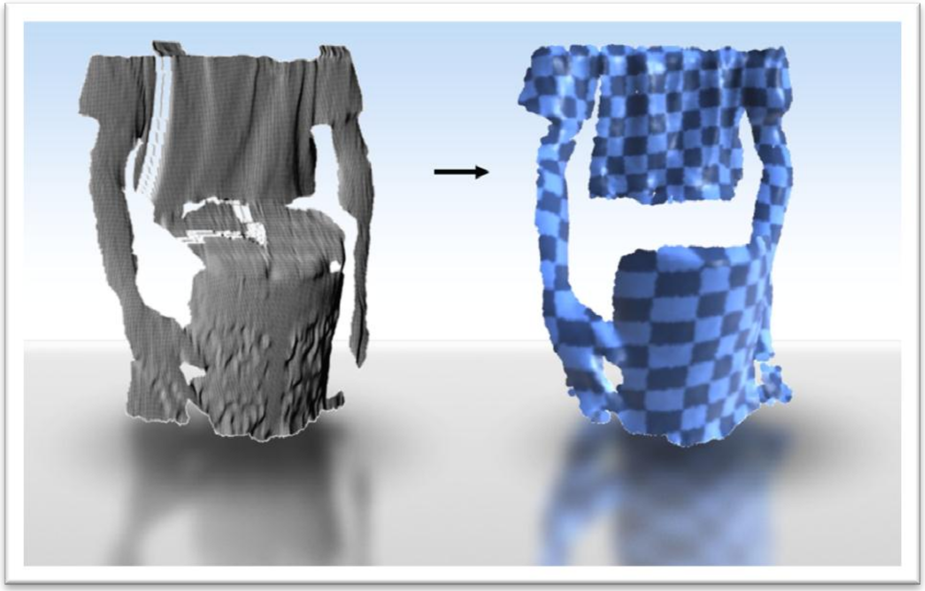
Statistical Geometry Processing

Winter Semester 2011/2012



(Deformable) Shape Matching

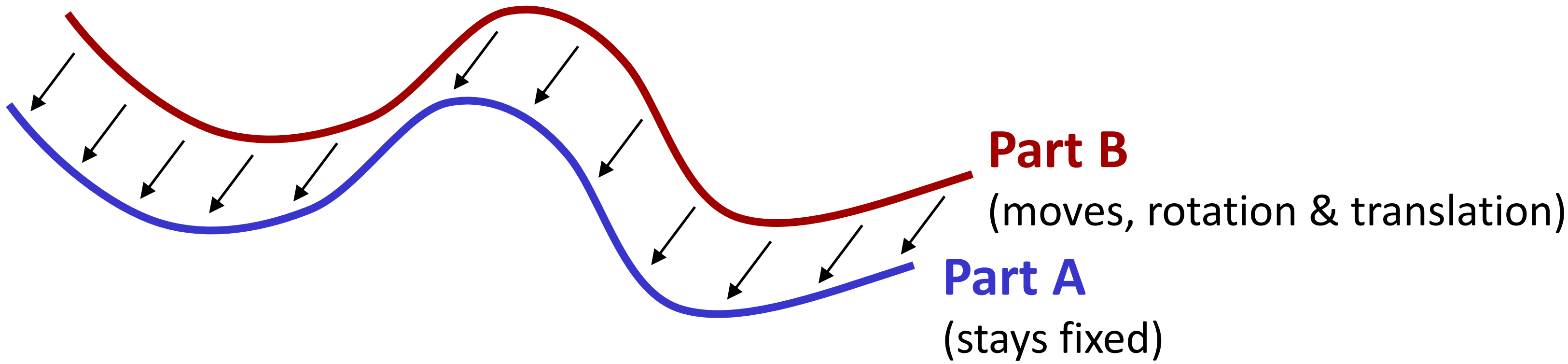
UNIVERSITÄT DES SAARLANDES

CLUSTER OF EXCELLENCE

max planck institut informatik

Rigid Shape Matching

Iterated Closest Points (ICP)



The main idea:

- Pairwise matching technique
  - Registers two scans
  - Multi-part matching is a different story (more on this later)
- We want to minimize the distance between the two parts
  - We set up a variational problem
  - Minimize distance "energy" by rigid motion of one part

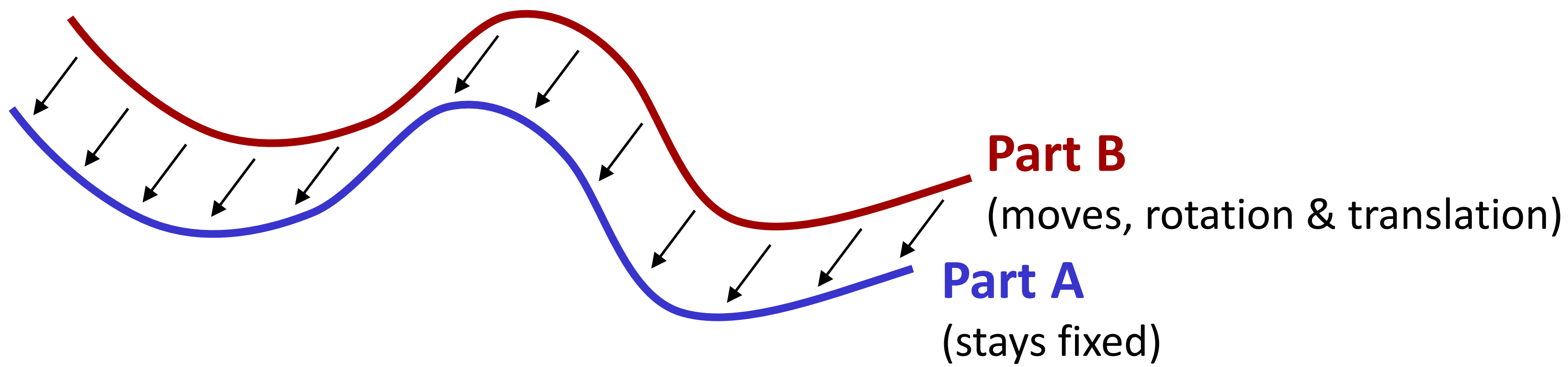
Iterated Closest Points (ICP)

Problem:

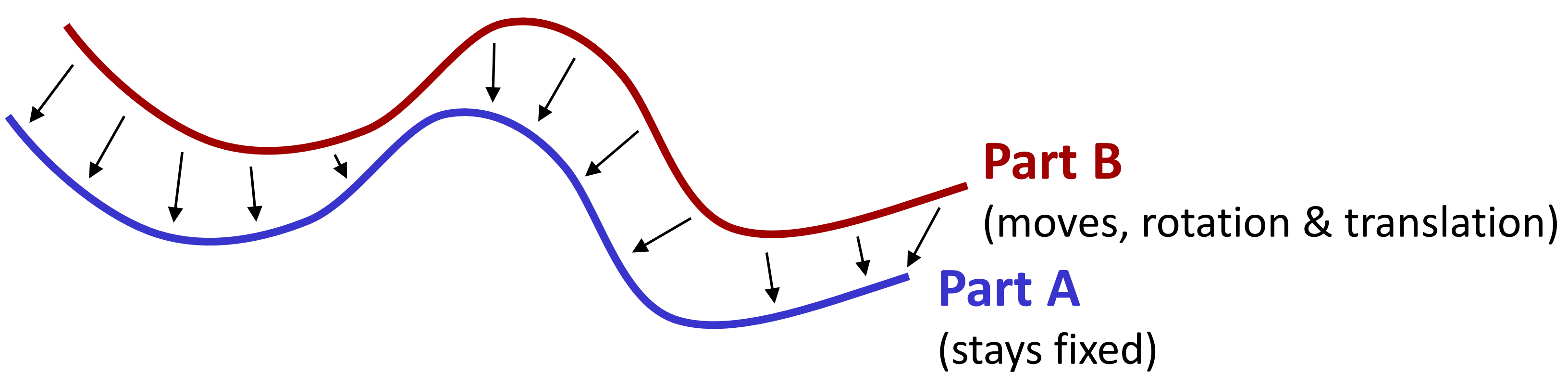
- How to compute the distance
- This is simple if we know the *corresponding points*.
  - Of course, we have in general no idea of what corresponds...
- ICP-idea: set closest point as corresponding point
- Full algorithm:
  - Compute closest point points
  - Minimize distance to these closest points by a rigid motion
  - Recompute new closest points and iterate

Closest Points

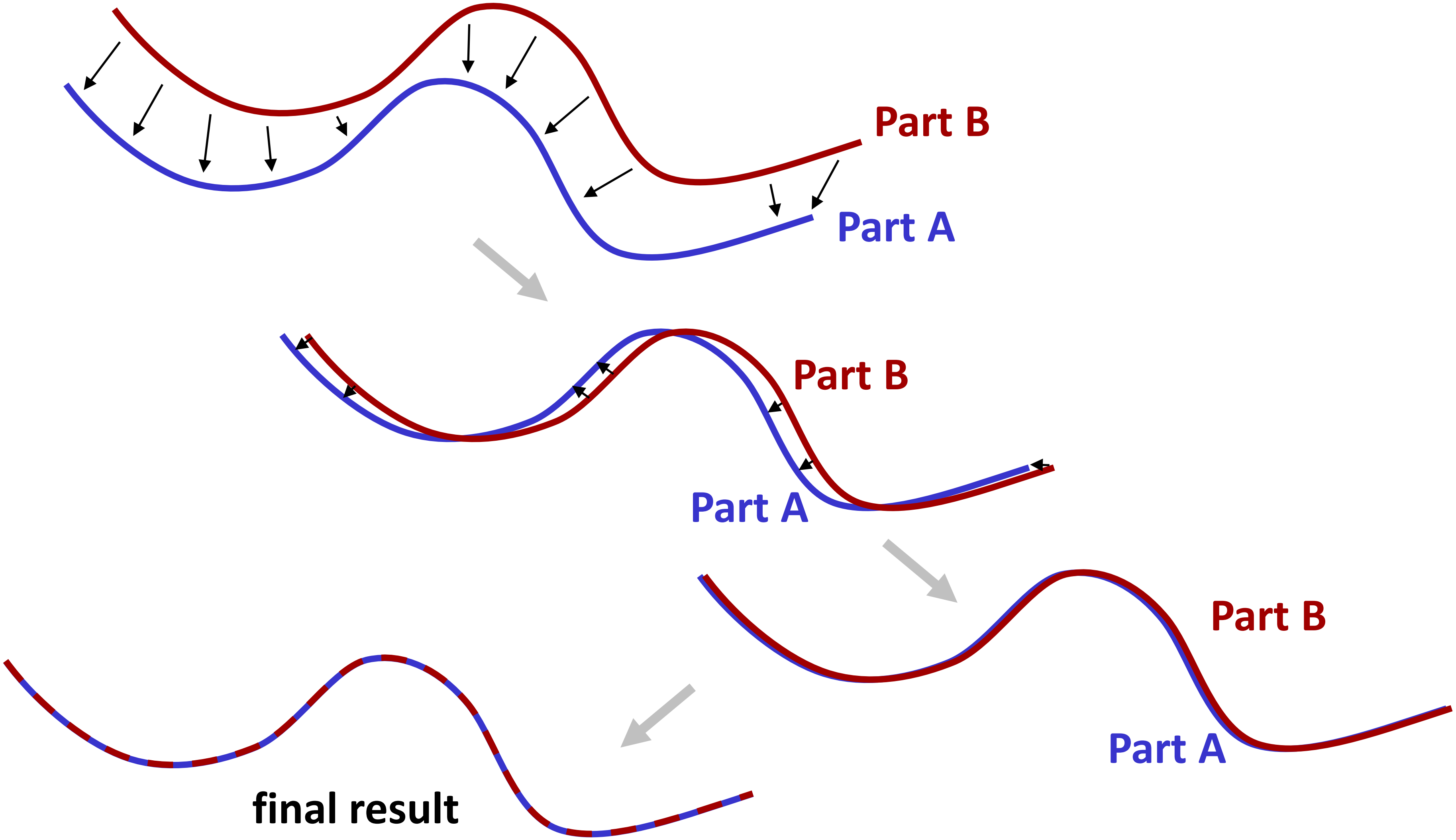
Distances:



Closest points distances:



Iteration



Variational Formulation

Variational Formulation:

$$\underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \int_B dist(\mathbf{R}\mathbf{x}+\mathbf{t},A)^2\,d\mathbf{x} \approx \underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \sum_{i=1}^{n}\left(\mathbf{R}\mathbf{p}_i^{(A)}+\mathbf{t}-\mathbf{p}_{nearest(i)}^{(B)}\right)^2$$

Variables: Orthogonal matrix **R**, translation vector **t**

Numerical Solution

Question: How to minimize this energy?

- The energy is quadratic
- There is only one problem...
  - Constraint optimization
  - We have to use an orthogonal matrix...
- This problem can (still) be solved exactly.

$$\underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \int_B di$$

Solution

First step: computing the translation

- Easy to see: average translation is optimal (c.f. total least squares)
- $\mathbf{t} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i^{(A)} - \mathbf{p}_{nearest(i)}^{(B)}$
- This is independent of the rotation

Second step: compute the rotation

- (2a) Compute optimal linear map
- (2b) Orthogonalize

Optimal Linear Map

First:

- Subtract translation from points $\tilde{\mathbf{p}}_i^{(A)} = \mathbf{p}_i^{(A)} - \mathbf{t}$
- Then: Solve an unconstrained least-squares problem

$$\forall i = 1..n : \mathbf{M}\tilde{\mathbf{p}}_i^{(A)} = \mathbf{p}_{nearest(i)}^{(B)}$$

unknowns
(9 variables)

$$\forall i = 1..n : \begin{pmatrix} m_{1,1} & m_{2,1} & m_{3,1} \\ m_{1,2} & m_{2,2} & m_{3,2} \\ m_{1,3} & m_{2,3} & m_{3,3} \end{pmatrix} \tilde{\mathbf{p}}_i^{(A)} = \mathbf{p}_{nearest(i)}^{(B)}$$

- Finally: compute the orthogonal matrix **R** that is closest to **M**.

Least-Squares Optimal Rotation

How to compute a least-squares (Frobenius norm) orthogonal matrix that fits a general matrix:

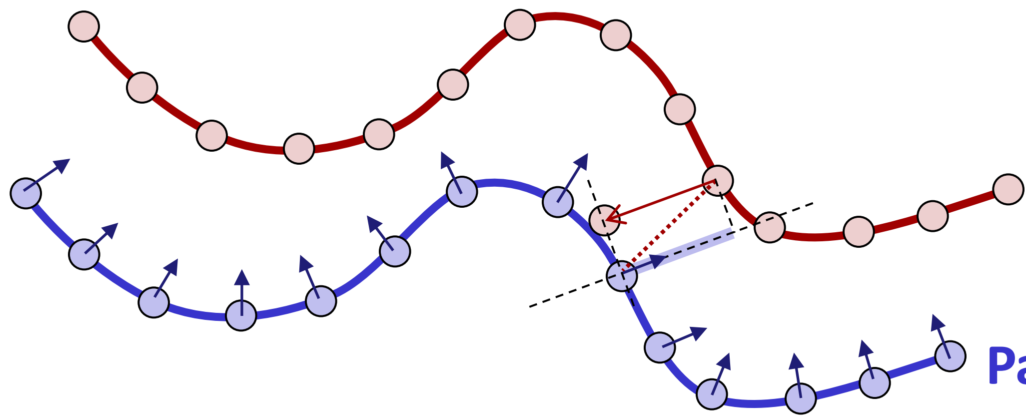
- Compute the SVD: $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^T$
- The least-squares orthogonal fit is: $\mathbf{R} = \mathbf{U}\mathbf{V}^T$
  (just set all singular values to one)
- We can compute this in one step:
  - Solve the least-squares matrix fitting problem using SVD
  - Omit the diagonal matrix straight ahead

Generalizations

Convergence speed:

- Convergence of basic “*point-to-point*” ICP is not so great
  - Typically: 20-50 iterations for simple examples
  - Problem: Zero-th order method
    (flip point correspondences in each step)
- Improvement: “*point-to-plane*” ICP
  - First order approximation
  - Match points to tangential planes rather than points
  - Converges much faster (3-5 iterations for similar examples)

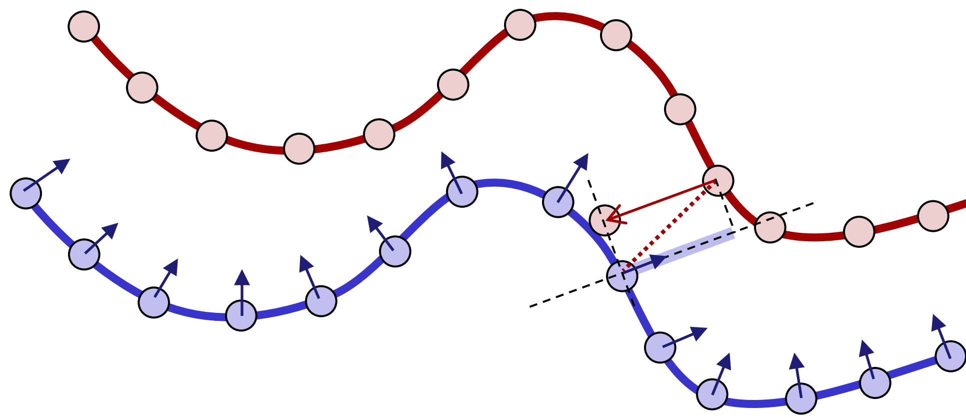
Implementation



$$\underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \int_B \left\langle \mathbf{Rx} + \mathbf{t} - nearest(A), \mathbf{n}(nearest(A)) \right\rangle^2 dx$$

$$\approx \underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \sum_{i=1}^{n} \left\langle \mathbf{Rp}_i^{(A)} + \mathbf{t} - \mathbf{p}_{nearest(i)}^{(B)}, \mathbf{n}_{nearest(i)}^{(B)} \right\rangle^2$$

Implementation

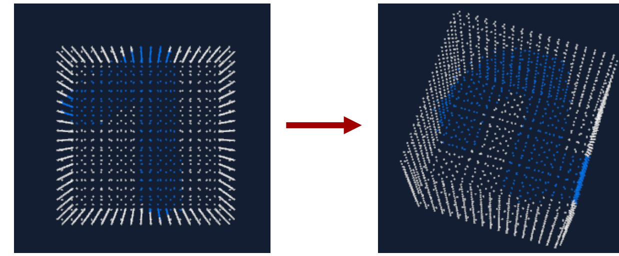


Implementation:

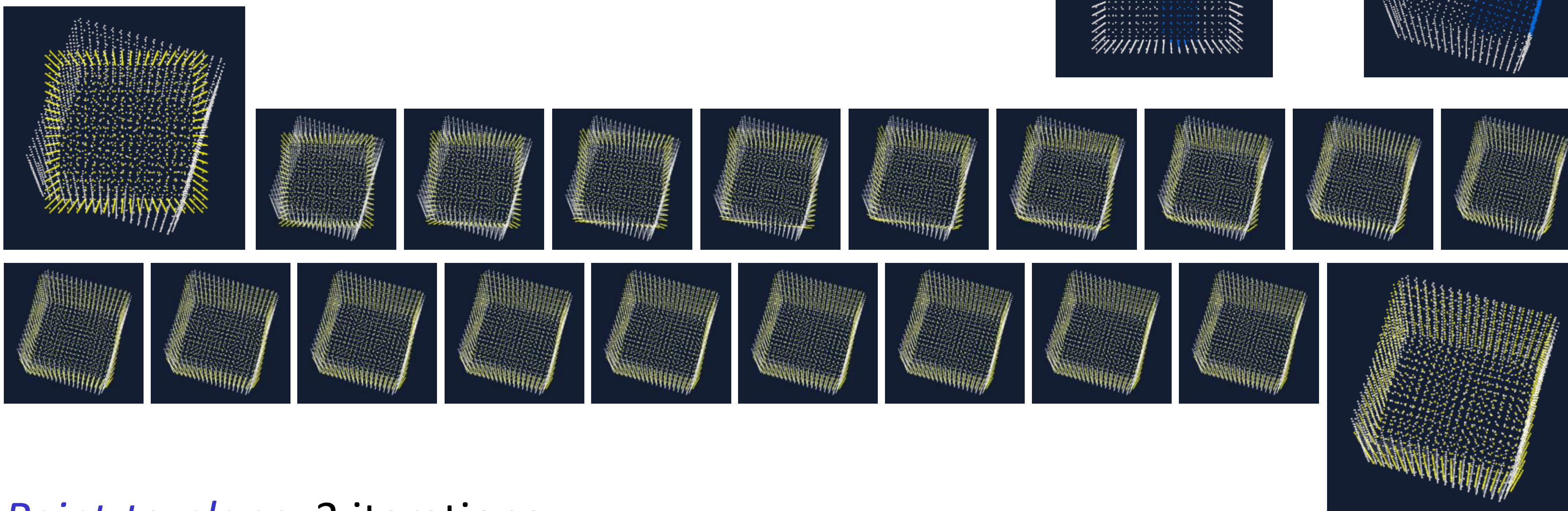
- We need normals for each point (unoriented) $\rightarrow$ kNN+PCA
- Compute closest point, project distance vector to its normal
- Minimize the sum of all such distances:

$$\underset{\substack{\mathbf{R}\in SO(3),\\ \mathbf{t}\in\mathbb{R}^3}}{\arg\min} \sum_{i=1}^{n} \left\langle \mathbf{R}\mathbf{p}_i^{(A)} + \mathbf{t} - \mathbf{p}_{nearest(i)}^{(B)}, \mathbf{n}_{nearest(i)}^{(B)} \right\rangle^2$$

Comparison

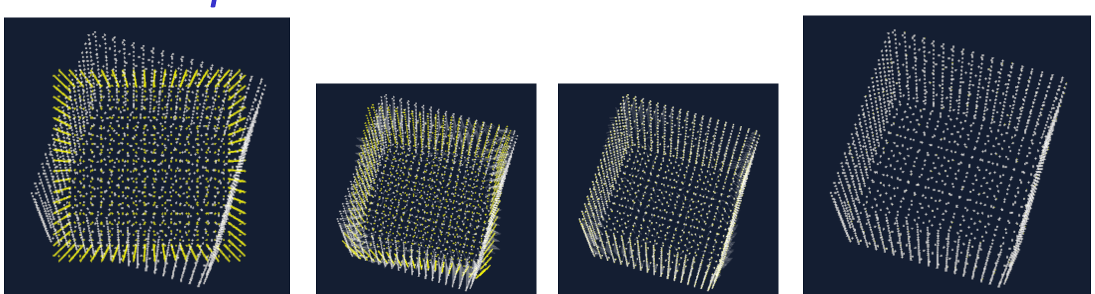


Point-to-point: 19 iterations



(accuracy problems)

Point-to-plane: 3 iterations



(much more accurate result)

Implementation

Problem:

- No closed form solution for the optimal rotation with point-to-plane correspondences

Solution:

- Numerical solution
- Setup non-linear optimization problem (rotation, translation = 6 parameters)
- Use non-linear optimization technique
- Remaining problem: *Parametrization of the rotations*
  - Trouble with singularities (spherical topology)

Local Linearization

Standard technique: local linearization

- Transformation: **T**(**x**) = **Rx** + **t**
- Linearize rotations:

$$T_{\alpha,\beta,\gamma} = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & \sin(\gamma) \\ 0 & -\sin(\gamma) & \cos(\gamma) \end{pmatrix}$$

$$\nabla T_{\alpha,\beta,\gamma} = \begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -y & 0 \end{pmatrix}$$

$$\Rightarrow T_{\alpha,\beta,\gamma}(x) \approx \mathbf{x} + \nabla T_{\alpha,\beta,\gamma}\mathbf{x} = \left(\begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -y & 1 \end{pmatrix} + \mathbf{I} \right) \mathbf{x}$$

Local Linearization

Standard technique: local linearization

- Numerical solution: iterative solver
- We have a current rotation $\mathbf{R}^{(i-1)}$ from the last iteration:
- Taylor expension at $\mathbf{R}^{(i-1)}$:

$$T_{\alpha,\beta,\gamma}^{(i)}(\mathbf{x}) \approx \left(\begin{pmatrix} 0 & \alpha & \beta \\ -\alpha & 0 & \gamma \\ -\beta & -y & 1 \end{pmatrix} + \mathbf{I} \right) \mathbf{R}^{(i-1)} \mathbf{x}$$

- Solve for $\mathbf{t}, \alpha, \beta, \gamma$ (linear expressison $\rightarrow$ quadratic opt.)

$$\underset{\substack{\alpha,\beta,\gamma \in \mathbb{R} \\ \mathbf{t} \in \mathbb{R}^3}}{\arg\min} \sum_{j=1}^{n} \left\langle \mathbf{R}^{(i)} \mathbf{p}_j^{(A)} + \mathbf{t} - \mathbf{p}_{nearest(j)}^{(B)}, \mathbf{n}_{nearest(j)}^{(B)} \right\rangle^2$$

Local Linearization

Then:

- Project $\mathbf{R}^{(i)}$ back on the manifold of orthogonal matrices. (for example using the SVD-based algorithm discussed before)
- Then iterate, until convergence.

Why does this work?

- The parametrization is non-degenerate
  - For large α, β, γ, the norm of the matrix increases arbitrarily (i.e.: the object size increases, away from the data)
  - Therefore, the least-squares optimization will perform a number of small steps rather than collapse.

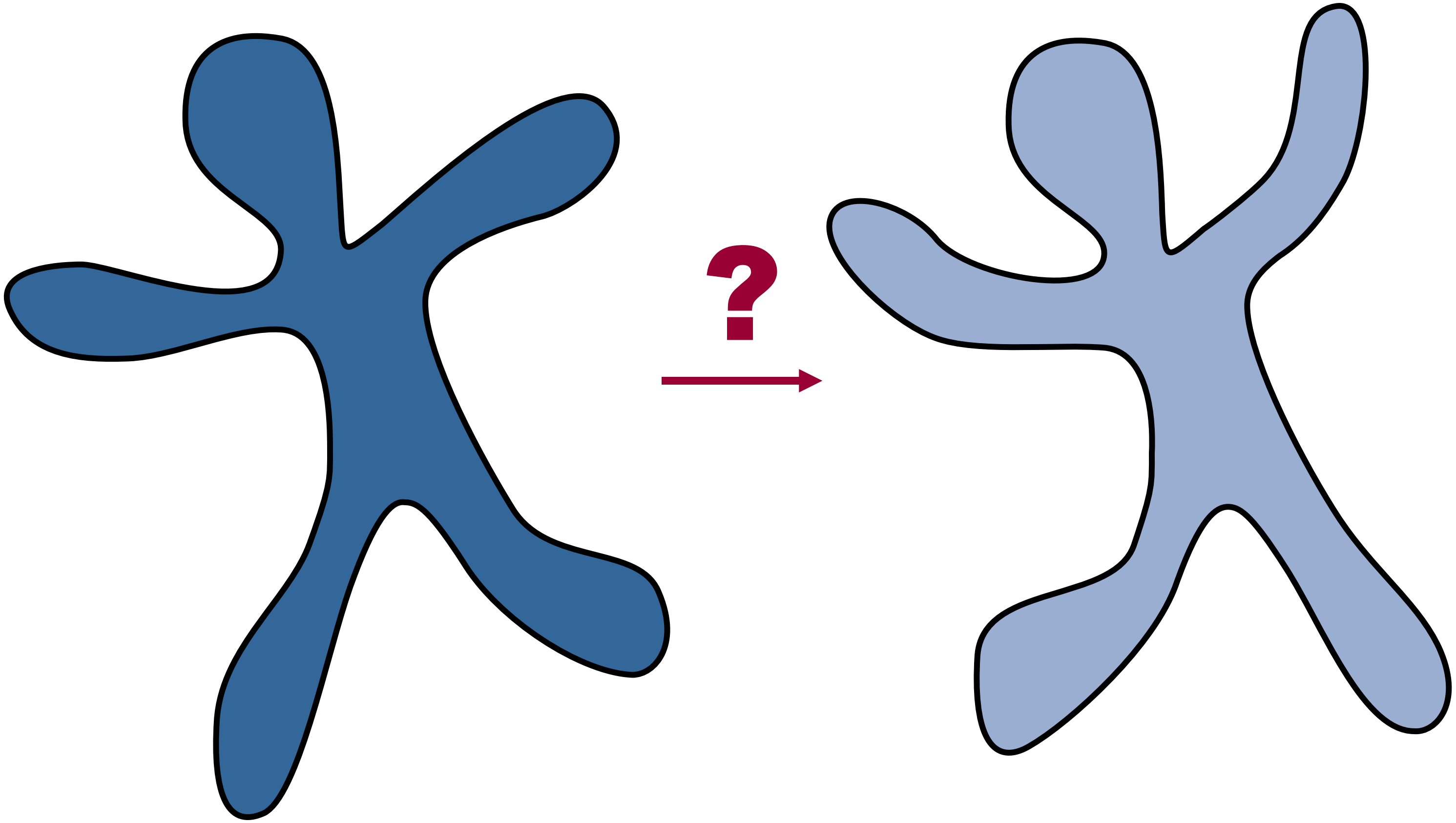
More Tricks & Tweaks

ICP Problems:

- Partial matching might lead to distortions / bias
  - Remove outliers (M-estimator, delete “far away points”, e.g. 20% percentile in point-to-point distance)
  - Remove normal outliers
    (if connection direction deviates from normal direction)
- Sampling problems
  - Problem: for example flat surface with engraved letters
  - No convergence in that case
  - Improvement: Sample correspondence points with distribution to cover unit sphere of normal directions as uniformly as possible

Deformable Shape Matching

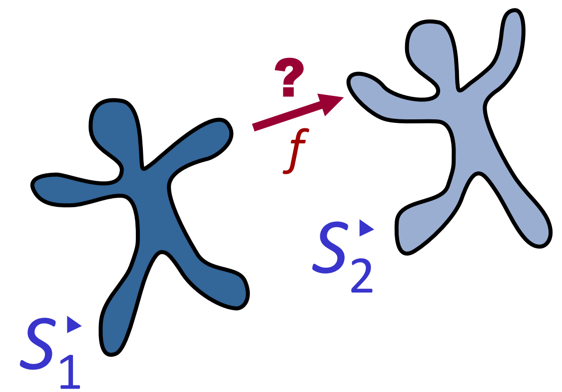
Example



What are the Correspondences?

What are we looking for?

Problem Statement:



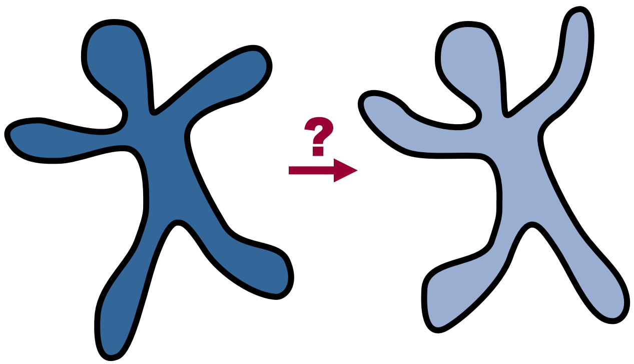
Given:

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

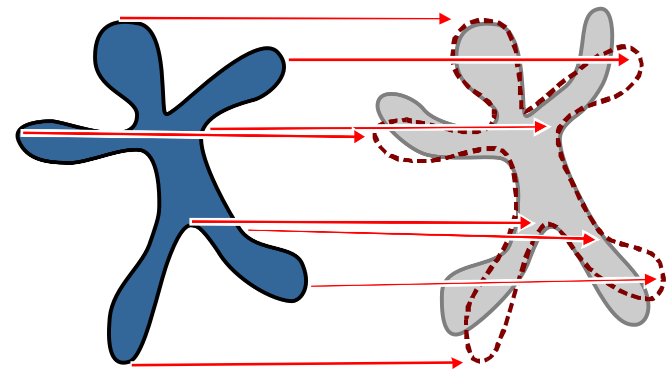
We are looking for:

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

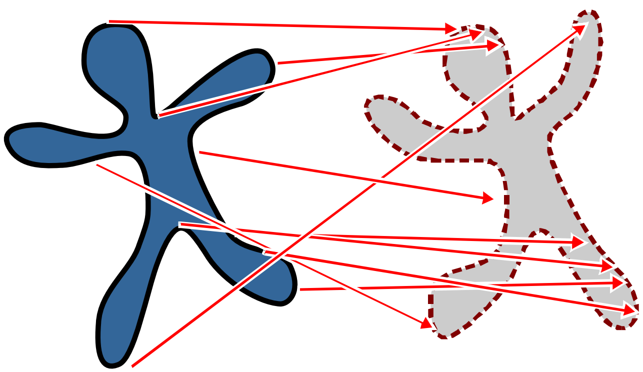
Example



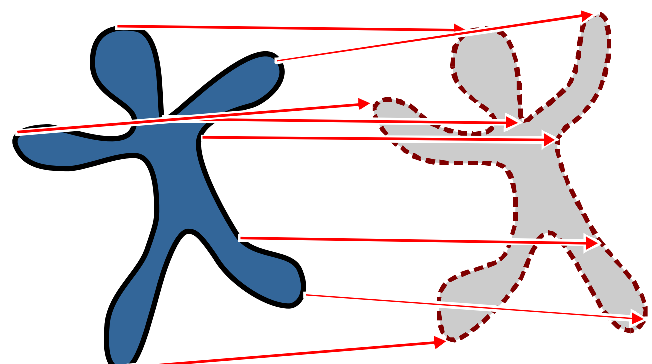
Correspondences?



no shape match



too much deformation

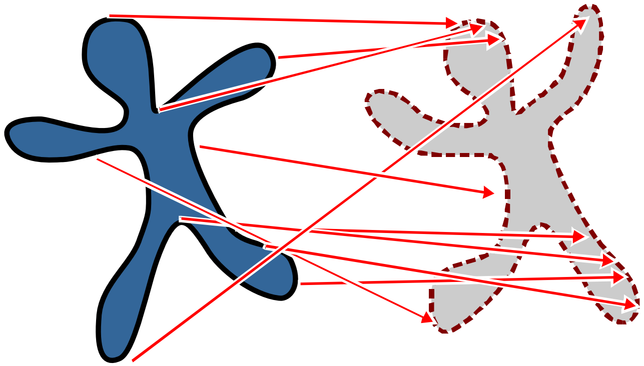


optimum

This is a Trade-Off

Deformable Shape Matching is a Trade-Off:

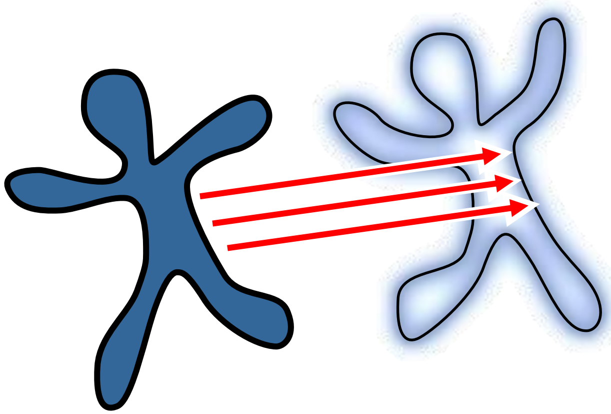
- We can match any two shapes using a weird deformation field



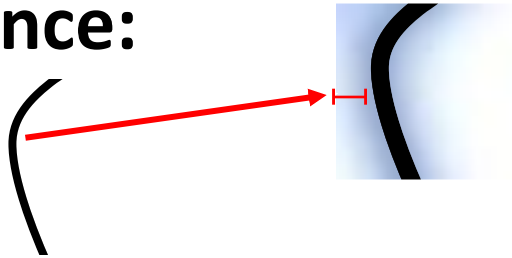
- We need to trade-off:
  - Shape matching (close to data)
  - Regularity of the deformation field (reasonable match)

Variational Model

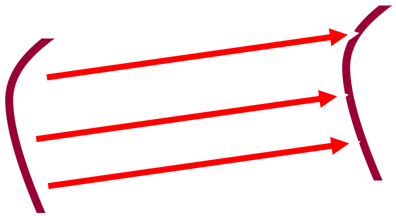
Components:



Matching Distance:



Deformation / rigidity:

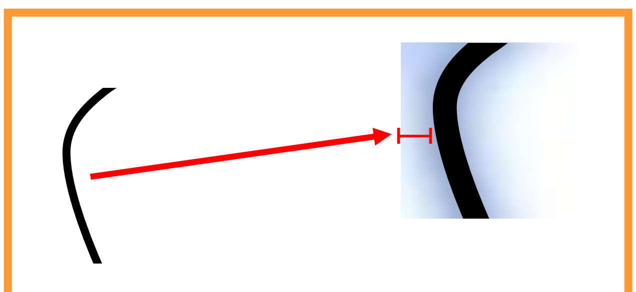


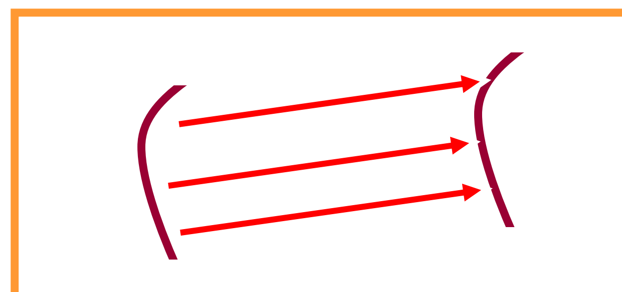
Variational Model

Variational Problem:

- Formulate as an energy minimization problem:

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



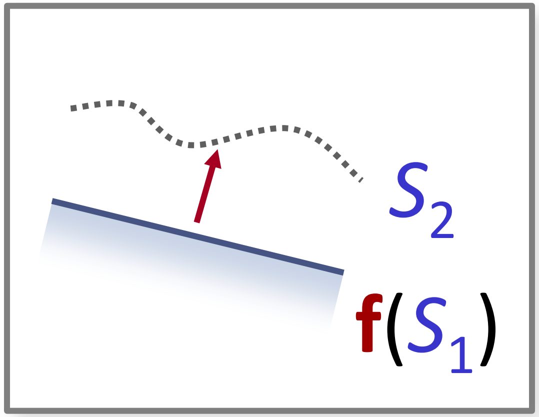


Part 1: Shape Matching

Assume:

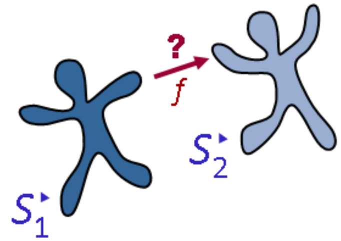
- Objective Function:

$$E^{(match)}(f) = dist\left(f_{1,2}(S_1), S_2\right)$$



- Example: least squares distance

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



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

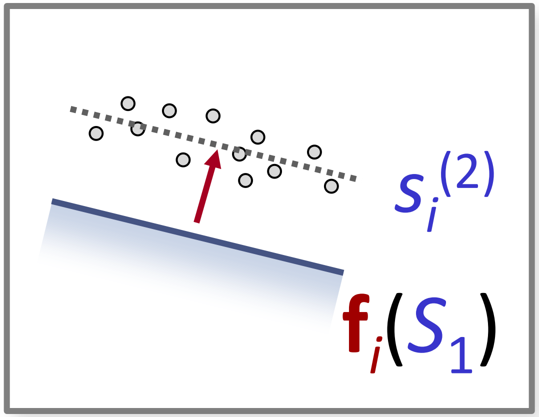
Point Cloud Matching

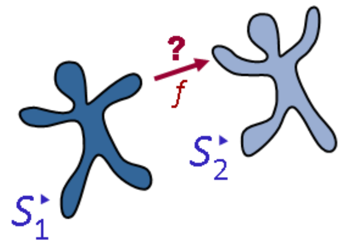
Implementation example: Scan matching

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

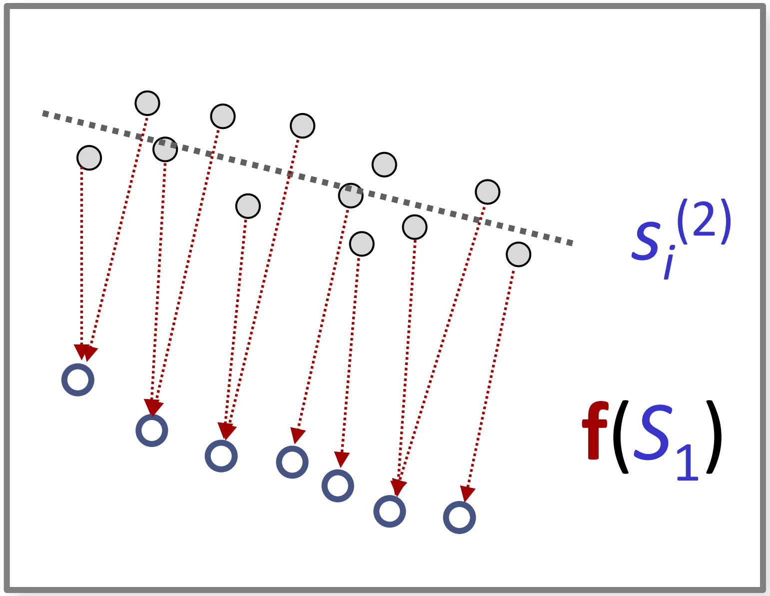
$$E^{(match)}(f) = \frac{|S_1|}{m} \sum_{i=1}^{m} dist\left(S_1, \mathbf{s}_i^{(2)}\right)^2$$

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





Surface approximation

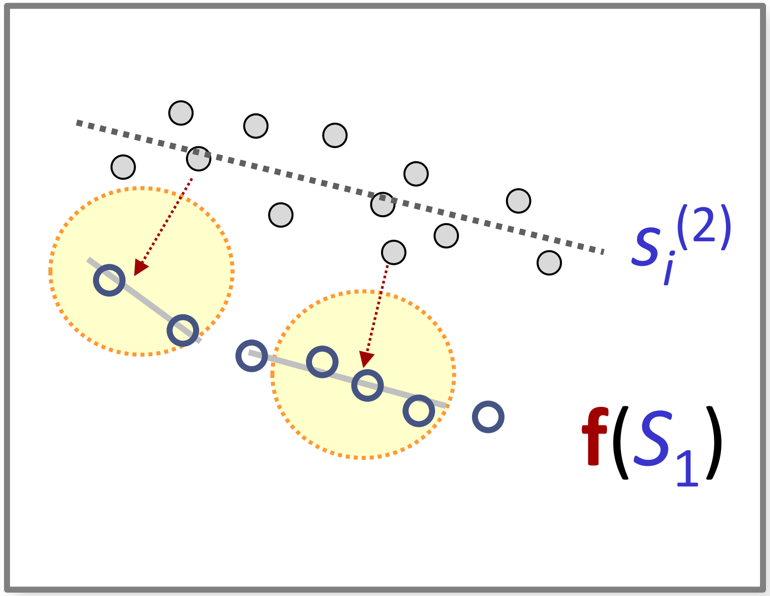


Solution #1: Closest point matching

- “Point-to-point” energy

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

Surface approximation



Solution #2: Linear approximation

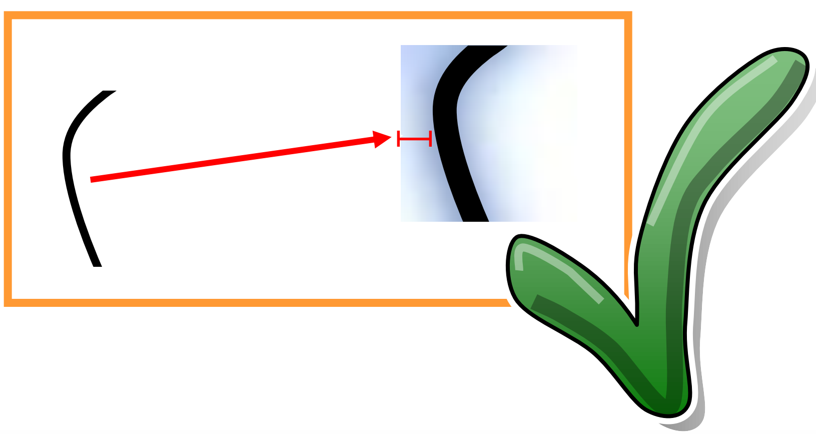
- "Point-to-plane" energy
- Fit plane to k-nearest neighbors
- k proportional to noise level, typically $k \approx 6 \ldots 20$

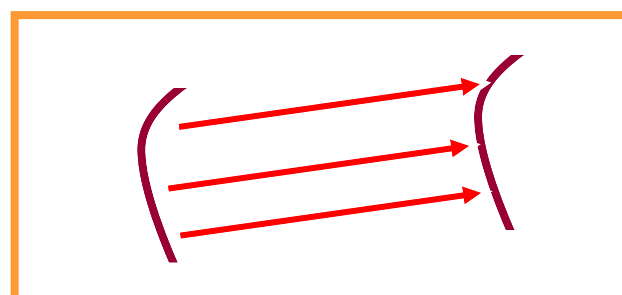
Variational Model

Variational Problem:

- Formulate as an energy minimization problem:

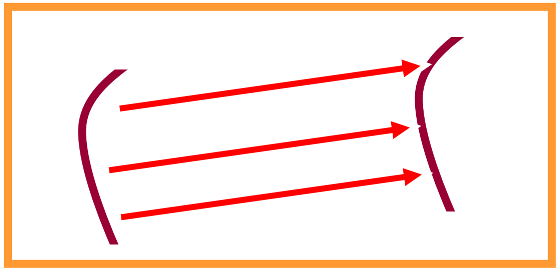
$$E(f) = \boxed{E^{(match)}(f)} + \boxed{E^{(regularizer)}(f)}$$





Part II: Deformation Model

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



What is a "nice" deformation field?

- Isometric "elastic" energies
  - Extrinsic ("volumetric deformation")
  - Intrinsic ("as-isometric-as possible embedding")
- Thin shell model
  - Preserves shape (metric *plus curvature*)
- Thin-plate splines
  - Allowing strong deformations, but keep shape

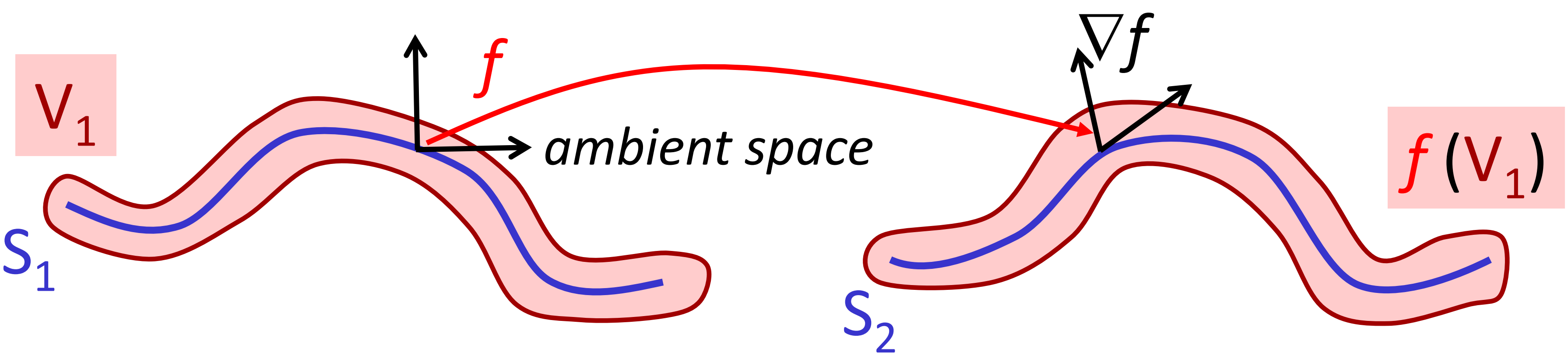
Elastic Volume Model

Extrinsic Volumetric "As-Rigid-As Possible"

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

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

first fundamental form $\mathbf{I}$ ($\mathbb{R}^{3\times 3}$)



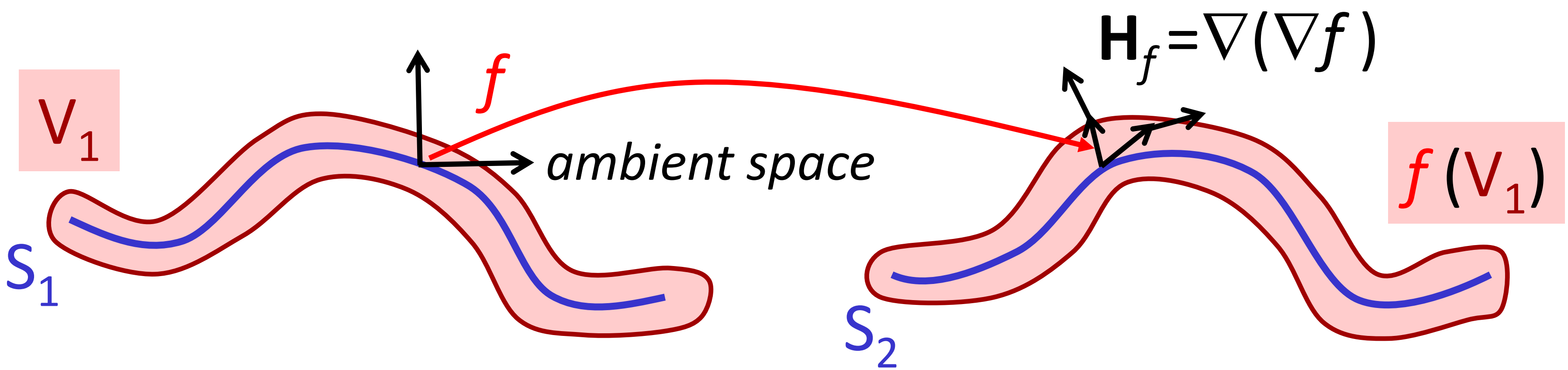
Volume Model

Variant: Thin-Plate-Splines

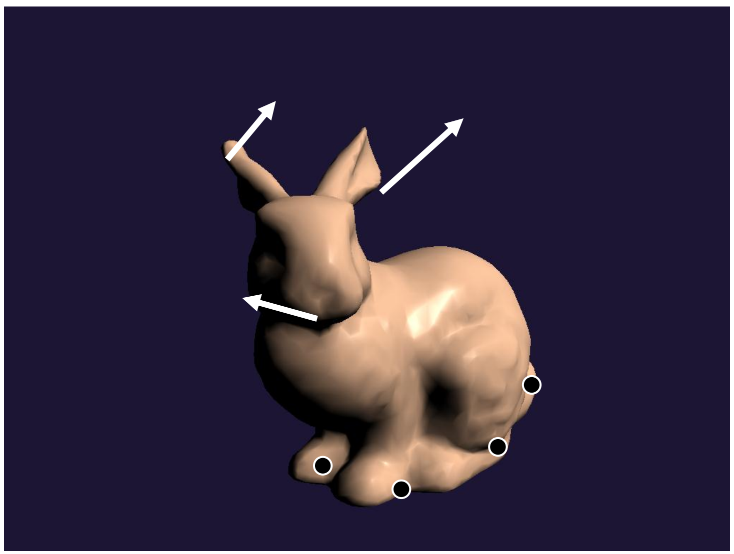
- Use regularizer that penalizes curved deformation

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

second derivative ($\mathbb{R}^{3\times 3}$)

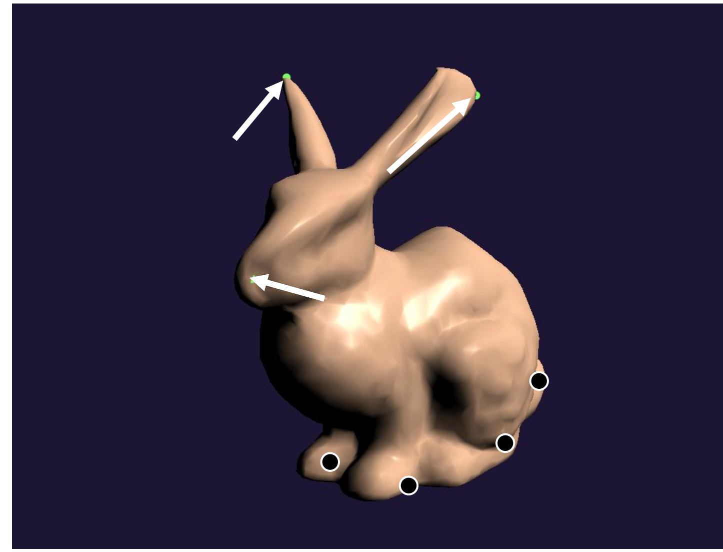


How does the deformation look like?

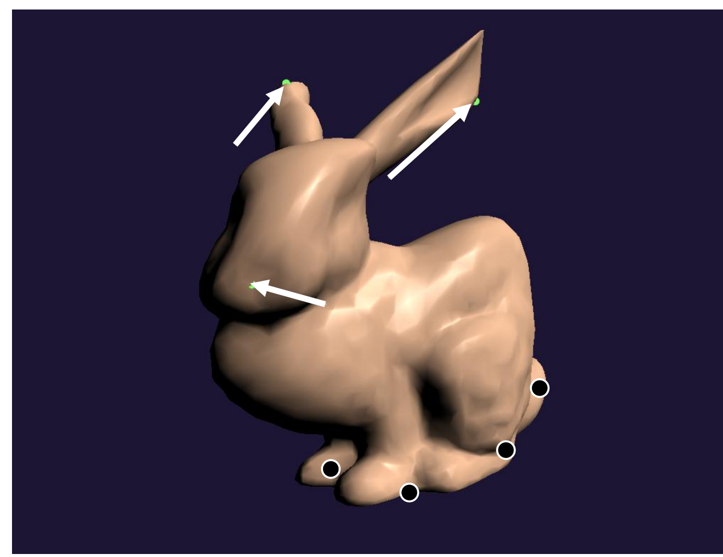


original

as-rigid-as possible volume



thin plate splines



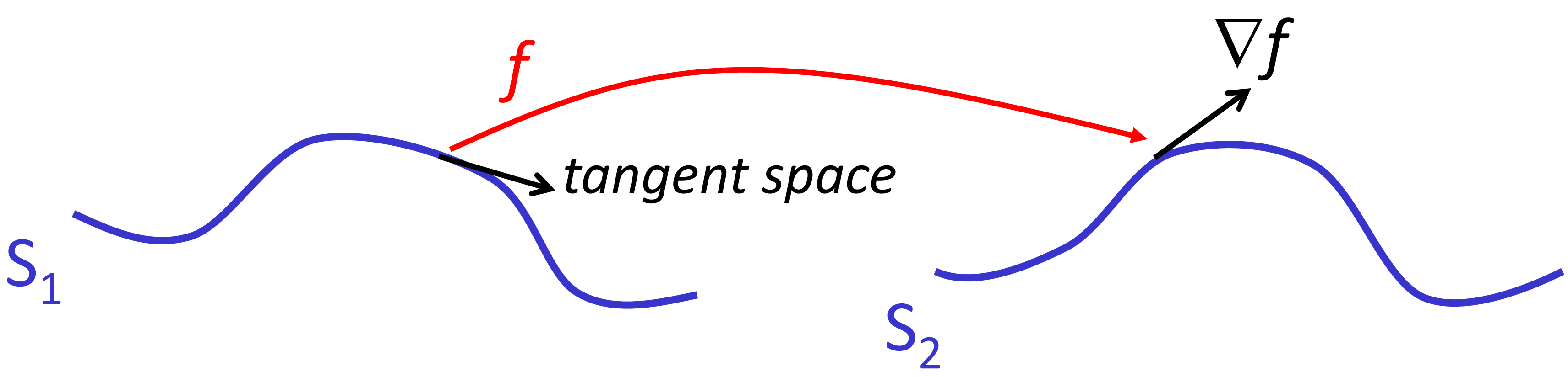
Isometric Regularizer

Intrinsic Matching (2-Manifold)

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

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

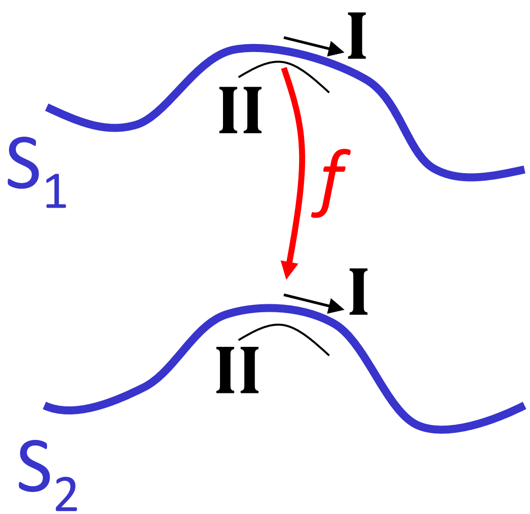
first fund. form (S_1, intrinsic)



Elastic "Thin Shell" Regularizer

"Thin Shell" Energy

- Differential geometry point of view
  - Preserve first fundamental form **I**
  - Preserve second fundamental form **II**
  - …in a least least-squares sense
- Complicated to implement
- Usually approximated

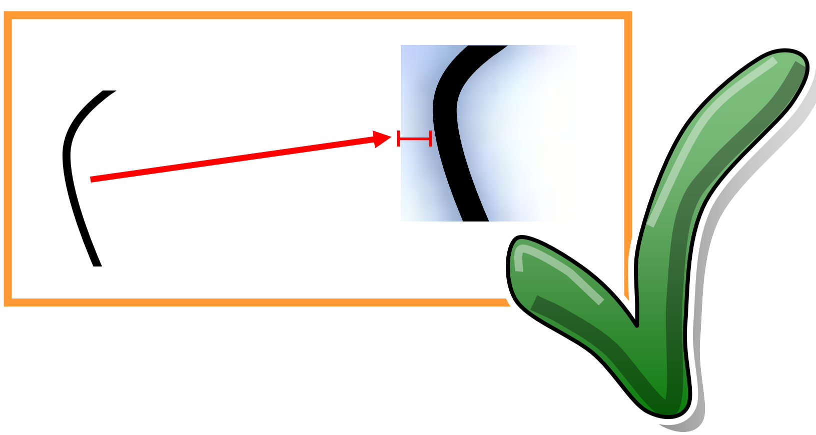


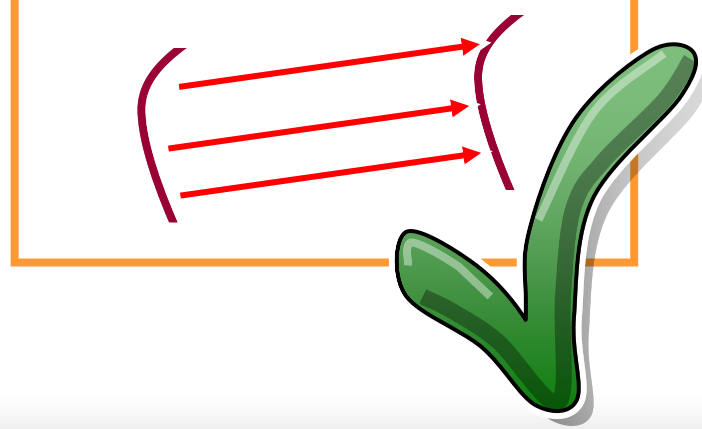
Deformable ICP

How to build a deformable ICP algorithm

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

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





Deformable ICP

How to build a deformable ICP algorithm

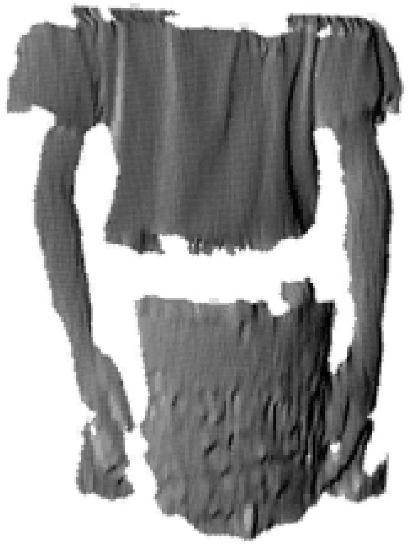
- Pick a surface distance measure
- Pick an deformation model / regularizer
- Initialize $f(S_1)$ with S_1 (i.e., f = id)
- Pick a non-linear optimization algorithm
  - Gradient decent (easy, but bad performance)
  - Preconditioned conjugate gradients (better)
  - Newton, Gauss Newton (even better, but more work)
  - LBGFS (quick & effective)
  - Always use analytical derivatives!
- Run optimization

Animation Reconstruction

Real-time Scanners

space-time stereo

courtesy of James Davis, UC Santa Cruz



color-coded structured light

courtesy of Phil Fong, Stanford University

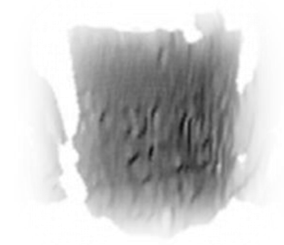
motion compensated structured light

courtesy of Sören König, TU Dresden

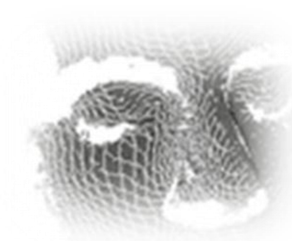
Animation Reconstruction

Problems

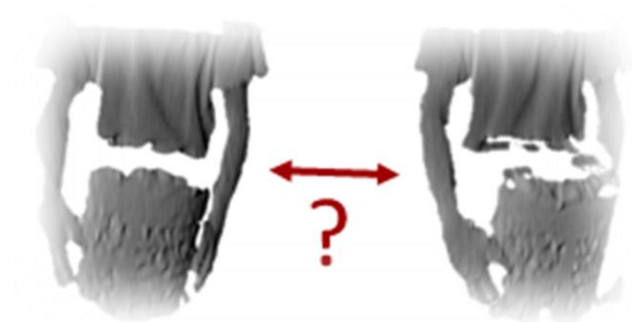
- Noisy data
- Incomplete data (acquisition holes)
- No correspondences



noise



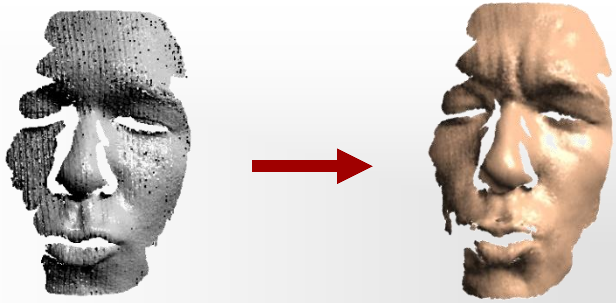
holes



missing correspondences

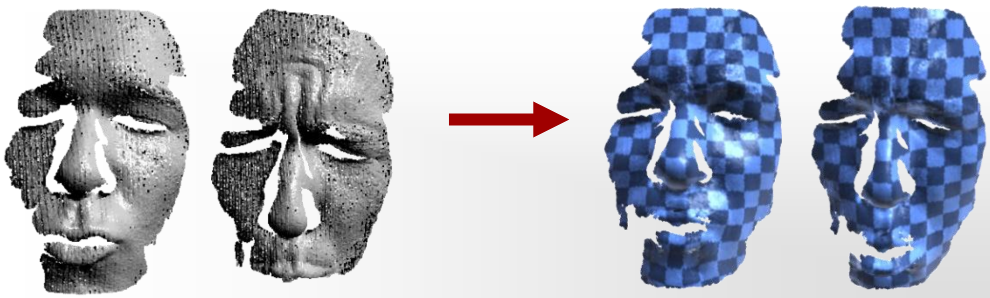
Animation Reconstruction

Remove noise, outliers



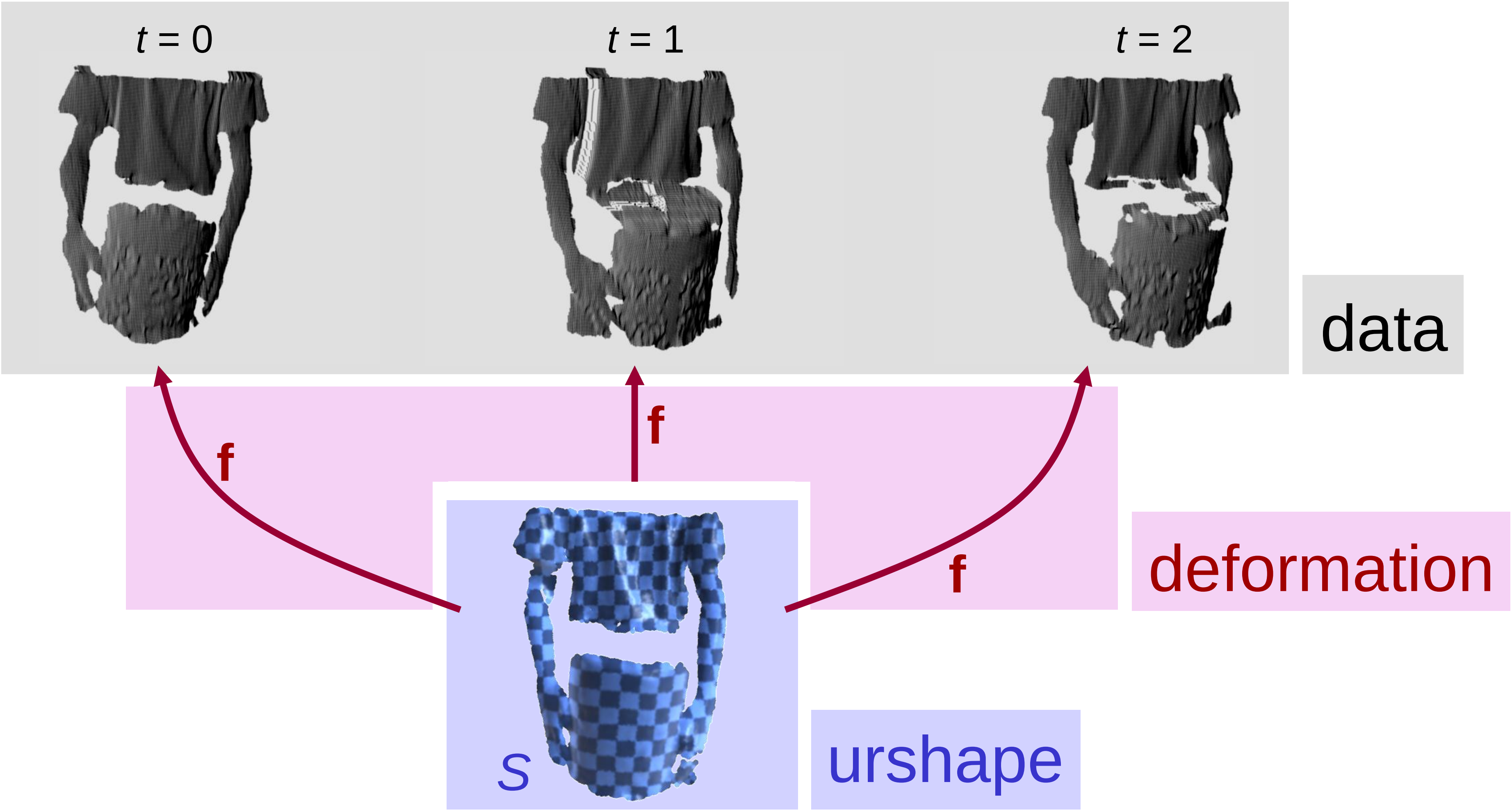
Fill-in holes
(from all frames)

Dense correspondences



Urshape Factorization Approach

“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*

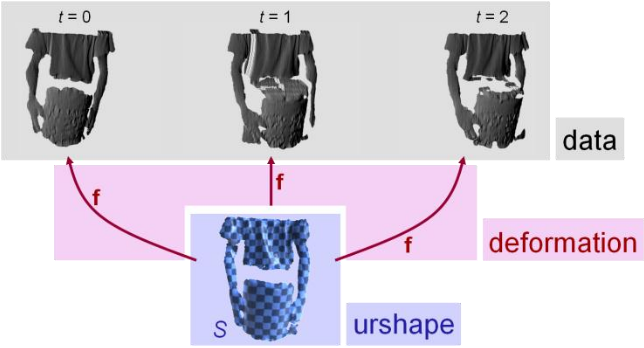
Numerical Discretization

- *Shape*
- *Deformation*

Domain Assembly

- Getting an initial estimate
- *Urshape* assembly

Energy Minimization



Energy Function

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

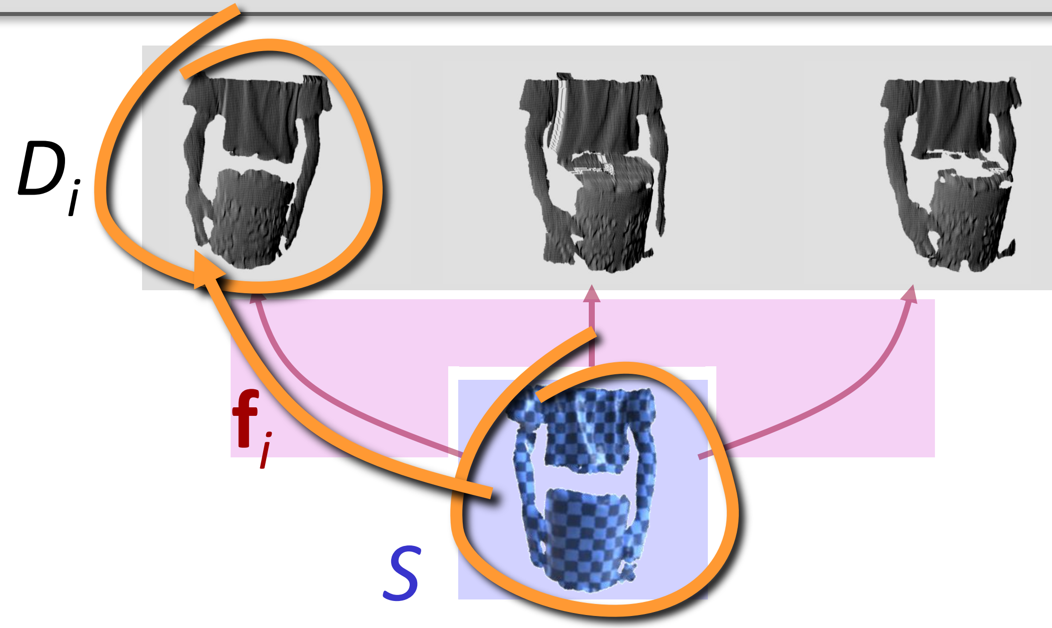
Components

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

Optimize S, $\mathbf{f}$ alternatingly

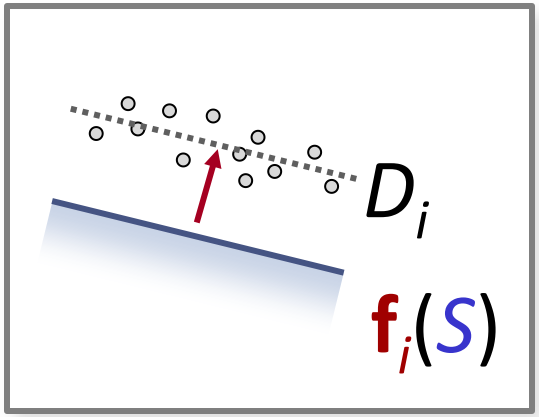
Data Fitting

$E_{data}(\mathbf{f}, S)$



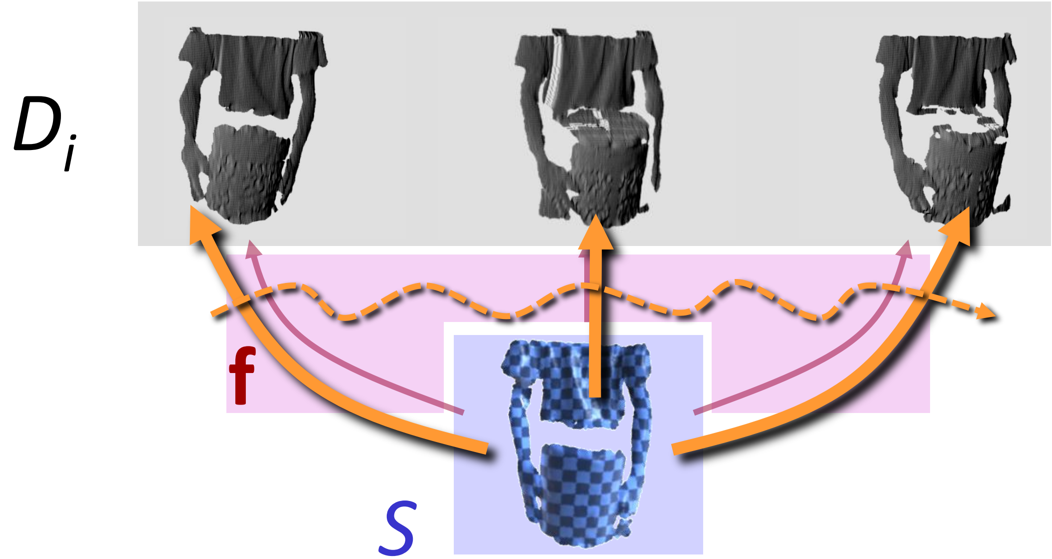
Data fitting

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



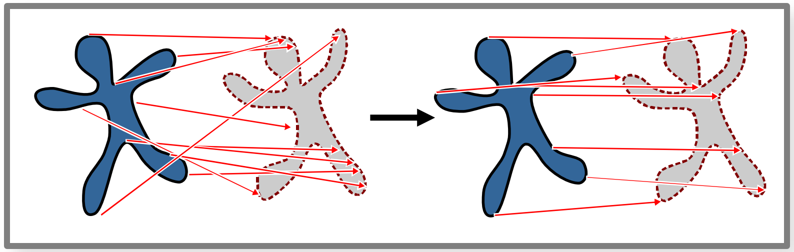
Elastic Deformation Energy

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

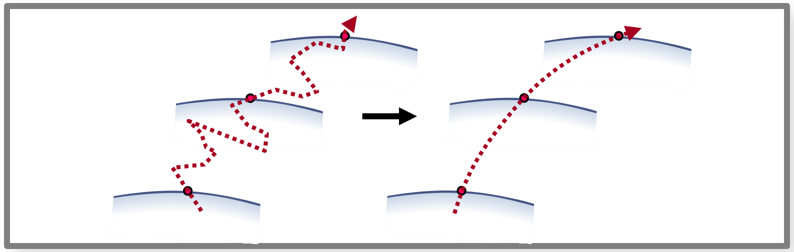


Regularization

- Elastic energy

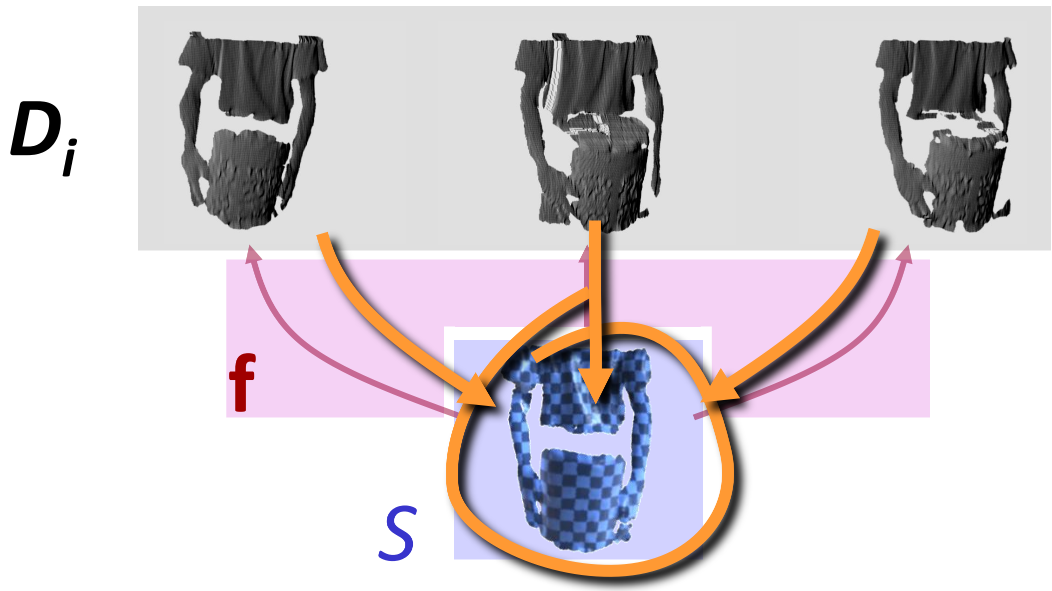


- Smooth trajectories



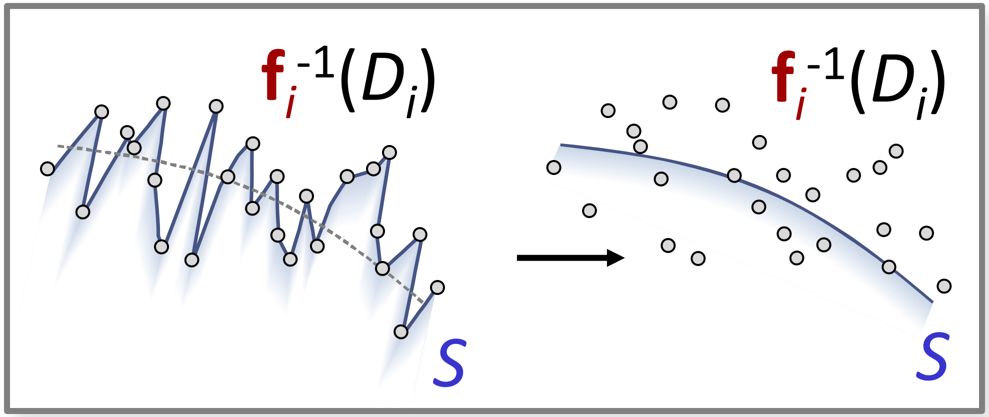
Surface Reconstruction

$E_{smooth}(S)$

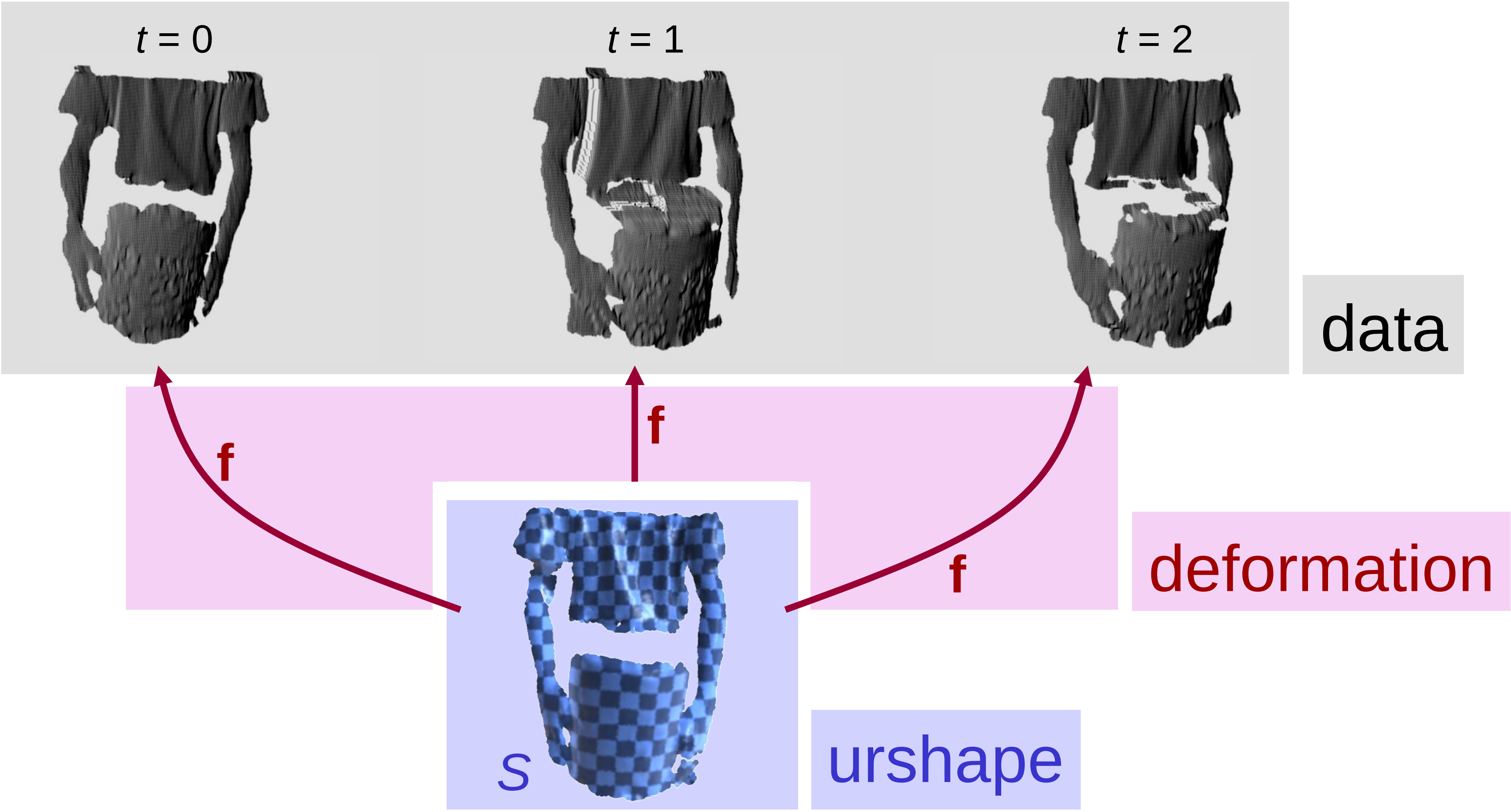


Data fitting

- Smooth surface
- Fitting to noisy data



Factorization



Summary: Variational Model

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

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

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

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

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

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

Components

Variational Model

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

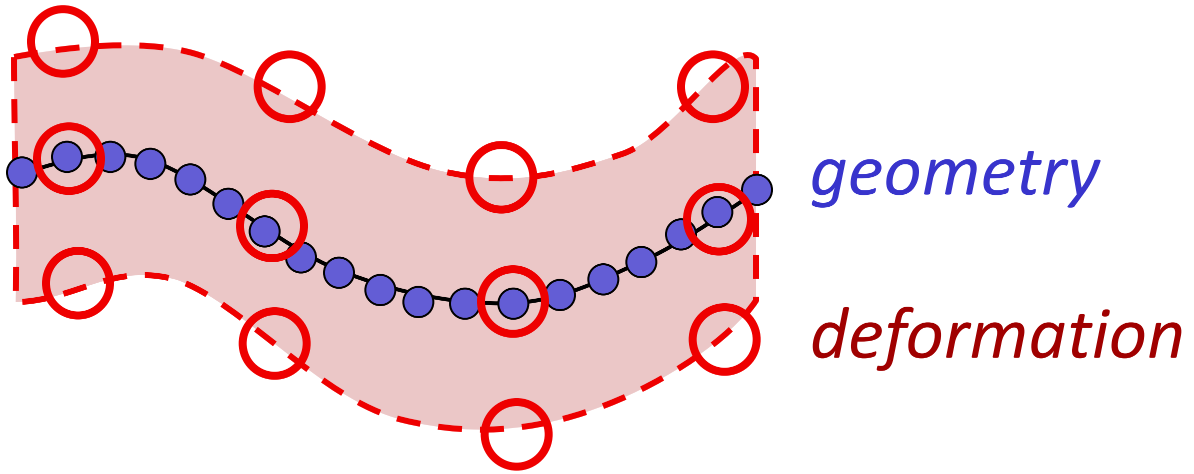
Numerical Discretization

- *Shape*
- *Deformation*

Domain Assembly

- Getting an initial estimate
- *Urshape* assembly

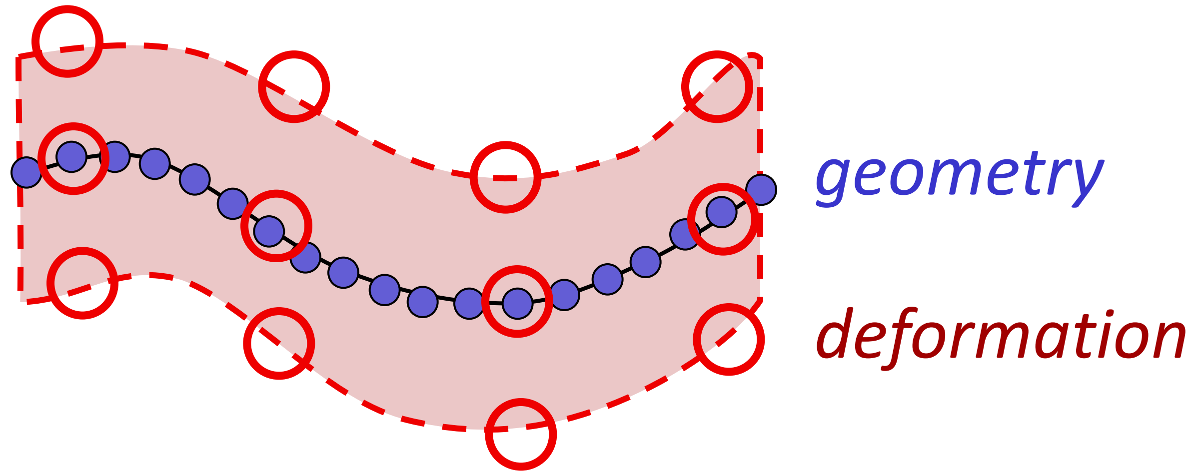
Discretization



Sampling:

- Full resolution *geometry*
- Subsample *deformation*

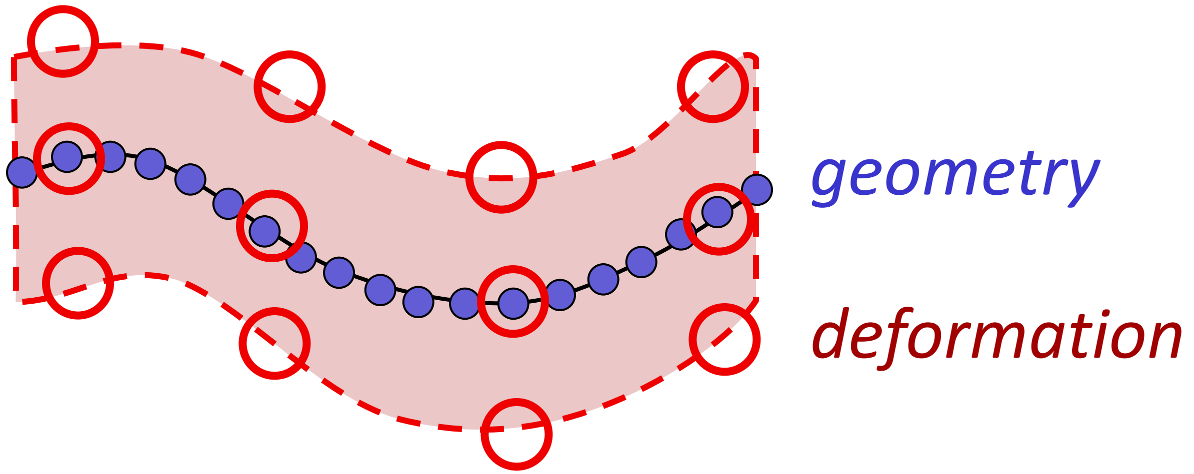
Discretization



Sampling:

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

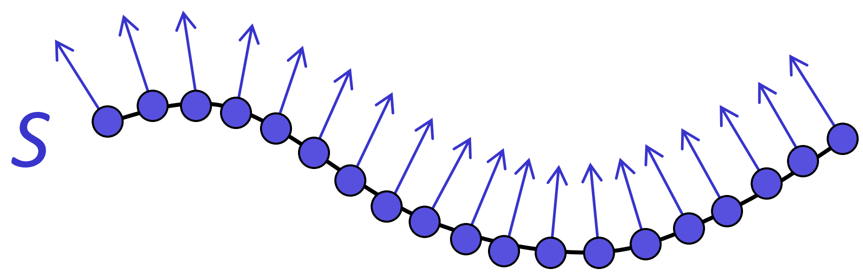
Discretization



Sampling:

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

Shape Representation



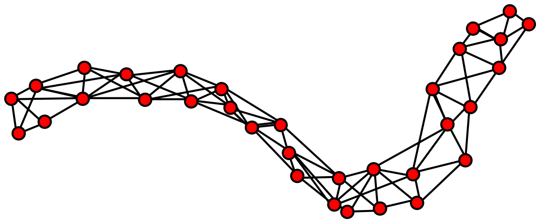
Shape Representation:

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

...but how about Neighborhoods?

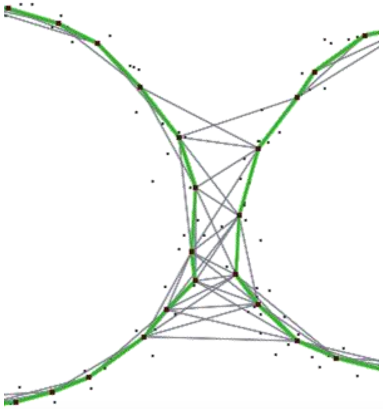
Topology estimation

- 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

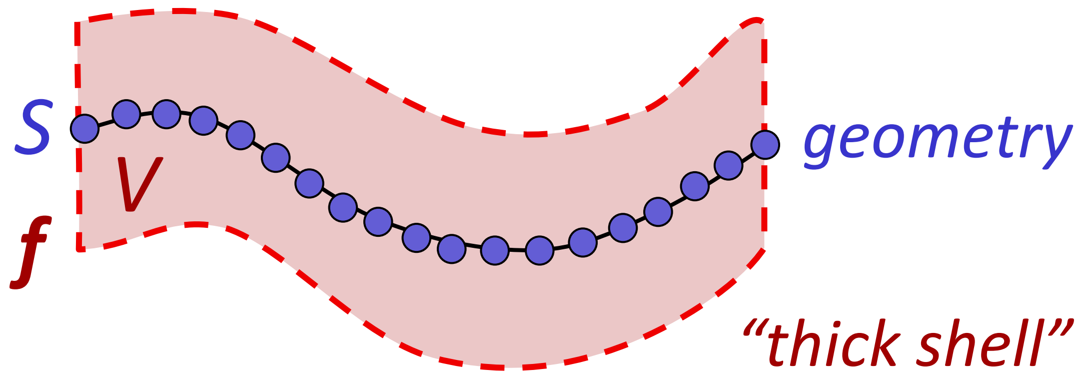


Limitations

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



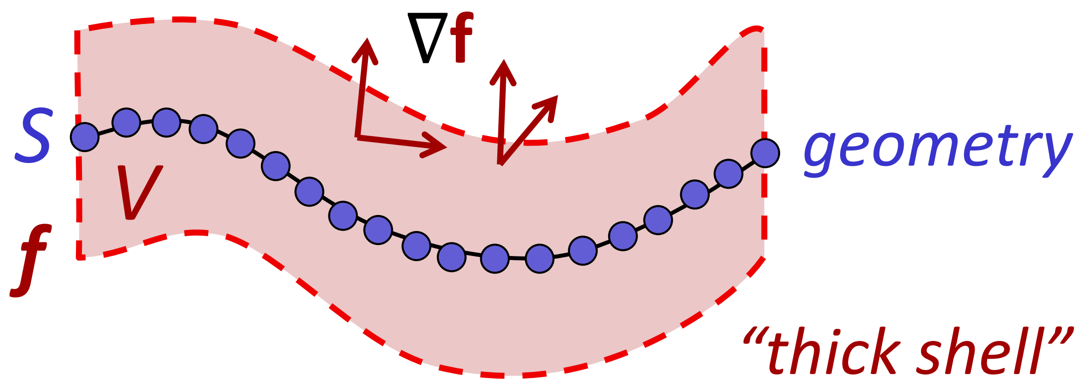
Deformation



Volumetric Deformation Model

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

Deformation



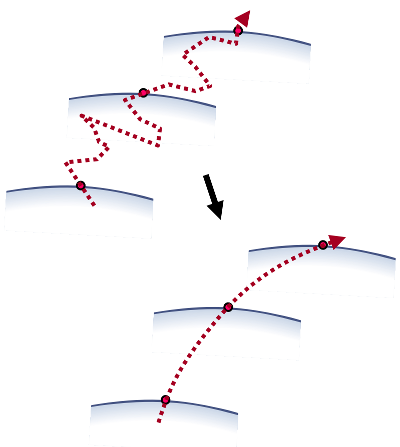
Deformation Energy

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

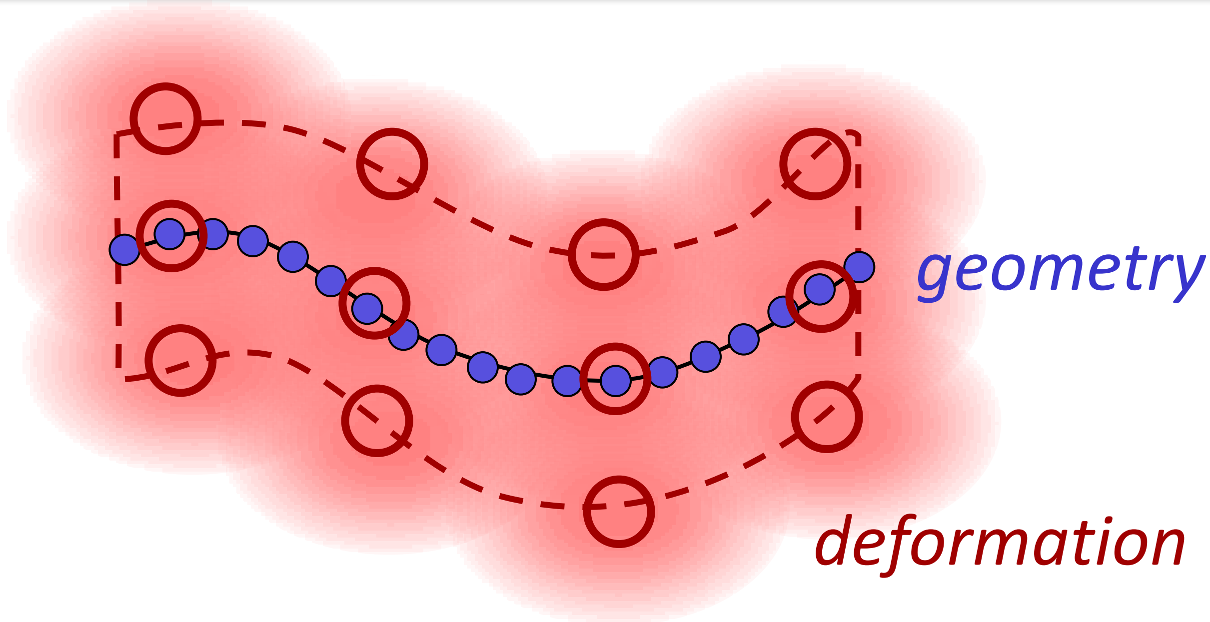
Additional Terms

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



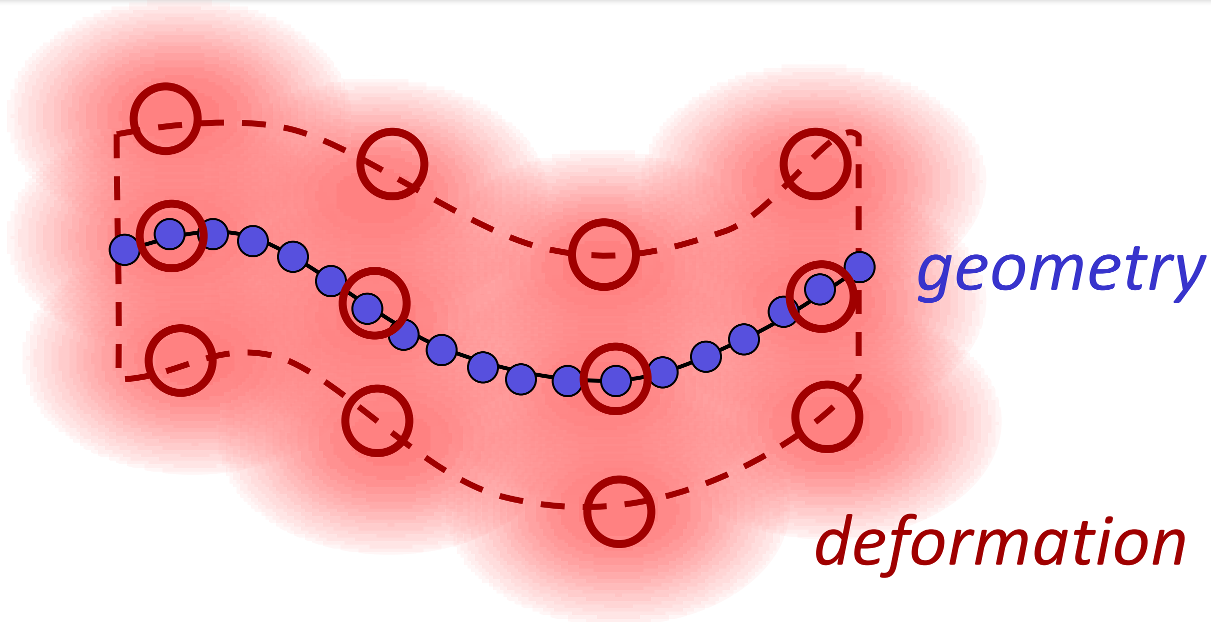
Discretization



How to represent the deformation?

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

Discretization



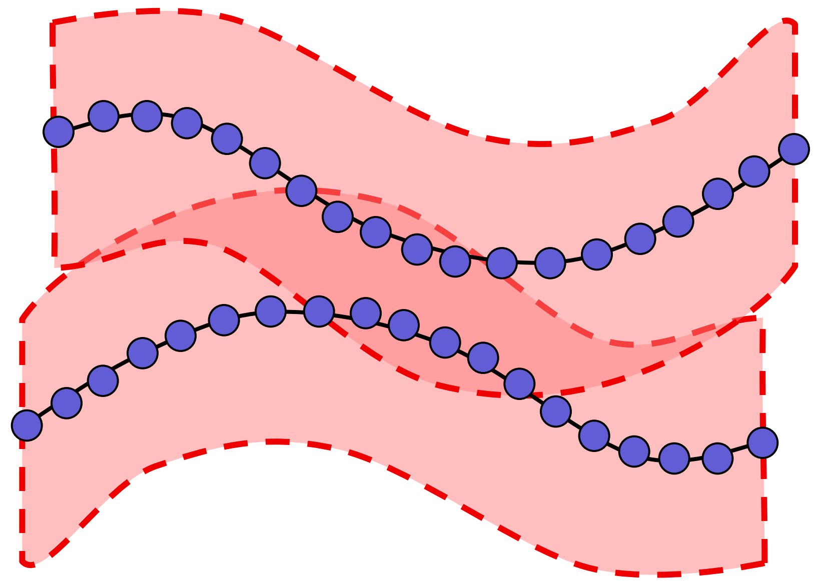
Meshless finite elements

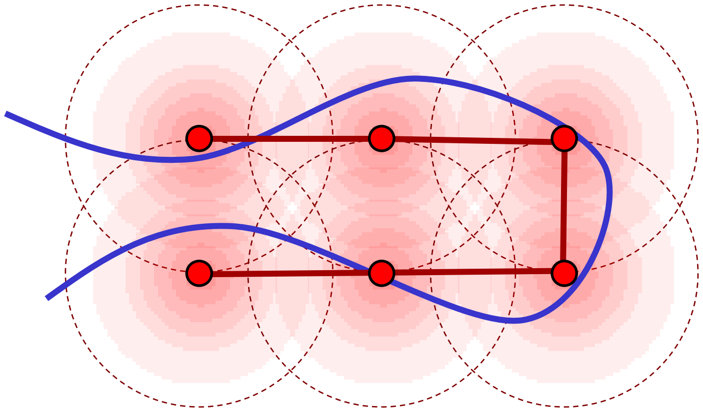
- Partition of unity, smoothness
- Linear precision
- Adaptive sampling is easy

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

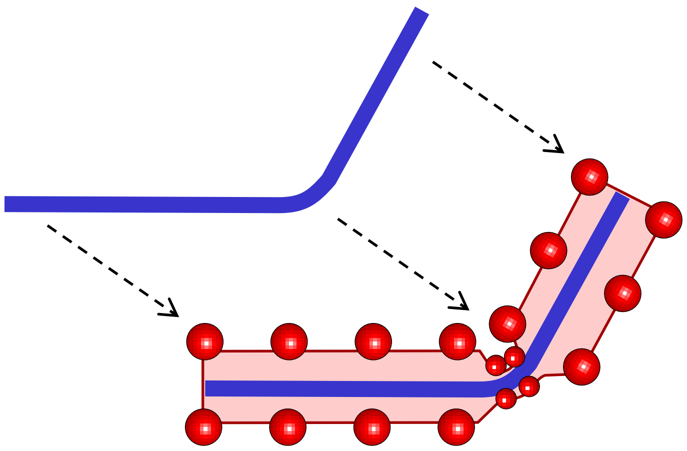




Adaptive Sampling

Adaptive Sampling

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



Components

Variational Model

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

Numerical Discretization

- *Deformation*
- *Shape*

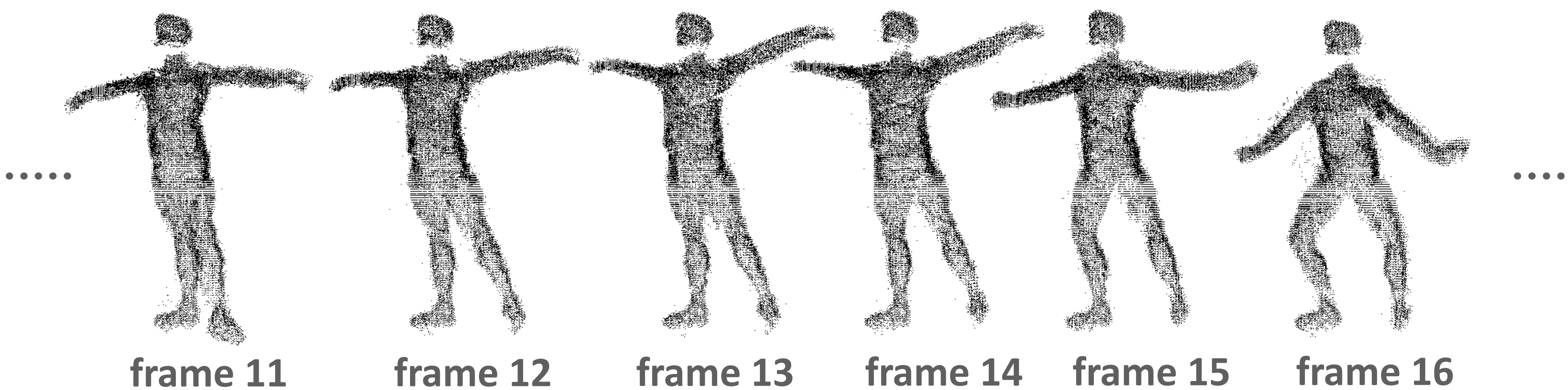
Domain Assembly

- Getting an initial estimate
- *Urshape* assembly

Urshape Assembly

Adjacent frames are similar

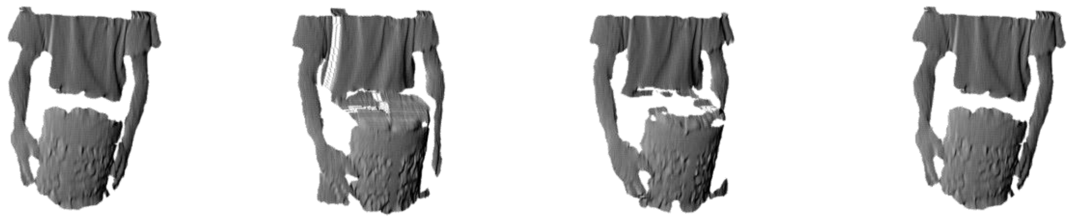
- Solve for frame pairs first
- Assemble urshape step-by-step



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

Hierarchical Merging

data



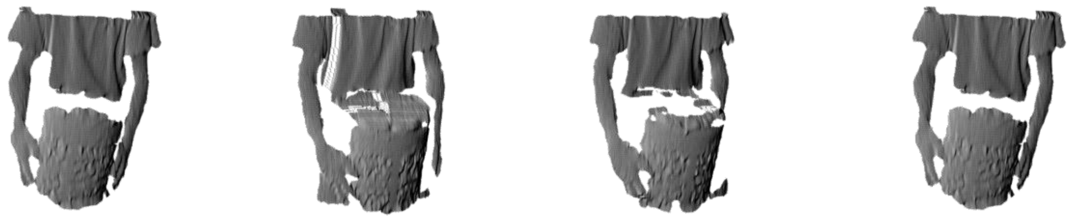
$\mathbf{f}(S)$

$\mathbf{f}$

S

Hierarchical Merging

data



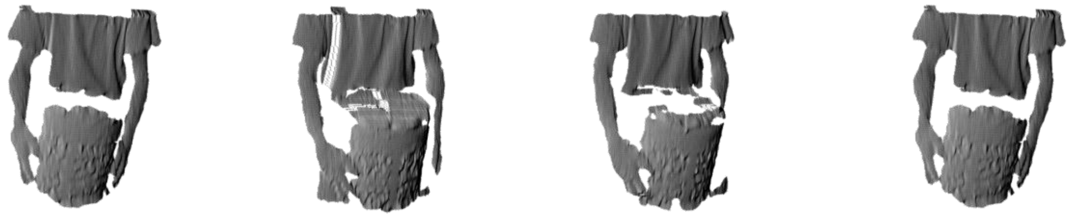
$\mathbf{f}(S)$

$\mathbf{f}$

S

Initial Urshapes

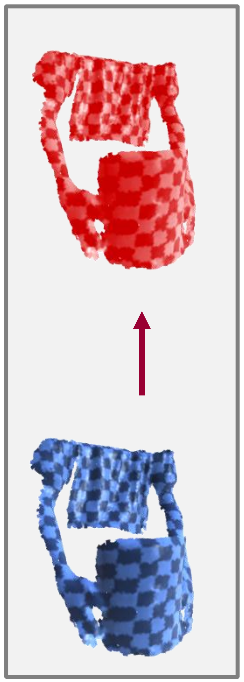
data



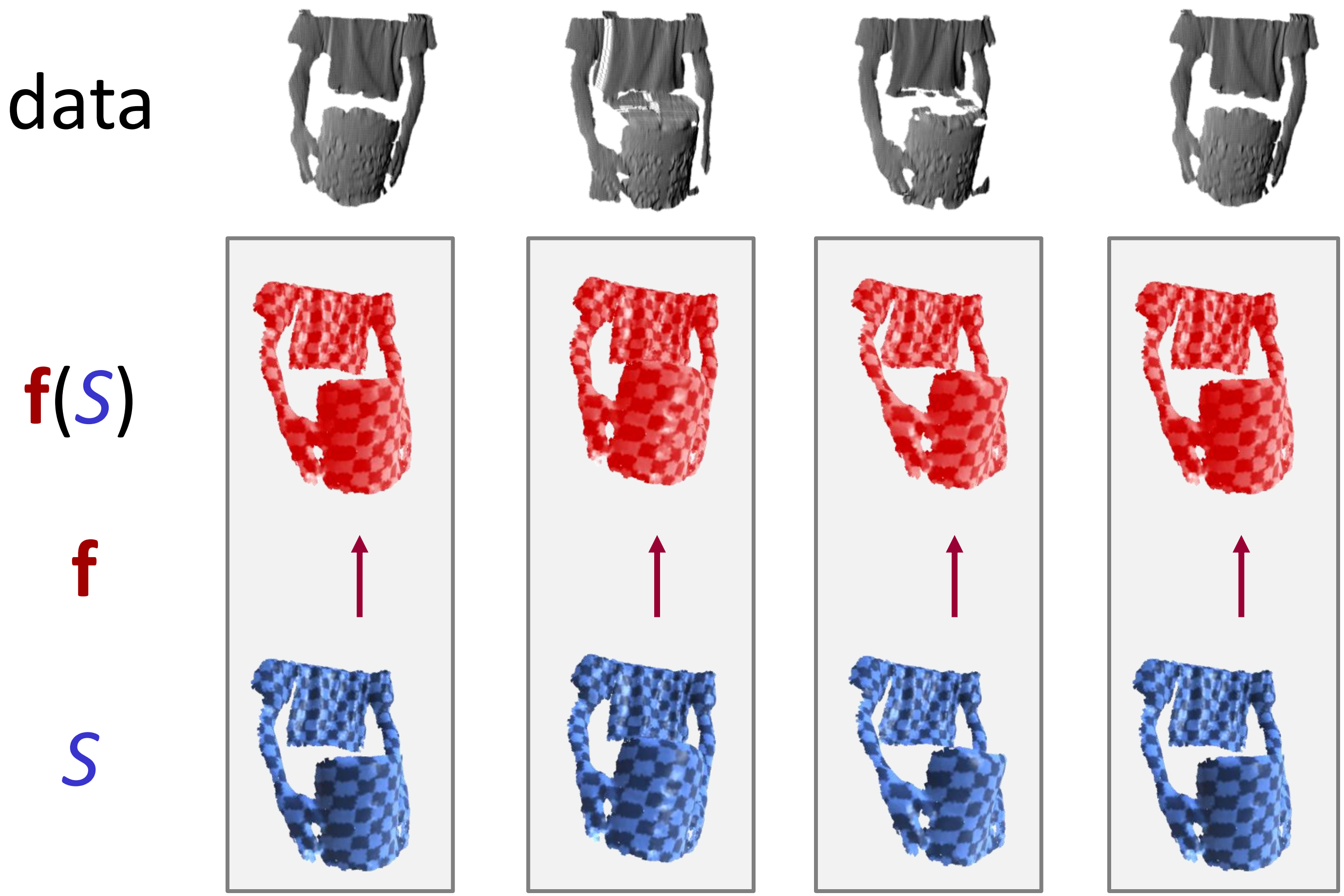
f(S)

f

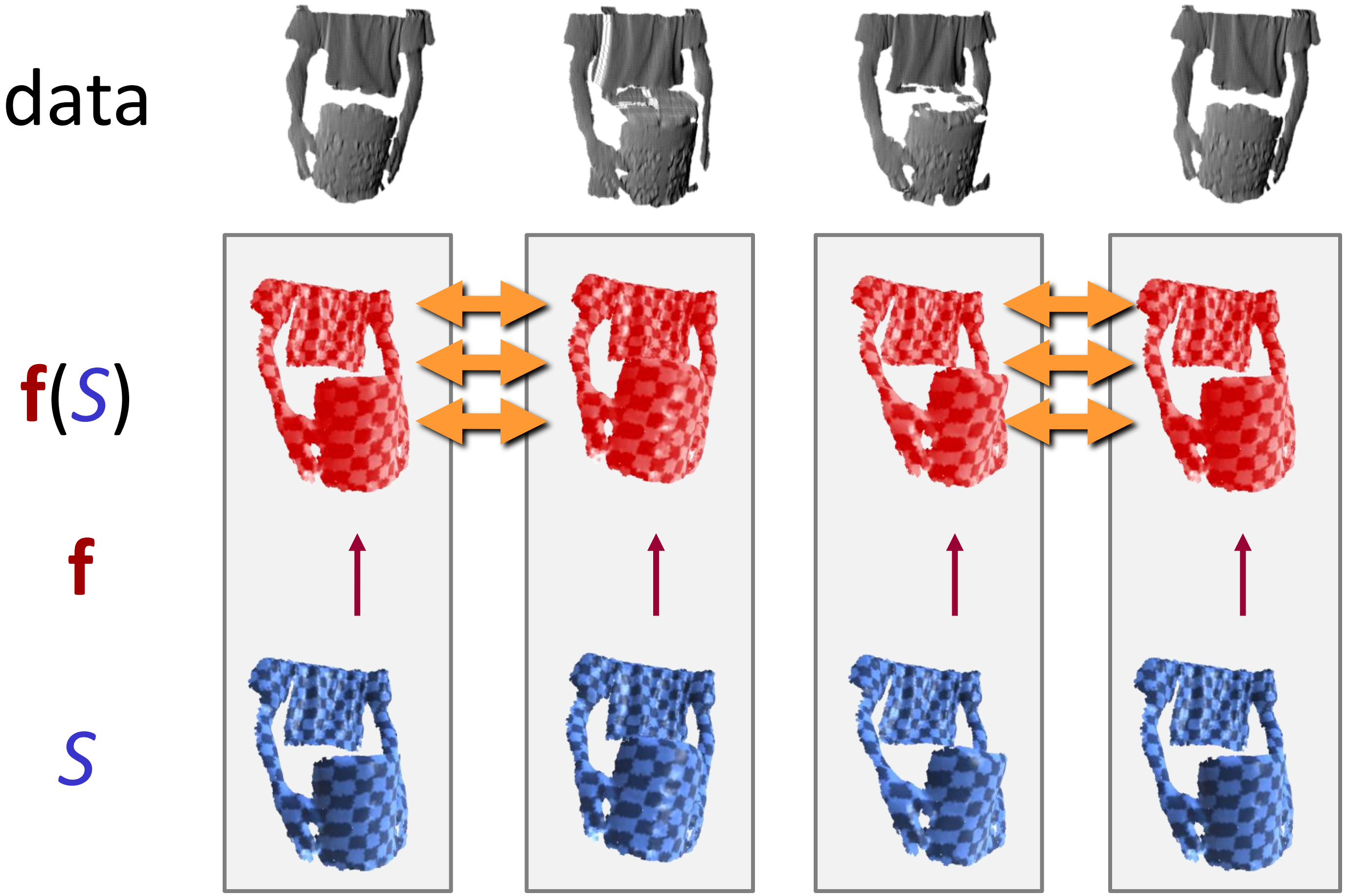
S



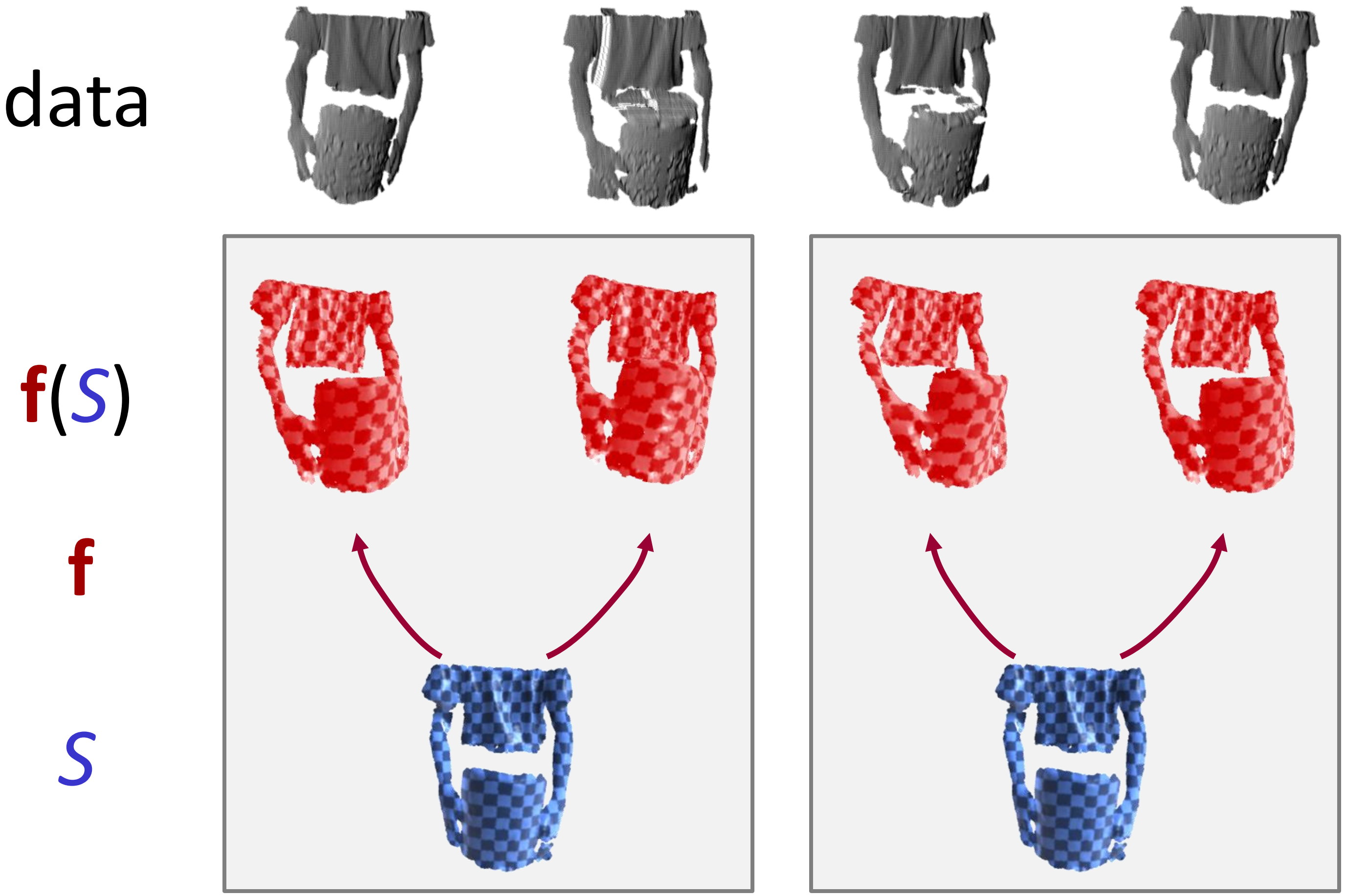
Initial Urshapes



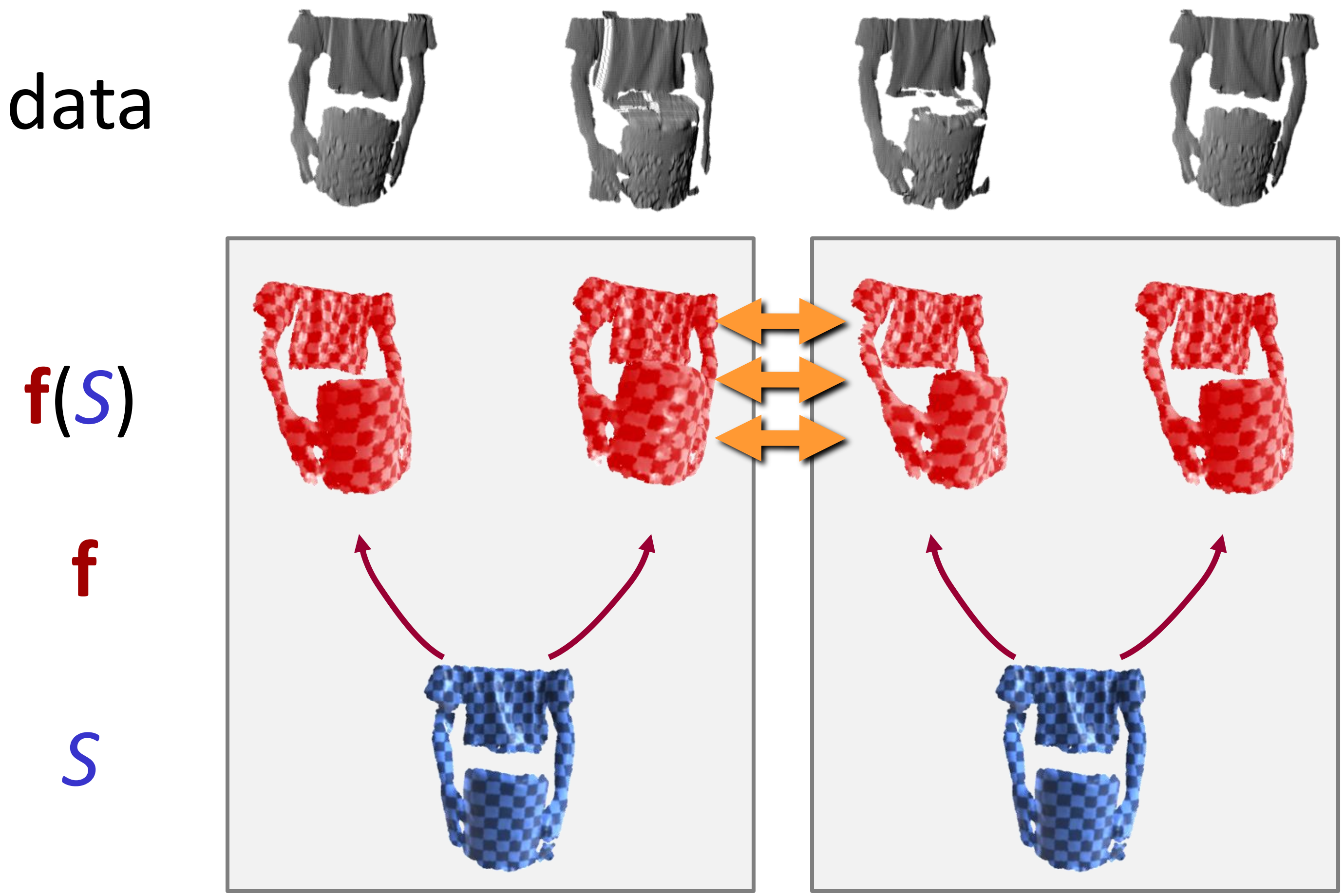
Alignment



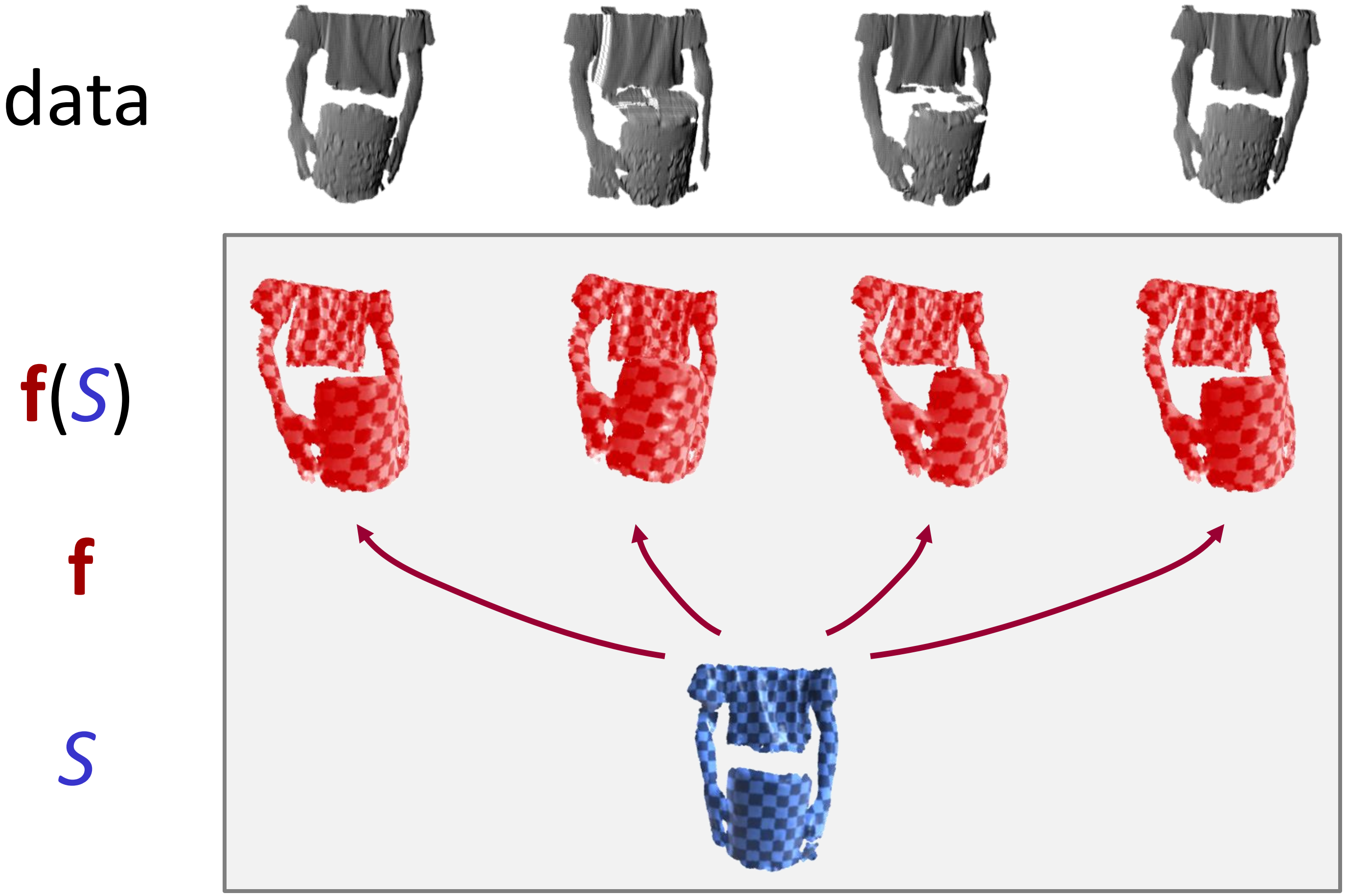
Align & Optimize



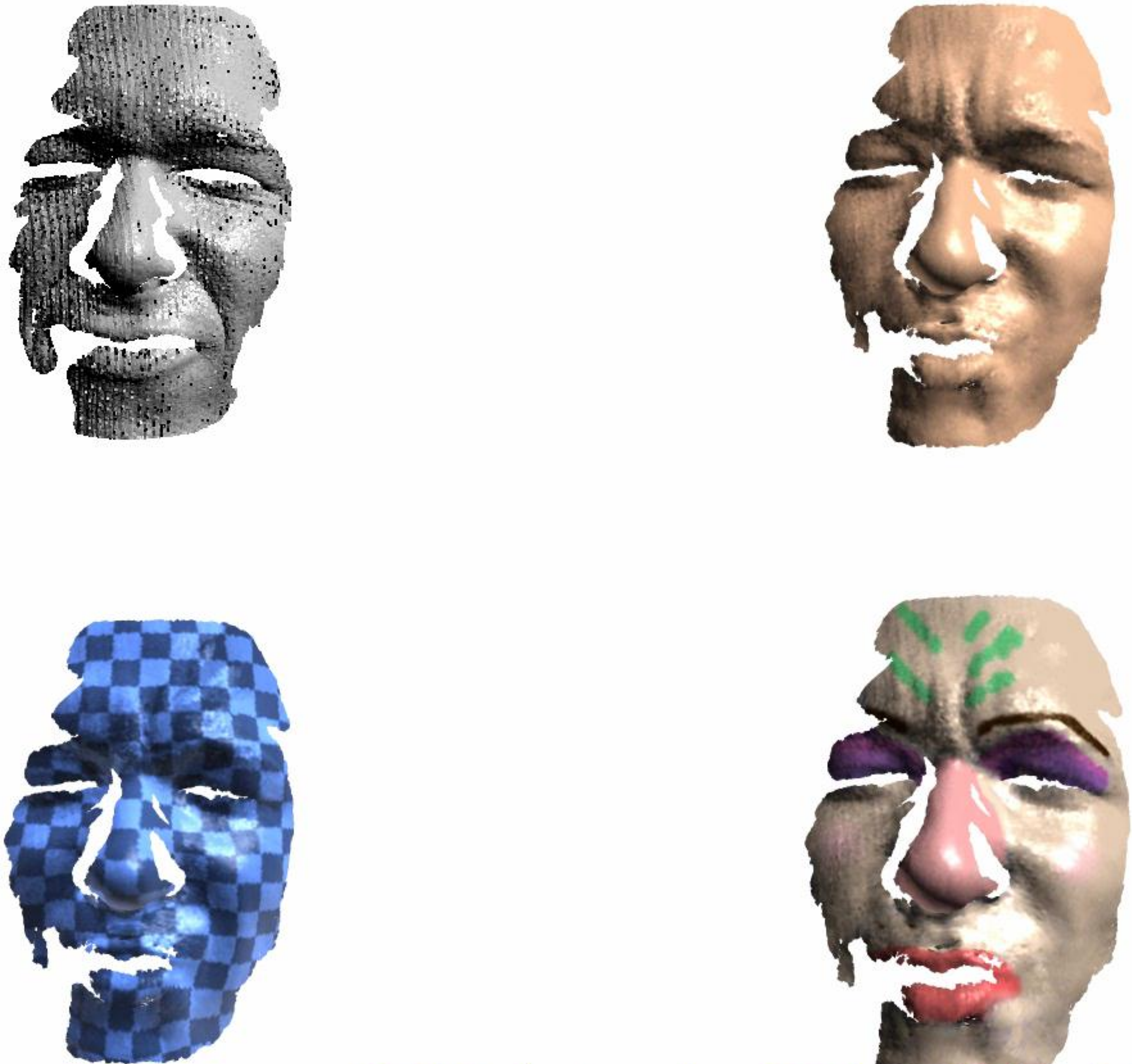
Hierarchical Alignment



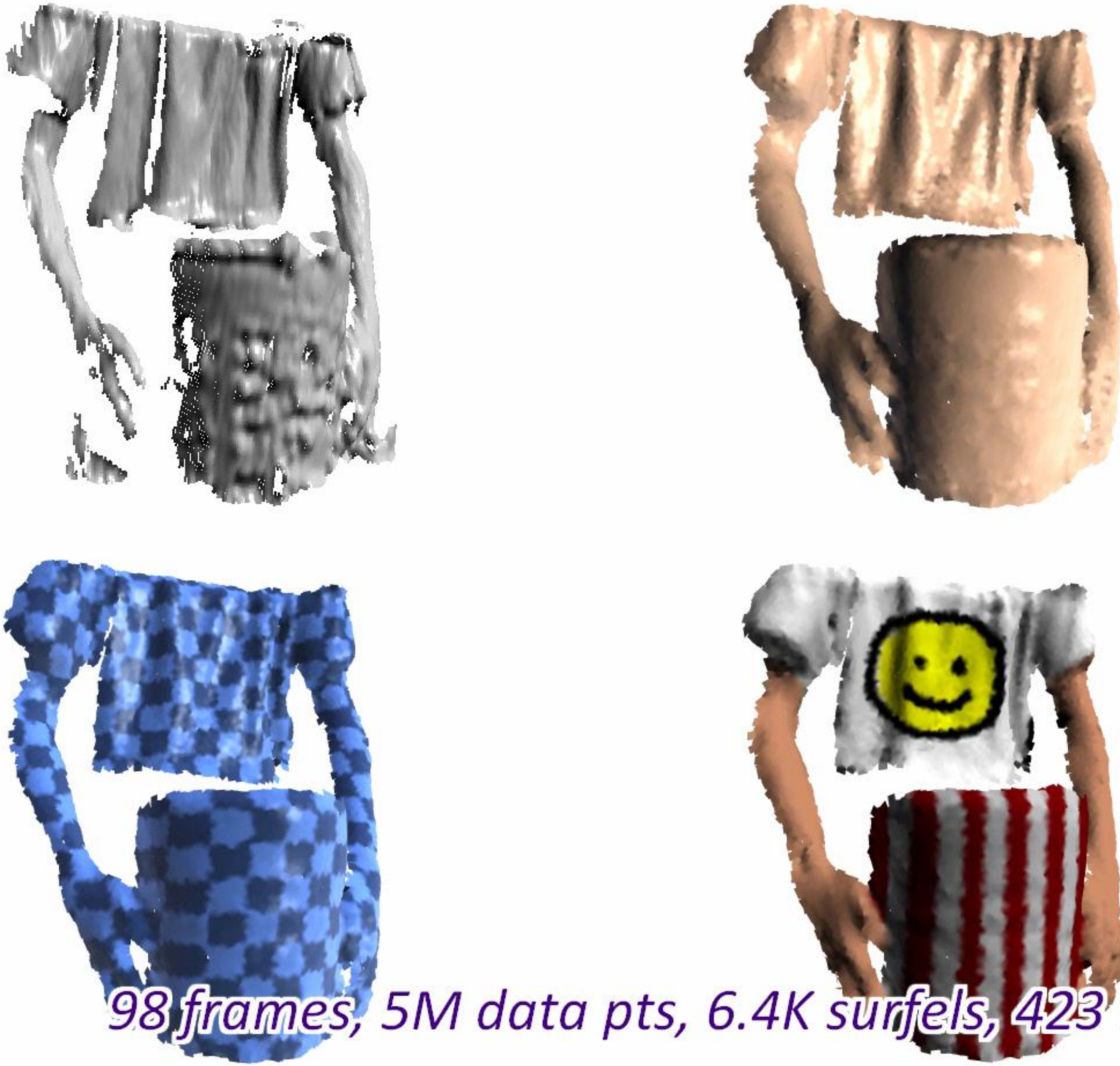
Hierarchical Alignment



Results

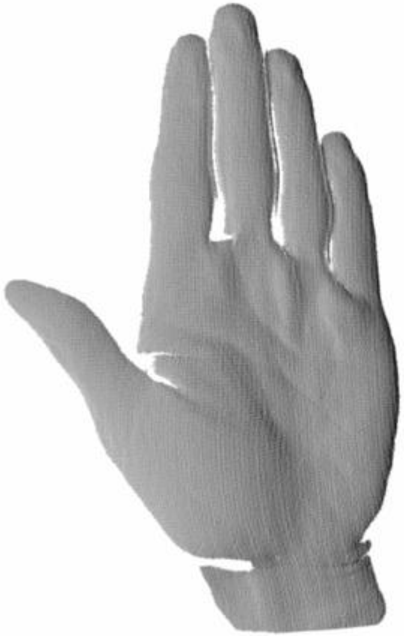


79 frames, 24M data pts, 21K surfels, 315 nodes



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