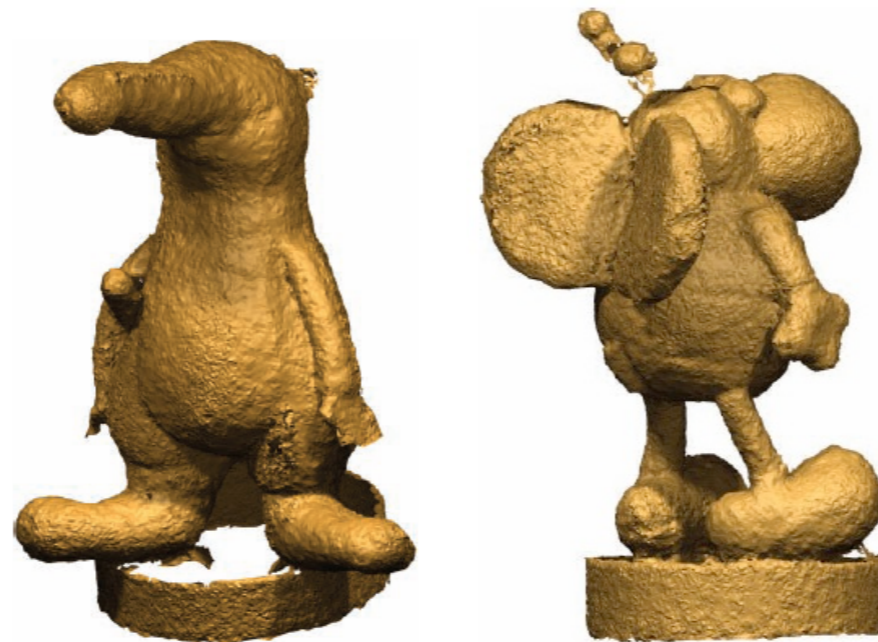


# 13.2 **Dynamic Geometry Processing II**



Hao Li

<http://cs621.hao-li.com>

Prof. Niloy J. Mitra  
UCL



# Registration without Correspondences

# Scan Registration

**Bee**

Input frames (Selection)

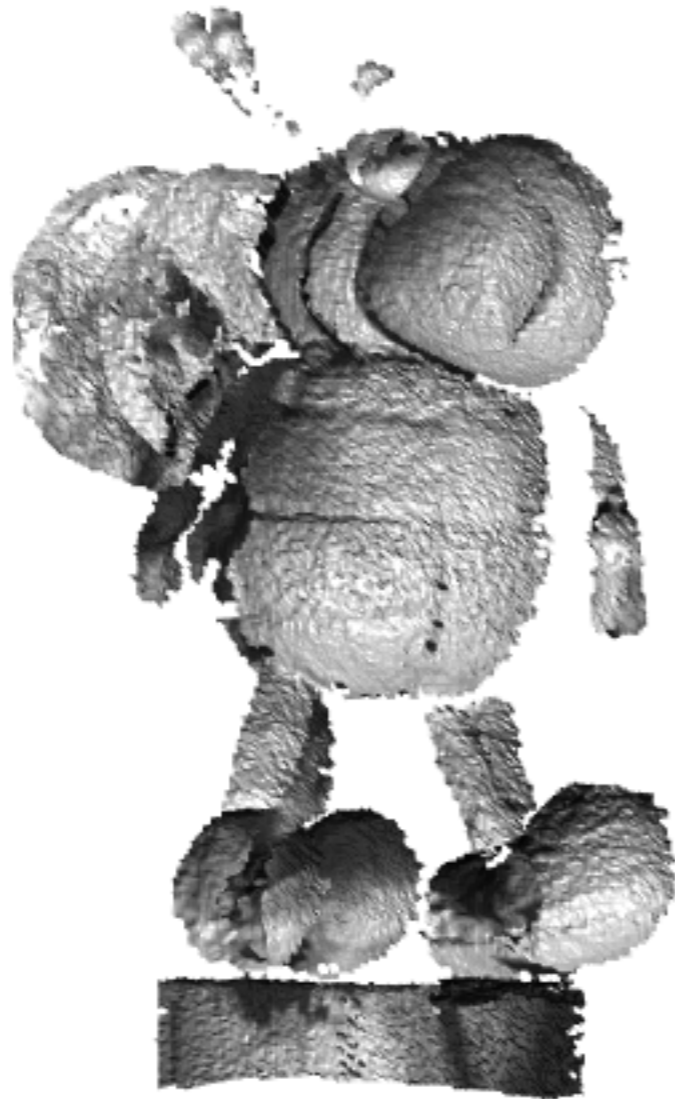
2200 pointclouds scanned at 17 Hz

transformations only for adjacent frames considered

no global error correction

no noise smoothing

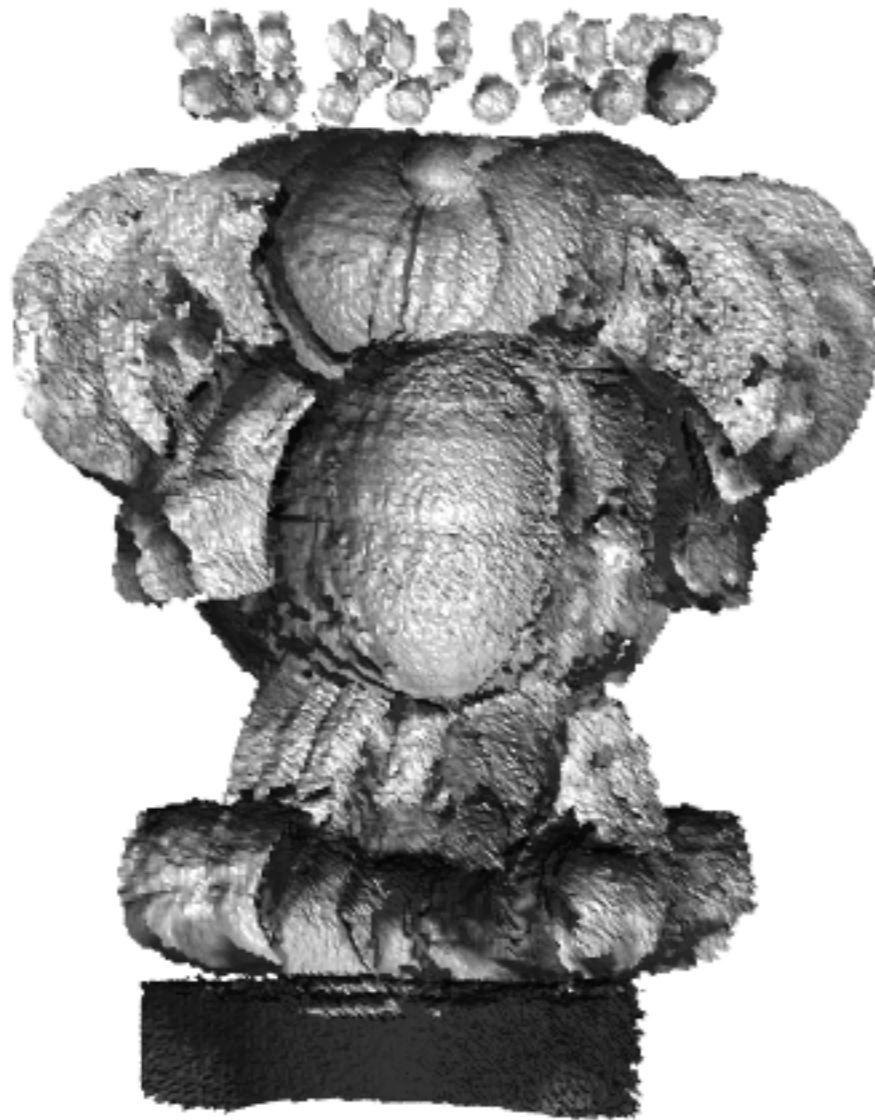
# Scan Registration



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

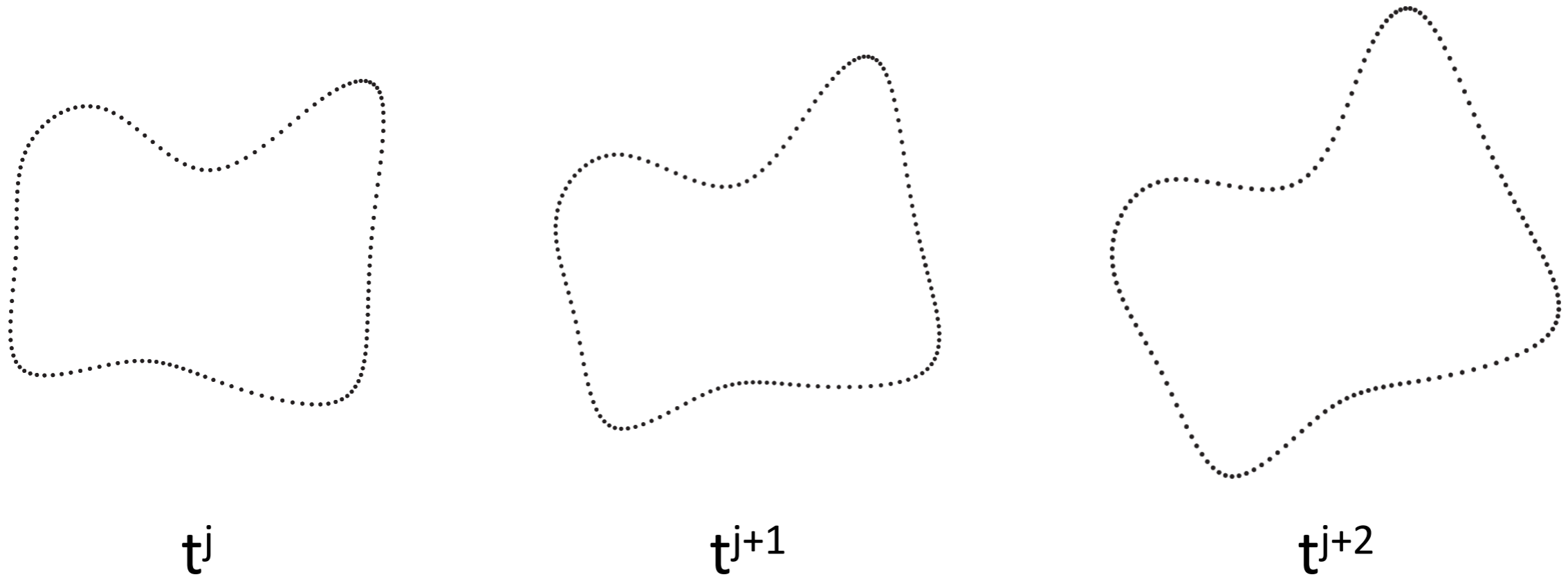
- Smoothly varying object motion
- Unknown correspondence between scans
- Fast acquisition !  
motion happens between frames

# Contributions

- Rigid registration ! kinematic property of space-time surface (locally exact)
- Registration ! surface normal estimation
- Analysis (see paper for details)
  - Relation to ICP (Iterated Closest Point)
  - Stability analysis
- Extension to deformable/articulated bodies

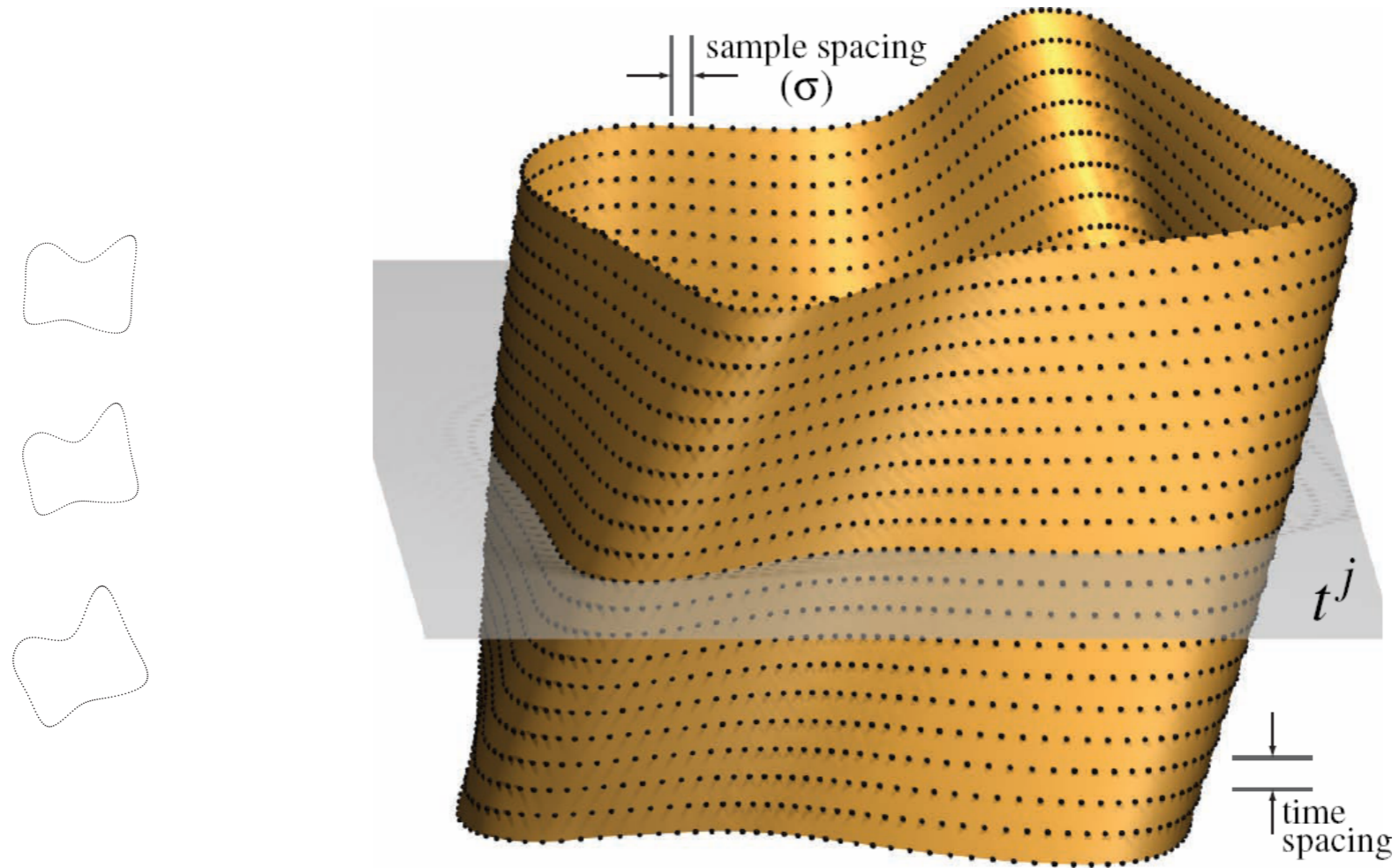


# Time Ordered Scans

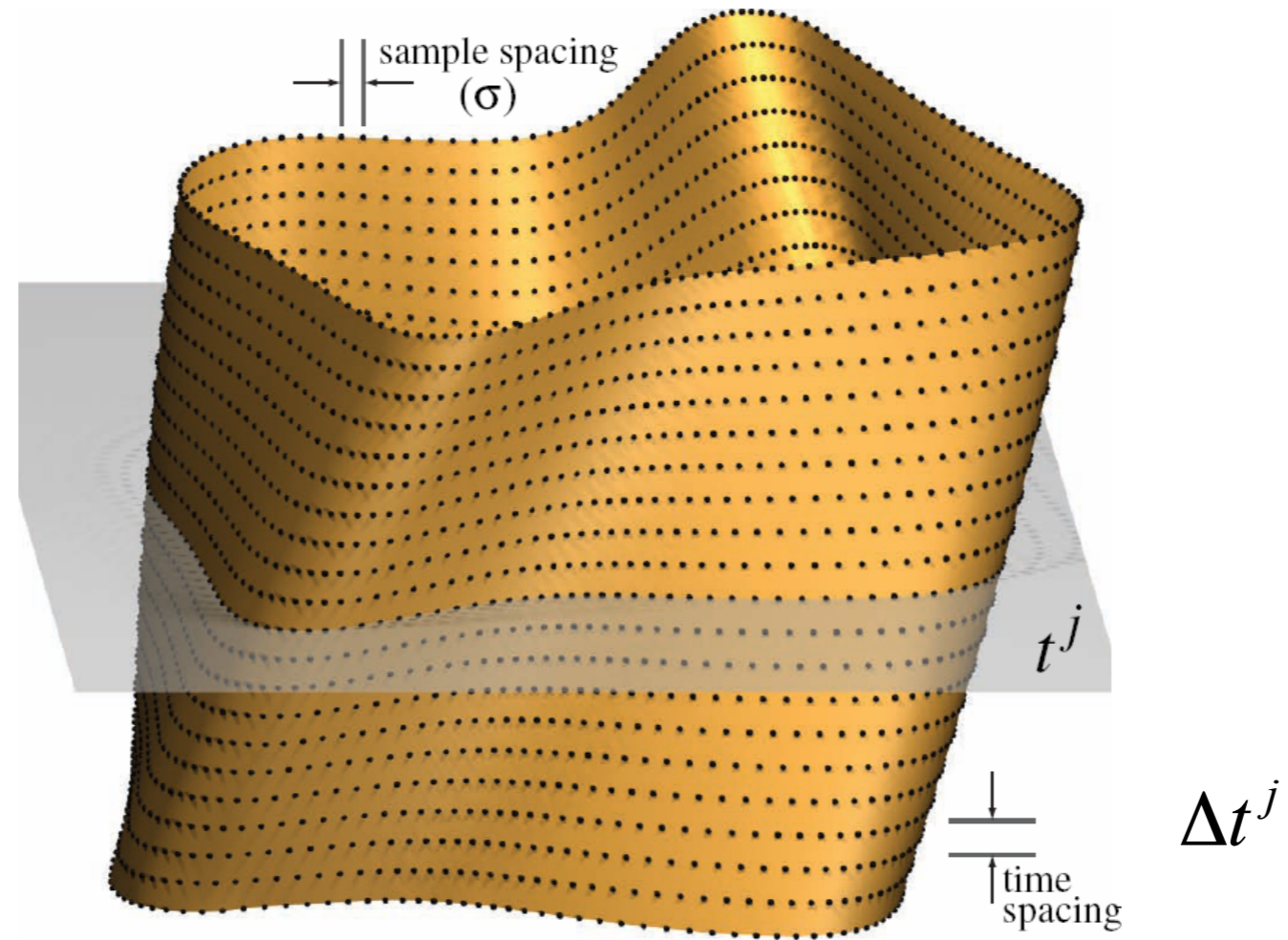


$$\tilde{P}^j \equiv \{\tilde{\mathbf{p}}_i^j\} := \{(\mathbf{p}_i^j, t^j), \mathbf{p}_i^j \in \mathbb{R}^d, t^j \in \mathbb{R}\}$$

# Space-Time Surface



# Space-Time Surface

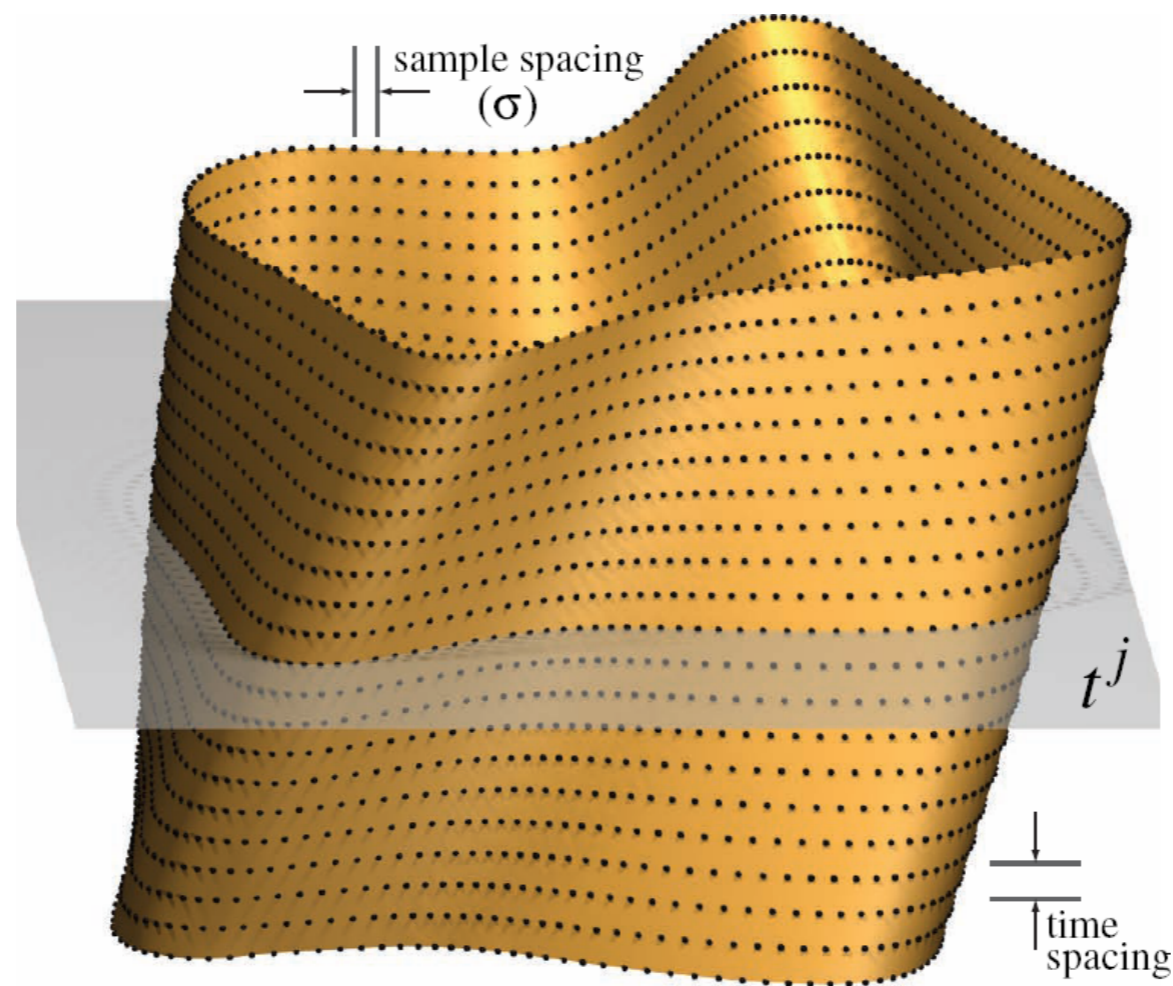


$\tilde{\mathbf{p}}_i^j$

!

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

# Space-Time Surface

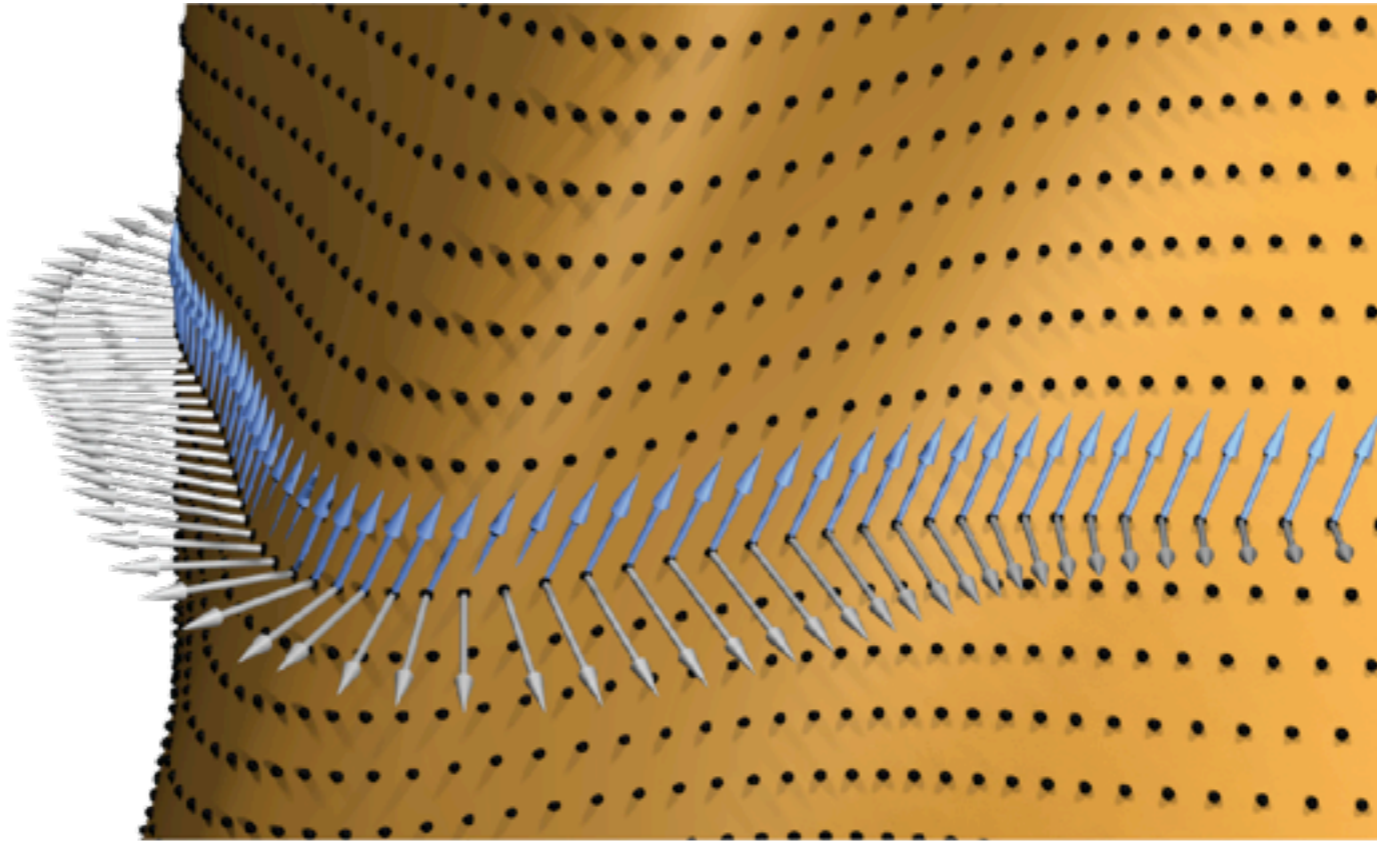


$\tilde{\mathbf{p}}_i^j$

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

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

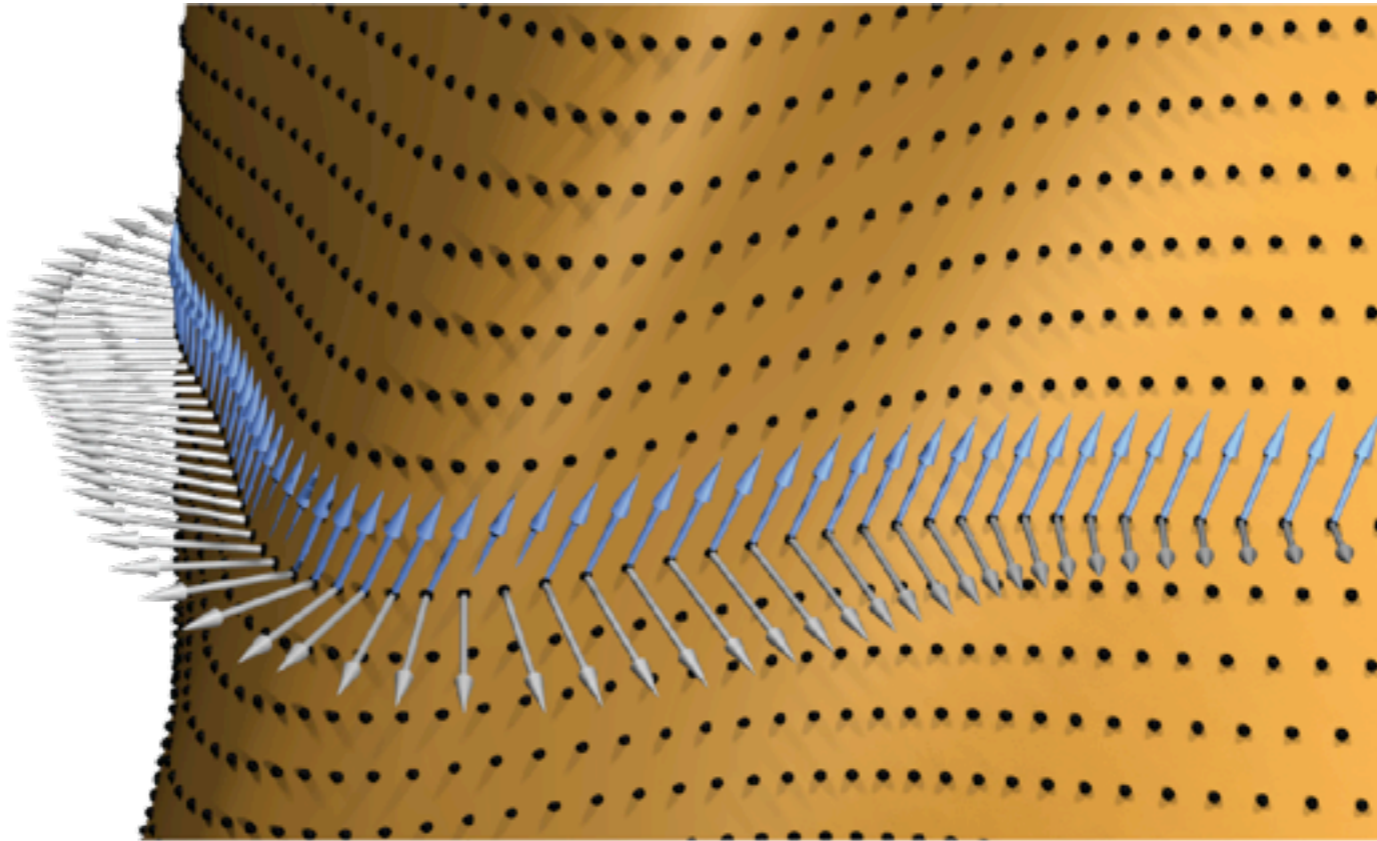
# Space-Time Velocity Vectors



Tangential point movement ! velocity vectors orthogonal to surface normals

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

# Space-Time Velocity Vectors



Tangential point movement ! velocity vectors orthogonal to surface normals

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

# Final Steps

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

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

# Final Steps

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

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

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

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

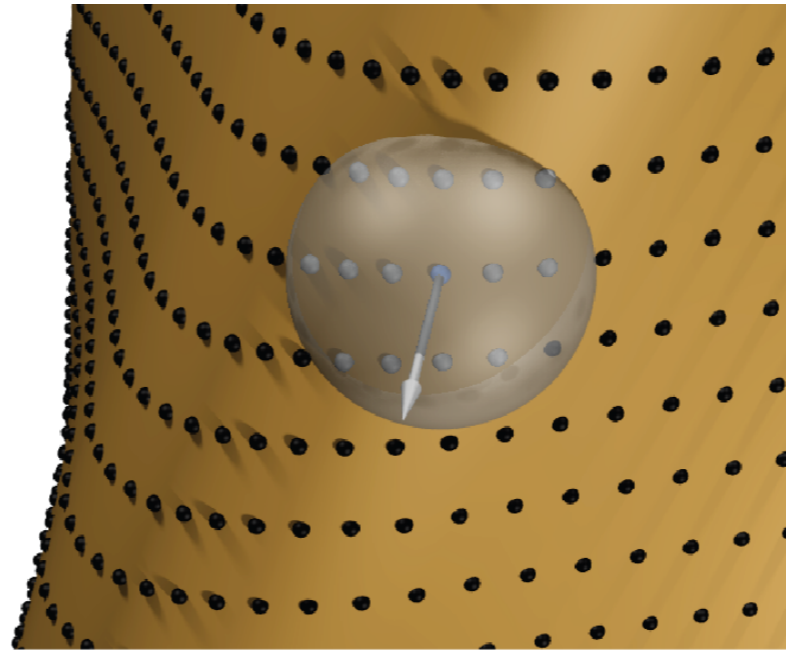
$$\mathbf{b} = \sum_{i=1}^{|P^j|} w_i^j n_i^j \begin{bmatrix} \tilde{\mathbf{n}}_i^j \\ \mathbf{p}_i^j \times \tilde{\mathbf{n}}_i^j \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \bar{\mathbf{c}}_j \\ \mathbf{c}_j \end{bmatrix}$$



# Registration Algorithm

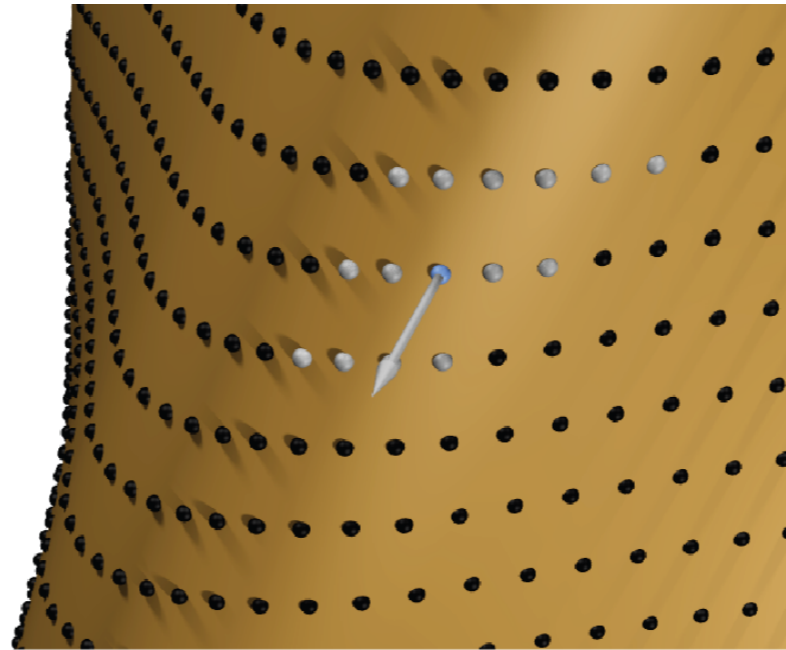
1. Compute time coordinate spacing ( $\sigma$ ), and form space-time surface.
2. Compute space time neighborhood using ANN, and locally estimate space-time surface normals.
3. Solve linear system to estimate  $(c_j, \bar{c}_j)$ .
4. Convert velocity vectors to rotation matrix + translation vector using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

# Normal Estimation: PCA Based



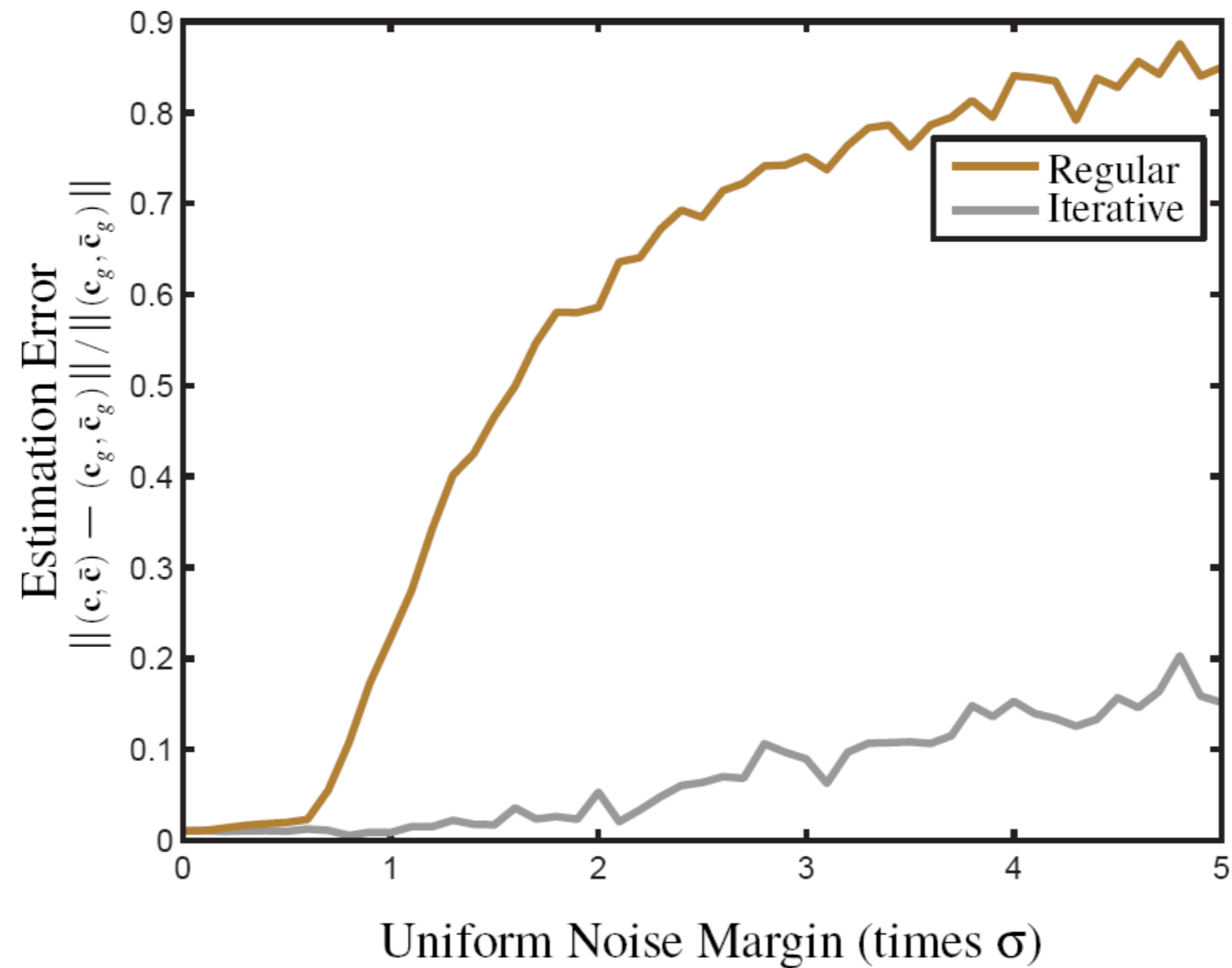
Plane fitting using PCA using chosen neighborhood points.

# Normal Estimation: Iterative Refinement



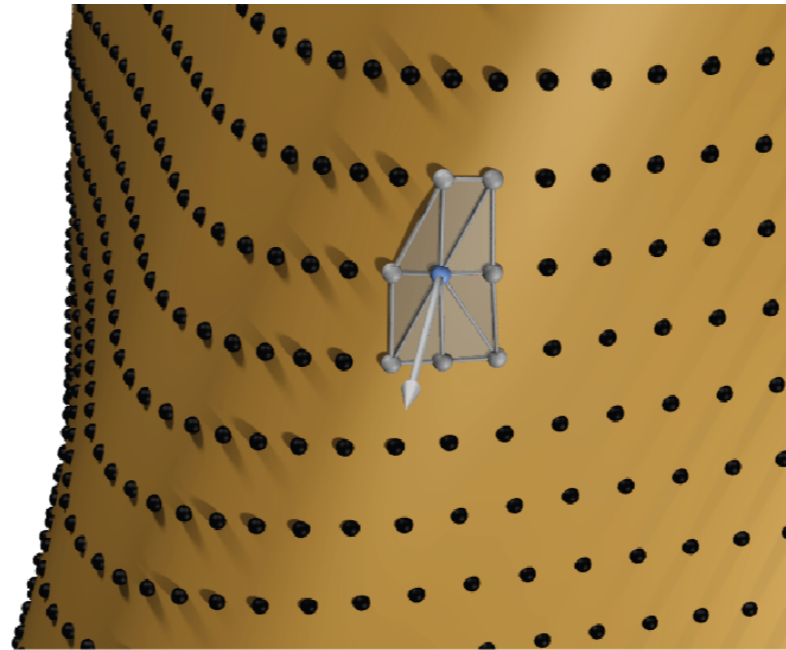
Update neighborhood with current velocity estimate.

# Normal Estimation: Effect of Noise



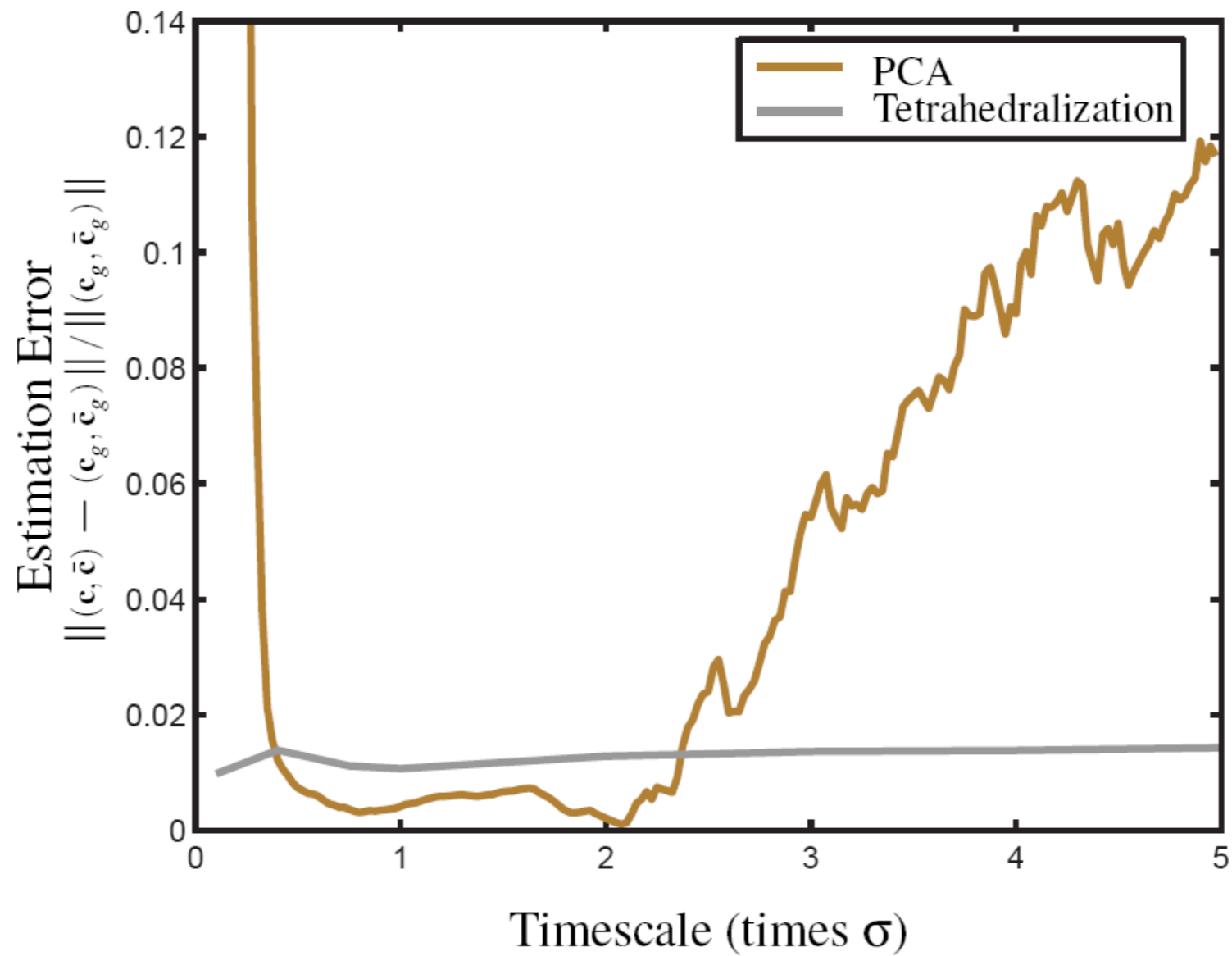
Stable, but more expensive.

# Normal Estimation: Local Triangulation



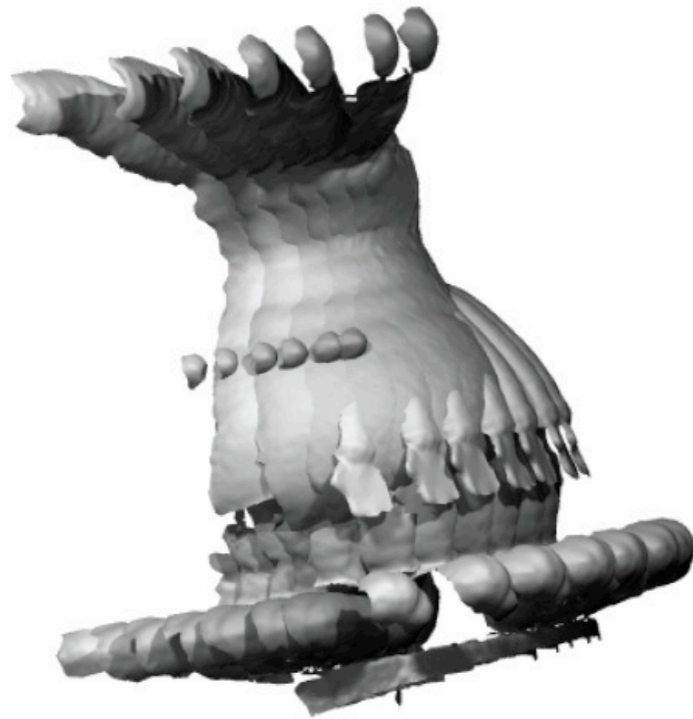
Perform local surface triangulation (tetrahedralization).

# Normal Estimation



Stable, but more expensive.

# Comparison with ICP



ICP point-plane



Dynamic registration

# Rigid/Deformable: Coati Sequence (2.2K frames)

**Coati**

Input frames (Selection)

2200 pointclouds scanned at 17 Hz

transformations only for adjacent frames considered

no global error correction

no noise smoothing



# Rigid: Bee Sequence (2,2K frames)

**Bee**

Input frames (Selection)

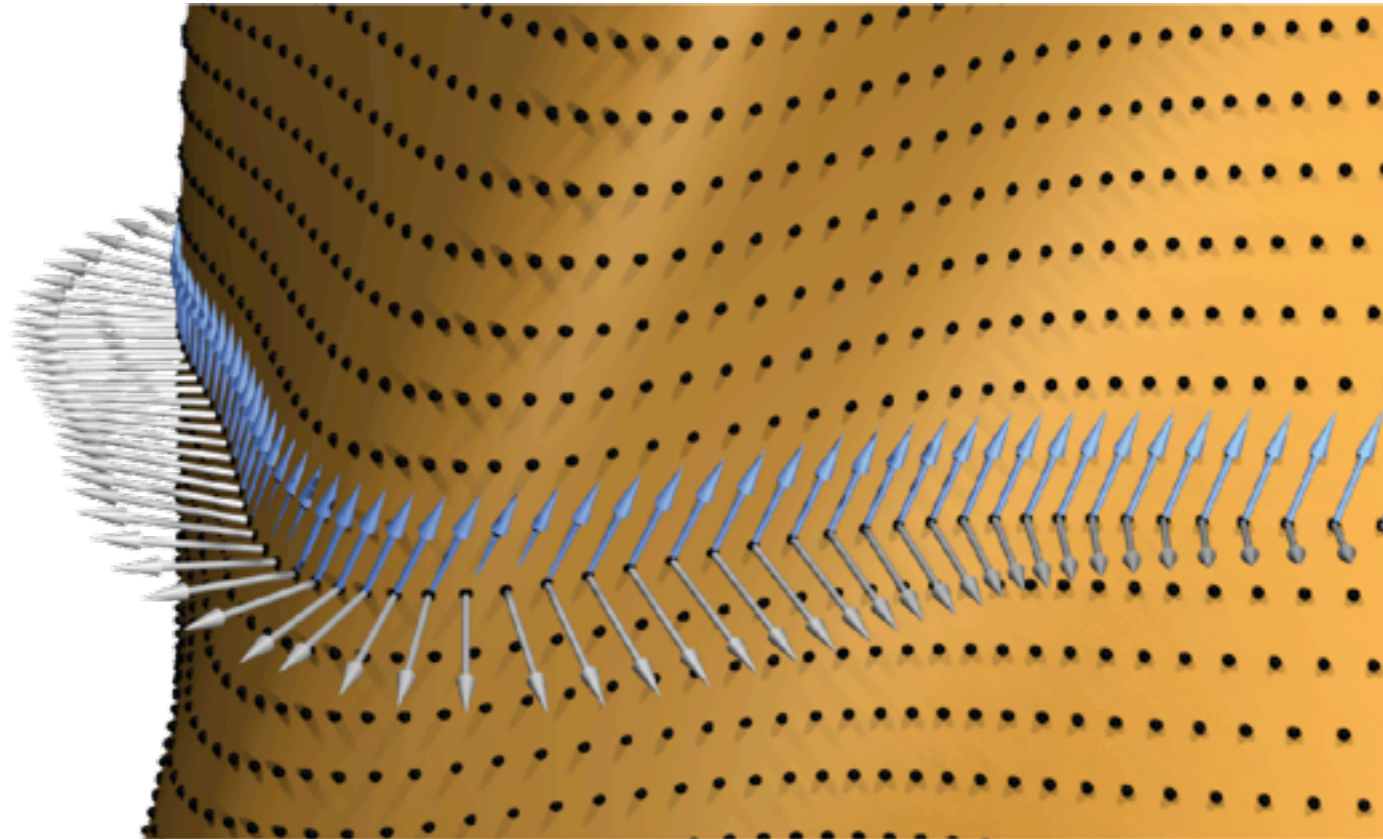
2200 pointclouds scanned at 17 Hz

transformations only for adjacent frames considered

no global error correction

no noise smoothing

# Handling Large Number of Frames



# Rigid/Deformable: Teapot Sequence (2.2K frames)

## Teapot Input frames (Selection)

2200 pointclouds scanned at 17 Hz

transformations only for adjacent frames considered

no global error correction

no noise smoothing

# Deformable Bodies

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

- Cluster points, and solve smaller systems.
- Propagate solutions with regularization.

# Deformable: Hand (100 frames)

## Hand

### Input frames & registered result

100 pointclouds scanned at 17 Hz

transformations only for adjacent frames considered

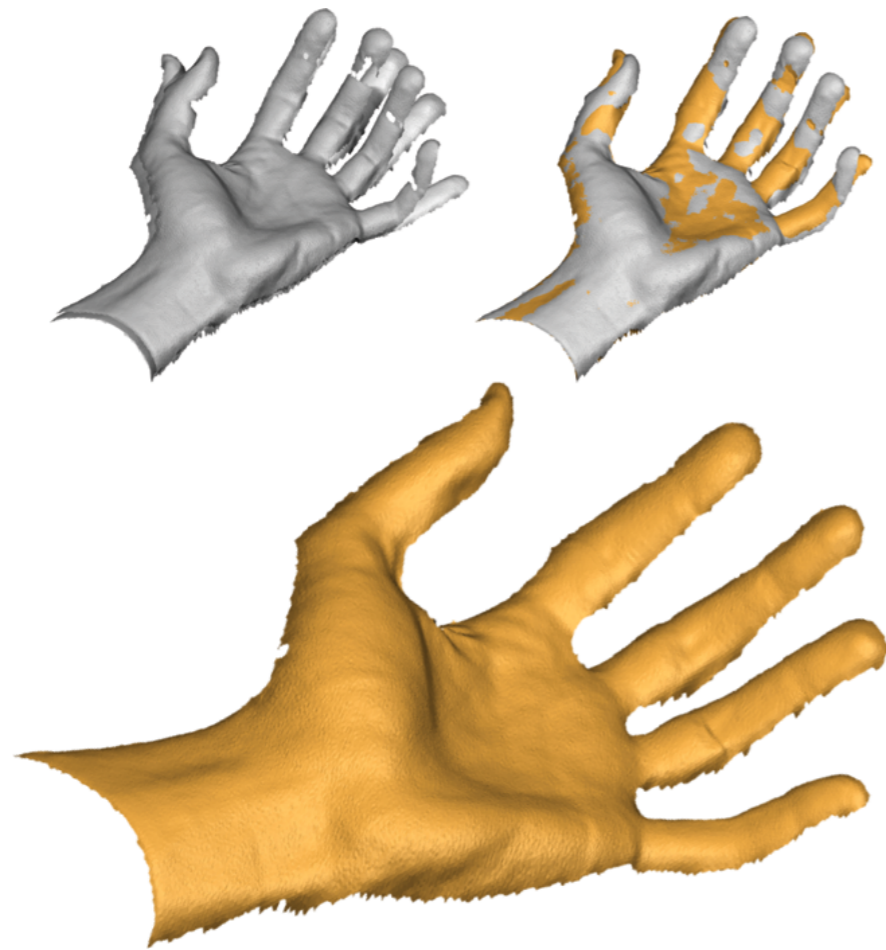
no global error correction

no noise smoothing

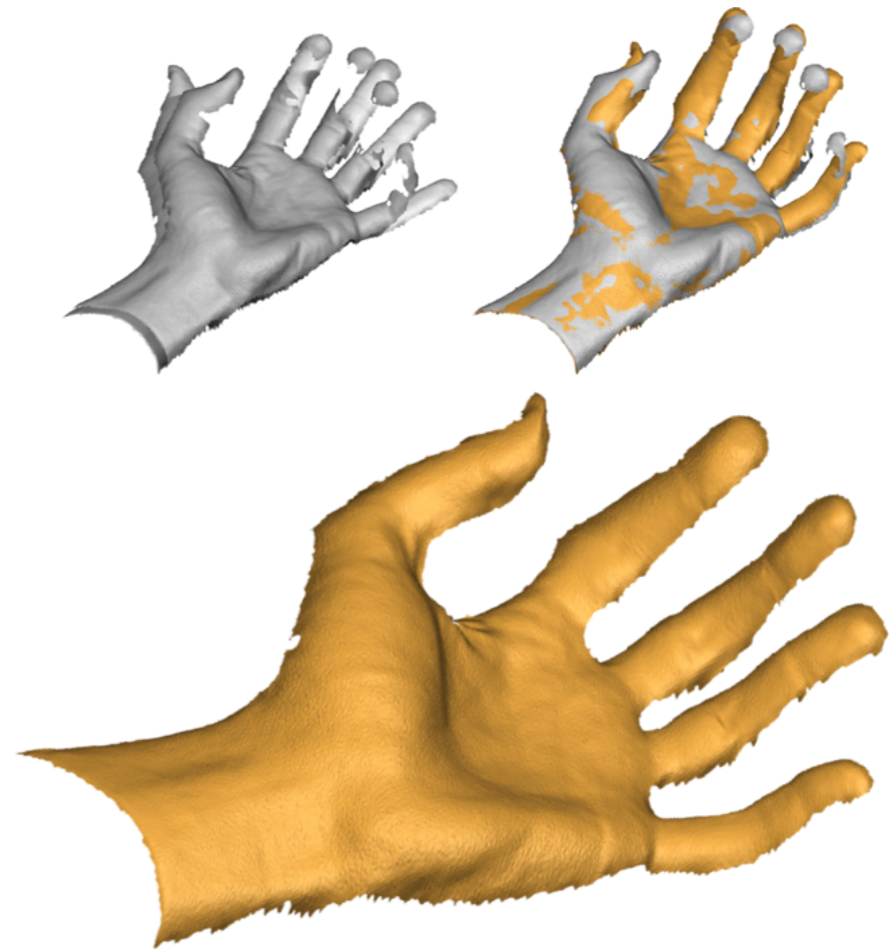
first frame is tracked

deformation due to severely missing data (e.g. ring finger)

# Deformable: Hand (100 frames)



scan #1 ! scan #50



scan #1 ! scan #100

# 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

- Need more scans, dense scans, ...
- Sampling condition ! time and space

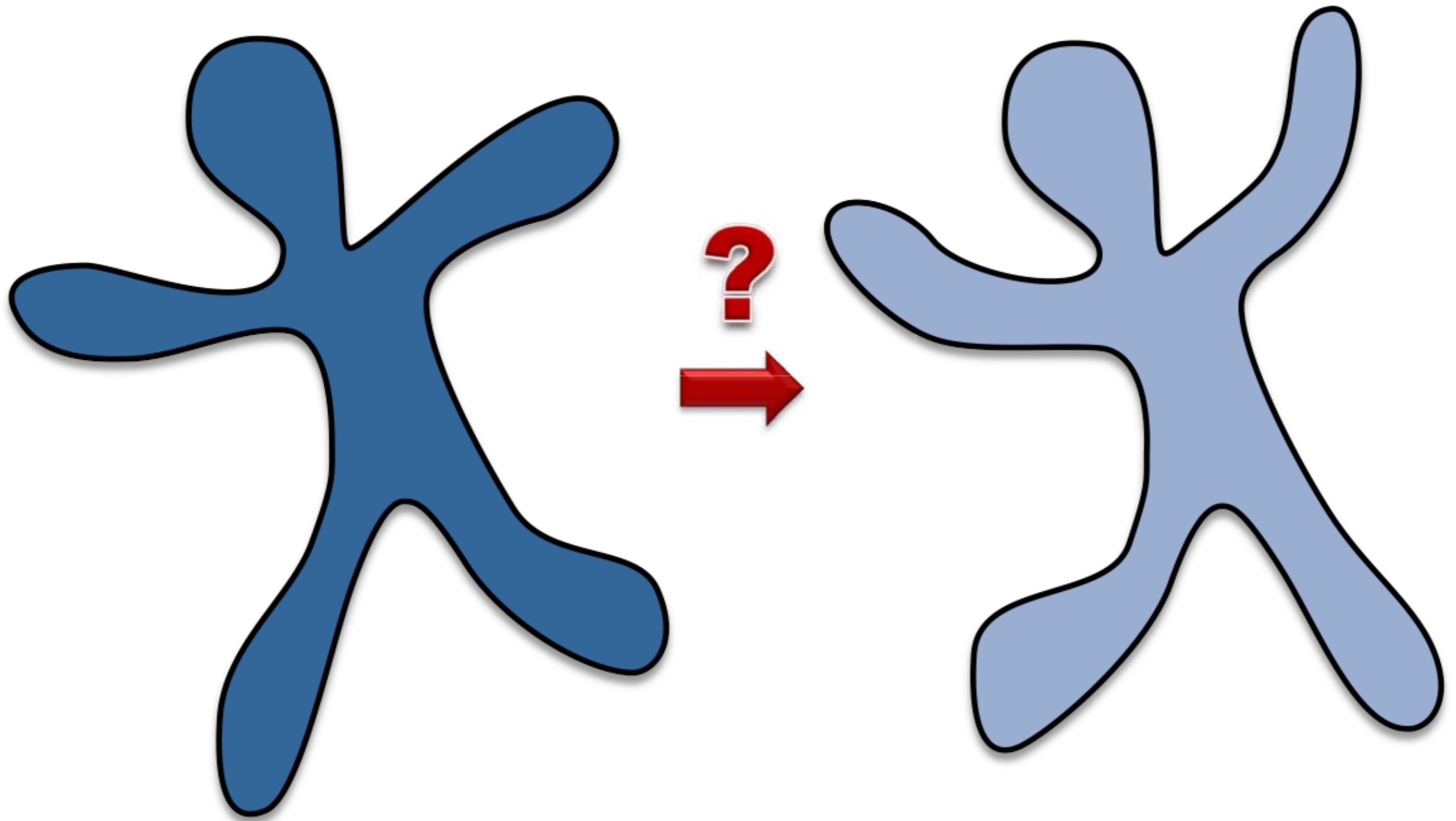


Prof. Michael Wand  
Utrecht University



# Animation Reconstruction “Urshapes”!

# Correspondences



# Problem

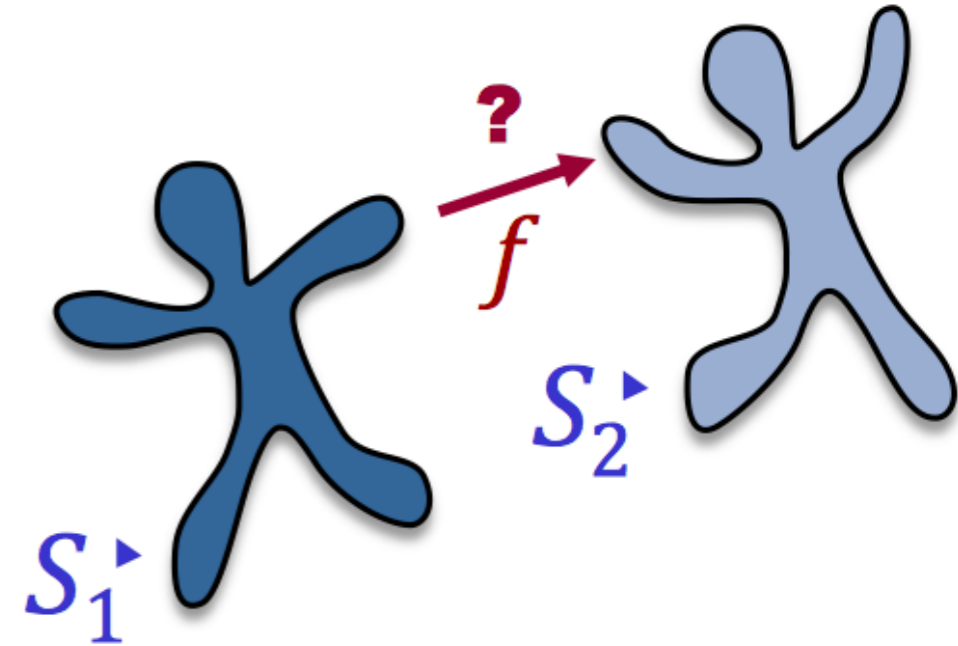
## Problem Statement:

## Given:

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

## We are looking for:

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

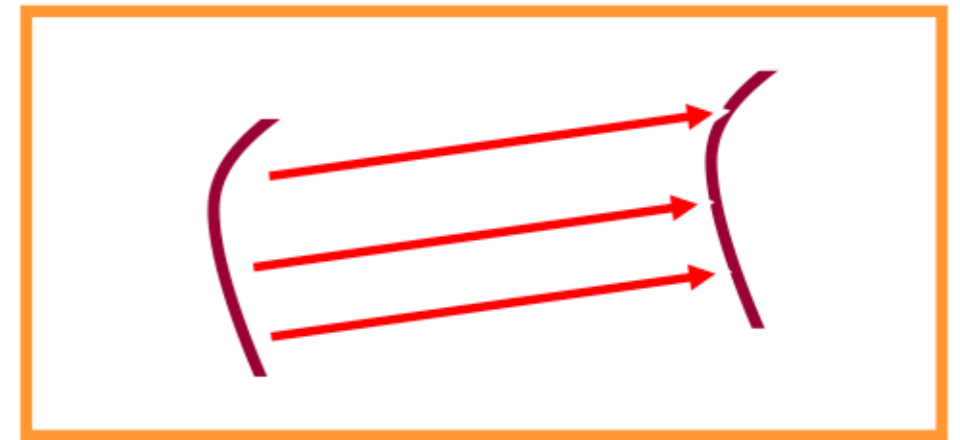
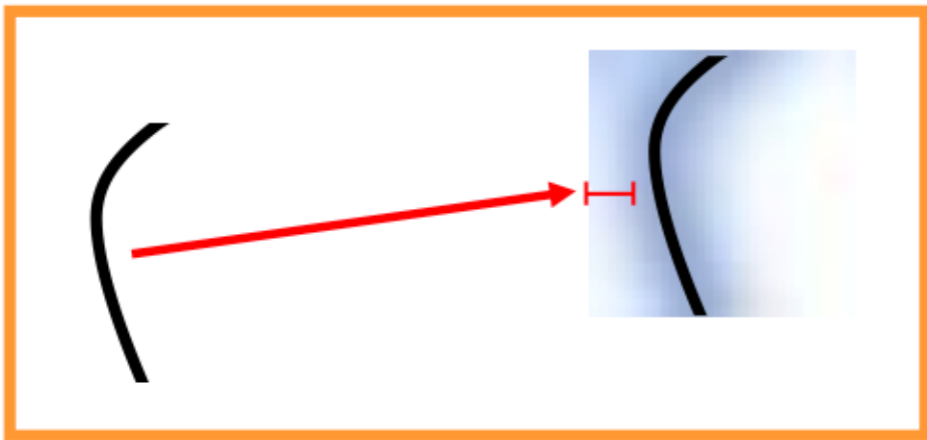


# Variational Problem

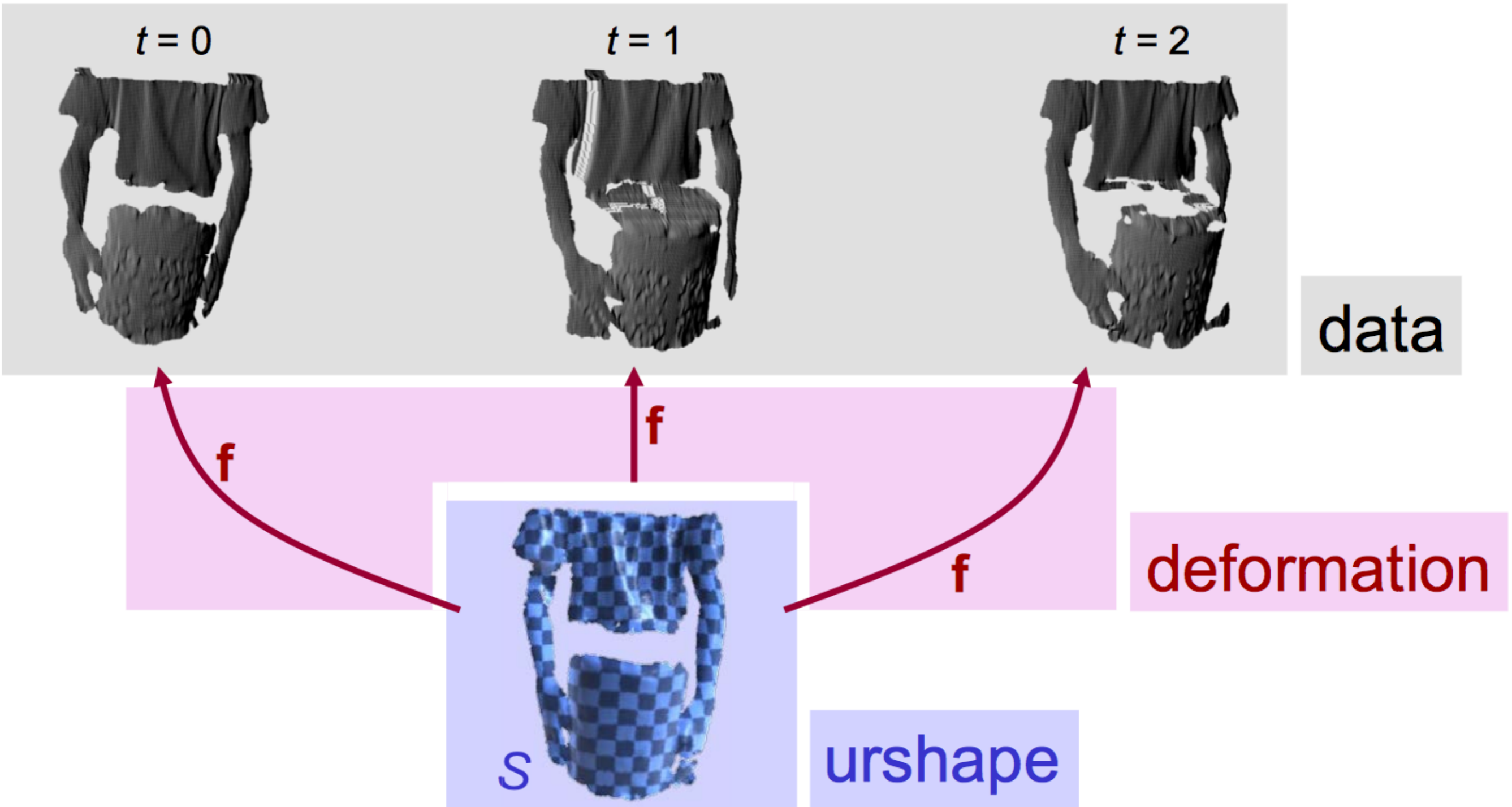
## Variational Problem:

- Formulate as an energy minimization problem:

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



# Scan Sequence



# Hierarchical Merging

data

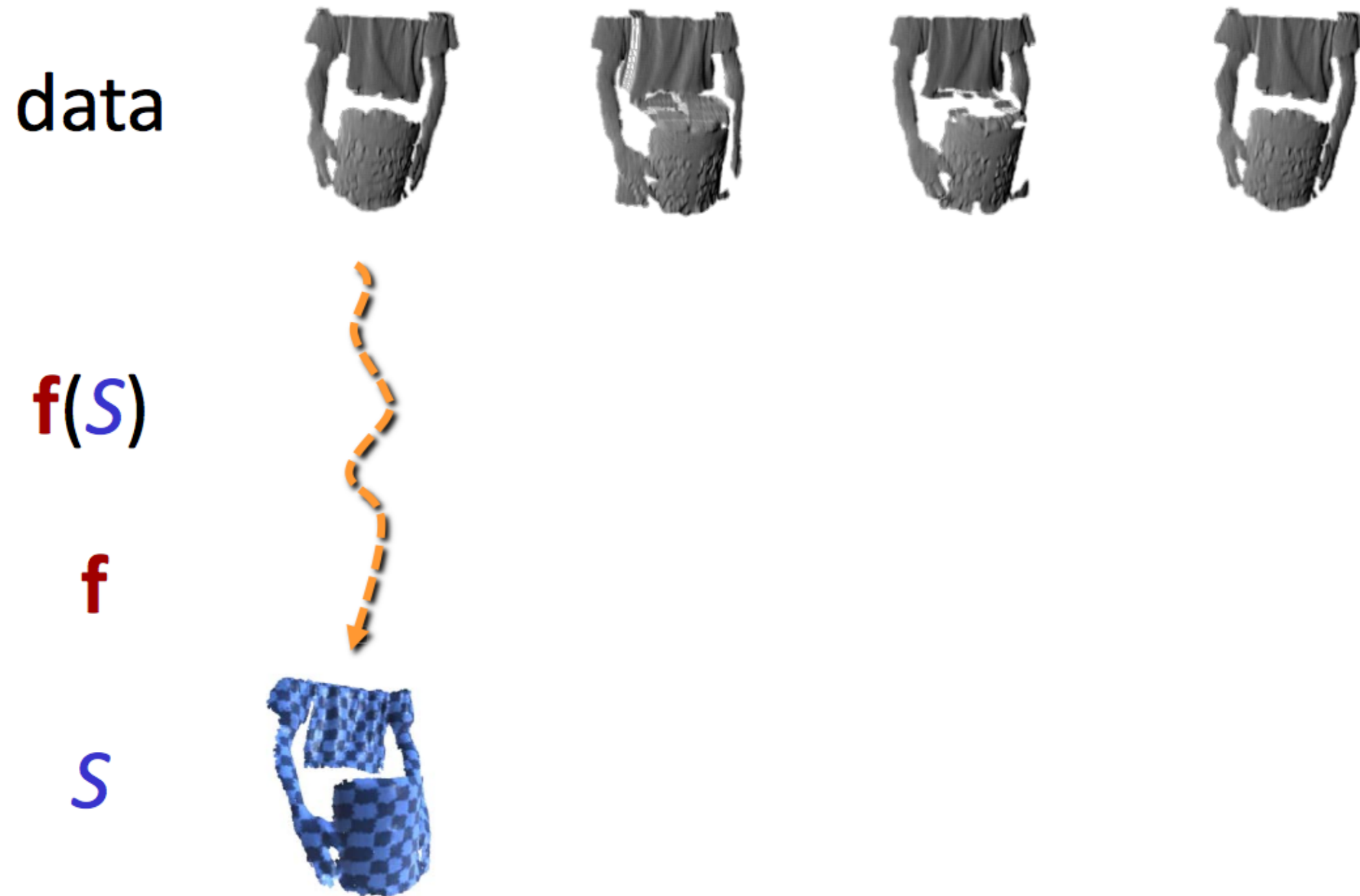


$f(S)$

$f$

$S$

# Hierarchical Merging



# Initial Urshape

data



$f(S)$



$f$



$S$





# Initial Urshape

data



$f(S)$



$f$



$S$

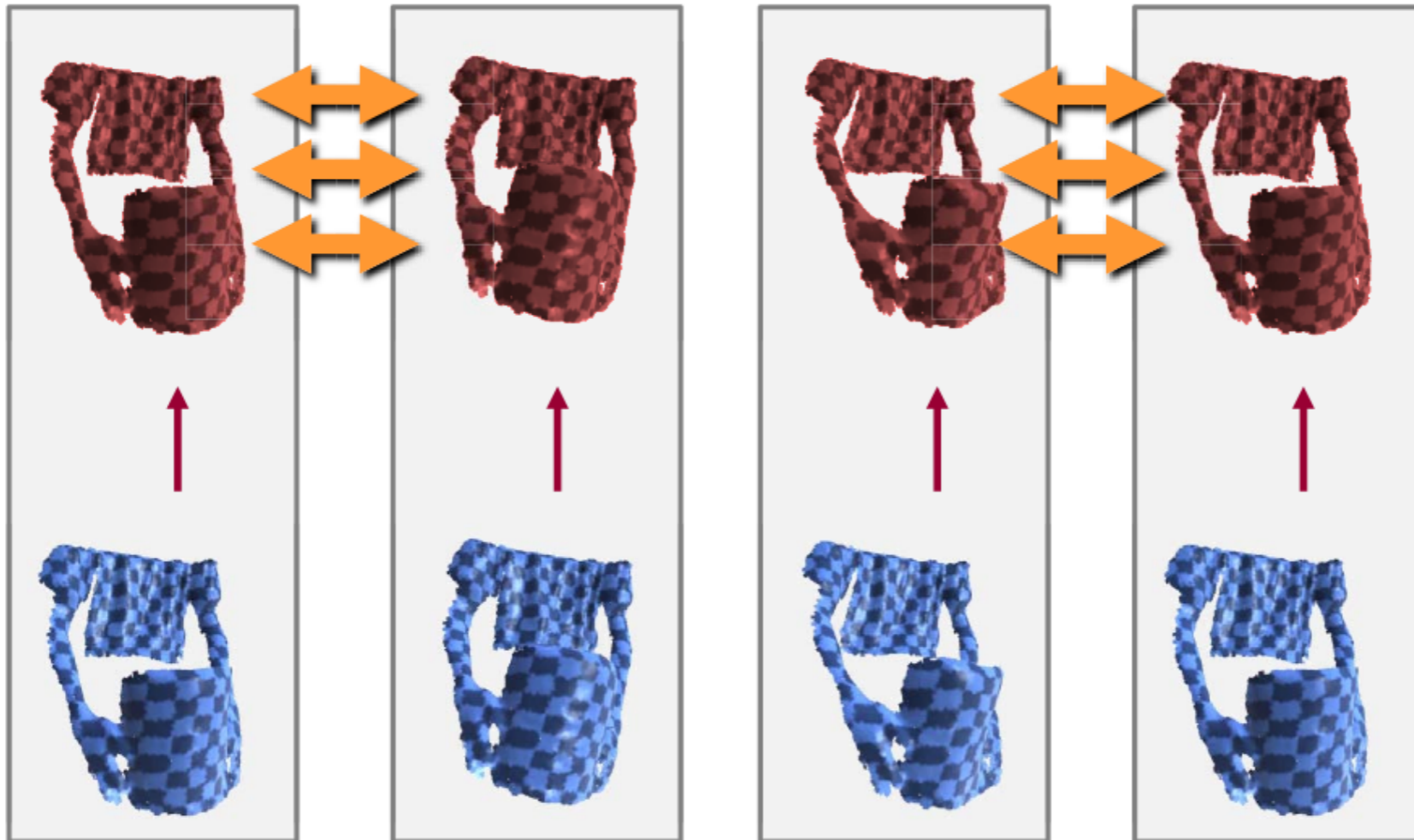


# Alignment

data



$f(S)$



$f$

$S$

# Alignment

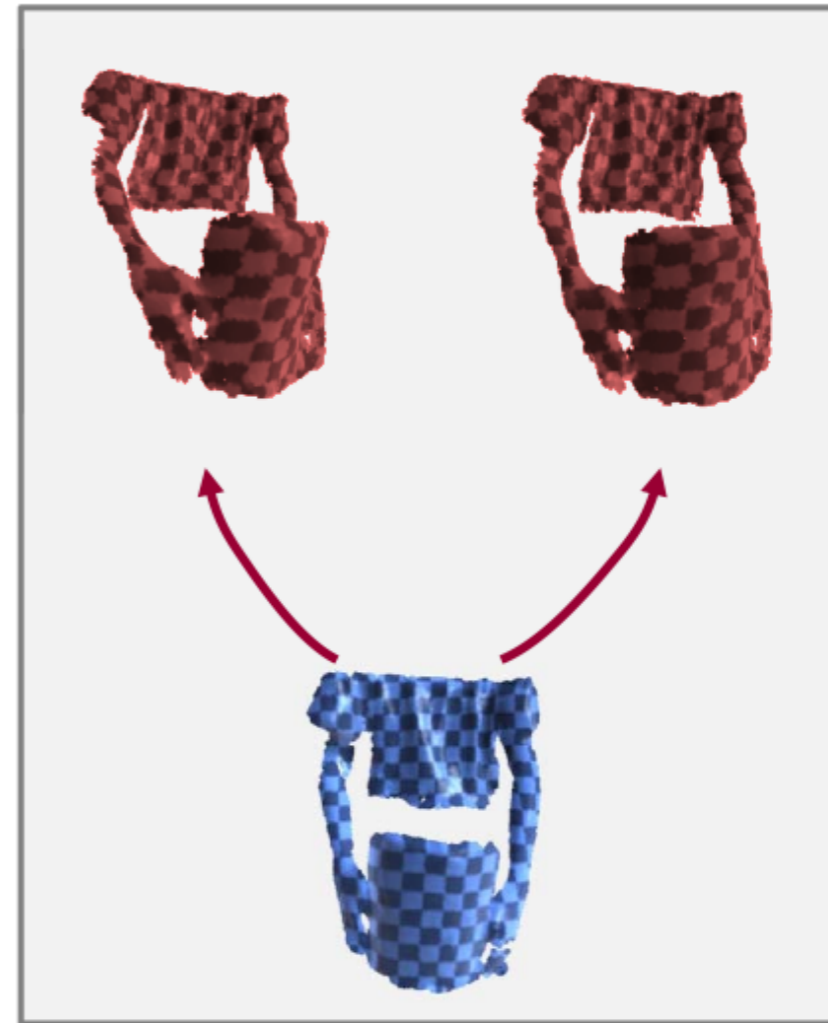
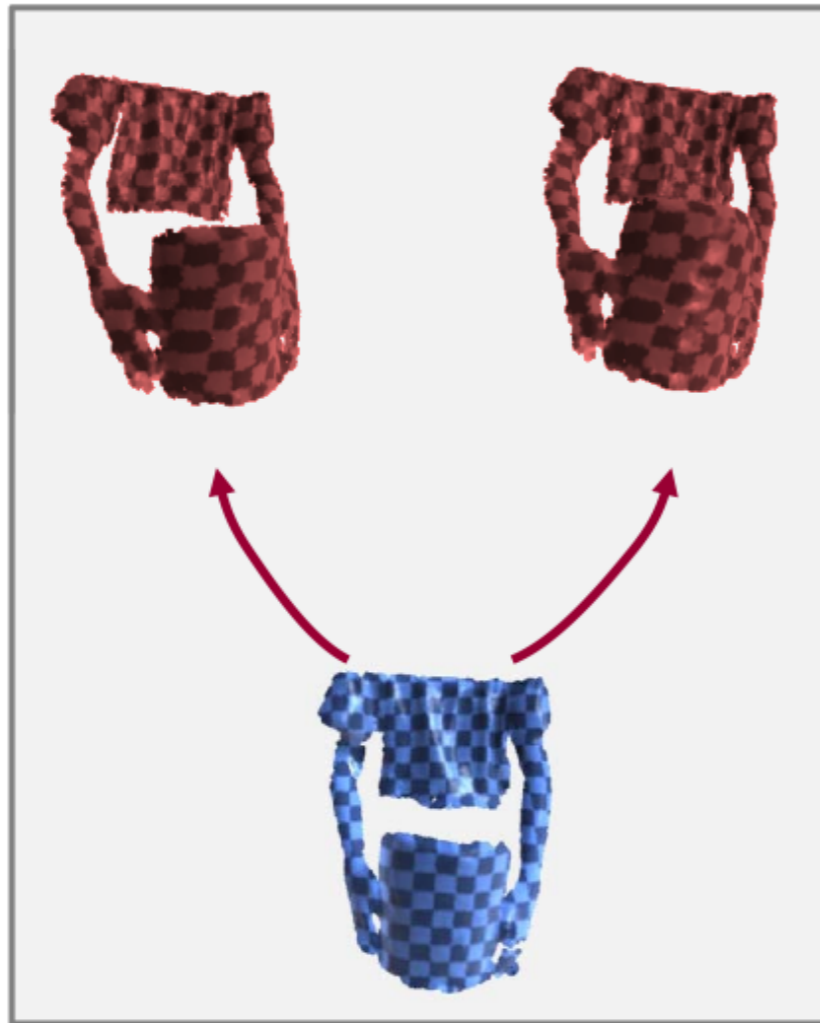
data



$f(S)$

$f$

$S$

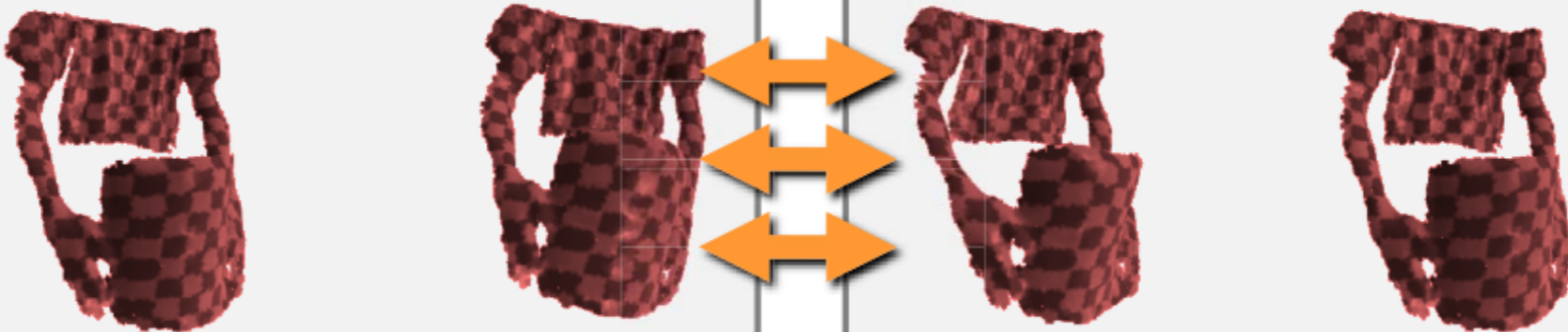


# Hierarchical Alignment

data



$f(S)$



$f$



$S$

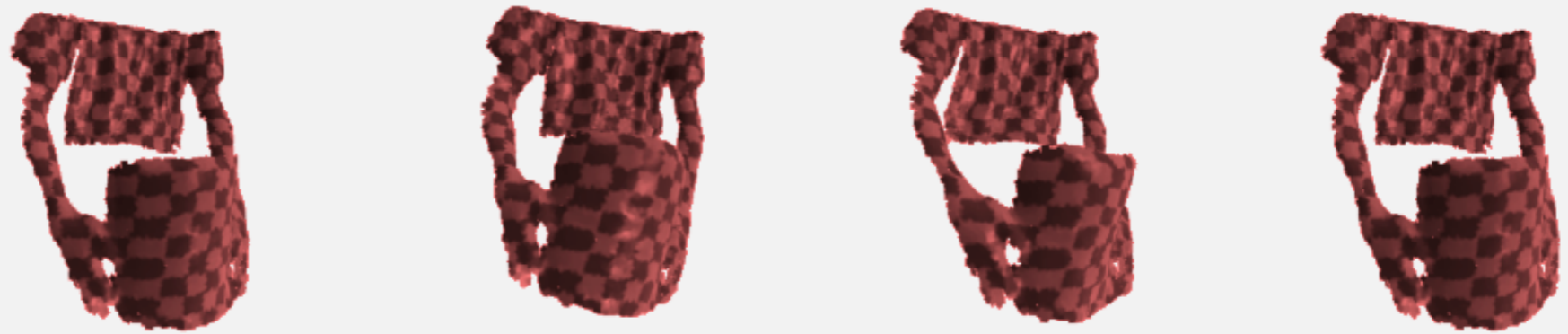


# Hierarchical Alignment

data

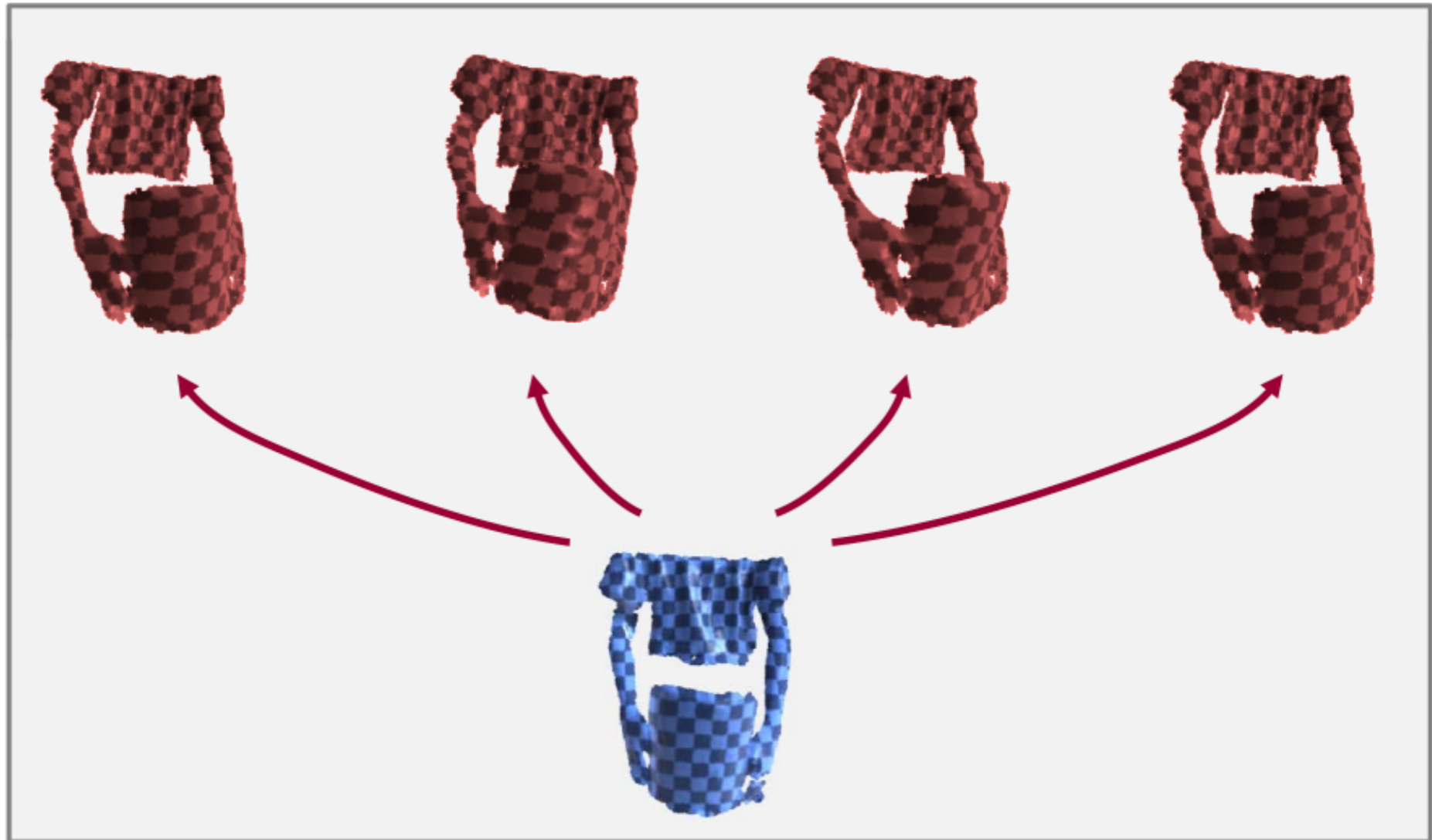


$f(S)$



$f$

$S$



## Energy Function

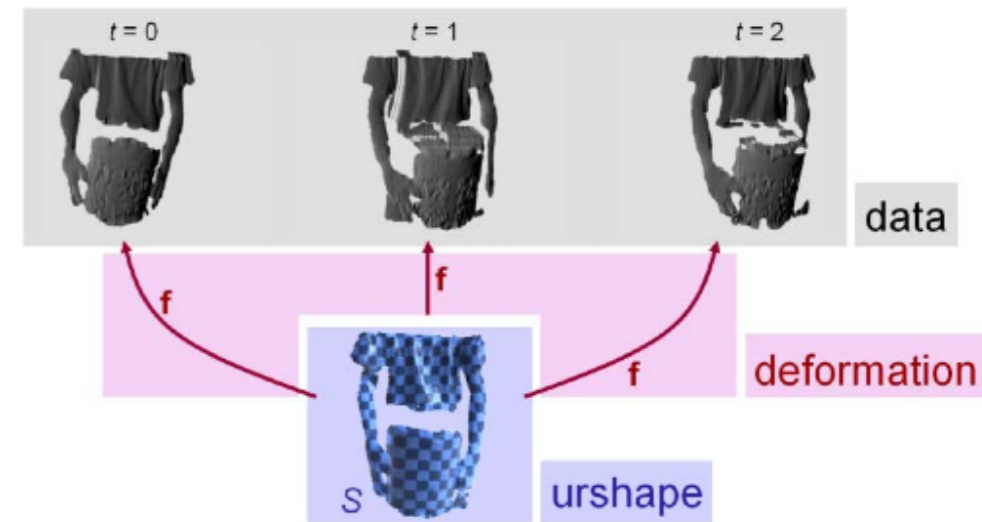
$$E(\mathbf{f}, S) = E_{data} + E_{deform} + E_{smooth}$$

## Components

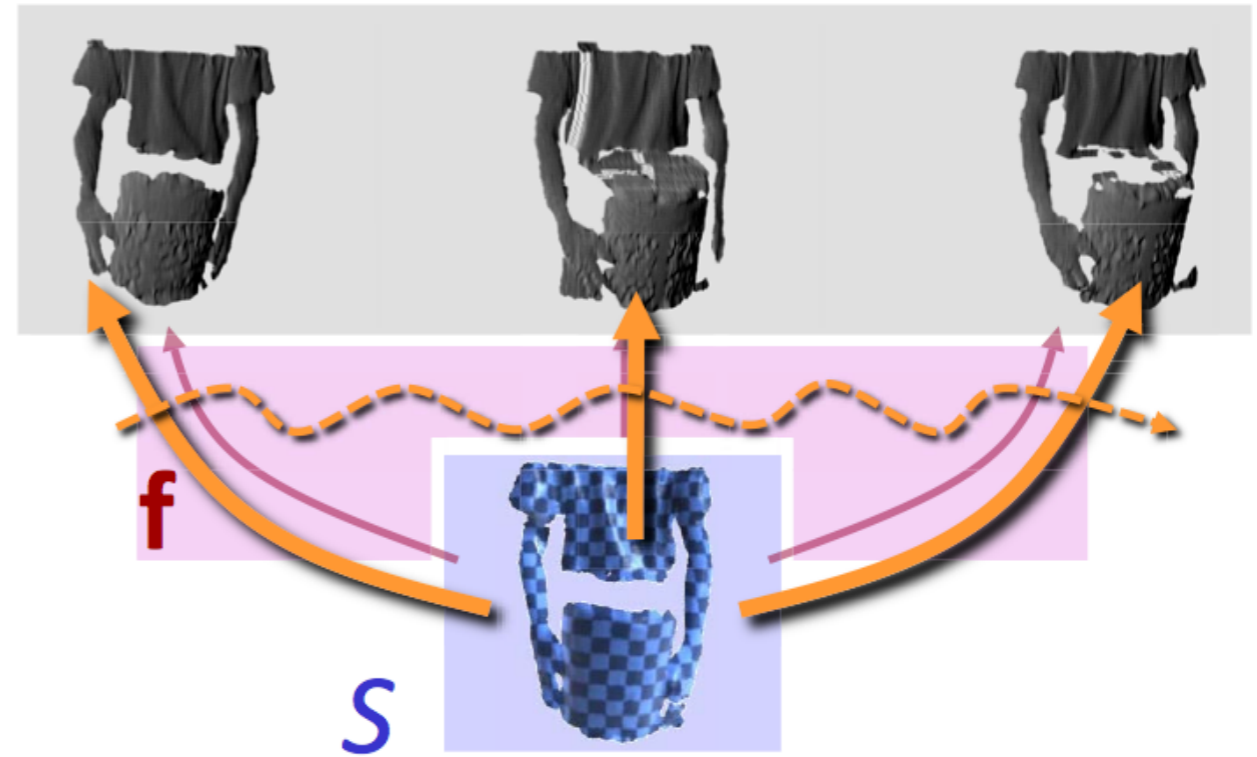
- $E_{data}(\mathbf{f}, S)$  – data fitting
- $E_{deform}(\mathbf{f})$  – elastic deformation, smooth trajectory
- $E_{smooth}(S)$  – smooth surface

## Final Optimization

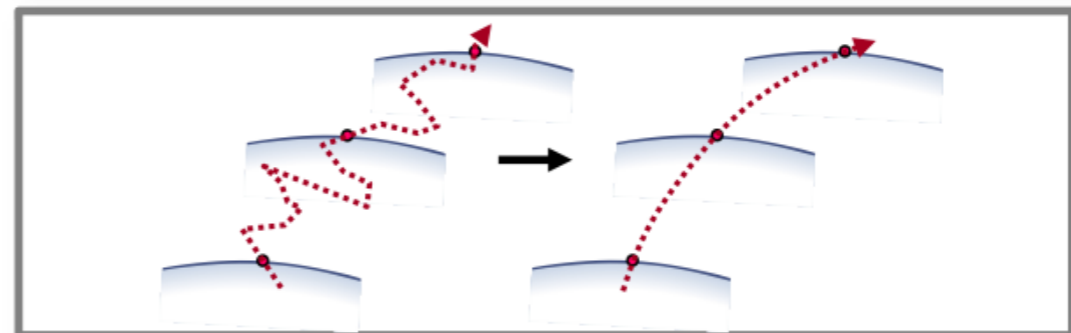
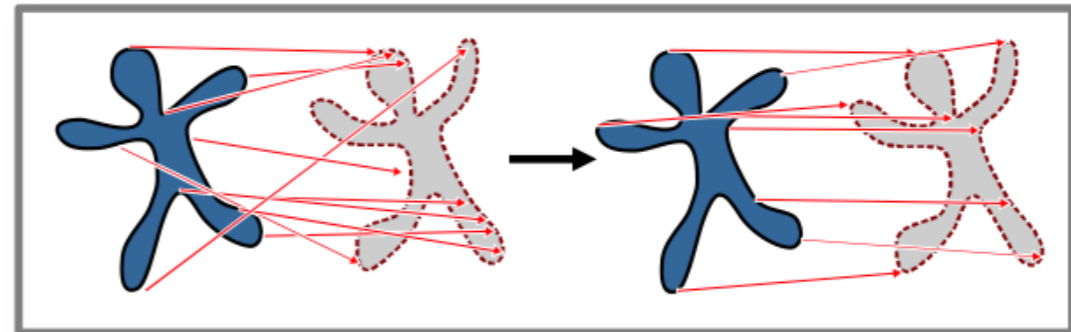
- Minimize over all frames



## Deformation Field



- Elastic energy
- Smooth trajectories

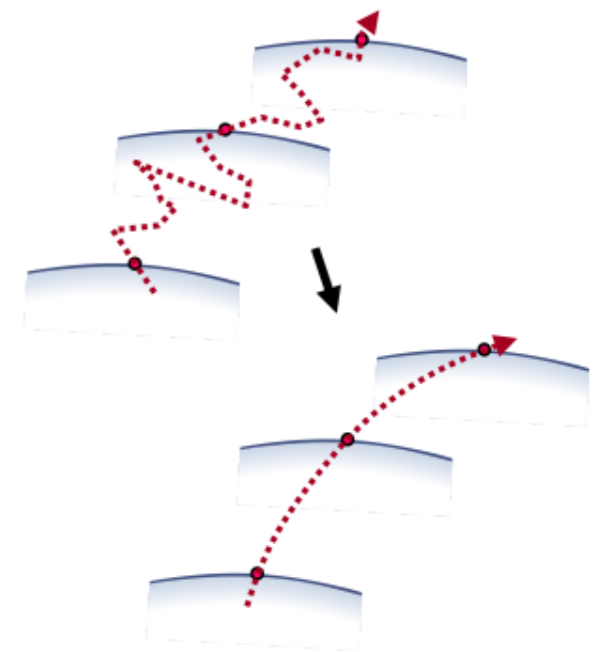


## More Regularization

- Acceleration:
  - Smooth trajectories
- Velocity (weak):
  - Damping

$$E_{acc} = \int_T \int_V |\partial_t^2 \mathbf{f}|^2$$

$$E_{vel} = \int_T \int_V |\partial_t \mathbf{f}|^2$$





# Results



input data

# Results



input data

# Extension: Animation Cartography

[Tevs et al. 2012]

<http://cs621.hao-li.com>

# Thanks!

