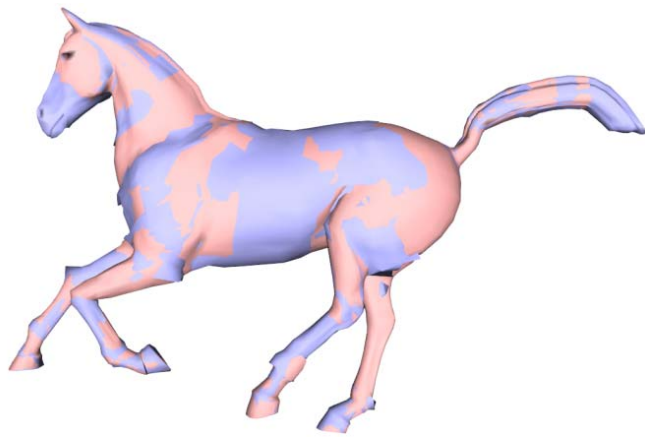
Synthetic Dataset w/ Holes



Source



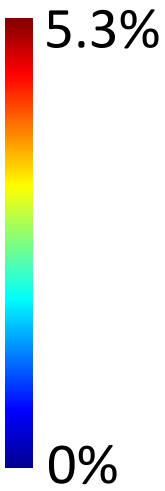
Target



Aligned Result



Distance (from Target) to the closest point
(% bounding box diagonal)

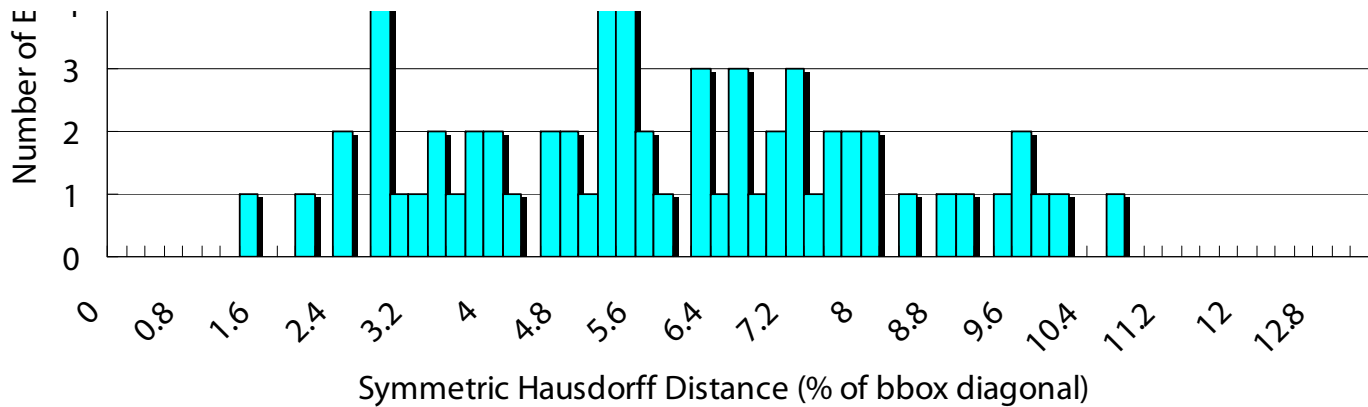


5.3%

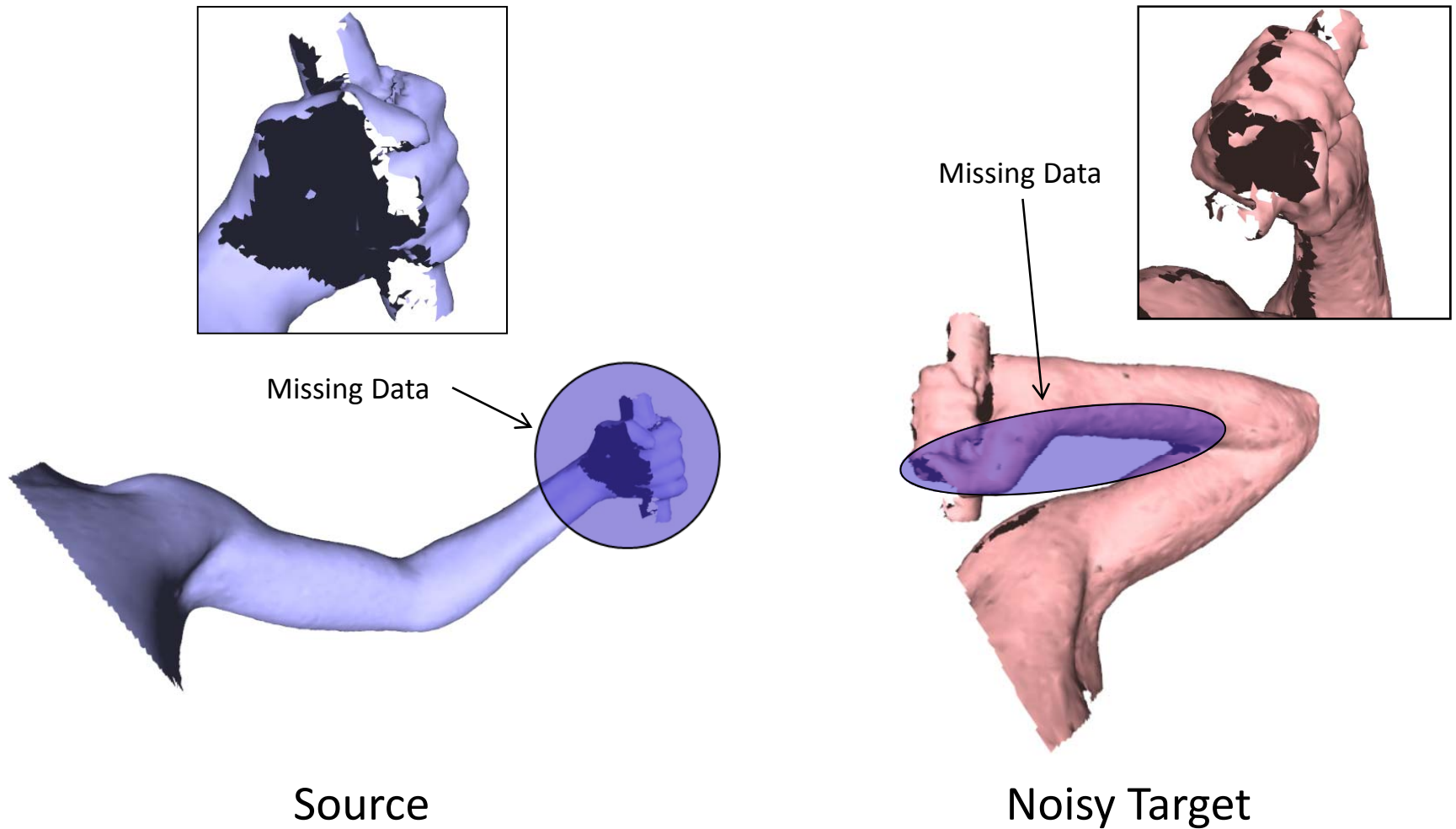
0%

Arm Dataset Results

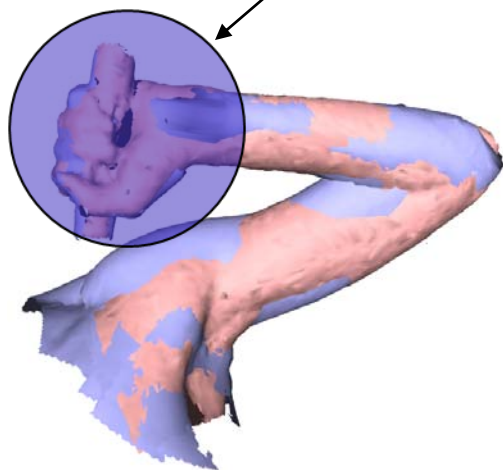
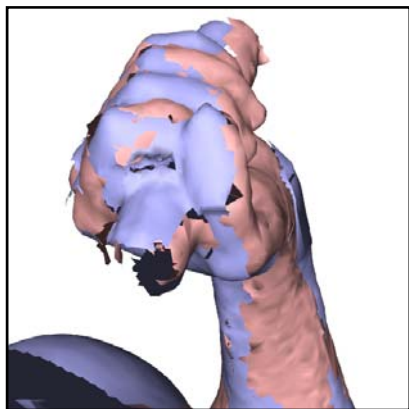
12 poses of arm scans: total of 66 pairs, arm & hand orientation matched in all pairs



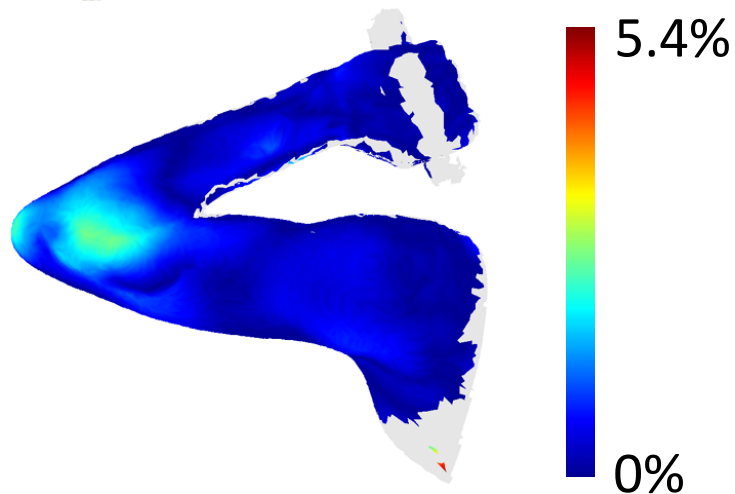
Arm Dataset Example



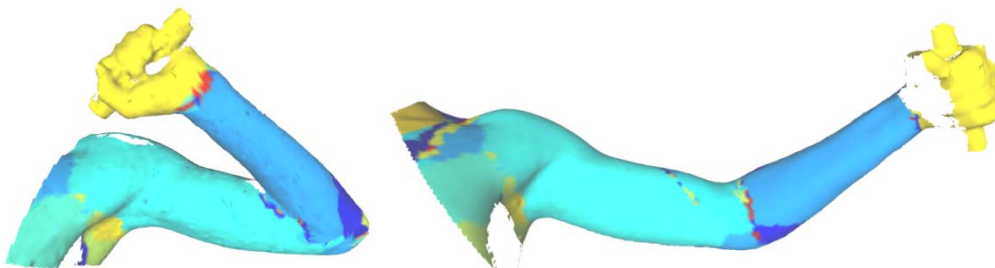
Arm Dataset Example



Aligned Result

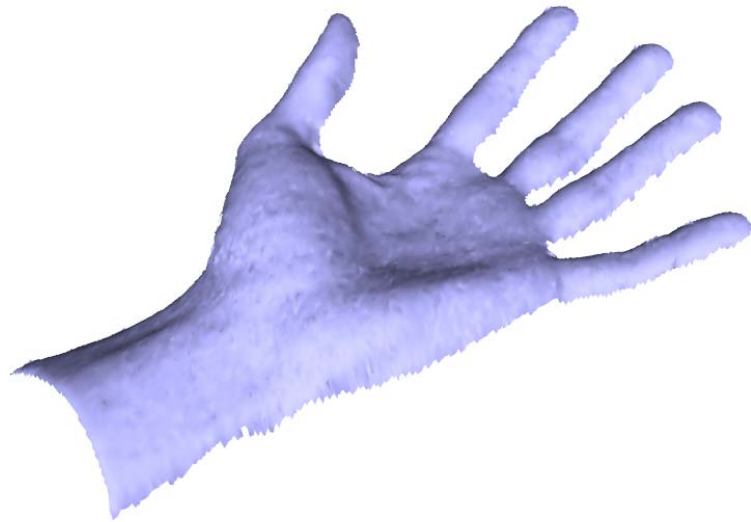


Distance (from Target) to the closest point
(% bounding box diagonal)

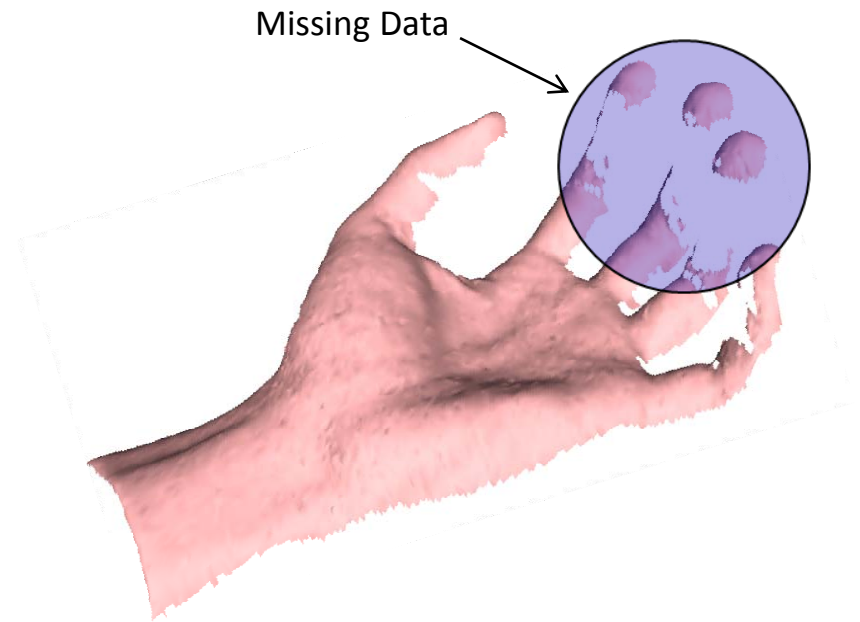


Motion Segmentation

Hand Dataset Example

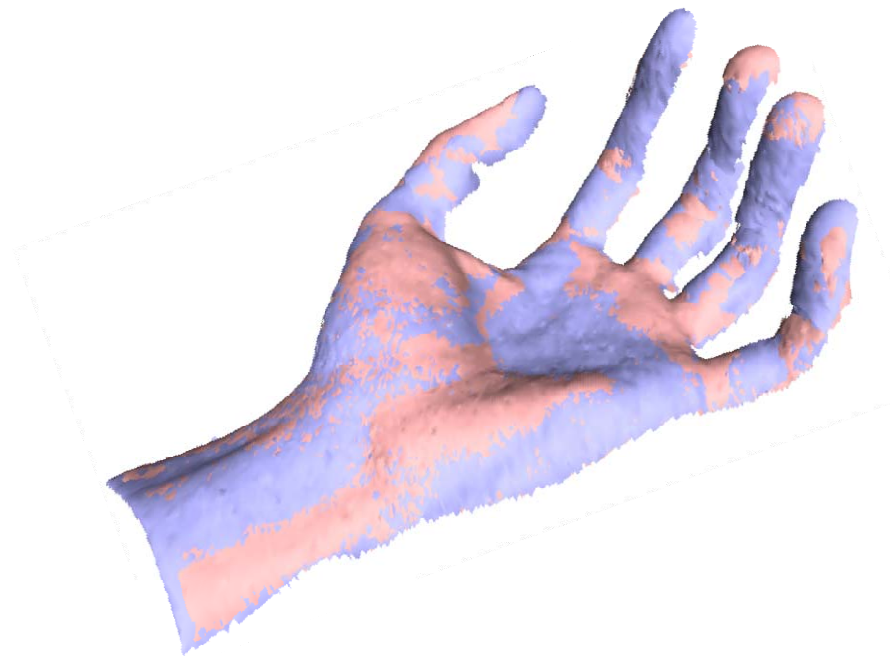


Source

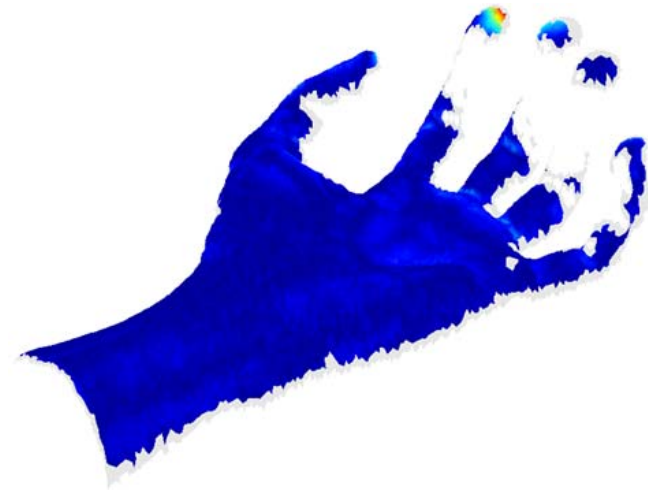


Target

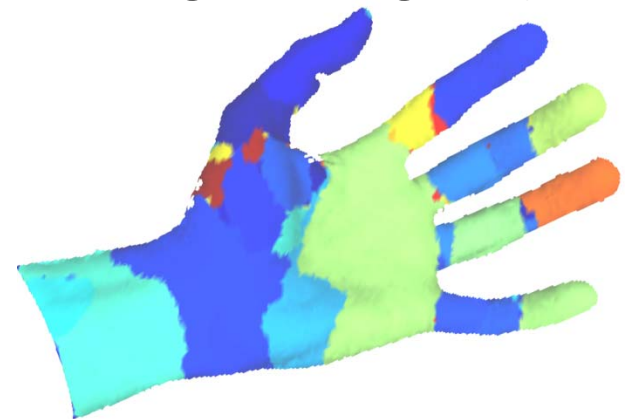
Hand Dataset Example



Aligned Result



Distance (from Target) to the closest point
(% bounding box diagonal)



Motion Segmentation

Performance

| Dataset | #Points | # Labels | Matching | Clustering | Pruning | Graph Cuts |
|--------------|---------|----------|----------|------------|----------------|------------|
| Horse | 8431 | 1500 | 2.1 min | 3.0 sec | (skip) 1.6 sec | 1.1 hr |
| Arm | 11865 | 1000 | 55.0 sec | 0.9 sec | 12.4 min | 1.2 hr |
| Hand (Front) | 8339 | 1500 | 14.5 sec | 0.7 sec | 7.4 min | 1.2 hr |
| Hand (Back) | 6773 | 1500 | 17.3 sec | 0.9 sec | 9.4 min | 1.6 hr |

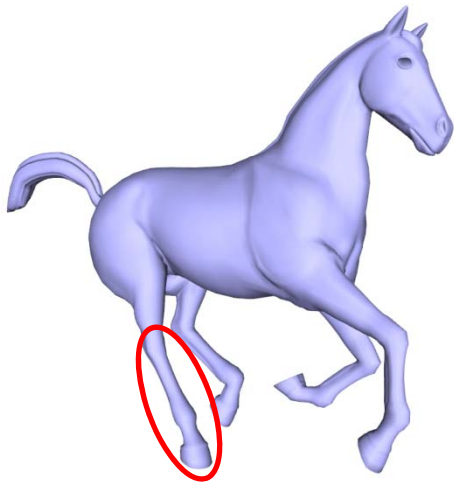
Graph cuts optimization is most time-consuming step

- Symmetric optimization doubles variable count
- Symmetric consistency term introduces many edges

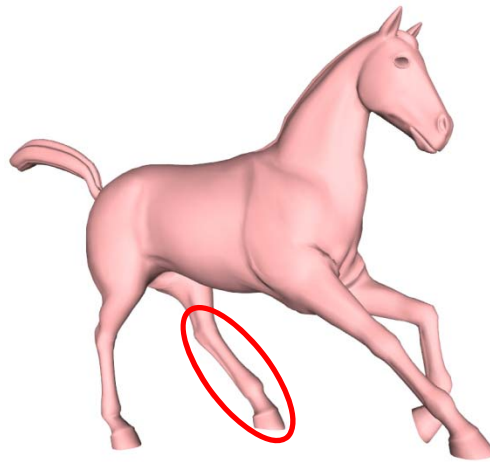
Limitations

Errors in registration

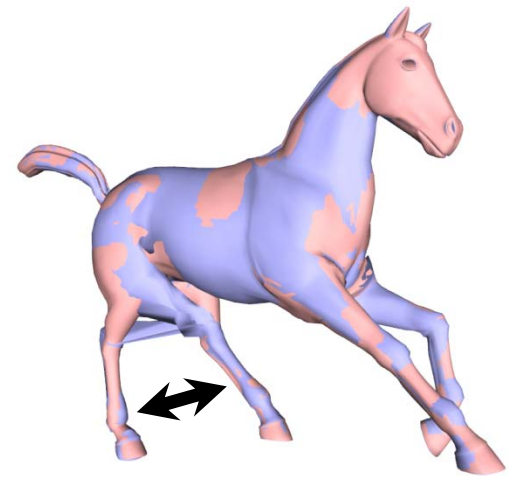
- Trade-off between data and smoothness costs
 - Data weight too high → May break smoothness
 - Smoothness weight too high → Prefer bad alignment



Source



Target

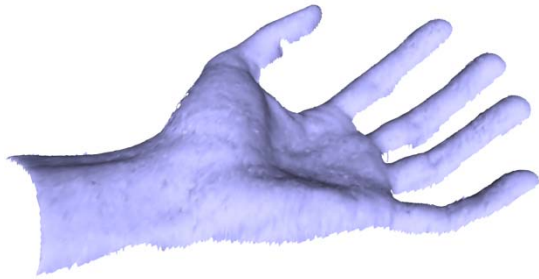


Registration

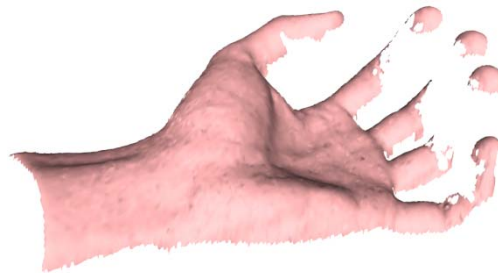
Limitations

Errors in registration

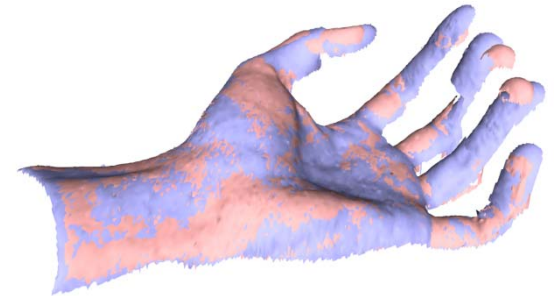
- Motion sampling: may fail to sample properly when too much missing data, non-rigid motion
- Hard assignment of transformations



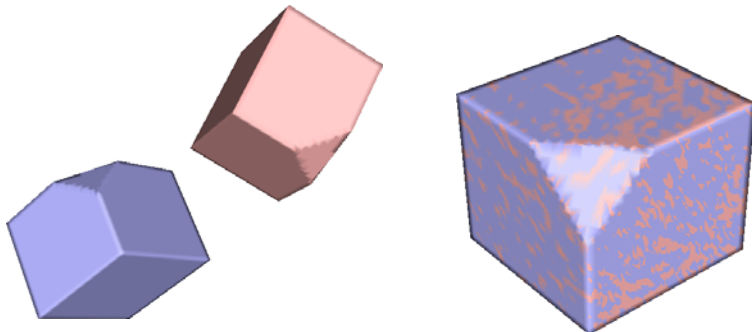
Source



Target



Registration



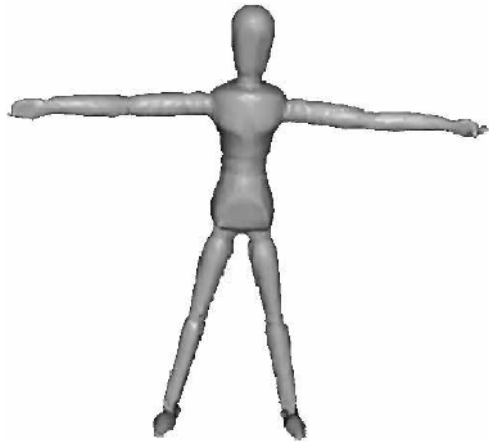
Compare to CC algorithm

Correlated correspondence

- Template required
- No articulation needed
- Optimizes assignment of corresponding points

This method

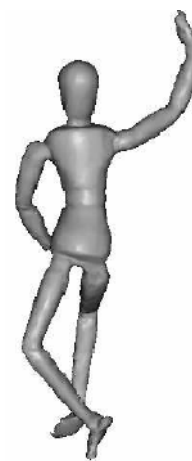
- No template required
- Articulation needed
- Optimizes assignment of transformations



Template Model



Partial Example



Registered result



Ground Truth

(from Angelov et al. 2004)

Implementation Issues

Computing principal curvatures

- See papers by Rusinkiewicz, Kalogerakis et al.

Best-candidate sampling

- Speed boosted by using ANN library
- Further speed up by reconstructing kd-tree every 100 pts

Mean-shift clustering

- Clustering performed directly on transformations in $se(3)$

Verification using ICP

- See Mitra et al. (partial symmetry detection)

Implementation Issues

Setting up symmetric graph instance

- Graph sites include vertices in both source and target
- For each label L , edge between source pt and closest corresponding target pt (when source pt is transformed using L)

Performing graph-cut optimization

- Use publicly downloadable implementation by Boykov, Veksler, Kolmogorov

Conclusions

Automatic method for registering articulated shapes

- No template, markers, or manual segmentation needed
- Explicitly sample a discrete set of motion
- Optimize the assignment of transformations
- Graph cut result gives intuitive segmentation

Useful for obtaining a robust initialization of the registration

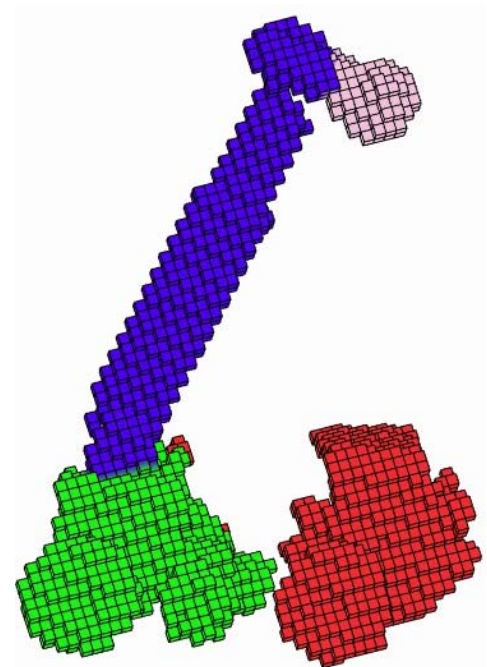
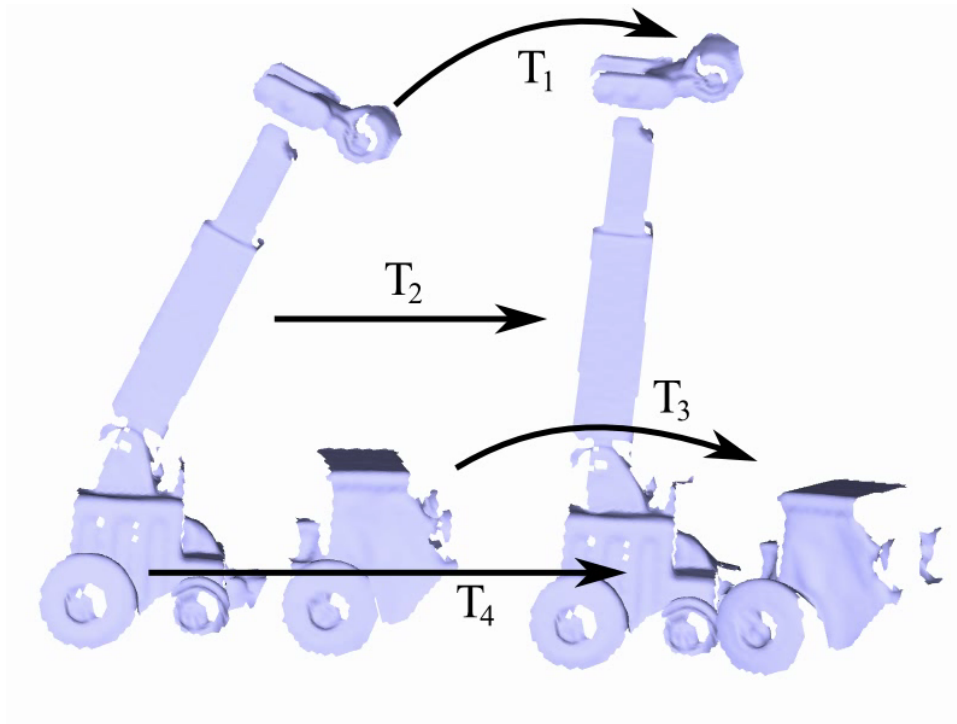
- Does not provide an articulated motion model

Range Scan Registration Using Reduced Deformable Models

Problem Statement

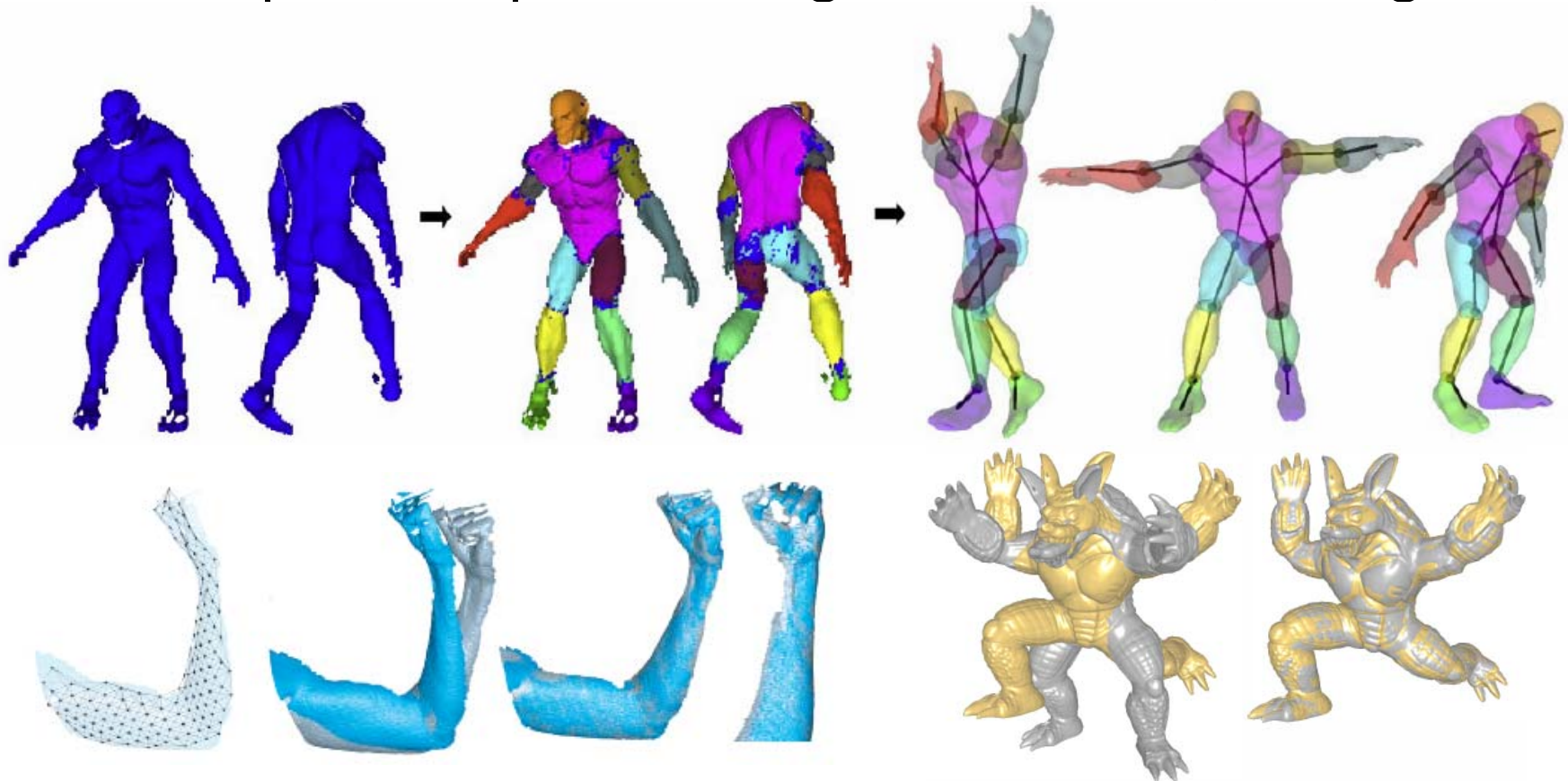
Fit a model of the surface motion to a pair of scans

- Articulated model (e.g. joints, smooth weights)
- Serves as the basis for fitting on multiple frames



Related Work

- User provided segmentation: Pekelnny08
- Unsupervised pairwise registration: Li08, Huang08



(from Pekelnny and Gotsman 2008, Li et al. 2008 and Huang et al. 2008)


Model: Linear Blend Skinning

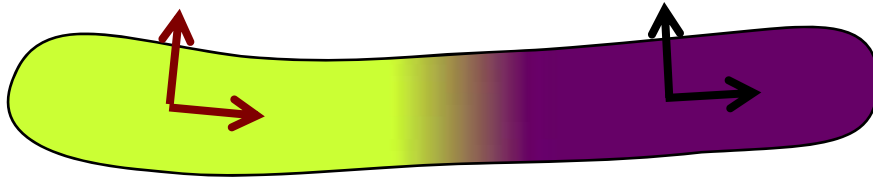
Transformations (bones) and weights

Shape



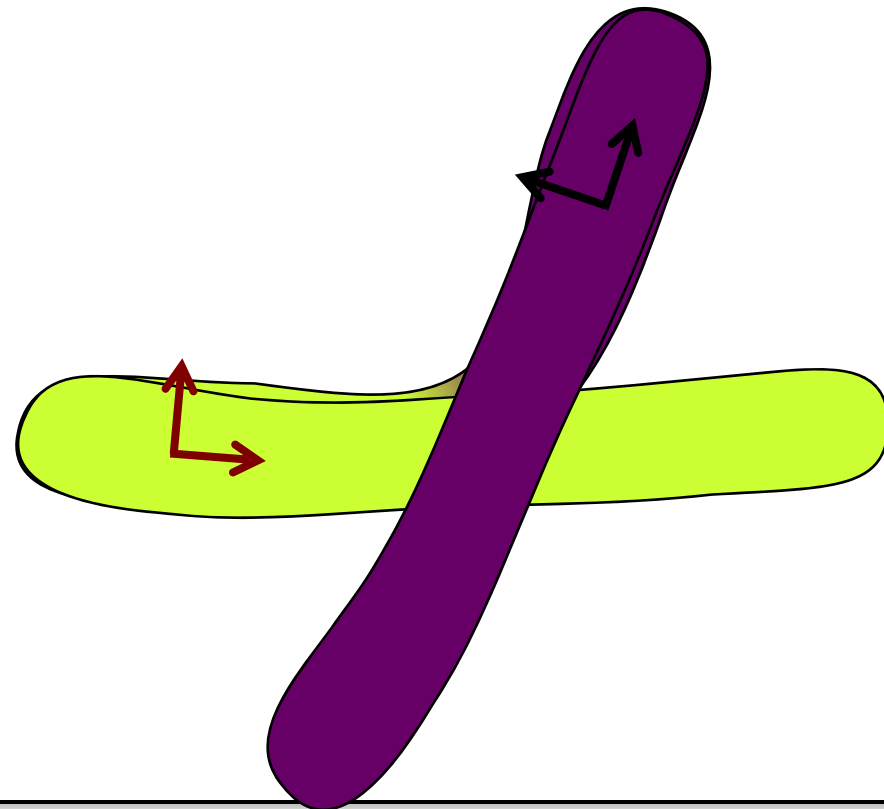
 Bone 1

 Bone 2



Shape with Weights

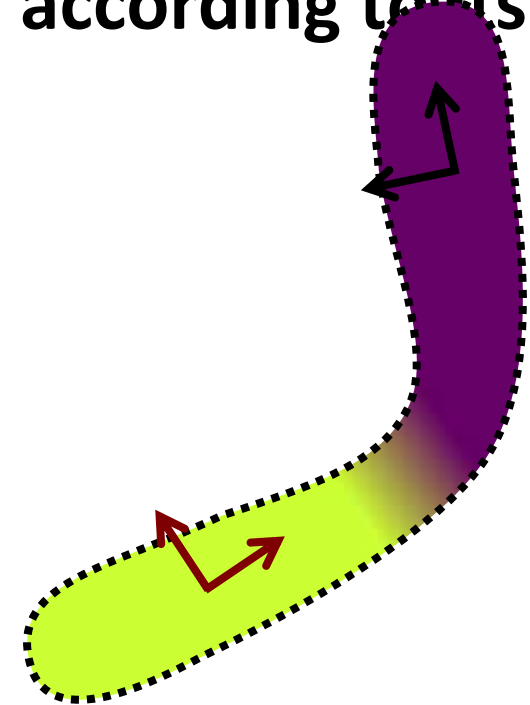
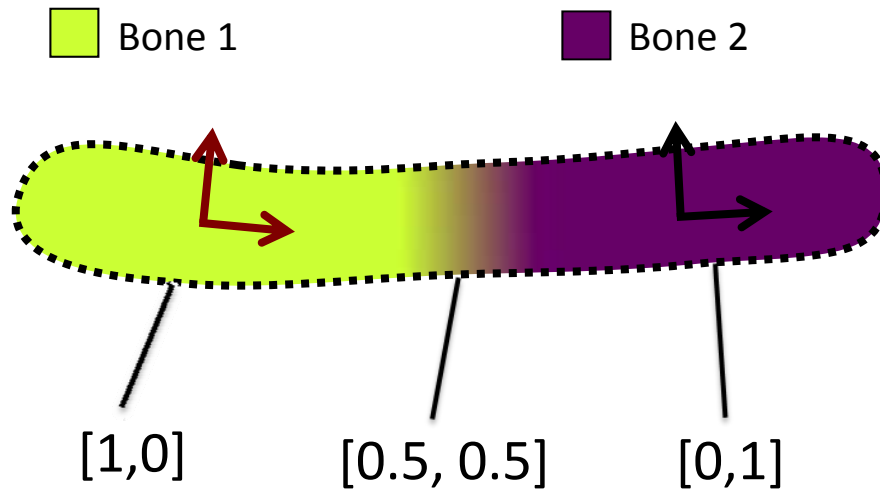
Weights from Bone 1
Weights from Bone 2



Model: Linear Blend Skinning

Each point assigned vector of weights

Transformations move each point according to its weights

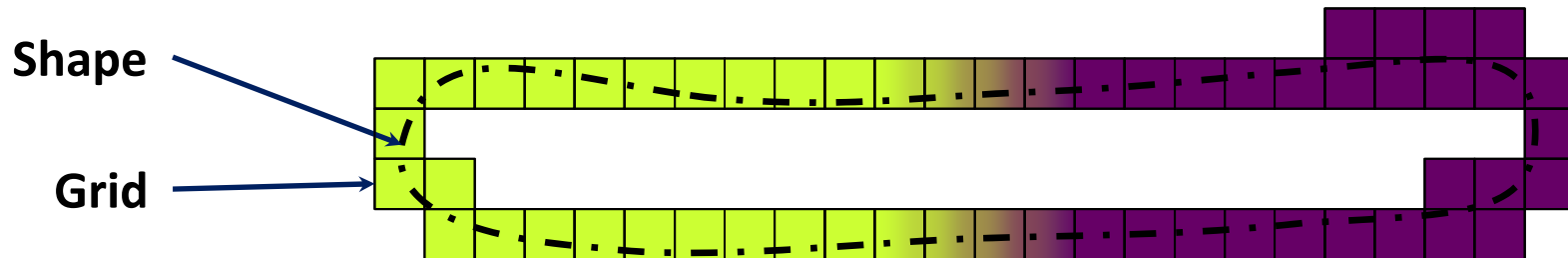


Weighted Blending Result

Weight Grid

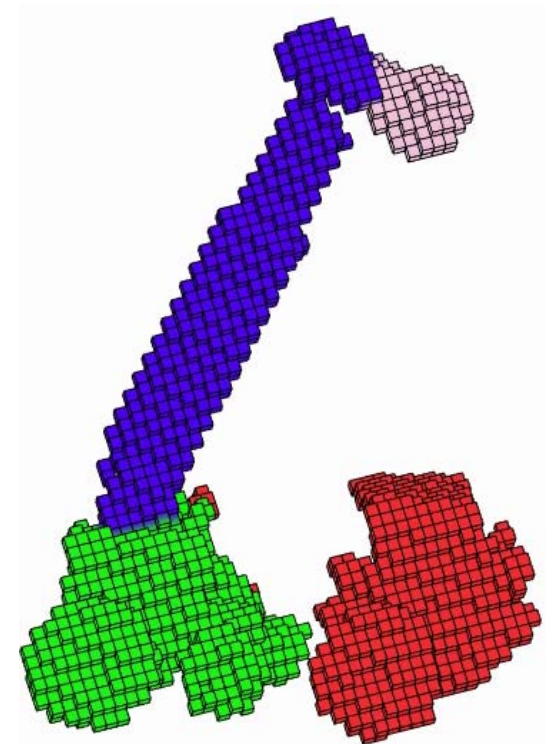
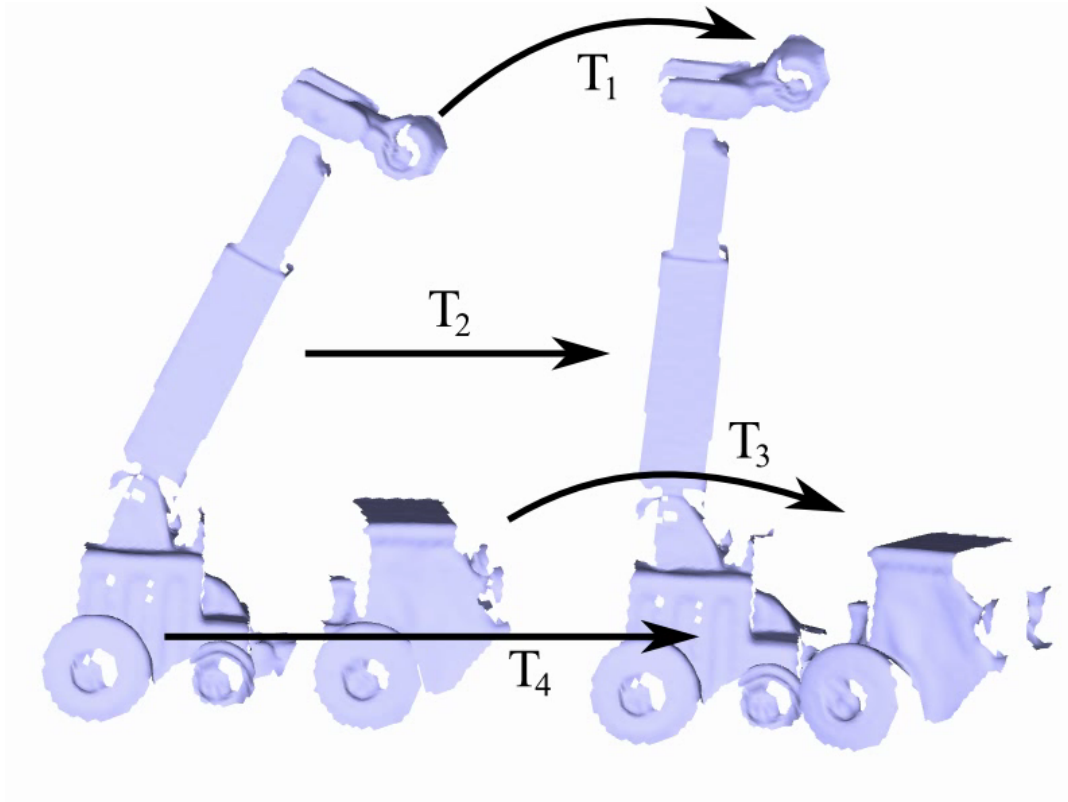
Define weights on grid enclosing surface

- Covers small holes, reduces variables
- Provides regular structure for optimization

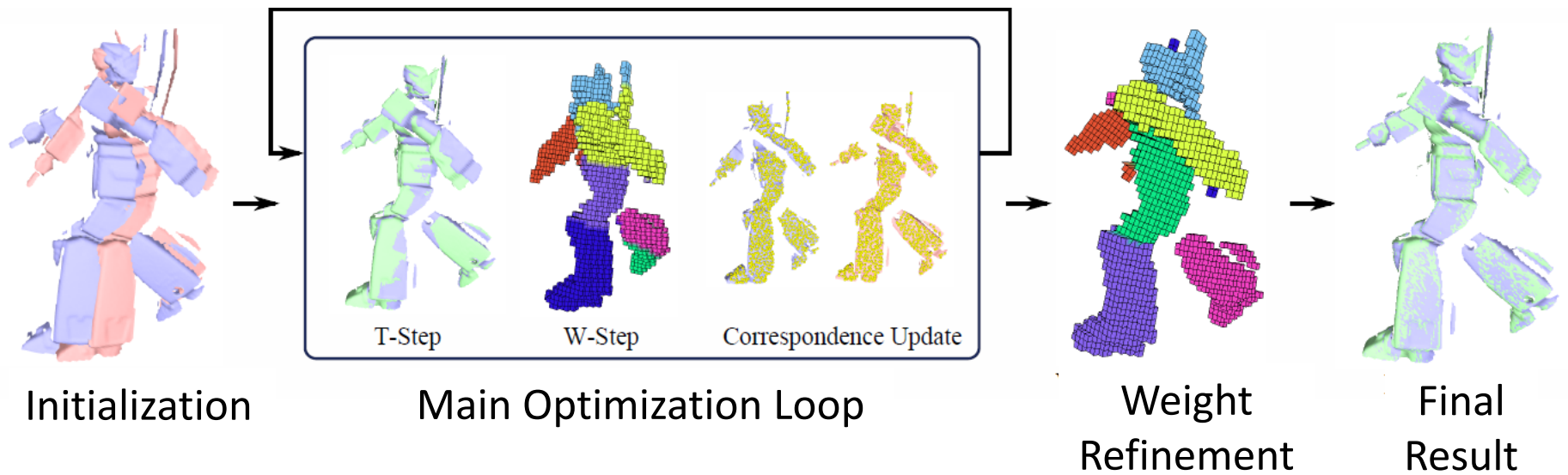


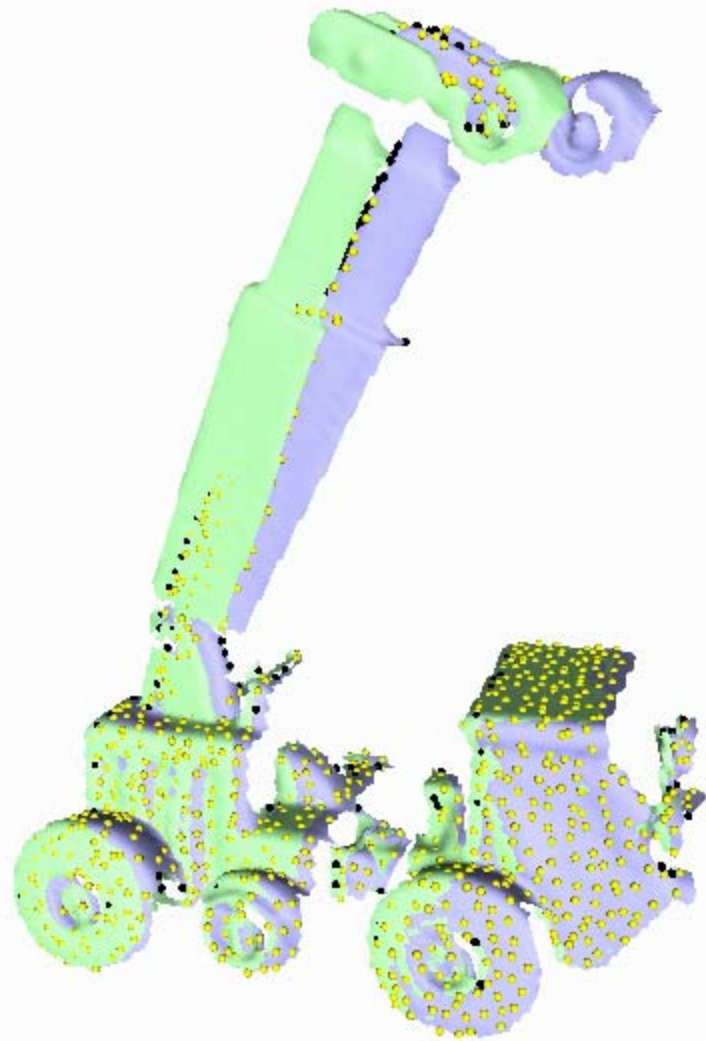
LBS for scan registration

Fit the transformations and weights to align a pair of range scans

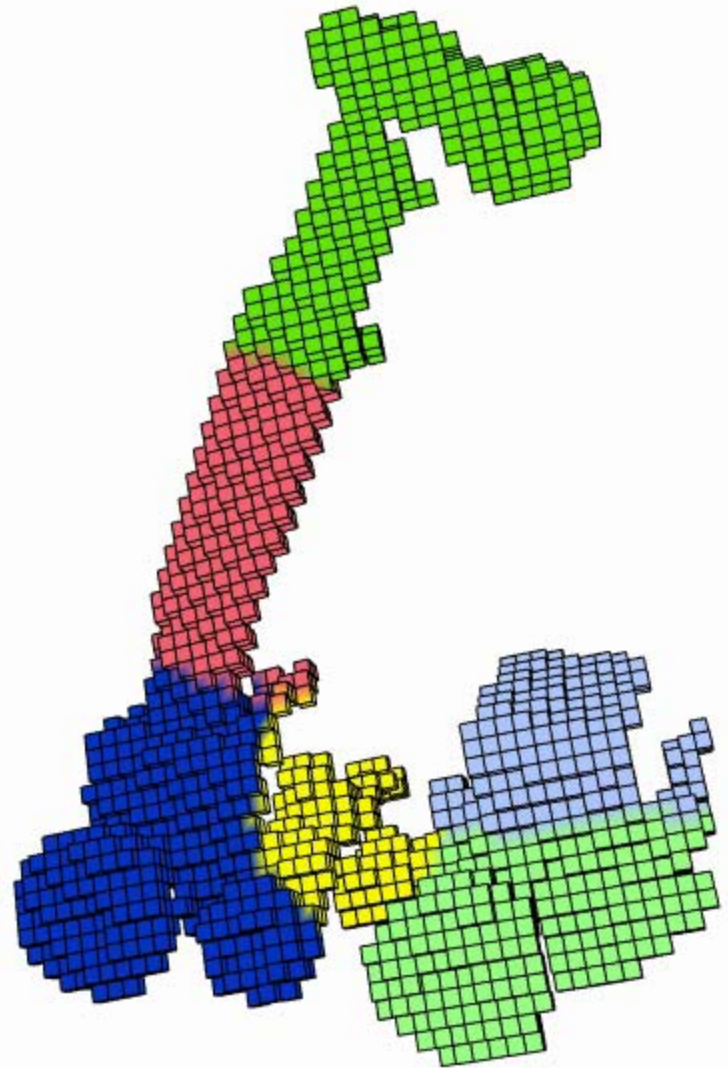


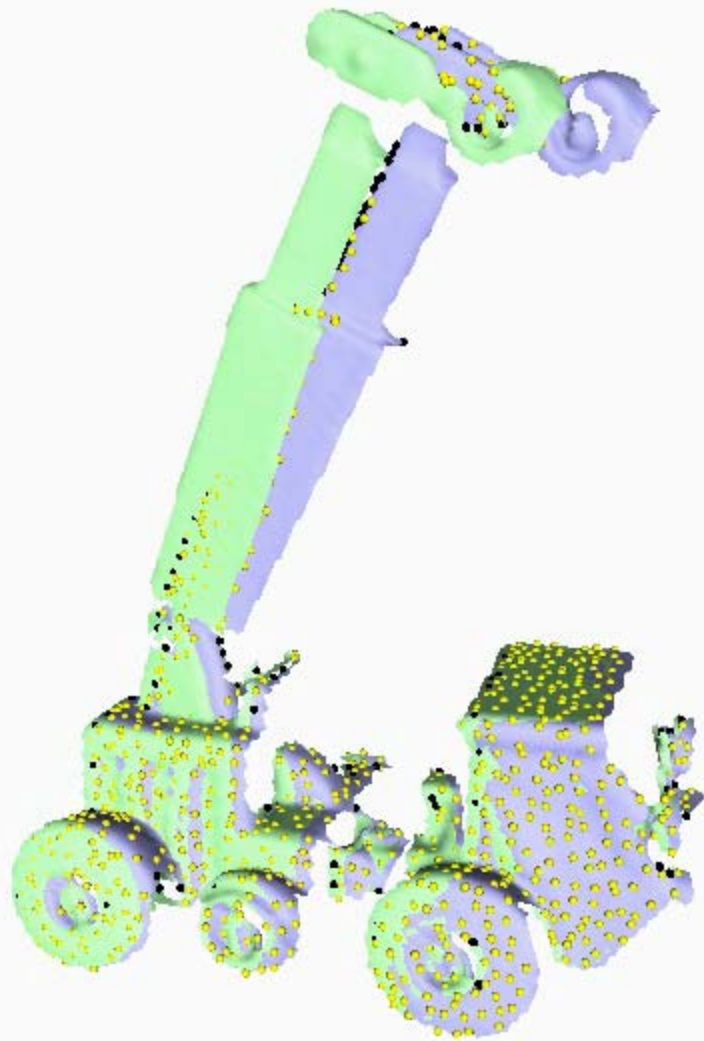
Algorithm Description



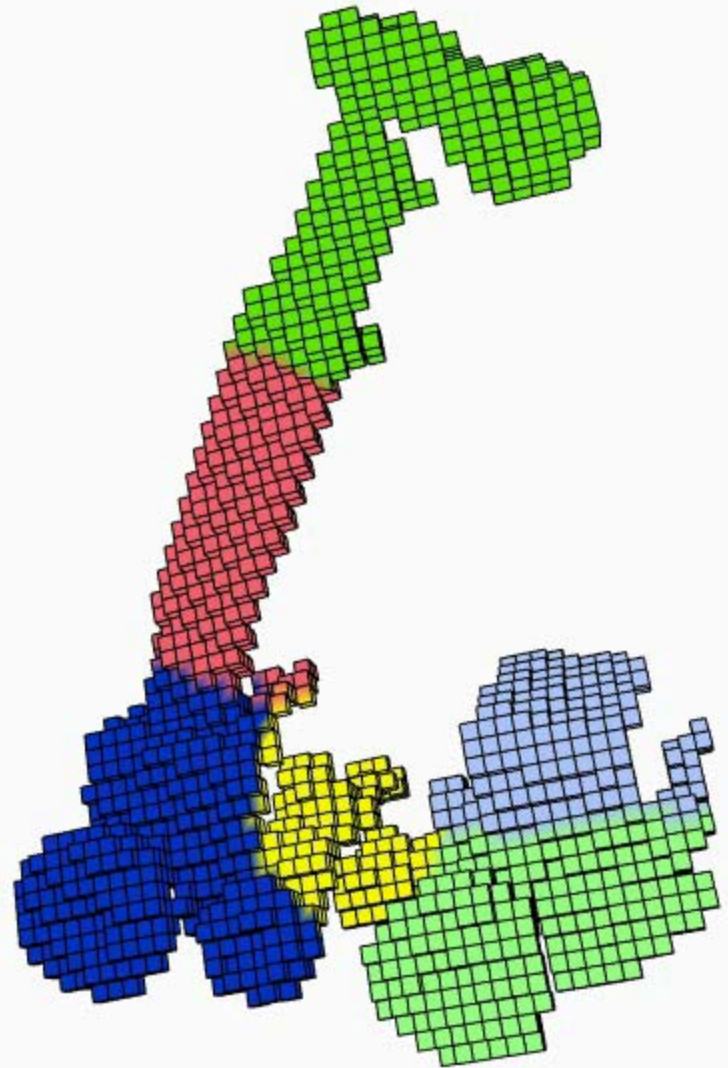


Initialization

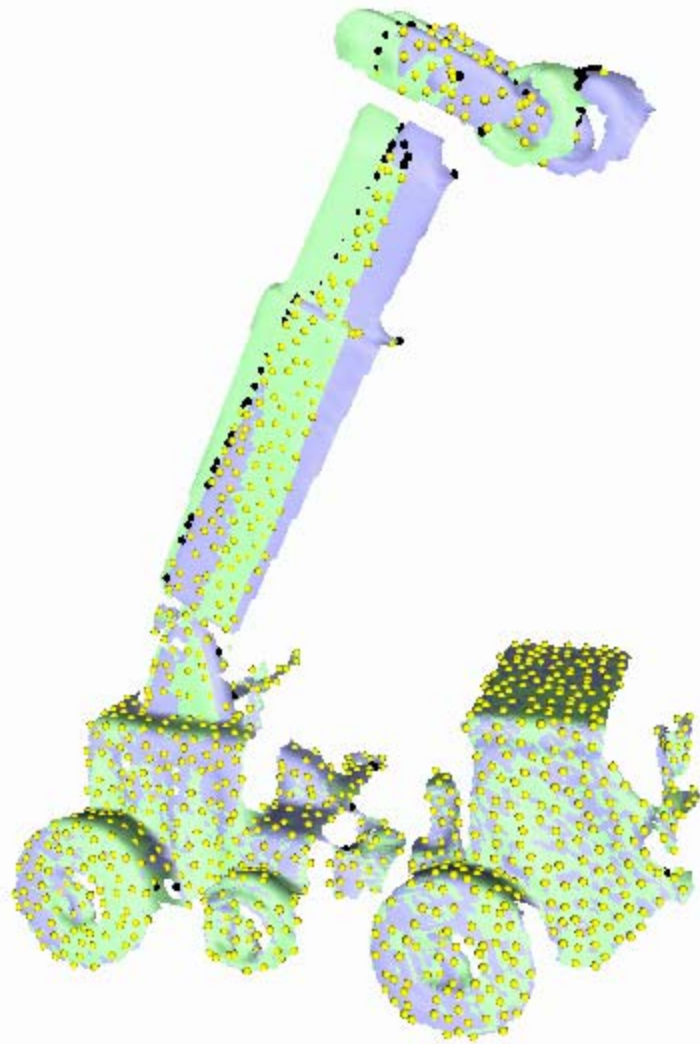




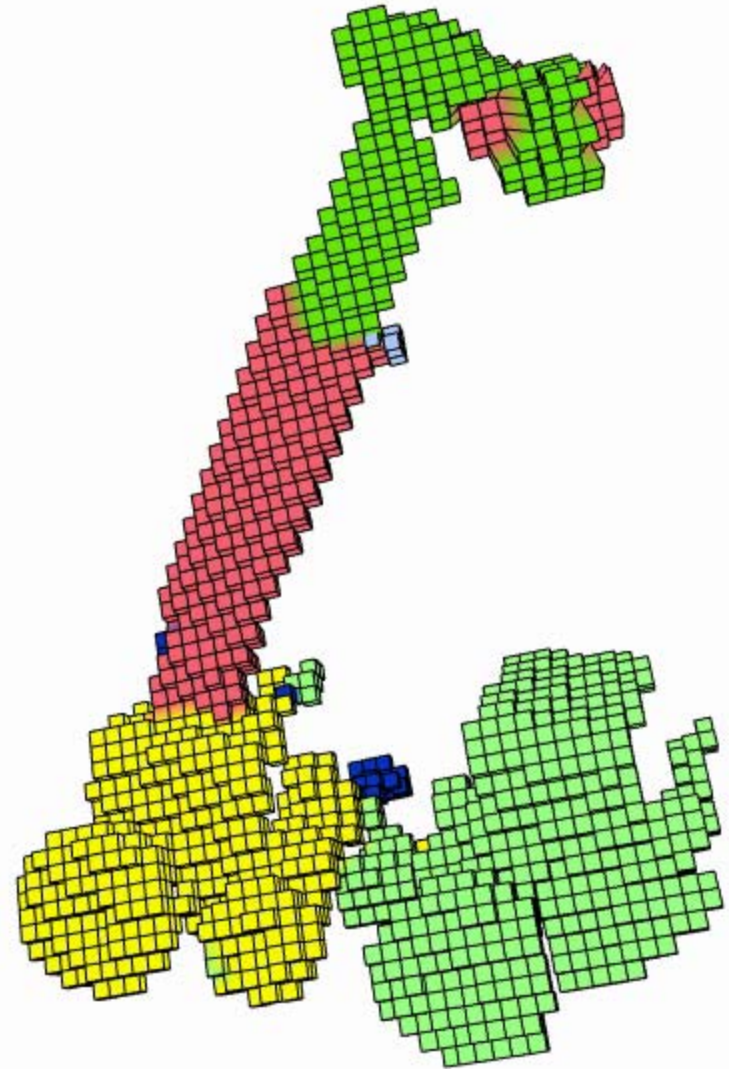
Initialization

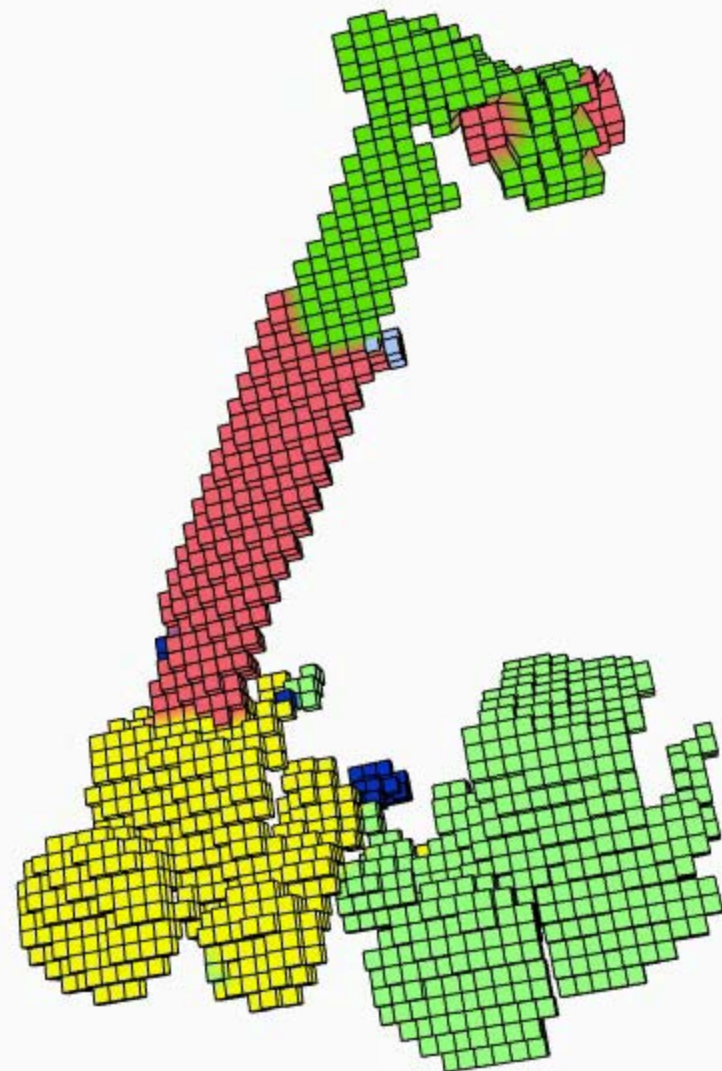
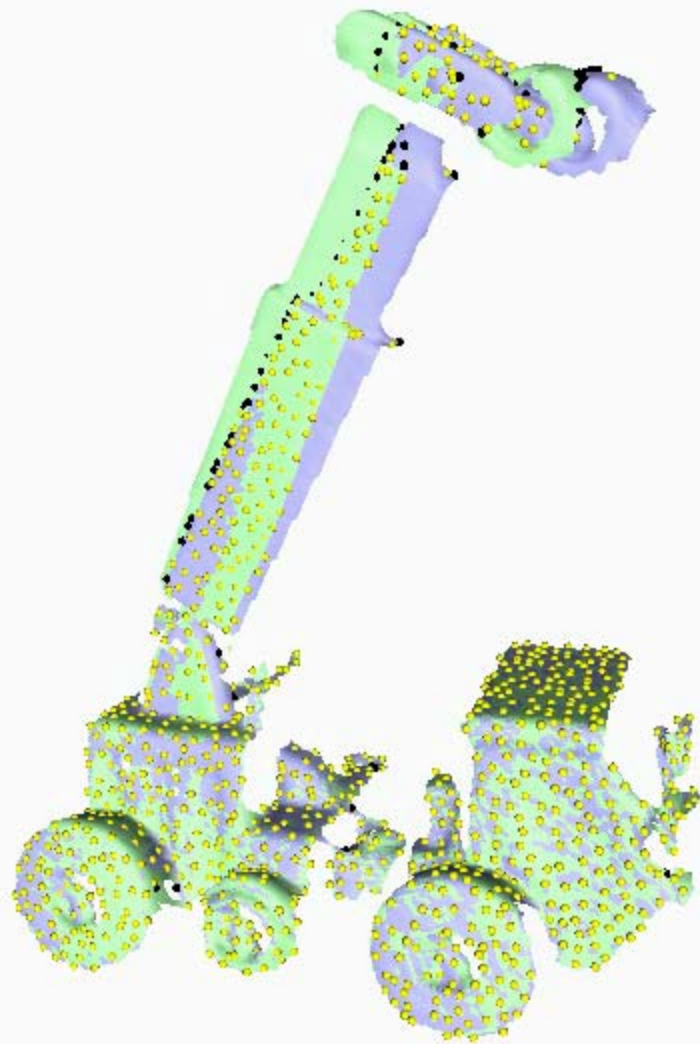


(Converged)



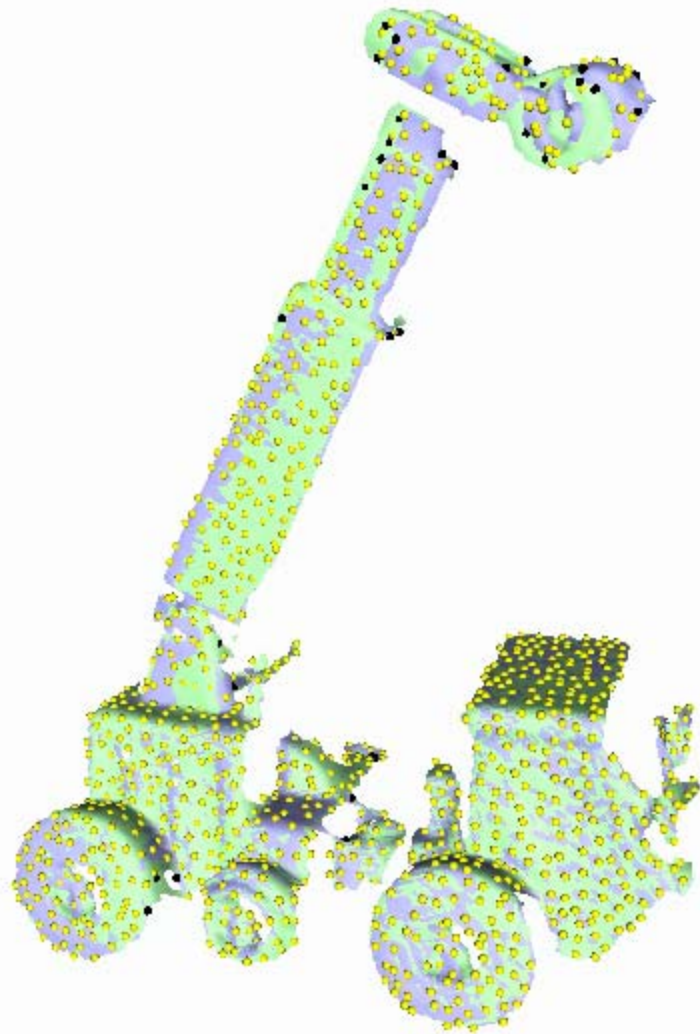
W-Step



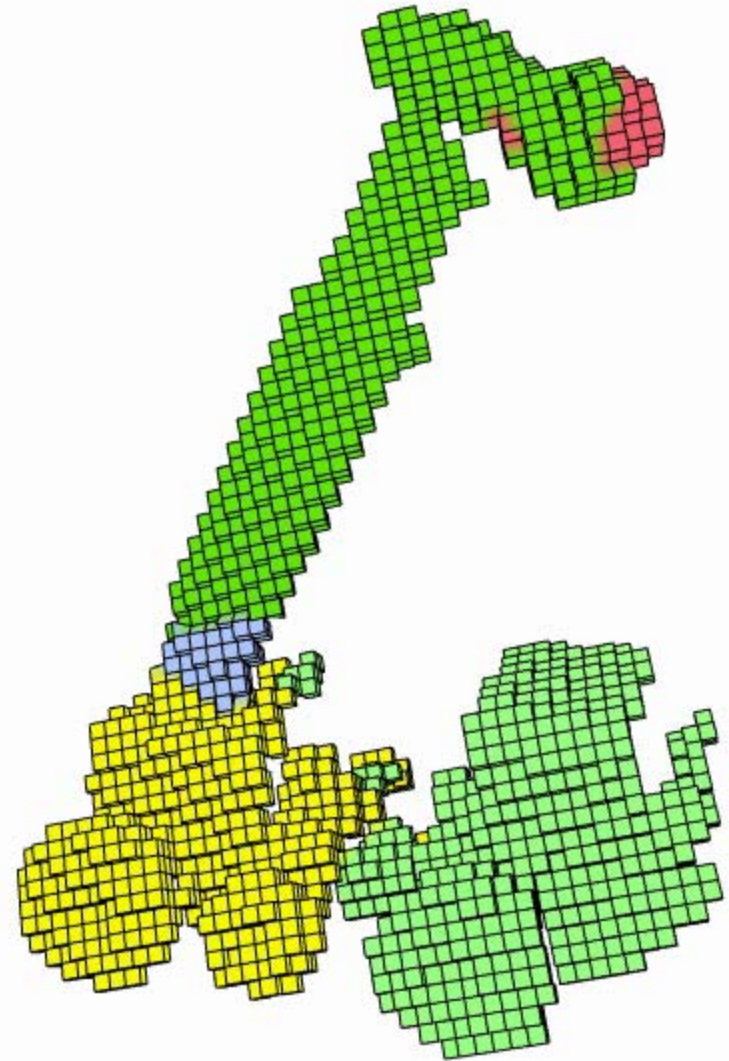


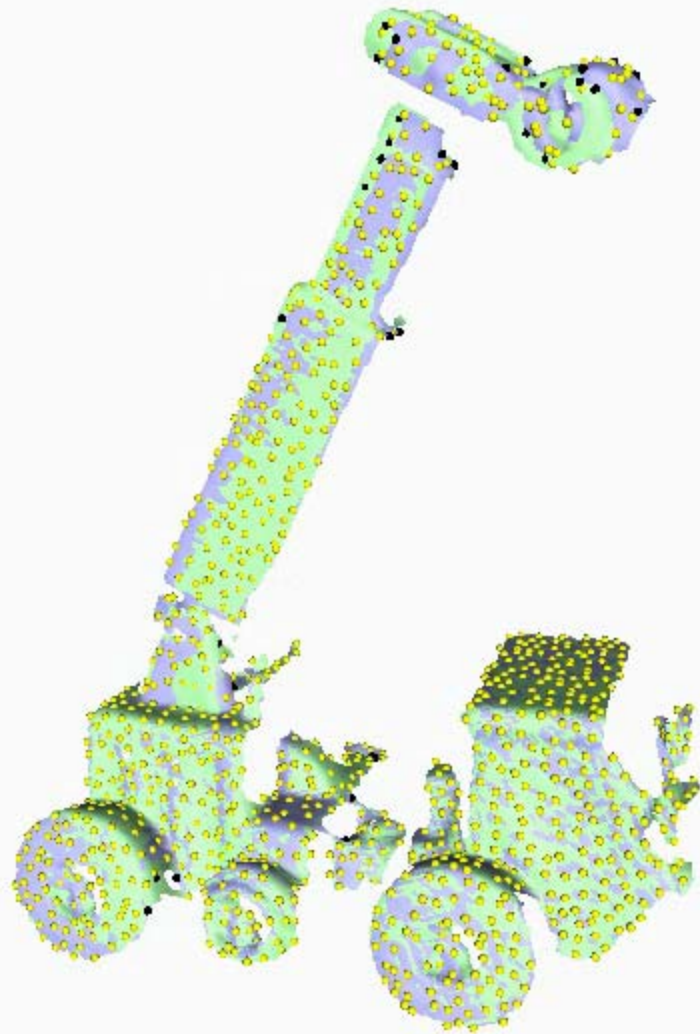
T-Step

(Converged)



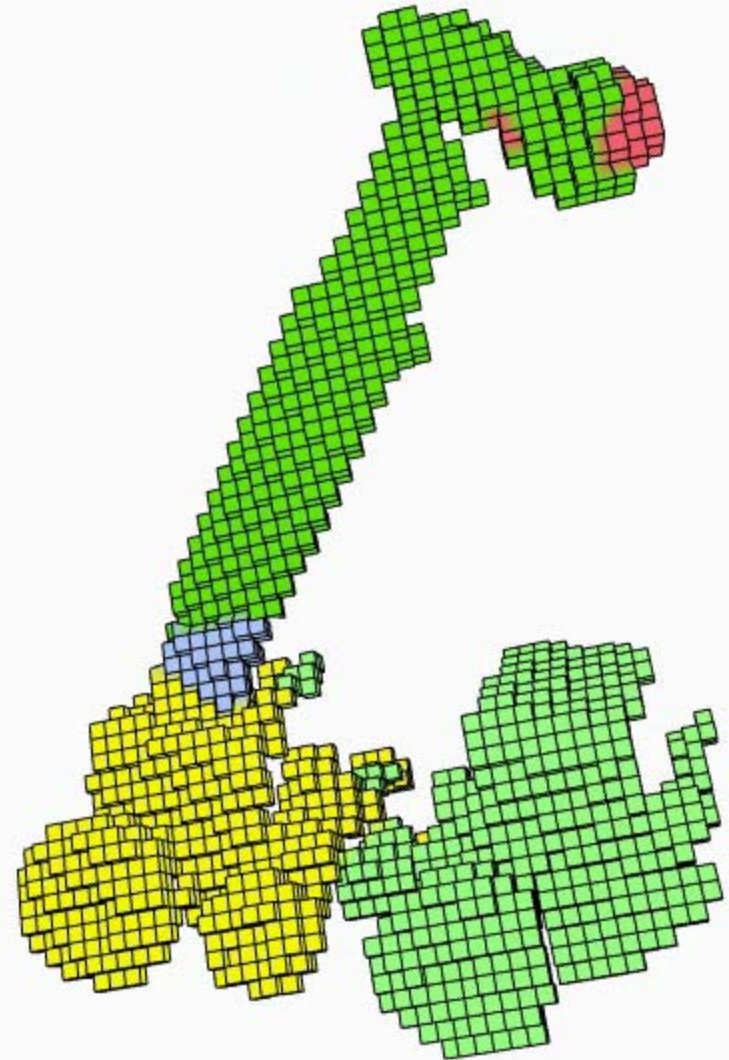
W-Step



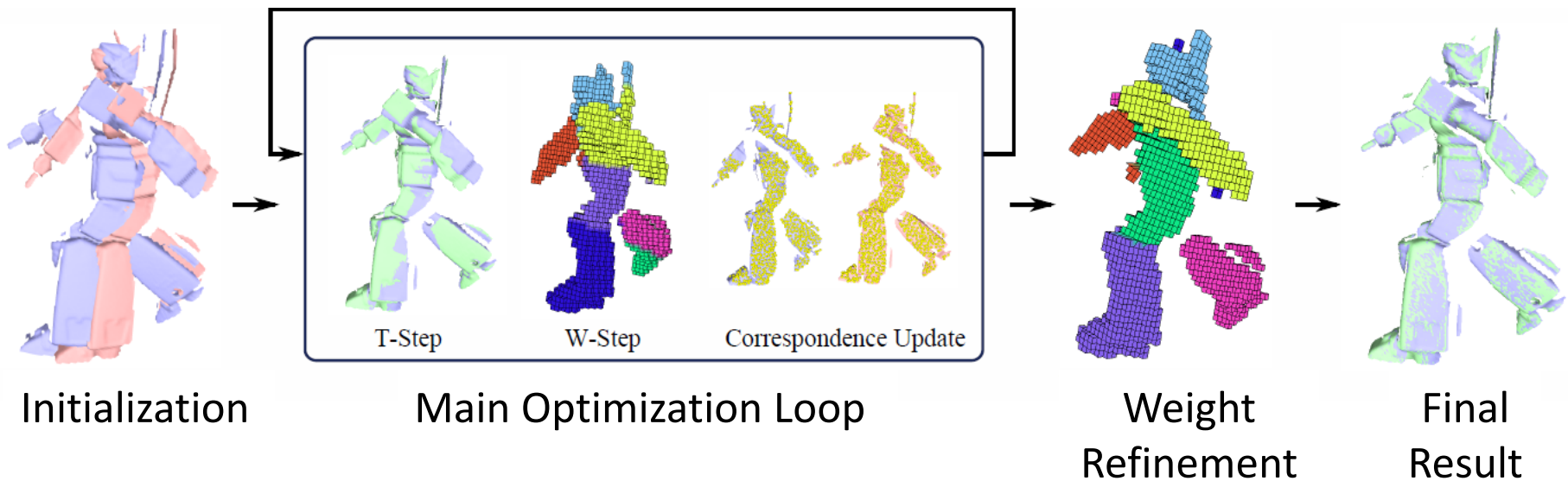


W-Step

(Finished)

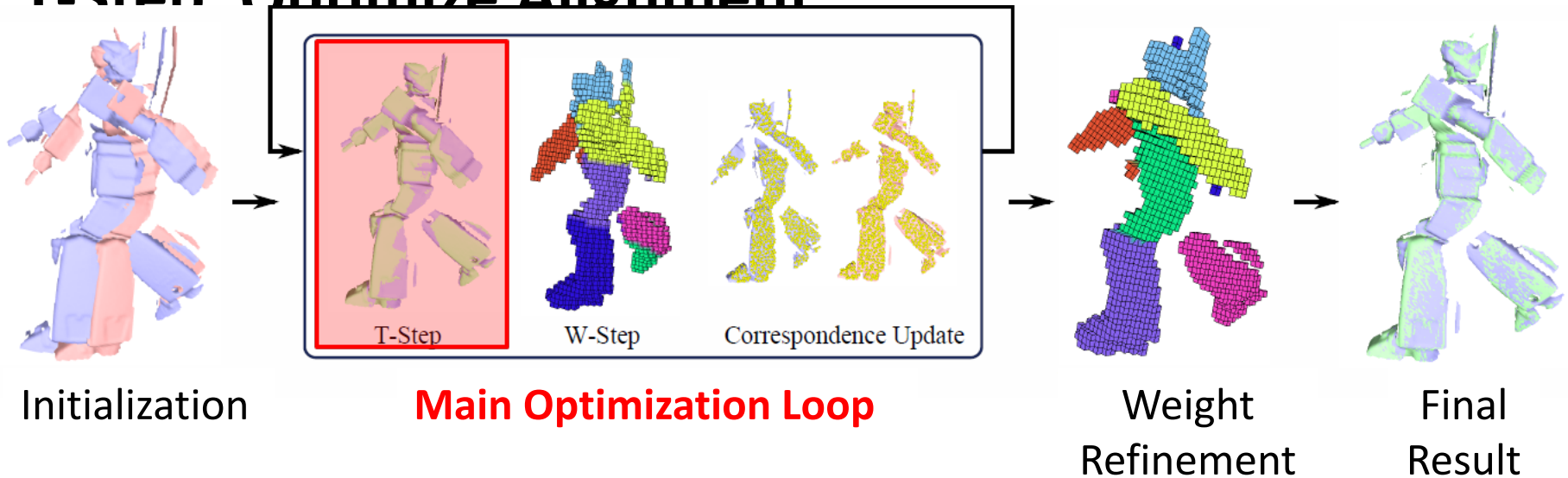


Optimization overview



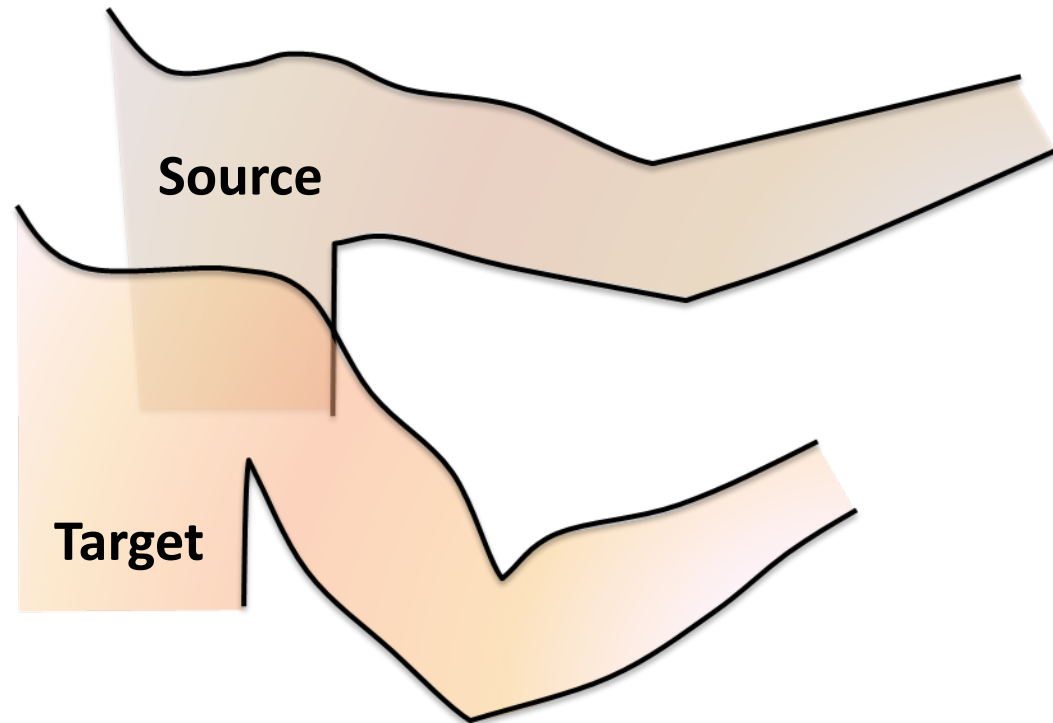
Optimization overview

T-Step: Optimize Alignment



T-Step: Distance Term

Fix weights & solve for transformations

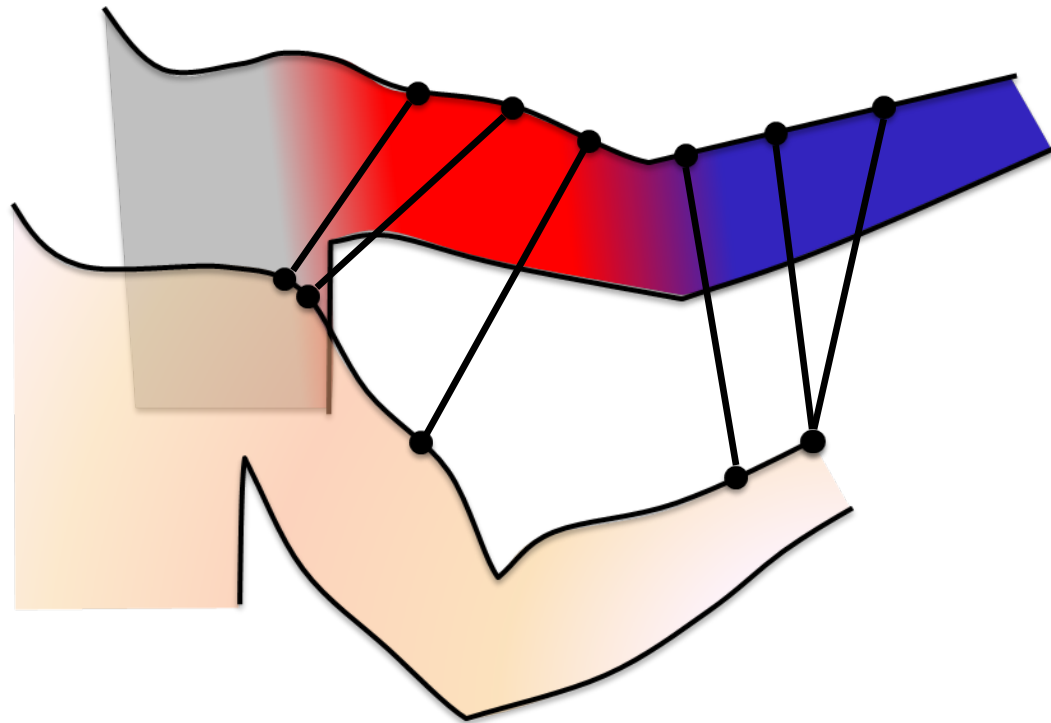


T-Step: Distance Term

Fix weights & solve for transformations

- Use closest point correspondences

- Bone 1
- Bone 2
- Bone 3

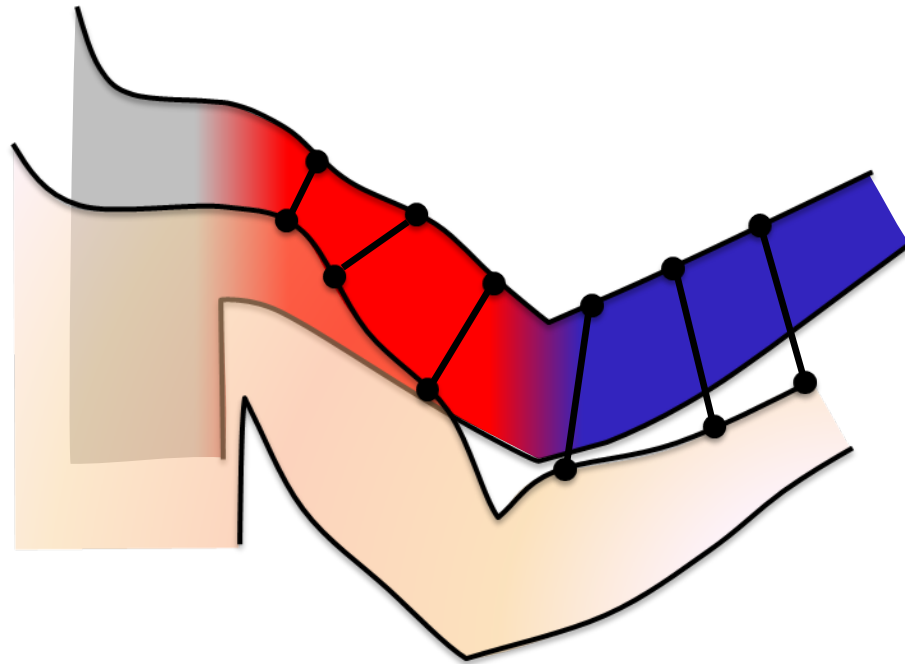


T-Step: Distance Term

Fix weights & solve for transformations

- Use closest point correspondences

- Bone 1
- Bone 2
- Bone 3

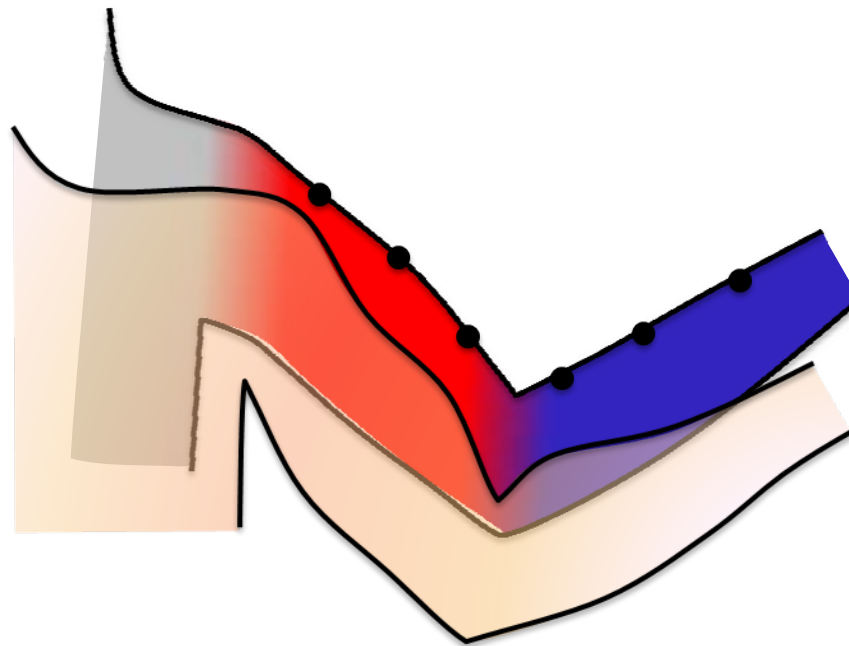


T-Step: Distance Term

Fix weights & solve for transformations

- Use closest point correspondences
- Iterate further until convergence

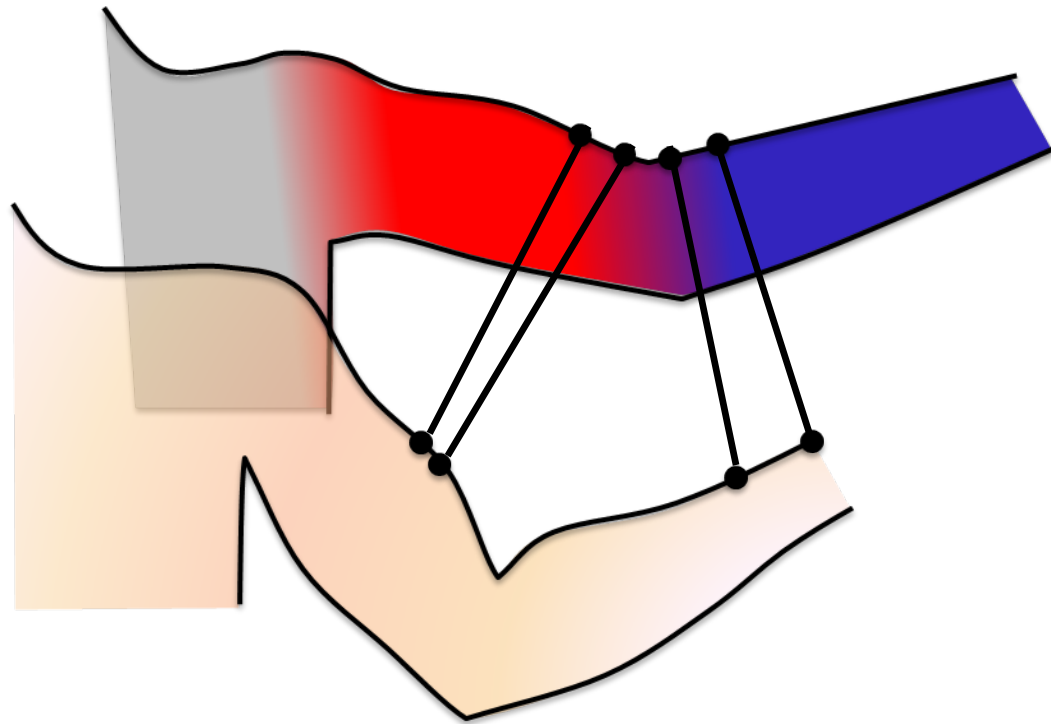
- Bone 1
- Bone 2
- Bone 3



T-Step: Joint Constraint Term

Prevent neighboring bones from separating

- Bone 1
- Bone 2
- Bone 3

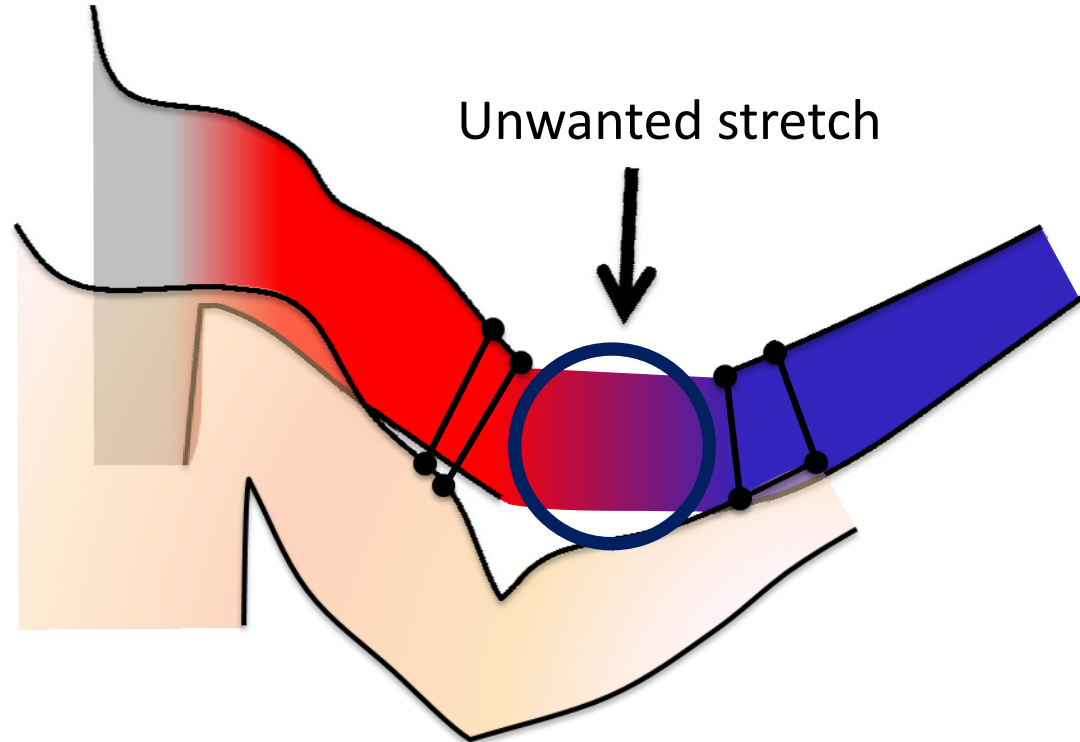


T-Step: Joint Constraint Term

Prevent neighboring bones from separating

- Constrain overlapping weight regions

- Bone 1
- Bone 2
- Bone 3

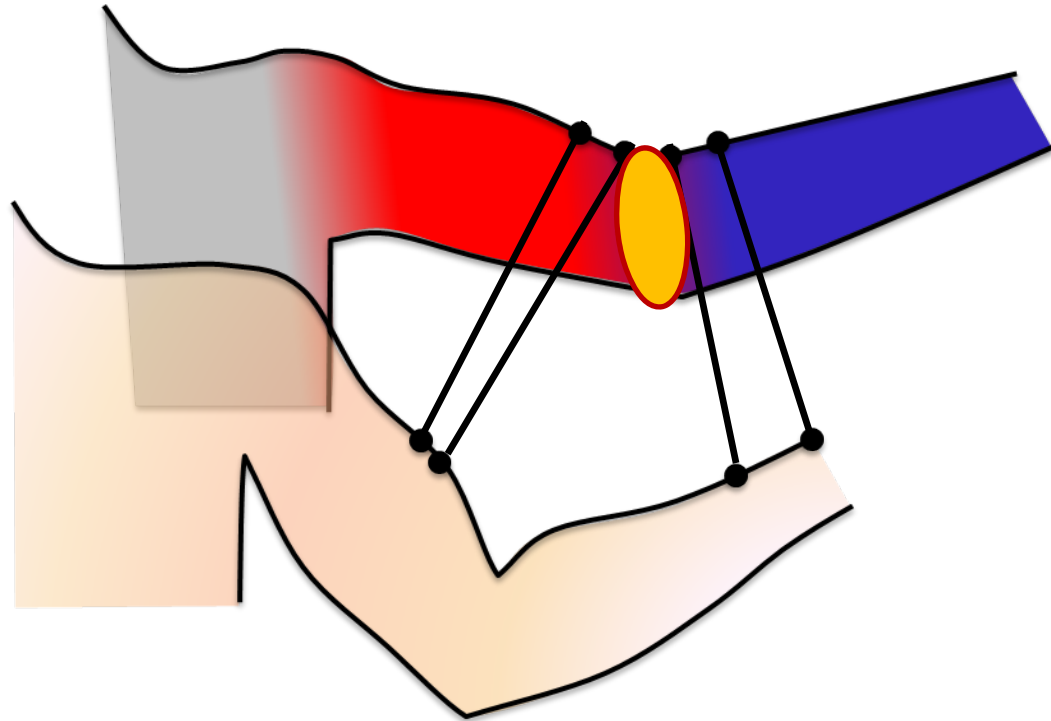


T-Step: Joint Constraint Term

Prevent neighboring bones from separating

- Constrain overlapping weight regions

- Bone 1
- Bone 2
- Bone 3

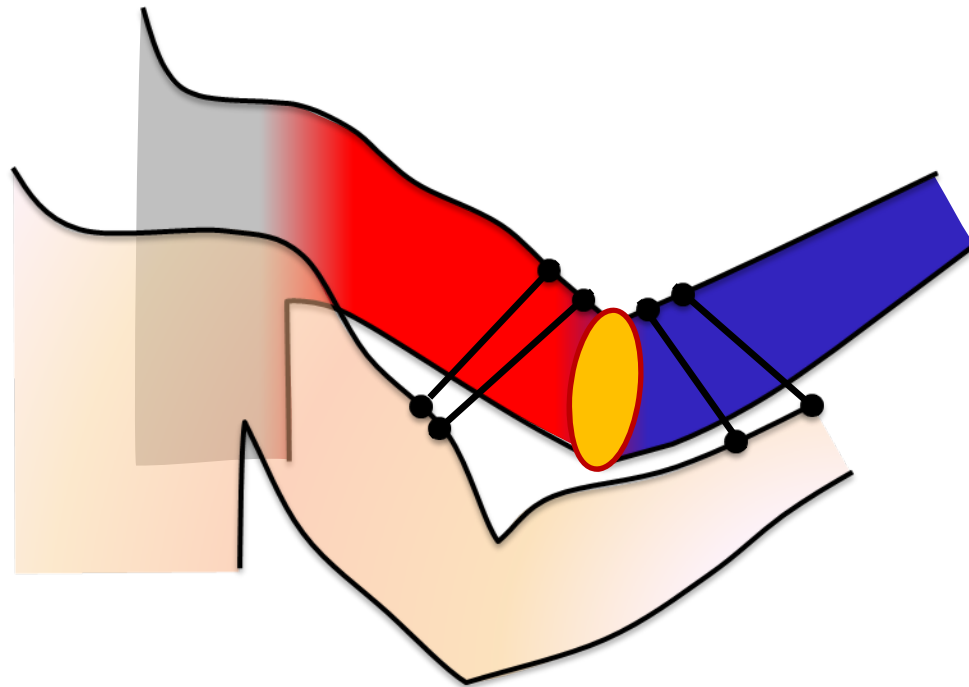


T-Step: Joint Constraint Term

Prevent neighboring bones from separating

- Constrain overlapping weight regions

- Bone 1
- Bone 2
- Bone 3



T-Step: Optimization summary

Like rigid registration

- Except multiple parts & joint constraints

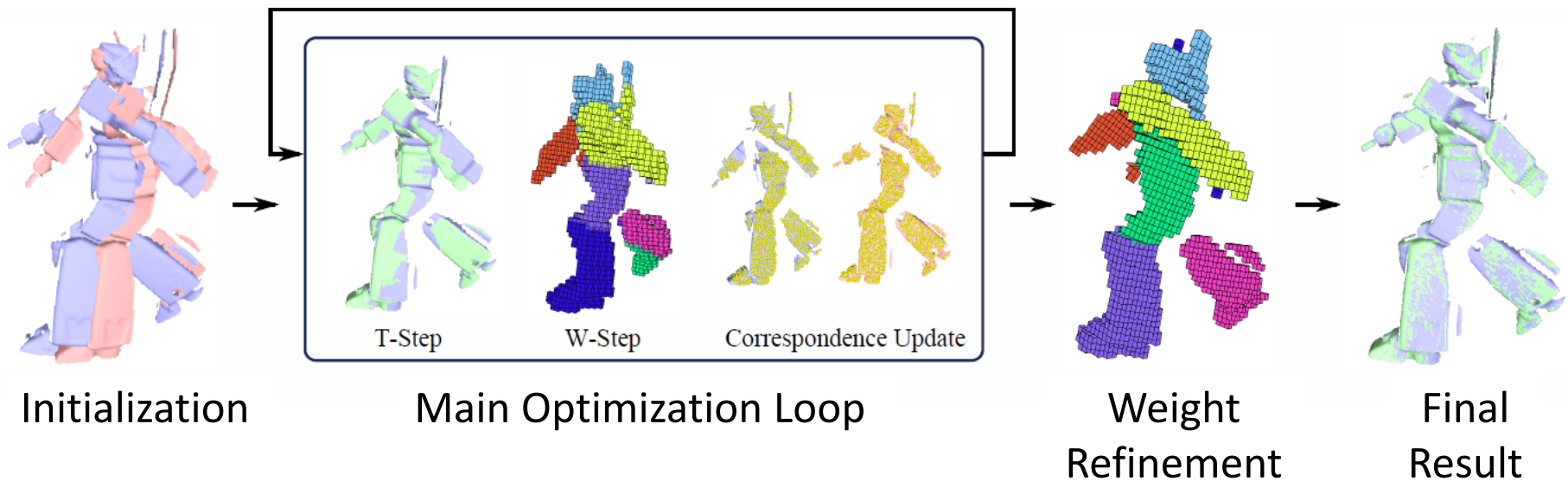
Non-linear least squares optimization

- Solving for a rotation matrix
- Gauss-Newton algorithm
- Solve by iteratively linearizing solution

Few variables → Fast performance

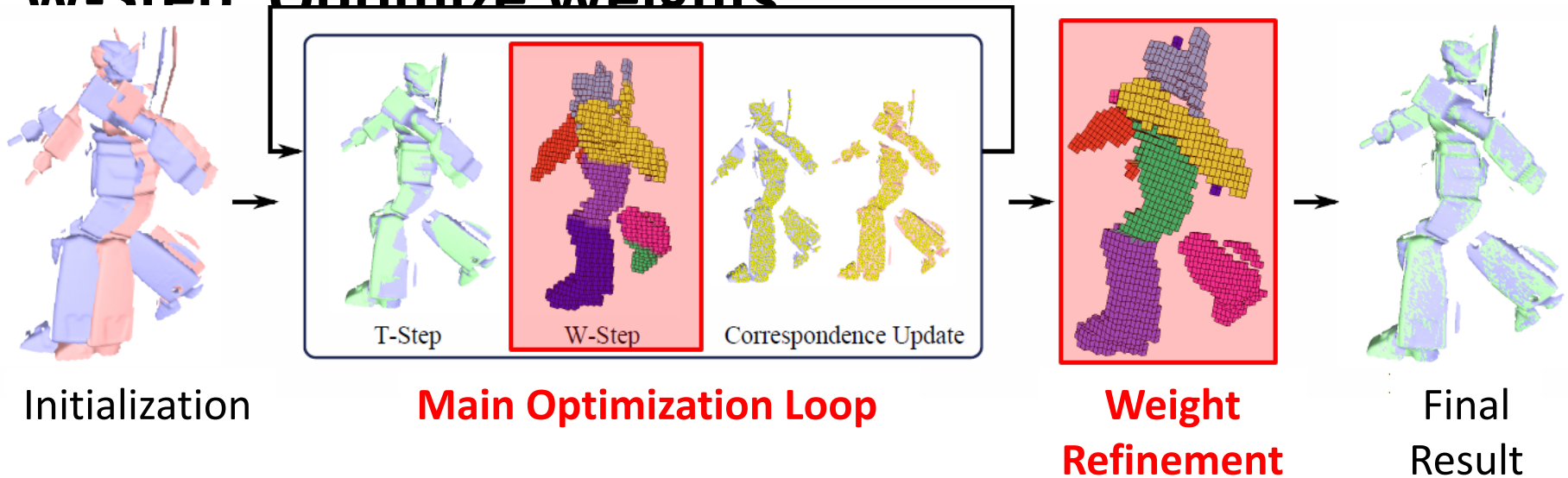
- # variables = 6 x #bones
- Typically 5~10 bones in our examples

Optimization overview



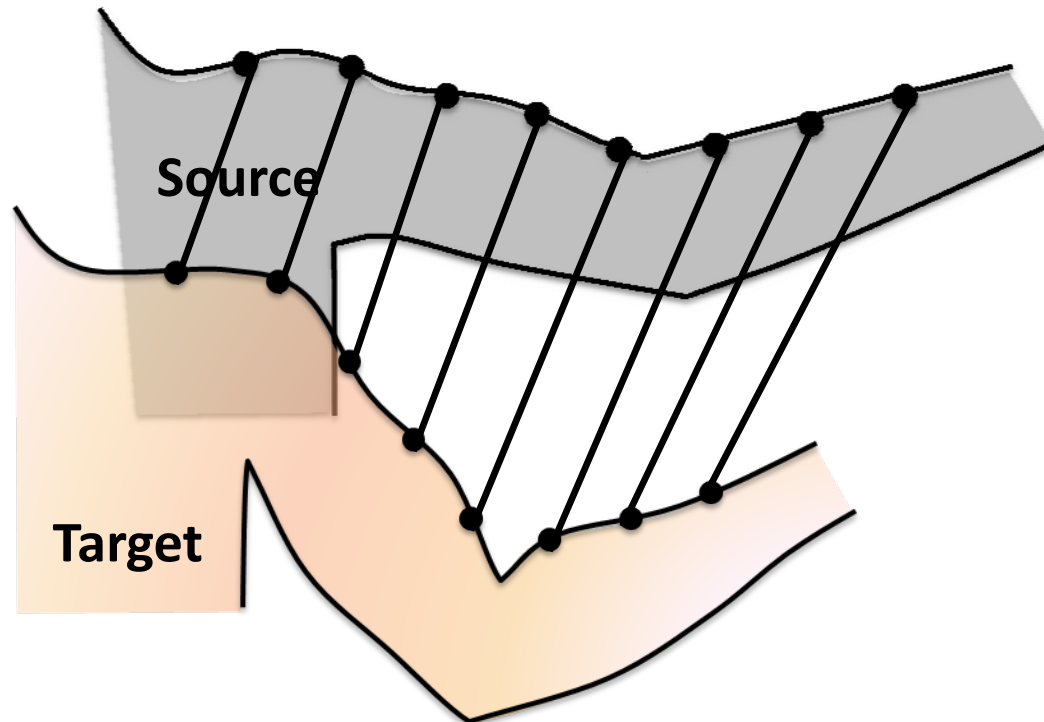
Optimization overview

W-Step: Optimize Weights



W-Step: Optimizing weights

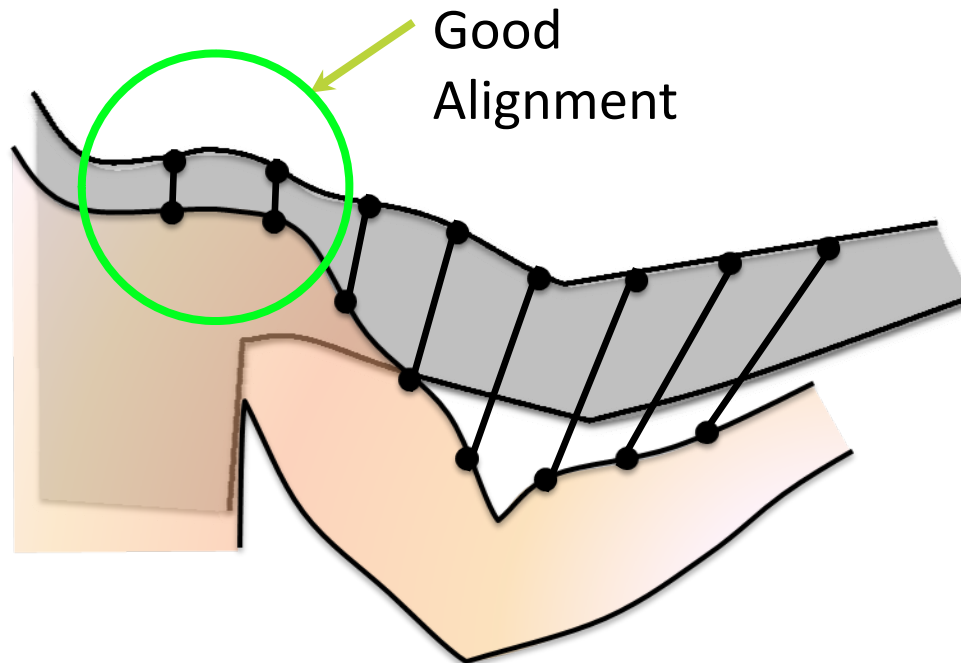
Fix transformations, solve for weights



Correspondences from last T-Step

W-Step: Optimizing weights

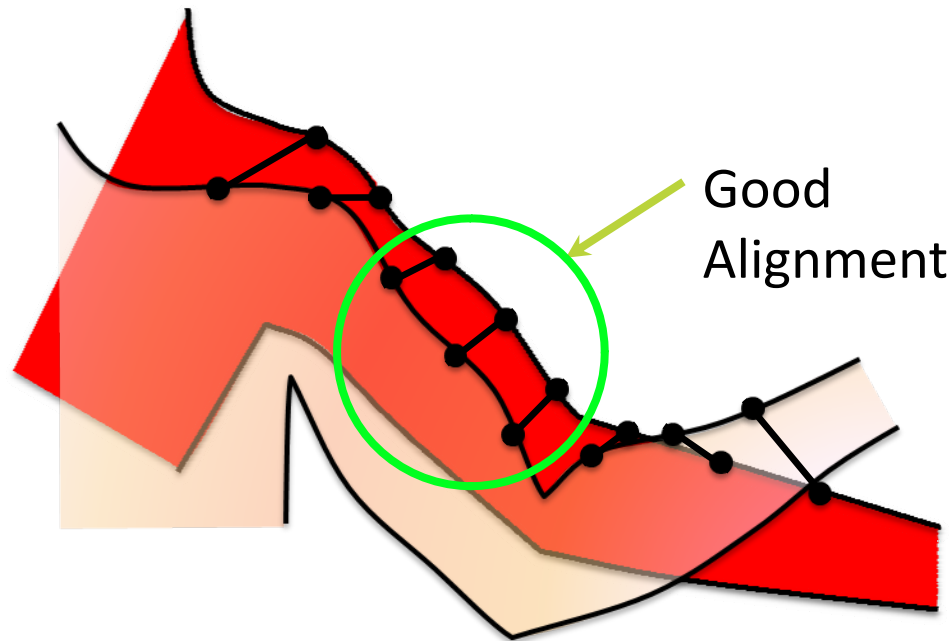
Fix transformations, solve for continuous weights



Bone 1
(Applied to entire shape)

W-Step: Optimizing weights

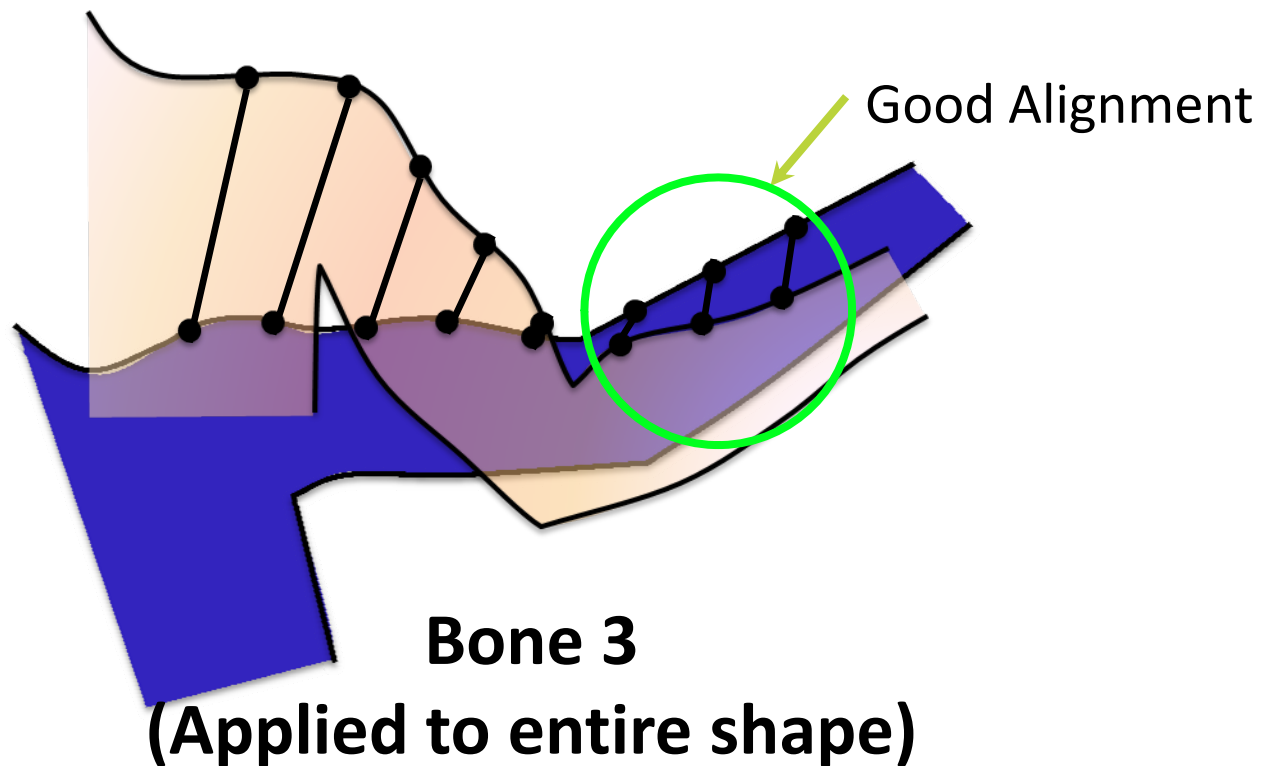
Fix transformations, solve for continuous weights



Bone 2
(Applied to entire shape)

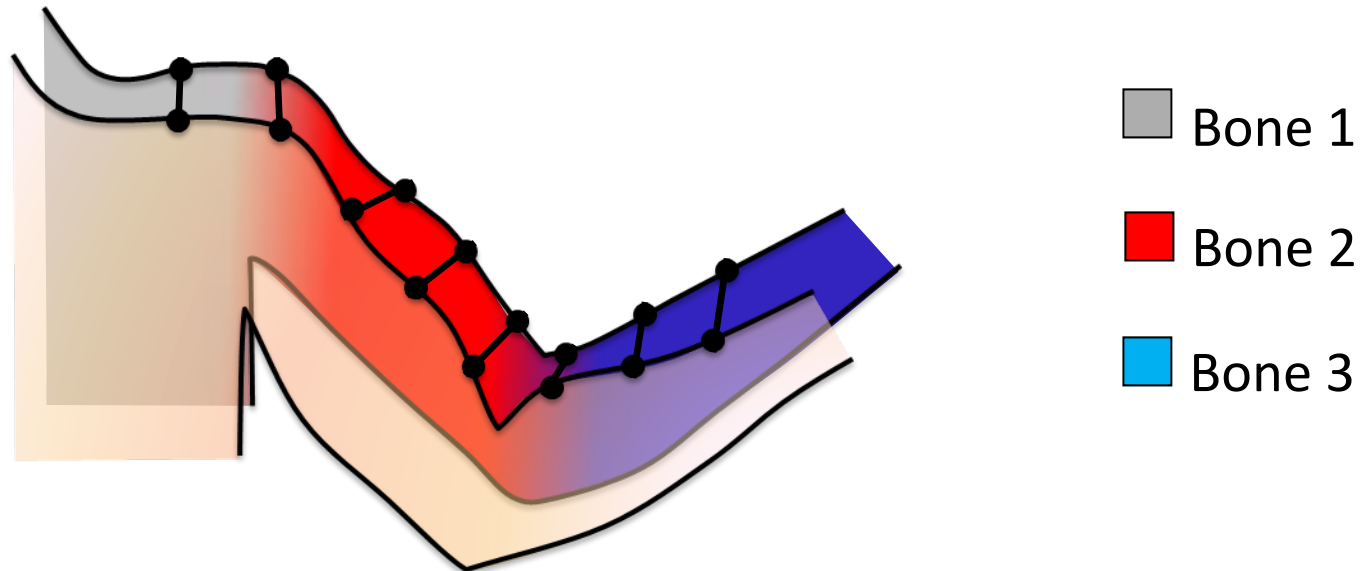
W-Step: Optimizing weights

Fix transformations, solve for continuous weights



W-Step: Optimizing weights

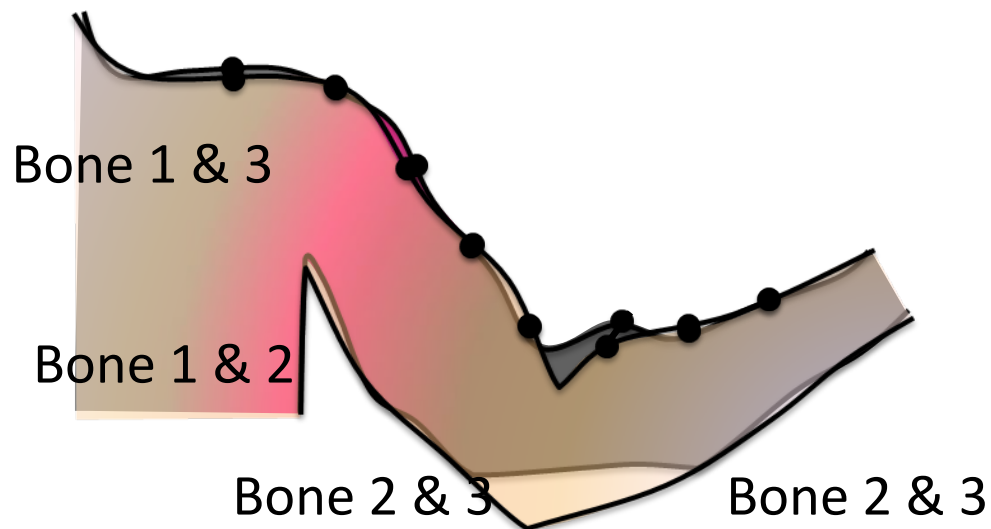
Fix transformations, solve for continuous weights



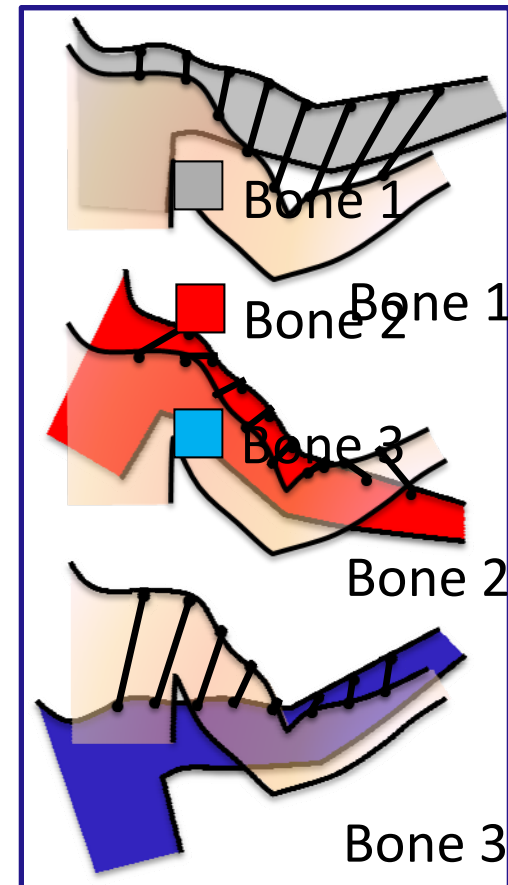
“Ideal” solved result

W-Step: Optimizing weights

Without additional constraints, problem is underconstrained



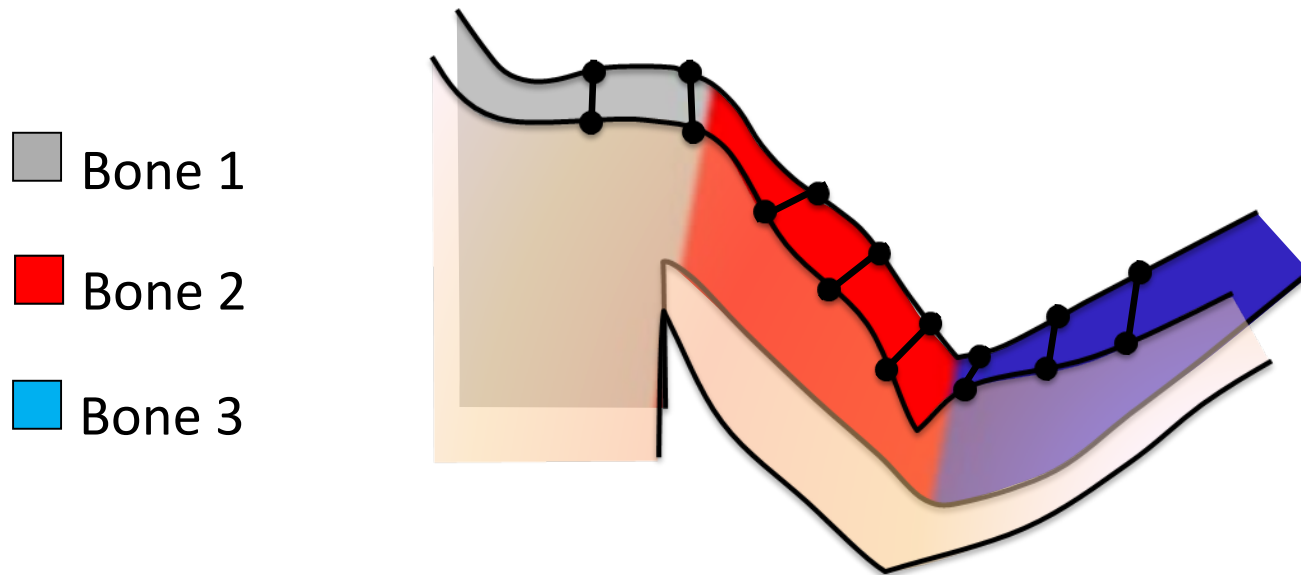
Typical solved result



Use discrete labeling

Our solution: one transformation per location

- Transformations = labels
- Becomes discrete labeling problem



Closer to “Ideal”
solved result!

W-Step: Optimization Summary

Use “graph cuts” to optimally label grid cells

- [Boykov, Veksler & Zabih PAMI '01]

Distance term + Smoothness term

- Distance: measures alignment for a given label
- Smoothness: penalizes different labels for adjacent cells

Good Performance

- Only 1000~5000 grid cells (graph nodes) & 5~10 labels
- Fast performance for graph cuts

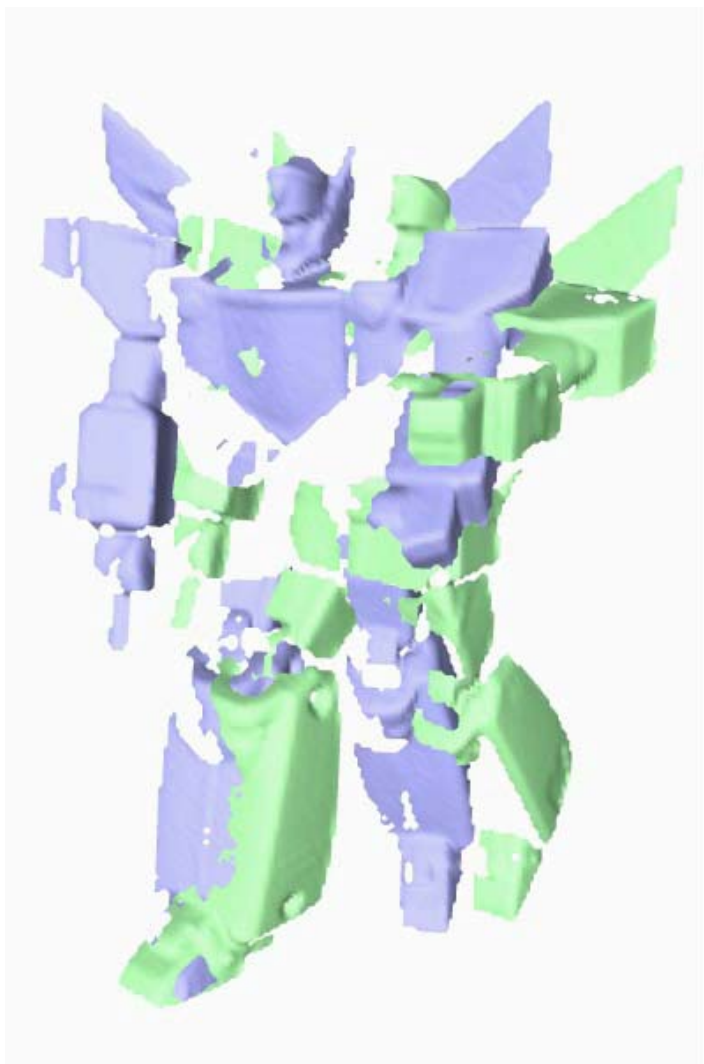
Results

Robot, torso video

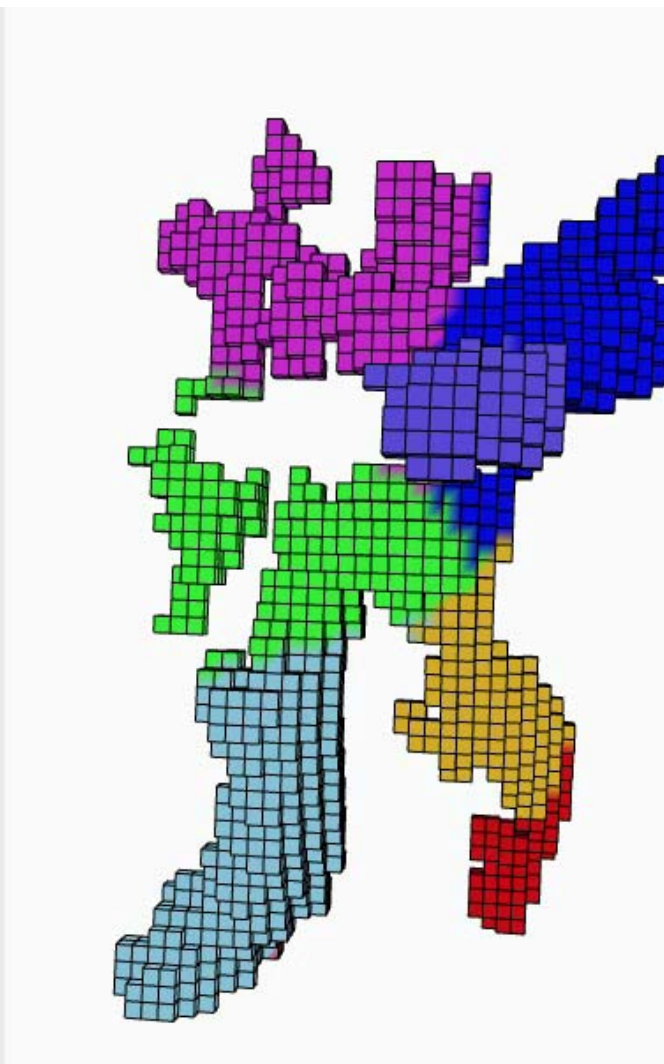
Interactive posing video

Additional results & statistics

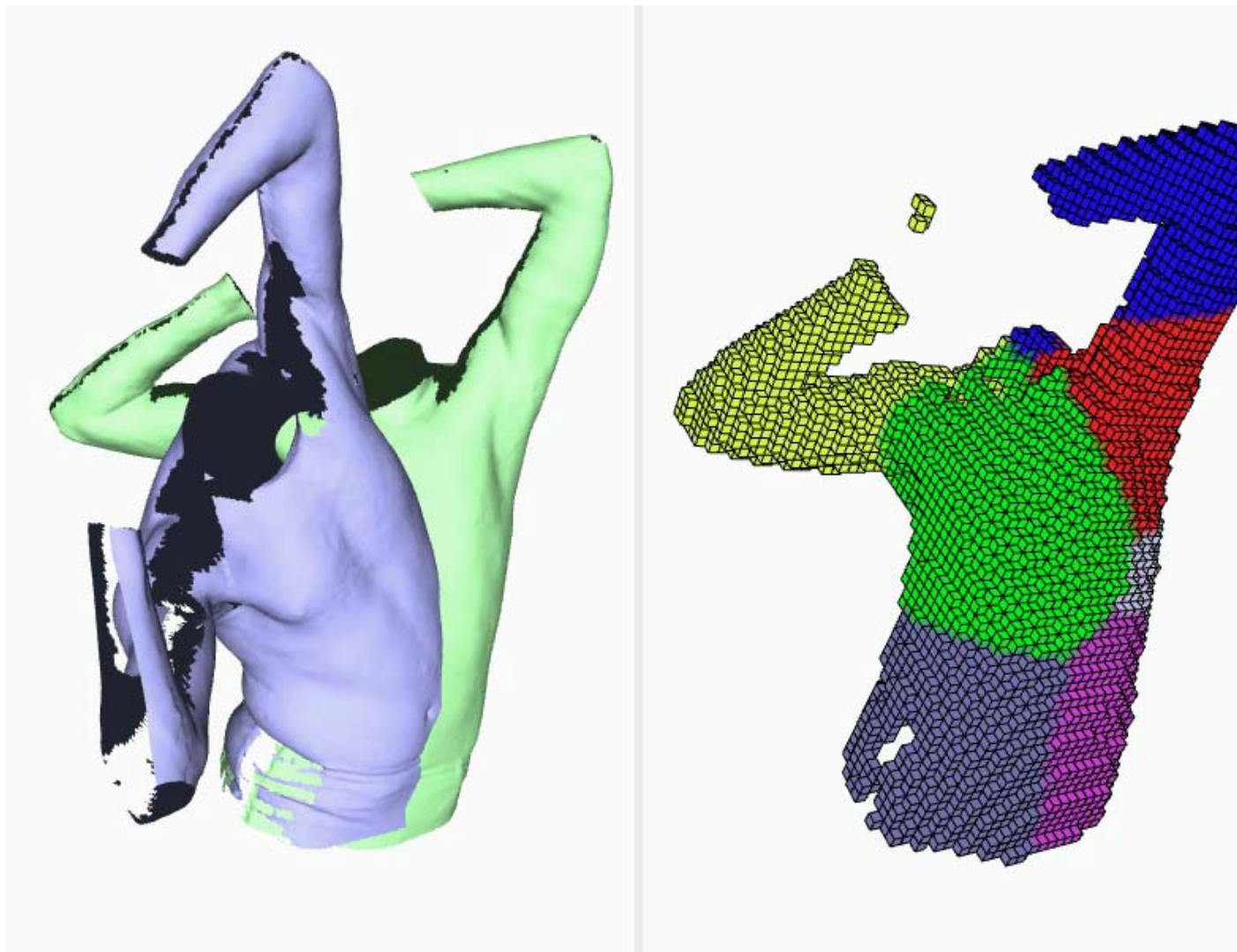
Robot video (real-time recording)



7 bones
1454 cells



Torso video (2x speed recording)

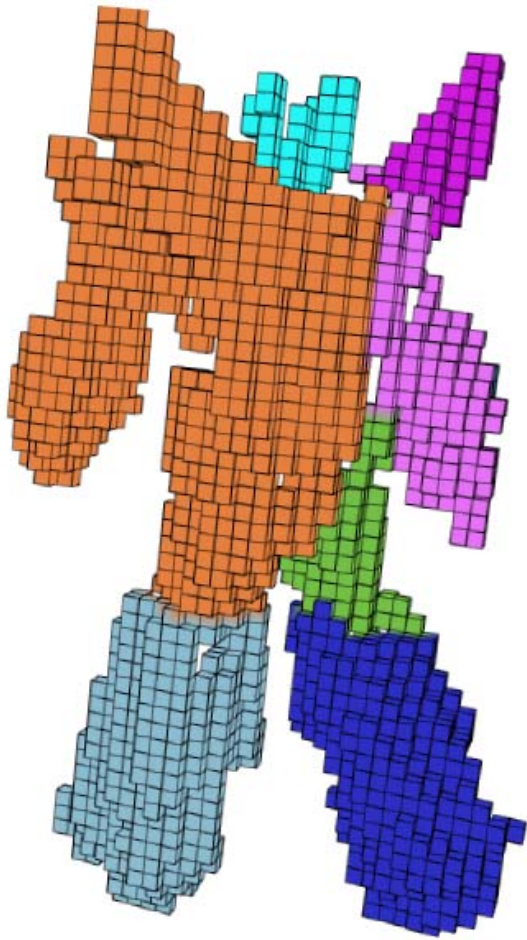


7 bones
4890 cells

Alignment Result

Solved Weights

Interactive posing (real-time recording)



Solved Weights

(7 bones, 1598 cells)



Interactive Posing Result

Average performance statistics

| | Car | Robot | Walk | Hand |
|--------------------|------------|------------|------------|------------|
| Bones | 7 | 7 | 10 | 12 |
| Corresp. | 1200 | 1200 | 1000 | 1500 |
| Vertices | 5389 | 9377 | 4502 | 34342 |
| Max Dist | 20 | 40 | 20 | 30 |
| Grid Res | 60 | 65 | 50 | 40 |
| Grid Cells | 1107 | 1295 | 1014 | 814 |
| Grid Points | 2918 | 3366 | 2553 | 1884 |
| Setup | 0.185 sec | 0.234 sec | 0.136s ec | 0.078 sec |
| RANSAC | 8.089 sec | 20.001 sec | 5.517 sec | N/A |
| Align | 9.945 sec | 19.644 sec | 23.092 sec | 49.918 sec |
| Weight | 6.135 sec | 10.713 sec | 10.497 sec | 3.689 sec |
| Total Time | 24.355 sec | 50.591 sec | 39.242 sec | 53.684 sec |

Limitations

Discussion

- Topology issues with grid
- Limited to a pair of scans
- Limitations with LBS

Conclusion

A new algorithm to align range scans by modeling the motion with a reduced deformable model

- Use LBS to represent the motion
- Represent weight function using a 3D grid
- Solve for the parameters using alternating optimization
- No marker, template, segmentation information
- Robust to occlusion & missing data

Implementation Issues

Determine grid enclosing geometry

- Intersect mesh triangles with regular grid (triangle-box intersection), mark grid cells that intersect

Compute joint constraint integrals

- Derive closed-form expression

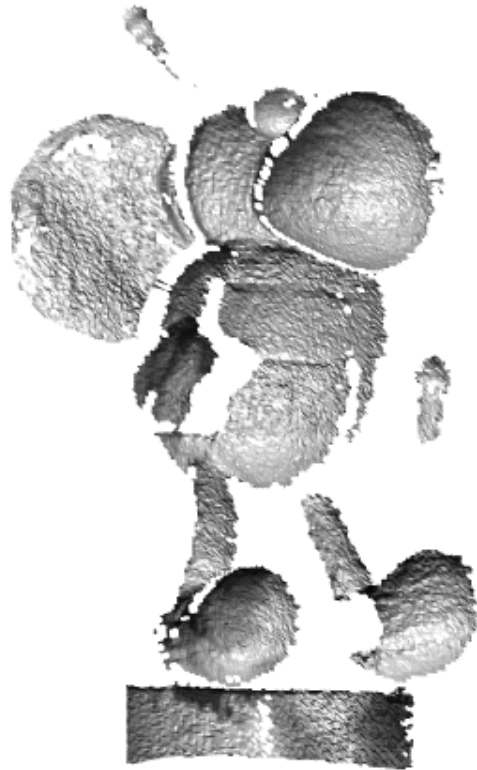
Derive equations for Gauss-Newton optimization

Geometric Registration for Deformable Shapes

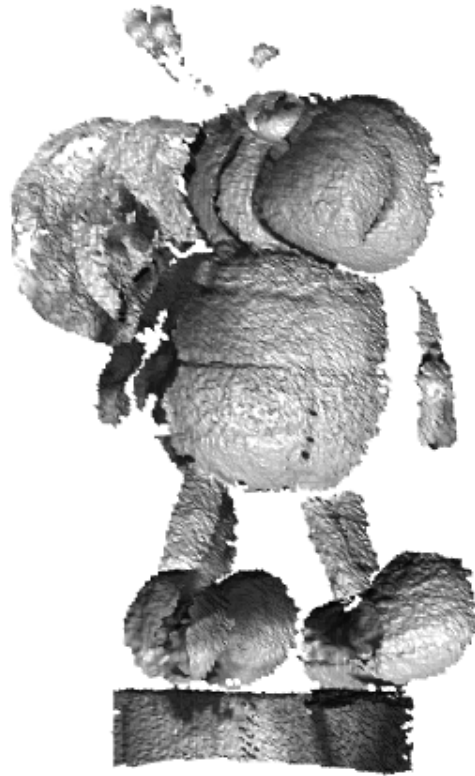
4.1 Dynamic Registration



Scan Registration

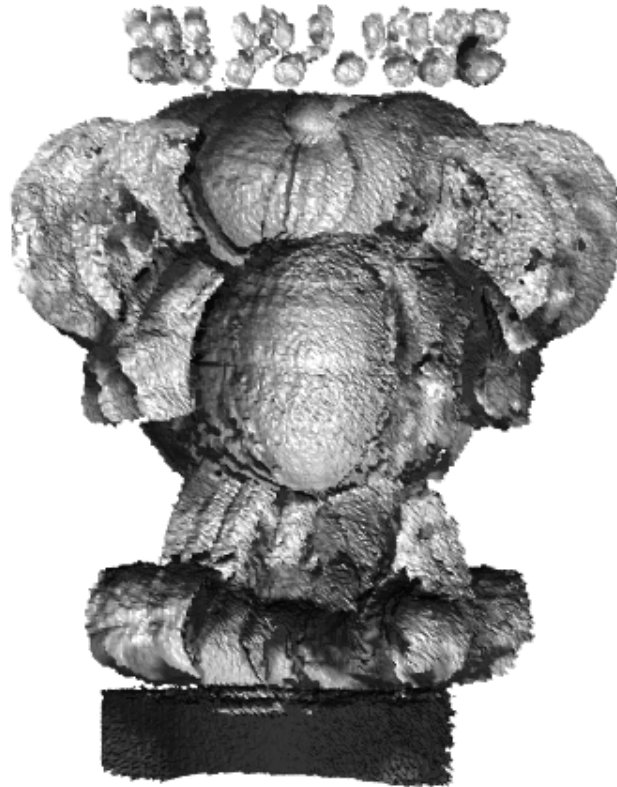


Scan Registration



Solve for inter-frame
motion: $\alpha := (\mathbf{R}, \mathbf{t})$

Scan Registration



Solve for inter-frame
motion: $\alpha_j := (\mathbf{R}_j, \mathbf{t}_j)$

The Setup

Given:

A set of frames $\{P_0, P_1, \dots, P_n\}$

Goal:

Recover rigid motion $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ between adjacent frames

The Setup

Smoothly varying object motion

Unknown correspondence between scans

**Fast acquisition →
motion happens between frames**

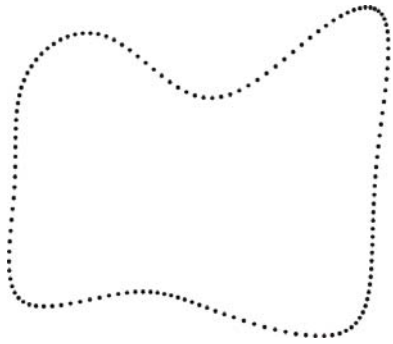
Insights

Rigid registration → kinematic property of space-time surface (locally exact)

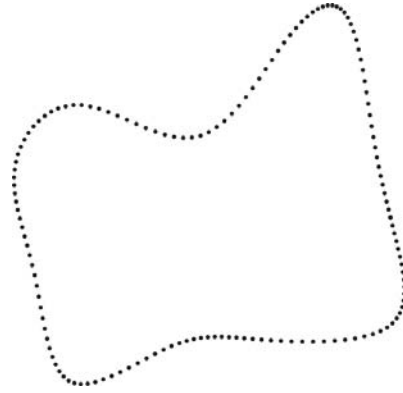
Registration → surface normal estimation

Extension to deformable/articulated bodies

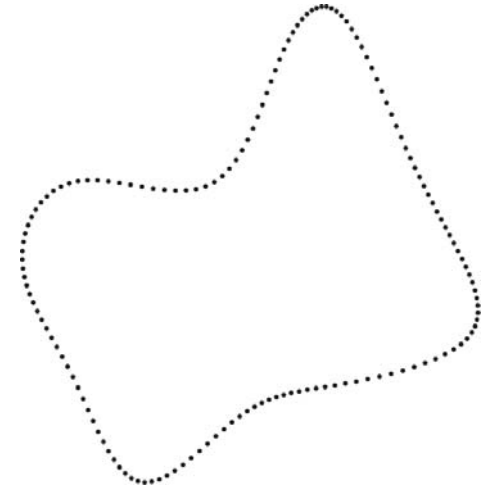
Time Ordered Scans



t^j



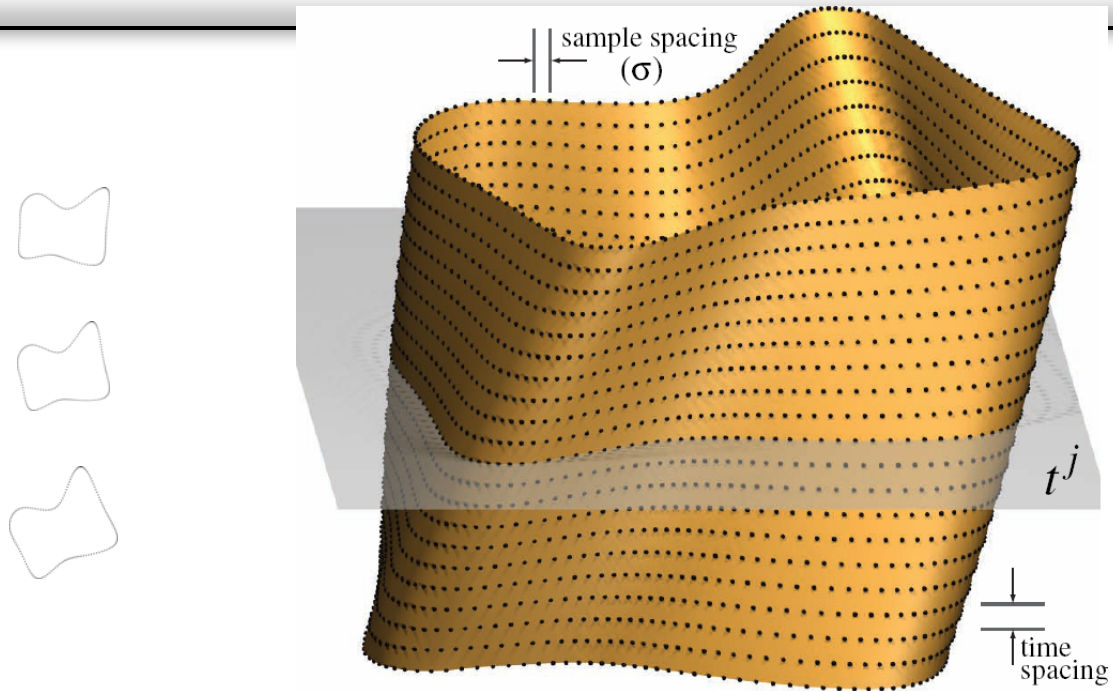
t^{j+1}



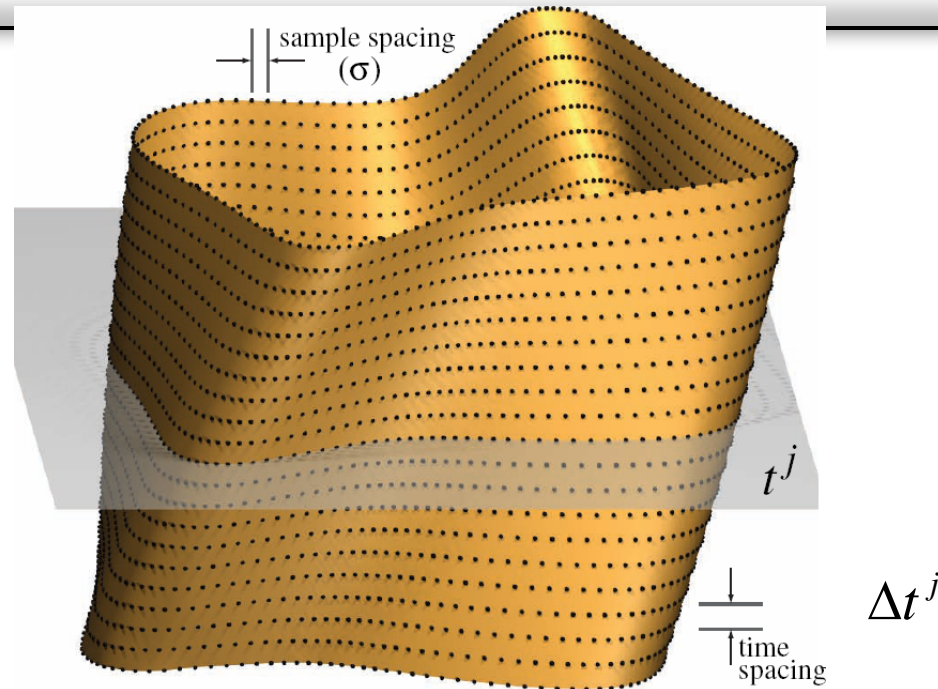
t^{j+2}

$$\tilde{P}^j \equiv \{\tilde{\mathbf{p}}_i^j\} := \{(\mathbf{p}_i^j, t^j), \mathbf{p}_i^j \in \mathbb{R}^d, t^j \in \mathbb{R}\}$$

Space-time Surface

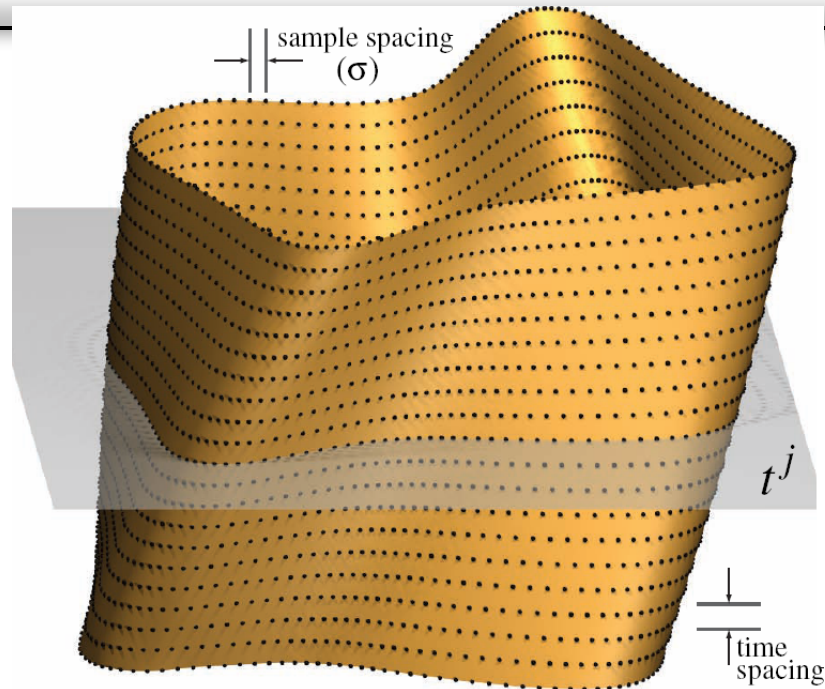


Space-time Surface



$$\tilde{\mathbf{p}}_i^j \rightarrow \tilde{\alpha}_j(\tilde{\mathbf{p}}_i^j) = \left(\mathbf{R}_j \mathbf{p}_i^j + \mathbf{t}_j, \boxed{t^j + \Delta t^j} \right)$$

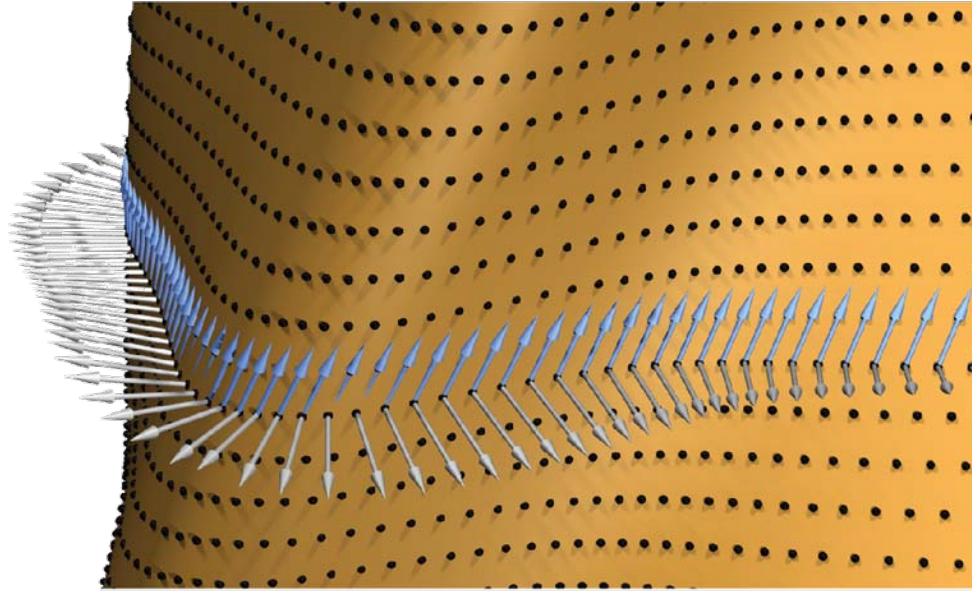
Space-time Surface



$$\tilde{\mathbf{p}}_i^j \rightarrow \tilde{\alpha}_j(\tilde{\mathbf{p}}_i^j) = \left(\mathbf{R}_j \mathbf{p}_i^j + \mathbf{t}_j, t^j + \Delta t^j \right)$$

$$\tilde{\alpha}_j = \operatorname{argmin} \sum_{i=1}^{|P^j|} d^2(\tilde{\alpha}_j(\tilde{\mathbf{p}}_i^j), S)$$

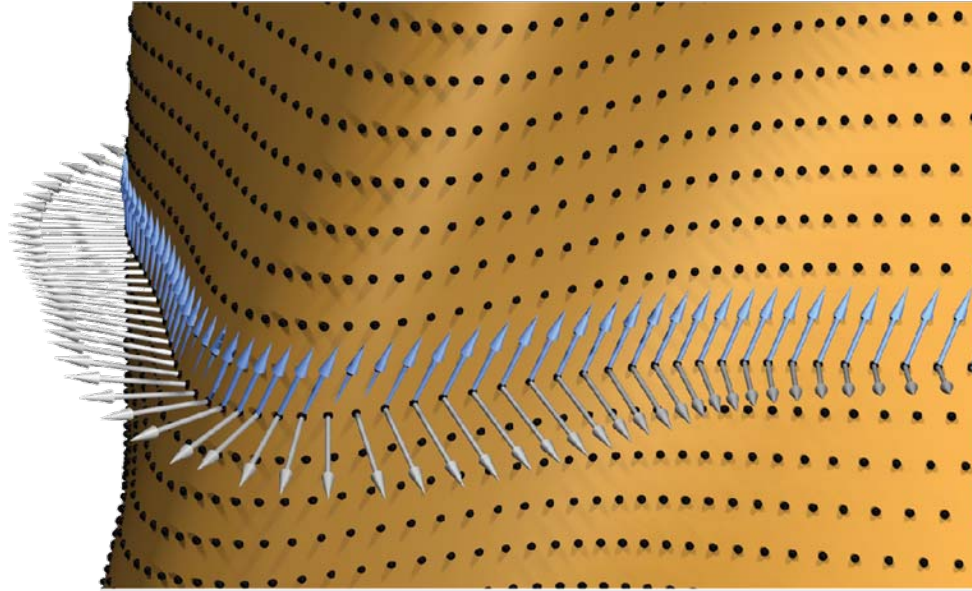
Spacetime Velocity Vectors



Tangential point movement \rightarrow velocity vectors orthogonal to surface normals

$$\tilde{\alpha}_j = \operatorname{argmin} \sum_{i=1}^{|P^j|} d^2(\tilde{\alpha}_j(\tilde{\mathbf{p}}_i^j), S)$$

Spacetime Velocity Vectors



Tangential point movement \rightarrow velocity vectors orthogonal to surface normals

$$v(\tilde{p}_i) \cdot n(\tilde{p}_i) = 0$$

Final Steps

(rigid) velocity vectors $\rightarrow \tilde{\mathbf{v}}(\tilde{\mathbf{p}}_i^j) = (\mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j, 1)$

$$\min_{\mathbf{c}_j, \bar{\mathbf{c}}_j} \sum_{i=1}^{|P^j|} w_i^j \left[(\mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j, 1) \cdot \tilde{\mathbf{n}}_i^j \right]^2$$

Final Steps

(rigid) velocity vectors $\rightarrow \tilde{\mathbf{v}}(\tilde{\mathbf{p}}_i^j) = (\mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j, 1)$

$$\min_{\mathbf{c}_j, \bar{\mathbf{c}}_j} \sum_{i=1}^{|P^j|} w_i^j \left[(\mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j, 1) \cdot \tilde{\mathbf{n}}_i^j \right]^2$$

$$\mathbf{A}\mathbf{x} + \mathbf{b} = 0$$

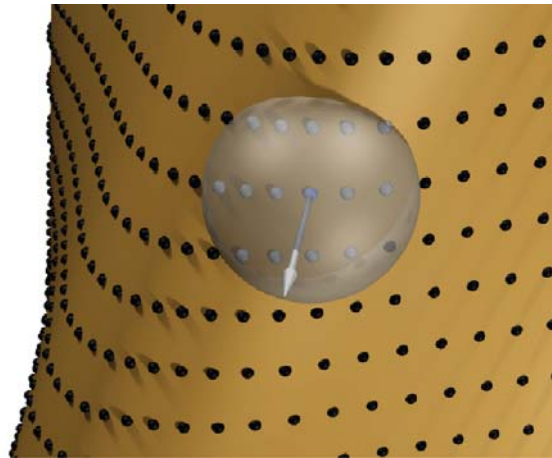
$$\mathbf{A} = \sum_{i=1}^{|P^j|} w_i^j \begin{bmatrix} \tilde{\mathbf{n}}_i^j \\ \mathbf{p}_i^j \times \tilde{\mathbf{n}}_i^j \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{n}}_i^j{}^T & (\mathbf{p}_i^j \times \tilde{\mathbf{n}}_i^j)^T \end{bmatrix}$$

$$\mathbf{b} = \sum_{i=1}^{|P^j|} w_i^j n_i^j \begin{bmatrix} \tilde{\mathbf{n}}_i^j \\ \mathbf{p}_i^j \times \tilde{\mathbf{n}}_i^j \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \bar{\mathbf{c}}_j \\ \mathbf{c}_j \end{bmatrix}$$

Registration Algorithm

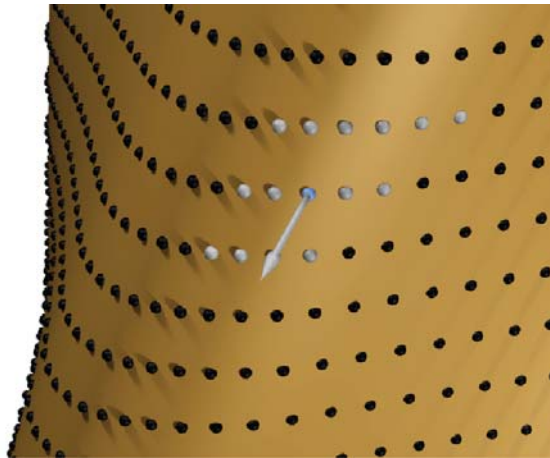
1. Compute time coordinate spacing (σ), and form space-time surface.
2. Compute space time neighborhood using ANN, and locally estimate space-time surface normals.
3. Solve linear system to estimate (c_j, \bar{c}_j) .
4. Convert velocity vectors to rotation matrix + translation vector using Plücker coordinates and quarternions.

Normal Estimation: PCA Based



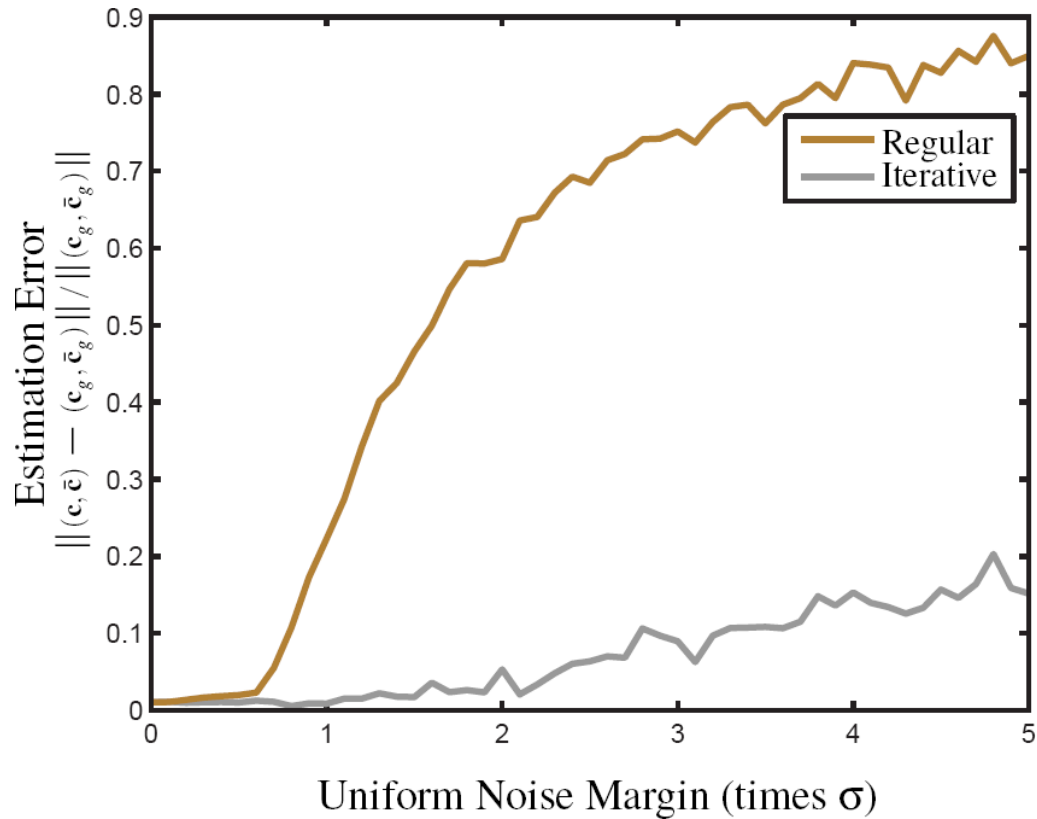
Plane fitting using PCA using chosen neighborhood points.

Normal Estimation: Iterative Refinement



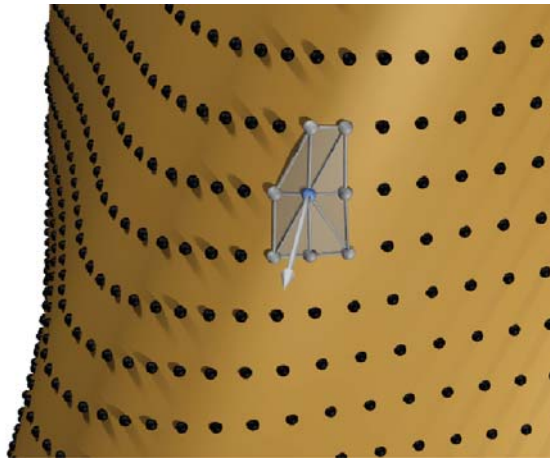
Update neighborhood with current velocity estimate.

Normal Refinement: Effect of Noise



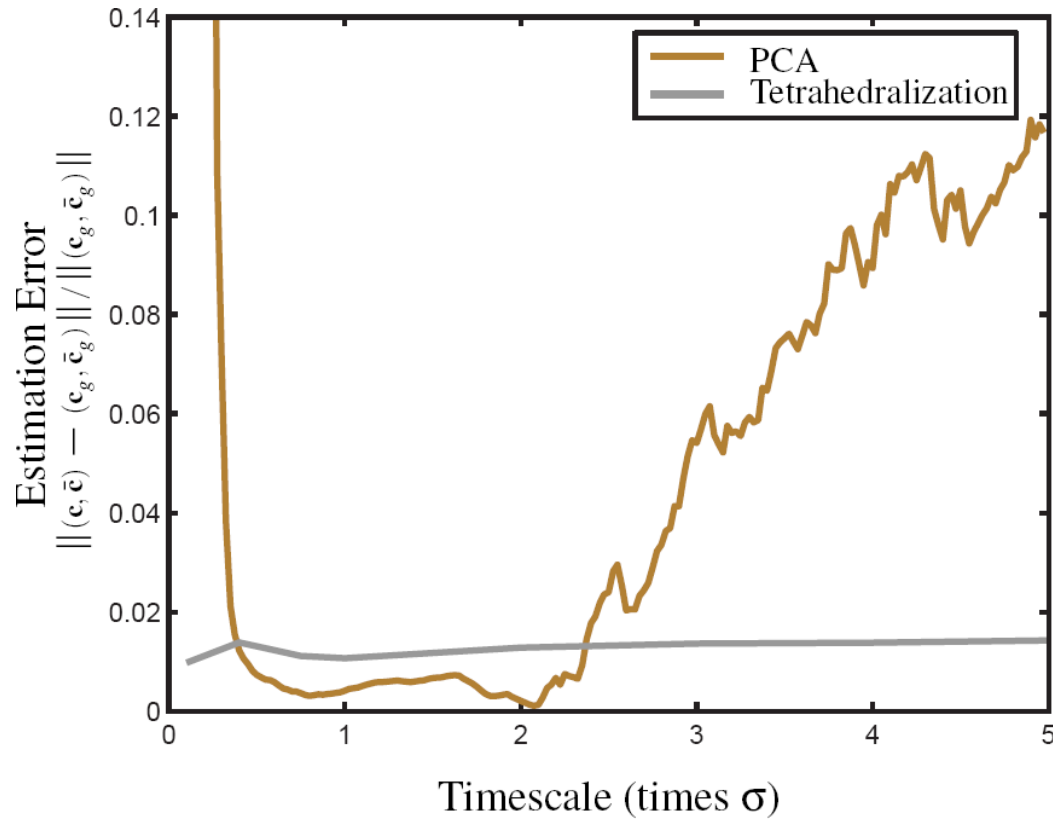
Stable, but more expensive.

Normal Estimation: Local Triangulation



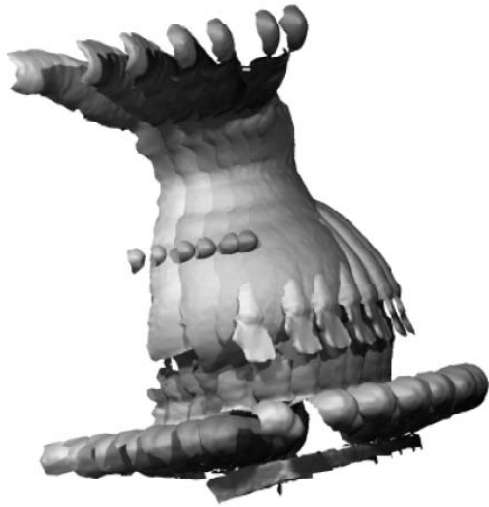
Perform local surface triangulation (tetrahedralization).

Normal Estimation



Stable, but more expensive.

Comparison with ICP



ICP point-
plane

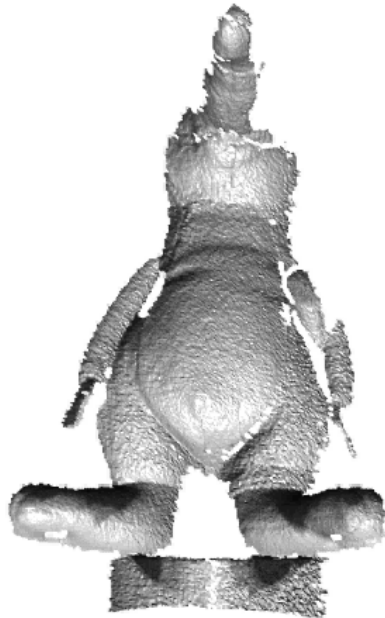


Dynamic registration

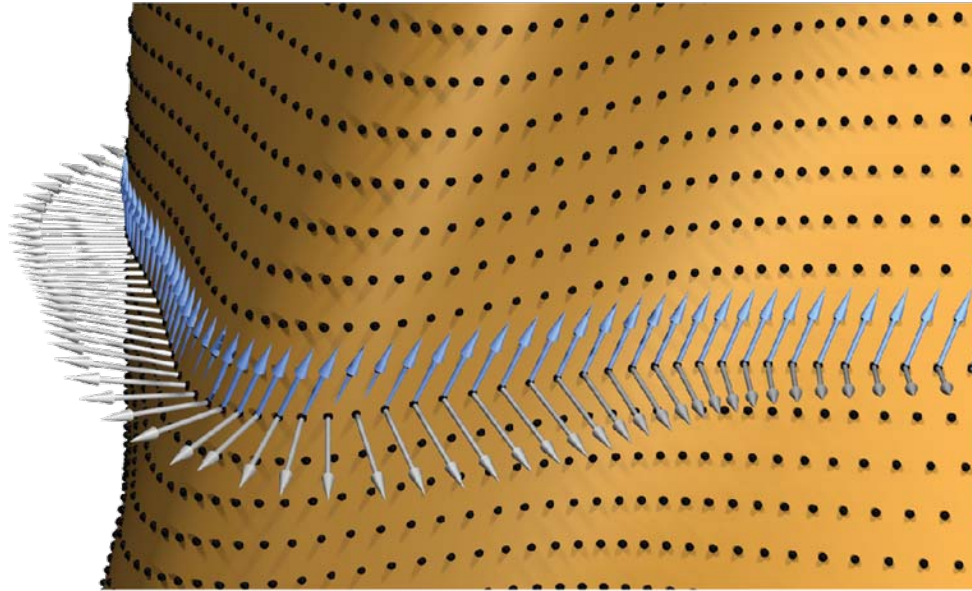
Rigid: Bee Sequence (2,200 frames)



Rigid: Coati Sequence (2,200 frames)

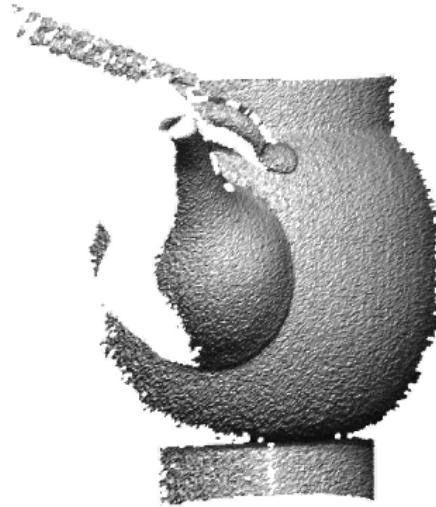


Handling Large Number of Frames



Rigid/Deformable: Teapot Sequence

(2,200 frames)



Deformable Bodies

$$\min_{\mathbf{c}_j, \bar{\mathbf{c}}_j} \sum_{i=1}^{|P^j|} w_i^j \left[(\mathbf{c}_j \times \mathbf{p}_i^j + \bar{\mathbf{c}}_j, 1) \cdot \tilde{\mathbf{n}}_i^j \right]^2$$

Cluster points, and solve smaller systems.

Propagate solutions with regularization.

Deformable: Hand (100 frames)

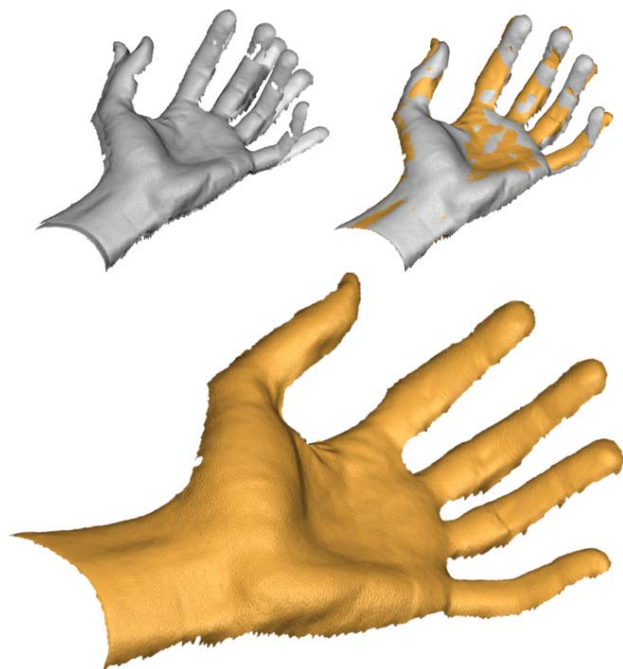


input frames

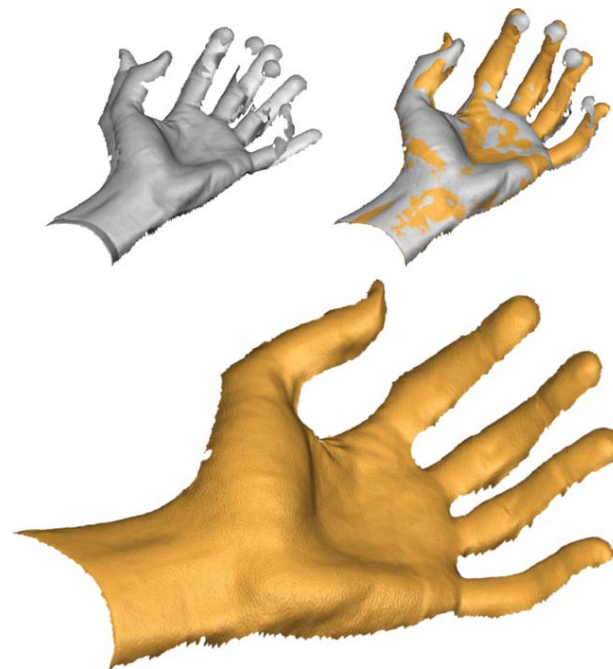


registered result

Deformable: Hand (100 frames)



scan #1 \rightarrow scan #50



scan #1 \rightarrow scan
#100

Deformation + scanner motion: Skeleton (100 frames)



input frames

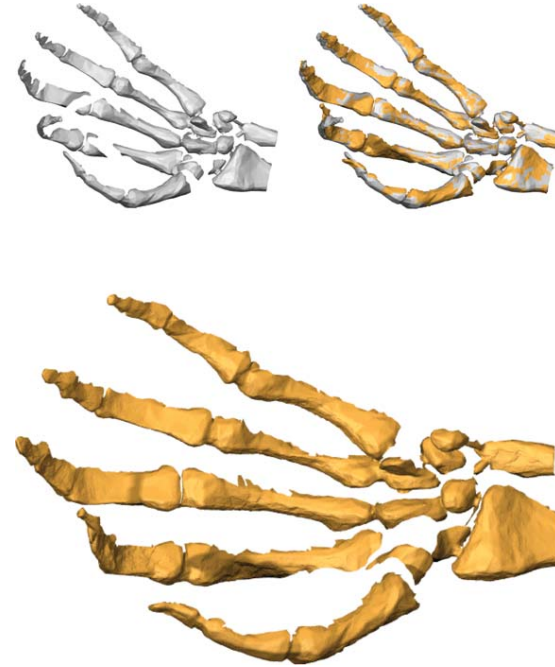


registered result

Deformation + scanner motion: Skeleton (100 frames)

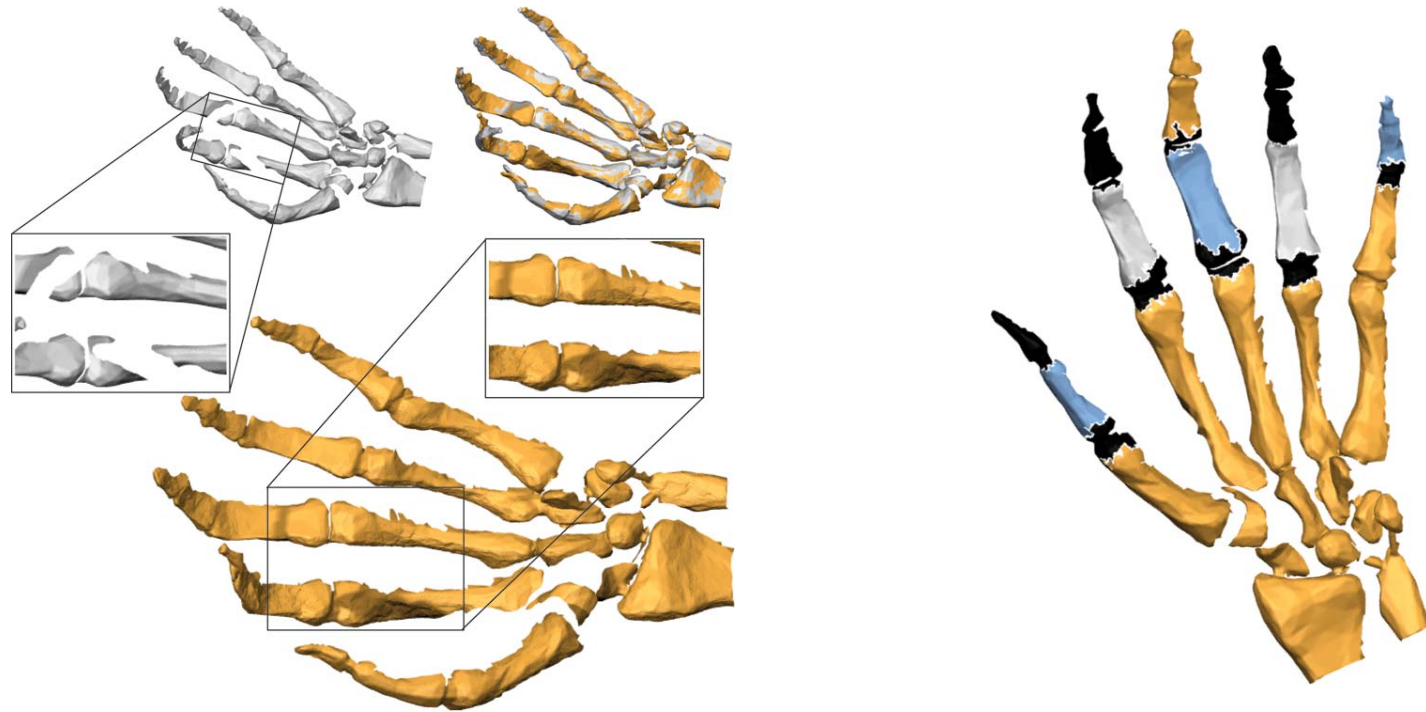


scan #1 \rightarrow scan #50



scan #1 \rightarrow scan
#100

Deformation + scanner motion: Skeleton (100 frames)



rigid components

Performance Table (on 2.4GHz Athlon Dual Core, 2GB RAM)

| Model | # scans | # points/scan (in 1000s) | Time (mins) |
|----------------------|---------|-----------------------------|----------------|
| bunny (simulated) | 314 | 33.8 | 13 |
| bee | 2,200 | 20.7 | 51 |
| coati | 2,200 | 28.1 | 71 |
| teapot (rigid) | 2,200 | 27.2 | 68 |
| skeleton (simulated) | 100 | 55.9 | 11 |
| hand | 100 | 40.1 | 17 |

Conclusion

Simple algorithm using kinematic properties of space-time surface.

Easy modification for deformable bodies.

Suitable for use with fast scanners.

Conclusion

Simple algorithm using kinematic properties of space-time surface.

Easy modification for deformable bodies.

Suitable for use with fast scanners.

Limitations/Future Work



thank you



Geometric Registration for Deformable Shapes

4.2 Animation Reconstruction

Basic Algorithm • Efficiency: Urshape Factorization

Animation Reconstruction

Basic Algorithm

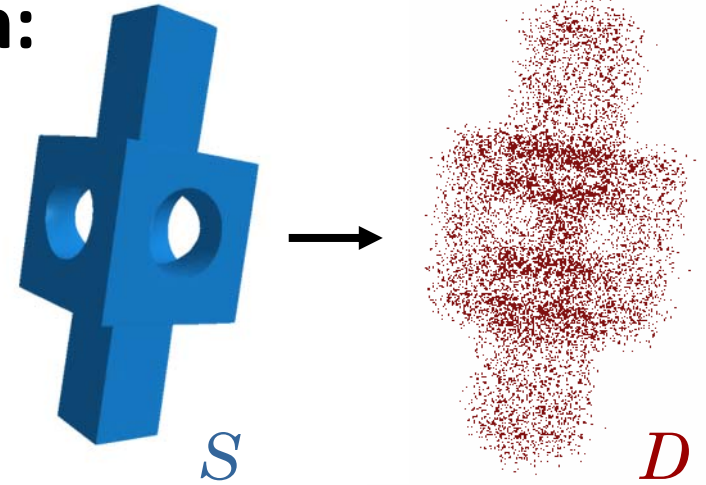
Bayesian Approach

Bayesian surface reconstruction:

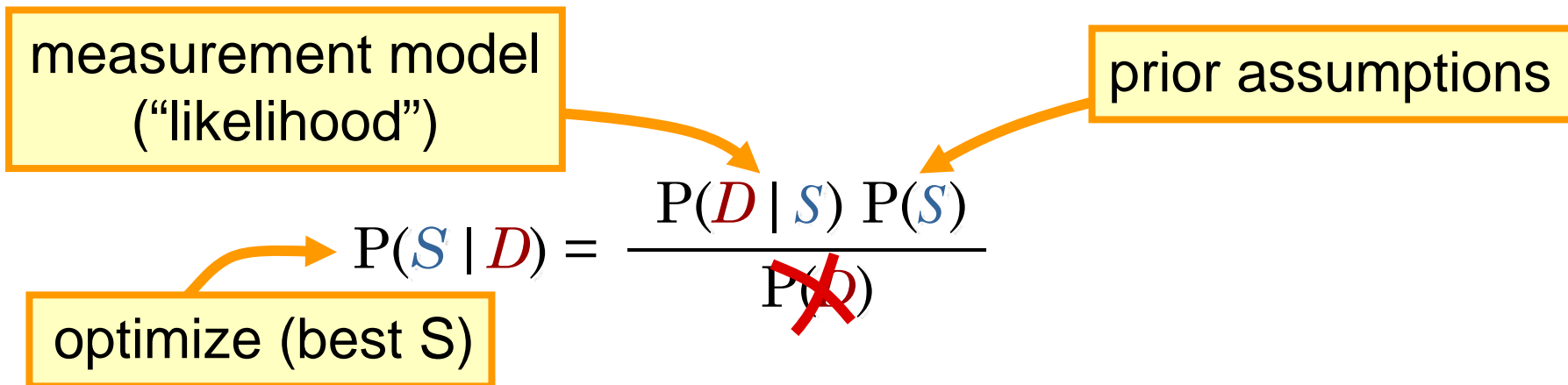
- Probability space
 $\Omega = \Omega_S \times \Omega_D$
- S – original model
 D – measurement data
- Bayes' rule:

$$P(S | D) = \frac{P(D | S) P(S)}{P(D)}$$

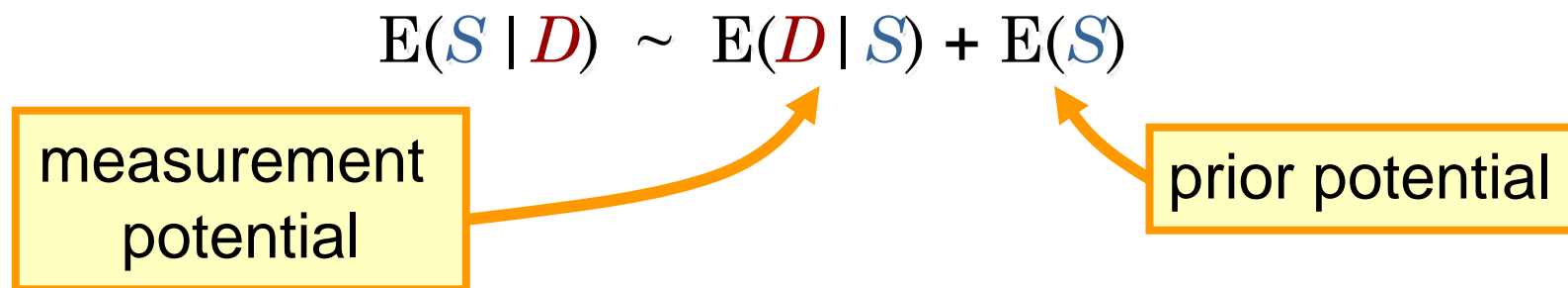
- Find most likely S



Bayesian Approach



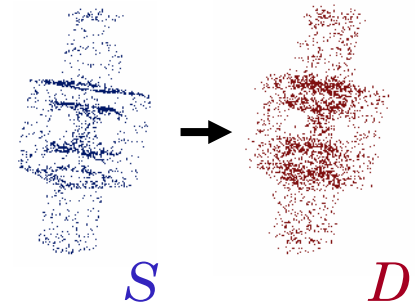
Log Space:



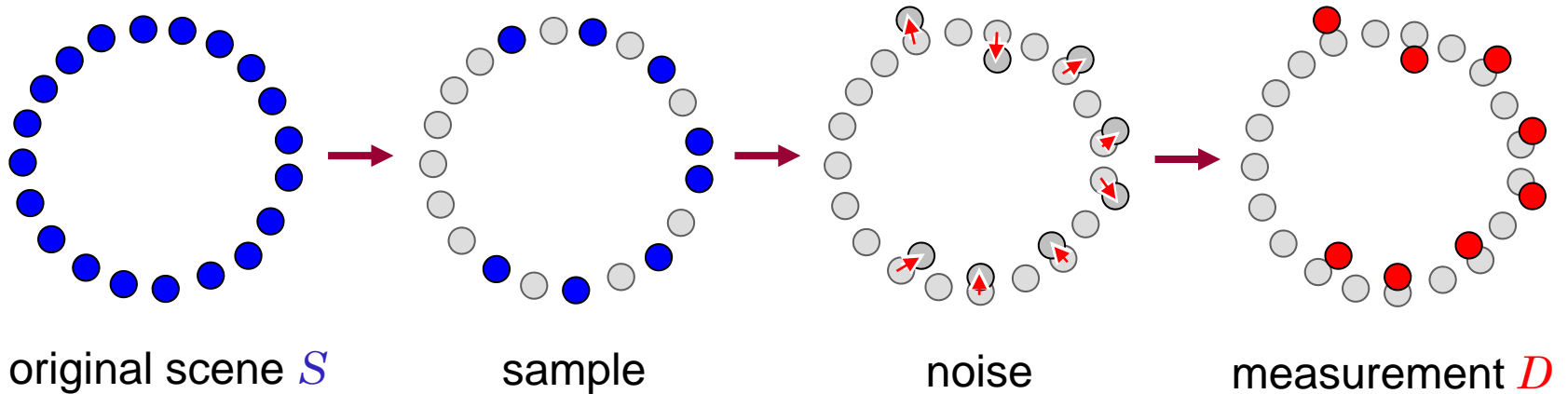
The Space of All Scenes

What is the space of all scenes?

- Discretized model
- Pretend that the original scene has been a point cloud, too.
- $\Omega_S = \mathbb{R}^{3n}$, $\Omega_D = \mathbb{R}^{3m}$
- Define probability density $p(D, S)$ on Ω .
- Truncate p to make it well defined (bounding box support).



Measurement Model – $P(D | S)$



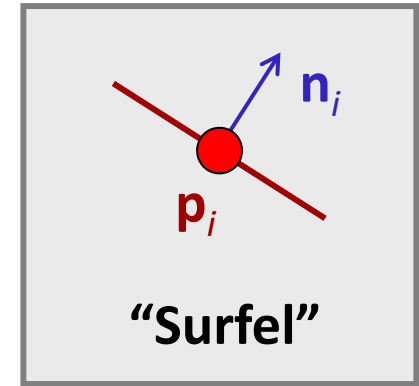
Generative Model:

- **Subsampling**: according to (known) $p_{\text{sampl}}^{(i)}$
- **Noise**: according to (known) $p_{\text{noise}}(x_1, \dots, x_m)$
(currently assuming independent, Gaussian noise)

Implementation...

Implementation: Point-based model

- Our model is a set of points
- “Surfels”: Every point has a latent surface normal
- We want to estimate *position* and *normals*



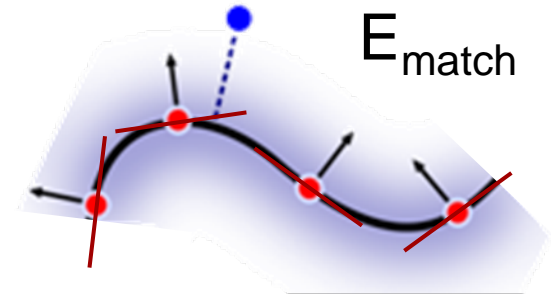
Data Term – $E(D|S)$

Data fitting term:

- Surface should be close to data
- Truncated squared distance function

$$E_{match}(D, S) = \sum_{data\ pts} trunc_{\delta}(dist(S, \mathbf{d}_i)^2)$$

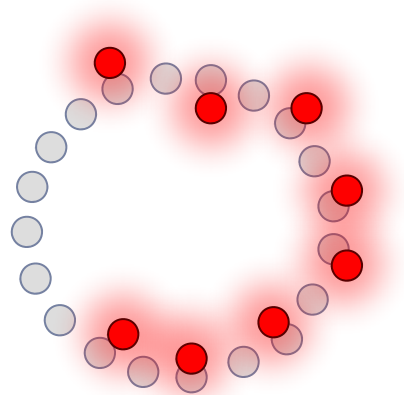
- Sum of distances² of data points to surfel planes
- Point-to-plane: No exact 1:1 match necessary
- Truncation (M-estimator): Robustness to outliers



Why do We Need Priors?

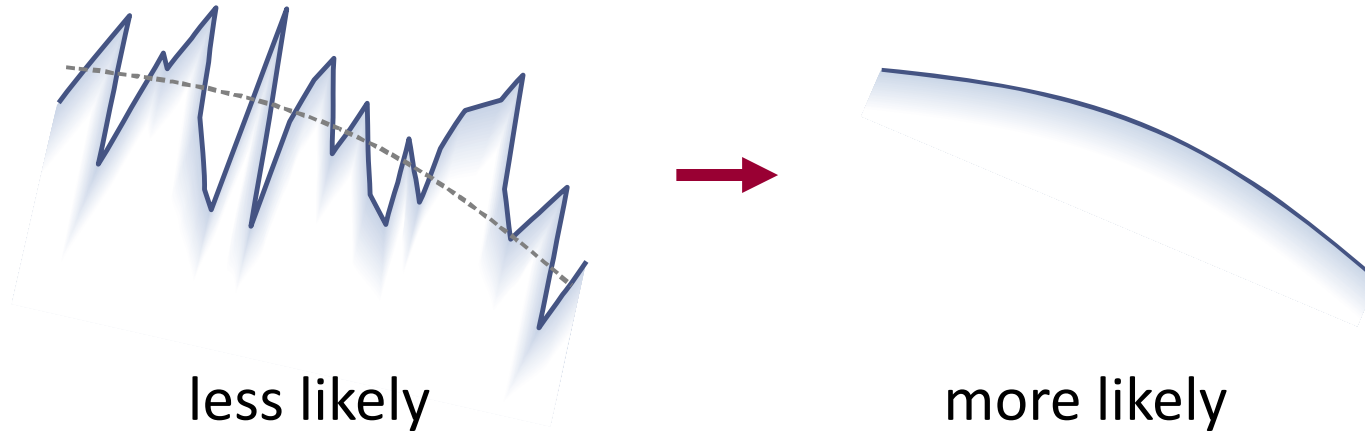
No Reconstruction without Priors

- Non-measurement points unconstrained
- For the rest:
Measurement itself has highest probability



measurement D

Priors – P(S)



Canonical assumption: smooth surfaces

- Correlations between neighboring points

Point-based Model

Simple Smoothness Priors:

- Similar surfel normals:

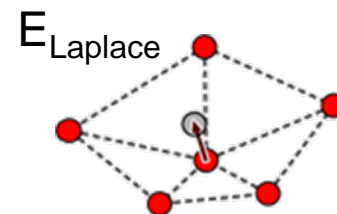
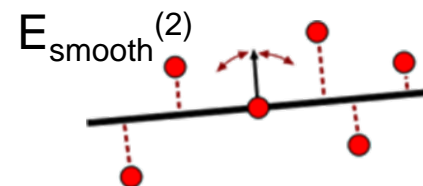
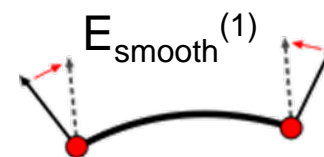
$$E_{smooth}^{(1)}(S) = \sum_{surfels} \sum_{neighbors} (n_i - n_{i_j})^2, \quad \|n_i\| = 1$$

- Surfel positions – flat surface:

$$E_{smooth}^{(2)}(S) = \sum_{surfels} \sum_{neighbors} \langle \mathbf{s}_i - \mathbf{s}_{i_j}, \mathbf{n}(\mathbf{s}_i) \rangle^2$$

- Uniform density:

$$E_{Laplace}(S) = \sum_{surfels} \sum_{neighbors} (\mathbf{s}_i - average)^2$$



[c.f. Szeliski et al. 93]

Nasty Normals

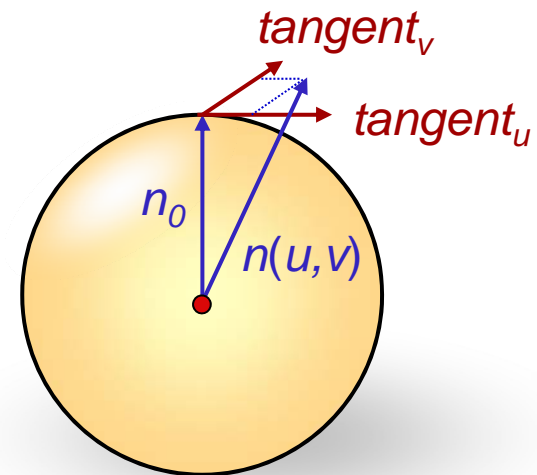
Optimizing Normals

- Problem: $E_{smooth}^{(1)}(S) = \sum_{surfels} \sum_{neighbors} (n_i - n_{i_j})^2, \text{ s.t. } \|n_i\| = 1$
- Need unit normals: constraint optimization
- Unconstraint: trivial solution (all zeros)

Nasty Normals

Solution: Local Parameterization

- Current normal estimate
- Tangent parameterization
- New variables u, v
- Renormalize
- Non-linear optimization
- No degeneracies



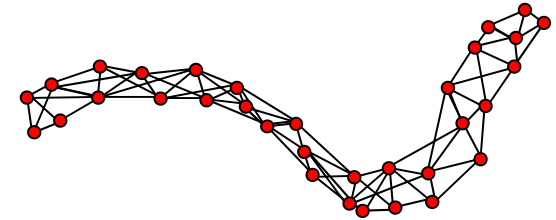
$$n(u, v) = n_0 + u \cdot tangent_u + v \cdot tangent_v$$

[Hoffer et al. 04]

Neighborhoods?

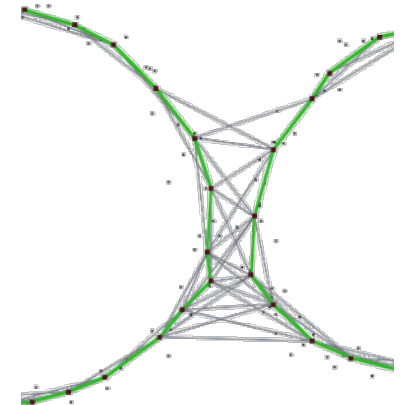
Topology estimation

- Domain of S , base shape (topology)
- Here, we assume this is easy to get
- In the following
 - k -nearest neighborhood graph
 - Typically: $k = 6..20$



Limitations

- This requires dense enough sampling
- Does not work for undersampled data



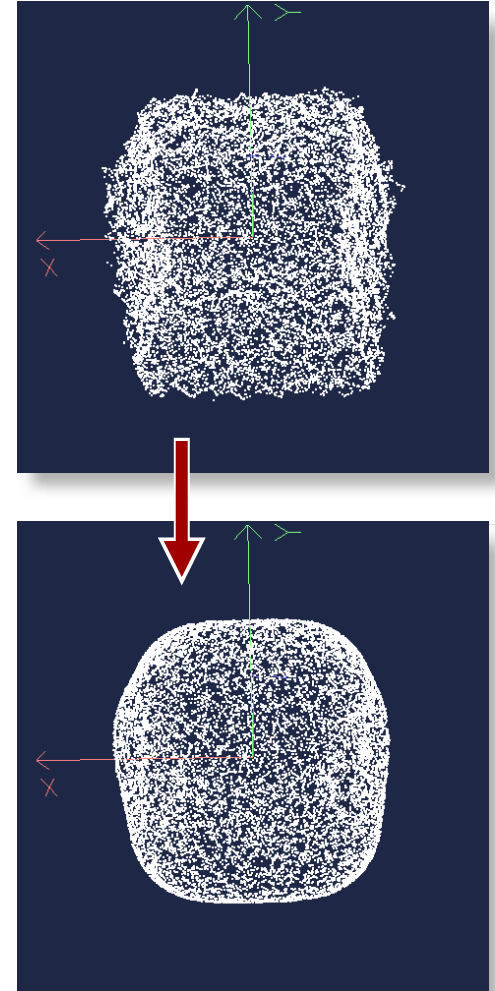
Numerical Optimization

Task:

- Compute most likely “original scene” S
- Nonlinear optimization problem

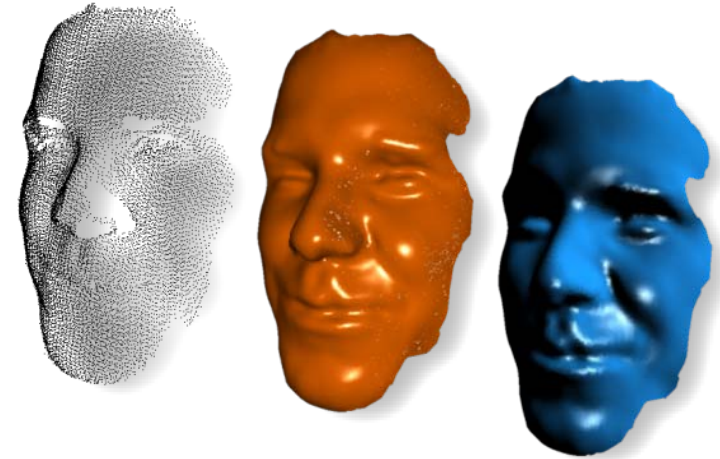
Solution:

- Create initial guess for S
 - Close to measured data
 - Use original data
- Find local optimum
 - (Conjugate) gradient descent
 - (Gauss-) Newton descent

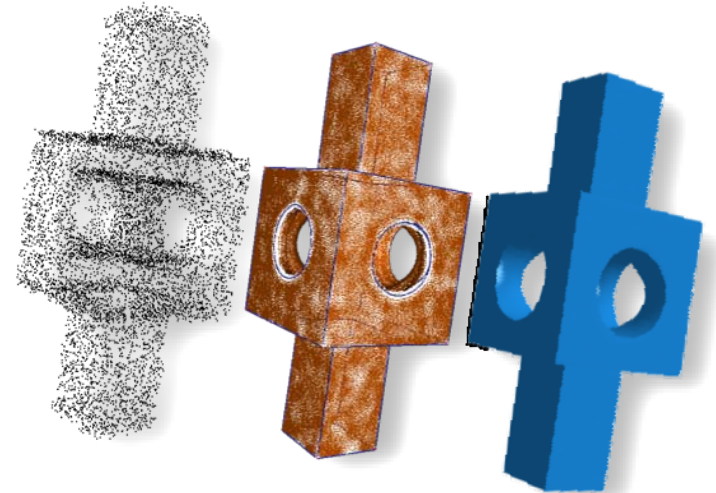


3D Examples

3D reconstruction results:



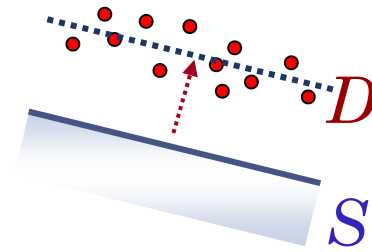
With discontinuity lines:



3D Reconstruction Summary

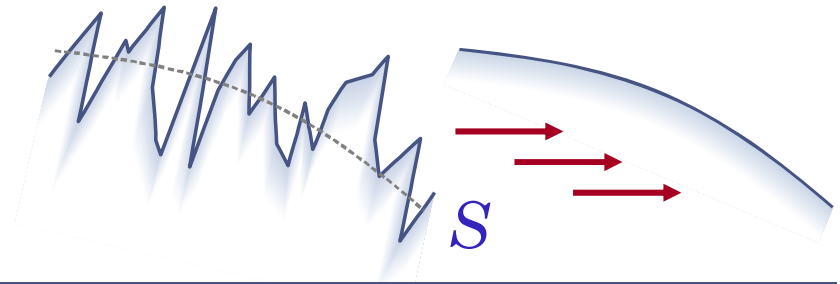
Data fitting:

$$E(D | S) \sim \sum_i \text{dist}(S, d_i)^2$$



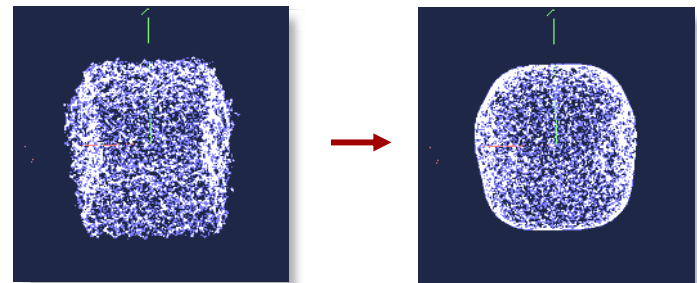
Prior: Smoothness

$$E_s(S) \sim \int_S \text{curv}(S)^2$$



Optimization:

Yields 3D Reconstruction



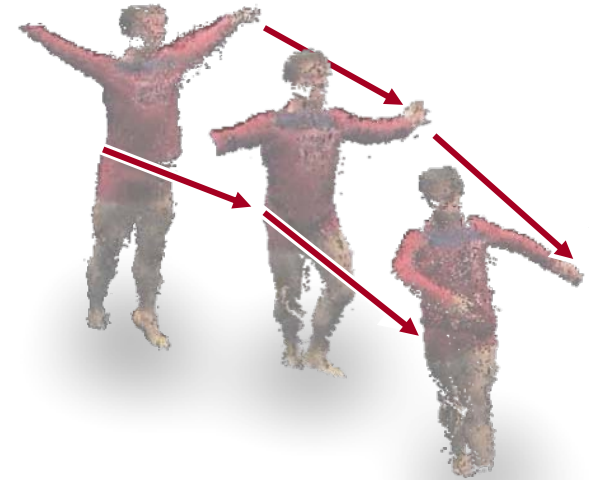
Animation Reconstruction

Improved Algorithm

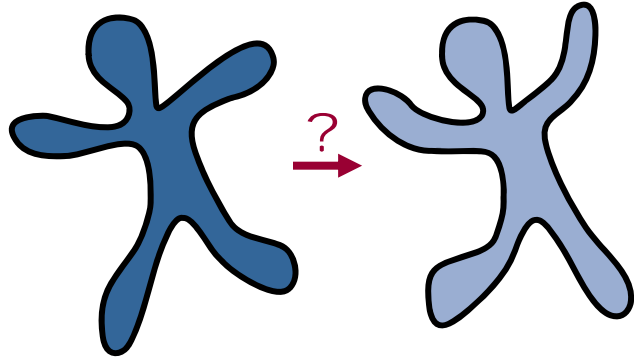
Extension to Animations

Animation Reconstruction

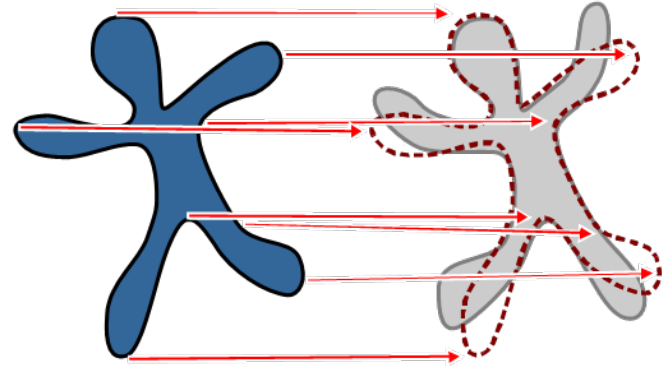
- Not just a 4D version
 - Moving geometry, not just some hypersurface
- Key component: correspondences
 - Latent variables (no direct measurement)
 - Inferred by *motion priors*
- Intuition for “good correspondences”:
 - Match target shape
 - Little deformation



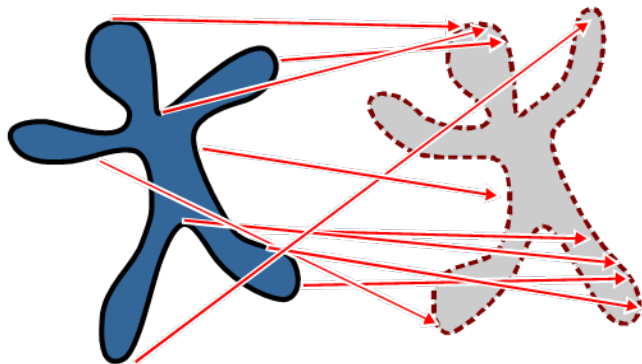
Recap: Correspondences



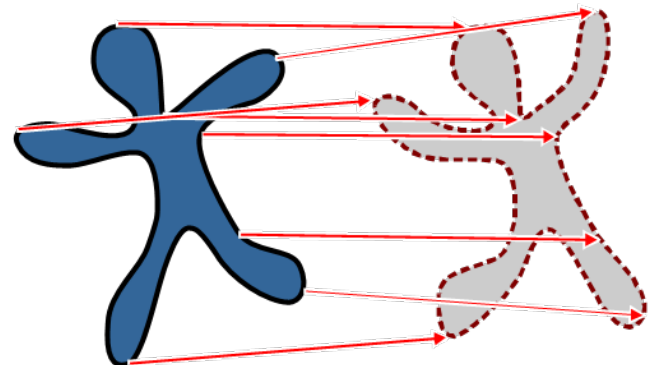
Correspondences?



X no shape match



X too much deformation

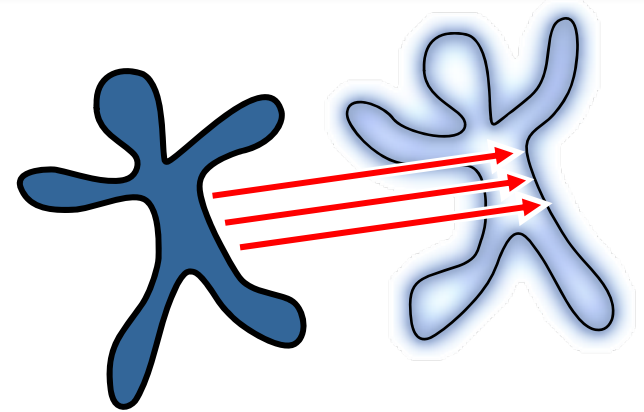


✓ optimum

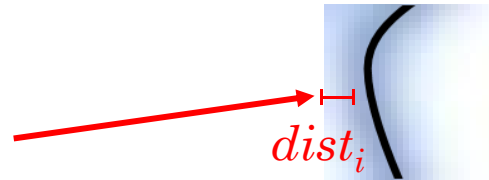
Recap: Correspondences

Model:

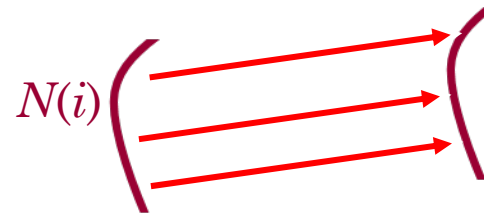
$$-\log p = \sum_{i=1}^n \left[\text{dist}_i^2 + \text{rigid}_{N(i)}^2 \right]$$



Distance:



Deformation / rigidity:

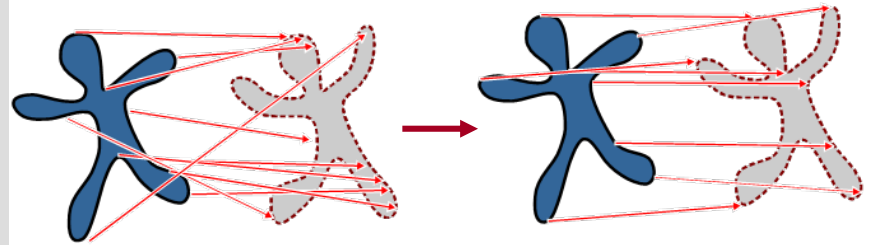


Animation Reconstruction

Two additional priors:

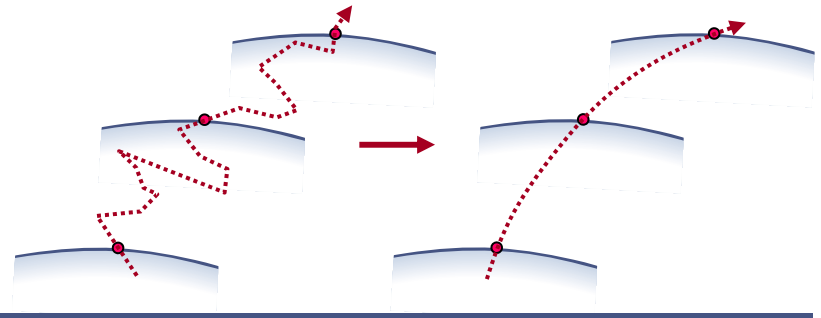
Deformation

$$E_d(\mathbf{S}) \sim \int_{\mathbf{S}} \text{deform}(\mathbf{S}_t, \mathbf{S}_{t+1})^2$$

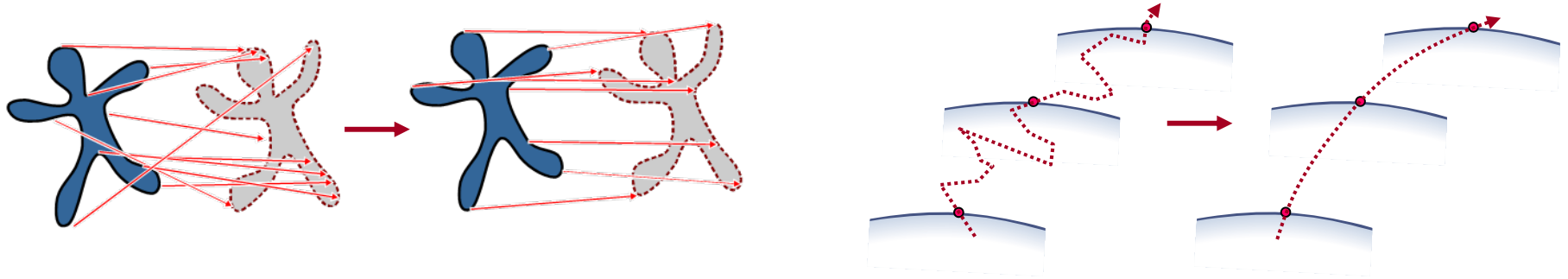


Acceleration

$$E_a(\mathbf{S}) \sim \int_{\mathbf{S}, t} \ddot{\mathbf{s}}(x, t)^2$$



Animation Reconstruction



Not just smooth 4D reconstruction!

- Minimize
 - Deformation
 - Acceleration
- This is quite different from smoothness of a 4D hypersurface.

Animations

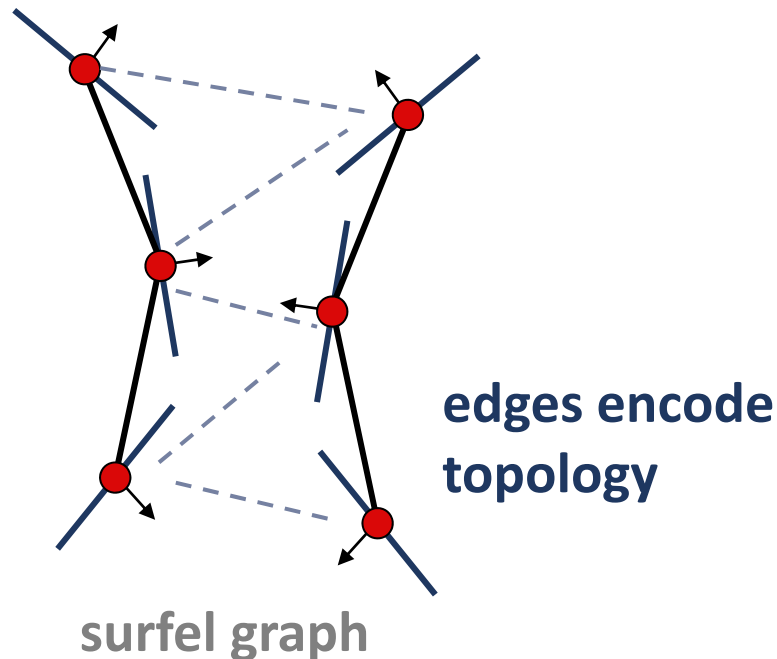
Refined parametrization of reconstruction **S**

- Surfel graph (3D)
- Trajectory graph (4D)

Discretization

Refined parametrization of reconstruction S

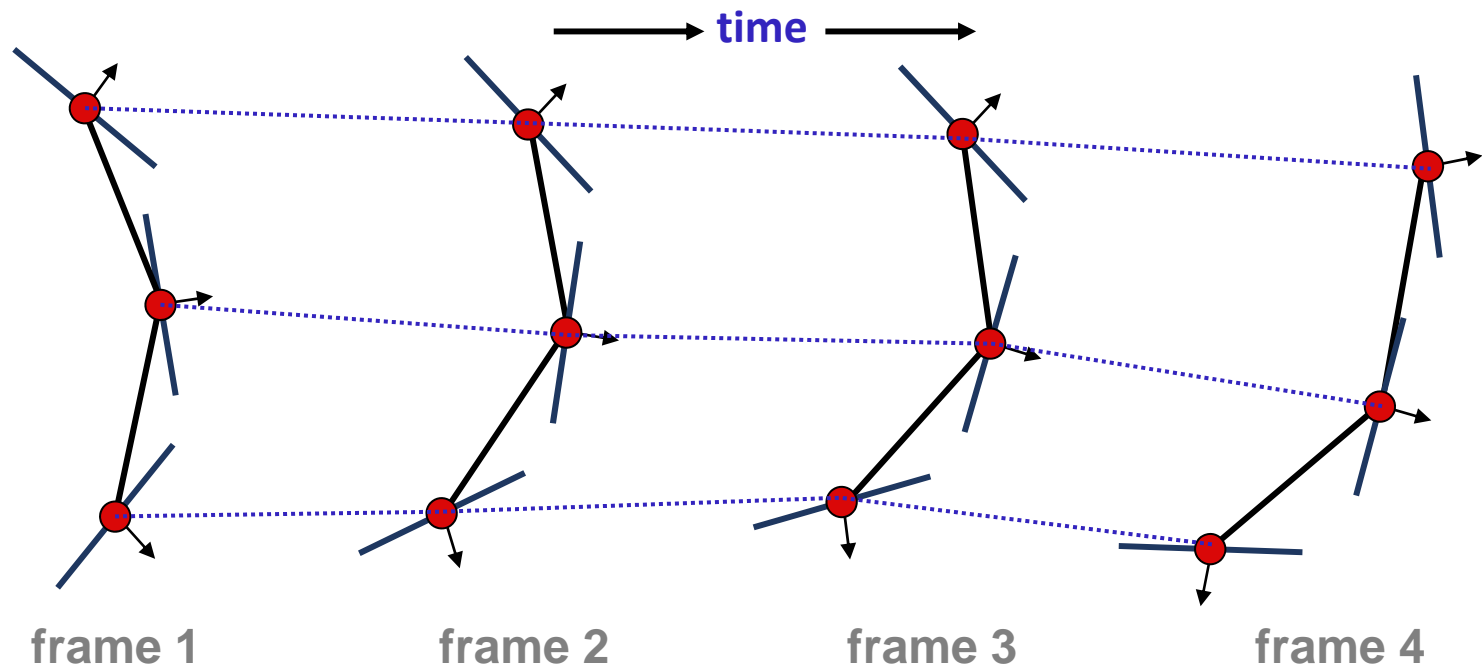
- Surfel graph (3D)
- Trajectory graph (4D)



Discretization

Refined parametrization of reconstruction S

- Surfel graph (3D)
- Trajectory graph (4D)



Missing Details...

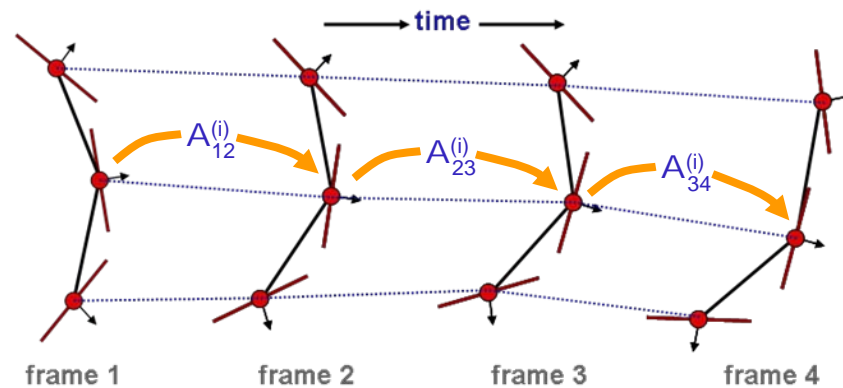
How to implement...

- The deformation priors?
 - We use one of the models previously developed
- Acceleration priors?
 - This is rather simple...

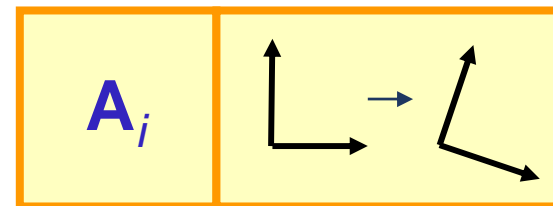
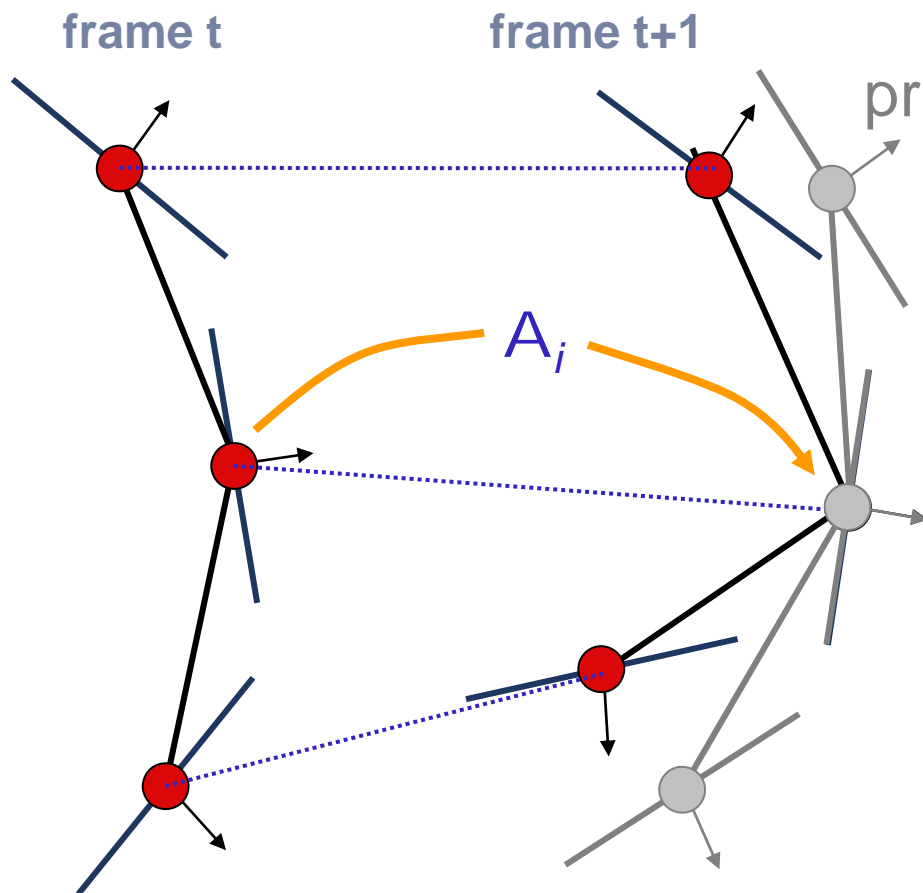
Recap: Elastic Deformation Model

Deformation model

- Latent transformation $\mathbf{A}^{(i)}$ per surfel
- Transforms *neighborhood* of s_i
- Minimize error (both surfels and $\mathbf{A}^{(i)}$)



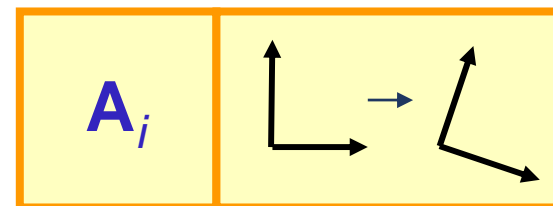
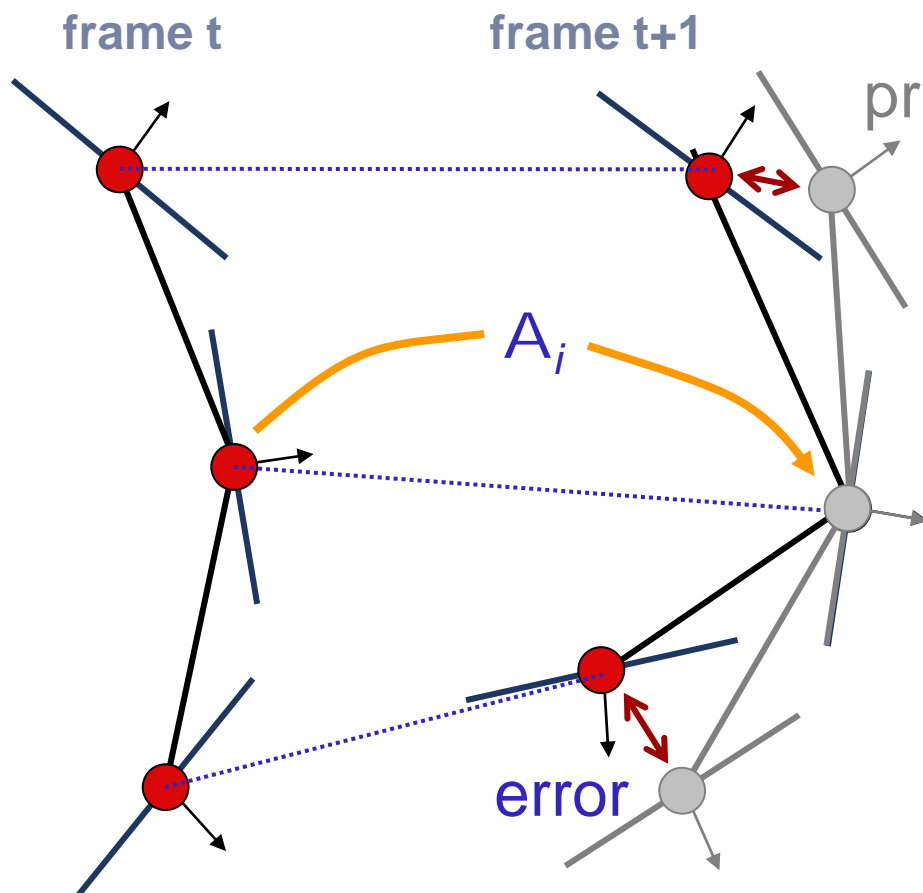
Recap: Elastic Deformation Model



Orthonormal Matrix A_i

per surfel (neighborhood),
latent variable

Recap: Elastic Deformation Model



Orthonormal Matrix \mathbf{A}_i

per surfel (neighborhood),
latent variable

$$E_{deform}(S) = \sum_{surfels} \sum_{neighbors} \left[\mathbf{A}_i^t (\mathbf{s}_i^{(t)} - \mathbf{s}_{i_j}^{(t)}) - (\mathbf{s}_i^{(t+1)} - \mathbf{s}_{i_j}^{(t+1)}) \right]^2$$

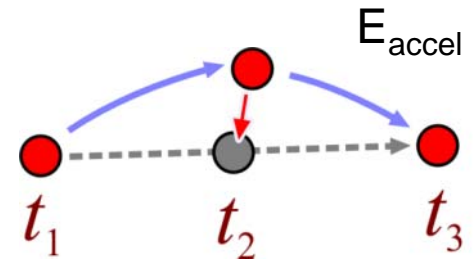
Acceleration

Acceleration priors

- Penalize non-smooth trajectories

$$E_{accel}(A) = \left[\mathbf{s}_i^{t-1} - 2\mathbf{s}_i^t + \mathbf{s}_i^{t+1} \right]^2$$

- Filters out temporal noise



Optimization

For optimization, we need to know:

- The surfel graph
- A (rough) initialization close to correct solution

Optimization:

- Non-linear *continuous optimization* problem
- Gauss-Newton solver (fast & stable)

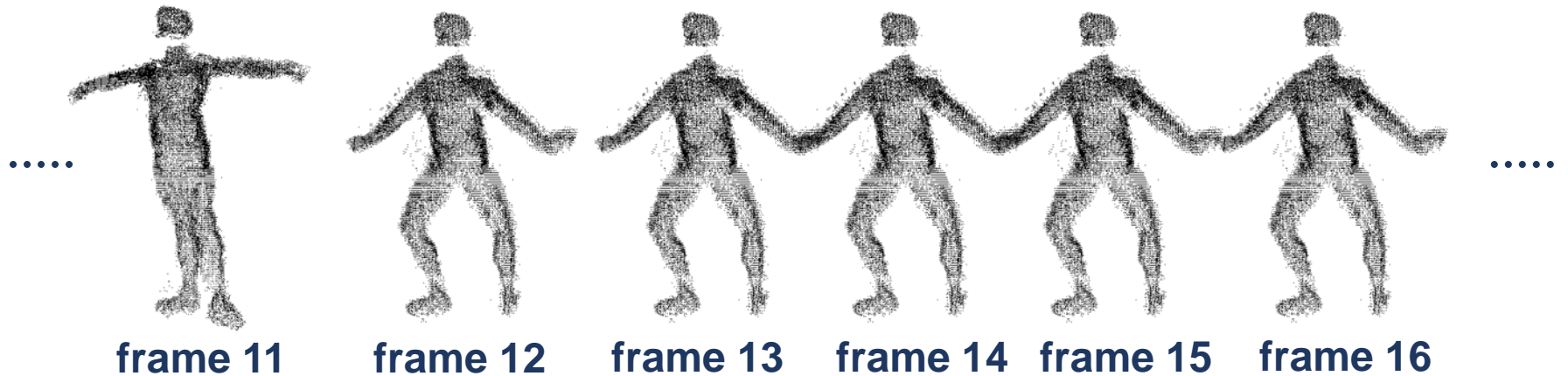
How do we get the initialization?

- *Iterative assembly* heuristic to build & init graph

Global Assembly

Assumption: Adjacent frames are similar

- Every frame is a good initialization for the next one
- Solve for frame pairs

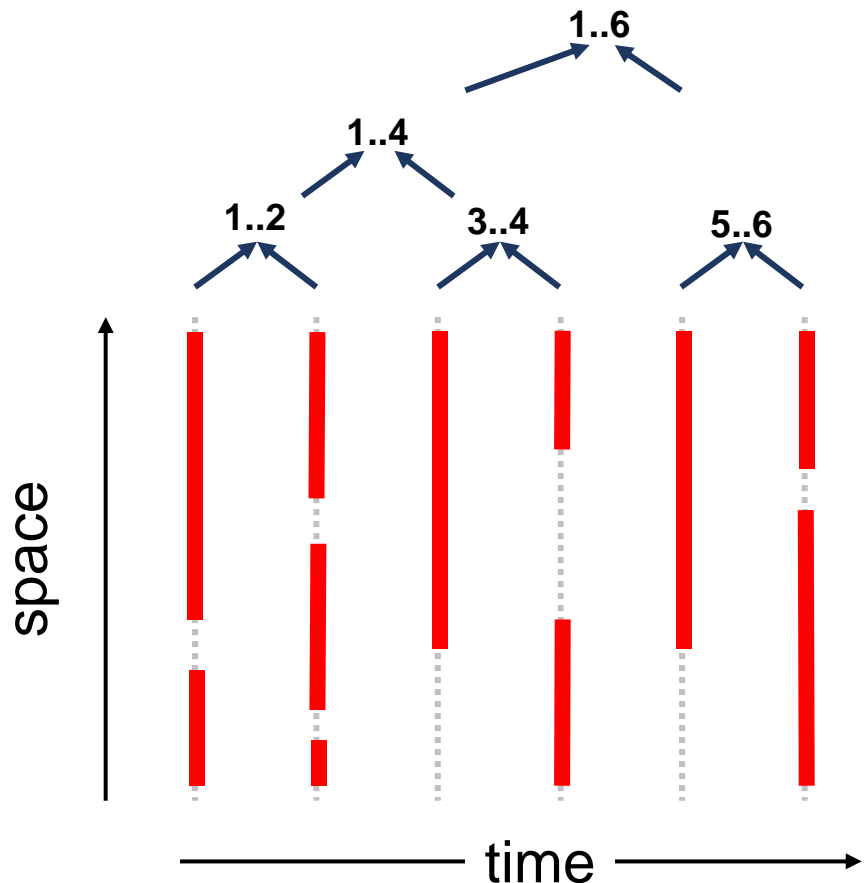


[data set courtesy of C. Theobald, MPI-Inf]

Iterative Assembly

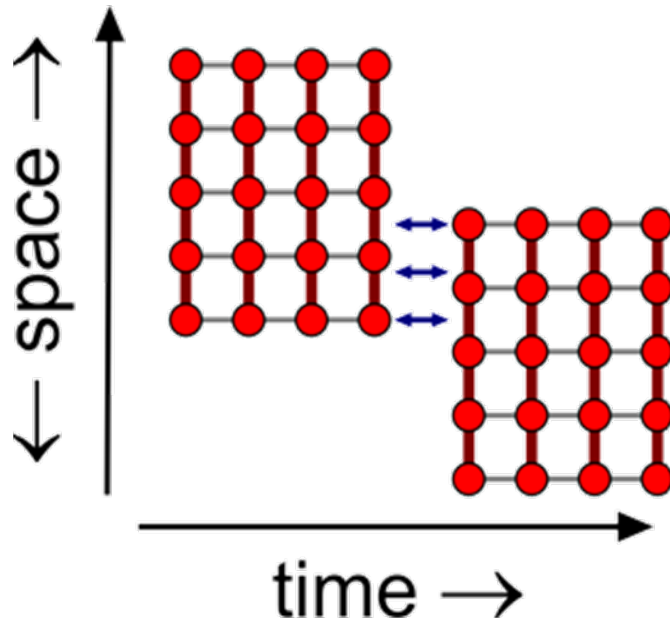
Iterative assembly

- Merge adjacent frames
- Propagate hierarchically
- Global optimization
(avoid error propagation)

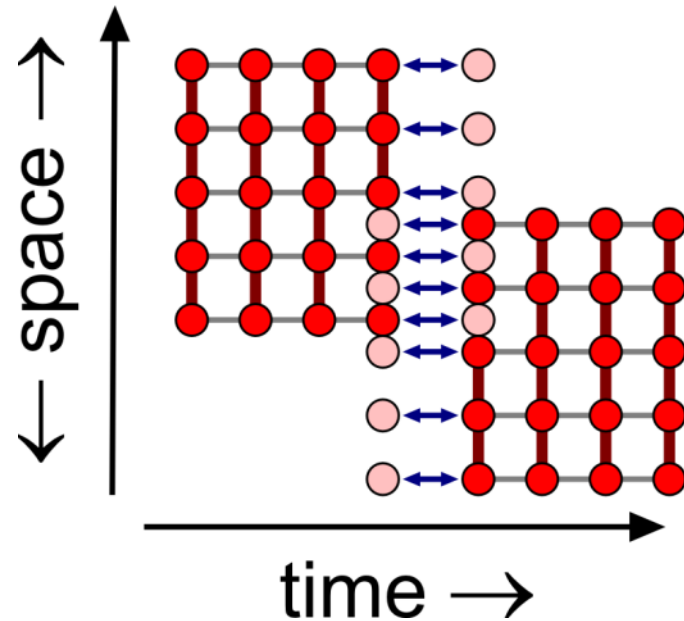


Iterative Assembly

Pairwise alignment



adjacent
trajectory sets

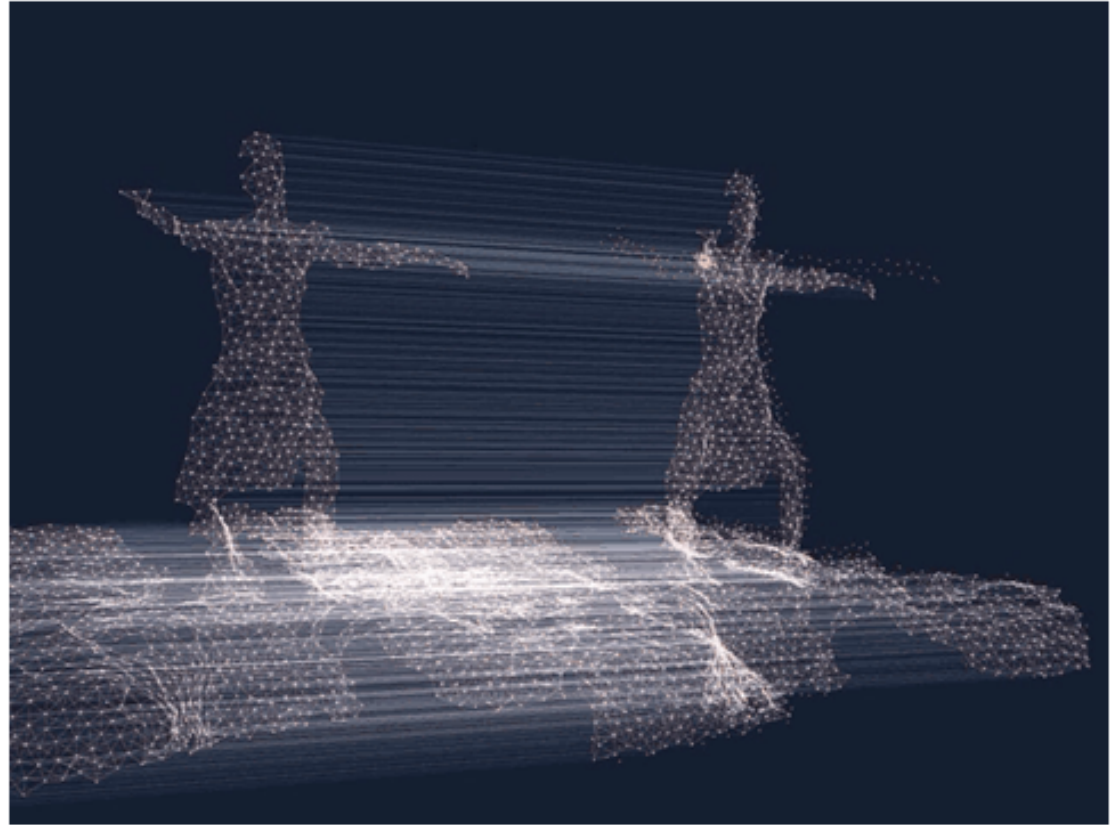


aligned
frames

Alignment

Alignment:

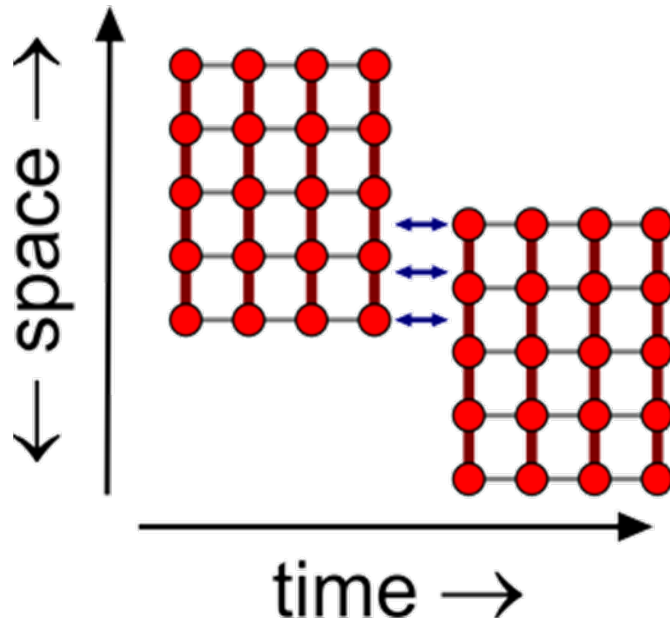
- Two frames
- Use one frame as initialization
- Second frame as “data points”
- Optimize



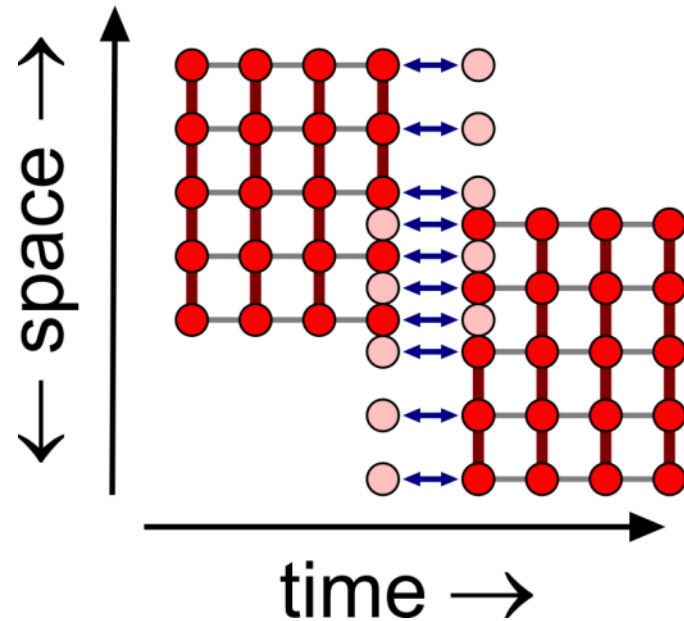
[data set: Zitnick et al., Microsoft Research]

Iterative Assembly

Pairwise alignment



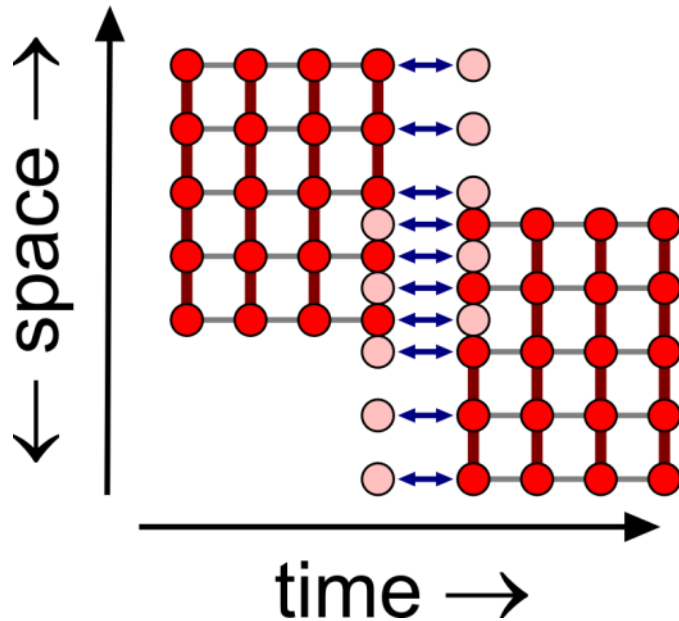
adjacent
trajectory sets



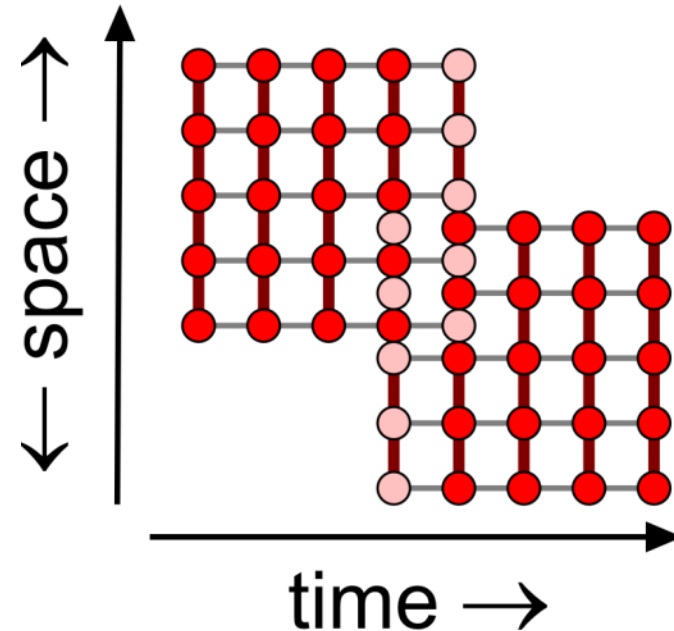
aligned
frames

Iterative Assembly

Topology stitching



aligned
frames

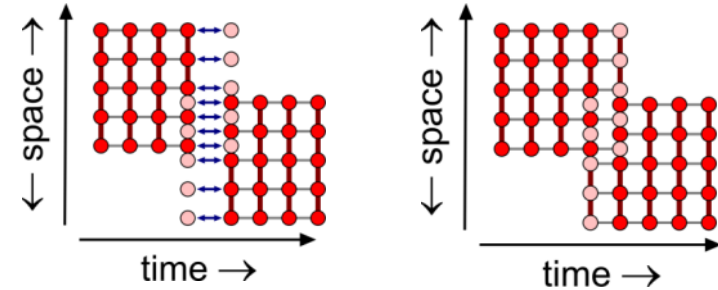


merged
topology

Topology Stitching

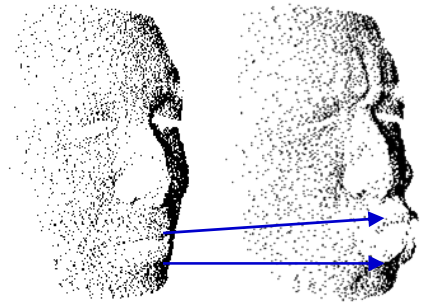
Recompute Topology

- Recompute kNN/ ϵ -graph
- Topology is global



Sanity Check:

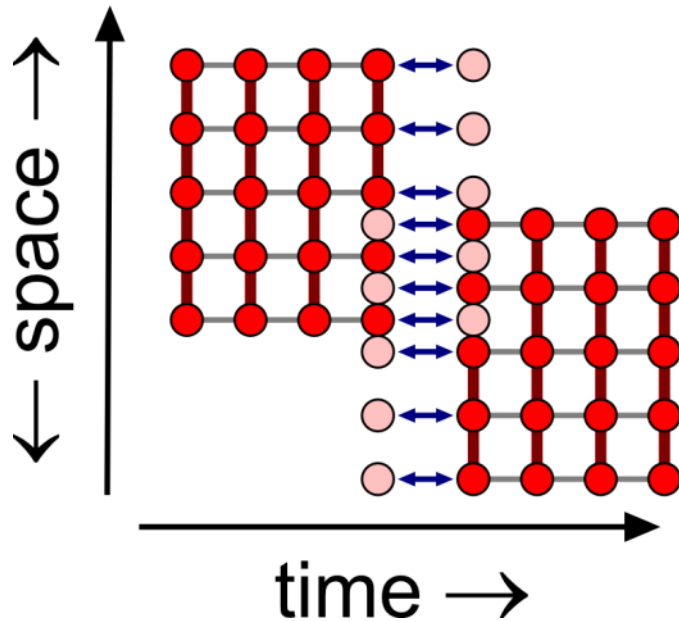
- No connection if distance changes



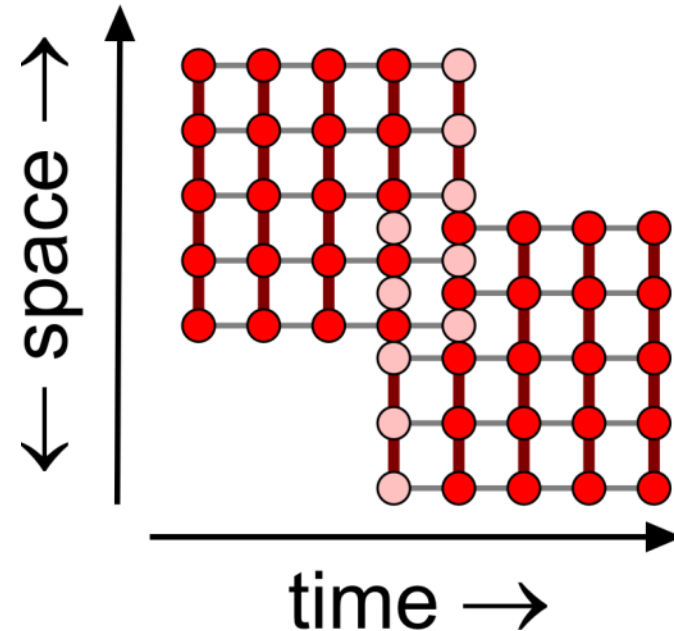
[data set courtesy of S. König, S. Gumhold, TU Dresden]

Iterative Assembly

Topology stitching



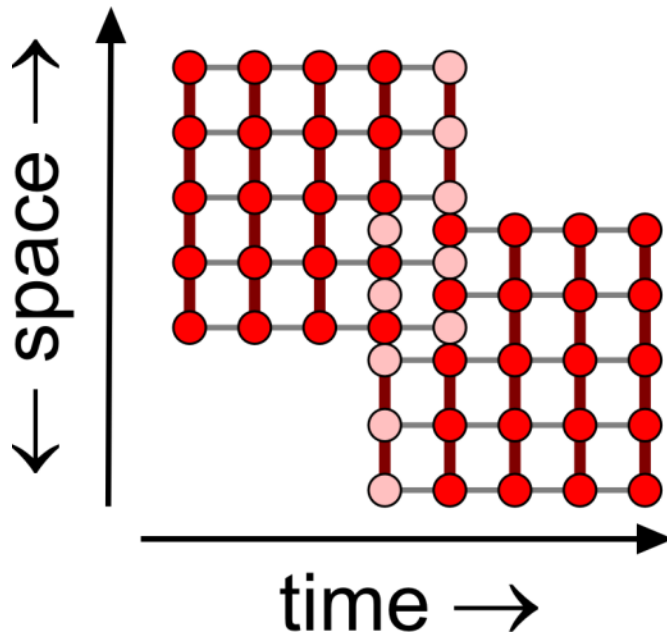
aligned
frames



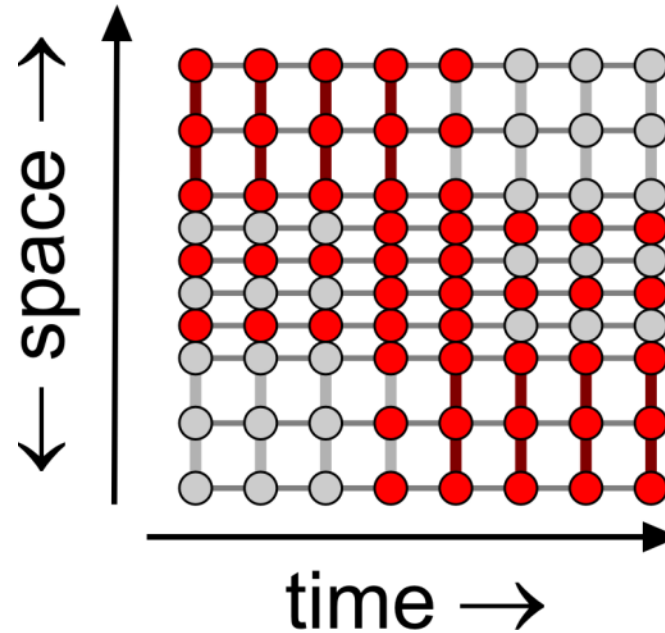
merged
topology

Iterative Assembly

Problem: incomplete trajectories



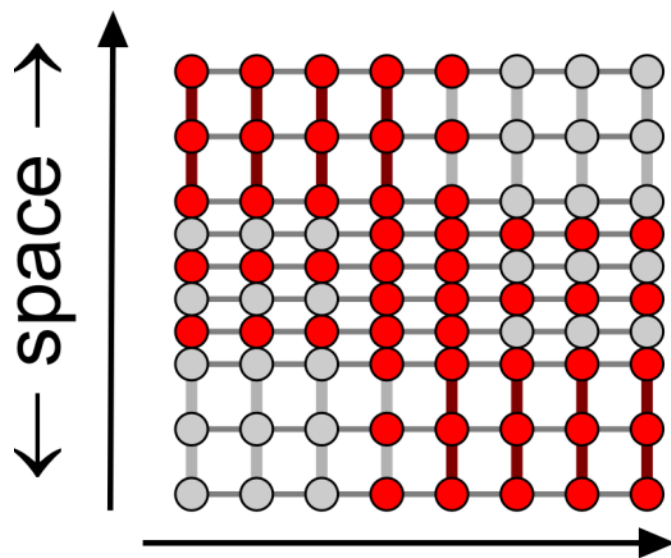
merged
topology



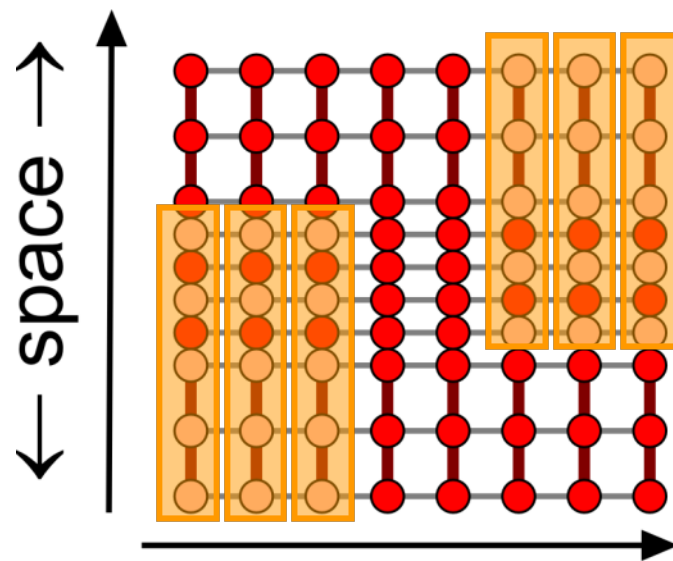
uninitialized
surfels

Iterative Assembly

Hole filling



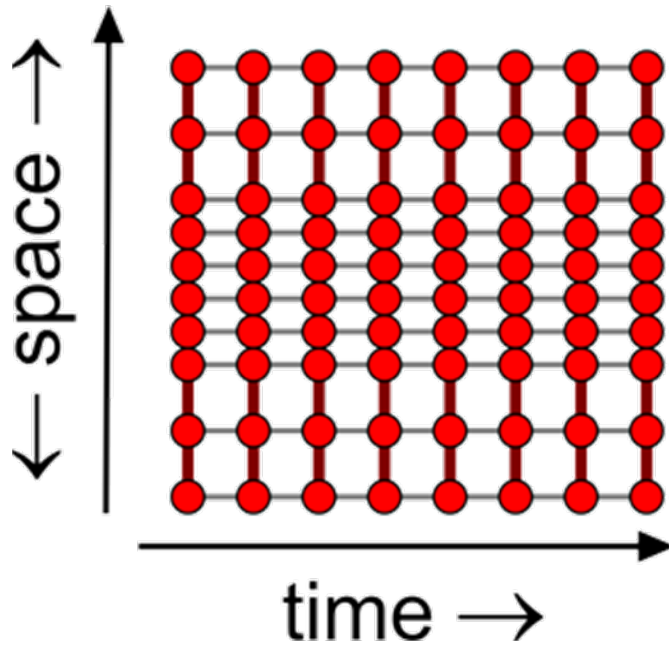
uninitialized
surfels



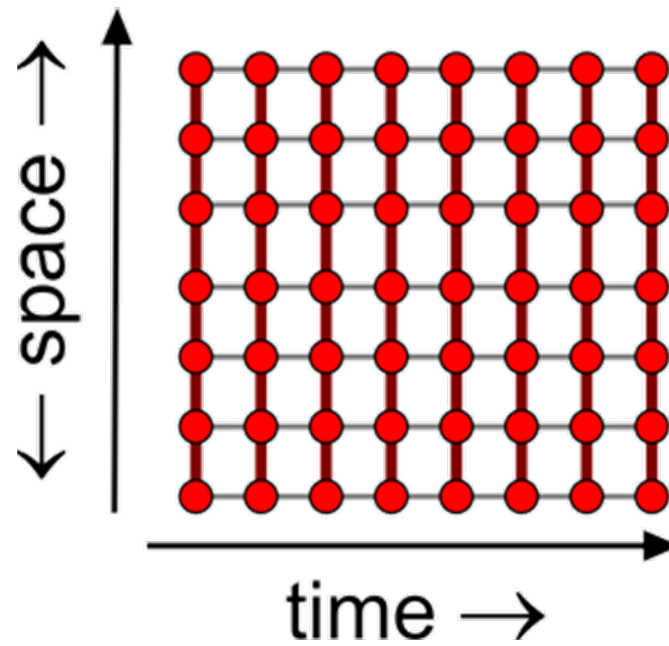
copy from neighbors,
optimize

Iterative Assembly

Resampling



hole filled
result



remove dense surfels
(constant complexity)

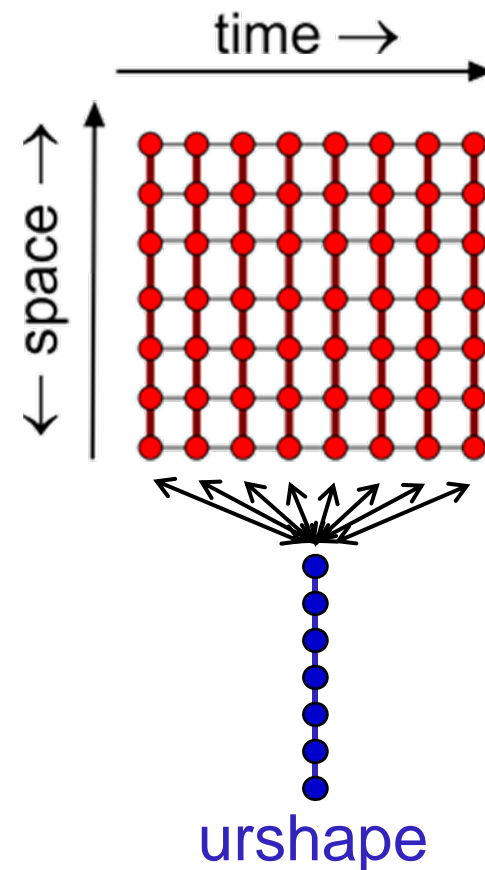
Global Optimization

Last step:

- Global optimization
- Optimize over all frames simultaneously

Improve stability: Urshapes

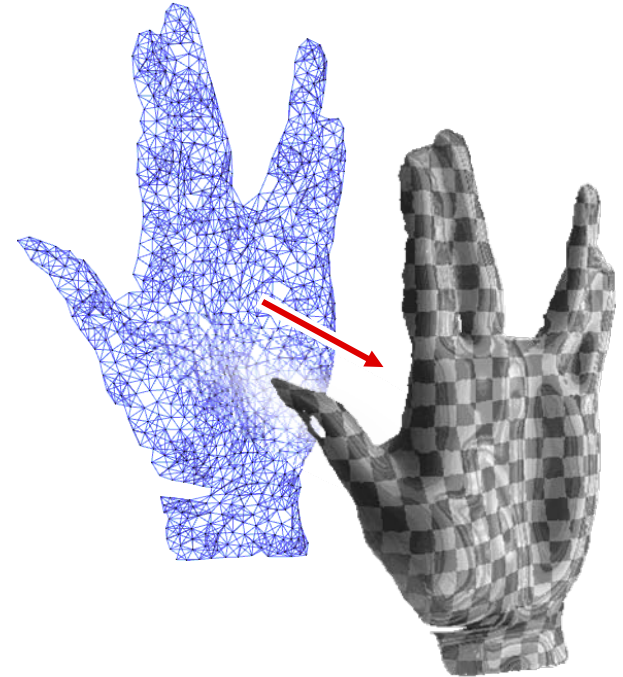
- Connect hidden “latent” frame to all other frames (deformation prior only)
- Initialize with one of the frames



Meshing

Last step: create mesh

- After complete surfel graph is reconstructed
- Pick one frame (or urshape)
- “Marching cubes” meshing
[*Hoppe et al. 92, Shen et al. 04*]
- Morph according to trajectories
(local weighted sum)



[data set courtesy of O. Schall, MPI Informatik Saarbrücken]

Results

Elephant

deformation & rotation,
noise, outliers, large holes

(synthetic data)

frames
20

surfels
49,500

data pts
963,671

preprocessing
6 min 52 sec

reconstruction
4 h 25 min

[Pentium-4, 3.4GHz]

Facial Expression

Dataset courtesy of S. Gumhold,
University of Dresden

(high speed structured light scan)

| | | | | | |
|--------|---------|----------|-----------------------------|----------------|---|
| frames | surfels | data pts | preprocessing | reconstruction | |
| 20 | 32,740 | 400,000 | 6 min 59 sec ^(*) | 7 h 31 min | [Pentium-4, 3.4GHz / ^(*) 3.0GHz] |

Improved Algorithm

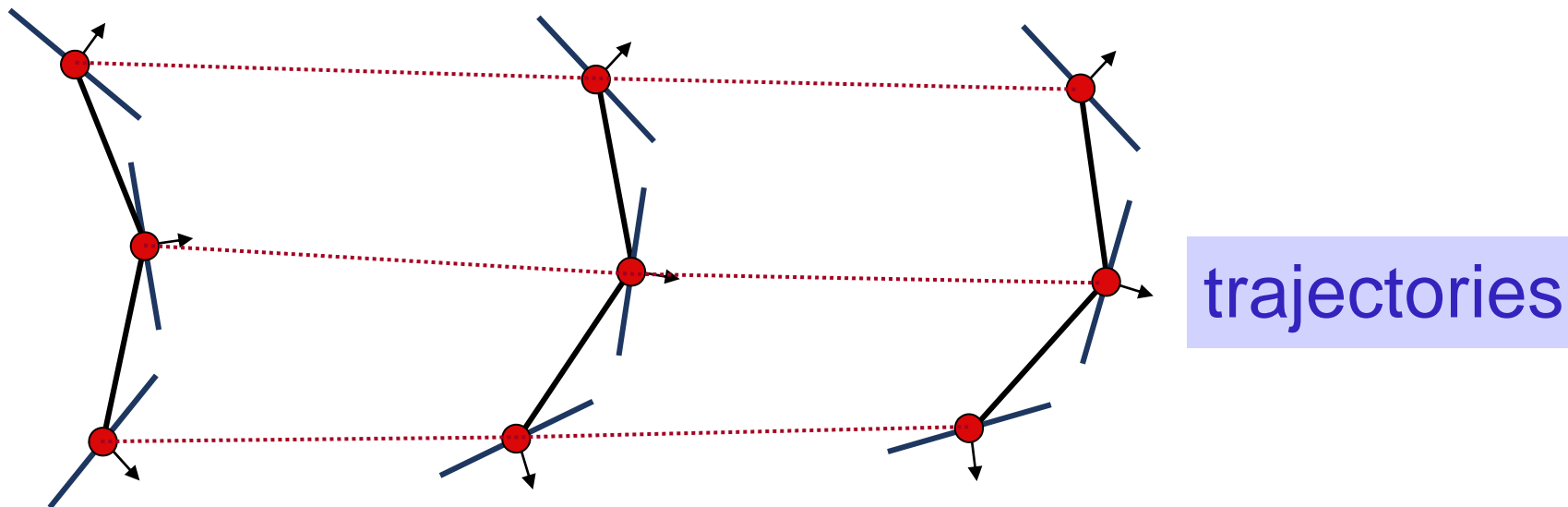
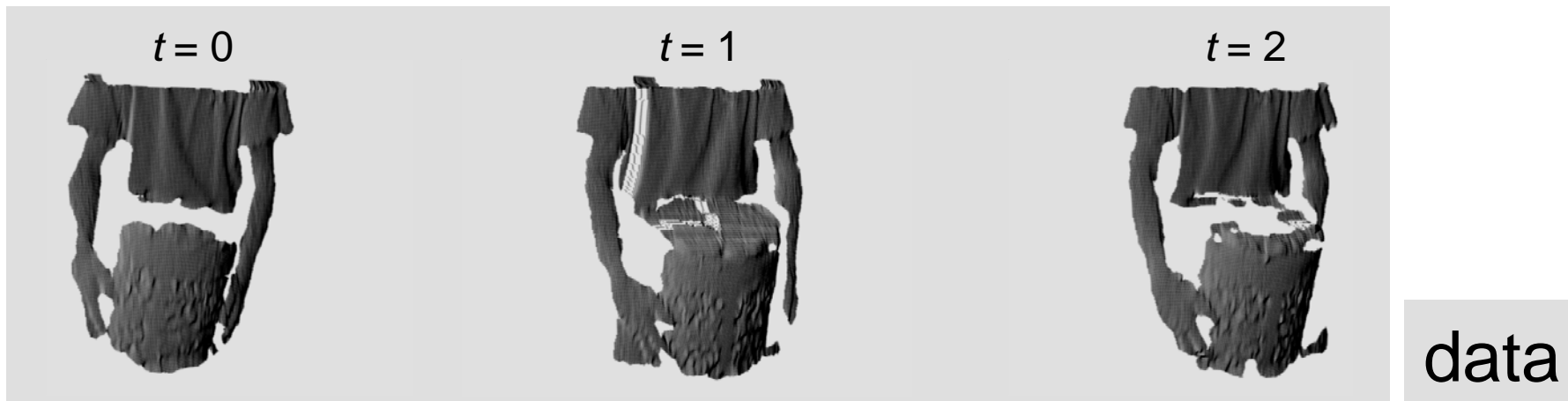
Urshape Factorization

Improved Version

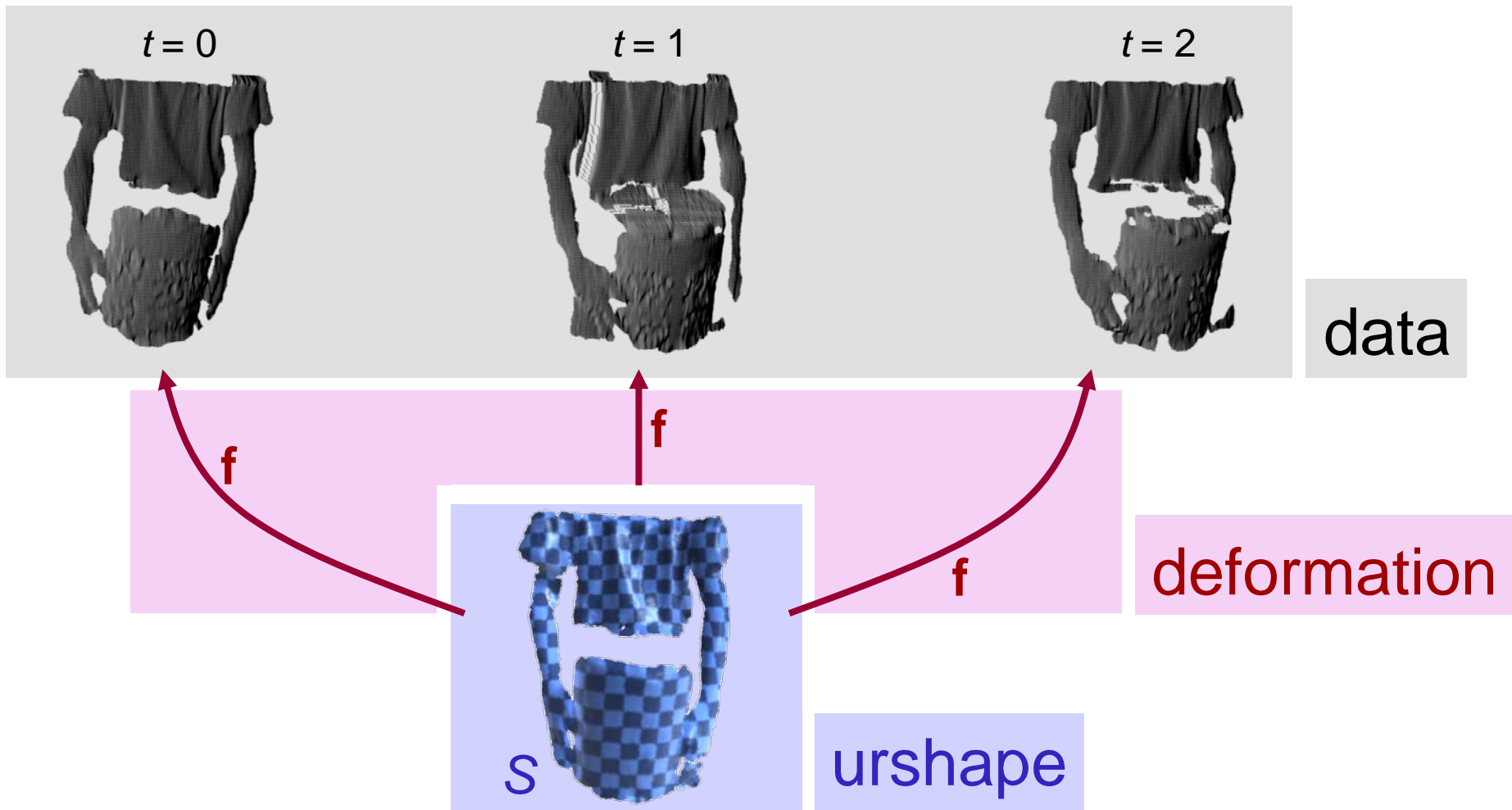
Factorization Model:

- Solving for the geometry in every frame wastes resources
- Store one urshape and a deformation field
 - High resolution geometry
 - Low resolution deformation (adaptive)
- Less memory, faster, and much more stable
- Streaming computation (constant working set)

We have so far...



New: Factorization



Components

Variational Model

- Given an initial estimate, improve *urshape* and *deformation*

Numerical Discretization

- *Shape*
- *Deformation*

Domain Assembly

- Getting an initial estimate
- *Urshape* assembly

Components

Variational Model

- Given an initial estimate, improve *urshape* and *deformation*

Numerical Discretization

- *Shape*
- *Deformation*

Domain Assembly

- Getting an initial estimate
- *Urshape* assembly

Energy Minimization

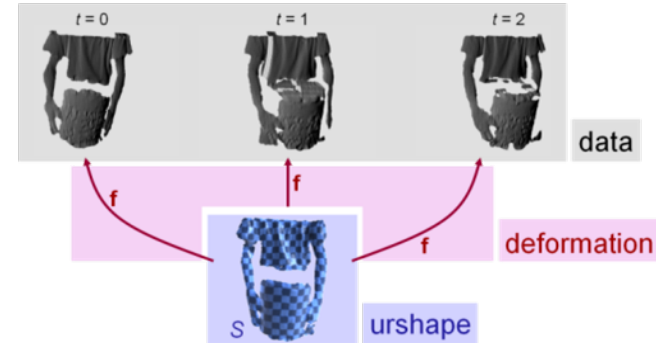
Energy Function

$$E(\mathbf{f}, S) = E_{data} + E_{deform} + E_{smooth}$$

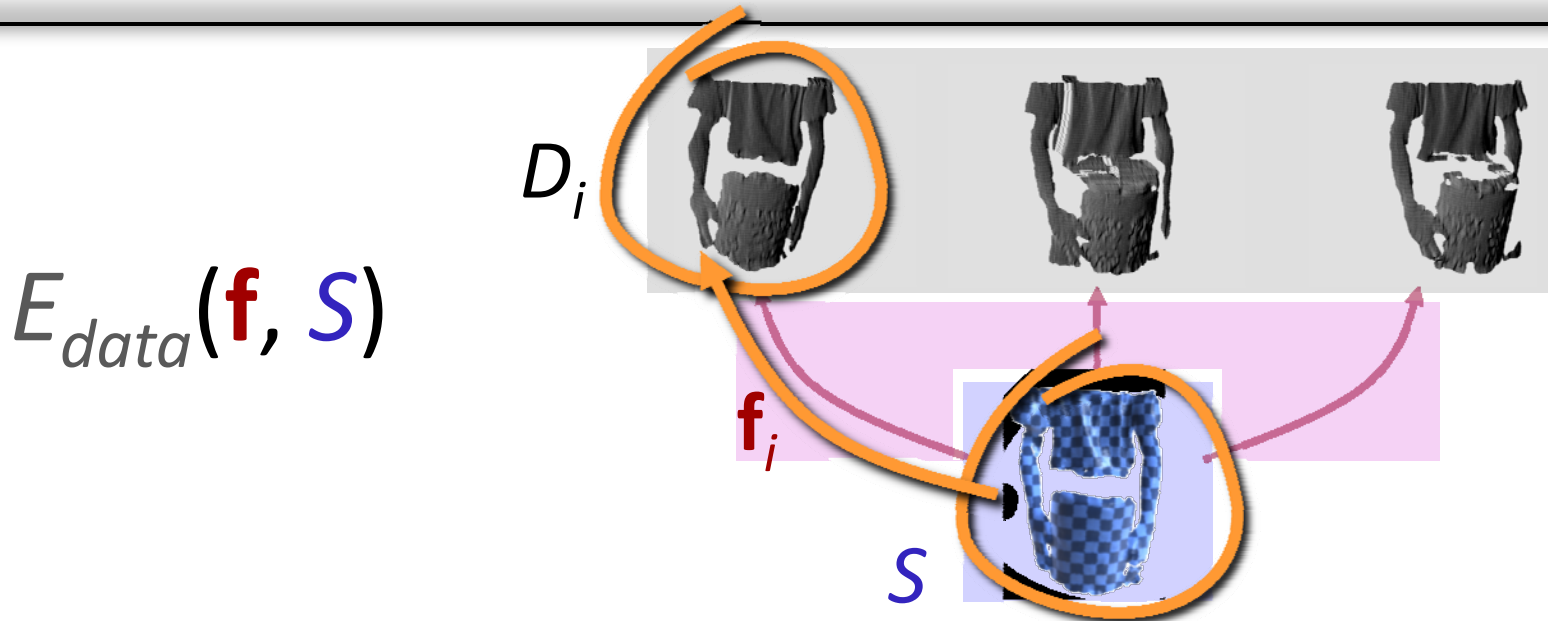
Components

- $E_{data}(\mathbf{f}, S)$ – data fitting
- $E_{deform}(\mathbf{f})$ – elastic deformation, smooth trajectory
- $E_{smooth}(S)$ – smooth surface

Optimize S , \mathbf{f} alternately

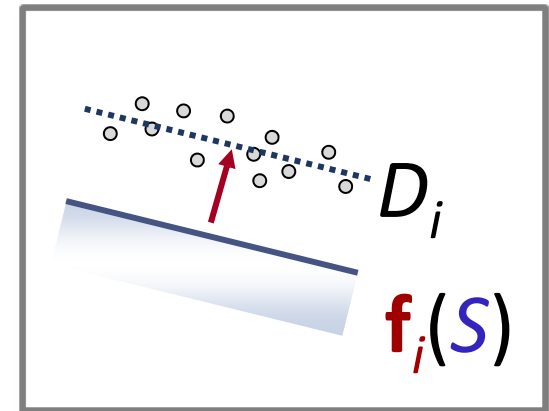


Data Fitting



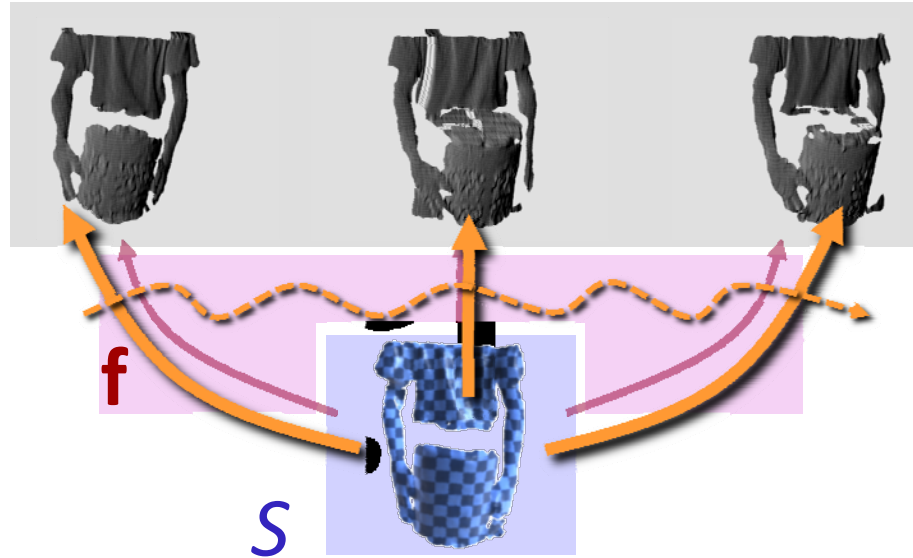
Data fitting

- Necessary: $\mathbf{f}_i(S) \approx D_i$
- Truncated squared distance function (point-to-plane)



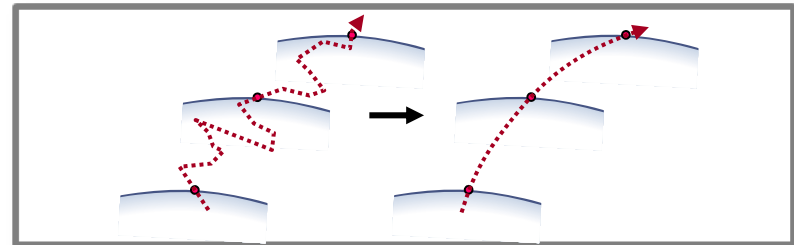
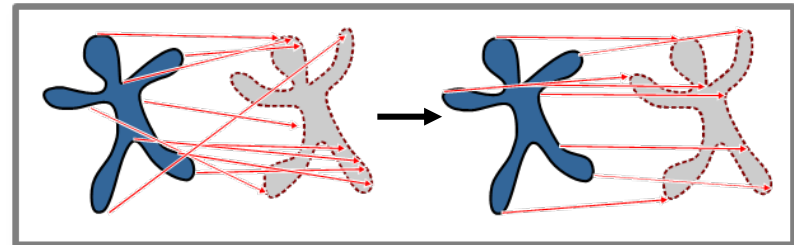
Elastic Deformation Energy

$$E_{deform}(\mathbf{f})$$

 D_i 

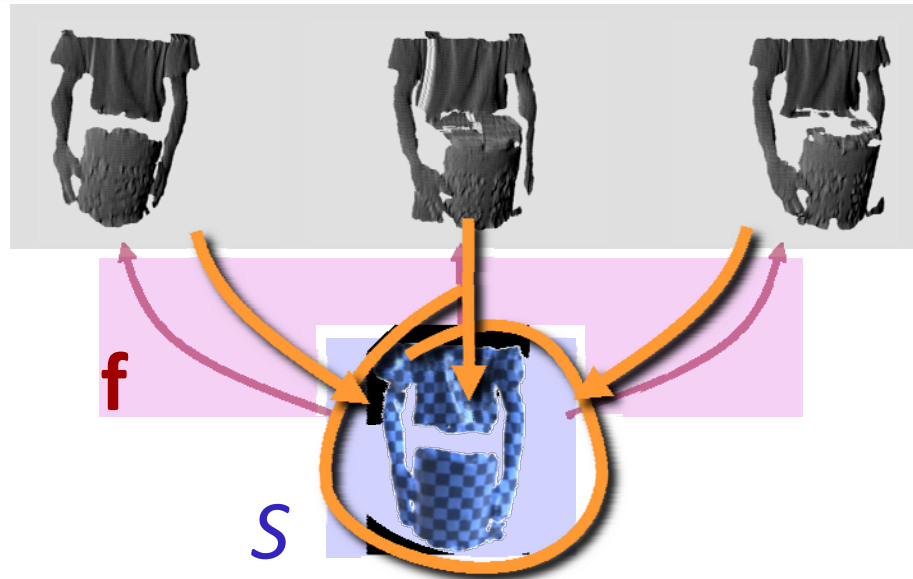
Regularization

- Elastic energy
- Smooth trajectories



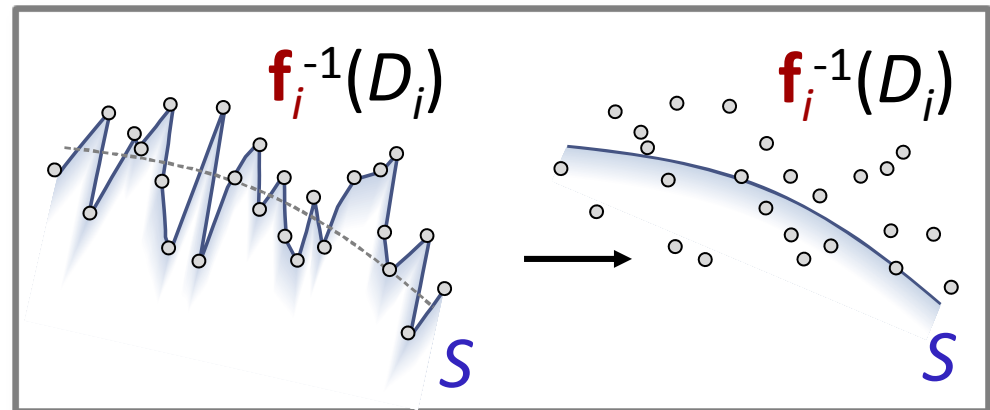
Surface Reconstruction

$$E_{smooth}(S)$$

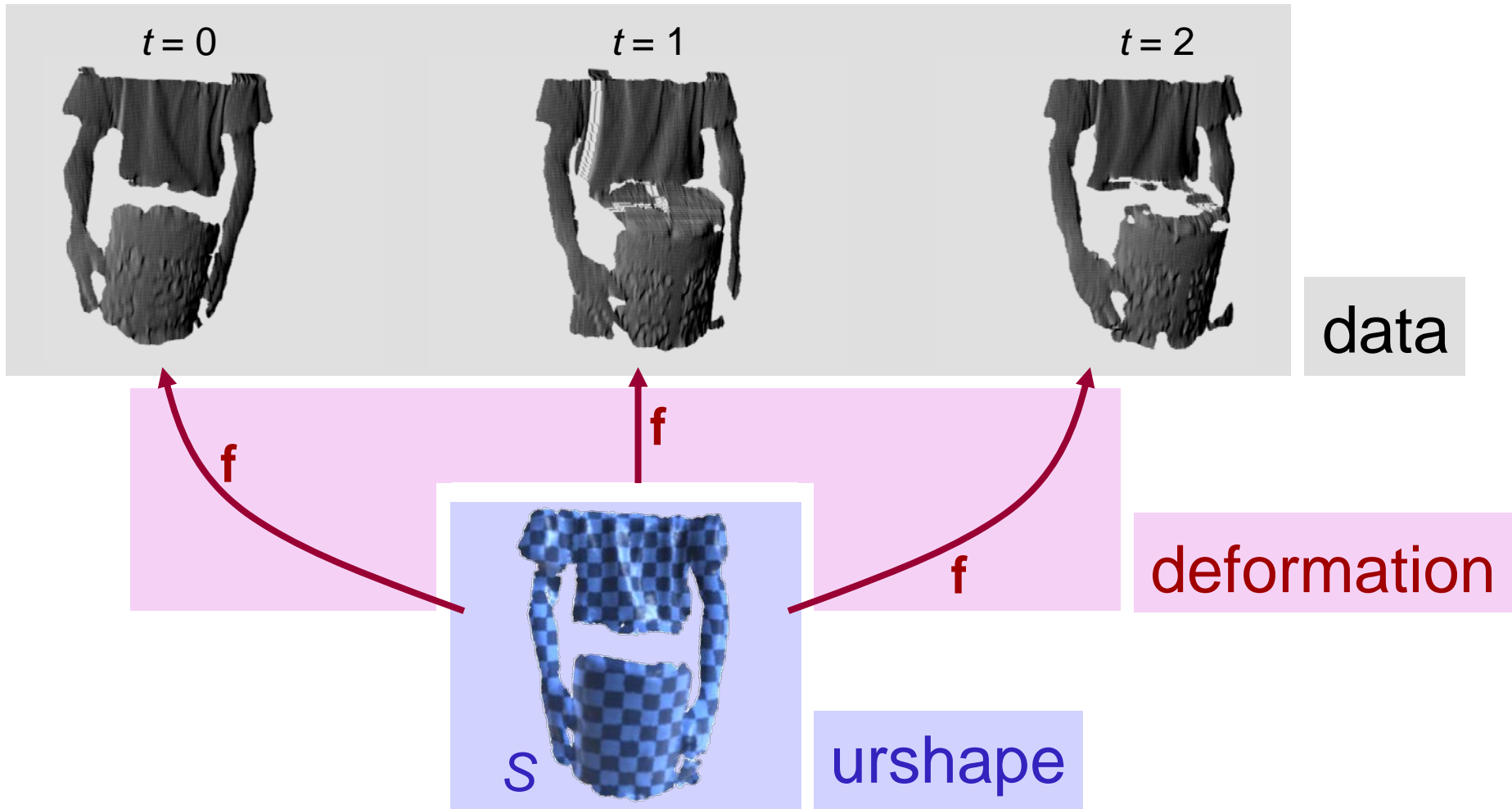
 D_i 

Data fitting

- Smooth surface
- Fitting to noisy data



Factorization



Components

Variational Model

- Given an initial estimate, improve *urshape* and *deformation*

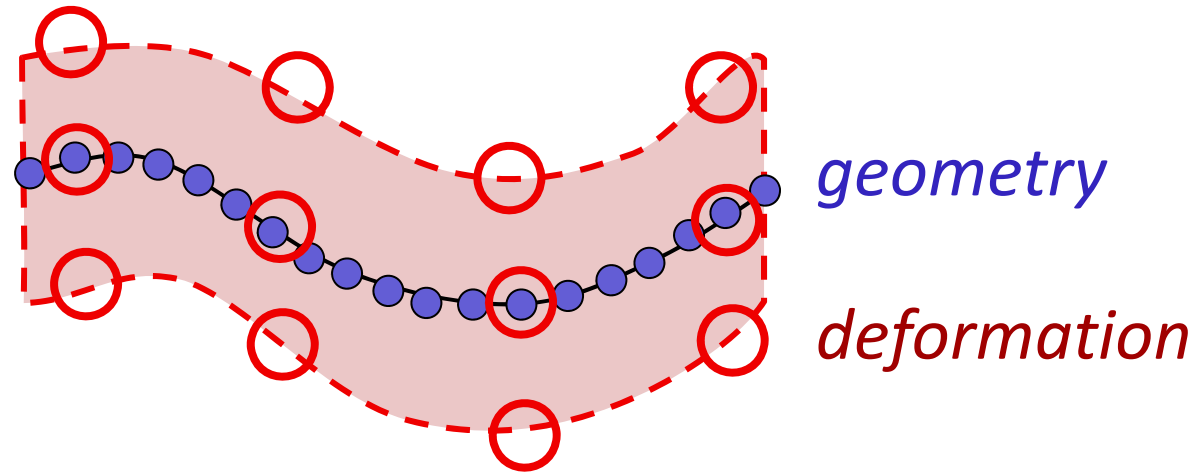
Numerical Discretization

- *Shape*
- *Deformation*

Domain Assembly

- Getting an initial estimate
- *Urshape* assembly

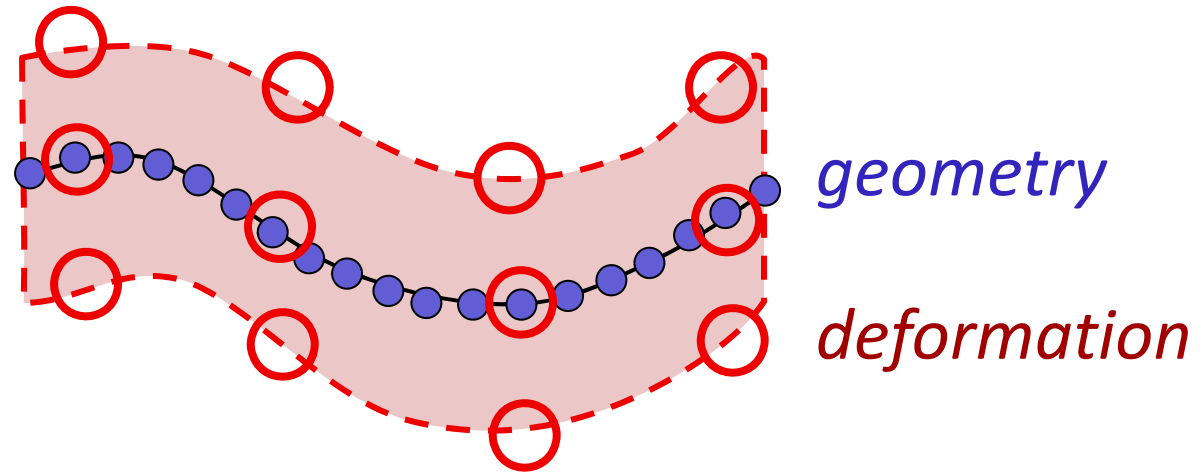
Discretization



Sampling:

- Full resolution *geometry*
- Subsample *deformation*

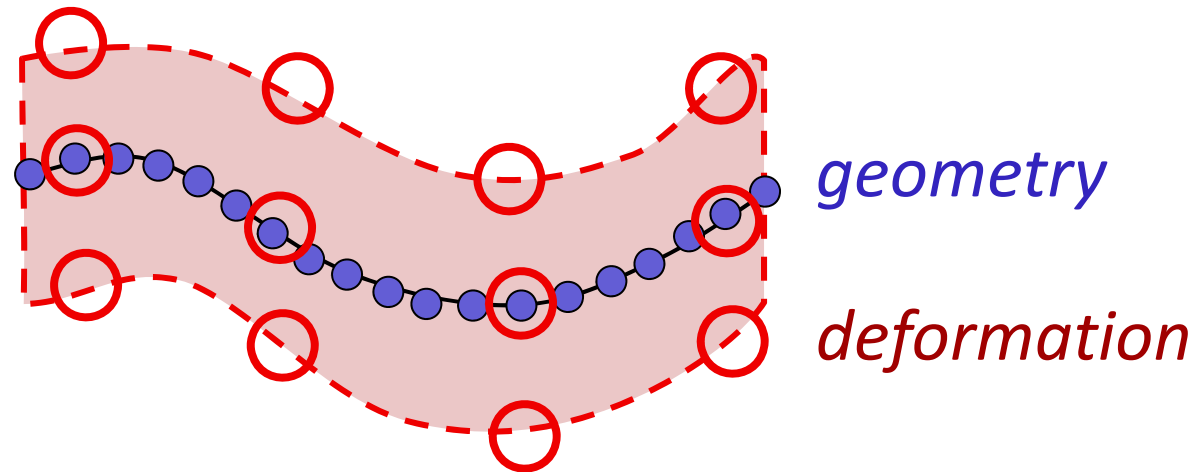
Discretization



Sampling:

- Full resolution *geometry*
 - High frequency
- Subsample *deformation*
 - Low frequency

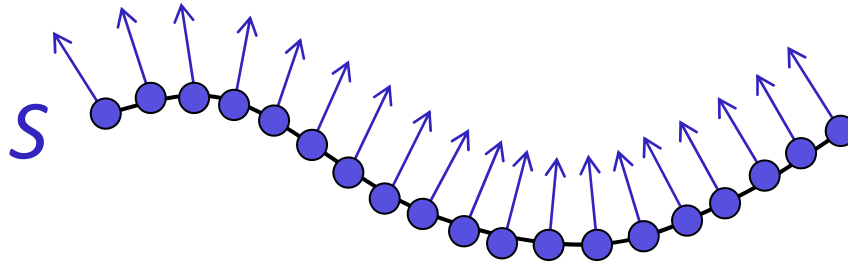
Discretization



Sampling:

- Full resolution *geometry*
 - High frequency, stored once
- Subsample *deformation*
 - Low frequency, all frames \Rightarrow more costly

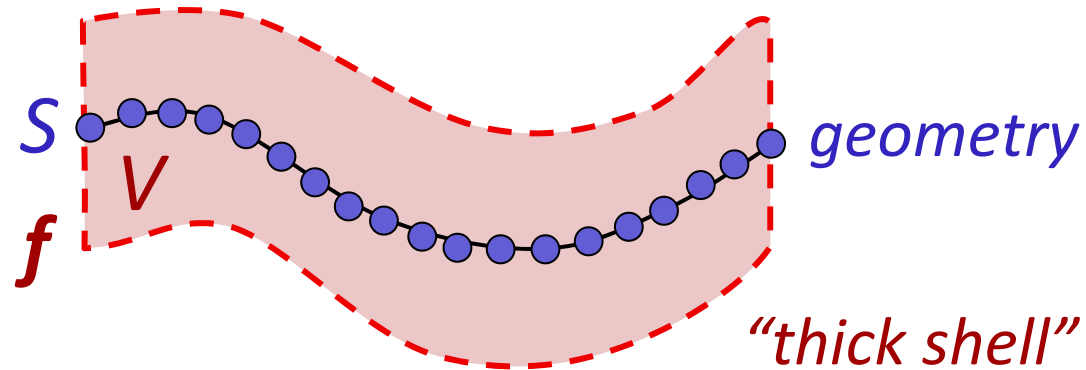
Shape Representation



Shape Representation:

- Graph of *surfels* (point + normal + local connectivity)
- E_{smooth} – neighboring planes should be similar
- Same as before...

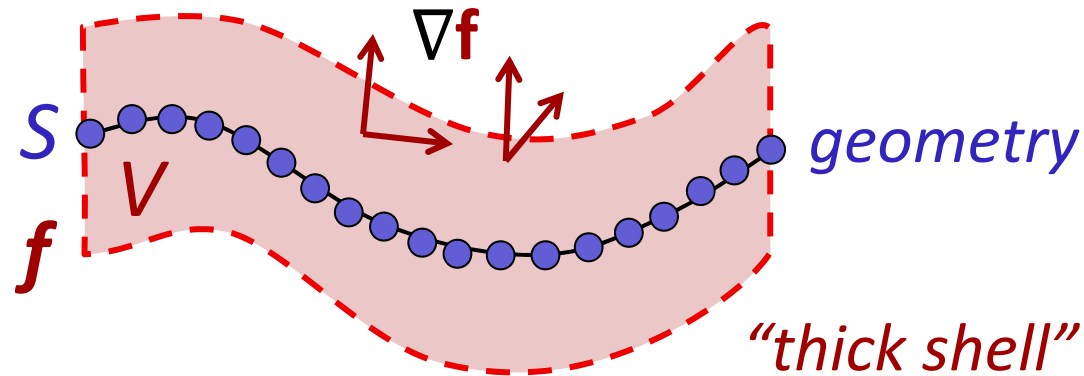
Deformation



Volumetric Deformation Model

- Surfaces embedded in “stiff” volumes
- Easier to handle than “thin-shell models”
- General – works for non-manifold data

Deformation



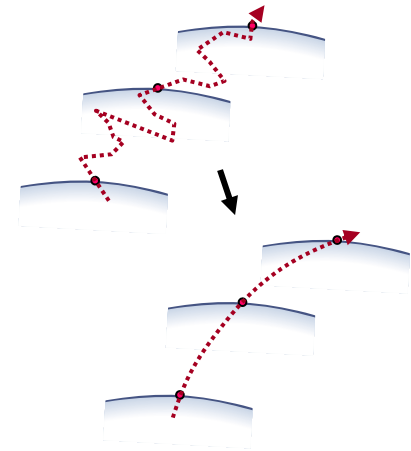
Deformation Energy

- Keep deformation gradients $\nabla \mathbf{f}$ as-rigid-as-possible
- This means: $\nabla \mathbf{f}^T \nabla \mathbf{f} = \mathbf{I}$
- Minimize: $E_{deform} = \int_T \int_V ||\nabla \mathbf{f}(\mathbf{x}, t)^T \nabla \mathbf{f}(\mathbf{x}, t) - \mathbf{I}||^2 d\mathbf{x} dt$

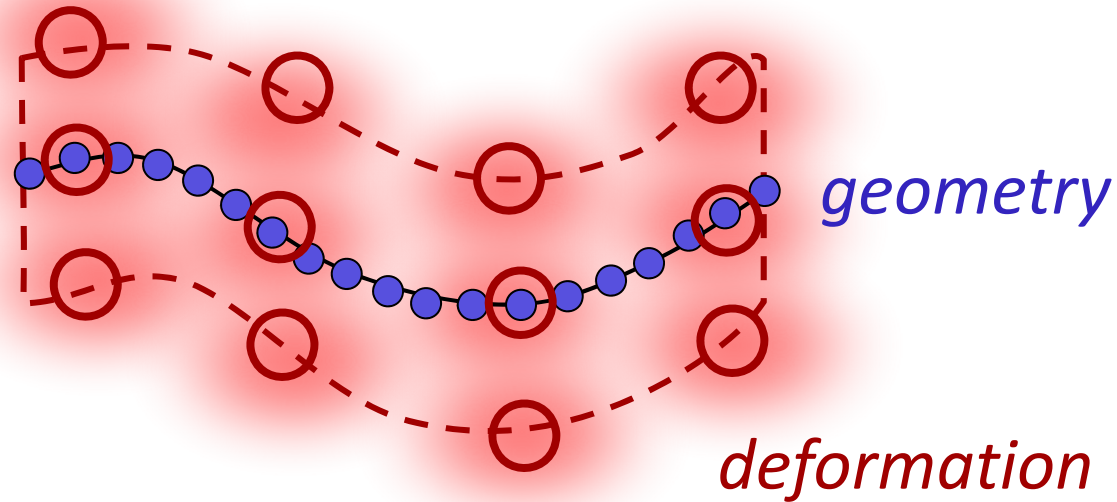
Additional Terms

More Regularization

- Volume preservation: $E_{vol} = \int_T \int_V ||\det(\nabla \mathbf{f}) - 1||^2$
 - Stability
- Acceleration: $E_{acc} = \int_T \int_V ||\partial_t^2 \mathbf{f}||^2$
 - Smooth trajectories
- Velocity (weak): $E_{vel} = \int_T \int_V ||\partial_t \mathbf{f}||^2$
 - Damping



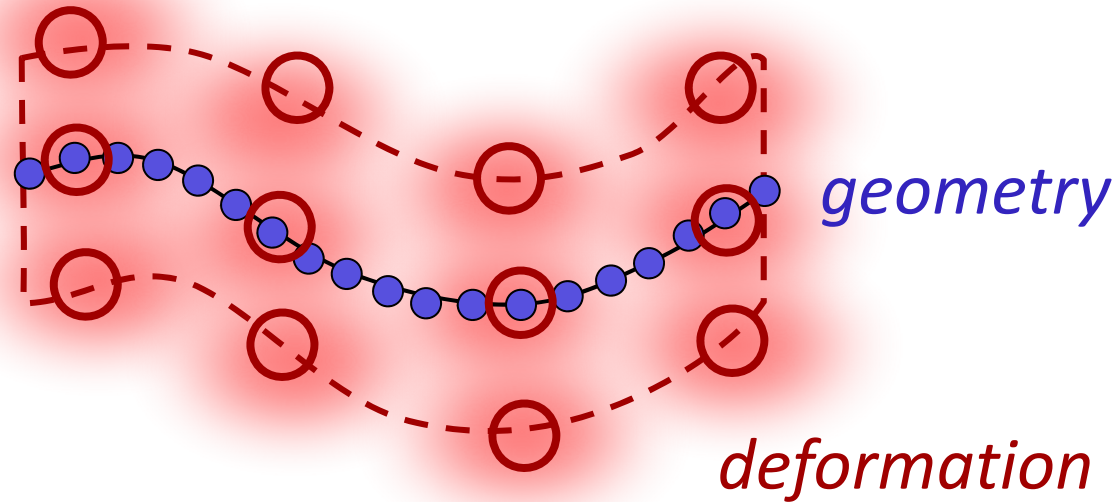
Discretization



How to represent the deformation?

- Goal: efficiency
- Finite basis:
As few basis functions as possible

Discretization



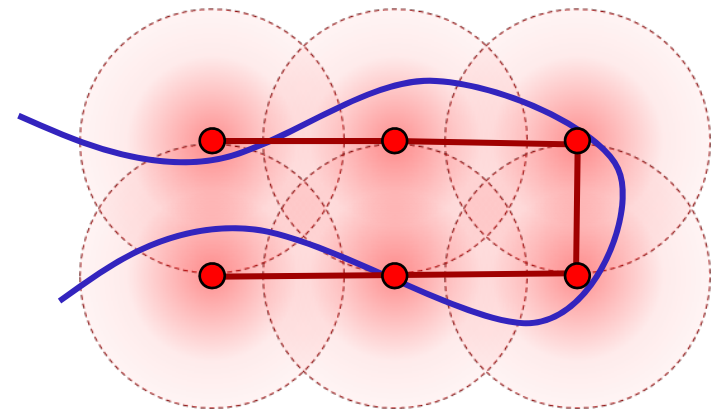
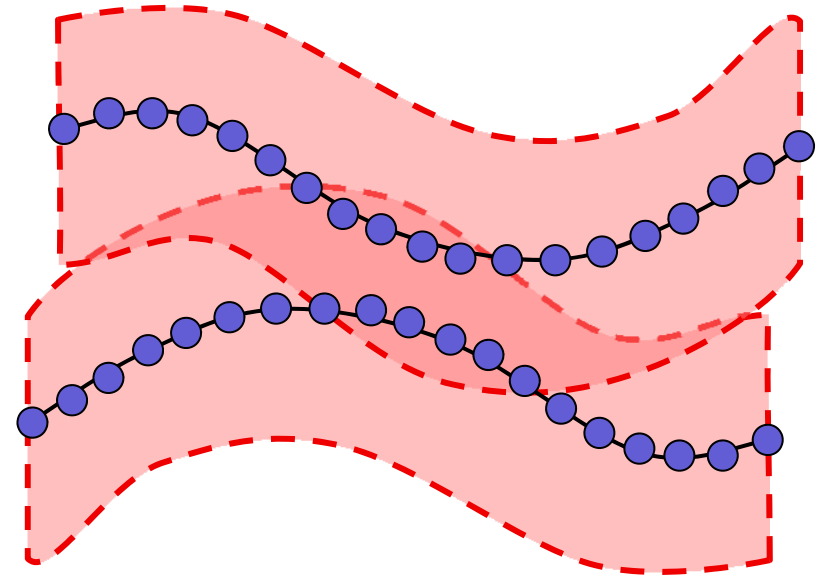
Meshless finite elements

- Partition of unity, smoothness
- Linear precision
- Adaptive sampling is easy

Meshless Finite Elements

Topology:

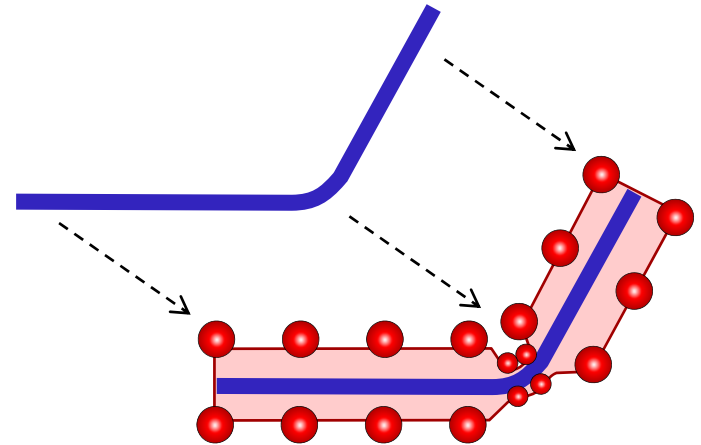
- Separate deformation nodes for disconnected pieces
- Need to ensure
 - Consistency
 - Continuity
- Euclidean / intrinsic distance-based coupling rule
 - See references for details



Adaptive Sampling

Adaptive Sampling

- Bending areas
 - Decrease rigidity
 - Decrease thickness
 - Increase sampling density
- Detecting bending areas:
residuals over many frames



Summary: Variational Model

$$E(S, \mathbf{f}, d) = \underbrace{E_{match}(S, \mathbf{f}, d)}_{\text{data}} + \underbrace{(E_{rigid} + E_{volume} + E_{accel} + E_{velocity})}_{\text{deformation}}(S, \mathbf{f}) + \underbrace{E_{smooth}(S)}_{\text{urshape}}$$

$$E_{match}(S, f, d) = \sum_{t=1}^T \sum_{i=1}^{n_t} \text{trunc}(\text{dist}(d_i, f(S)))^2$$

$$E_{rigid}(S, \mathbf{f}) = \int_{V(S)} \omega_{rigid}(x) \left\| \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}, t)^T \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}, t) - \mathbf{I} \right\|_F^2 dx$$

$$E_{volume}(S, \mathbf{f}) = \int_{V(S)} \omega_{vol}(x) \left(\left| \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x}, t) \right| - 1 \right)^2 dx$$

$$E_{accel}(S, \mathbf{f}) = \int_S \omega_{acc}(x) \left(\frac{\partial^2}{\partial t^2} \mathbf{f}(\mathbf{x}, t) \right)^2 dx \quad E_{velocity}(S, \mathbf{f}) = \int_S \omega_{velocity}(x) \left(\frac{\partial}{\partial t} \mathbf{f}(\mathbf{x}, t) \right)^2 dx$$

$$E_{smooth}(S) = \int_S \omega_{smooth}(x) \left(\nabla_{uv}^2 s(x) \right)^2 dx$$

Components

Variational Model

- Given an initial estimate, improve *urshape* and *deformation*

Numerical Discretization

- *Deformation*
- *Shape*

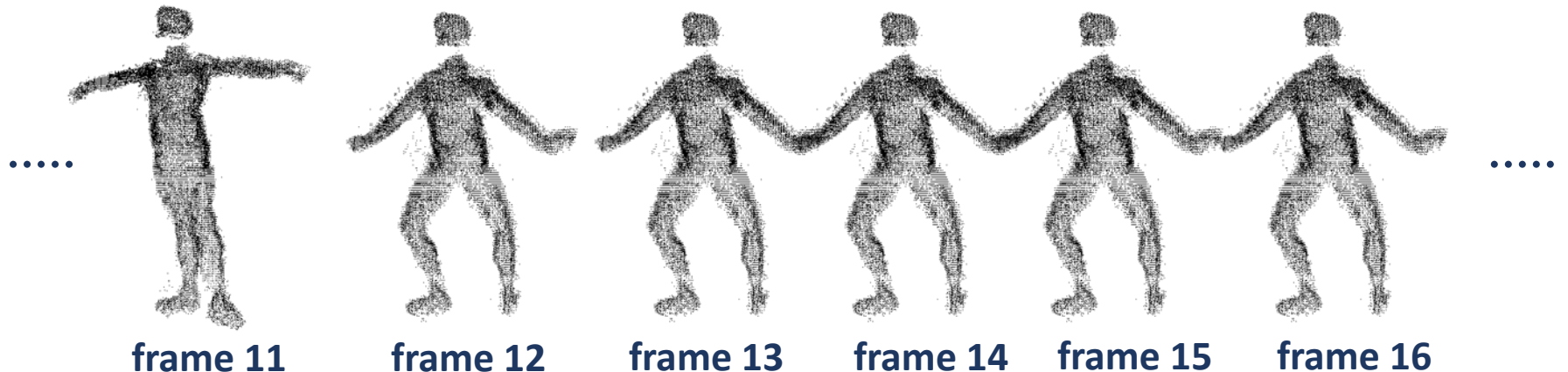
Domain Assembly

- Getting an initial estimate
- *Urshape* assembly

Urshape Assembly

Adjacent frames are similar

- Solve for frame pairs first
- Assemble urshape step-by-step



[data set courtesy of C. Theobald, MPC-VCC]

Hierarchical Merging

data

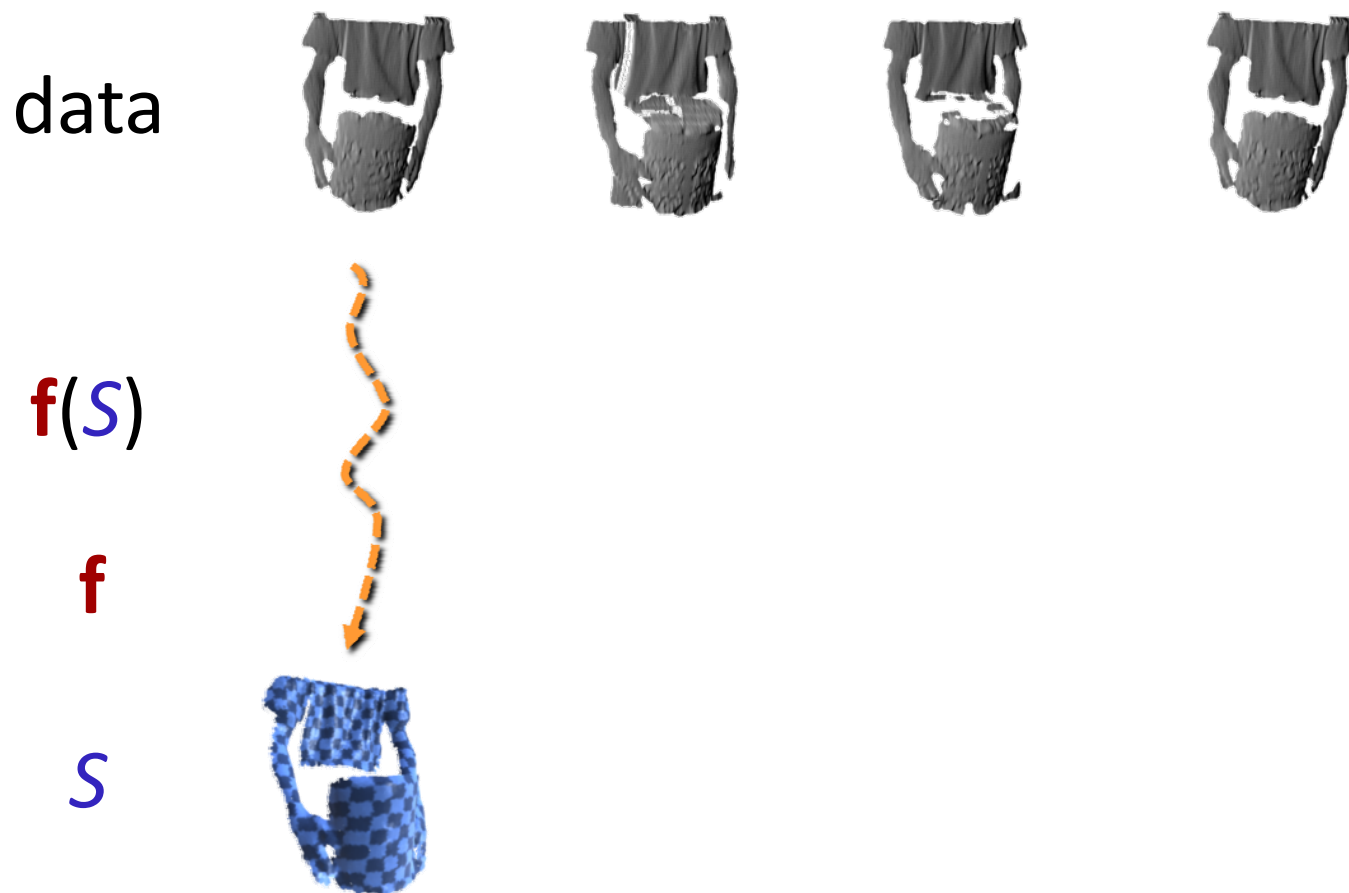


$f(S)$

f

S

Hierarchical Merging



Initial Urshapes

data



$f(S)$



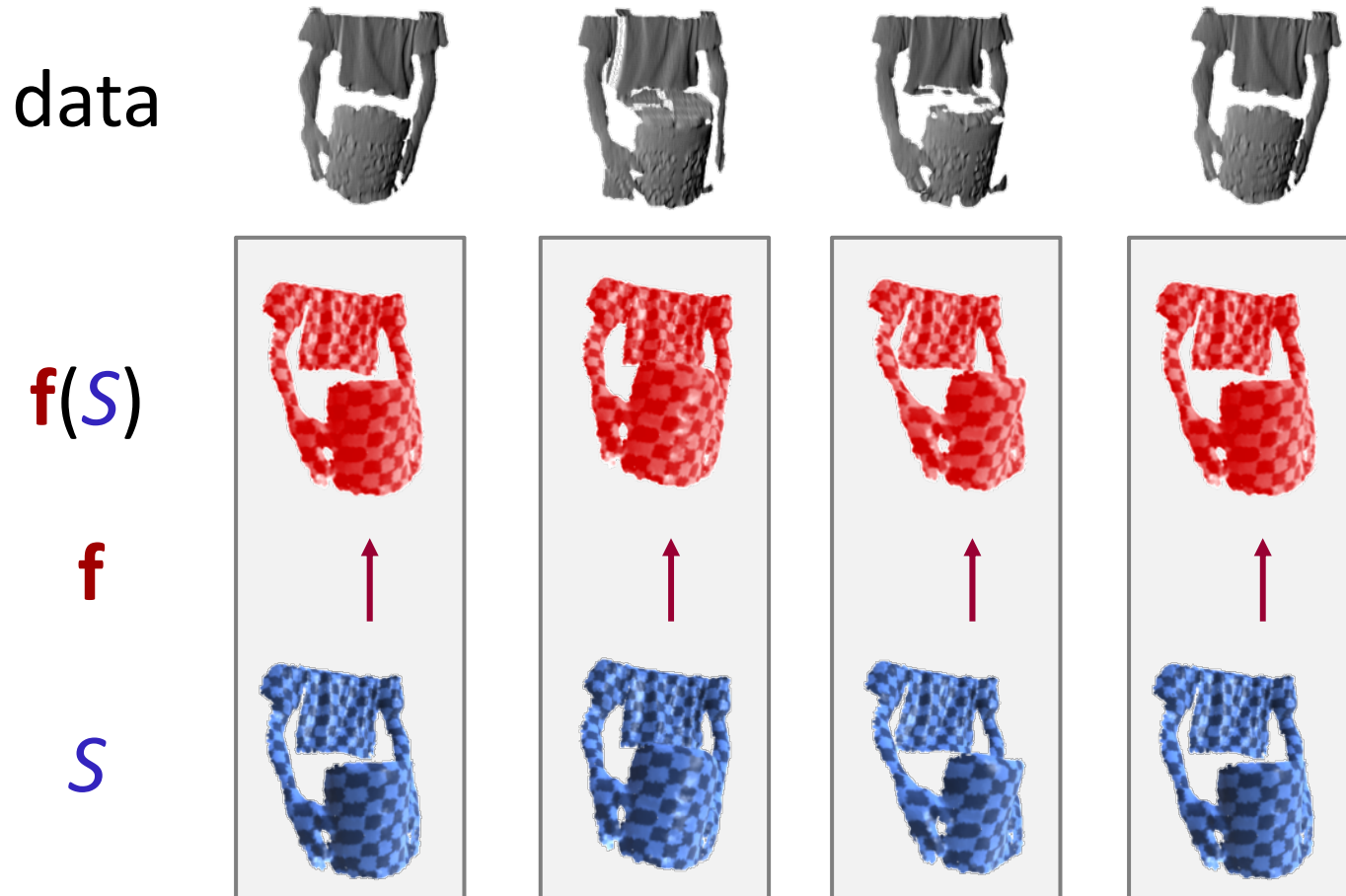
f



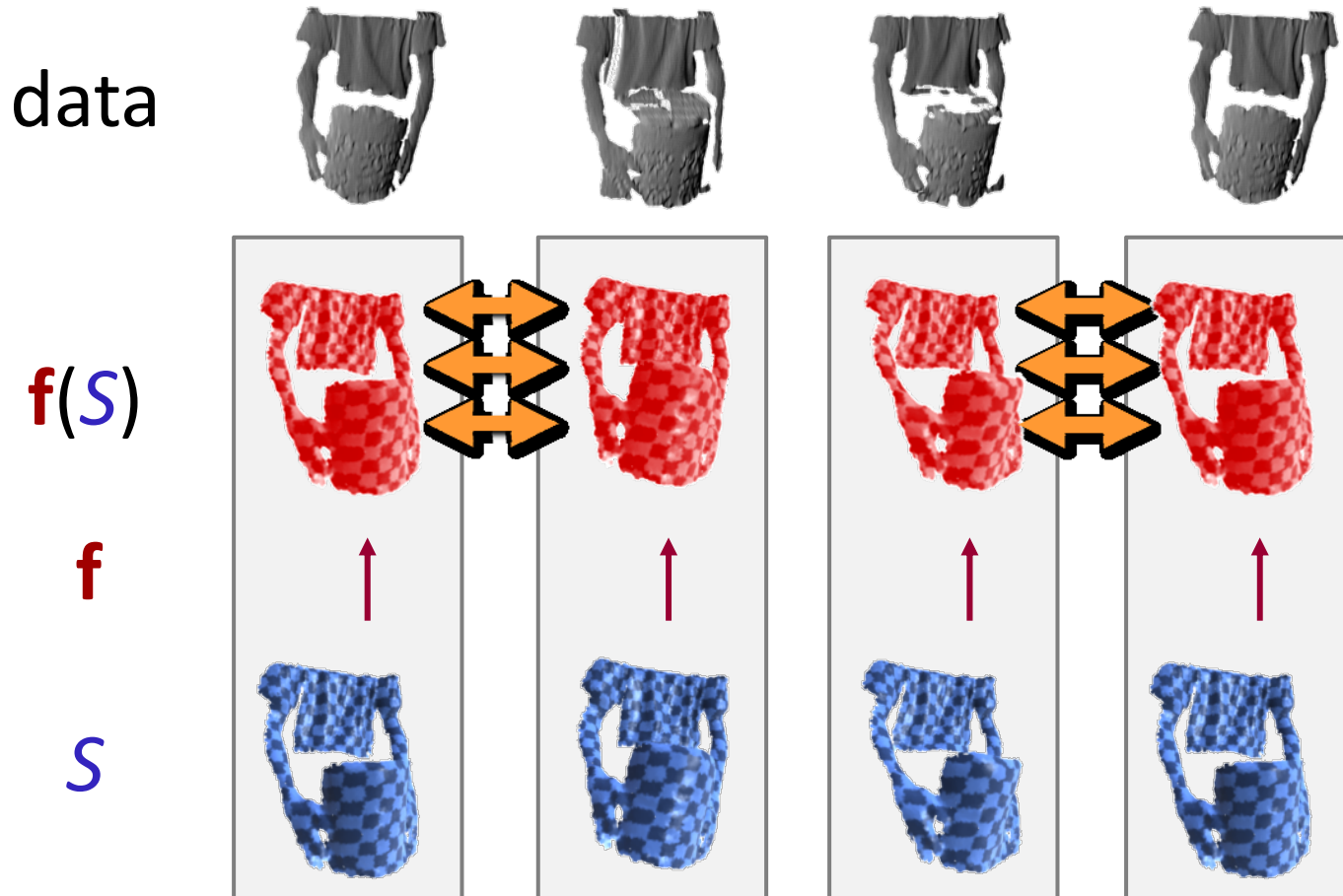
S



Initial Urshapes

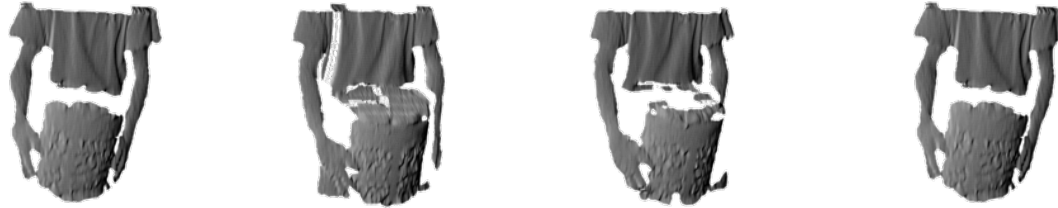


Alignment

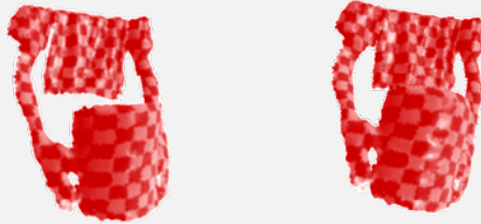


Align & Optimize

data



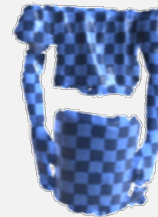
$f(S)$



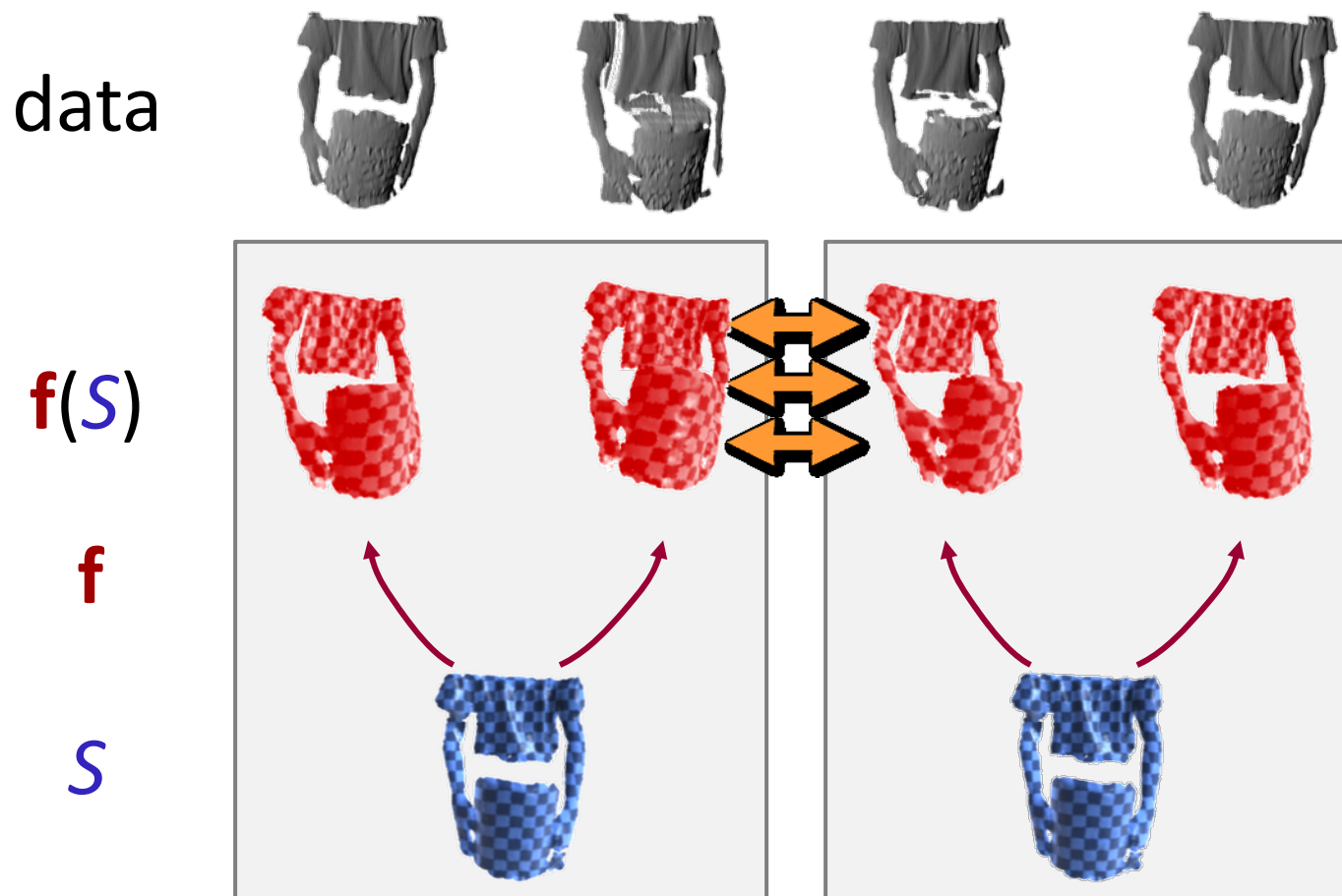
f



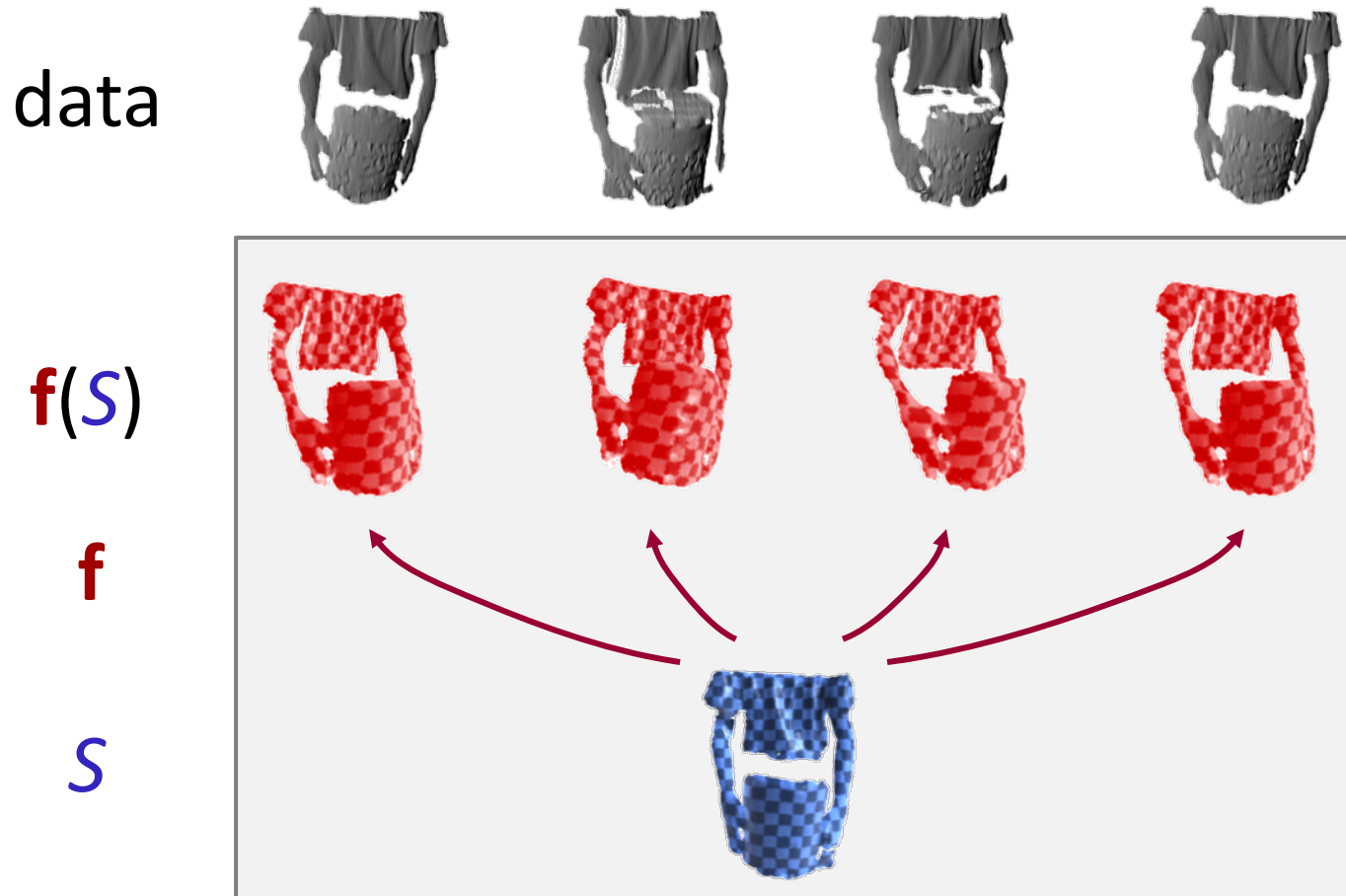
S



Hierarchical Alignment



Hierarchical Alignment



Results



79 frames, 24M data pts, 21K surfels, 315 nodes



98 frames, 5M data pts, 6.4K surfels, 423 nodes



*120 frames,
30M data pts,
17K surfels,
1,939 nodes*



*34 frames,
4M data pts,
23K surfels,
414 nodes*

Quality Improvement



old version



new result



old version



new result

Practical Animation Reconstruction



ETH Zurich / EPFL

Practical Animation Reconstruction

Hao Li



ETH Zurich / EPFL



Digitizing **Dynamic** Objects



Digitizing **Dynamic** Objects



Real-Time 3D Scanner



Real-Time 3D Scanner



Geometry and Motion Reconstruction



Input 3D Scan Sequence

Space-time Reconstruction

Geometry and Motion Reconstruction



Input 3D Scan Sequence



Space-time Reconstruction



Geometry and Motion Reconstruction



Input 3D Scan Sequence



Space-time Reconstruction



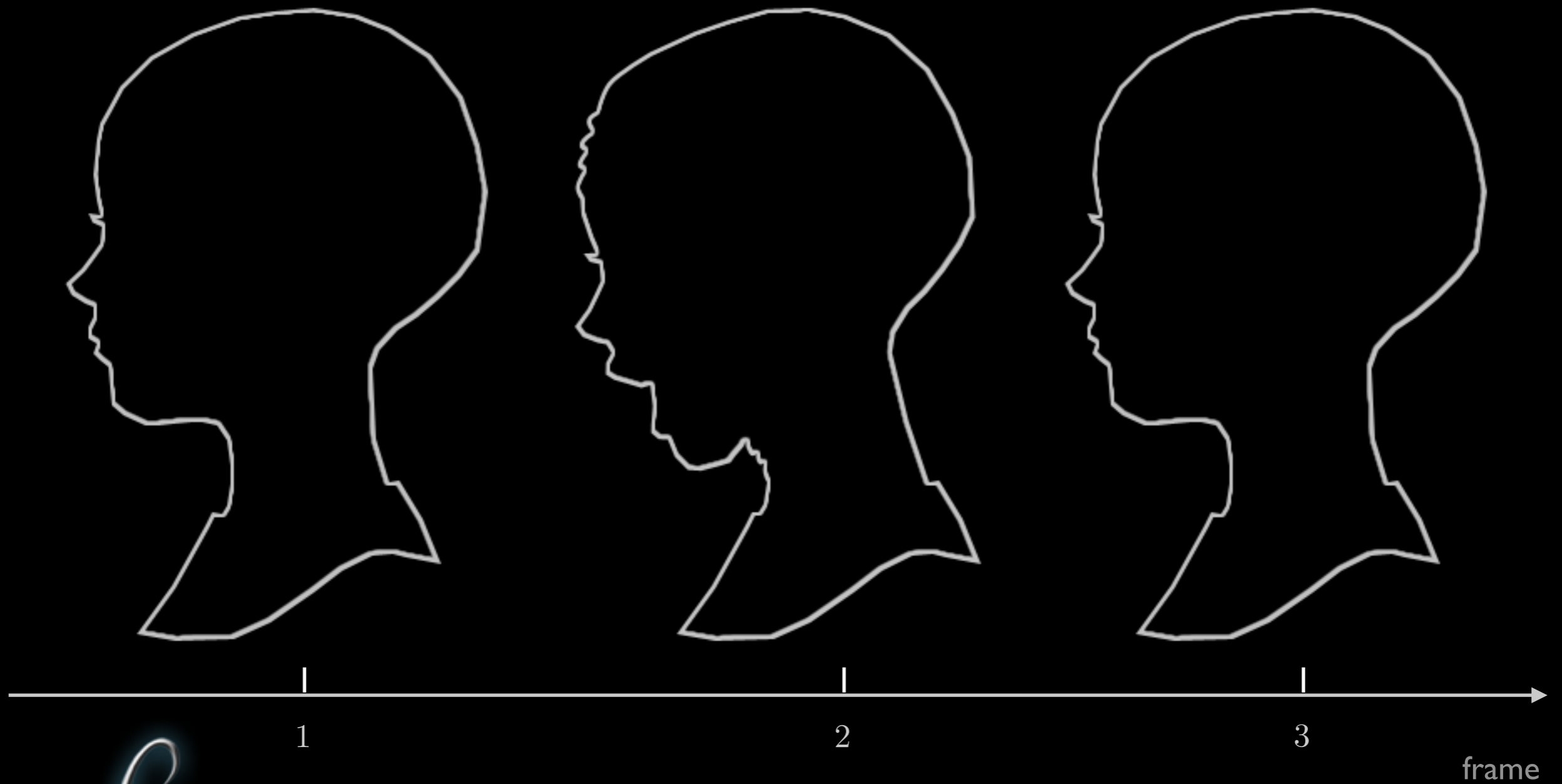
State of the Art



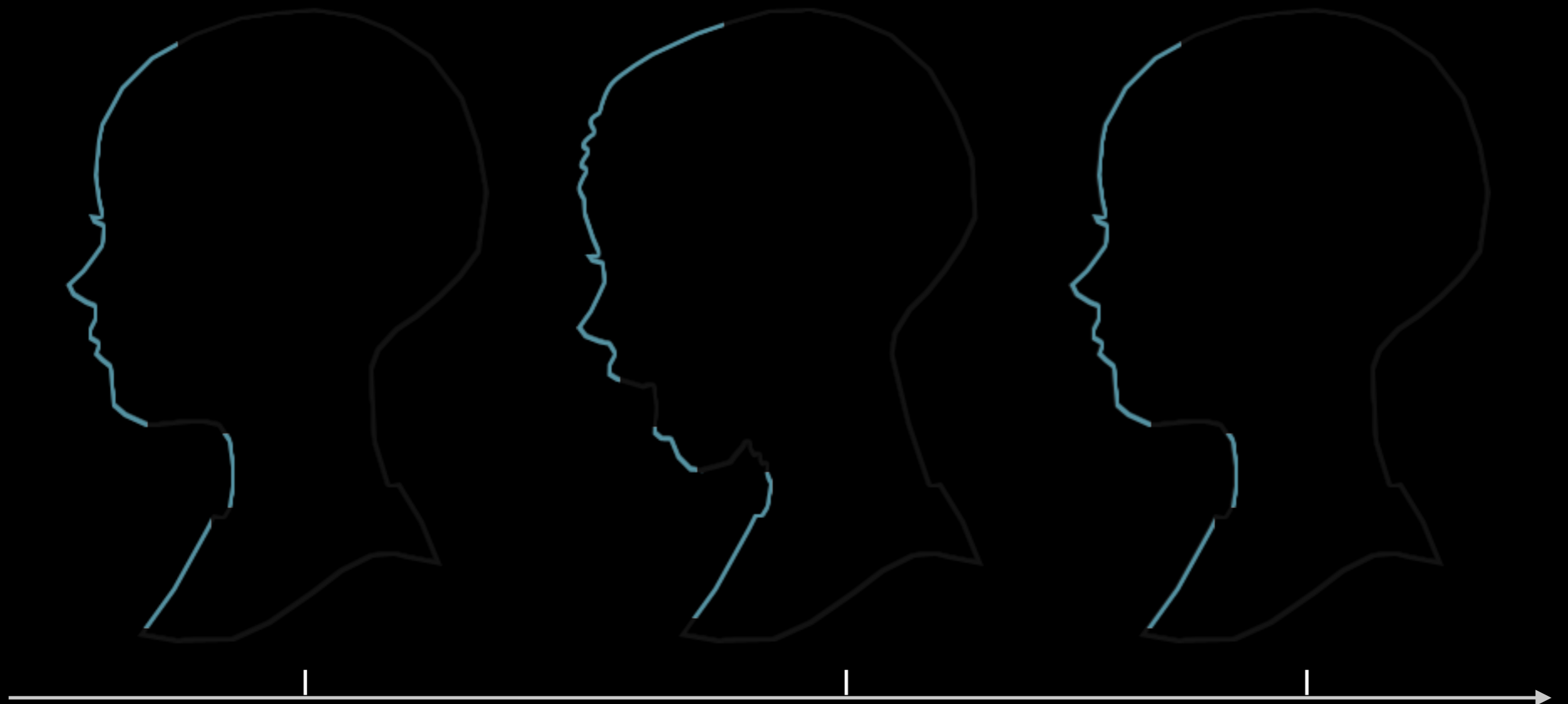
Bi-Resolution Approach



Deforming Subject



Partial Scans



1

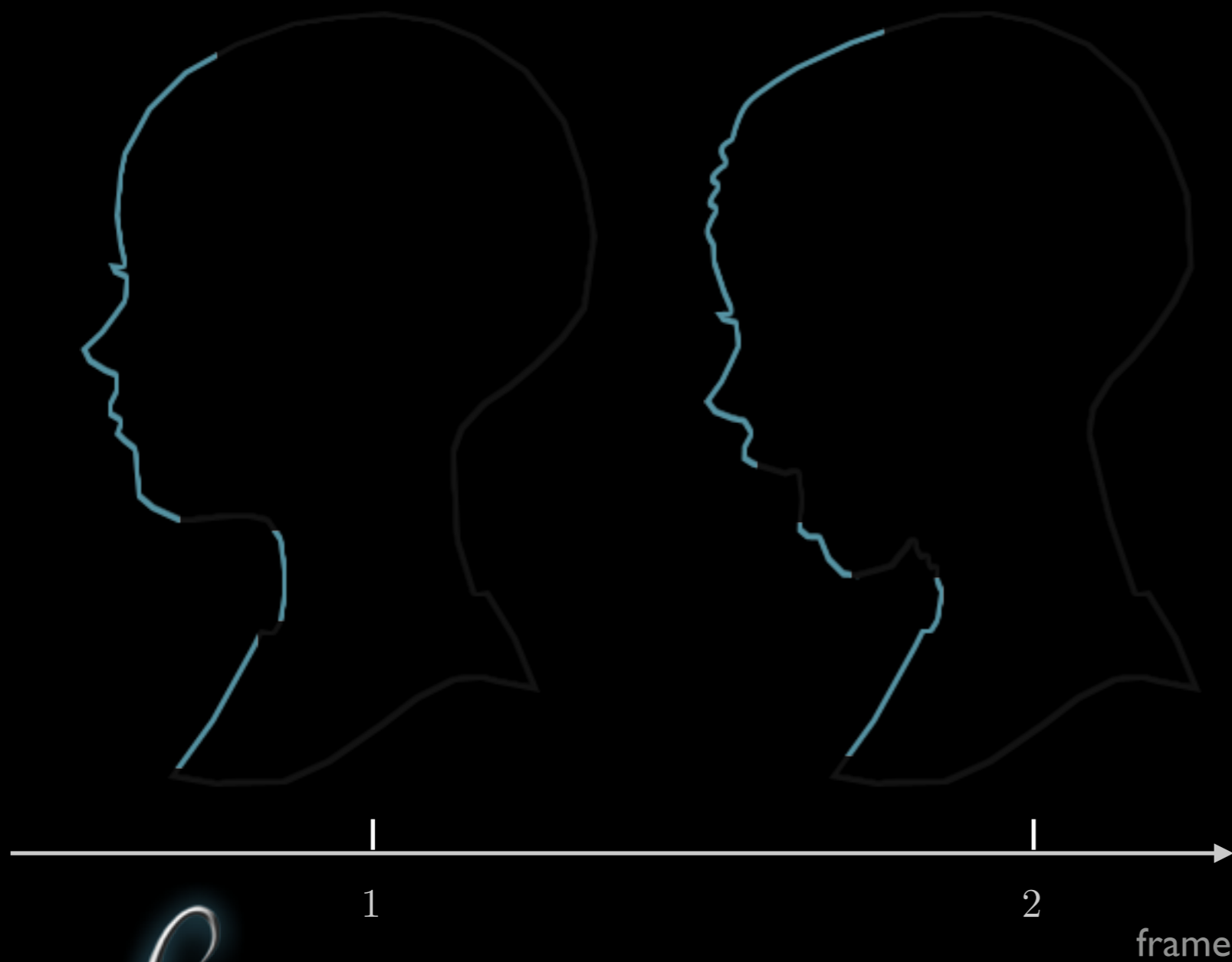
2

3

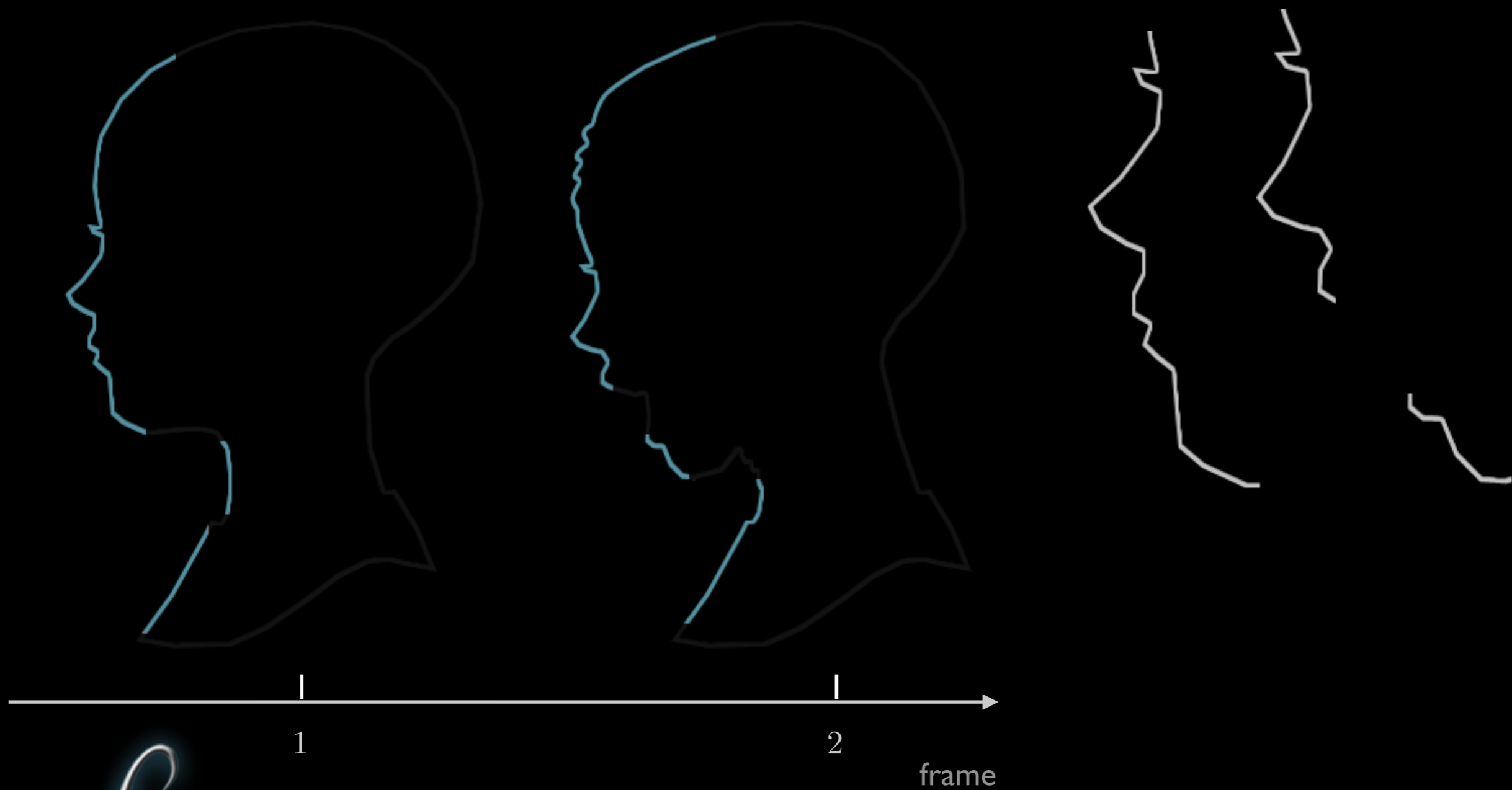
frame



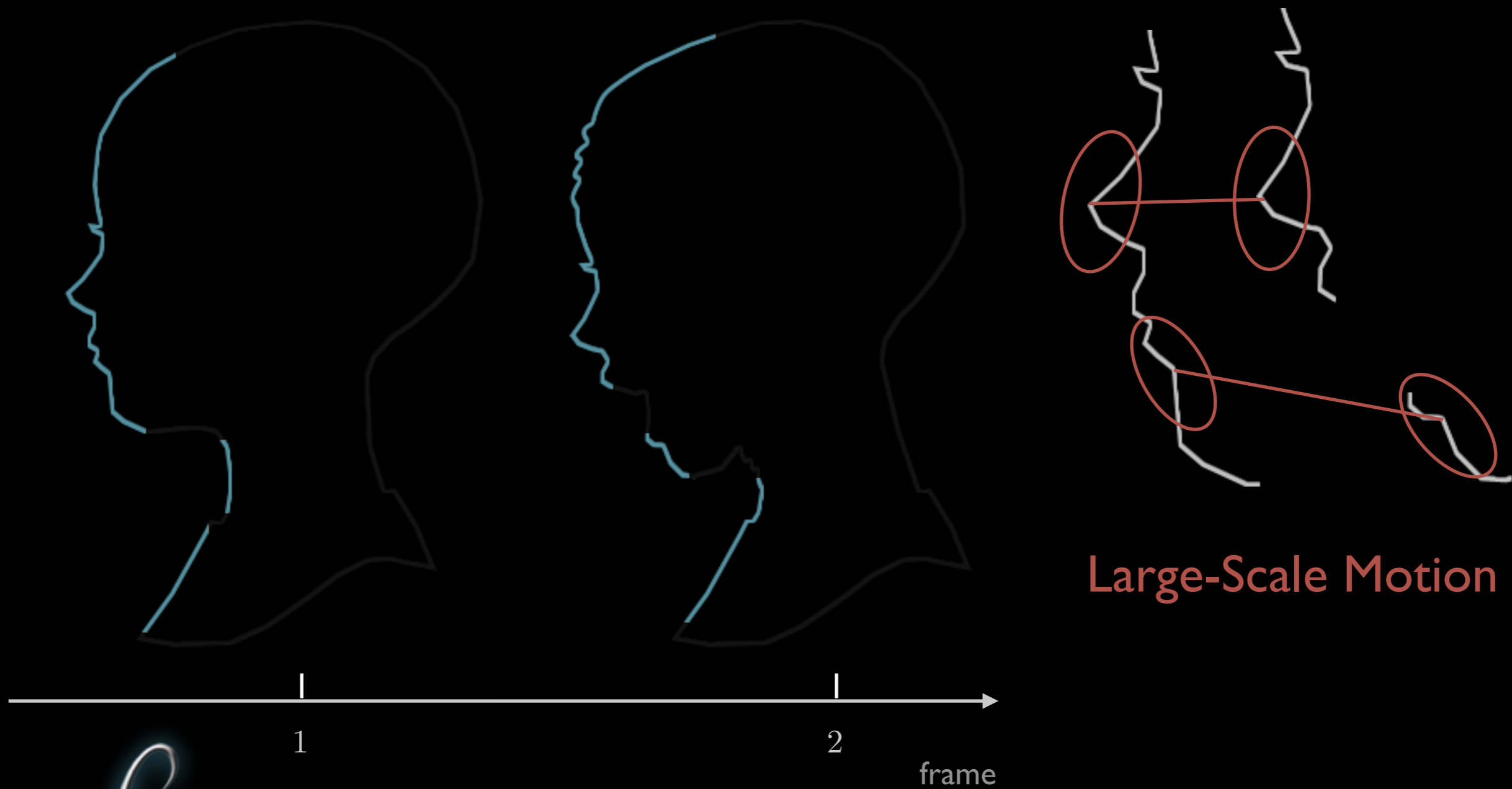
Partial and Non-Rigid Registration



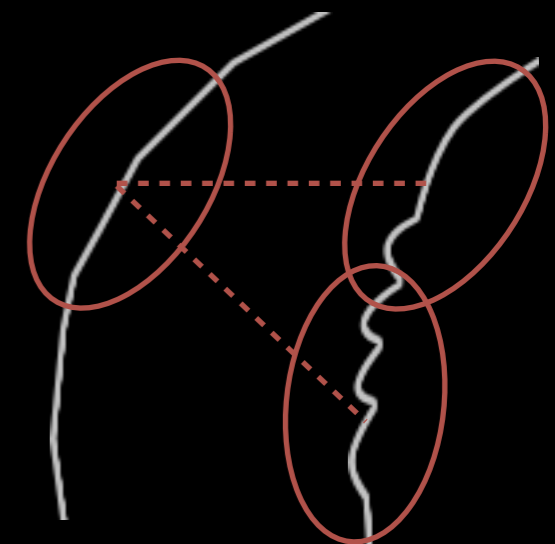
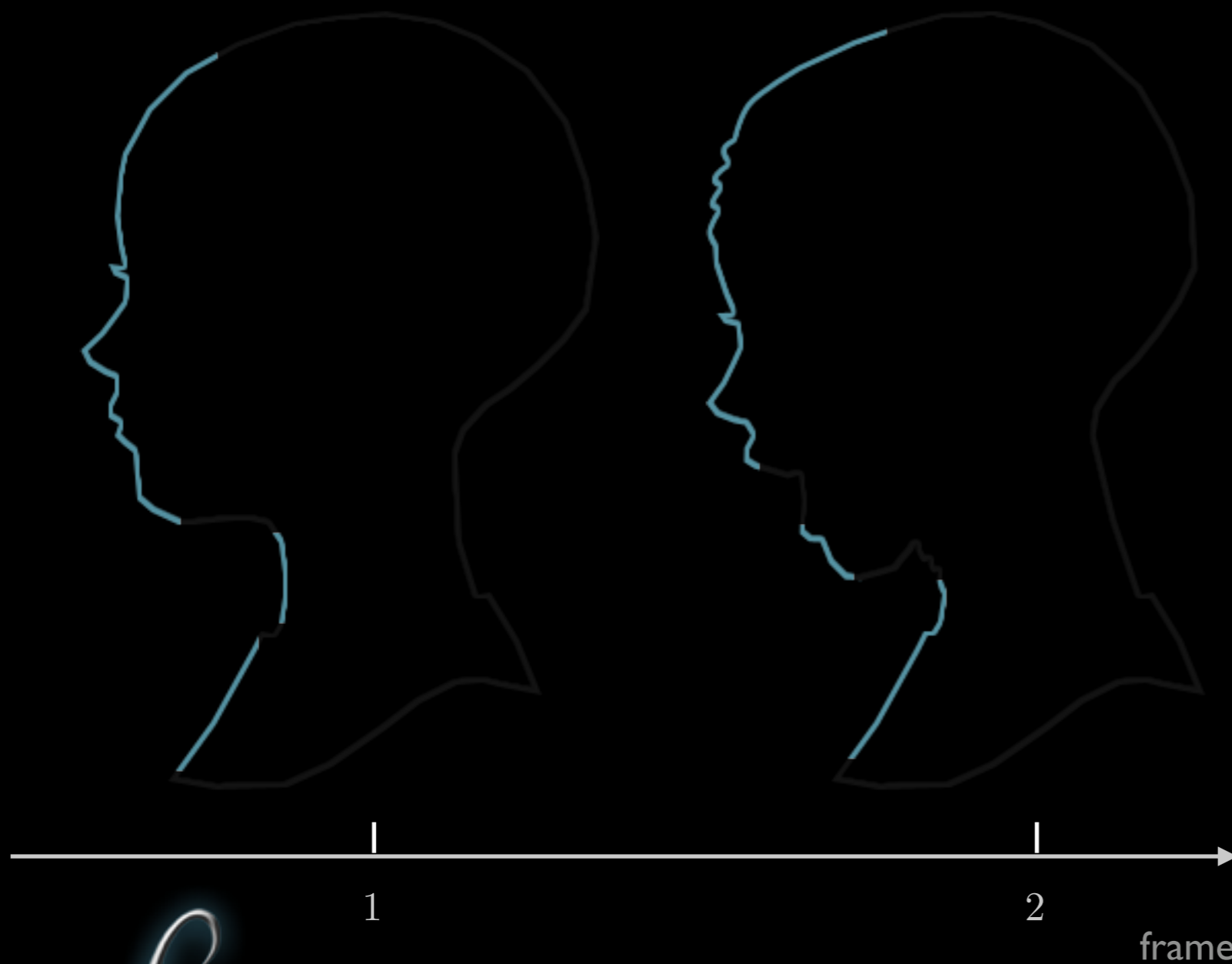
Partial and Non-Rigid Registration



Partial and Non-Rigid Registration



Partial and Non-Rigid Registration



Small-Scale Dynamics



Bi-Resolution Approach



Warping a **coarse** template



Bi-Resolution Approach



Warping a **coarse** template



Bi-Resolution Approach



Warping a **coarse** template



Bi-Resolution Approach



Warping a **coarse** template



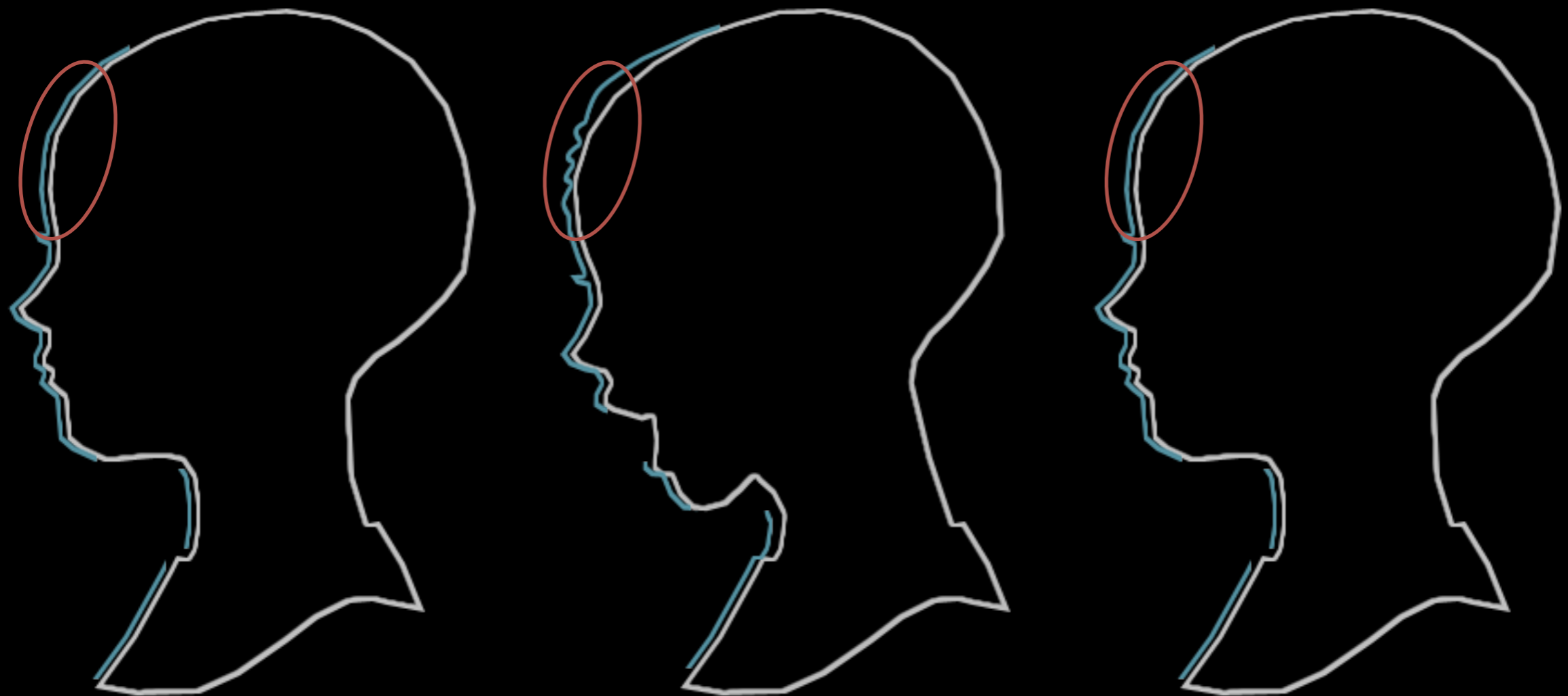
Bi-Resolution Approach



Warping a **coarse** template



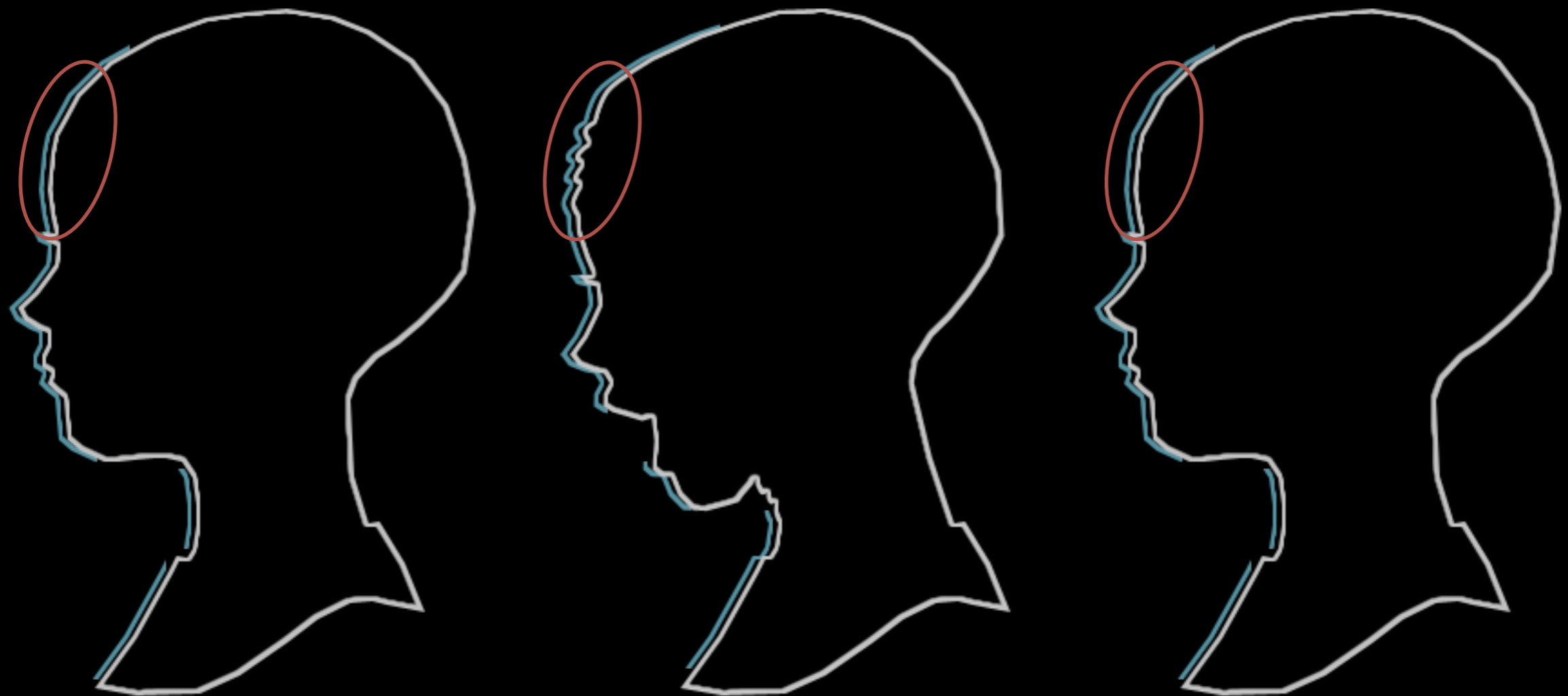
Bi-Resolution Approach



Warping a **coarse** template



Bi-Resolution Approach



Synthesizing **small scale** details



Reconstruction Framework



Reconstruction Framework

Input Scans



Reconstruction Framework

Input Scans



Coarse Template



Reconstruction Framework

Input Scans

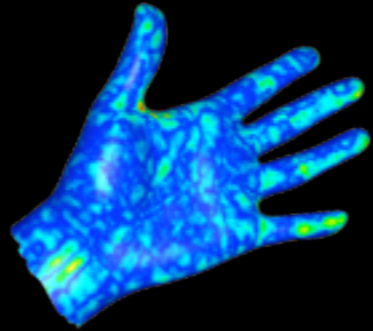


Coarse Template



Reconstruction Framework

Input Scans

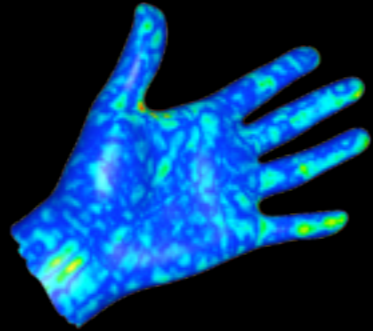


Coarse Template



Reconstruction Framework

Input Scans

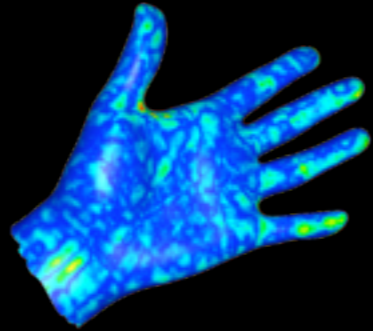


Coarse Template



Reconstruction Framework

Input Scans

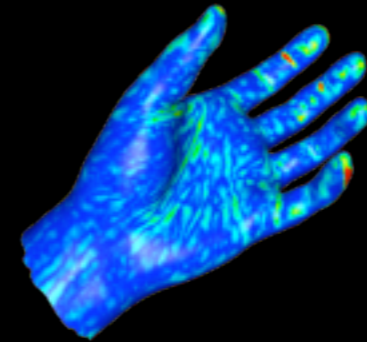
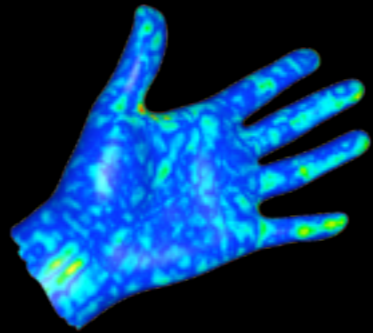


Coarse Template



Reconstruction Framework

Input Scans



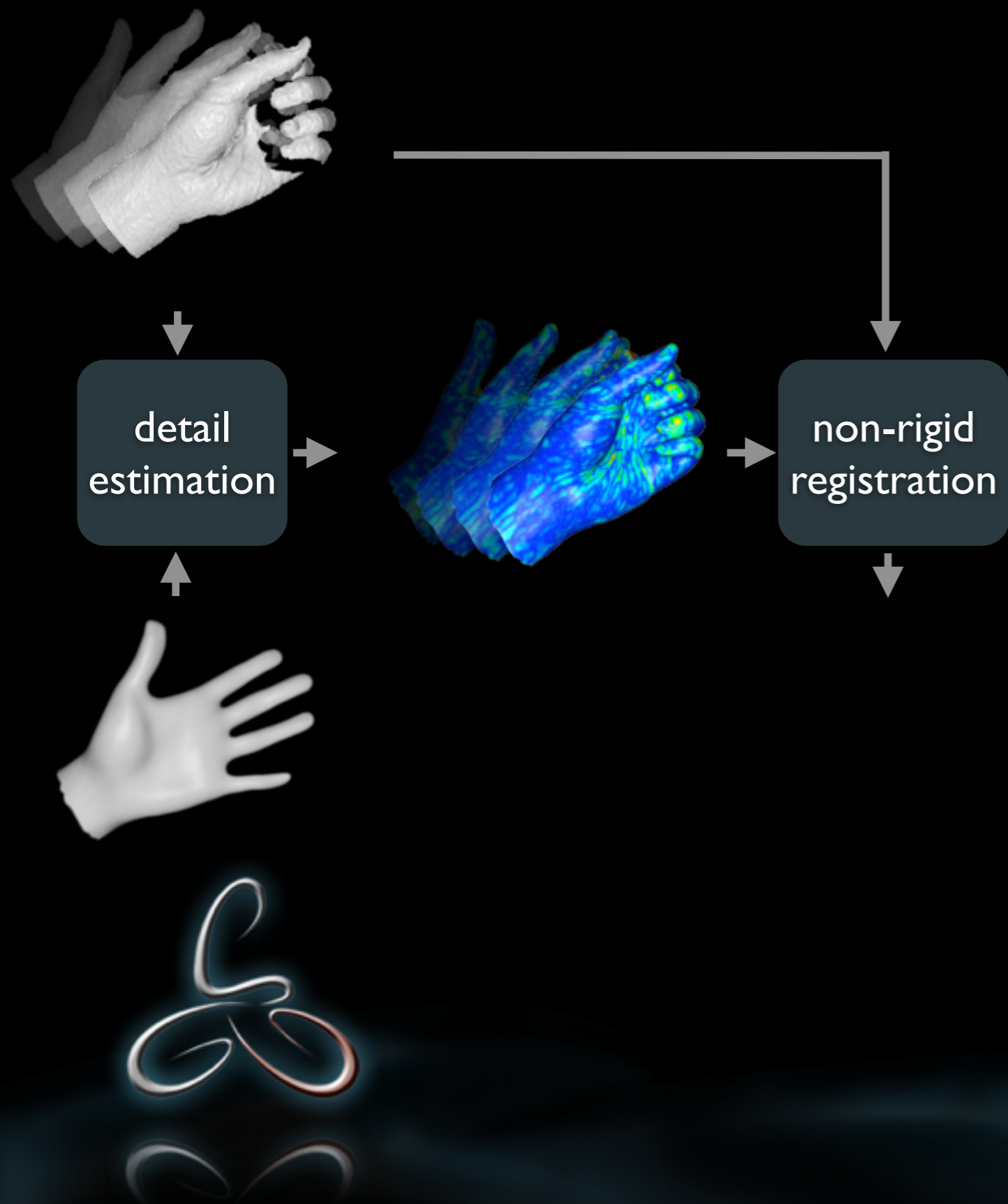
Coarse Template



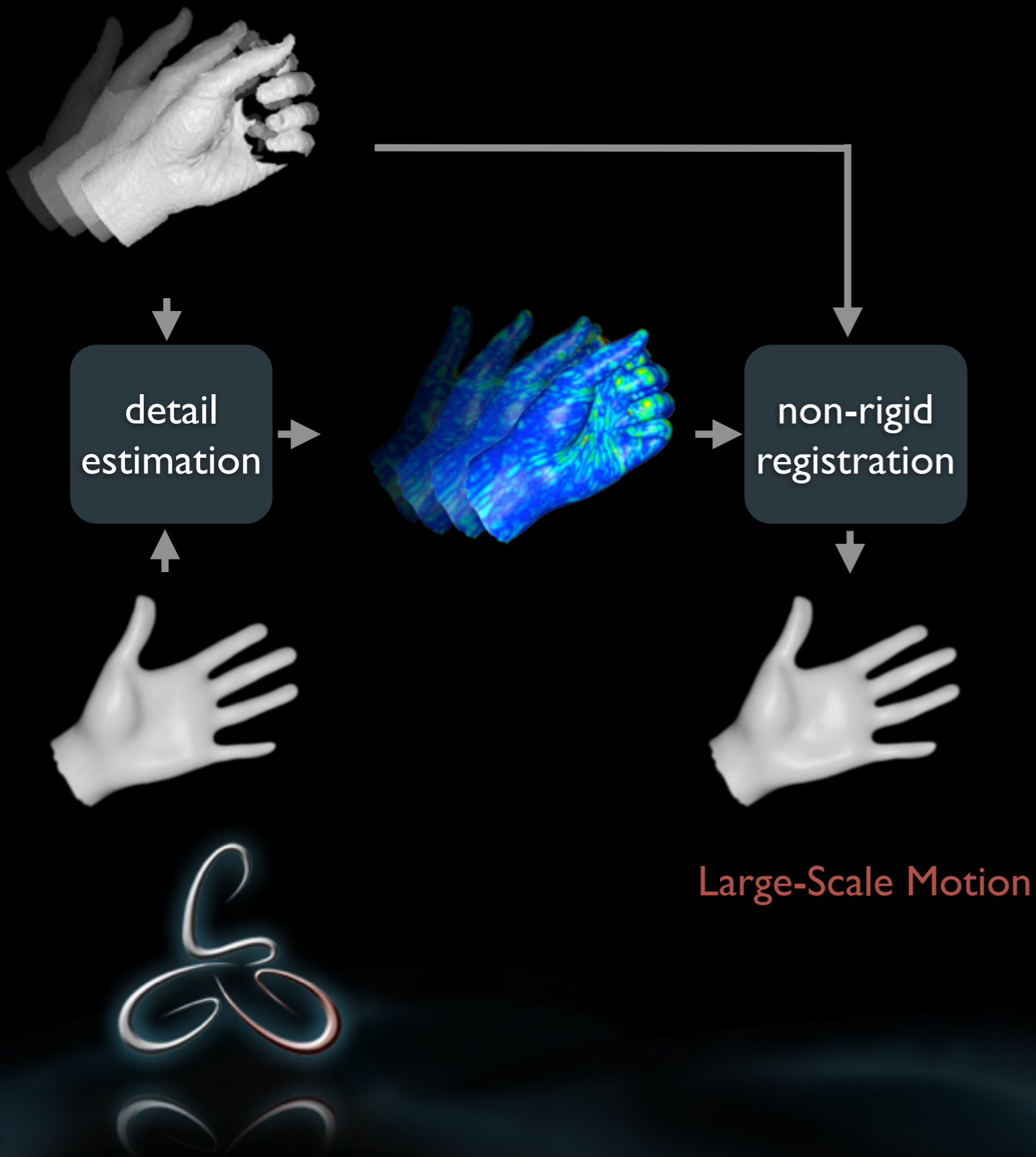
Reconstruction Framework



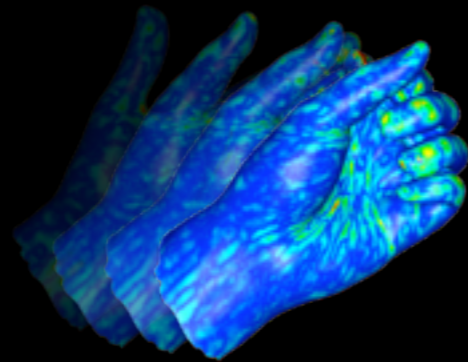
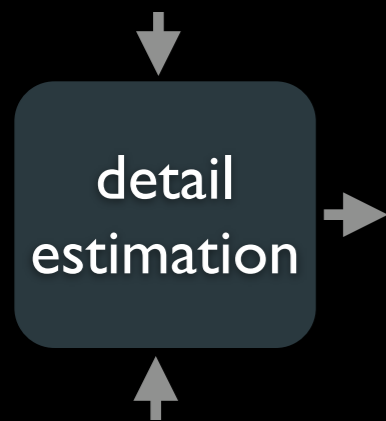
Reconstruction Framework



Reconstruction Framework



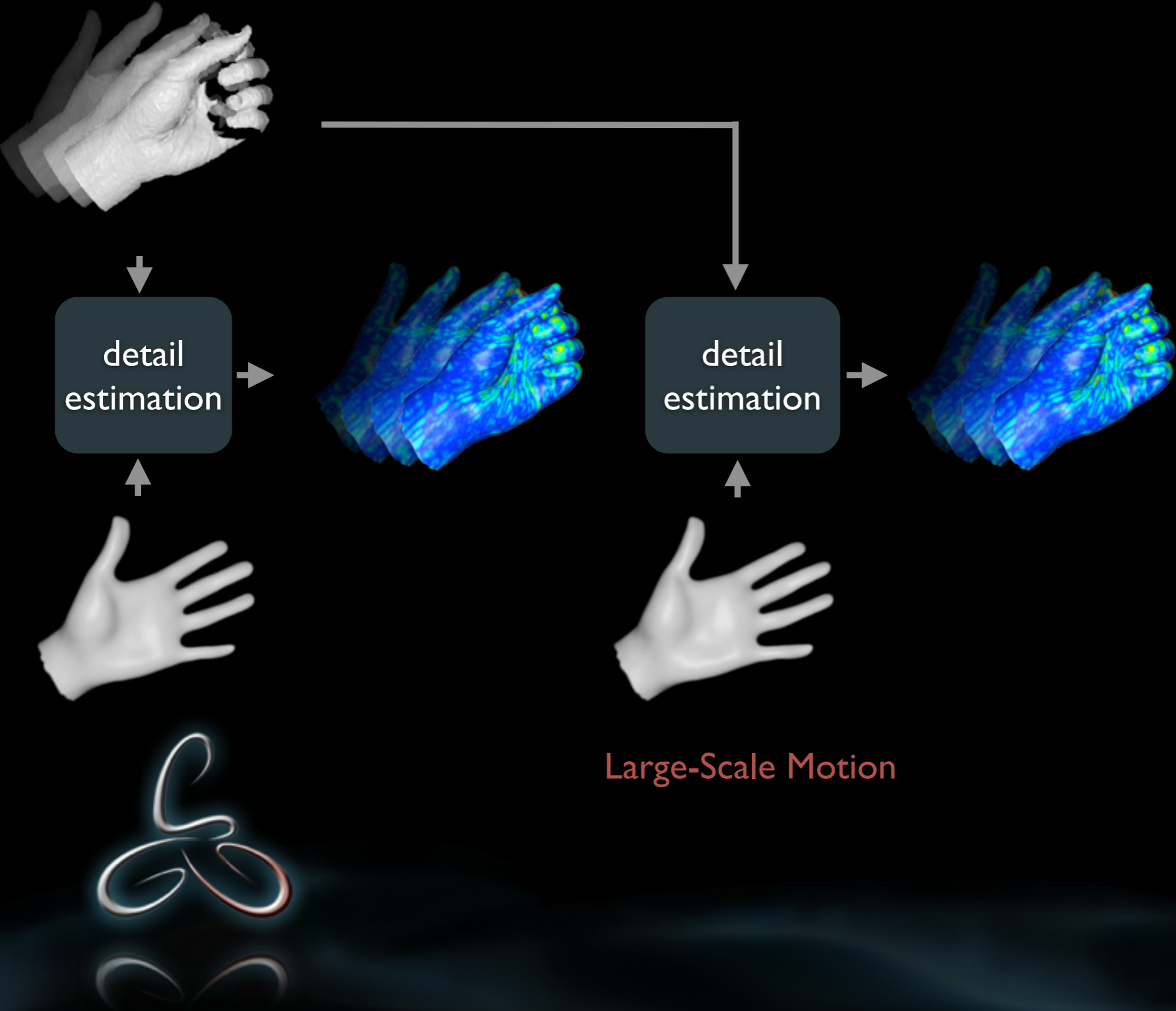
Reconstruction Framework



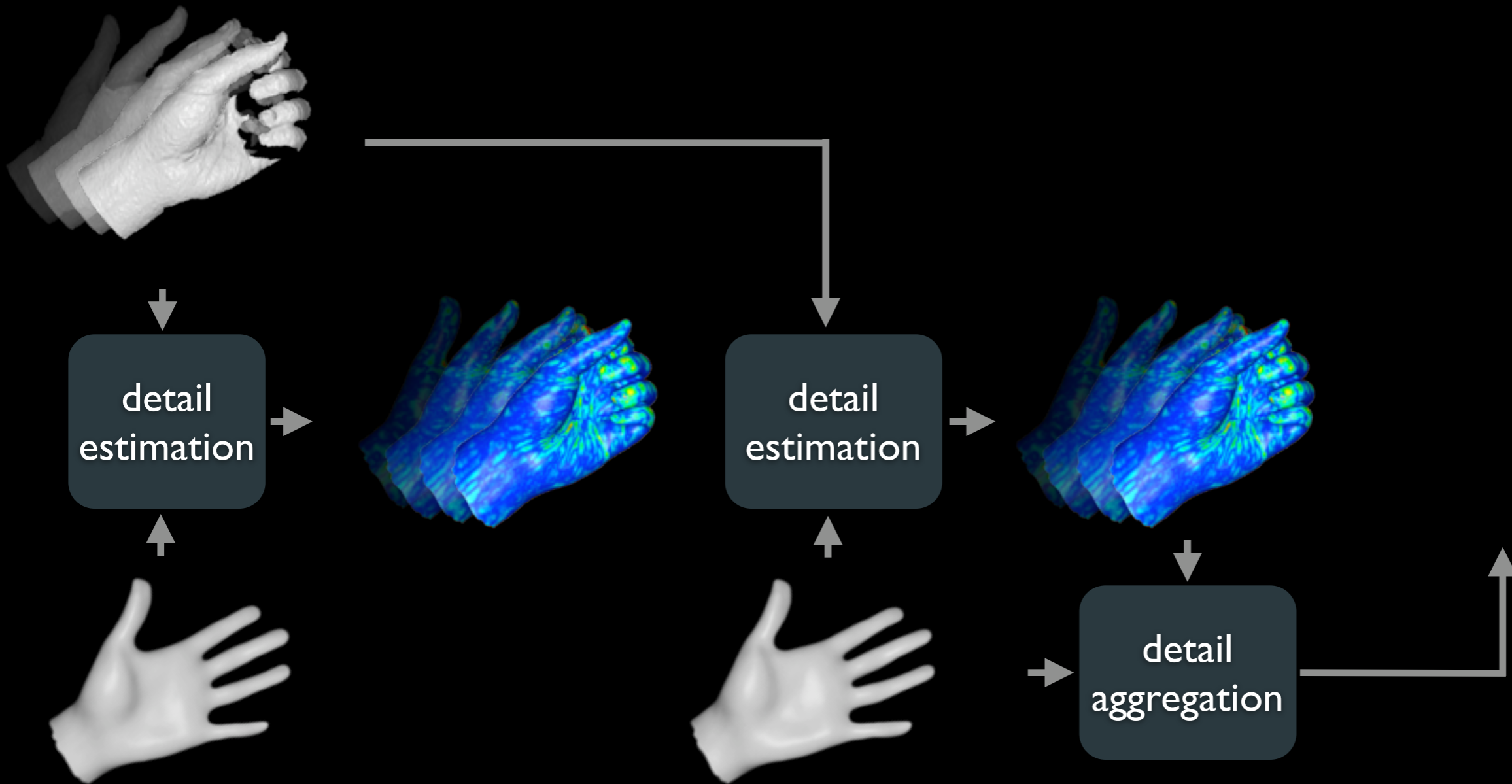
Large-Scale Motion



Reconstruction Framework



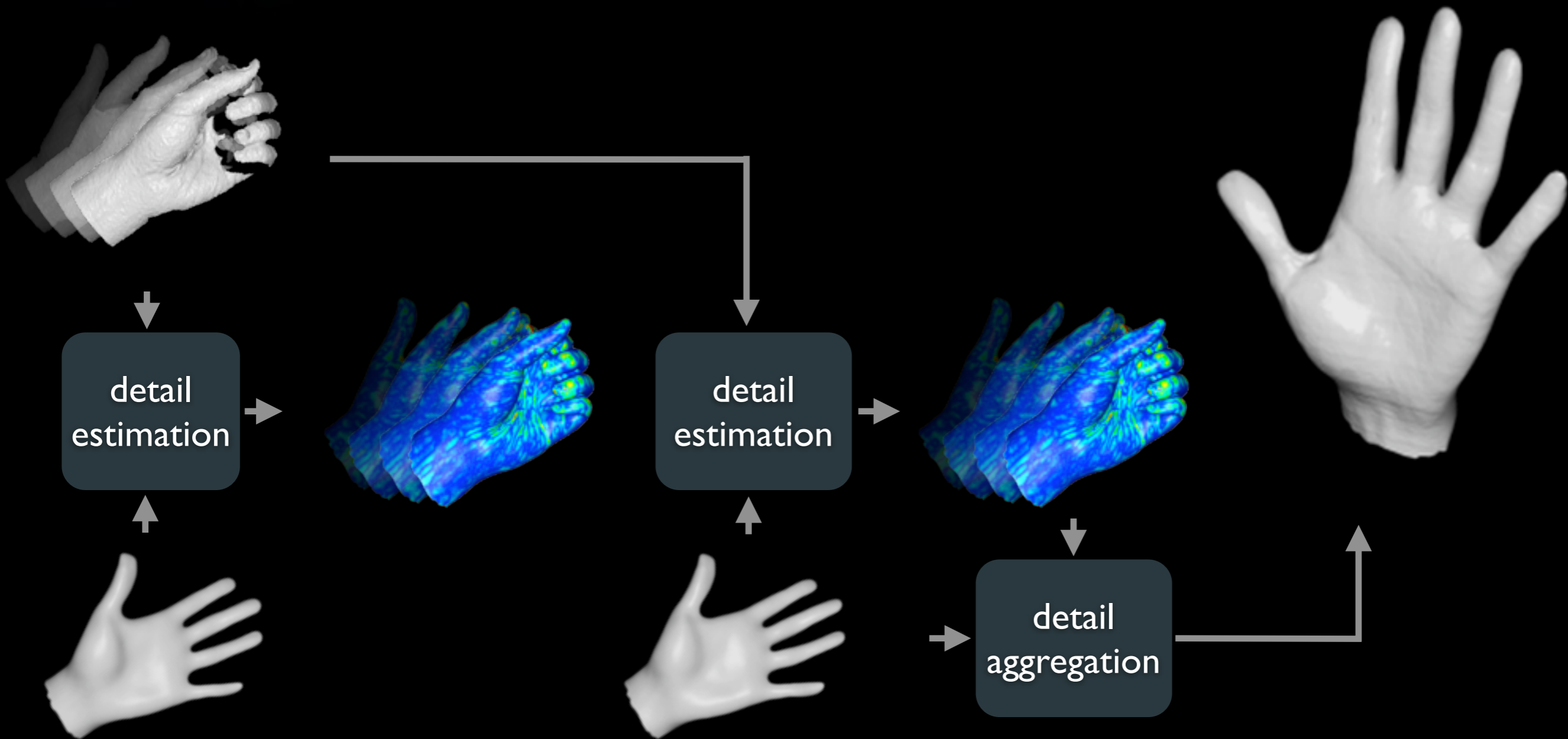
Reconstruction Framework



Large-Scale Motion



Reconstruction Framework



Large-Scale Motion

Fine-Scale Dynamics



Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

Maximize
Rigidity



Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

Maximize
Consistency



Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}} \quad \text{Stiffness}$$



Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

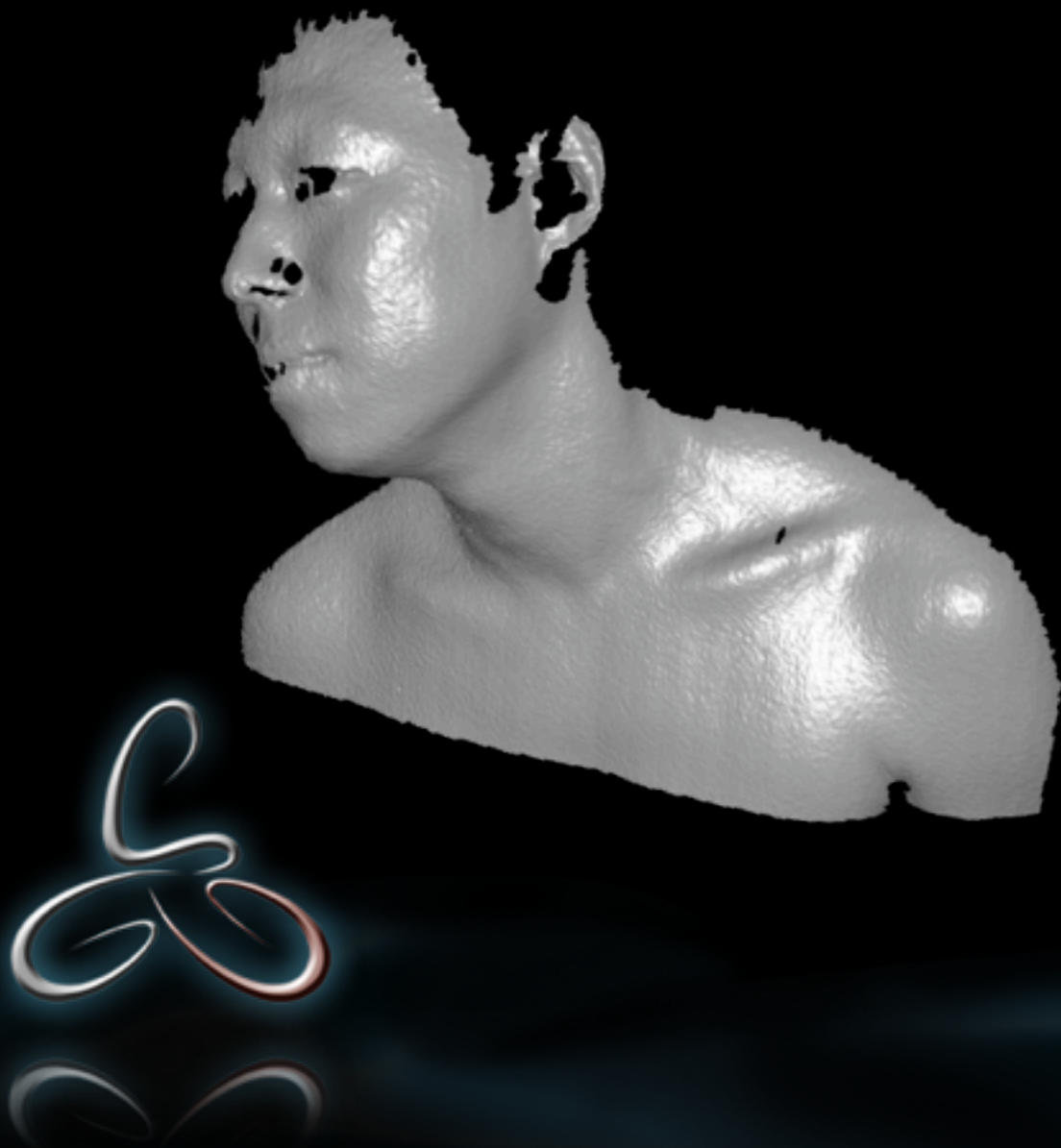


Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

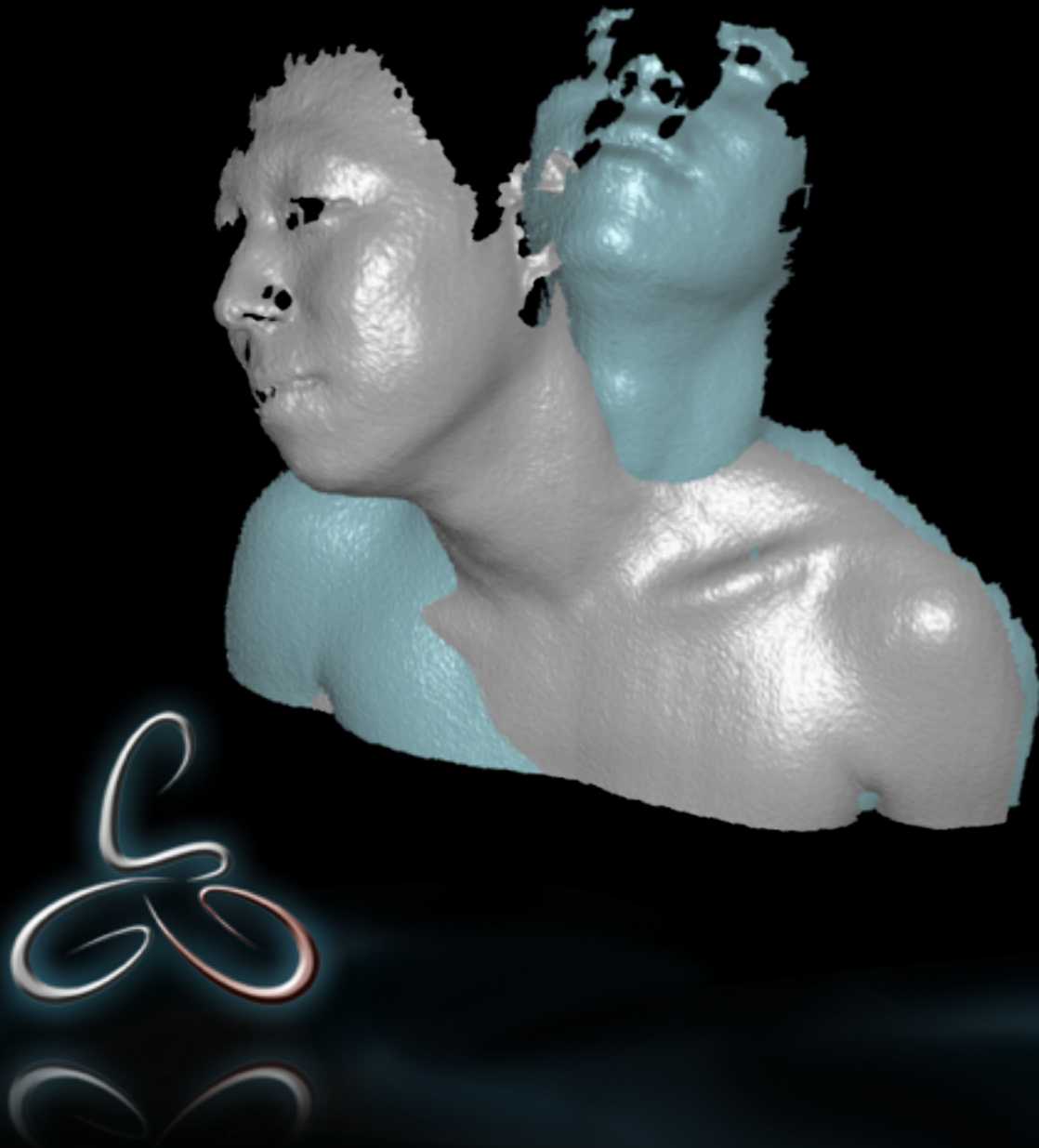


Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

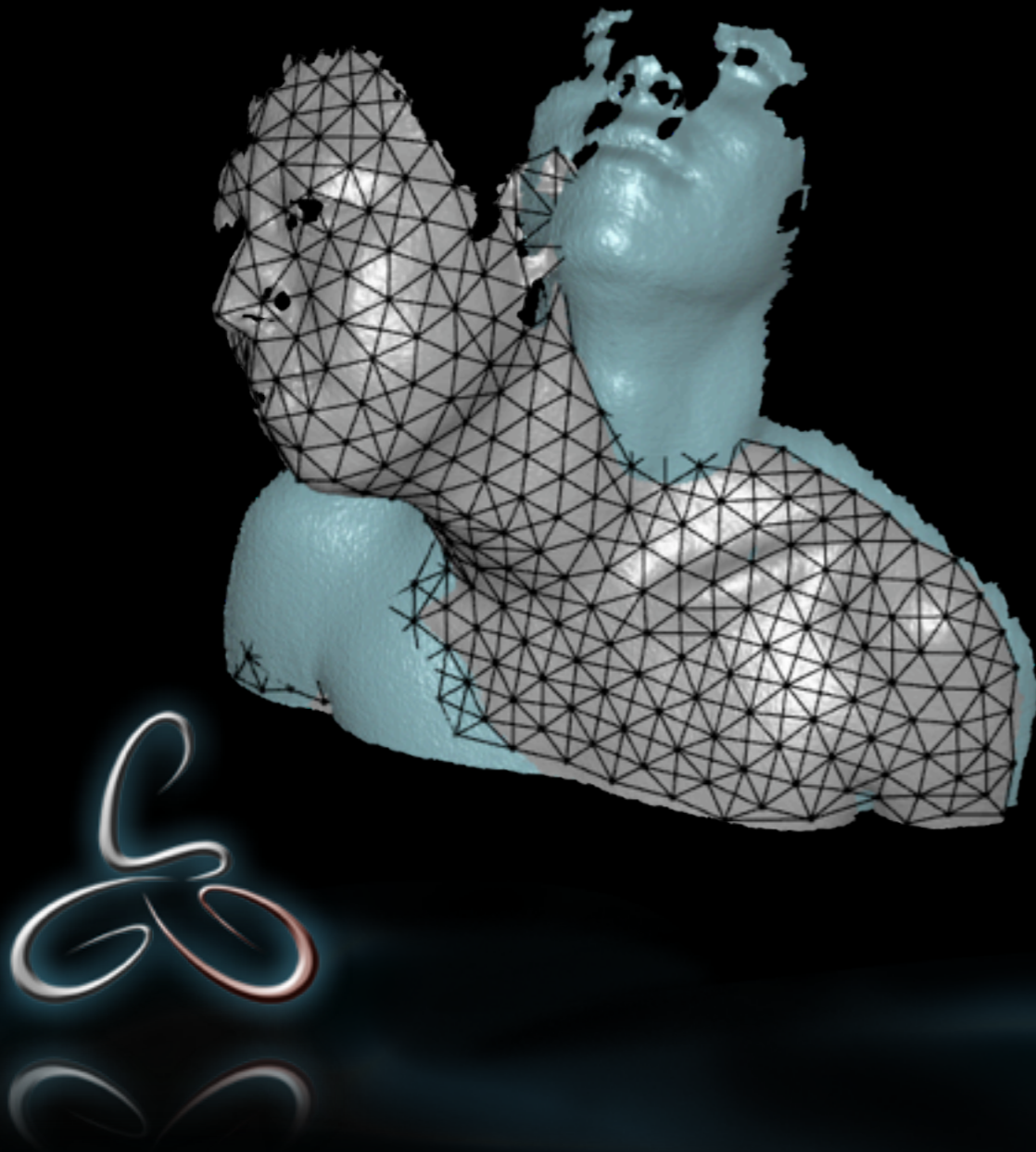


Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

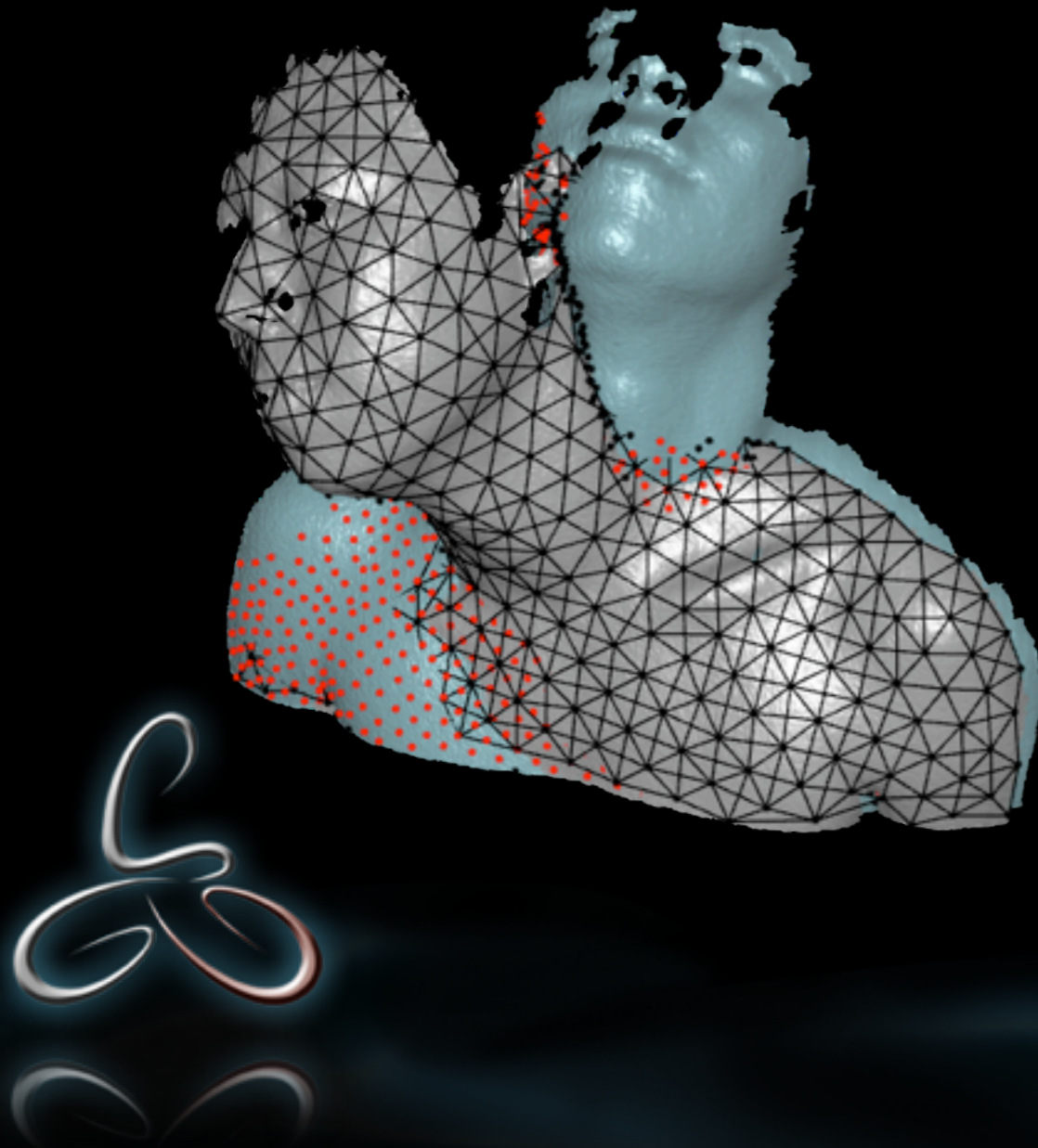


Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$

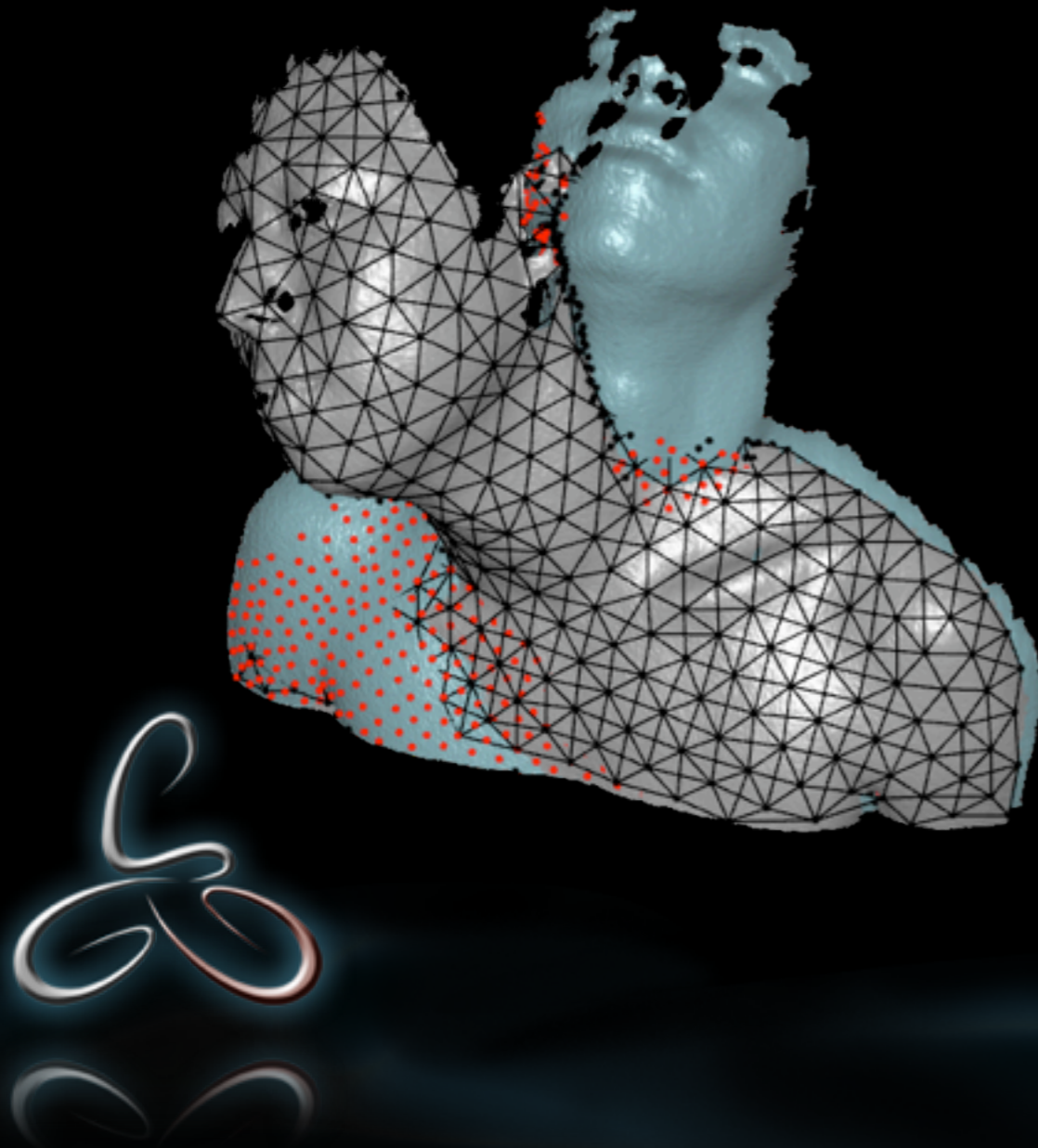


Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:



Extension of [Li et al. '08]

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate



Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate

Too many nodes:

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate

Too many nodes:

- inefficient

Non-Rigid ICP

Non-Linear Optimization

$$E_{\text{tot}} = \alpha_{\text{fit}} E_{\text{fit}} + \alpha_{\text{rigid}} E_{\text{rigid}} + \alpha_{\text{smooth}} E_{\text{smooth}}$$



Too few nodes:

- inaccurate

Too many nodes:

- inefficient
- less robust

Adaptive Deformation Model



Adaptive Deformation Model

Non-Rigid ICP



Adaptive Deformation Model

Non-Rigid ICP



Adaptive Deformation Model

Non-Rigid ICP



Adaptive Deformation Model

Non-Rigid ICP



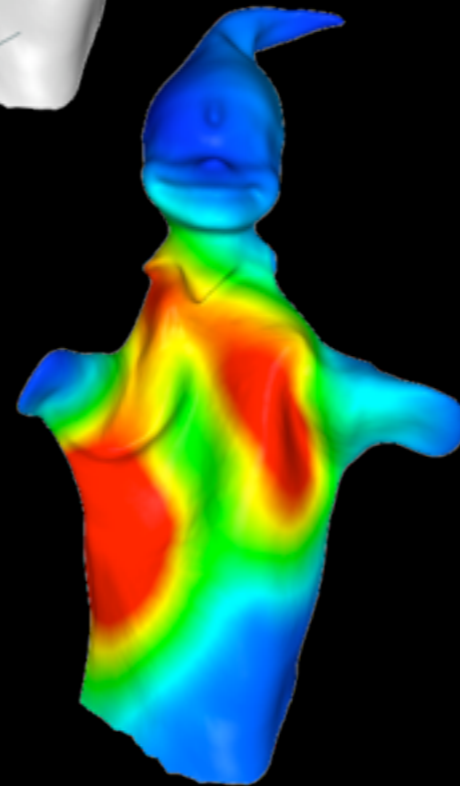
$E_{\text{smooth}} > \sigma?$



Adaptive Deformation Model

Non-Rigid ICP

$E_{\text{smooth}} > \sigma?$



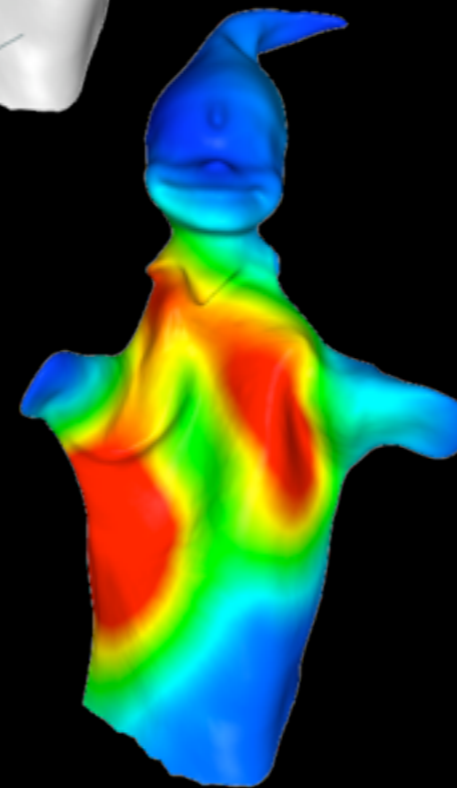
Adaptive Deformation Model

Non-Rigid ICP

$E_{\text{smooth}} > \sigma?$

yes

Refine Graph



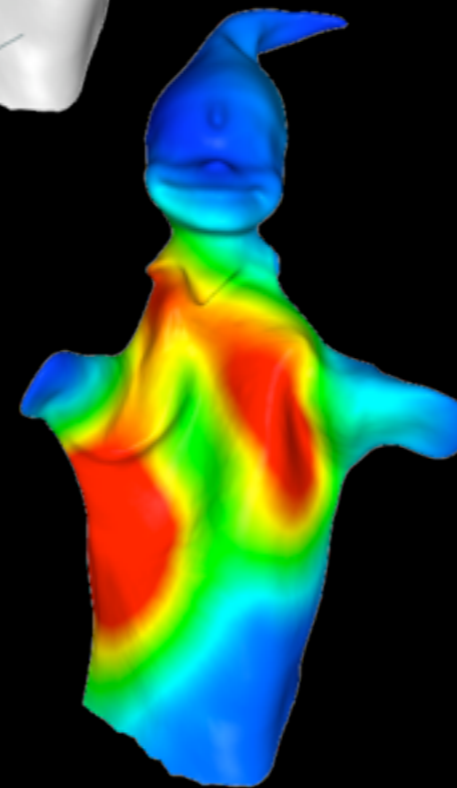
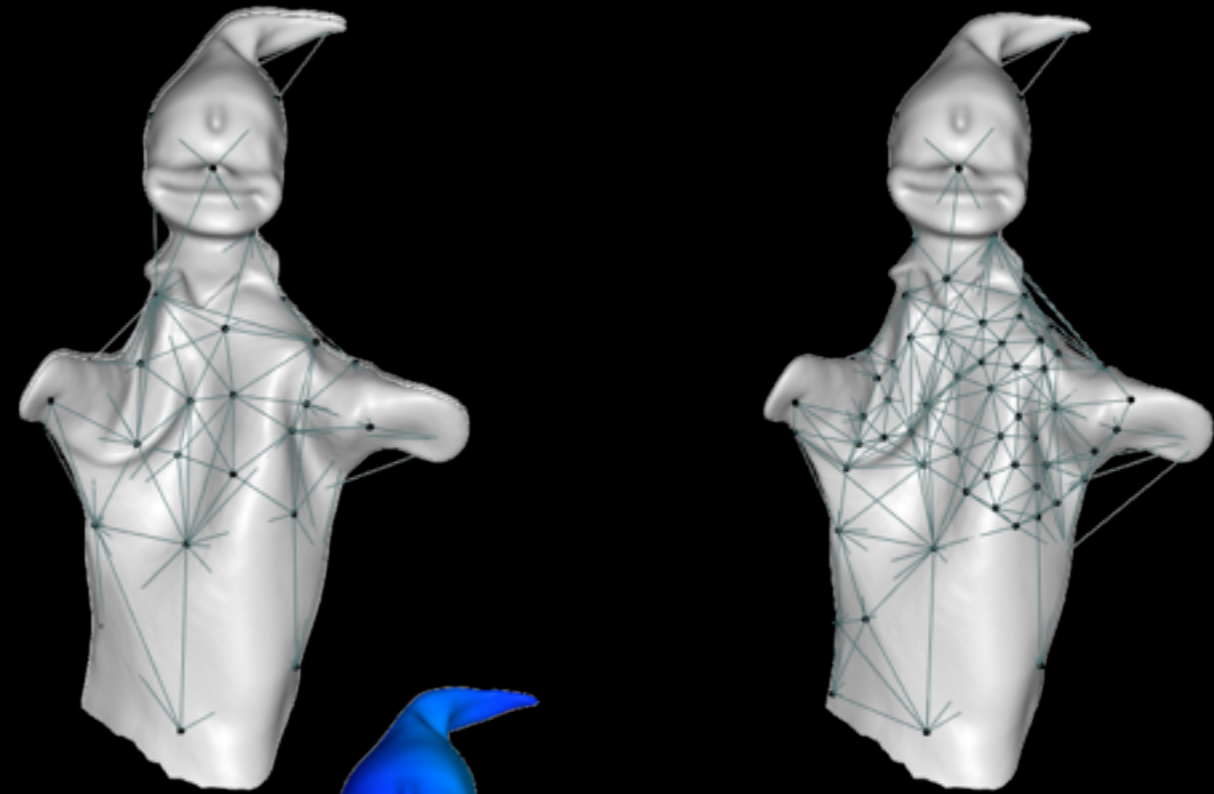
Adaptive Deformation Model

Non-Rigid ICP

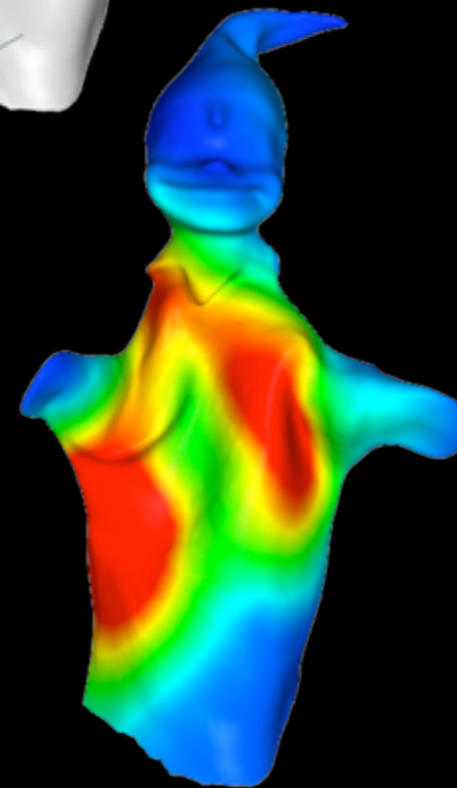
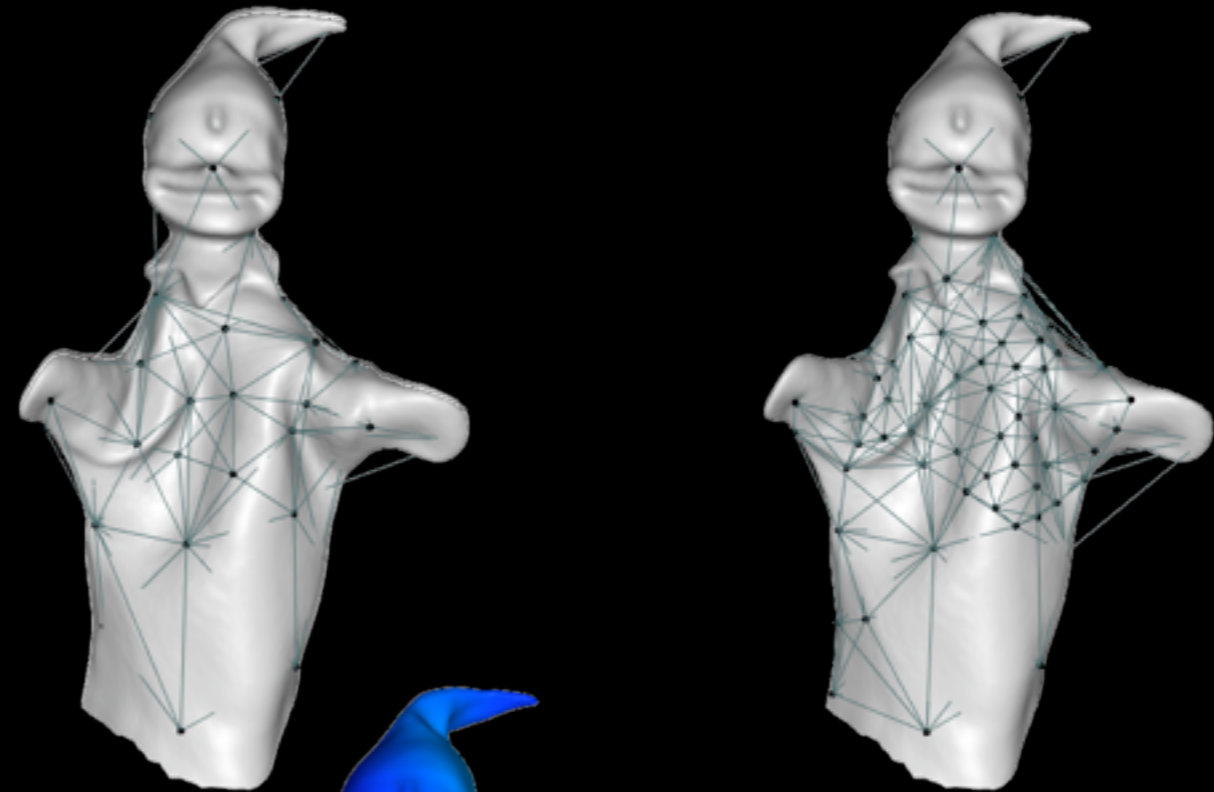
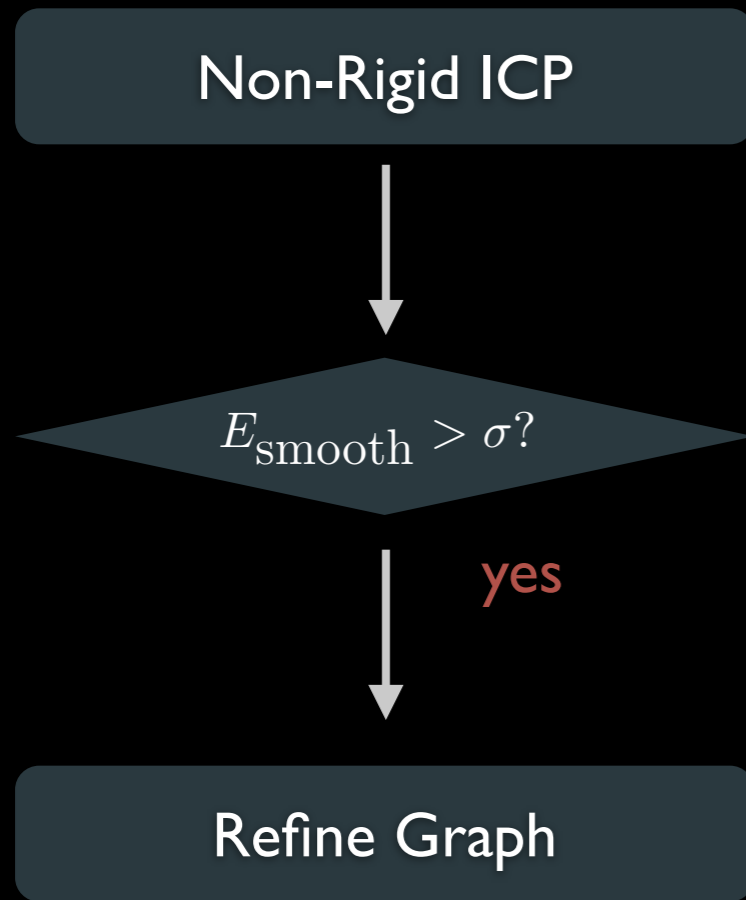
$E_{\text{smooth}} > \sigma?$

yes

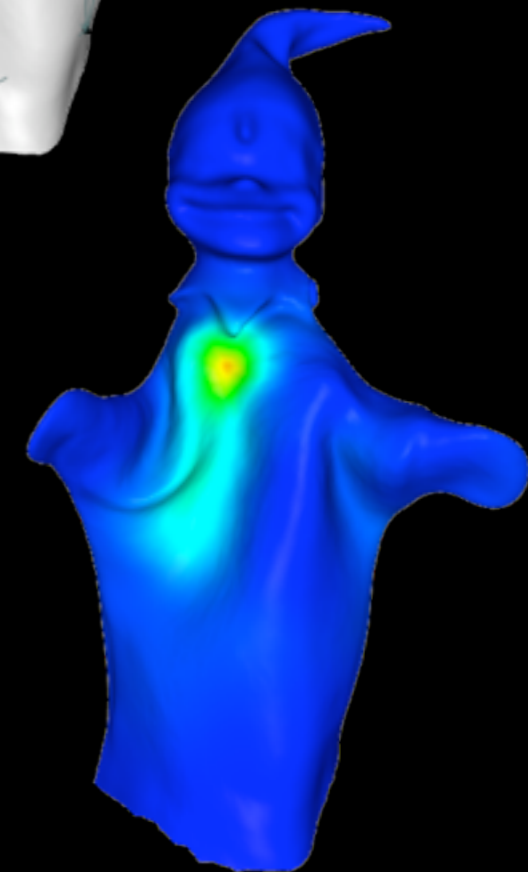
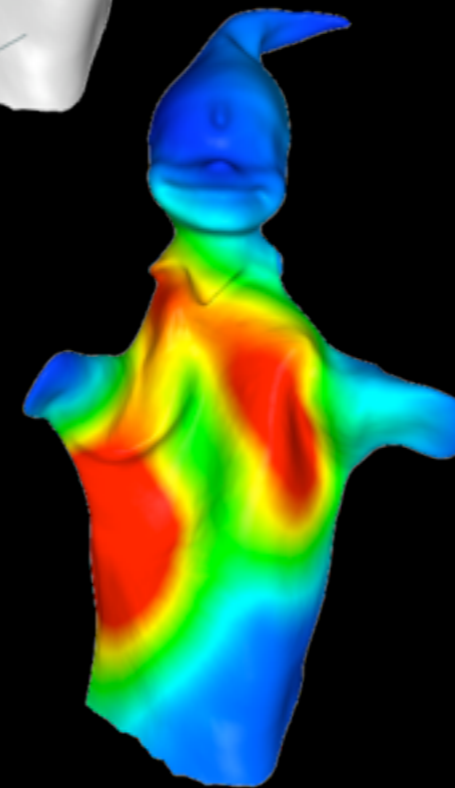
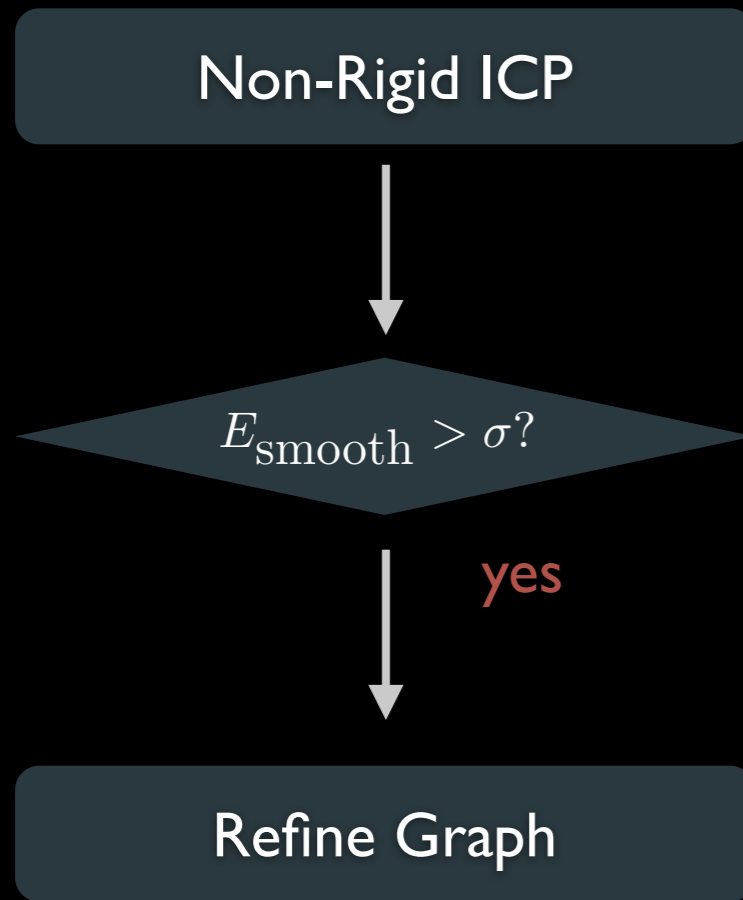
Refine Graph



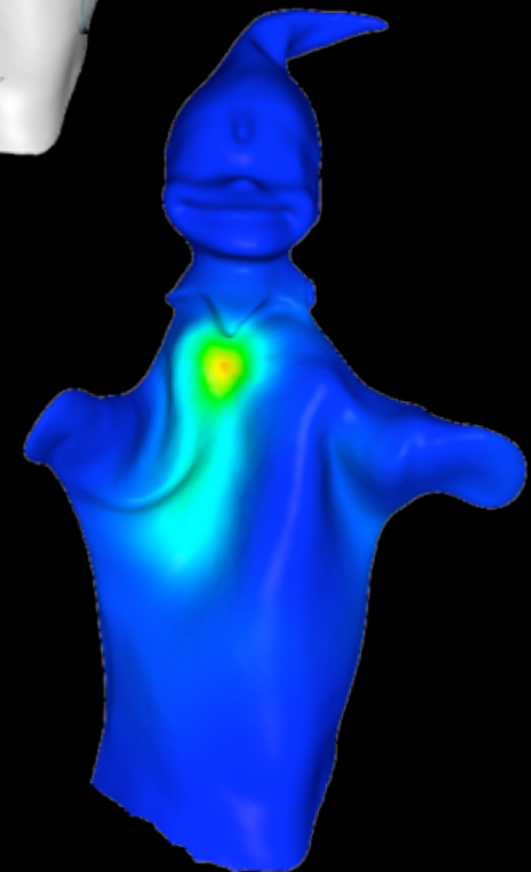
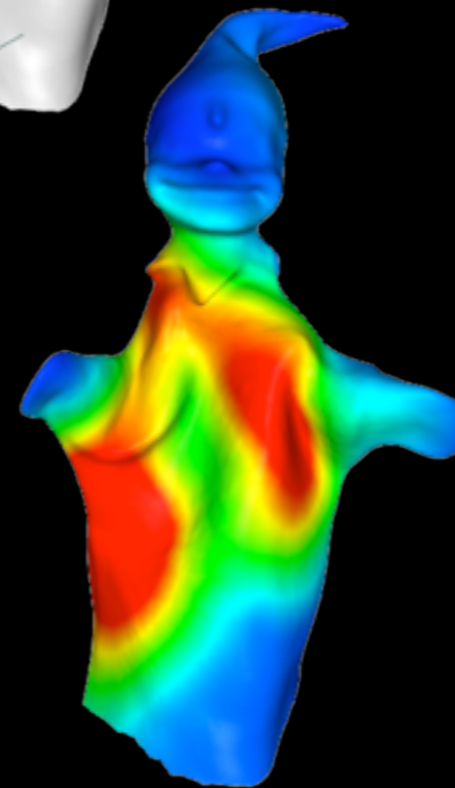
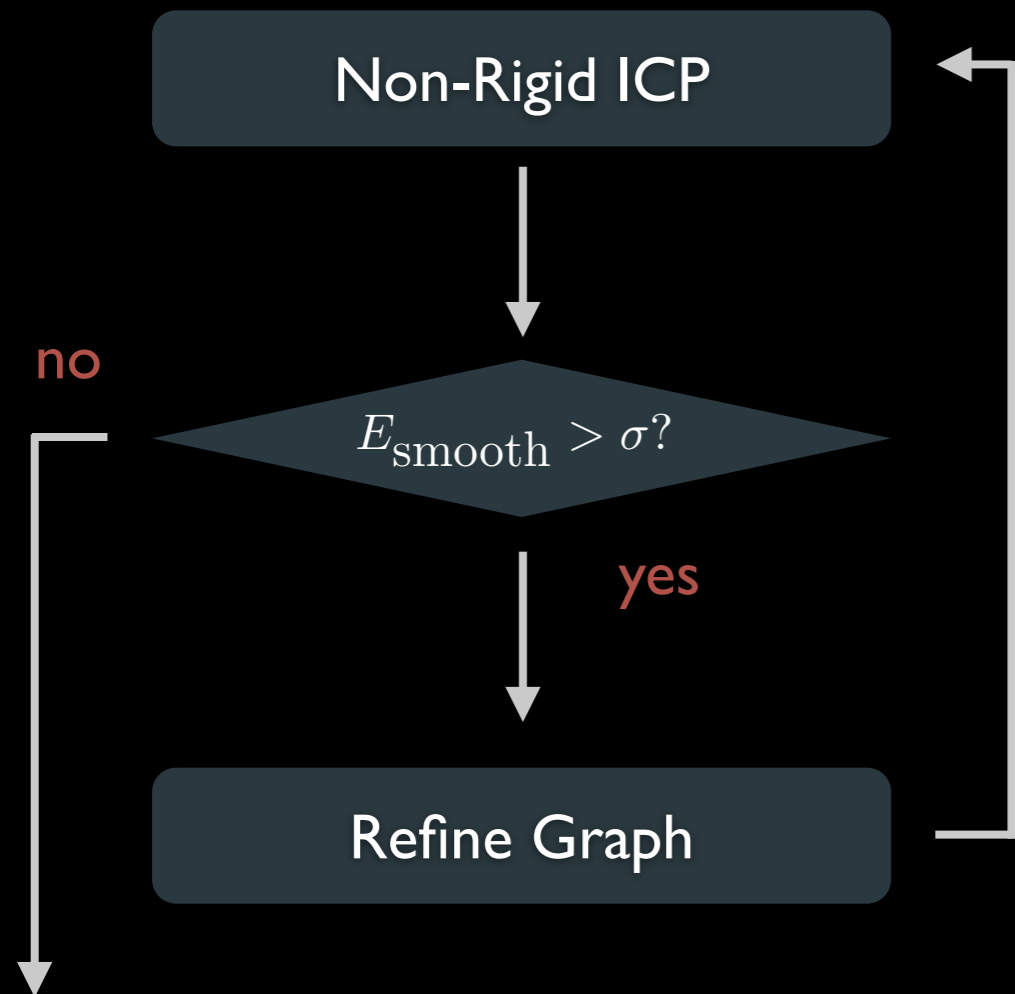
Adaptive Deformation Model



Adaptive Deformation Model



Adaptive Deformation Model



Next Scan



Adaptive Deformation Model



Input Scans

Warped Template with Graph

Adaptive Deformation Model



Input Scans



Warped Template with Graph



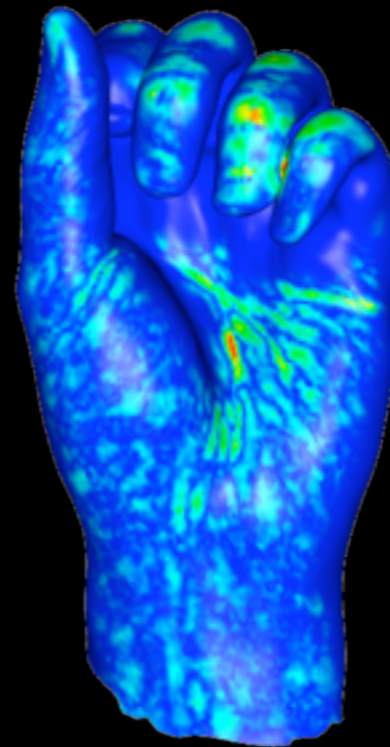
Detail Aggregation



Detail Aggregation



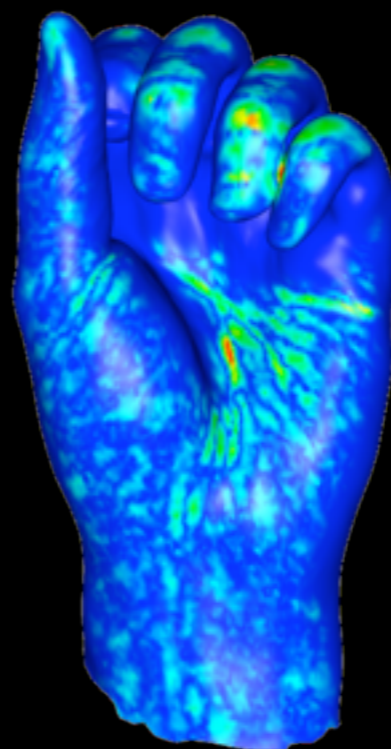
Detail Aggregation



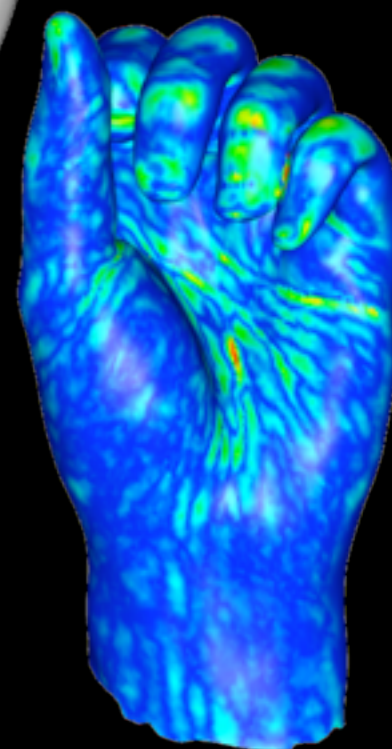
Single Frame
Synthesis



Detail Aggregation



Single Frame
Synthesis



Multi-Frame
Aggregation



Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$



Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$



Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$



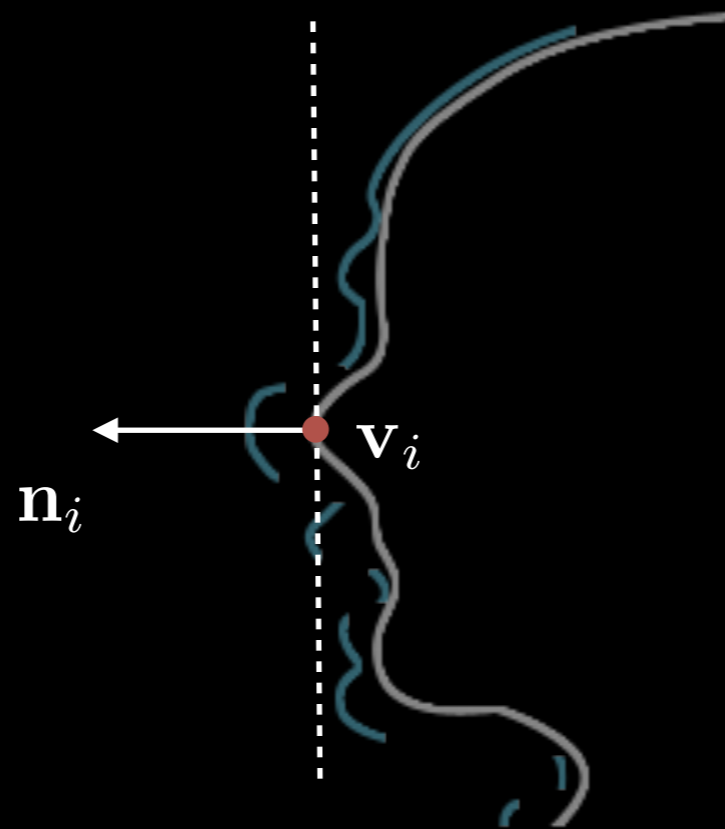
Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$



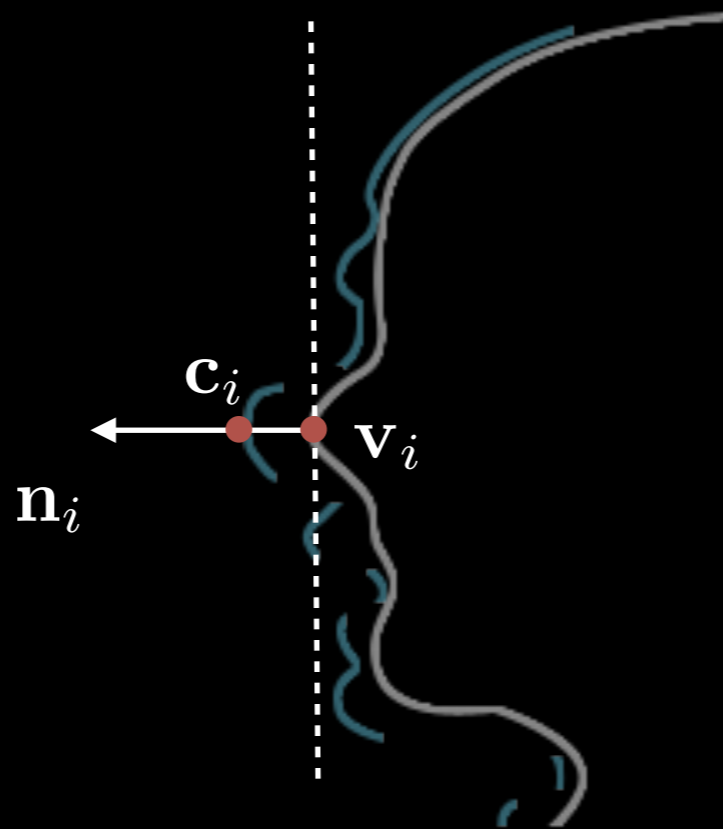
Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$



Detail Estimation

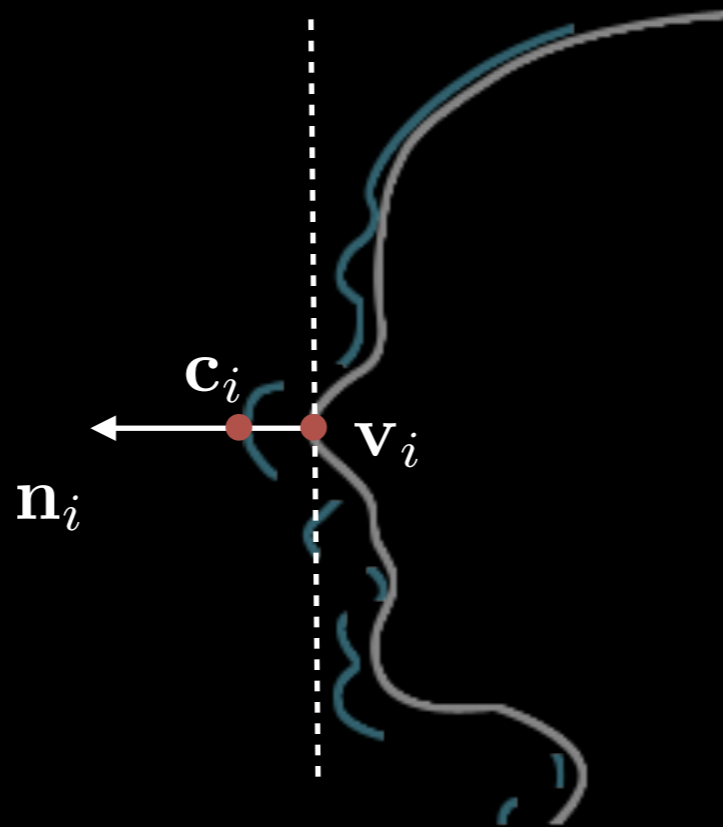
$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$



Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$

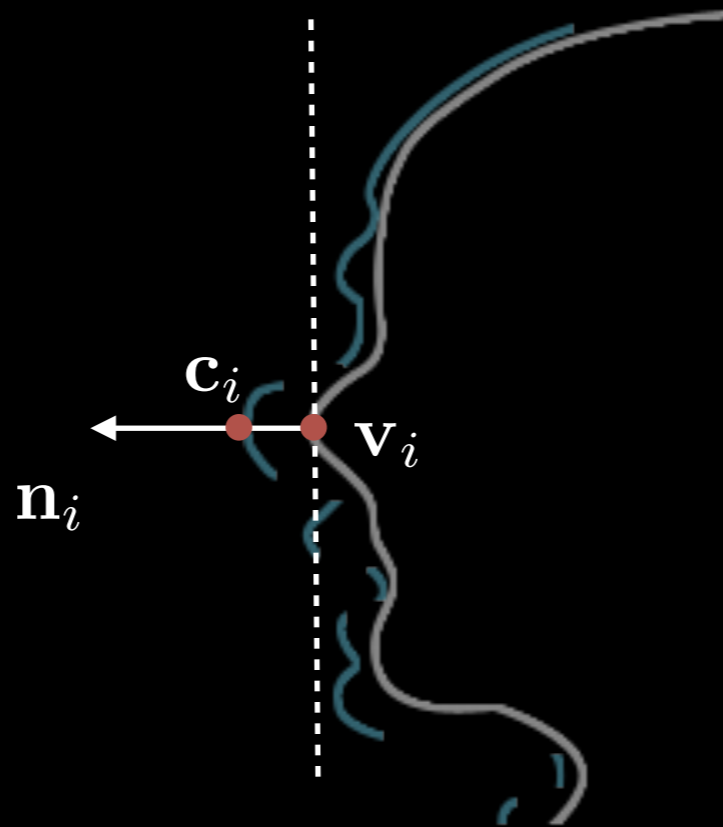
Point Constraint



Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$

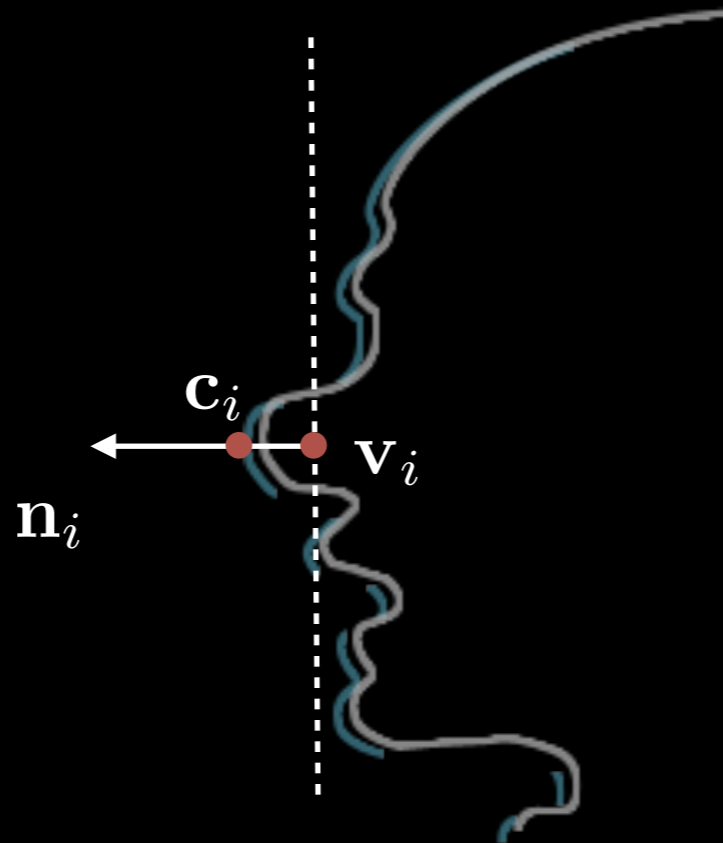
Point Constraint Regularization



Detail Estimation

$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$

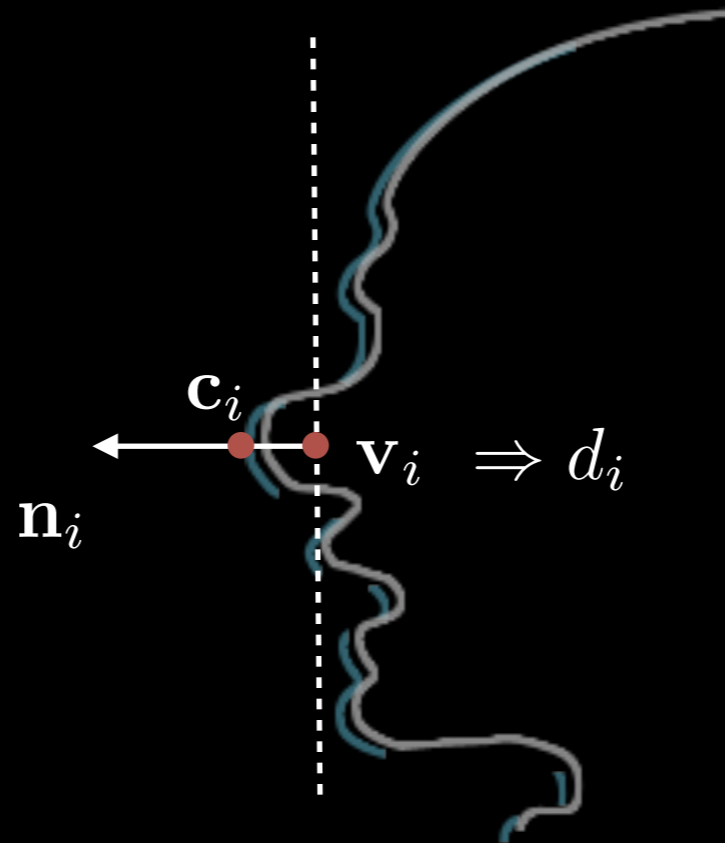
Point Constraint Regularization



Detail Estimation

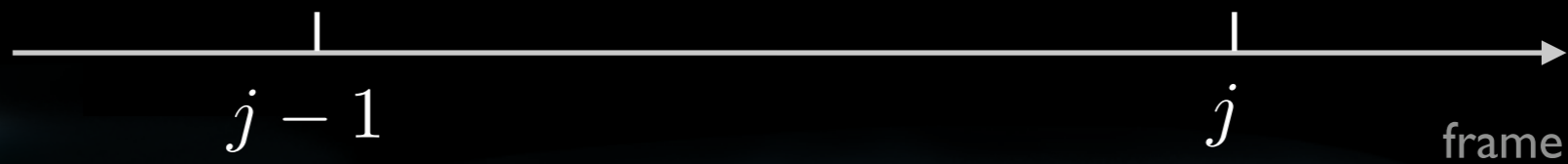
$$E_{\text{detail}} = \sum_{i \in \mathcal{V}} \|\mathbf{v}_i + d_i \mathbf{n}_i - \mathbf{c}_i\|^2 + \beta \sum_{(i,j) \in \mathcal{E}} |d_i - d_j|^2$$

Point Constraint Regularization

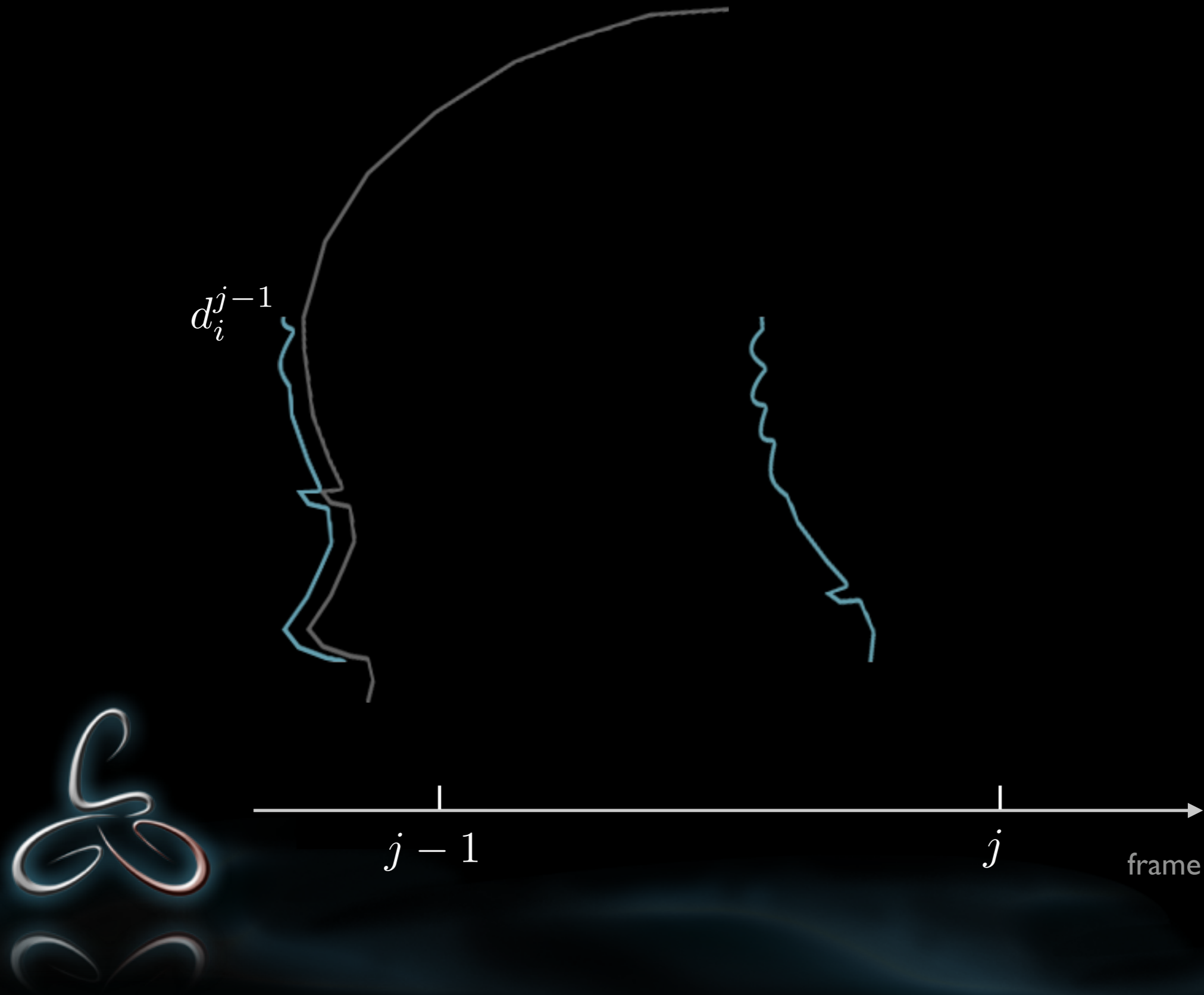


Detail Propagation

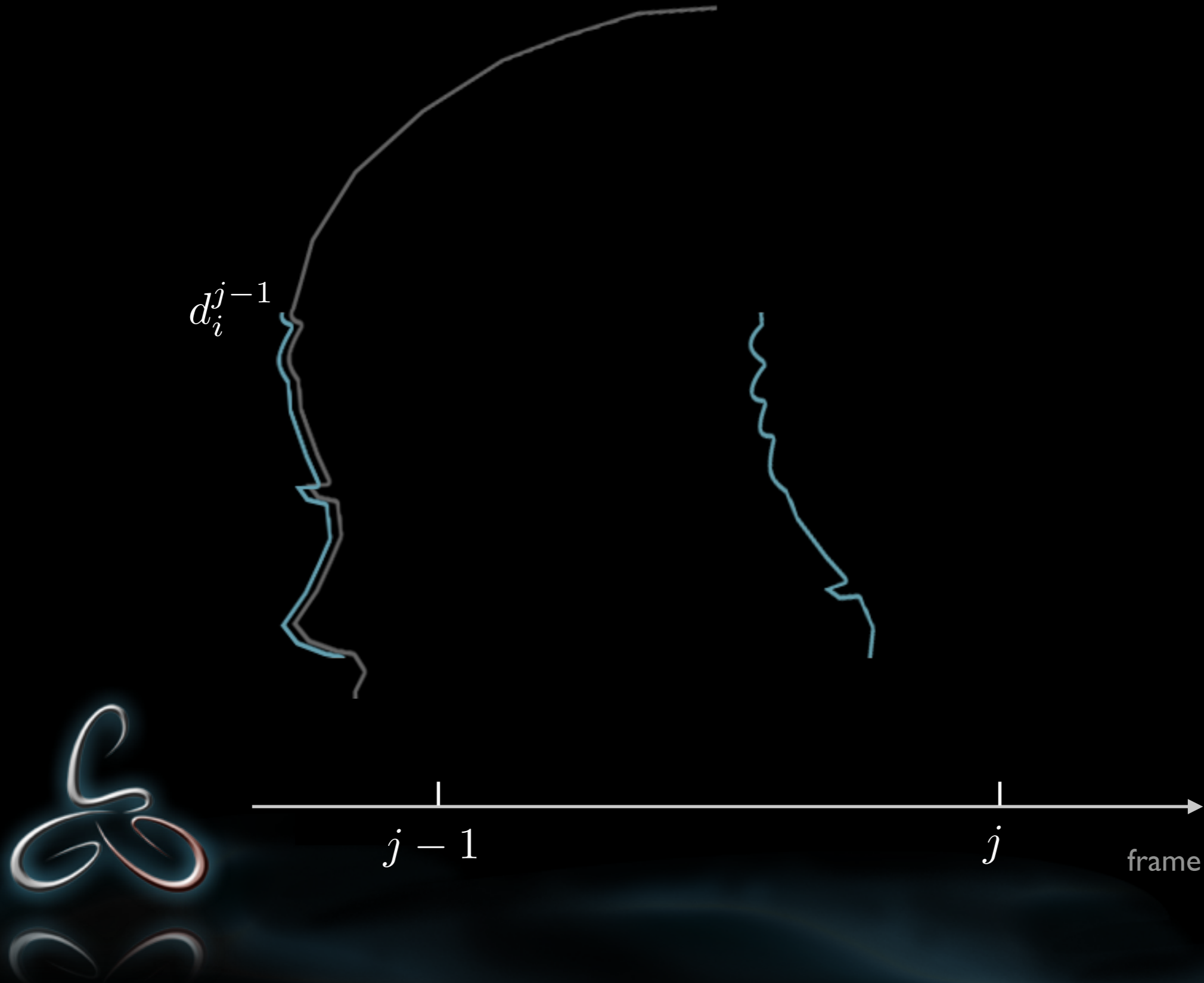
d_i^{j-1}



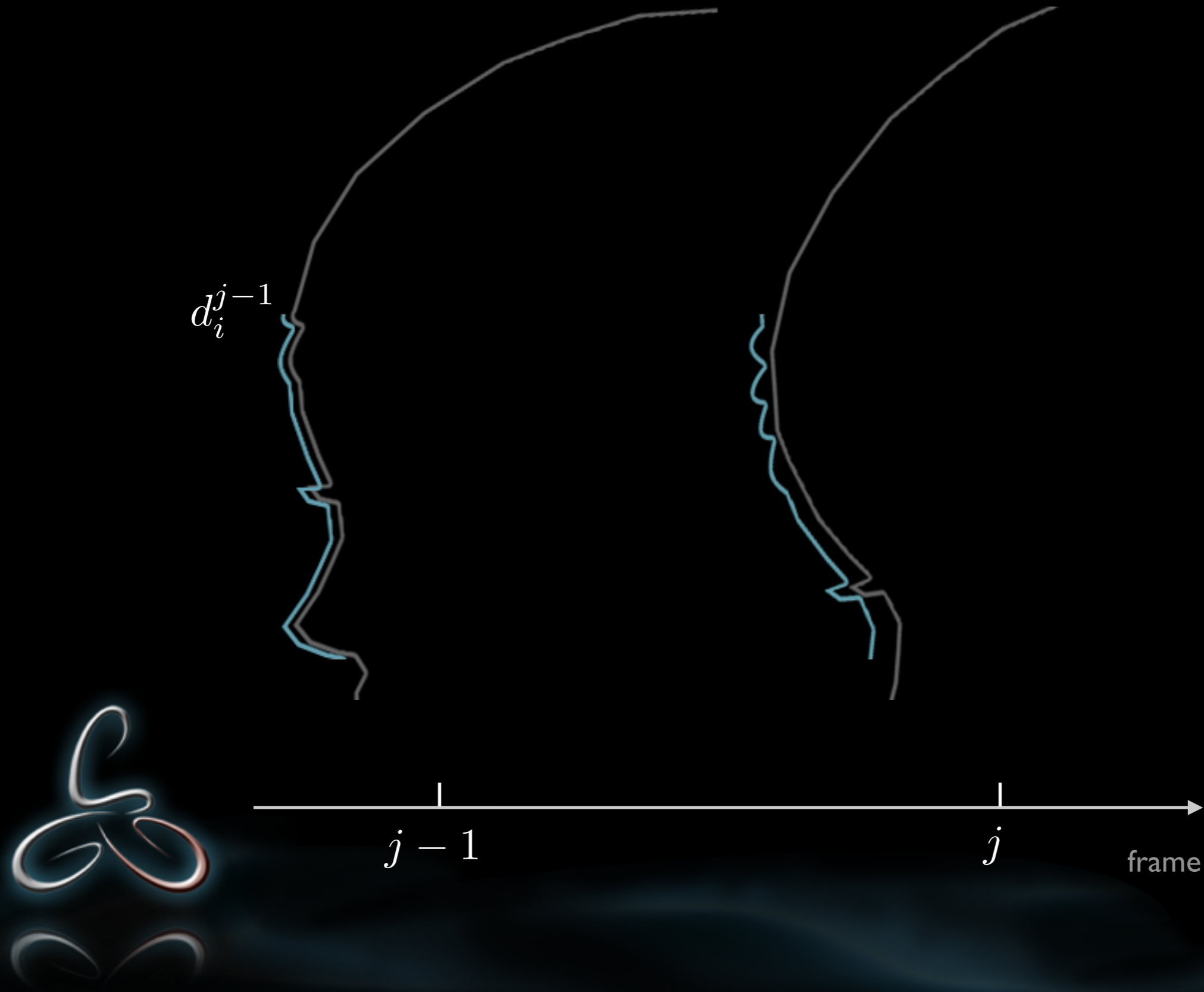
Detail Propagation



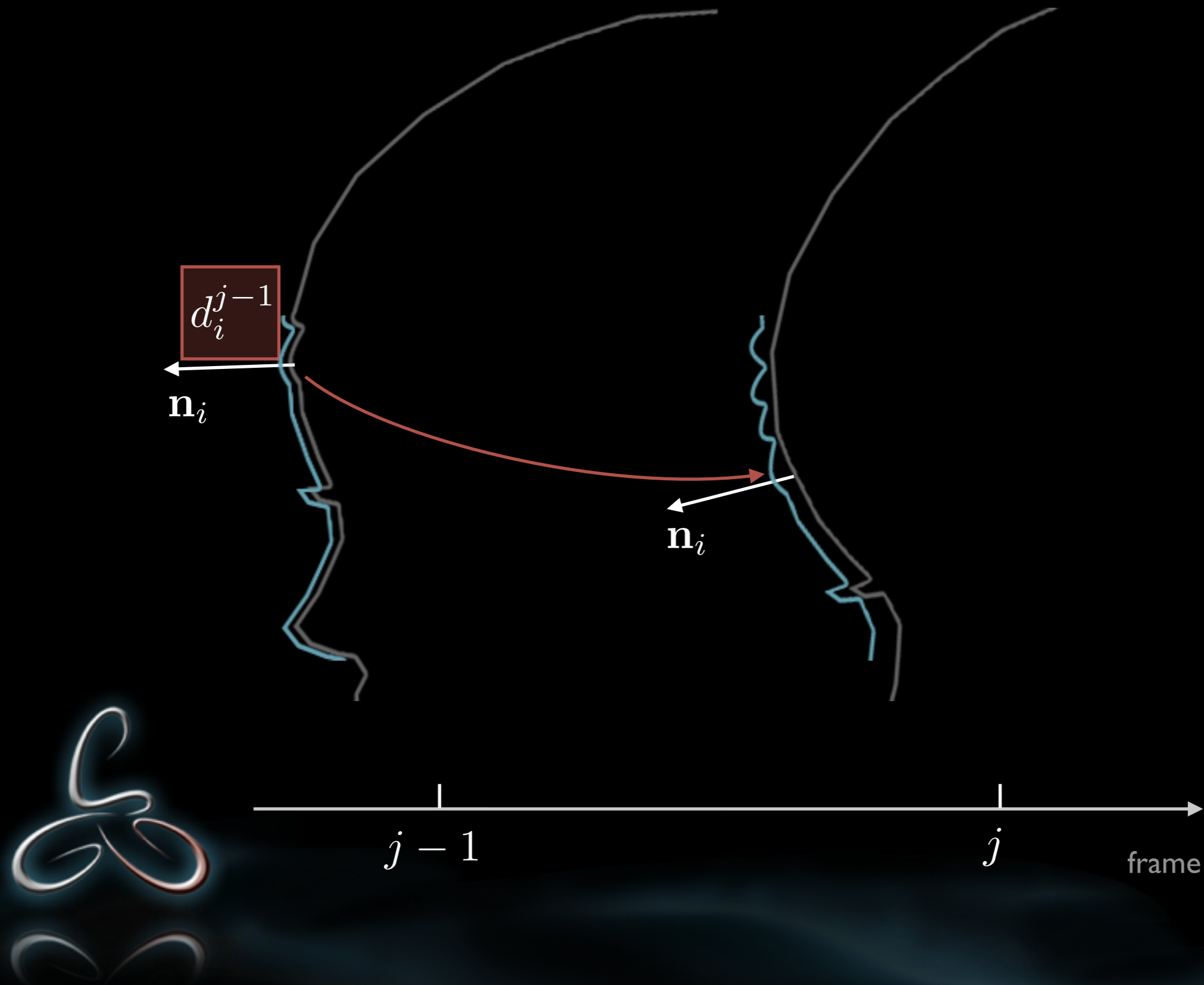
Detail Propagation



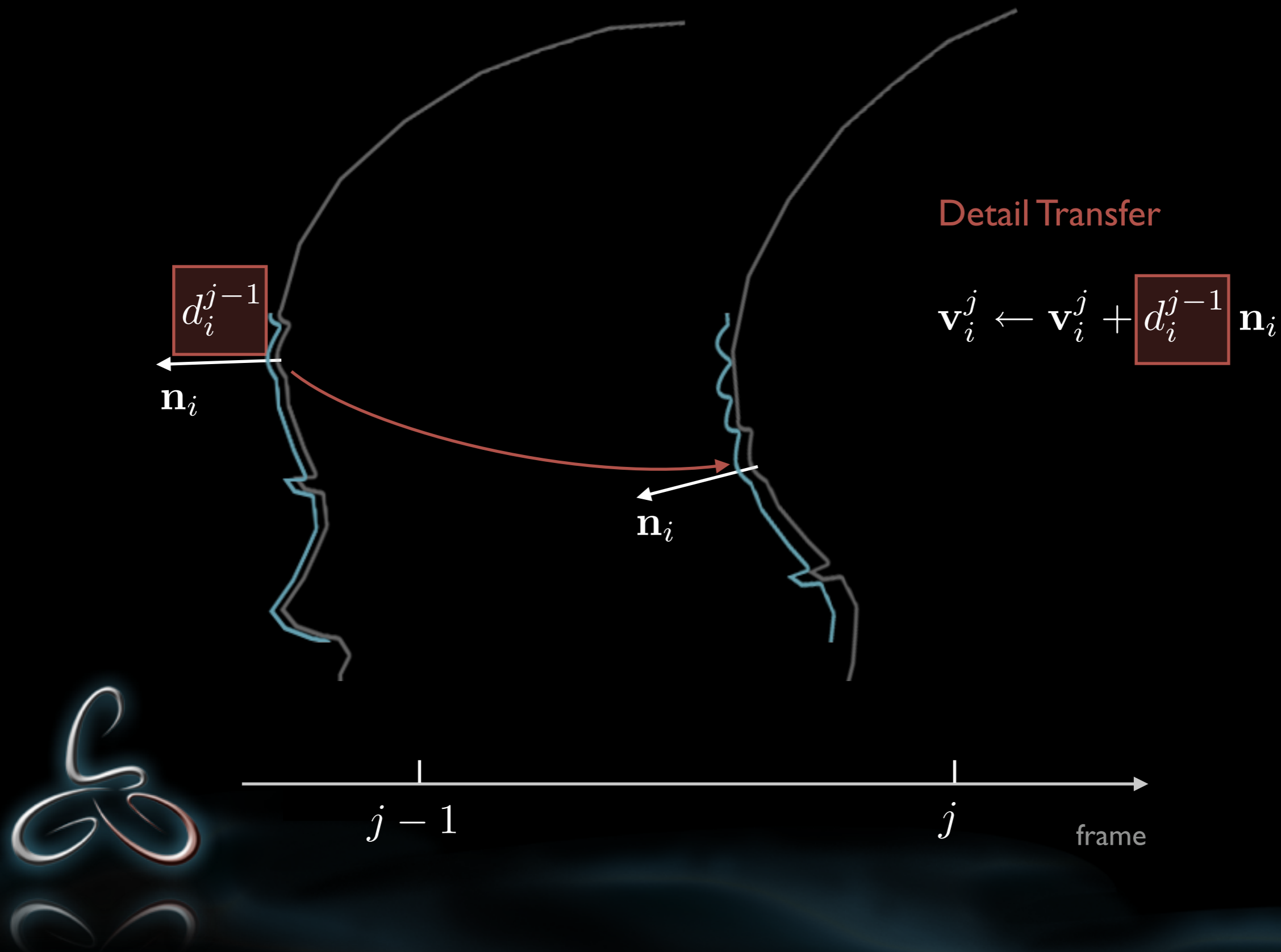
Detail Propagation



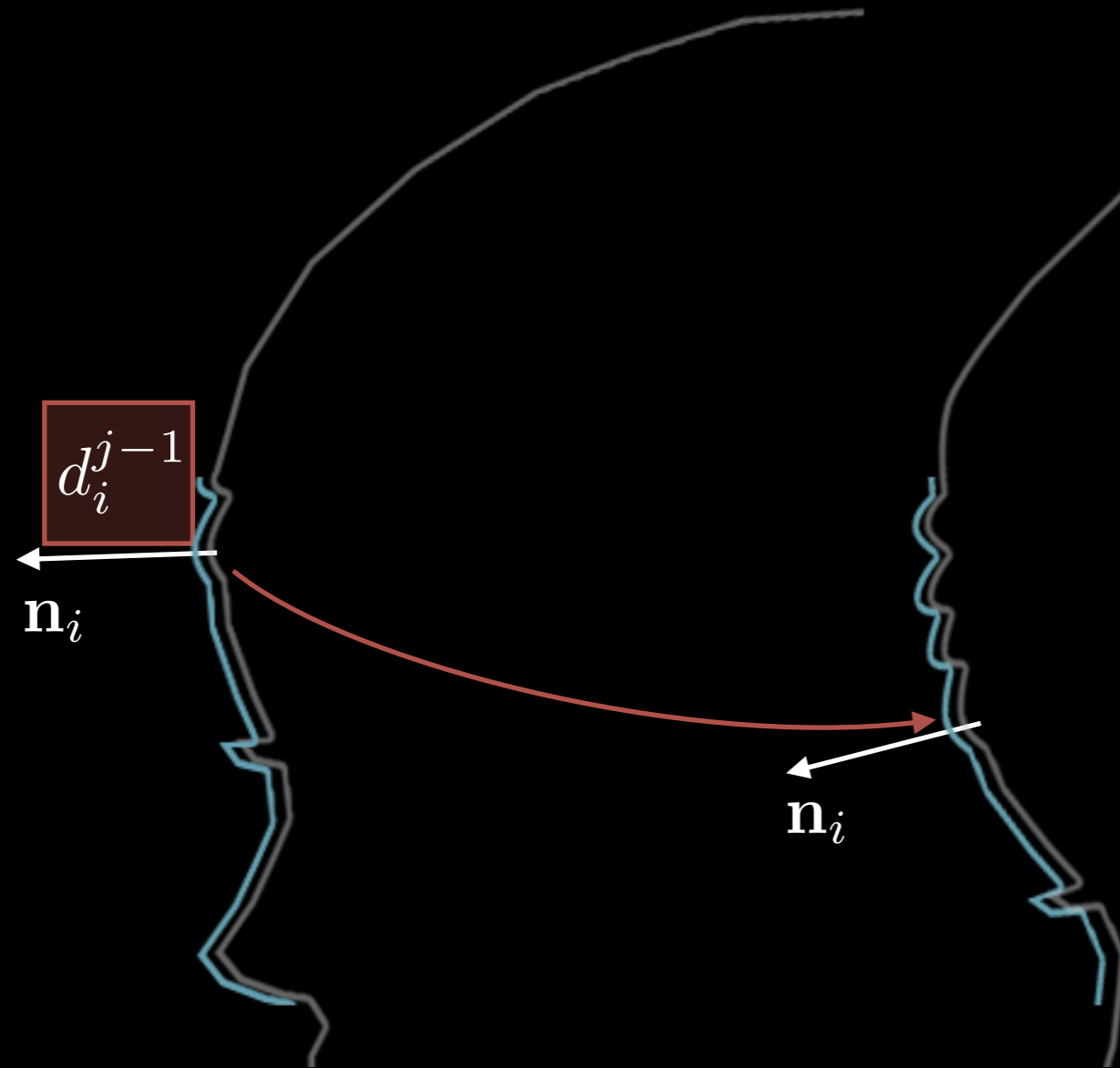
Detail Propagation



Detail Propagation



Detail Propagation

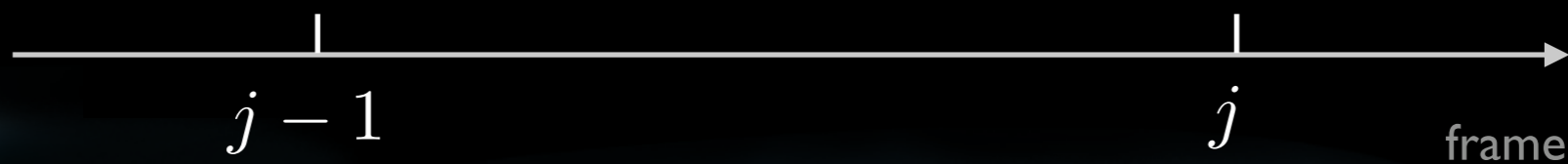


Detail Transfer

$$\mathbf{v}_i^j \leftarrow \mathbf{v}_i^j + \boxed{d_i^{j-1}} \mathbf{n}_i$$

$$\boxed{d_i^j} \leftarrow (1 - \gamma) \boxed{d_i^{j-1}} + \gamma \boxed{d_i^j}$$

Exponentially Weighted Moving Average



Forward-Backward Propagation

Coverage



Coverage

Forward-Backward Propagation



Coverage



Coverage



Multi-Frame Stabilization 1/4 x

Input Scans



Without
Stabilization

With
Stabilization

Input Data from [Park & Hodgins '06]

Multi-Frame Stabilization 1/4 x



Input Scans



Without
Stabilization



With
Stabilization



Input Data from [Park & Hodgins '06]

Multi-Frame Stabilization 1/4 x



Input Scans

Without
Stabilization

With
Stabilization



Input Data from [Park & Hodgins '06]

Multi-Frame Stabilization 1/4 x



Input Scans

Without
Stabilization



With
Stabilization

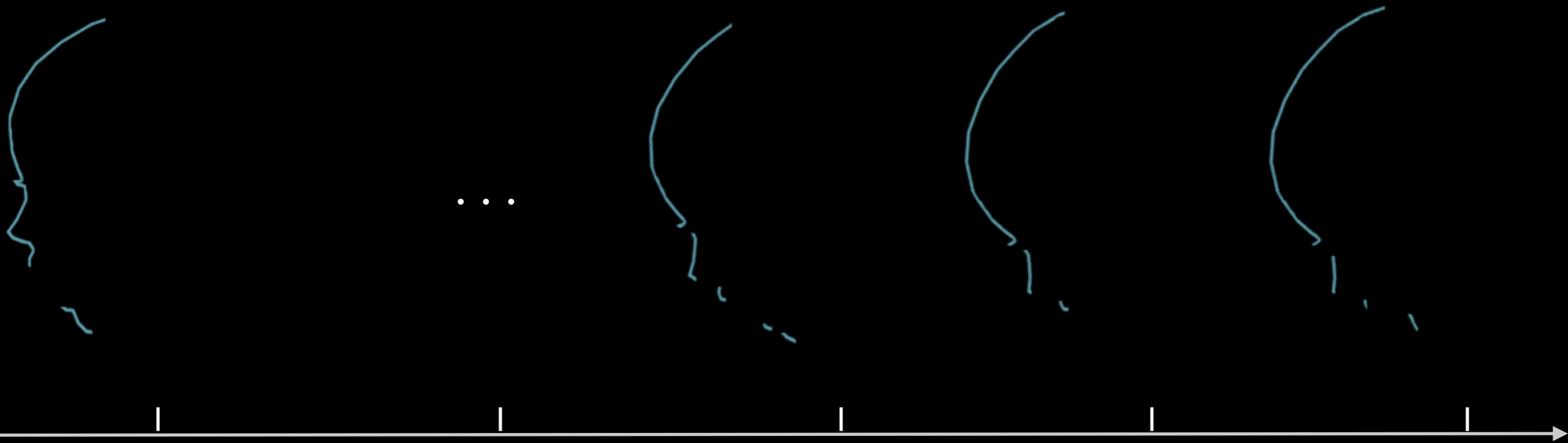


Input Data from [Park & Hodgins '06]

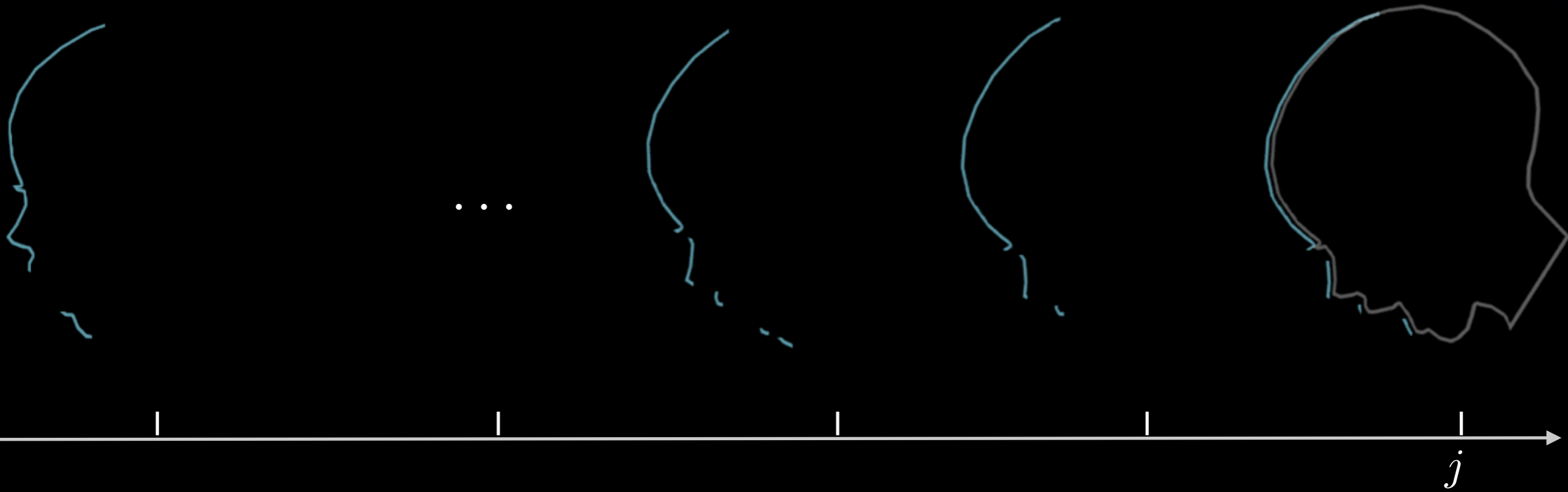
Hybrid Plastic-Elastic Model



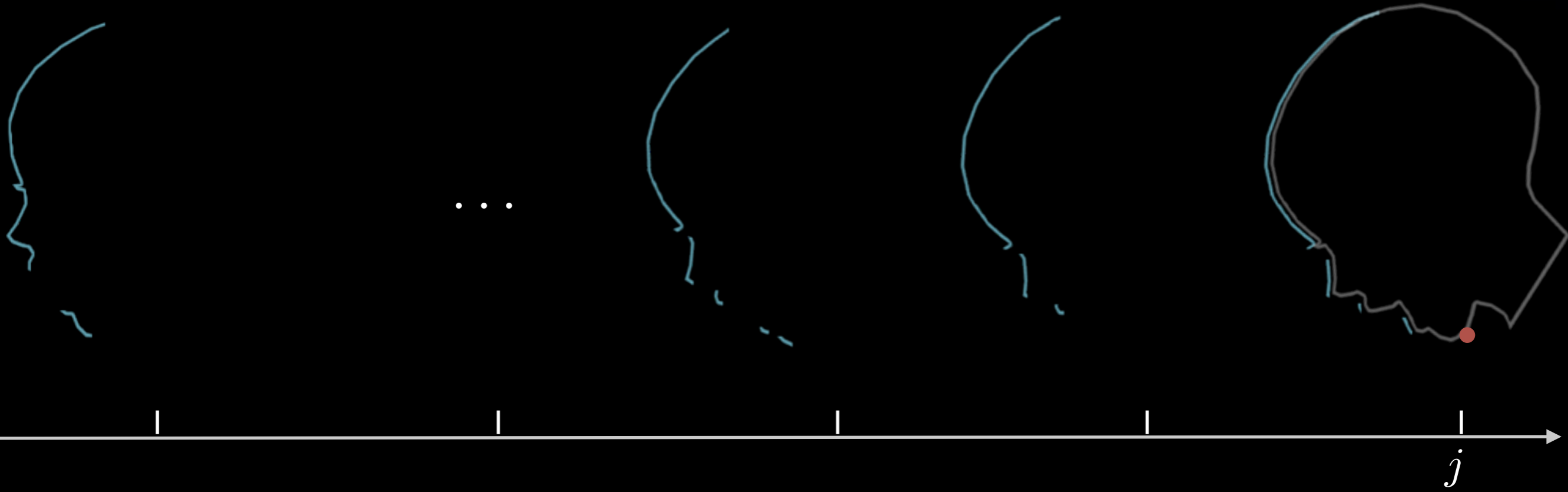
Hybrid Plastic-Elastic Model



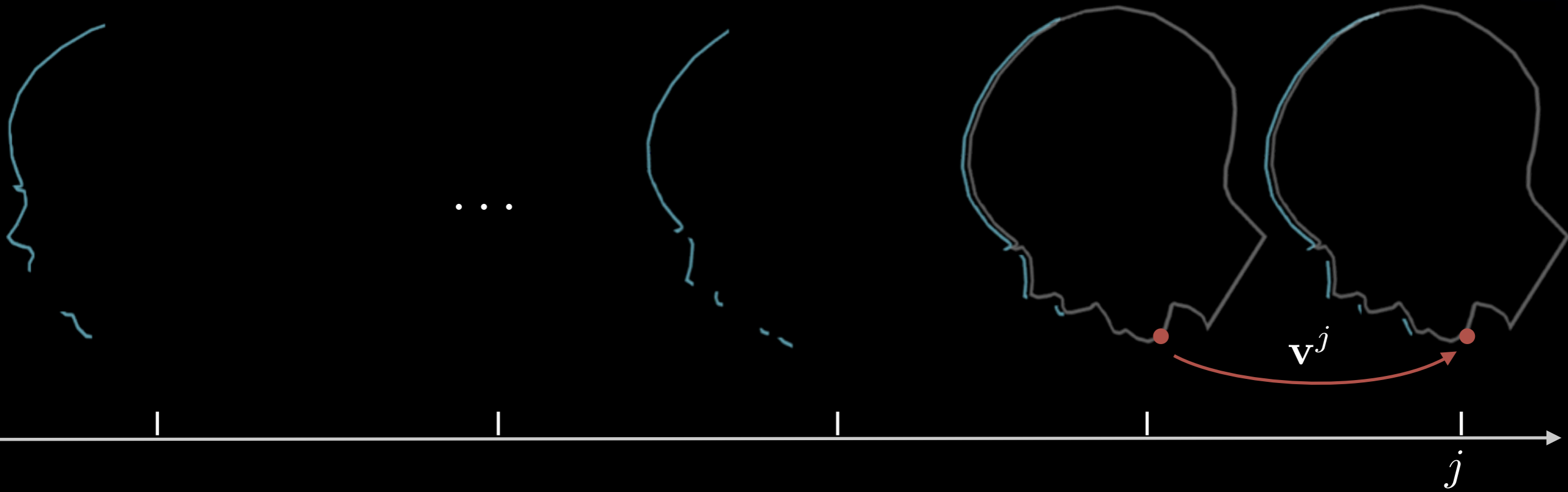
Hybrid Plastic-Elastic Model



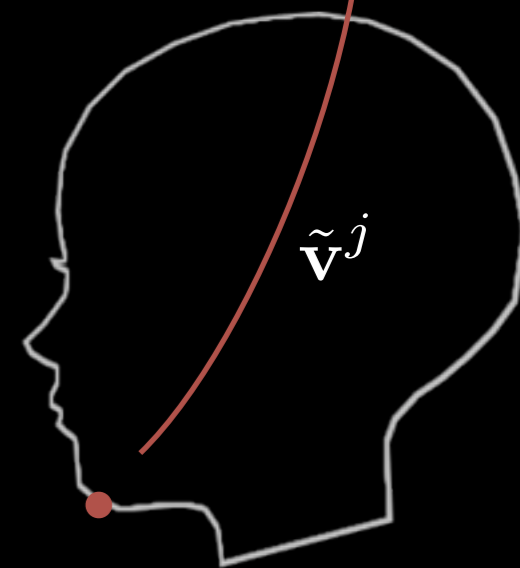
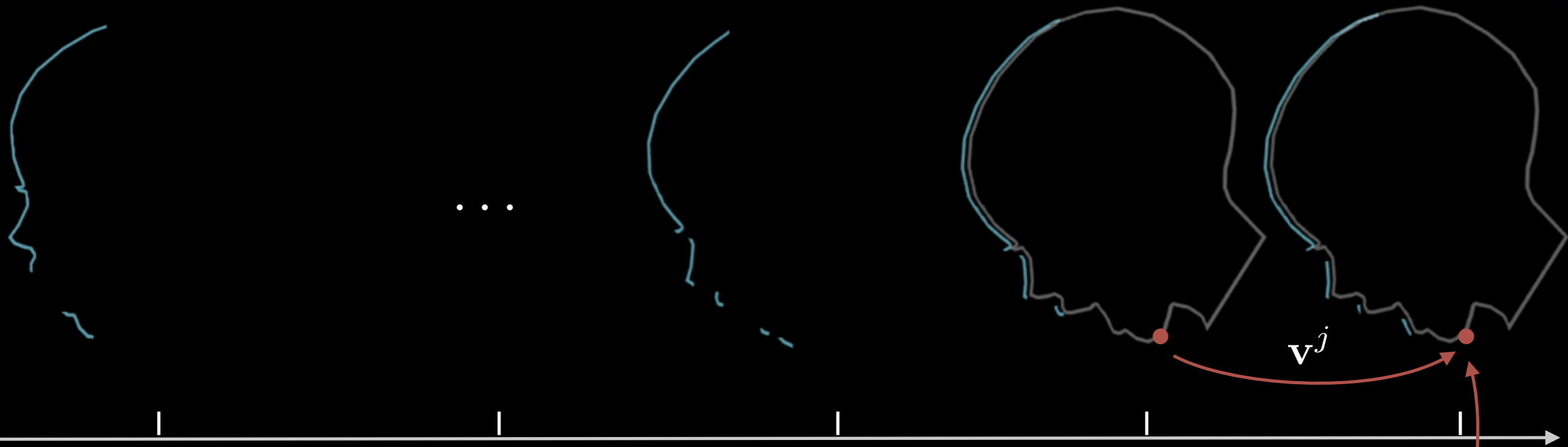
Hybrid Plastic-Elastic Model



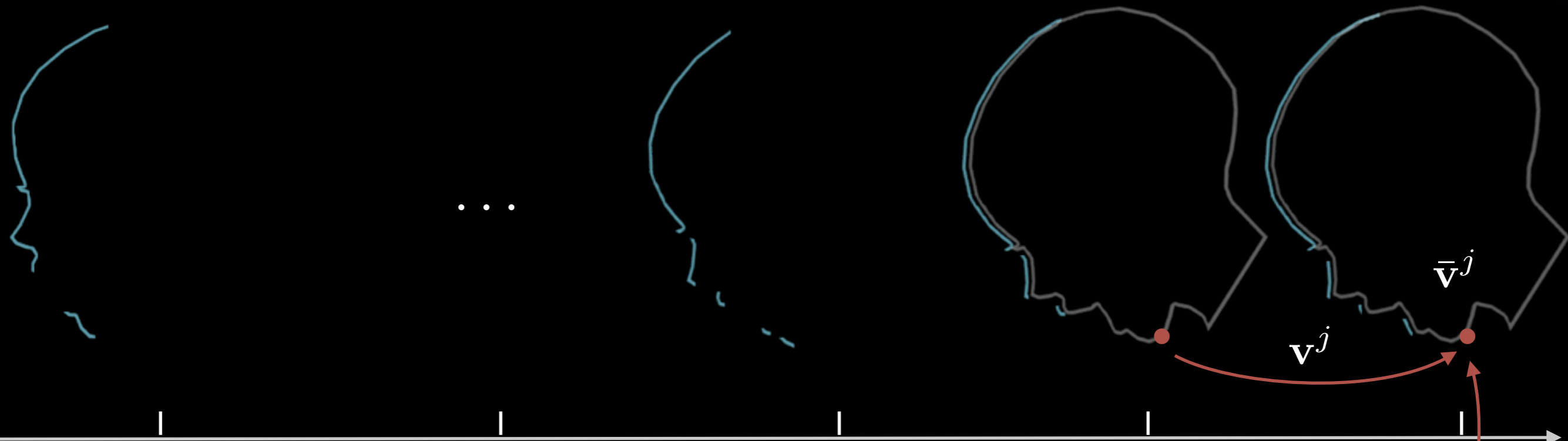
Hybrid Plastic-Elastic Model



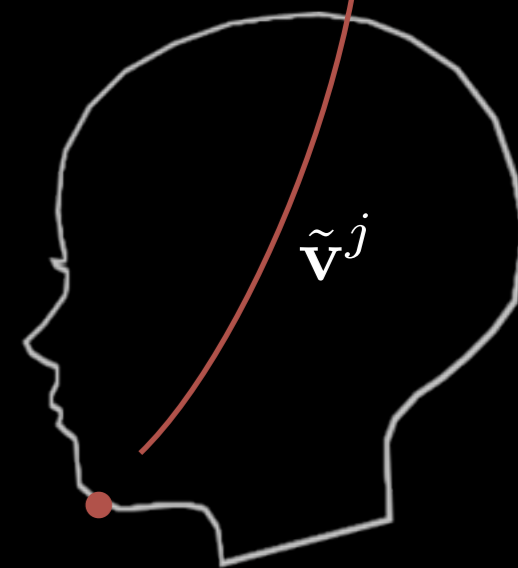
Hybrid Plastic-Elastic Model



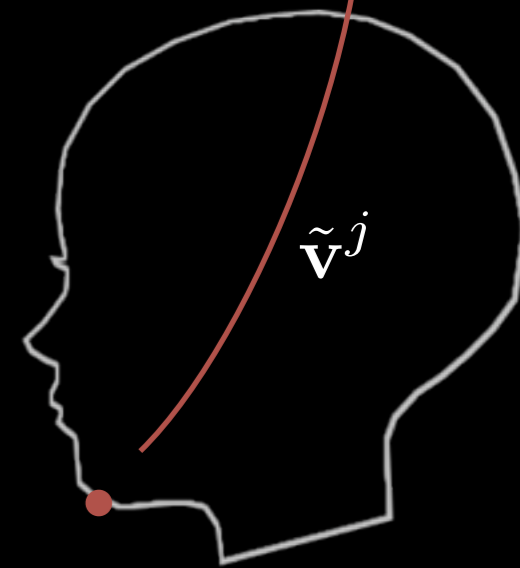
Hybrid Plastic-Elastic Model



$$\bar{\mathbf{v}}^j = c^j \mathbf{v}^j + (1 - c^j) \tilde{\mathbf{v}}^j$$



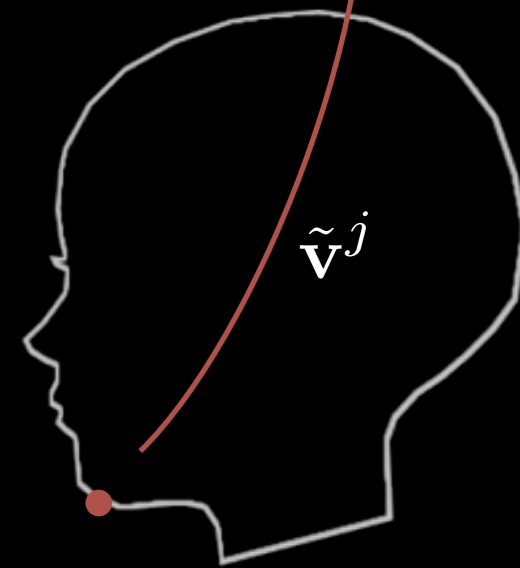
Hybrid Plastic-Elastic Model



Hybrid Plastic-Elastic Model



$$c^j = \max\left\{0, \frac{P + j_{\text{last}} - j}{P}\right\}$$



Reconstruction Process

The Puppet



3D Acquisition – 100 Frames

Input Scan Sequence



3D Acquisition – 100 Frames



Input Scan Sequence



Initial Registration

Coarse Template



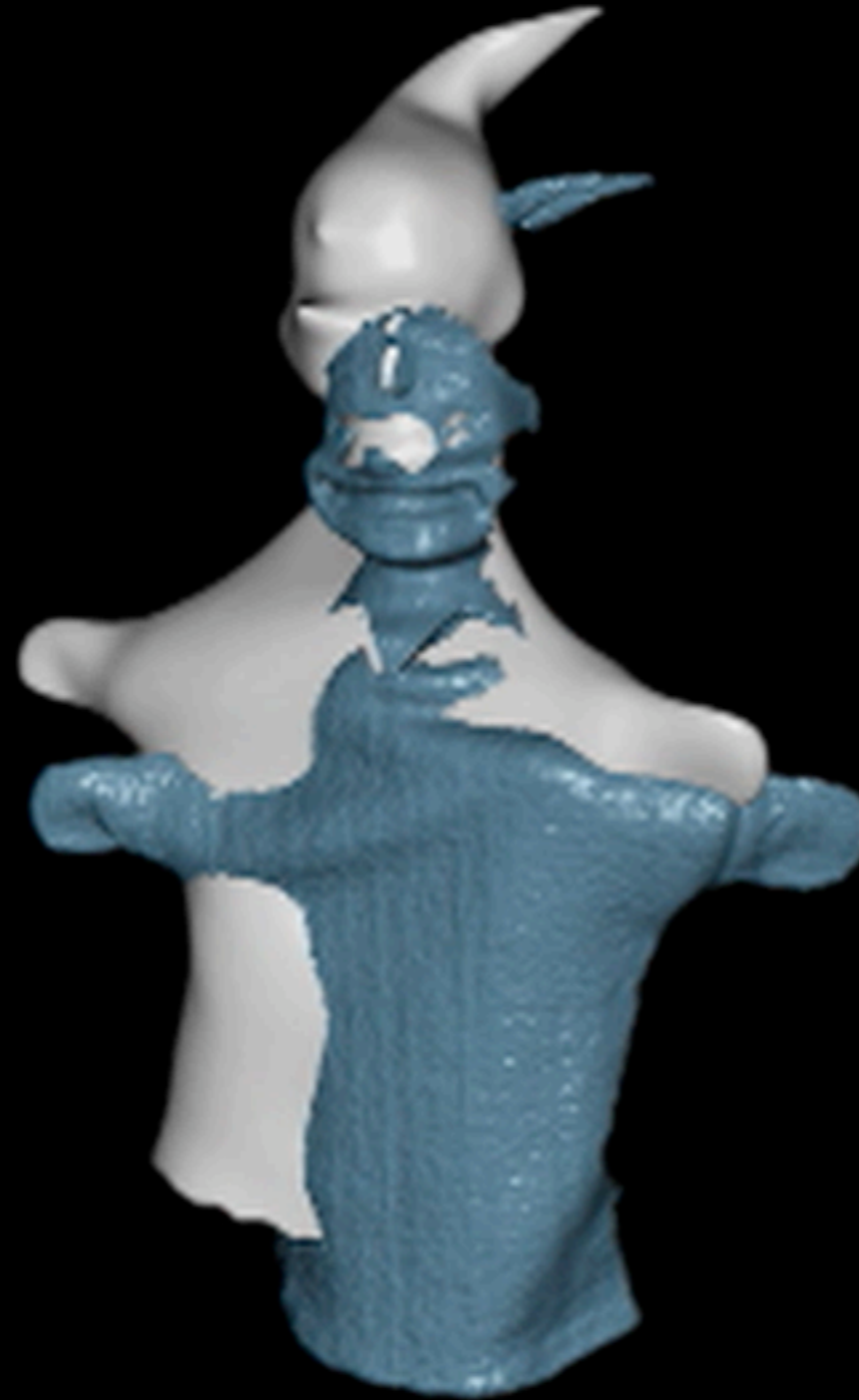
Initial Registration



Coarse Template



Initial Registration



Coarse Template

First Scan



Template **Warping**

Input Scans



Warped Template

Template **Warping**



Input Scans



Warped Template



Final Reconstruction – 100 Frames



Input Scans

Reconstruction

Final Reconstruction – 100 Frames



Input Scans



Reconstruction



Correspondence Visualization

Reconstruction



Textured Reconstruction

Correspondence Visualization



Reconstruction



Textured Reconstruction



Detail-Coefficient Stability

Reconstruction

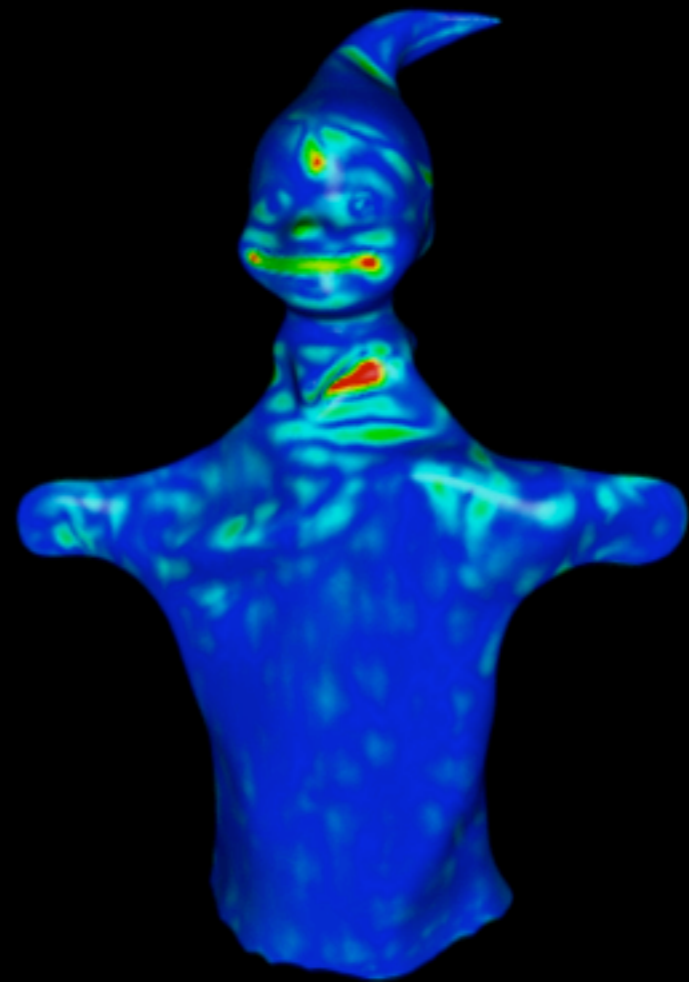
Detail Coefficients



Detail-Coefficient Stability



Reconstruction

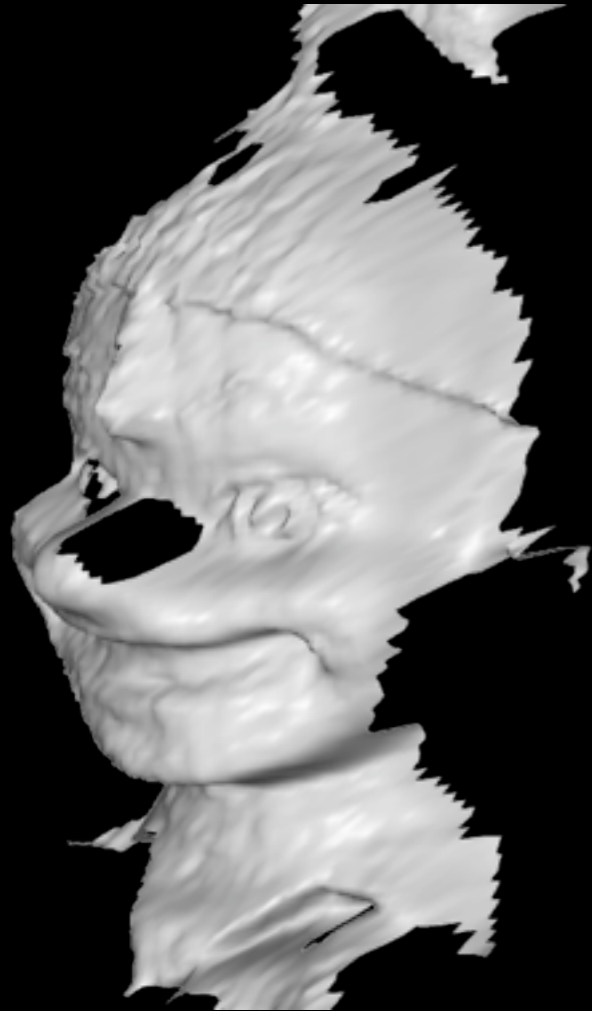


Detail Coefficients



Close-up Comparison

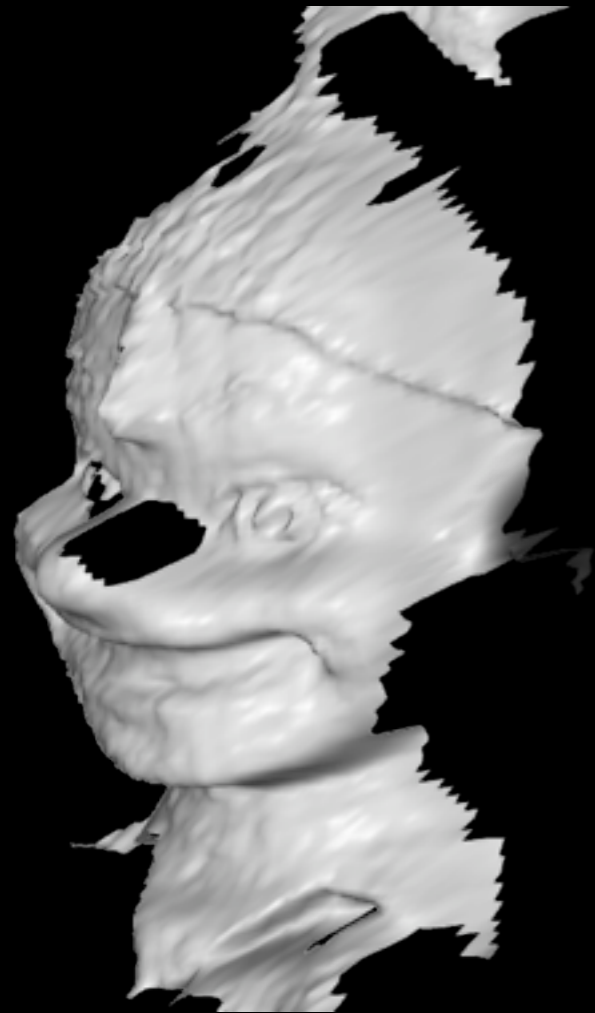
Input Scan



Close-up Comparison

Input Scan

Warped Template

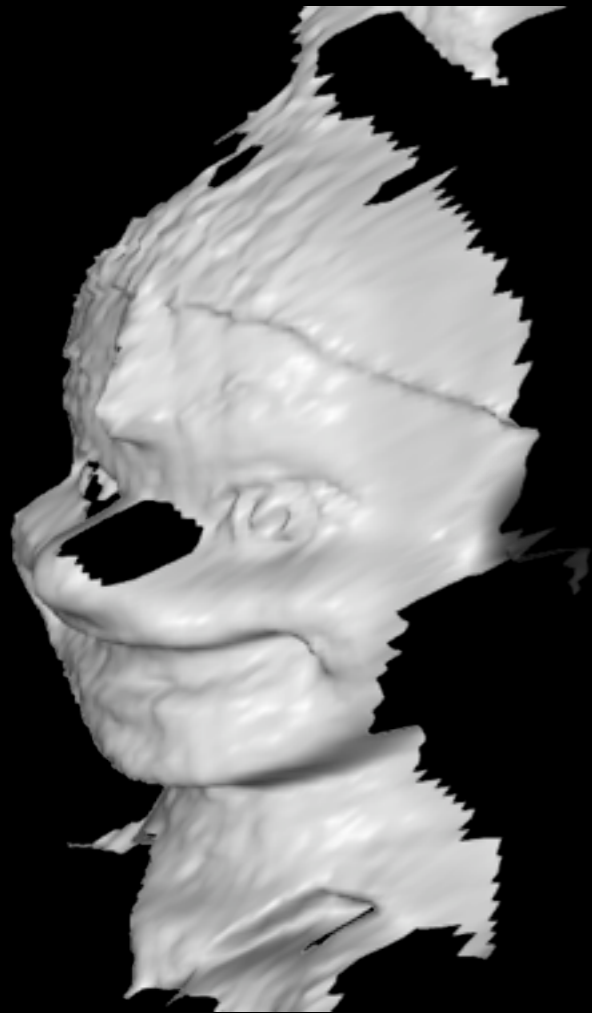


Close-up Comparison

Input Scan

Warped Template

Reconstruction



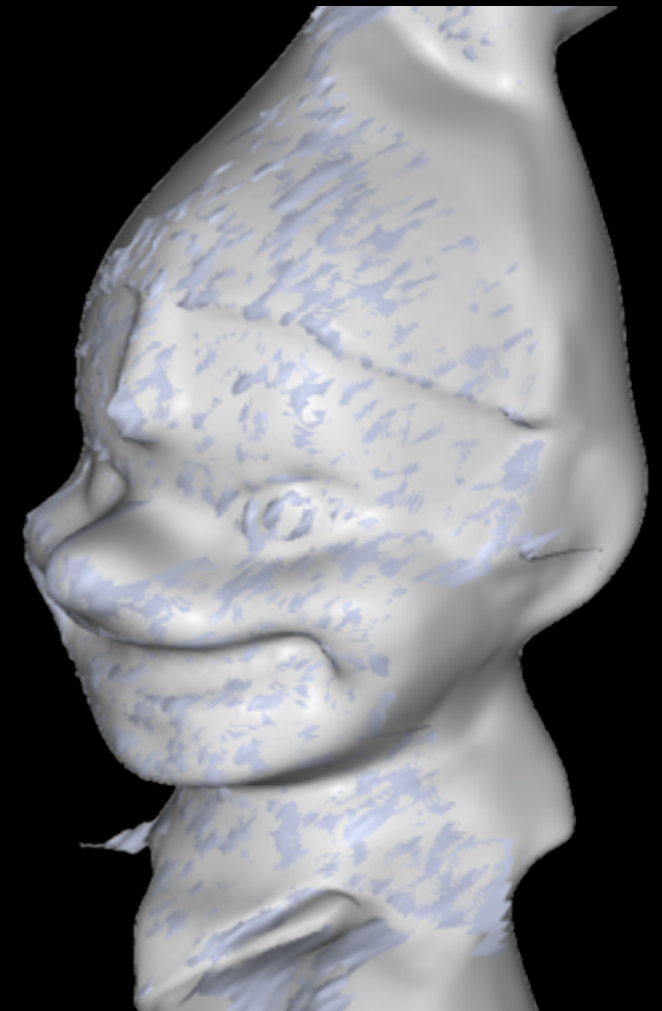
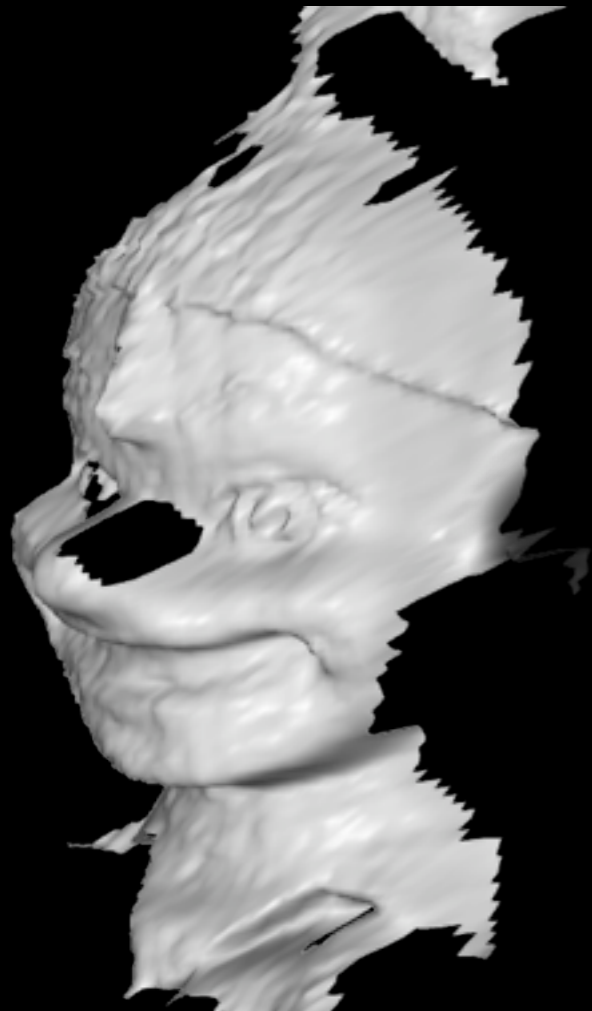
Close-up Comparison

Input Scan

Warped Template

Reconstruction

Overlaid Scan



More Results



Grasping Hand – 34 Frames

Input Scans



Reconstruction

Textured Reconstruction

Grasping Hand – 34 Frames



Input Scans



Reconstruction



Textured Reconstruction



Crumpling Paper Bag – 85 Frames

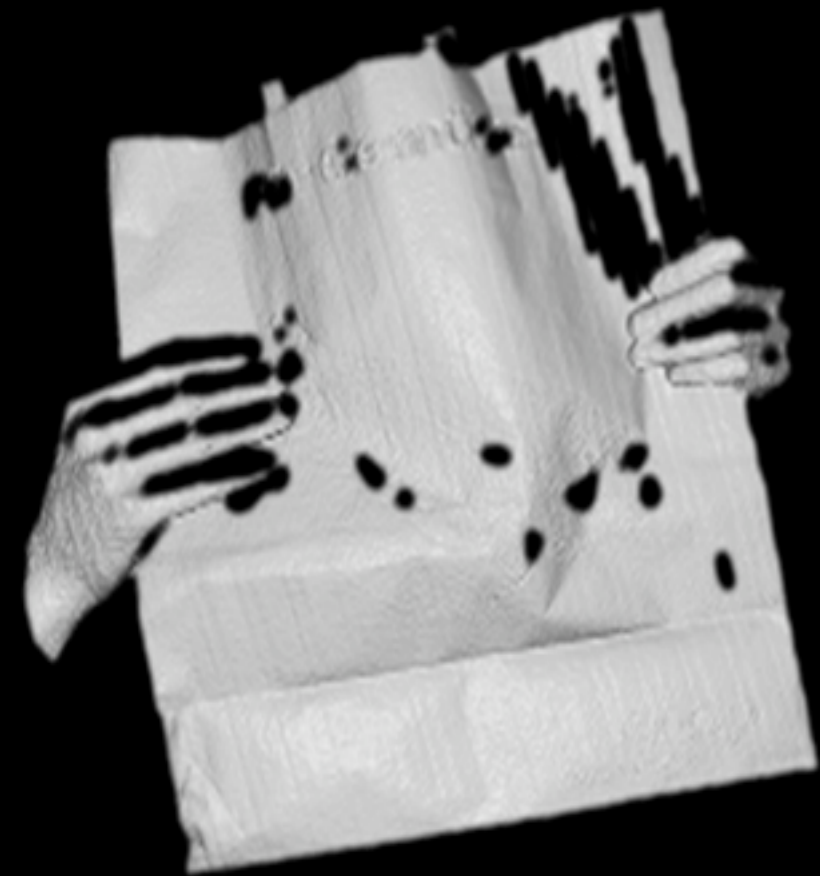
Input Scans



Reconstruction

Textured Reconstruction

Crumpling Paper Bag – 85 Frames



Input Scans



Reconstruction



Textured Reconstruction



Facial Expressions – 200 Frames

Input Scans

Reconstruction

Overlaid Scans



Facial Expressions – 200 Frames



Input Scans



Reconstruction



Overlaid Scans

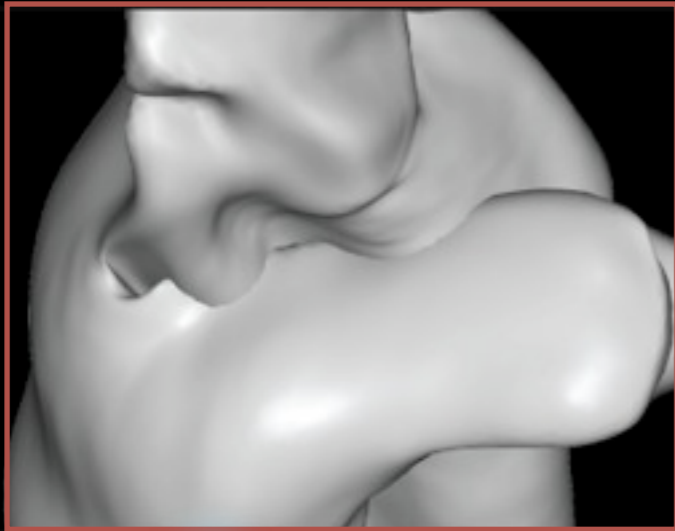
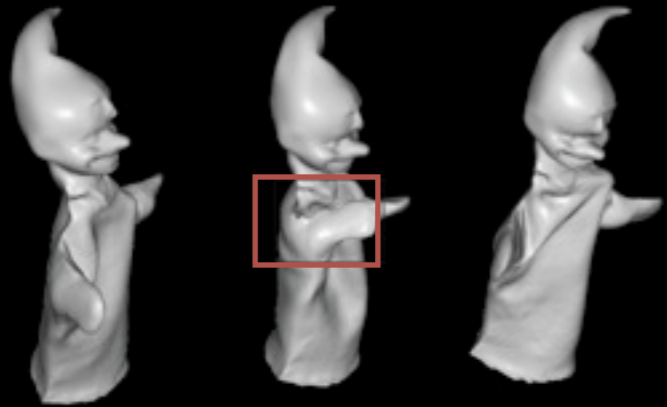


Limitations



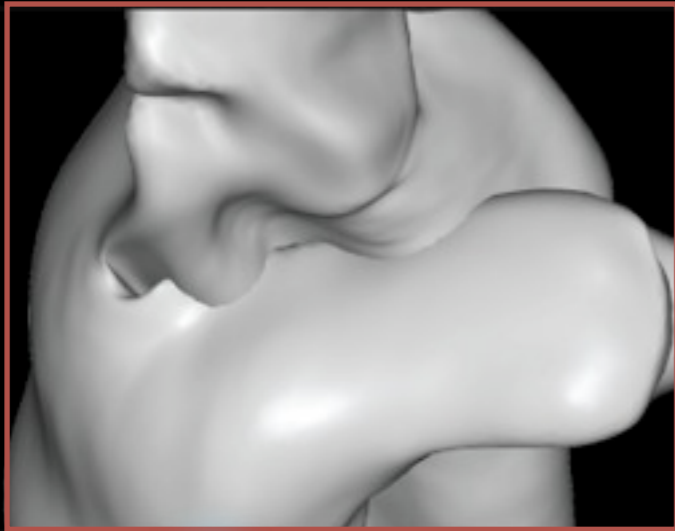
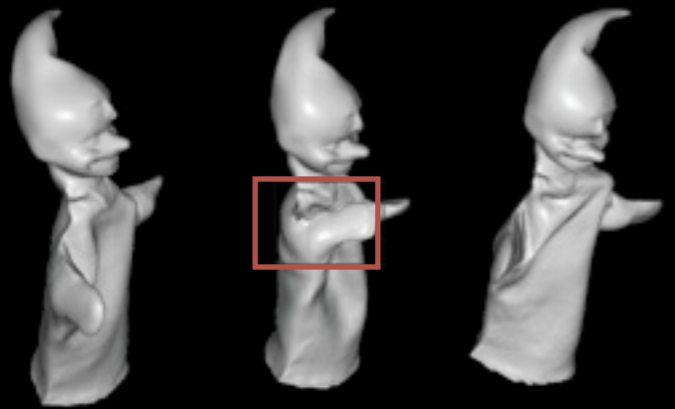
Limitations

Self-Intersection

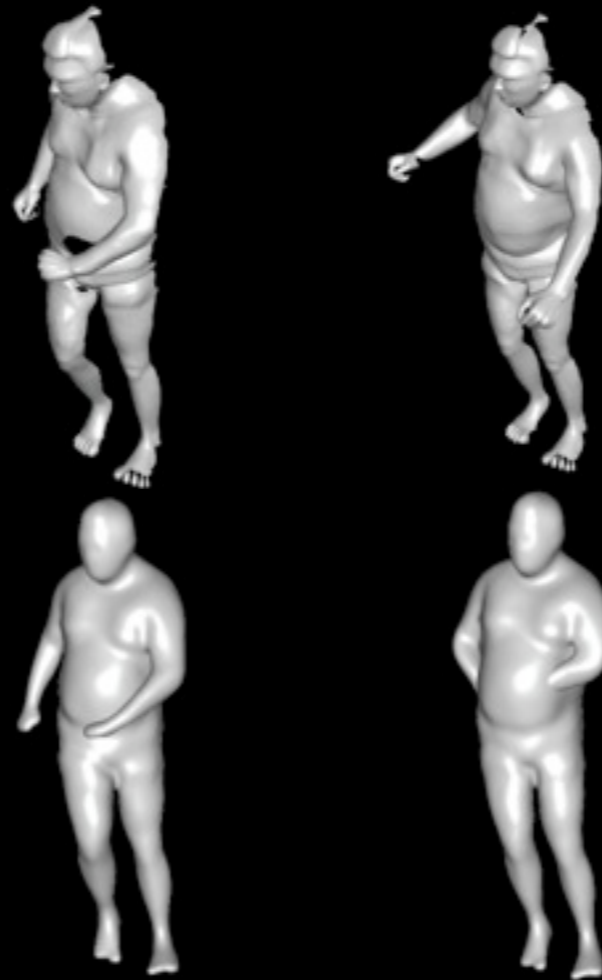


Limitations

Self-Intersection

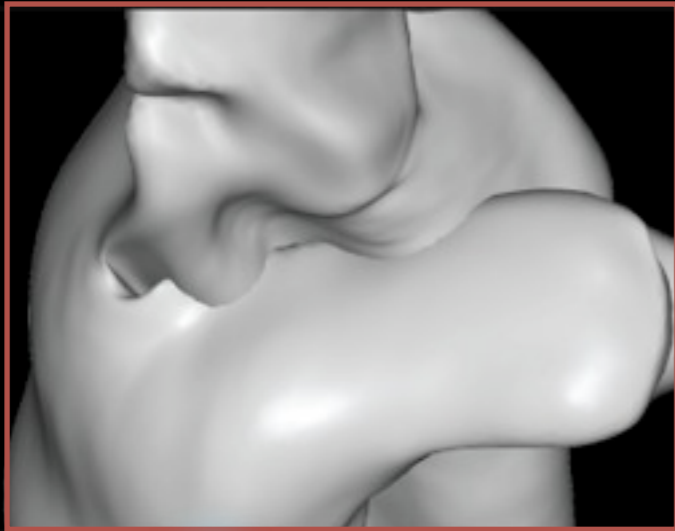
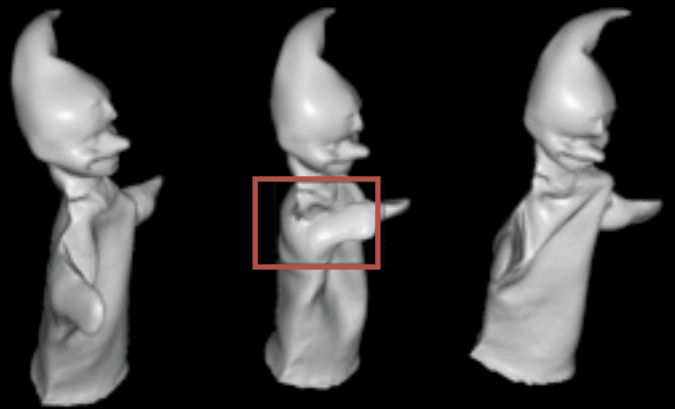


Large Motion



Limitations

Self-Intersection



Large Motion



Varying Topology



What's Next?

Multi-View and Textures



[Vlasic et al. '09], Princeton, ICT



What's Next?

Multi-View and Textures



[Vlasic et al. '09], Princeton, ICT

Complex Materials



National Geographic



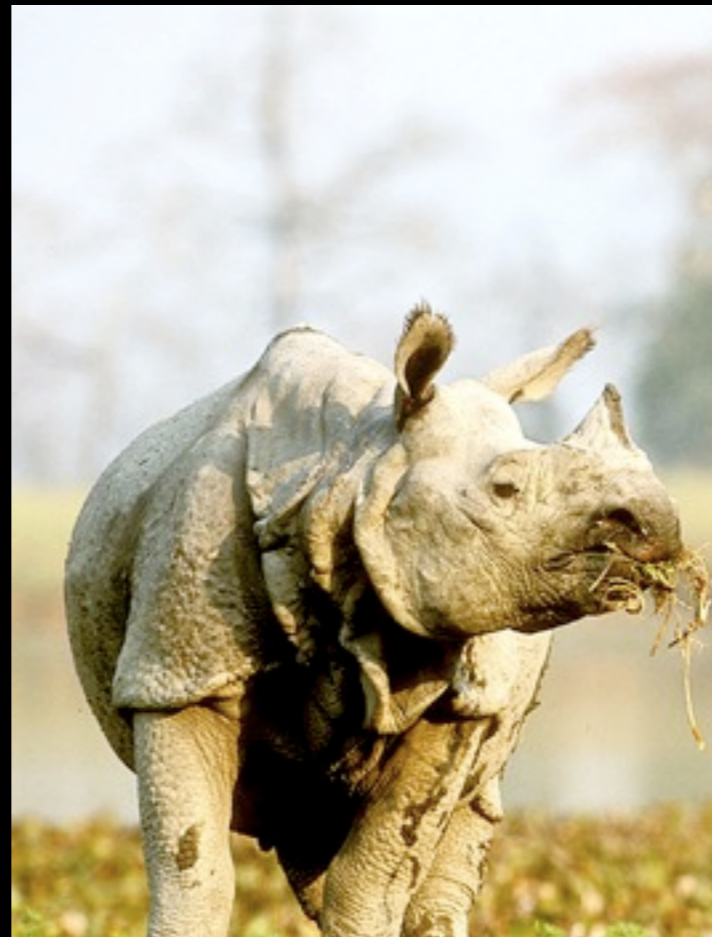
What's Next?

Multi-View and Textures



[Vlasic et al. '09], Princeton, ICT

Complex Materials



National Geographic

Surface Segmentation



WordPress



www.hao-li.com



www.hao-li.com

