Shape Deformation

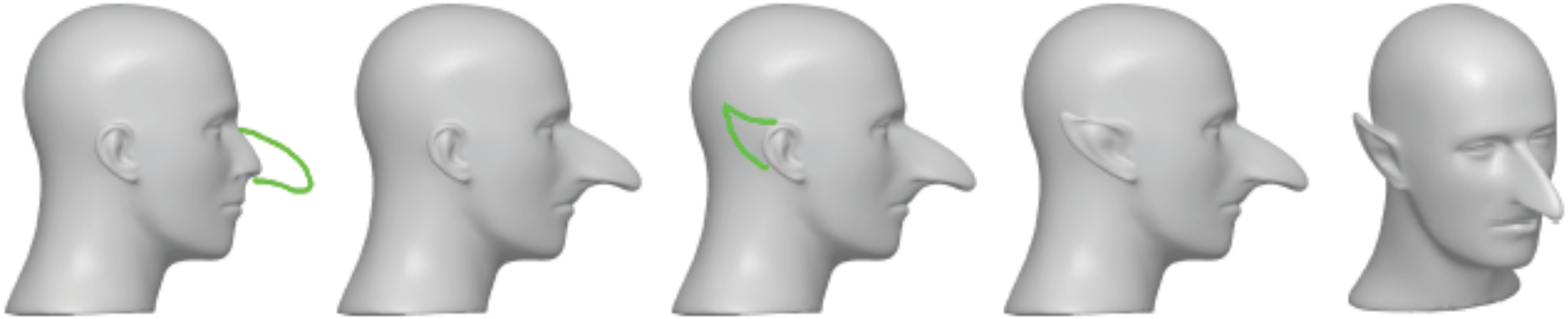
University of Victoria | Computer Science

Why Shape Deformation?

Animation

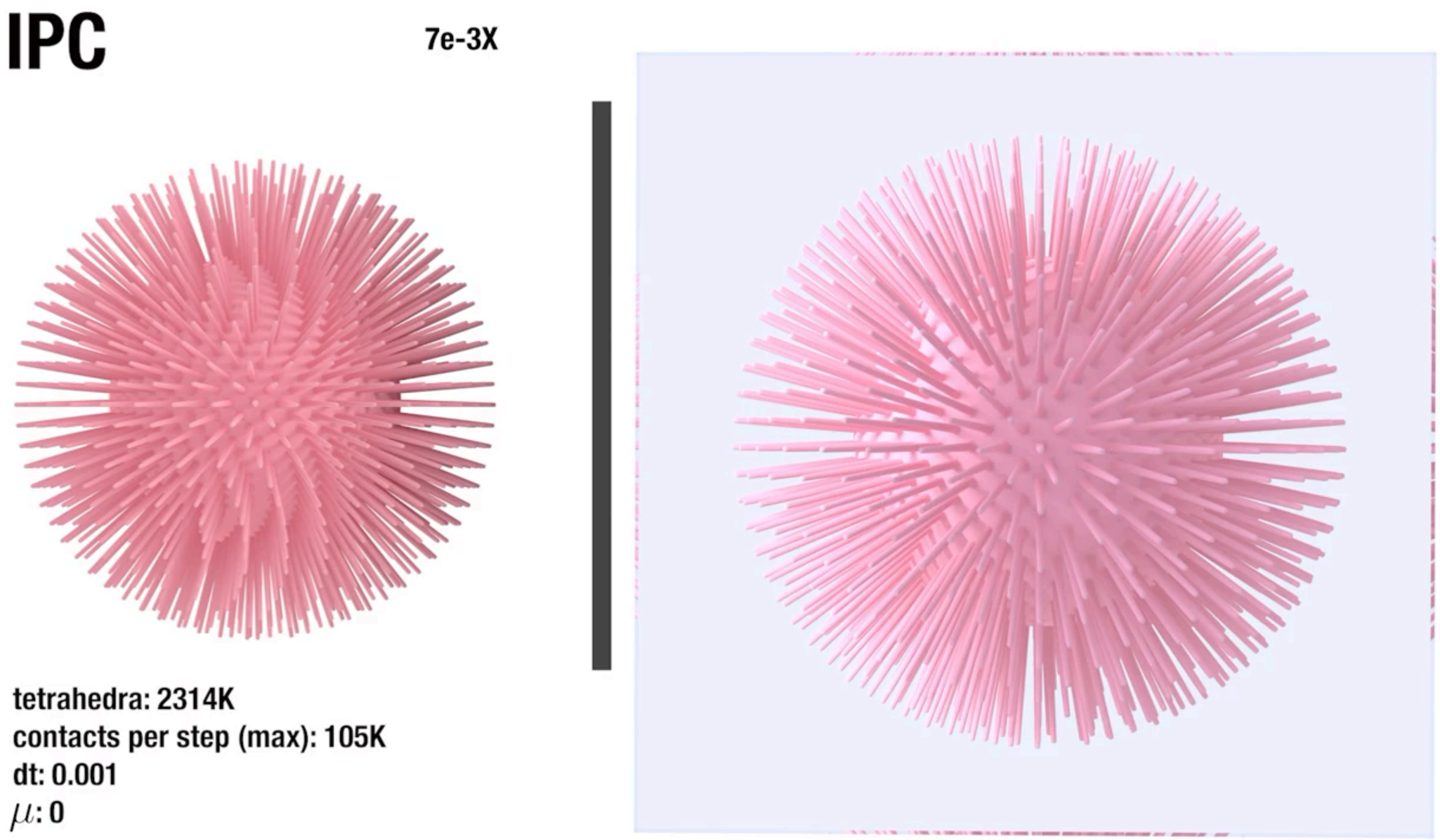
Why Shape Deformation?

Mesh Editing



Why Shape Deformation?

Simulation of Physical Phenomena



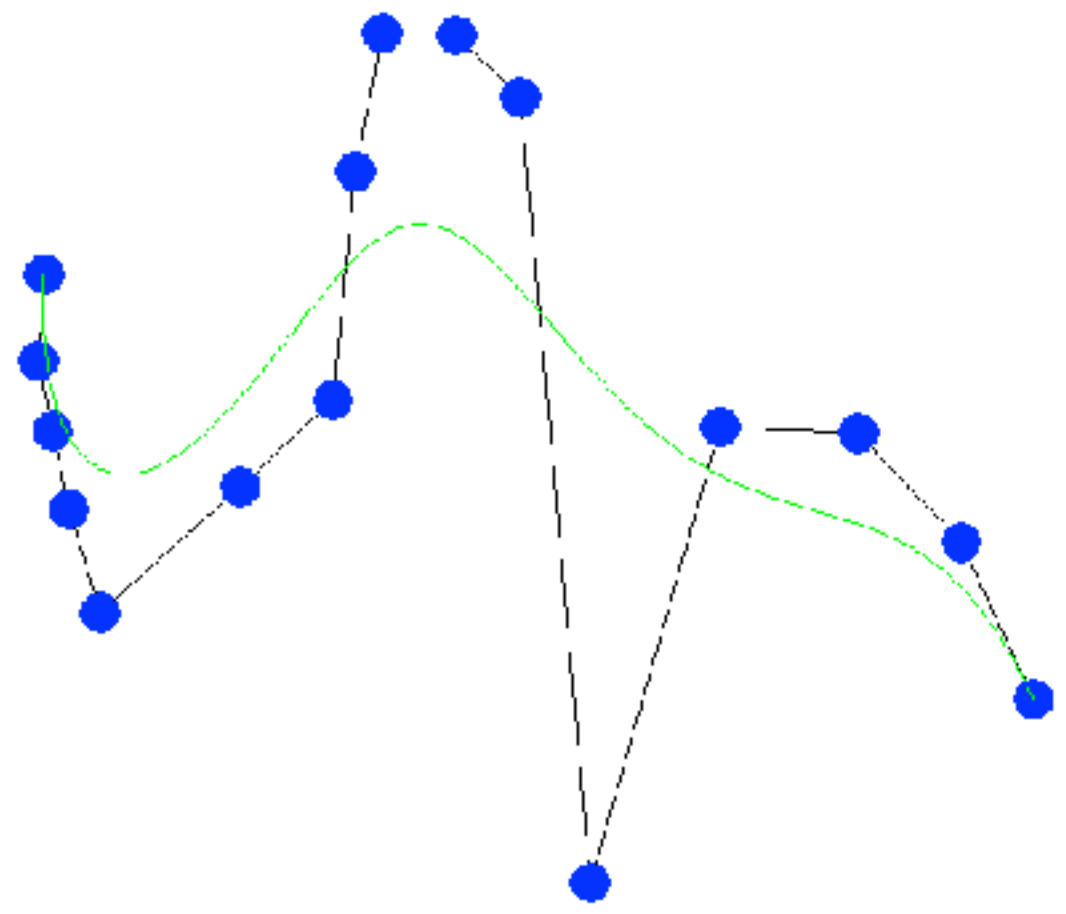
https://ipc-sim.github.io

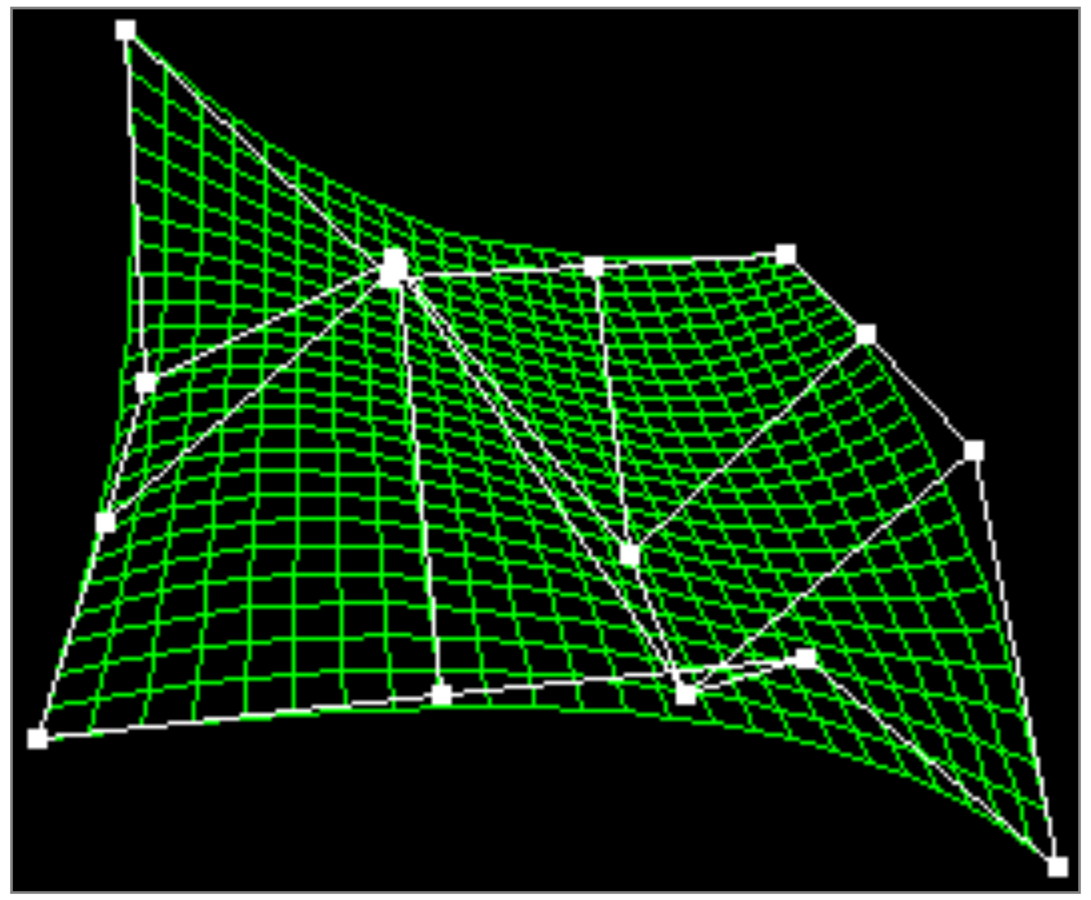
Parametric Curves and Surfaces

- Deformation by control point manipulation

- Built-in deformation mechanism

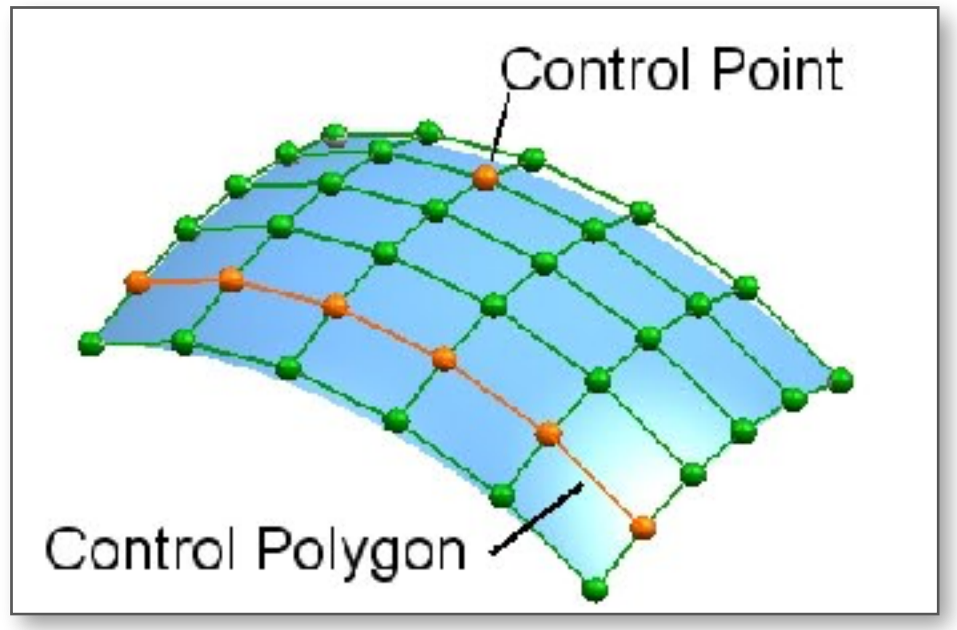
- Control structure is pre-set (you cannot pull on arbitrary points)



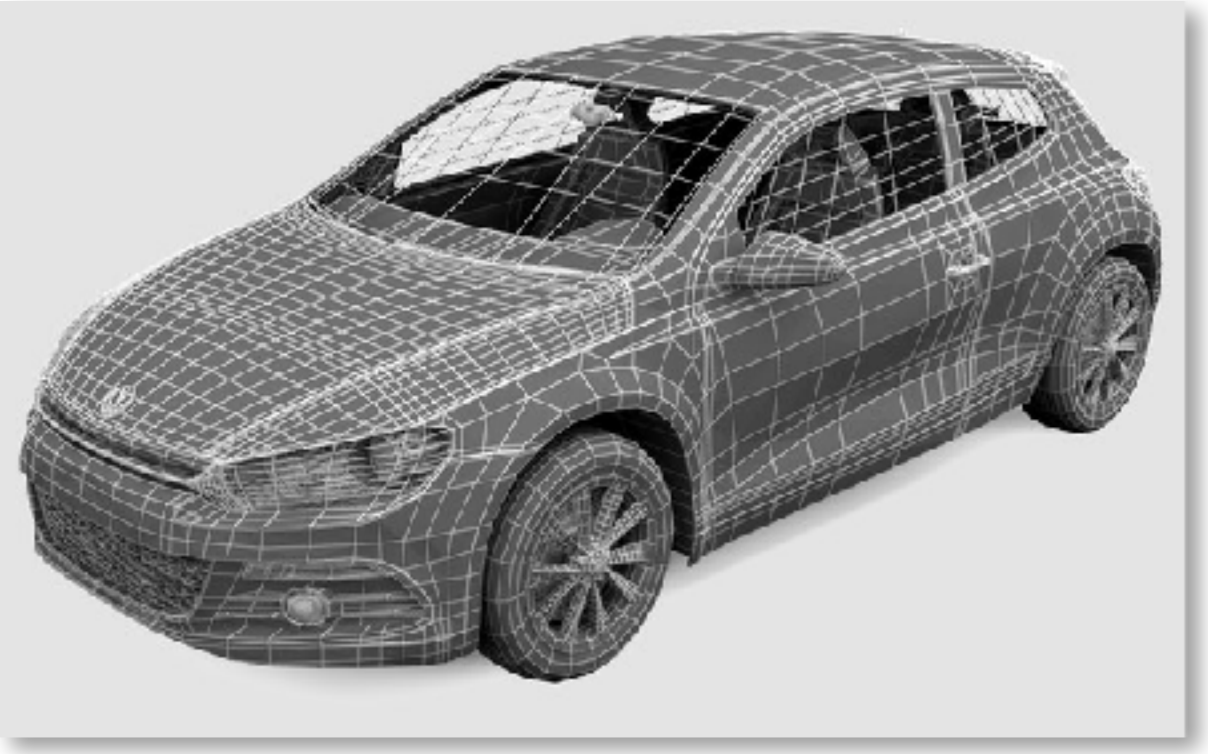


University of Victoria | Computer Science

Traditional CAD vs Unstructured Meshes



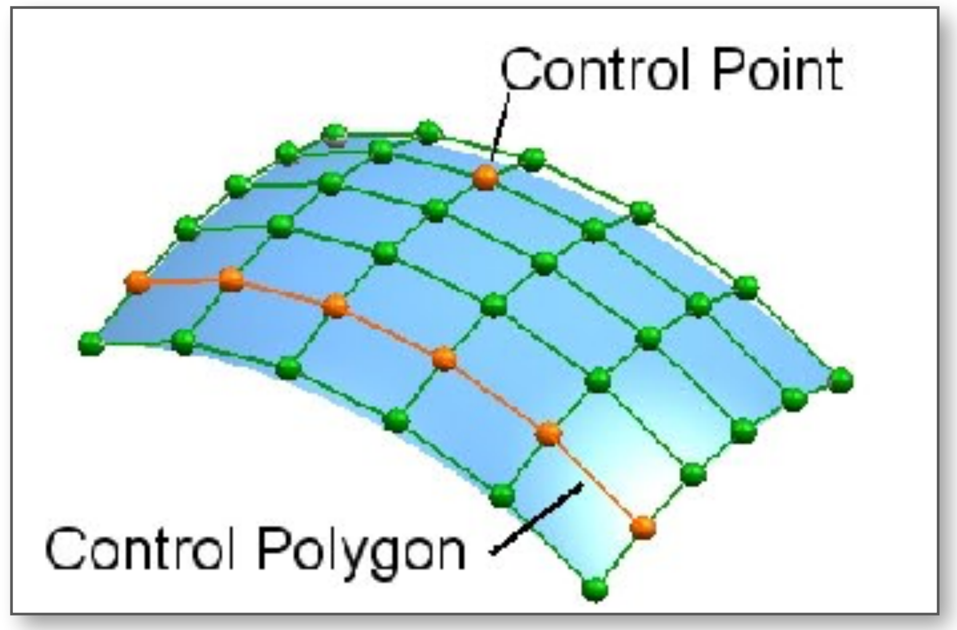
$$s(u,v) = \sum_{i,j} \boxed{\mathbf{p}_{i,j}} B_i(u) B_j(v)$$



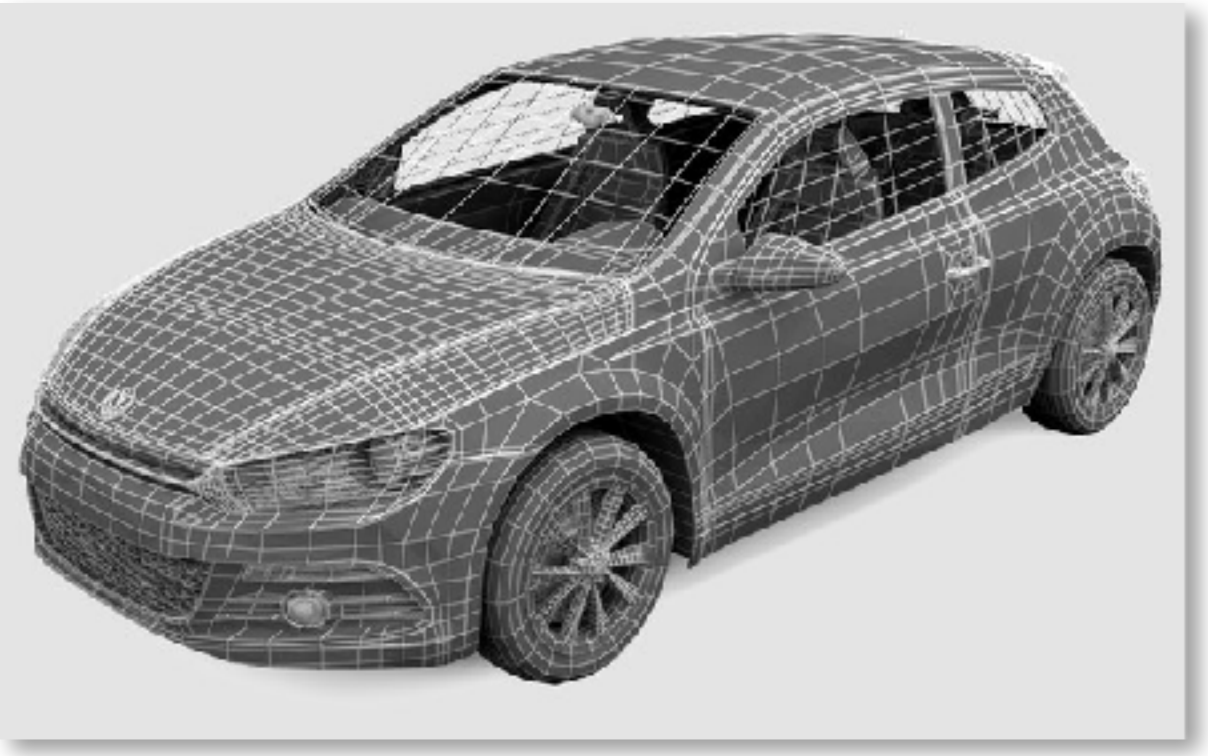
$$\min_{\mathbf{x}'} \int_{\mathcal{S}} \|\Delta_{\mathcal{S}} \mathbf{x}' - \delta_0\|^2$$
$$s.t. \; \boxed{\mathbf{x}'|_{\mathcal{C}}} = \mathbf{x}_{\text{fixed}}$$

images from Jacobson et al., SGP 2010

Traditional CAD vs Unstructured Meshes



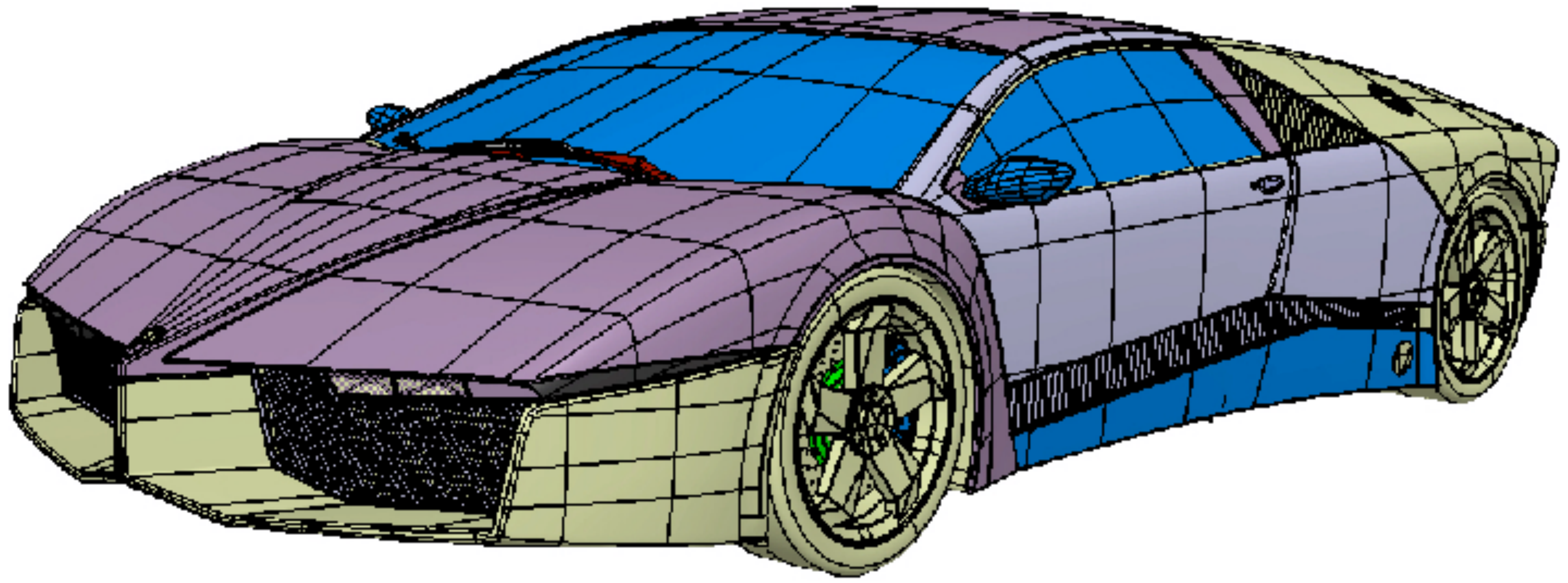
$$s(u,v) = \sum_{i,j} \boxed{\mathbf{p}_{i,j}} B_i(u) B_j(v)$$

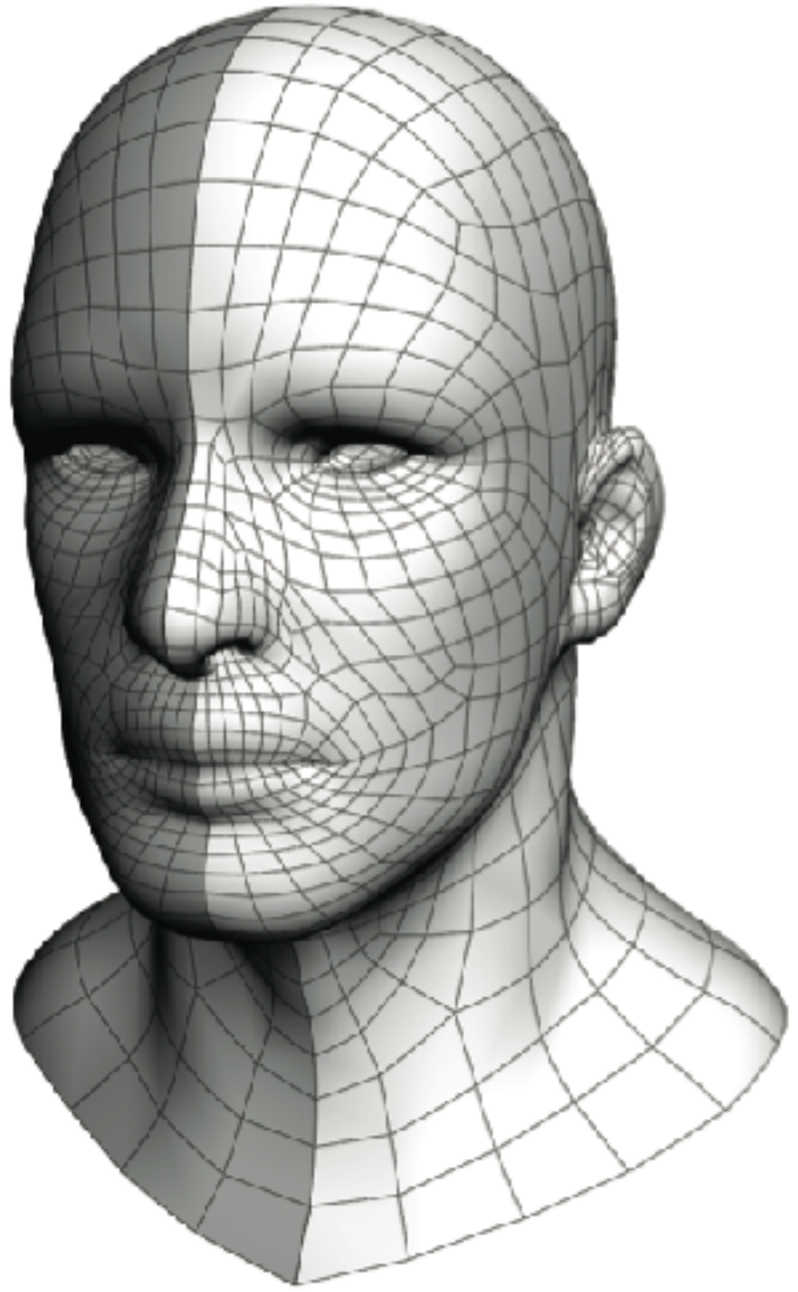


$$\min_{\mathbf{x}'} \int_{\mathcal{S}} \|\Delta_{\mathcal{S}} \mathbf{x}' - \delta_0\|^2$$
$$s.t. \ \boxed{\mathbf{x}'|_{\mathcal{C}}} = \mathbf{x}_{\text{fixed}}$$

Parametric Curves and Surfaces

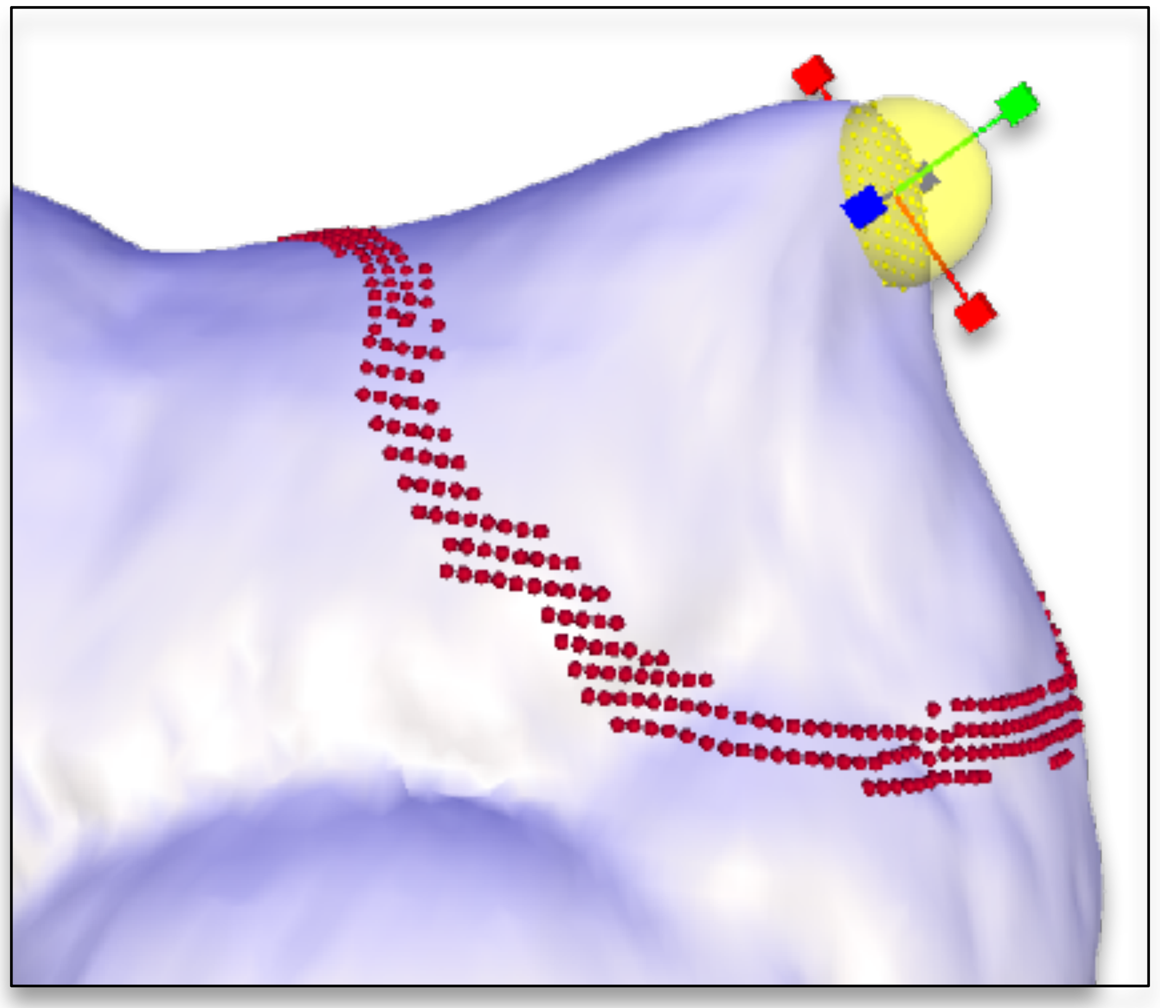
- Hard to change / adapt control structure to user needs
- Hard to experiment, need a precise idea of what will be modeled.





Mesh Deformation

- Naïve method: dragging single vertices
- Smarter:
  - Introduce a small set of deformation handles
    - Makes deformation/editing easier
    - Introduces a trade-off between degrees of freedom and simplicity of the deformation task
  - Create a small set of control parameters
    - Affine transformations



Common Paradigms

- Space deformations
  - Deforms some 2D/3D space using a *cage*
  - Deformation propagation to all points in the space
  - Independent of shape representation

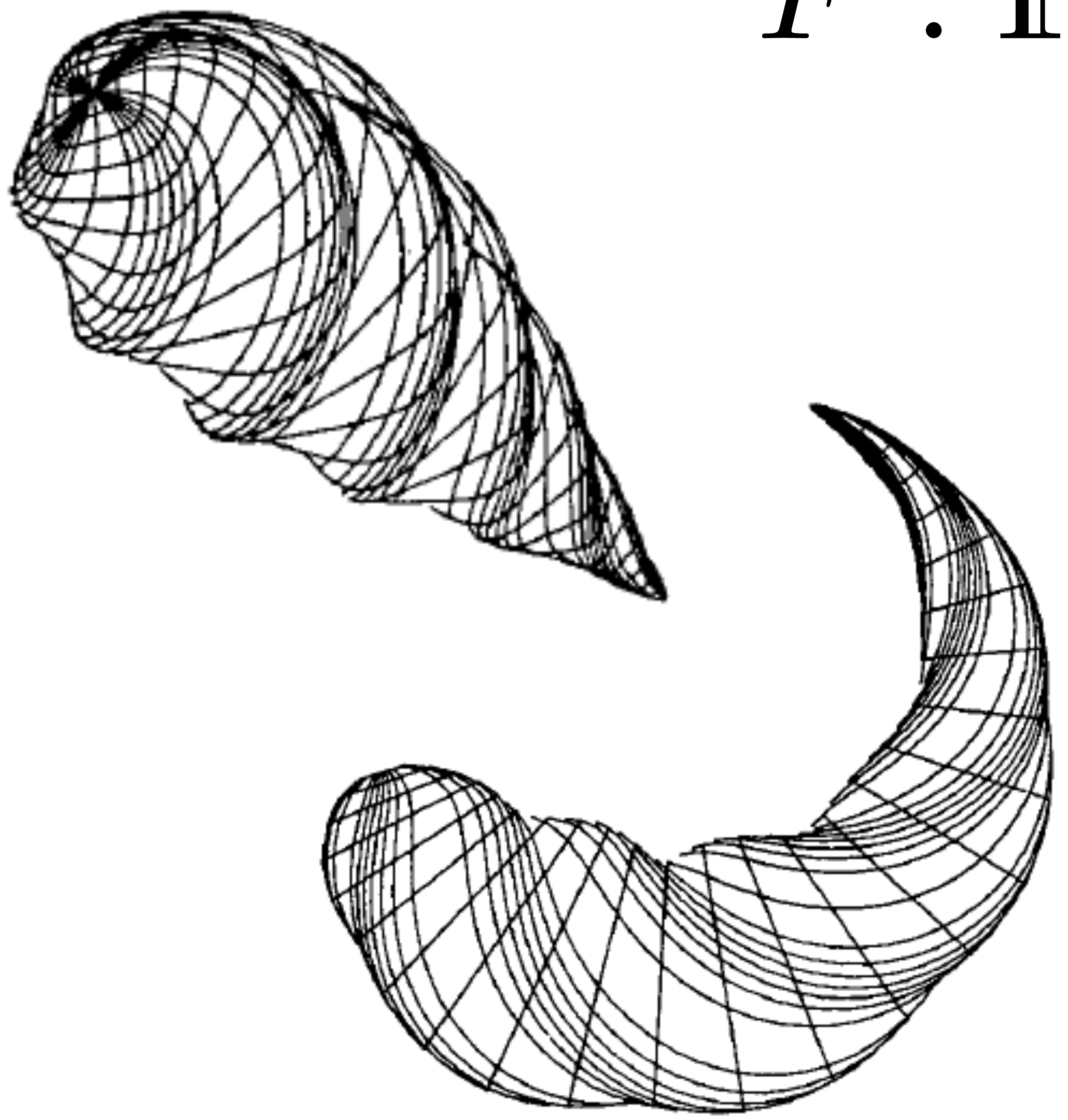
- Surface based deformations
  - Optimization on the surface
  - Physically motivated: variants of elastic energy minimization

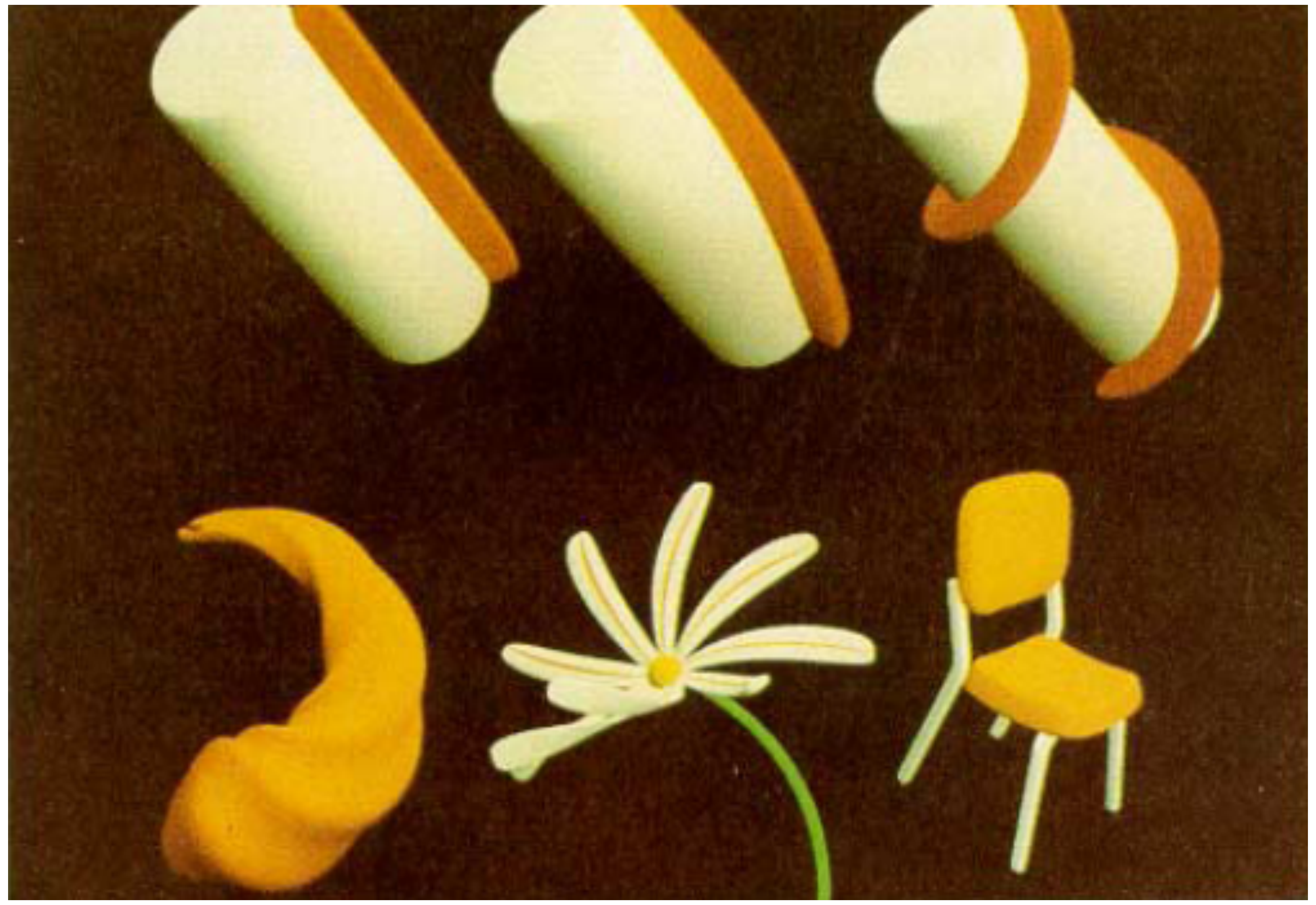
Space Deformations

Early seminal works in computer graphics

- Global and local deformation of solids [Barr 1984] http://dl.acm.org/citation.cfm?id=808573

$$F : \mathbb{R}^3 \to \mathbb{R}^3$$



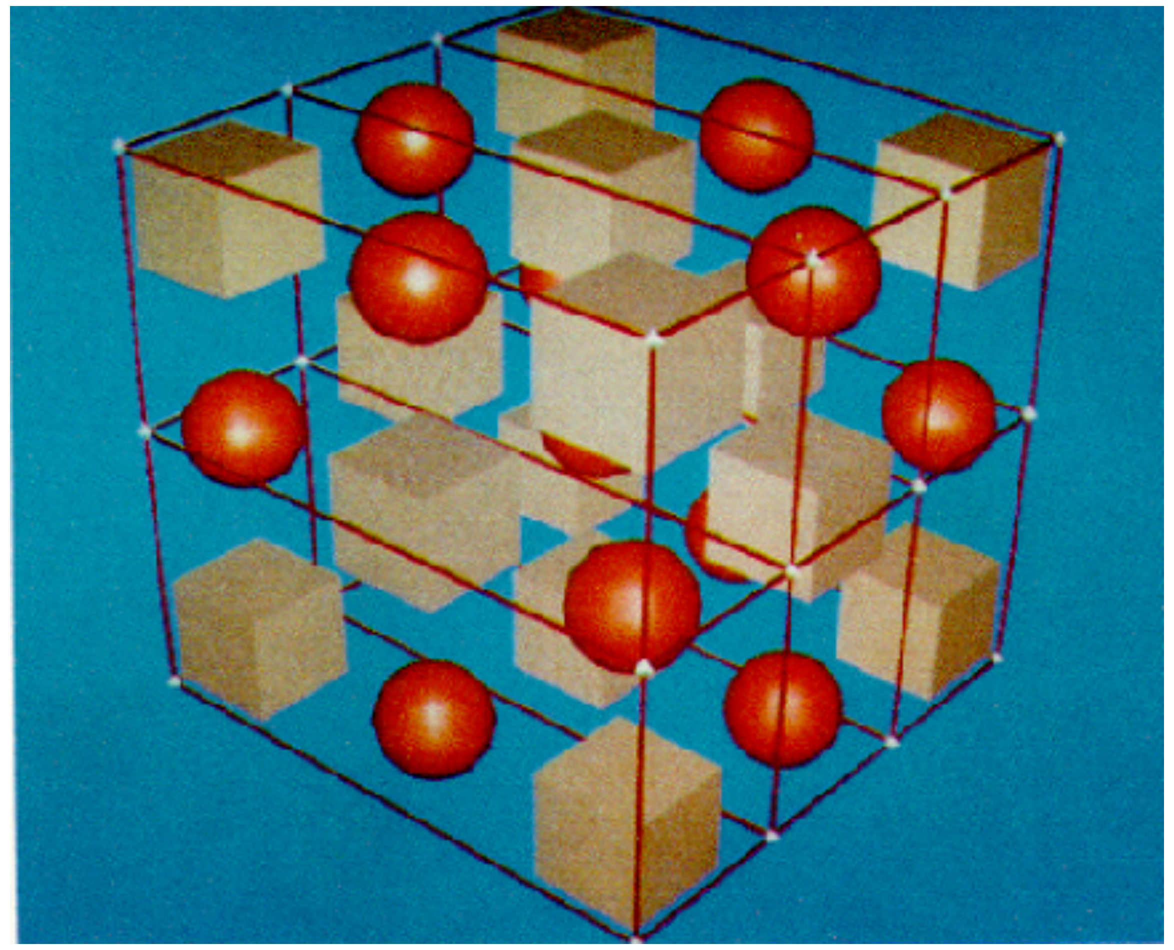


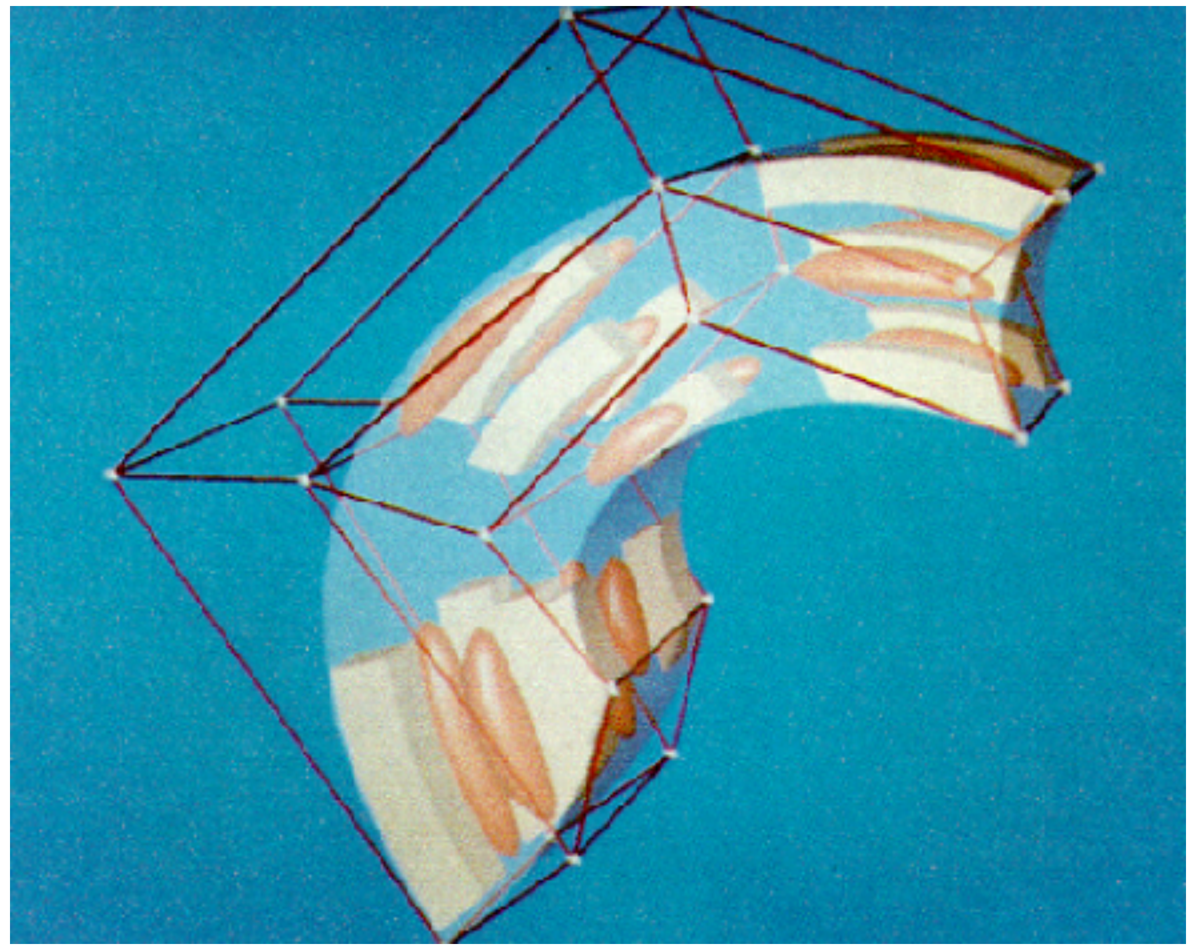
University of Victoria | Computer Science

Early seminal works in computer graphics

$$f : \mathbb{R}^3 \to \mathbb{R}^3$$

- Free form deformations
  [Sederberg and Parry 1986] http://dl.acm.org/citation.cfm?id=15903
  - Uses trivariate tensor product polynomial basis





Early seminal works in computer graphics

- Can be designed to be volume preserving

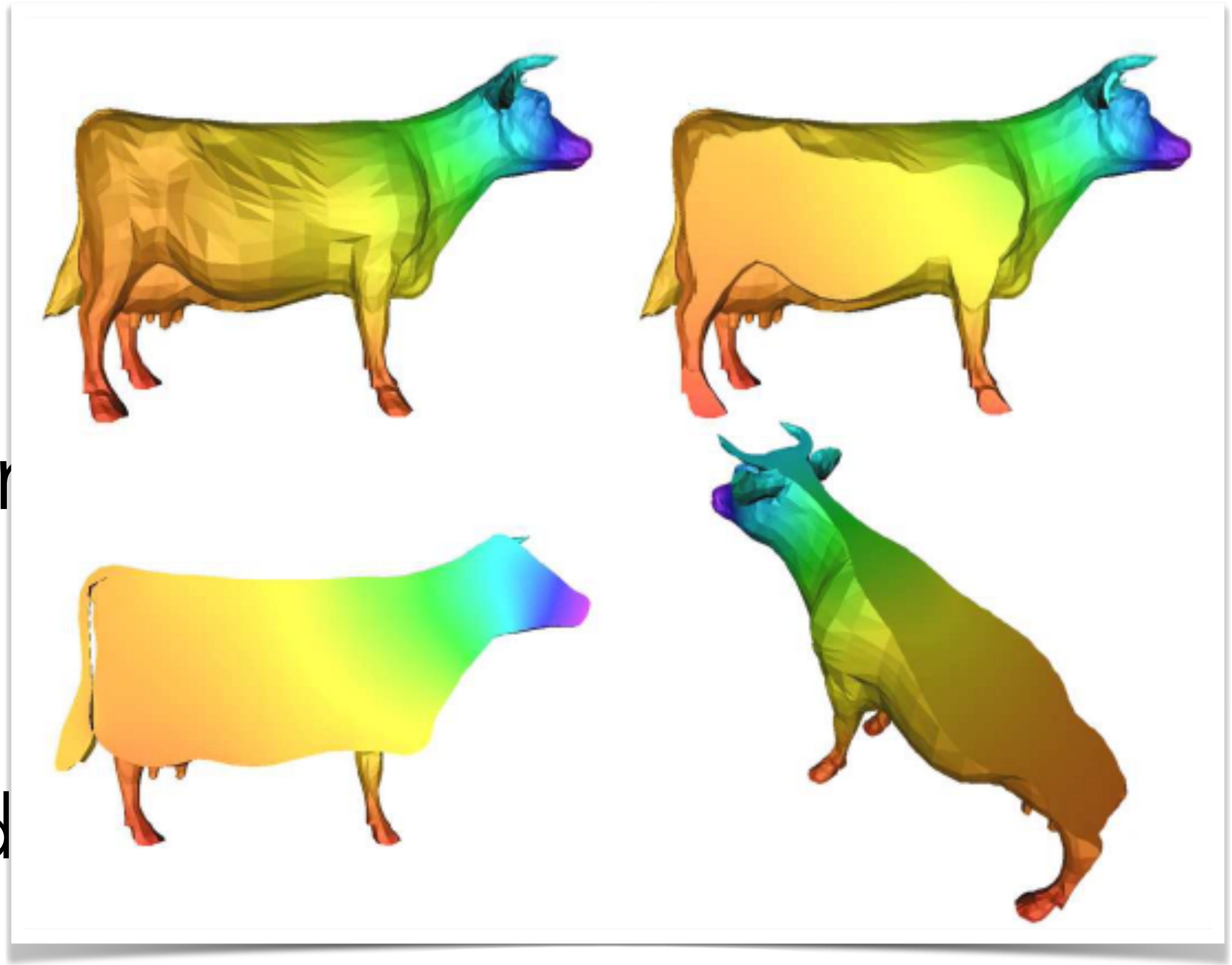
$$f : \mathbb{R}^3 \to \mathbb{R}^3$$

$$\mathbf{F}(x,y,z) = (F(x,y,z), G(x,y,z), H(x,y,z))$$

then the Jacobian is the determinant

$$Jac(\mathbf{F}) = \begin{vmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial z} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} & \frac{\partial G}{\partial z} \\ \frac{\partial H}{\partial x} & \frac{\partial H}{\partial y} & \frac{\partial H}{\partial z} \end{vmatrix}$$

Basic Idea



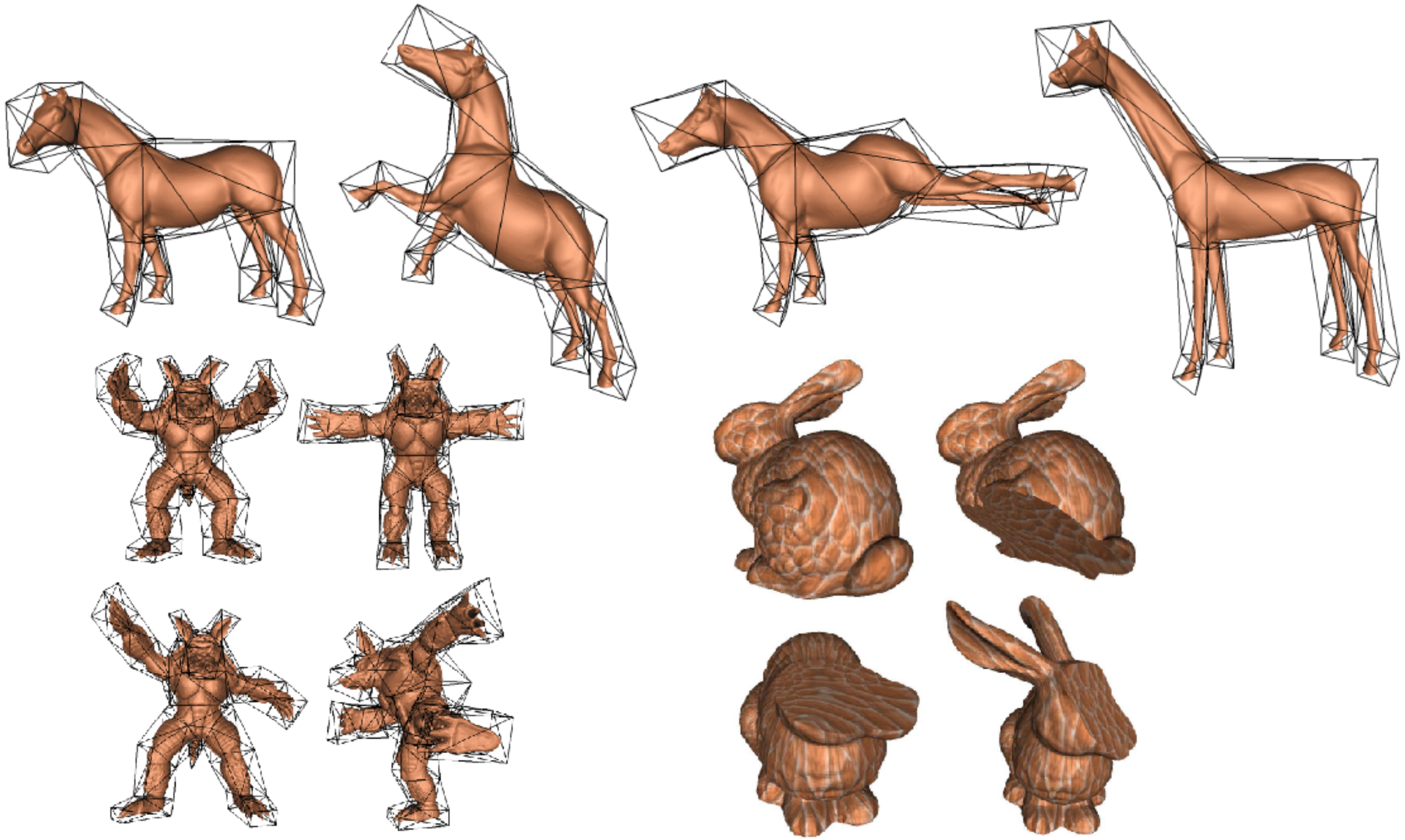
- Design a set of coordinates for all points i
w.r.t. the "cage" vertices
    - Each point **x** can be represented as a weighted

$$\mathbf{x} = \sum_{i=1}^{k} w_i(\mathbf{x})\, \mathbf{p}_i$$

    - When the cage changes, the coords stay the same, substitute the new cage geometry:

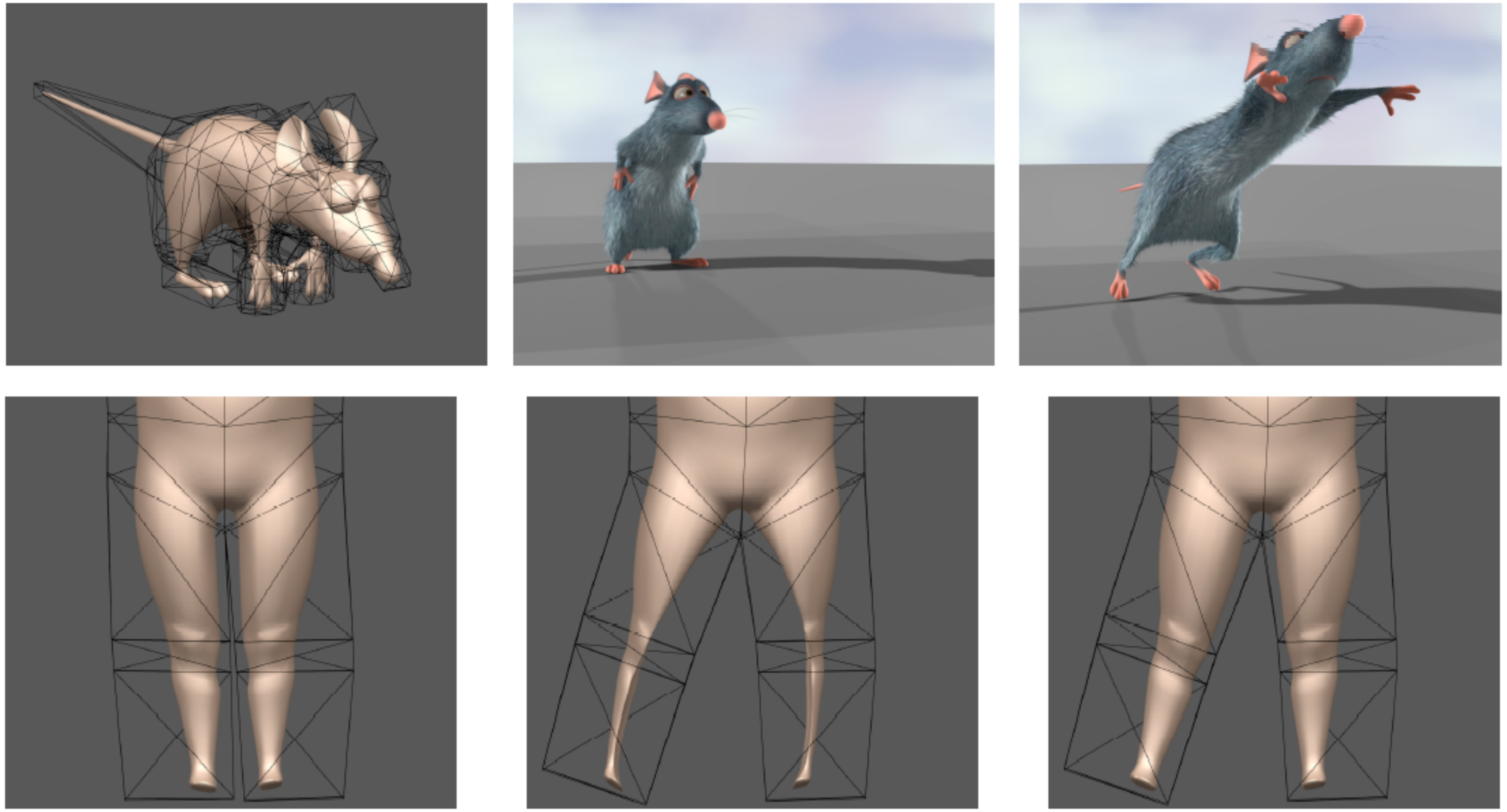
$$\mathbf{x}' = \sum_{i=1}^{k} w_i(\mathbf{x})\, \mathbf{p}'_i$$

Mean Value Coordinates



[Ju et al. 2005]
http://dl.acm.org/citation.cfm?id=1186822.1073229

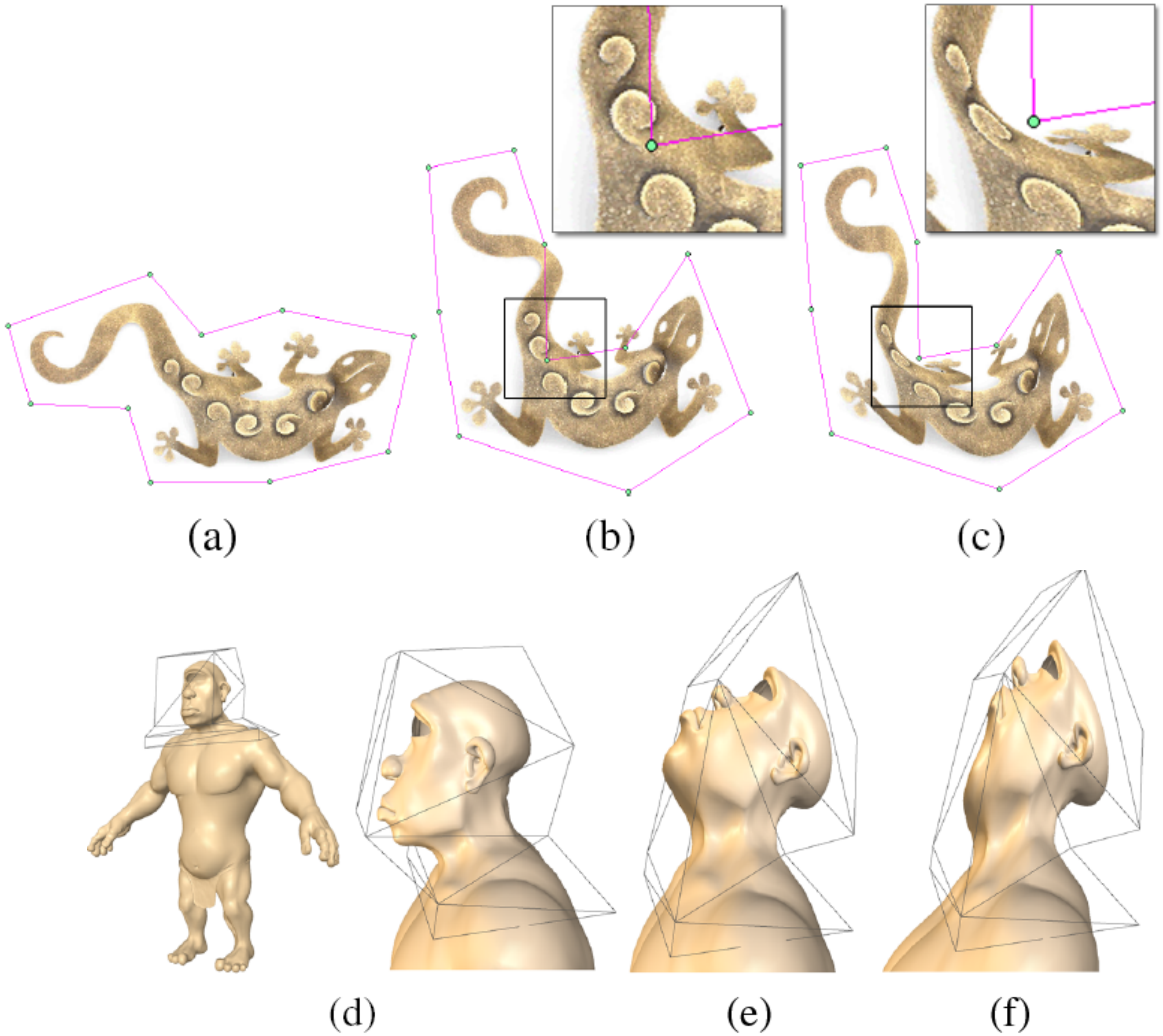
Harmonic Coordinates



[Joshi et al. 2007]

http://dl.acm.org/citation.cfm?id=1276466

Green Coordinates



http://dl.acm.org/citation.cfm?id=1360677

Space Deformations

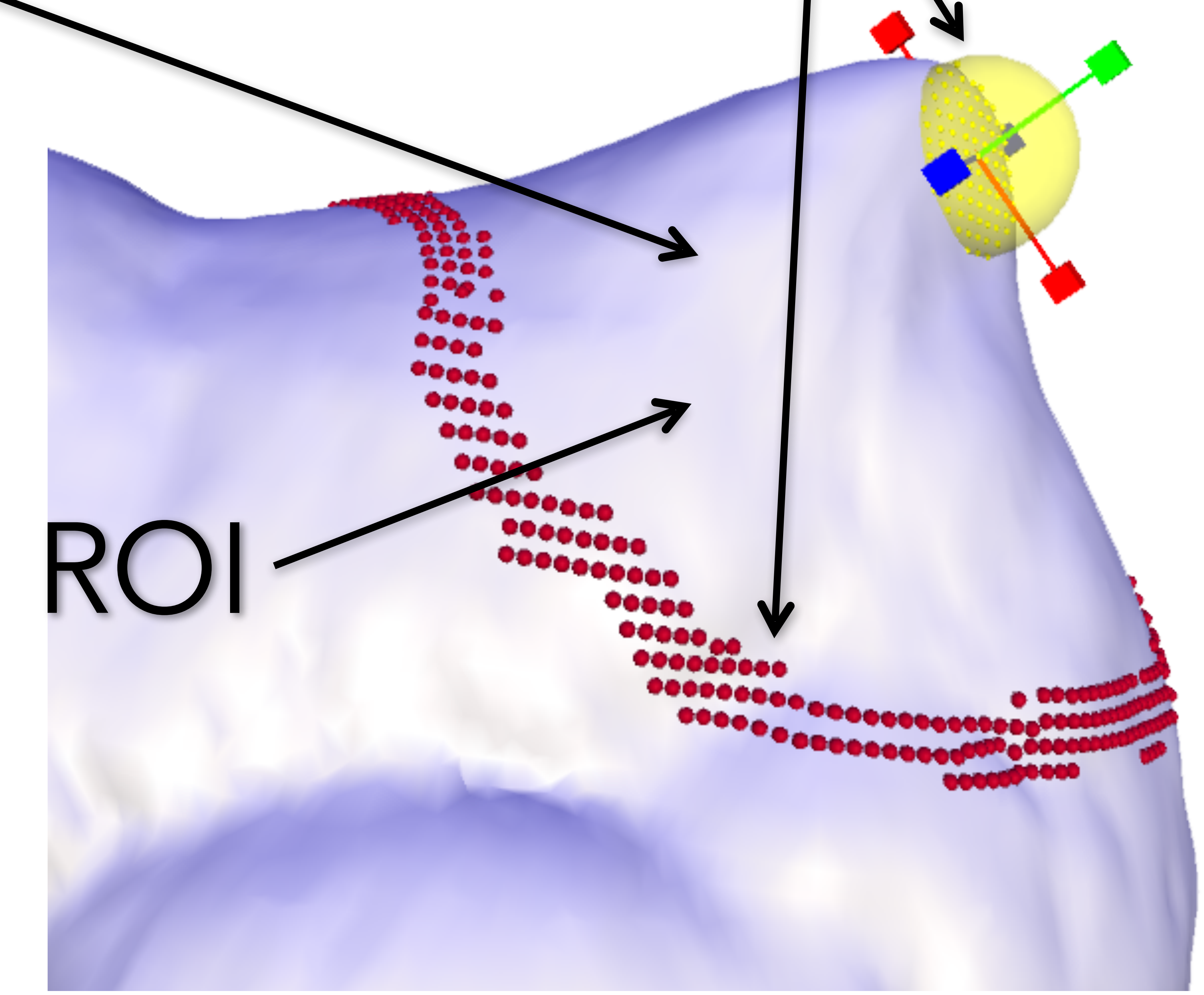
- Complexity depends mainly on the cage; linear in the number of mesh elements
  - Parallel execution with GPU accelerators
- Can handle disconnected components or even just point sets
- Harder to control the surface properties since the whole space is being warped

Surface-Based Deformations

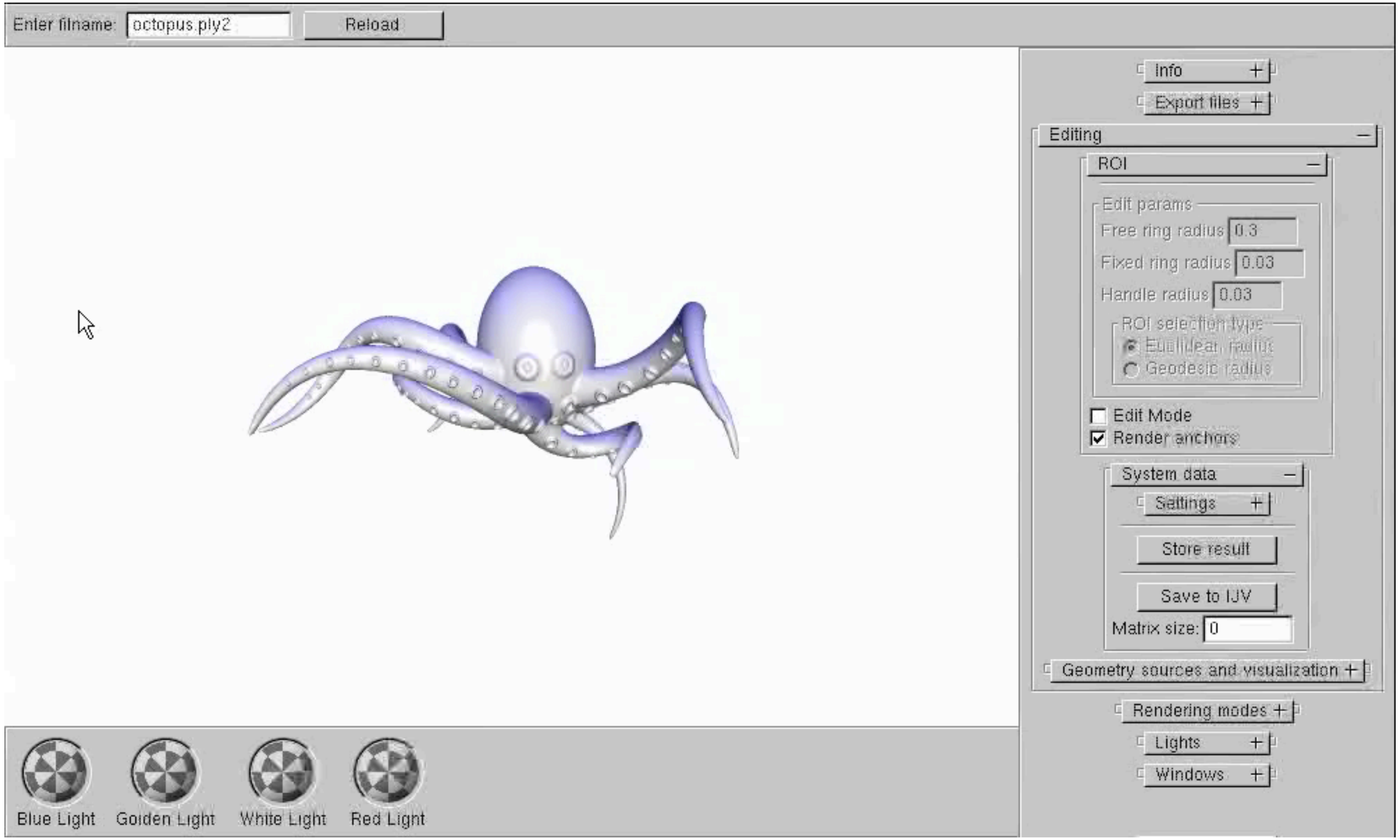
Surface-based Deformation: ROI-Handle Editing Metaphor

$$\mathbf{x}_{\text{def}} = \underset{\mathbf{x}'}{\operatorname{argmin}} \, E(\mathbf{x}') \quad s.t. \quad \mathbf{x}'_i = \mathbf{c}_i$$

- ROI is bounded by a belt (static anchors)
- Manipulation through handle(s) – affine transformations



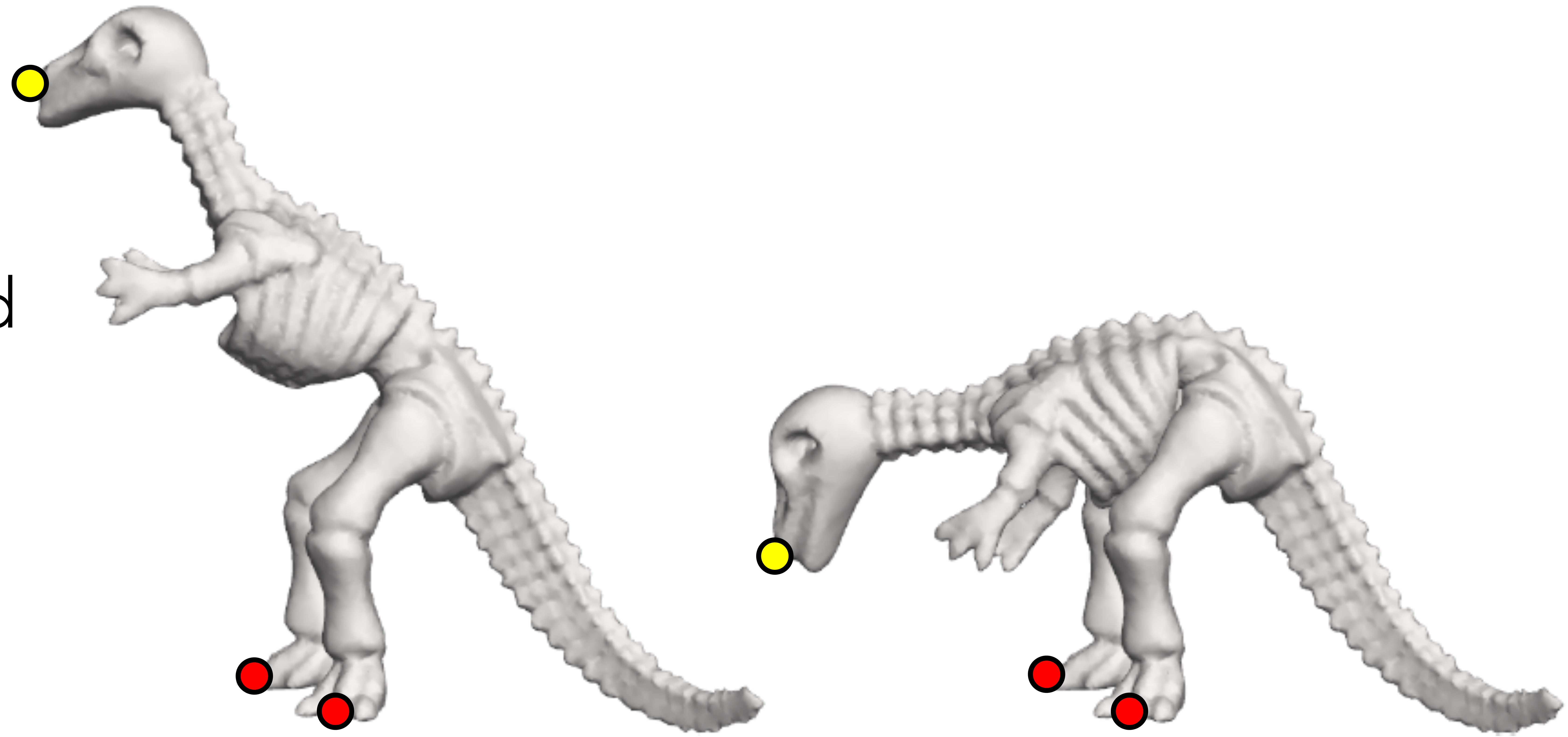
Surface-based Deformation: ROI-Handle Editing Metaphor



University of Victoria | Computer Science

How to Define $E(\mathbf{x}')$?

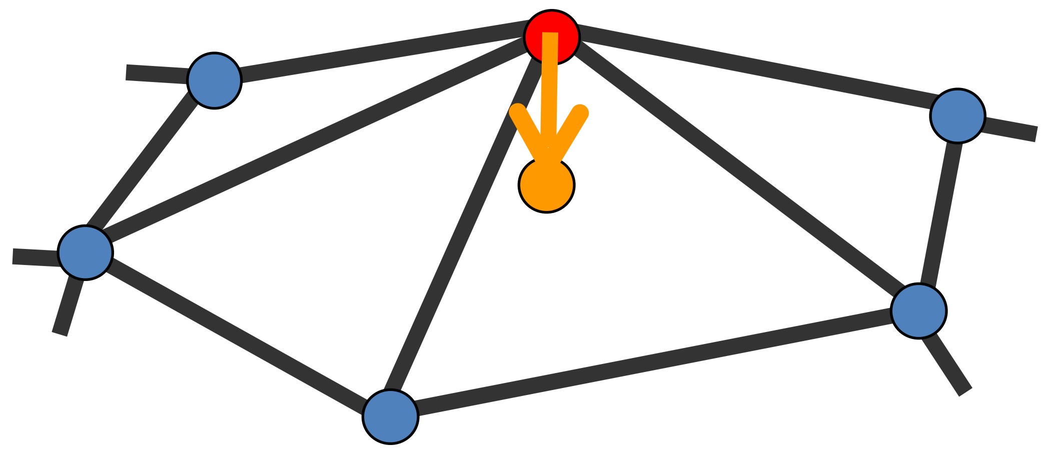
- Intuitive deformations:
  - Smooth deformation on the global scale
  - Preserve local details (curvatures)
- Invariants: $E(\mathbf{x}')$ should be zero if $\mathbf{x}'$ is a rigid transformation of $\mathbf{x}$ original geometry



University of Victoria | Computer Science

Recap: Differential Coordinates

- Detail = *smooth*(surface) – surface
- Smoothing = averaging



$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{x}_j - \mathbf{x}_i) \approx -2 H_i \mathbf{n}_i$$

Recap: Differential Coordinates

- Represent ***local detail*** at each surface point
  - More descriptive of the shape than just *xyz*
- Linear transition from *xyz* to **δ**
- Useful for operations on surfaces where surface details are important

University of Victoria | Computer Science

Simple Laplacian Editing

- Preserve mean curvature normal [≈differential coordinates] at every point in the ROI [≈ every vertex of the ROI]

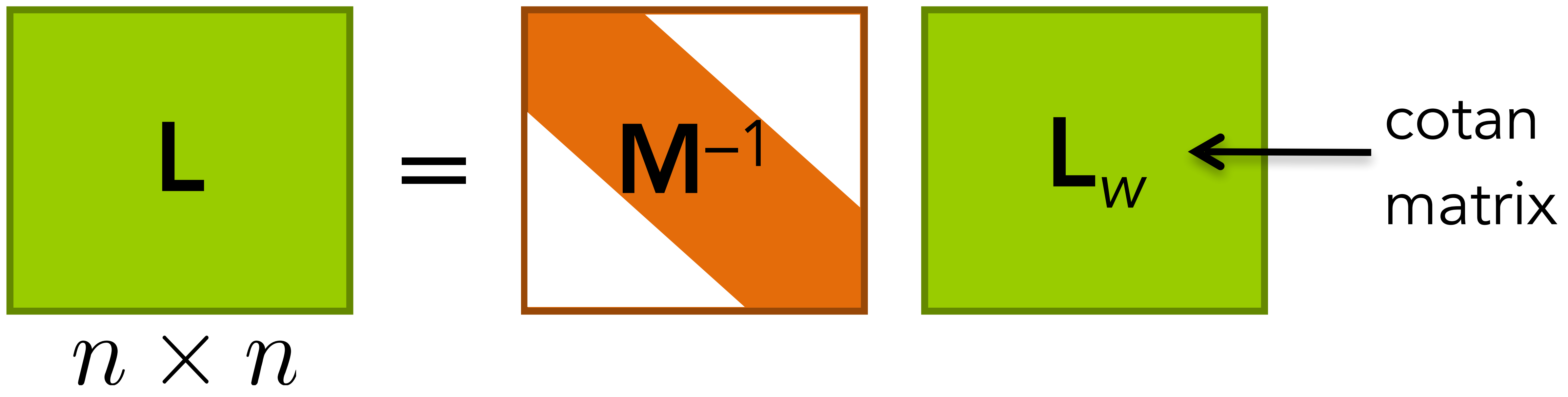
continuous: $$E(\mathcal{S}') = \int_{\mathcal{S}'} \|\Delta\mathbf{x}' - \delta\|^2 d\mathbf{x}'$$

discrete: $$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}_i') - \delta_i\|^2$$

Simplifying the Laplacian Energy

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}_i') - \delta_i\|^2 = \sum_{i=1}^{n} A_i \left(\Delta(\mathbf{x}_i')^T \Delta(\mathbf{x}_i') - 2\Delta(\mathbf{x}_i')^T \delta_i + \delta_i^T \delta_i\right) =$$

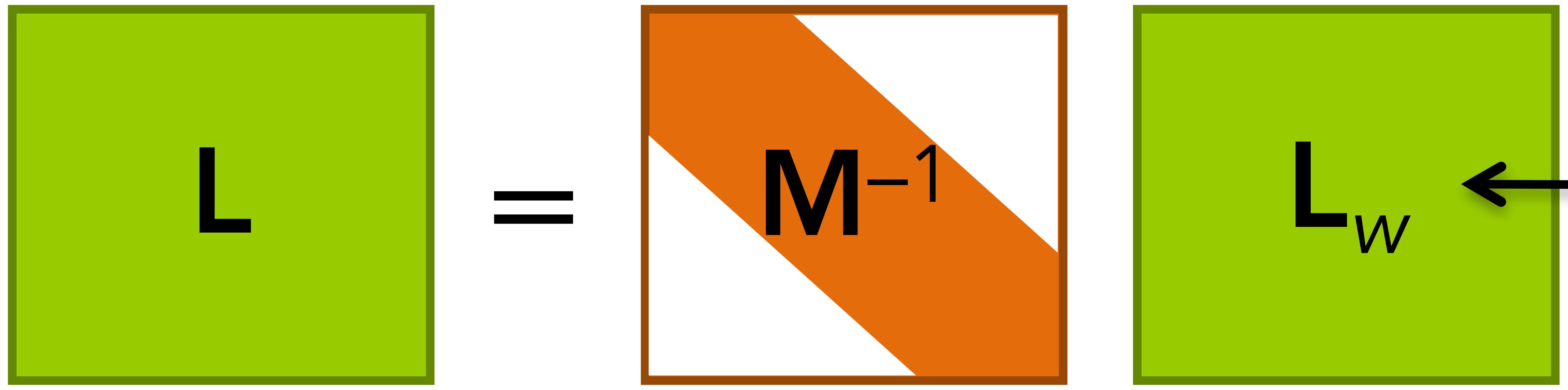
$$= \mathbf{x}'^T \underline{L^T M L}\mathbf{x}' - 2\mathbf{x}'^T \underline{L^T M}\delta + \text{const}$$



Simplifying the Laplacian Energy

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \delta_i\|^2 = \sum_{i=1}^{n} A_i \left(\Delta(\mathbf{x}'_i)^T \Delta(\mathbf{x}'_i) - 2\Delta(\mathbf{x}'_i)^T \delta_i + \delta_i^T \delta_i\right) =$$

$$= \mathbf{x}'^T \underline{L^T M} L\mathbf{x}' - 2\mathbf{x}'^T \underline{L^T M} \delta + \text{const}$$



$n \times n$

cotan matrix

$$L^T M L = (M^{-1} L_w)^T M (M^{-1} L_w) = L_w M^{-1} M M^{-1} L_w =$$

$$= L_w M^{-1} L_w$$

Symmetric sparse matrix!

Minimizing the Laplacian Energy

- To find the minimum, gradient = 0 and substitute the modeling constraints

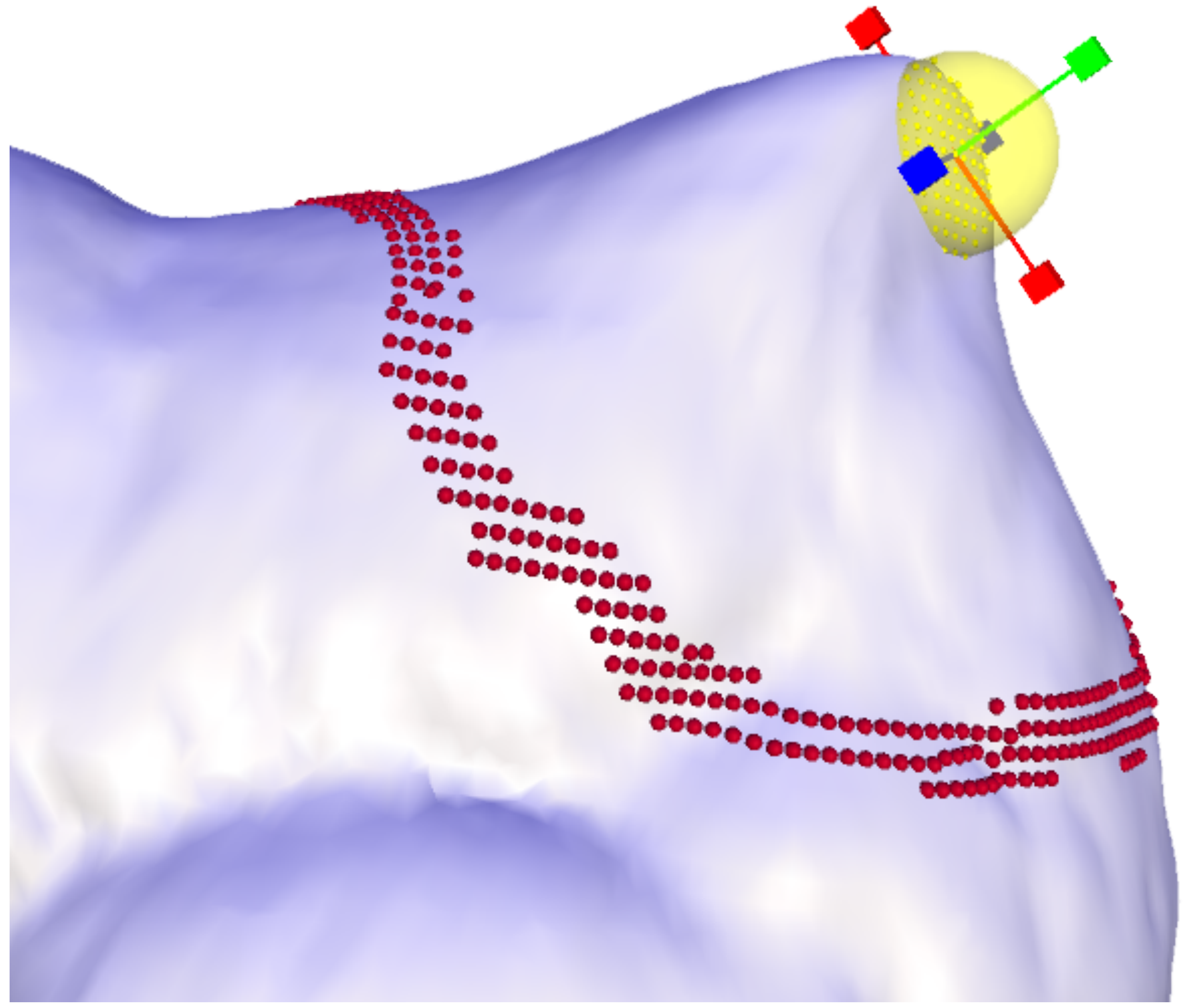
$$E(\mathbf{x}') = \mathbf{x}'^T L_w M^{-1} L_w \mathbf{x}' - 2\mathbf{x}'^T L_w \delta + \text{const}$$

$$\frac{\partial}{\partial \mathbf{x}'} E(\mathbf{x}') = 2 L_w M^{-1} L_w \mathbf{x}' - 2 L_w \delta$$

$$\mathbf{x}'_i = \mathbf{c}_i, \ i \in \mathcal{C}$$

Minimizing the Laplacian Energy

$$A\mathbf{x}' = \mathbf{b}$$



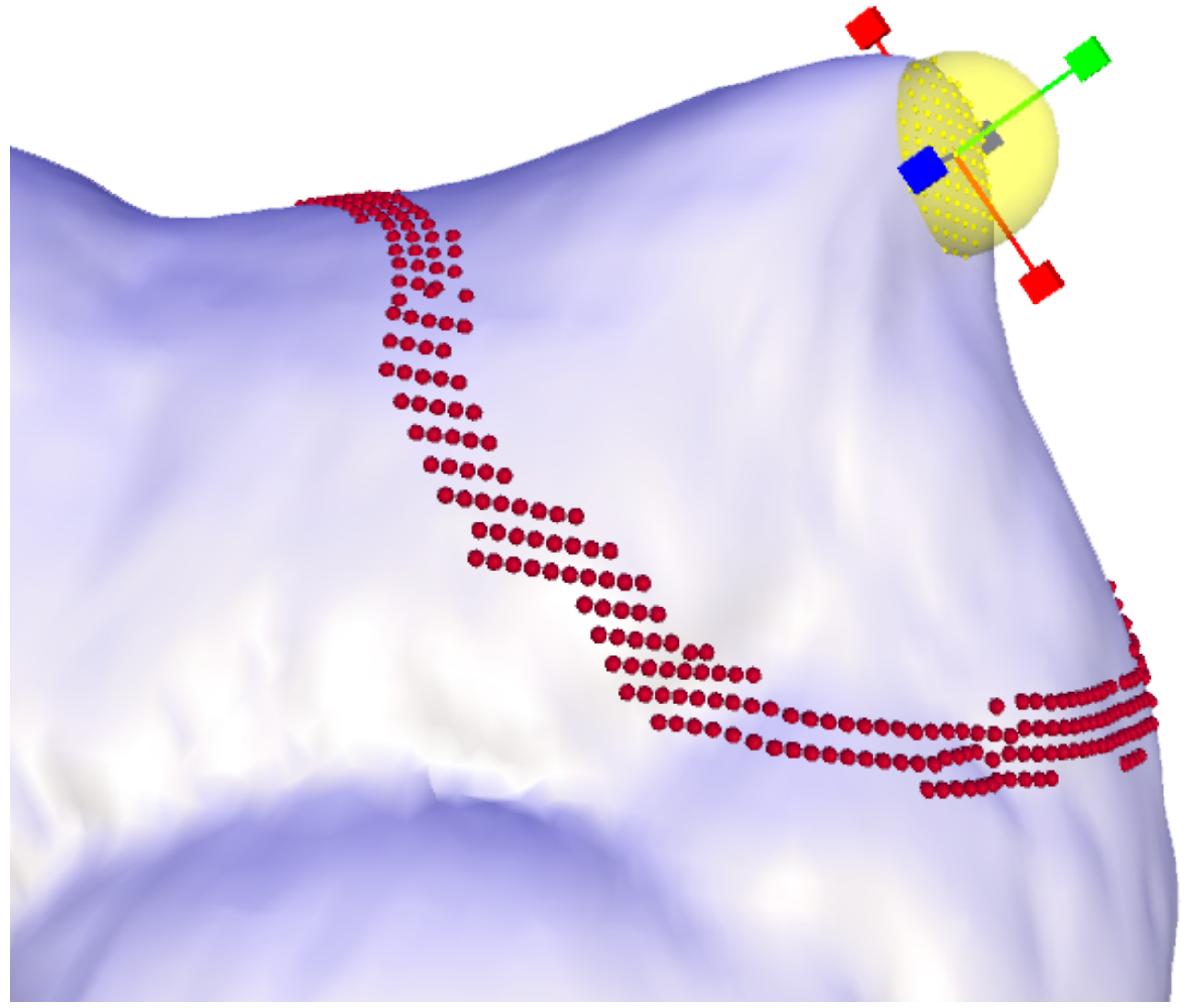
Matrix depends on the initial mesh and the indices of the constraints only. Matrix is fixed!

Right-hand side contains the coordinates of the constraints (handles)

University of Victoria | Computer Science

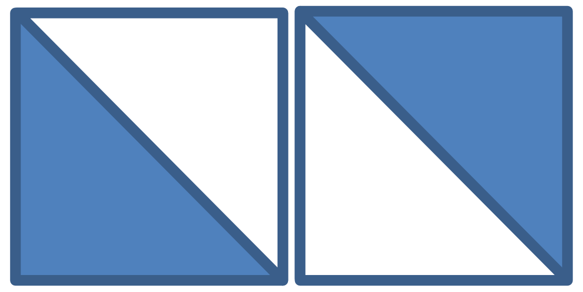
Minimizing the Laplacian Energy

$$A\mathbf{x}' = \mathbf{b}$$



Sparse Cholesky decomposition:

$$A = L_{\text{chol}} L_{\text{chol}}^T$$



At run-time: just back-substitution!

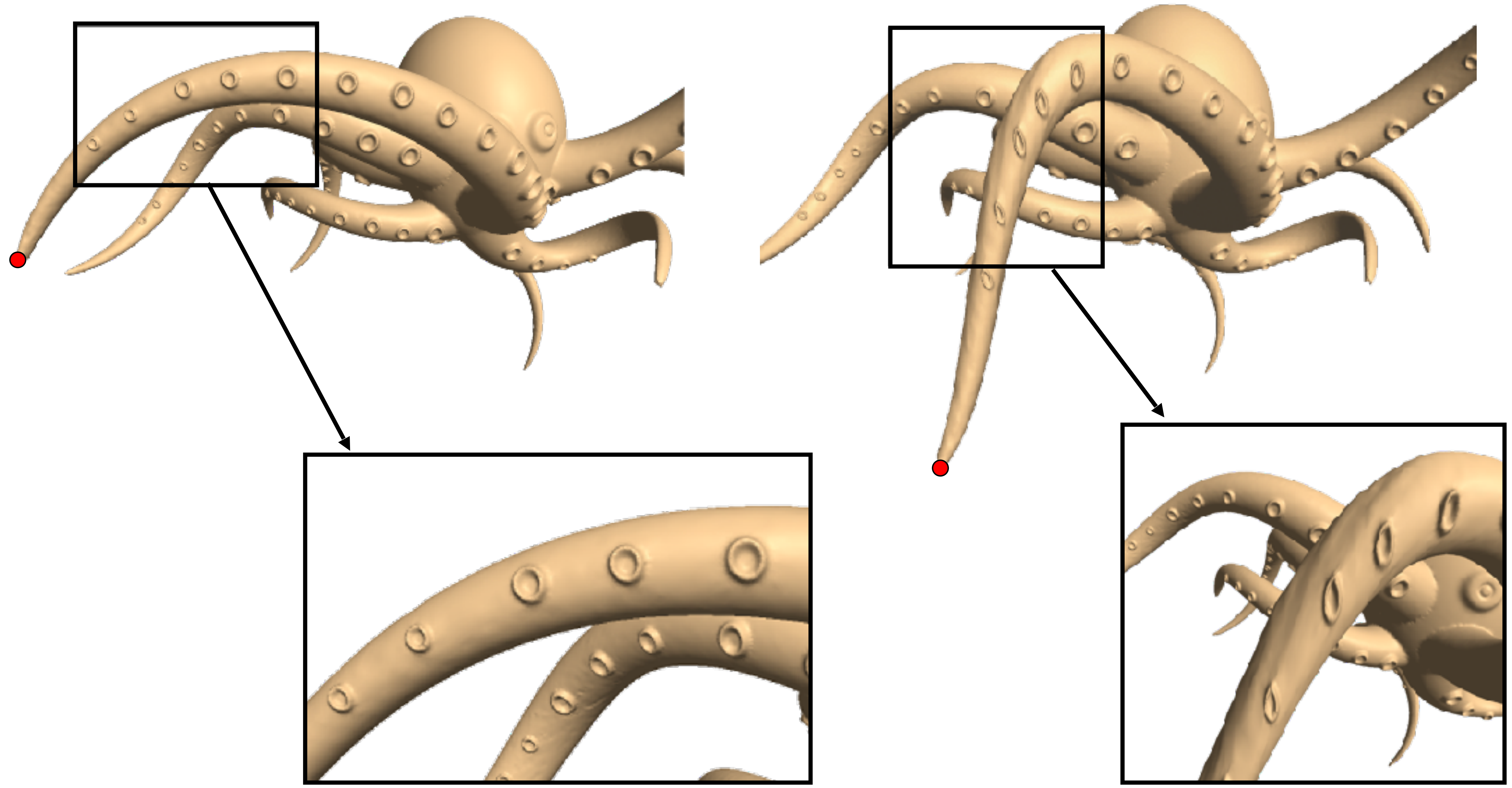
$$L_{\text{chol}} \mathbf{y} = \mathbf{b}$$

$$L_{\text{chol}}^T \mathbf{x}' = \mathbf{y}$$

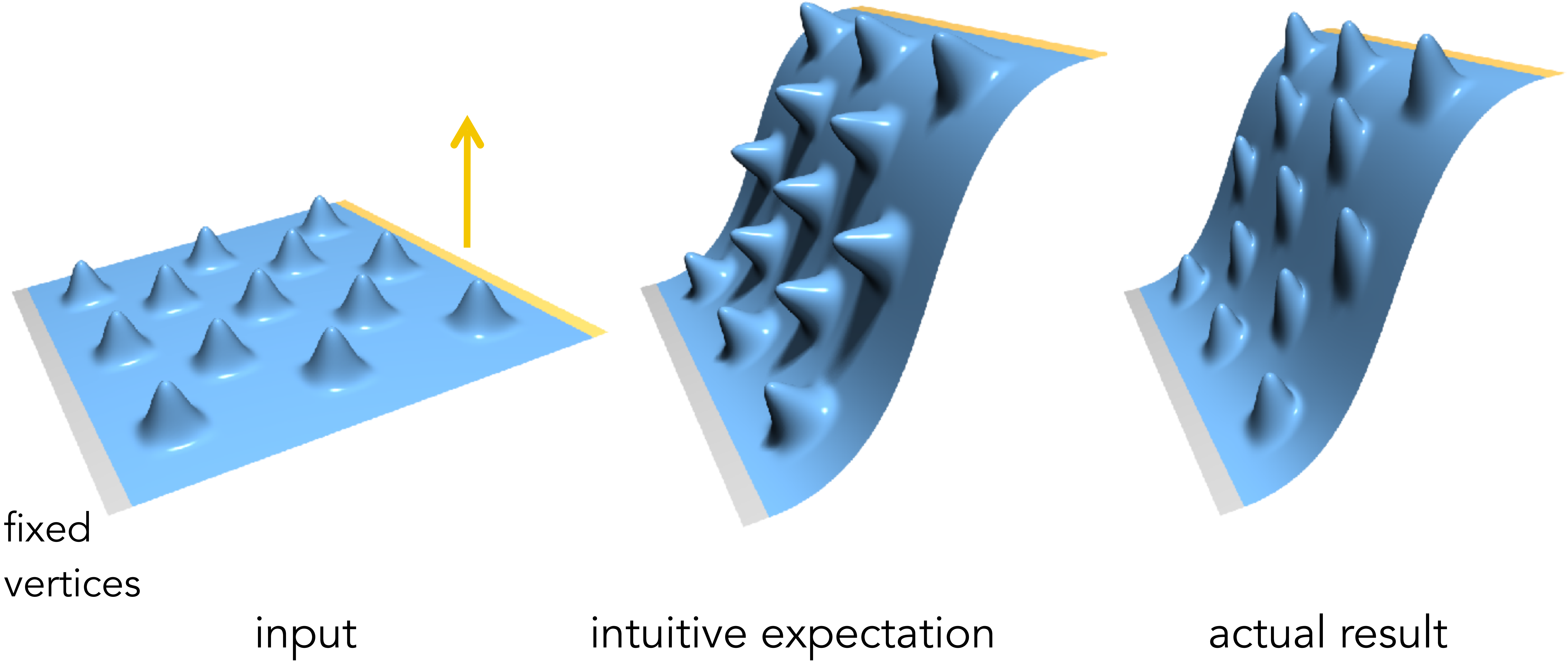
University of Victoria | Computer Science

Fundamental Problem: Invariance to Transformations

- The basic Laplacian operator is ***translation***-invariant, but not **rotation-**invariant
- E(**x**') attempts to preserve the **original global** orientation of the details (the normal directions)

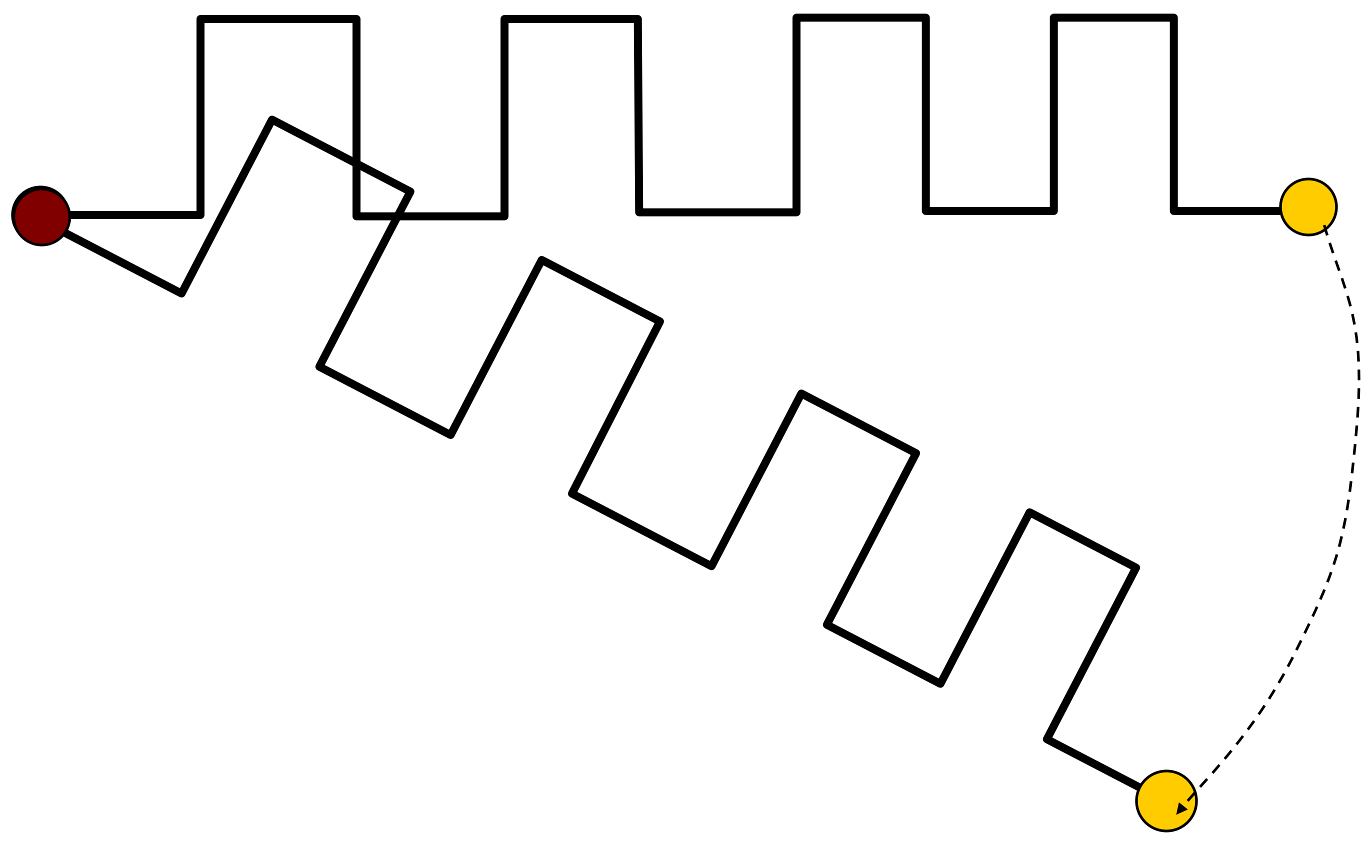


Fundamental Problem: Invariance to Transformations



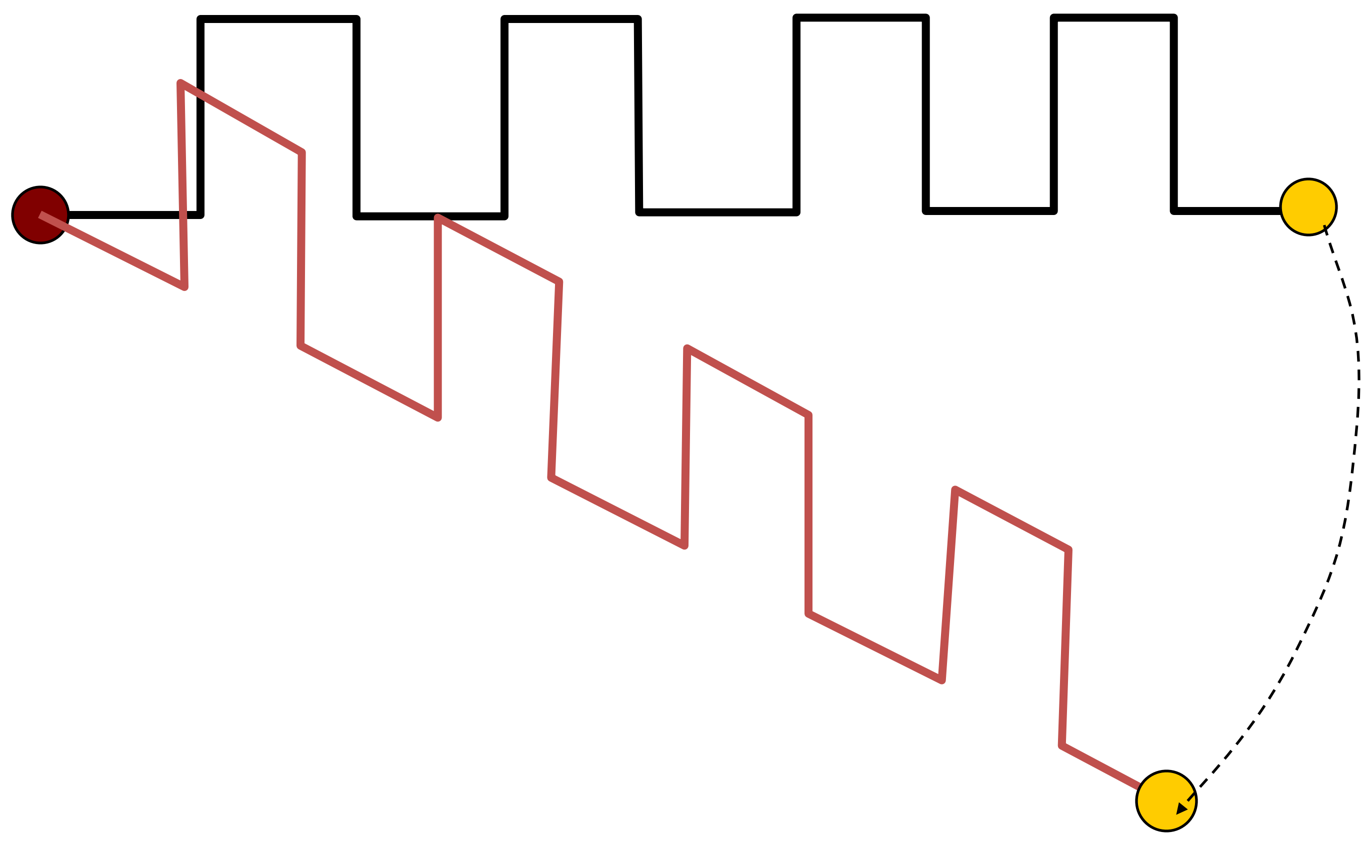
Fundamental Problem: Invariance to Transformations

- The basic Laplacian operator is ***translation***-invariant, but not **rotation-**invariant
- E(**x**′) attempts to preserve the **original global** orientation of the details (the normal directions)



Fundamental Problem: Invariance to Transformations

- The basic Laplacian operator is ***translation***-invariant, but not **rotation-**invariant
- E(**x**') attempts to preserve the **original global** orientation of the details (the normal directions)



Energy Functional

- We need a rigid-invariant energy…

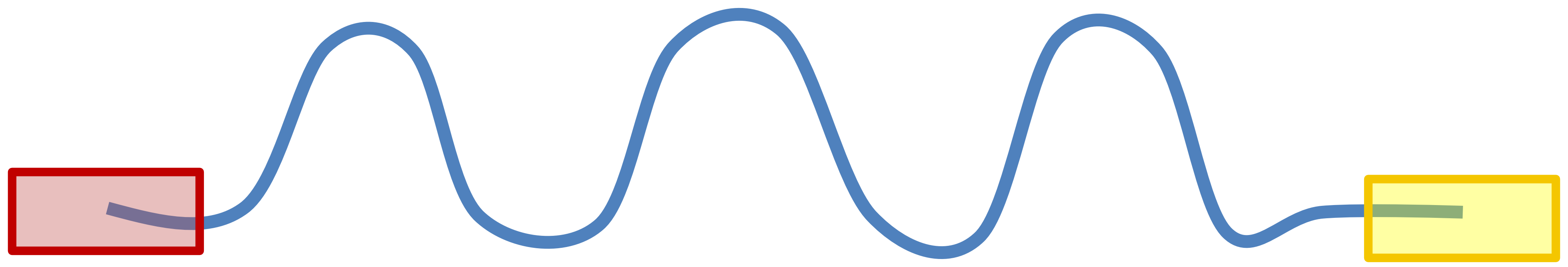
$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \delta_i\|^2$$

Need to locally
rotate the *target*
m.c. normals

Approach 1:
Multi-Resolution Mesh Editing

Multiresolution Approach

input



$\mathcal{S}$

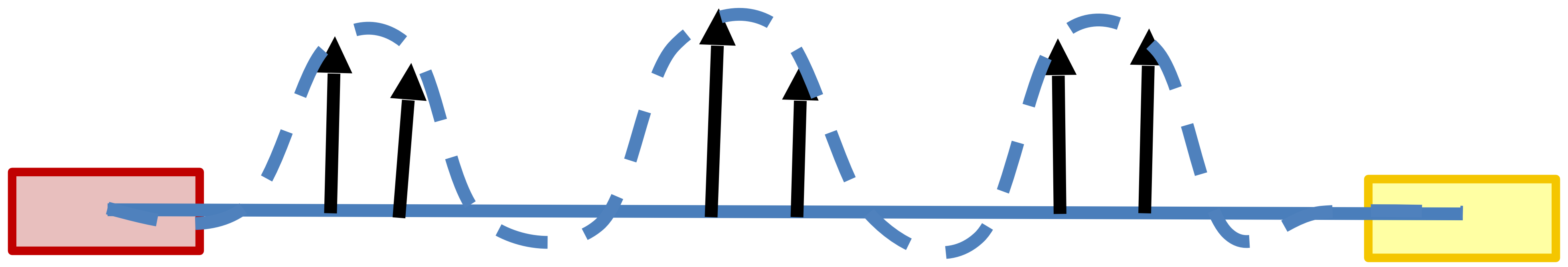
Multiresolution Approach

Smooth base surface

$\mathcal{B}$

Multiresolution Approach

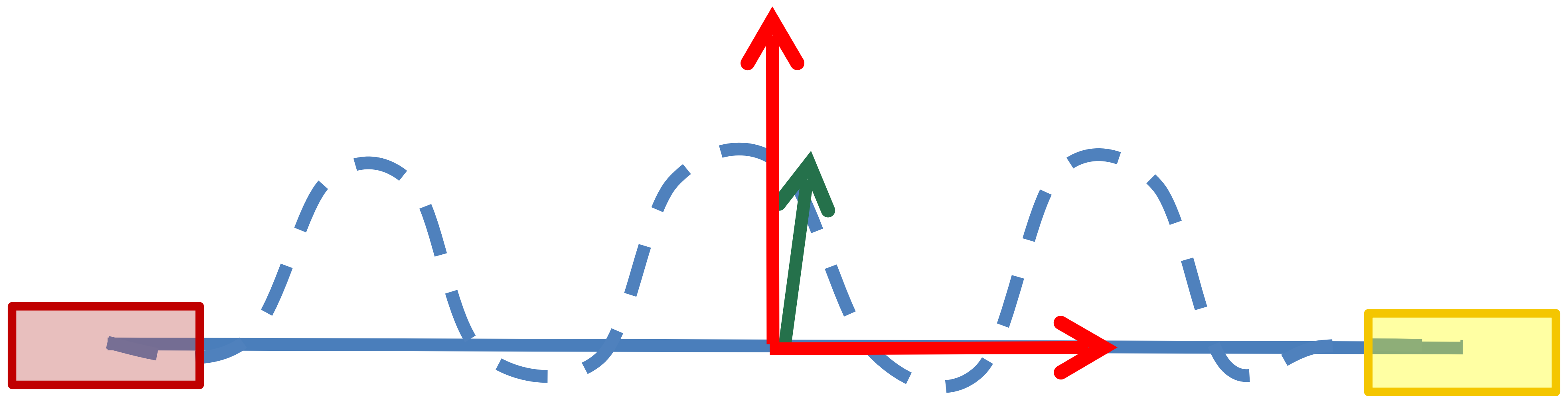
Details – displacement vectors



$$\mathcal{S} - \mathcal{B}$$

Multiresolution Approach

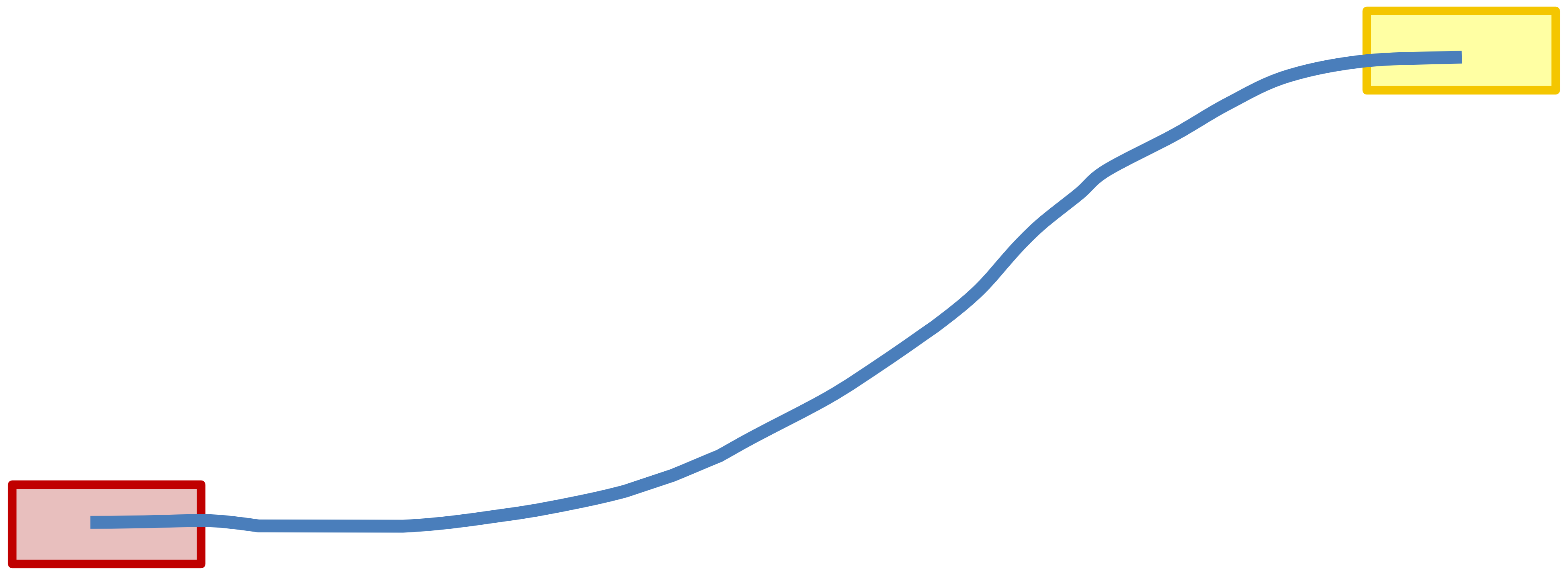
Encode details in the local frame of B



$$\mathbf{d}_i = a_1\mathbf{t}_i + a_2\mathbf{n}_i$$

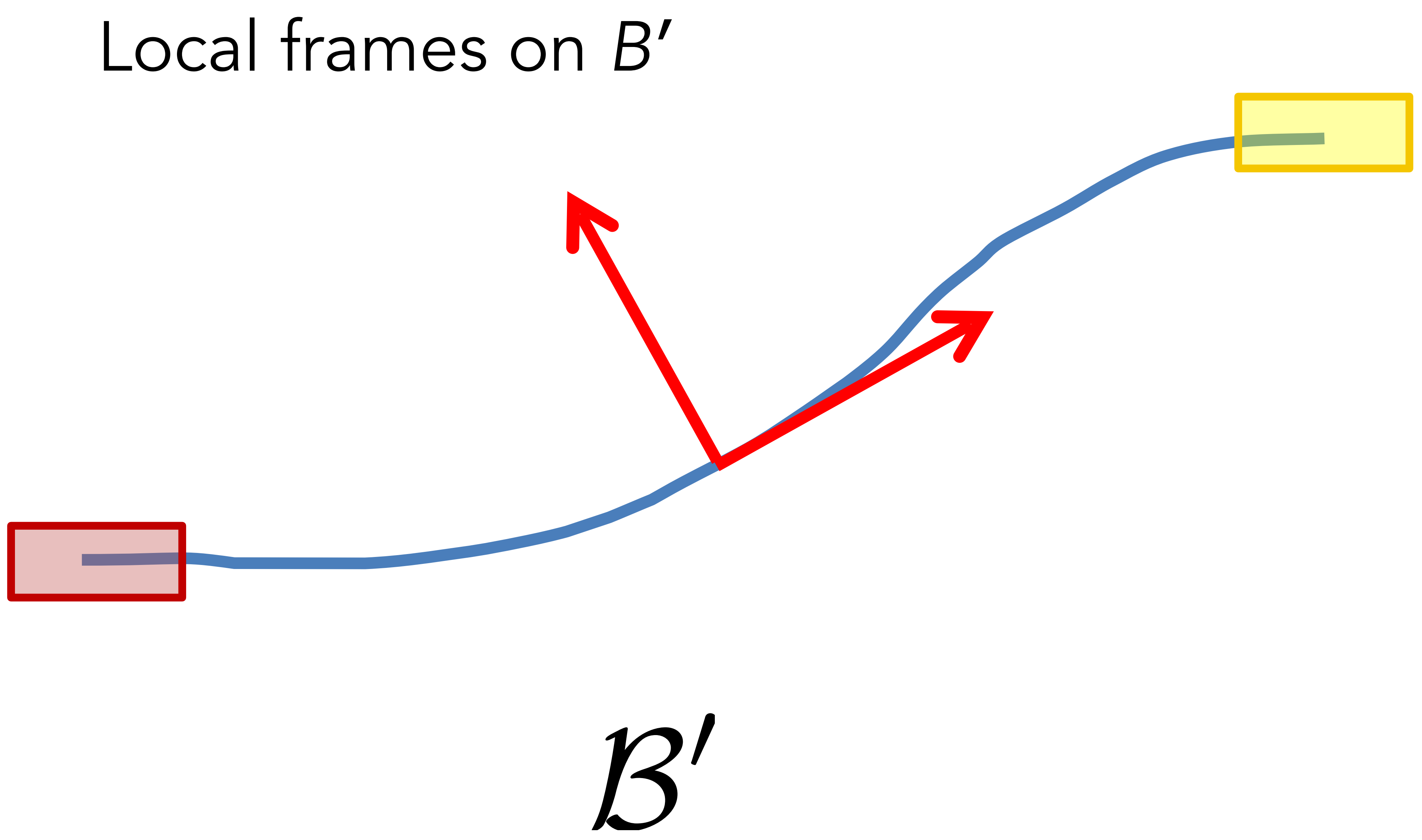
Multiresolution Approach

Deform smooth base surface



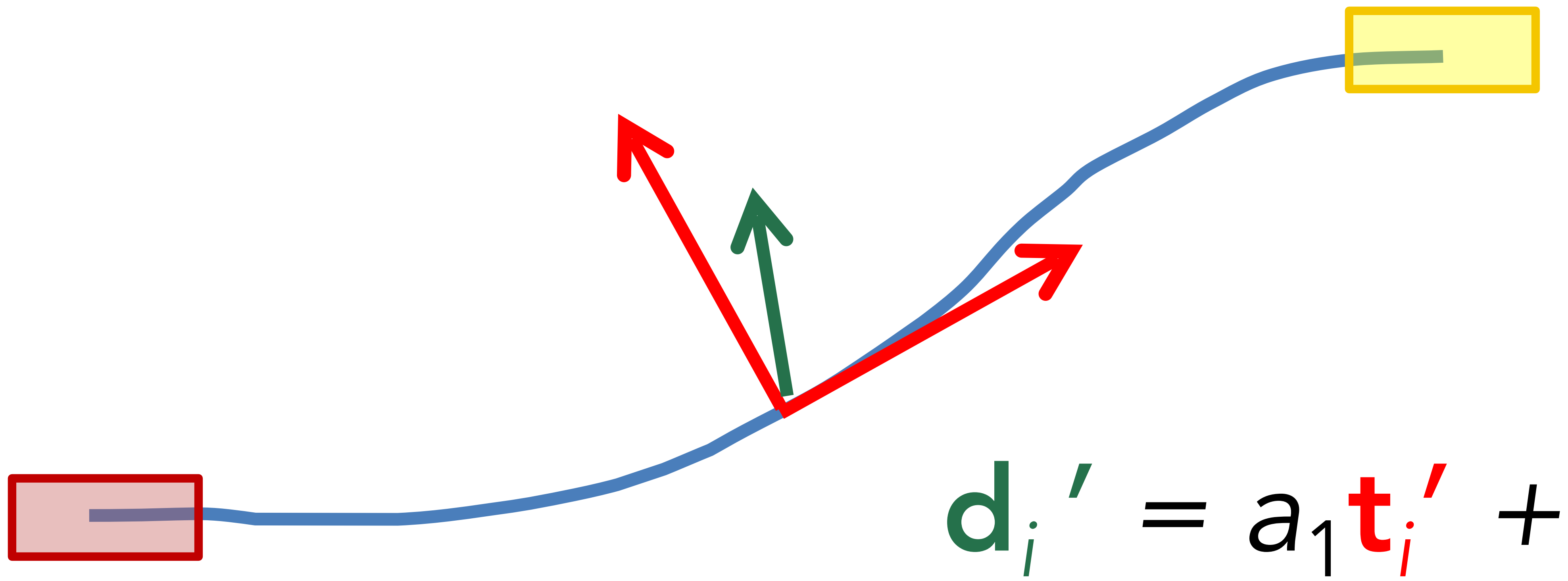
$\mathcal{B}'$

Multiresolution Approach



Multiresolution Approach

Add details back – in local frame!



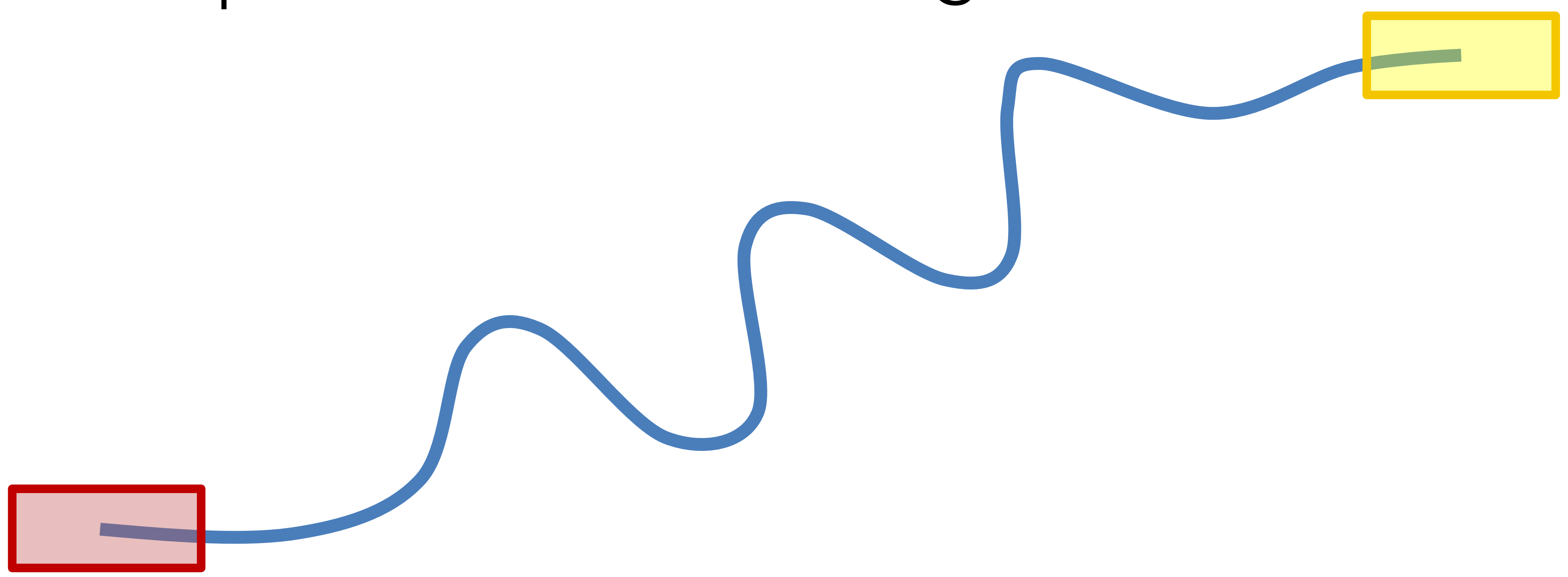
$$\mathbf{d}_i' = a_1\mathbf{t}_i' + a_2\mathbf{n}_i'$$

$$\mathcal{B}'$$

University of Victoria | Computer Science

Multiresolution Approach

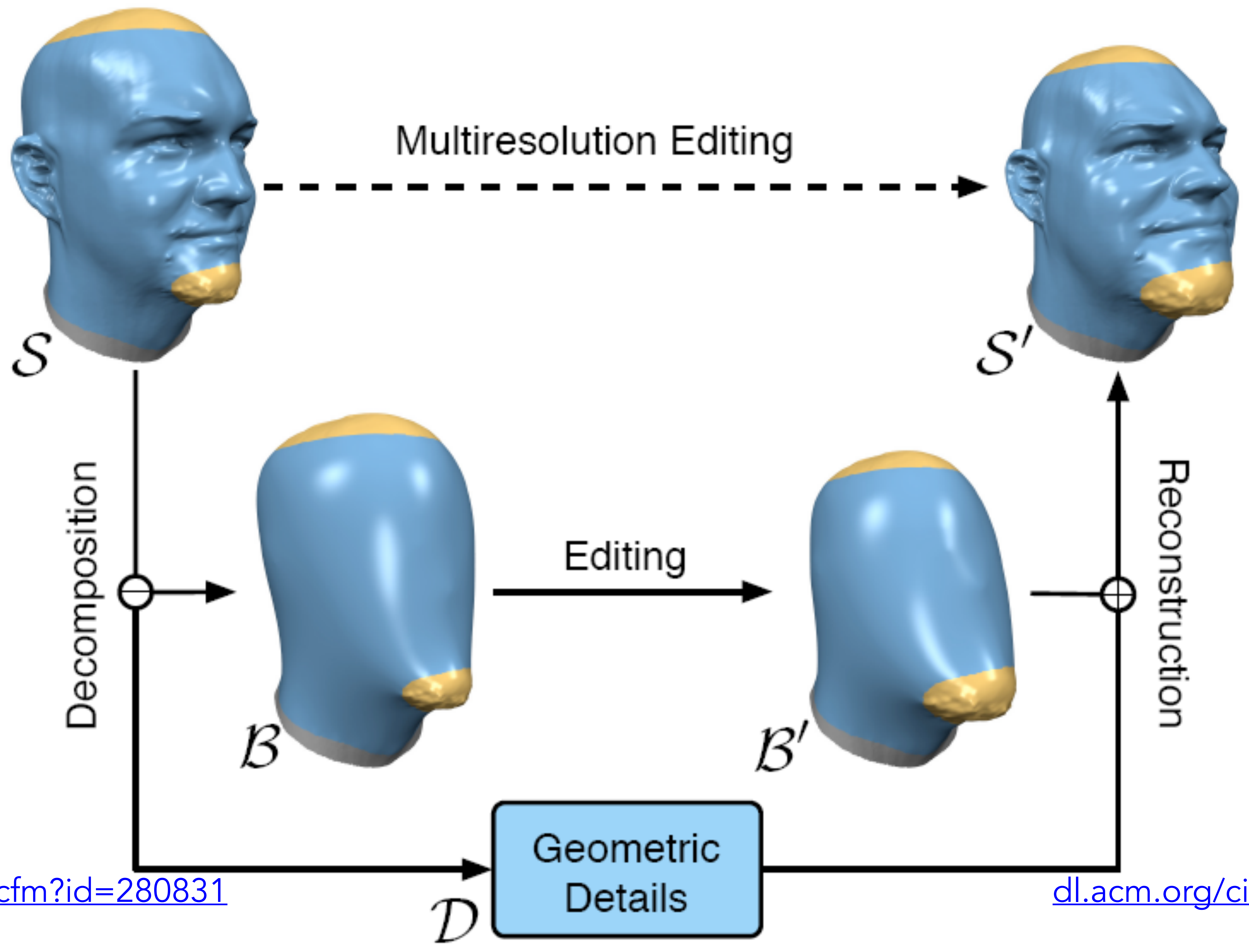
Displace the vertices to get the result



$\mathcal{S}'$

Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004

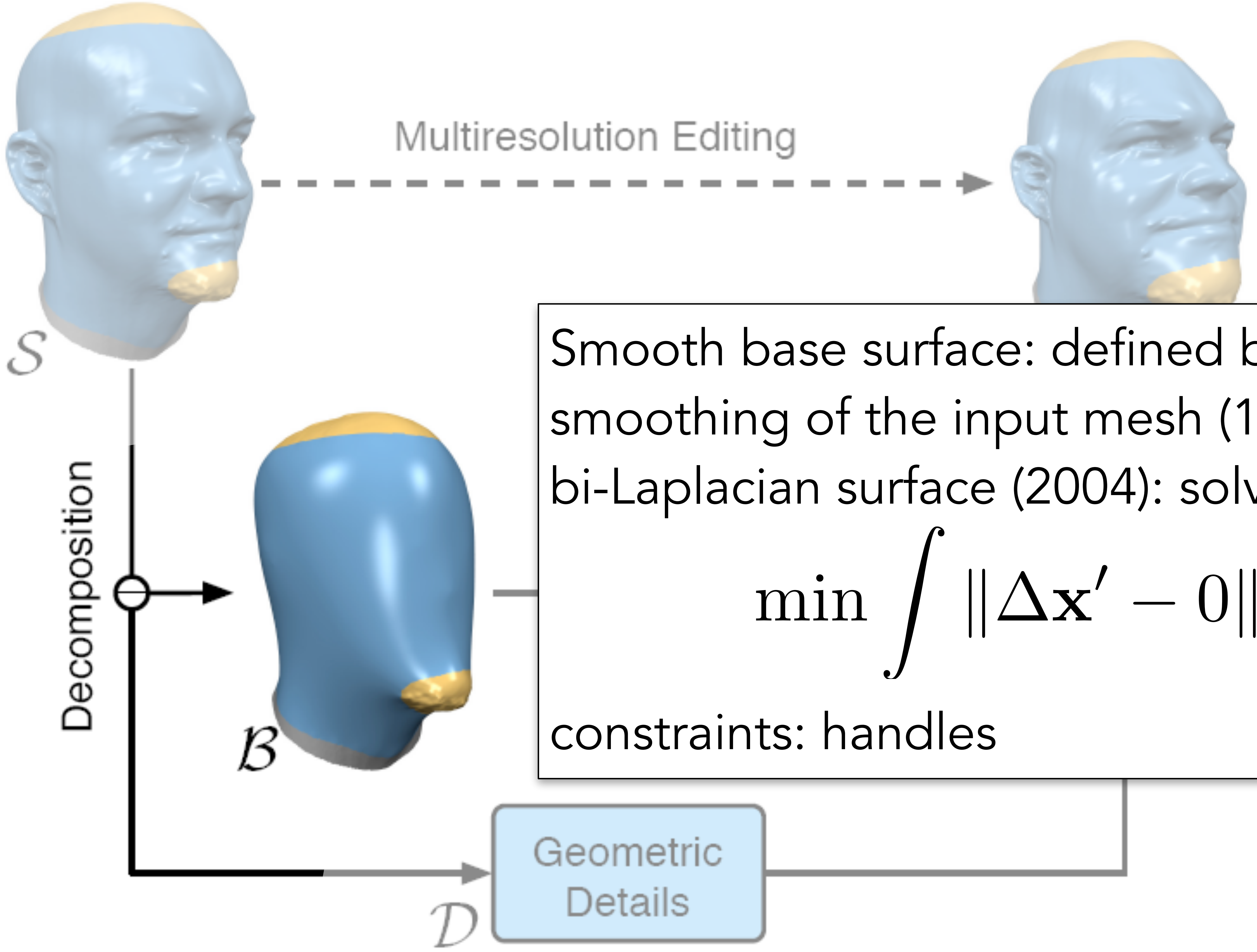


http://dl.acm.org/citation.cfm?id=280831

dl.acm.org/citation.cfm?id=1015772

Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



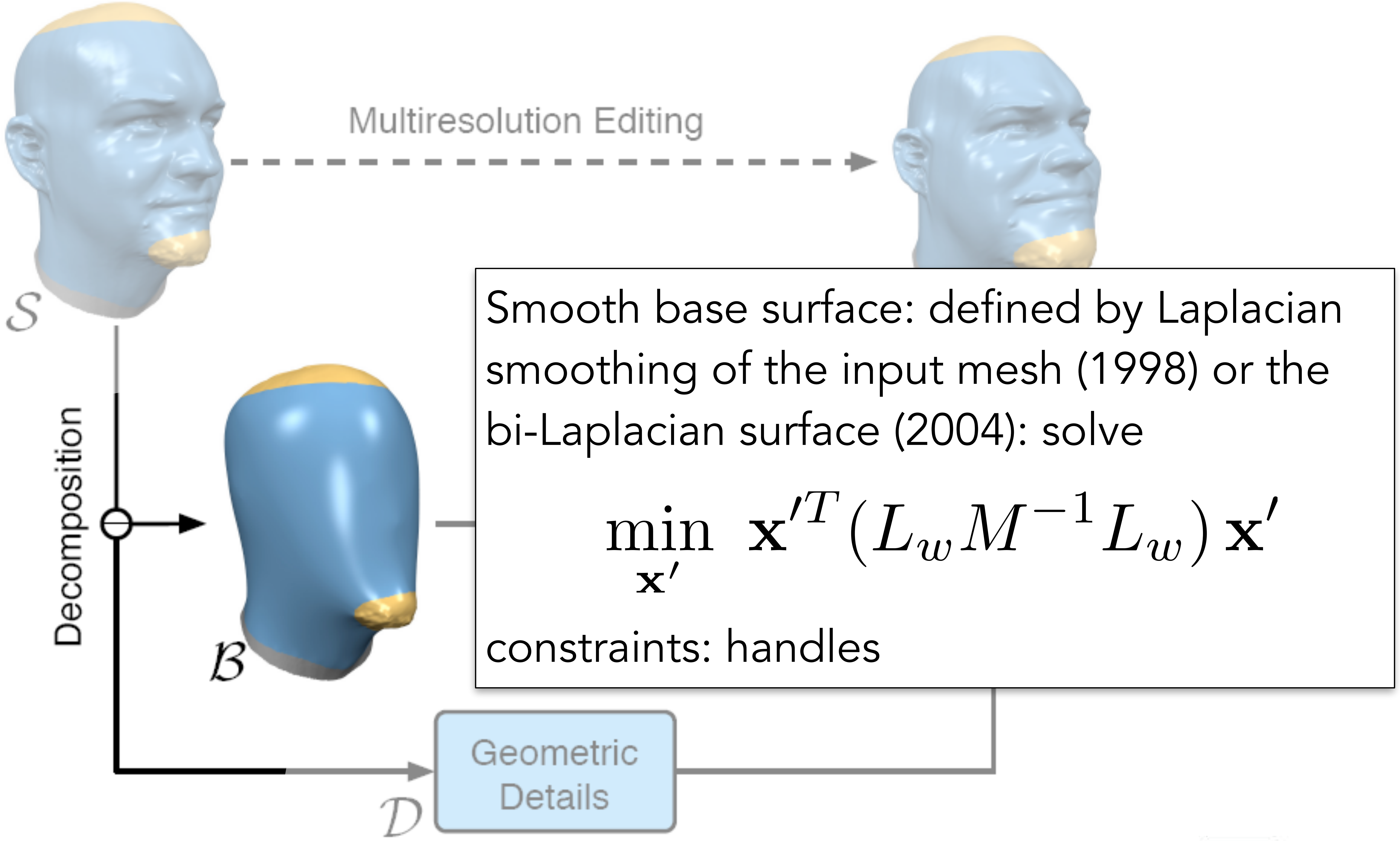
Smooth base surface: defined by Laplacian smoothing of the input mesh (1998) or the bi-Laplacian surface (2004): solve

$$\min \int \|\Delta \mathbf{x}' - 0\|^2 d\mathbf{x}'$$

constraints: handles

Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



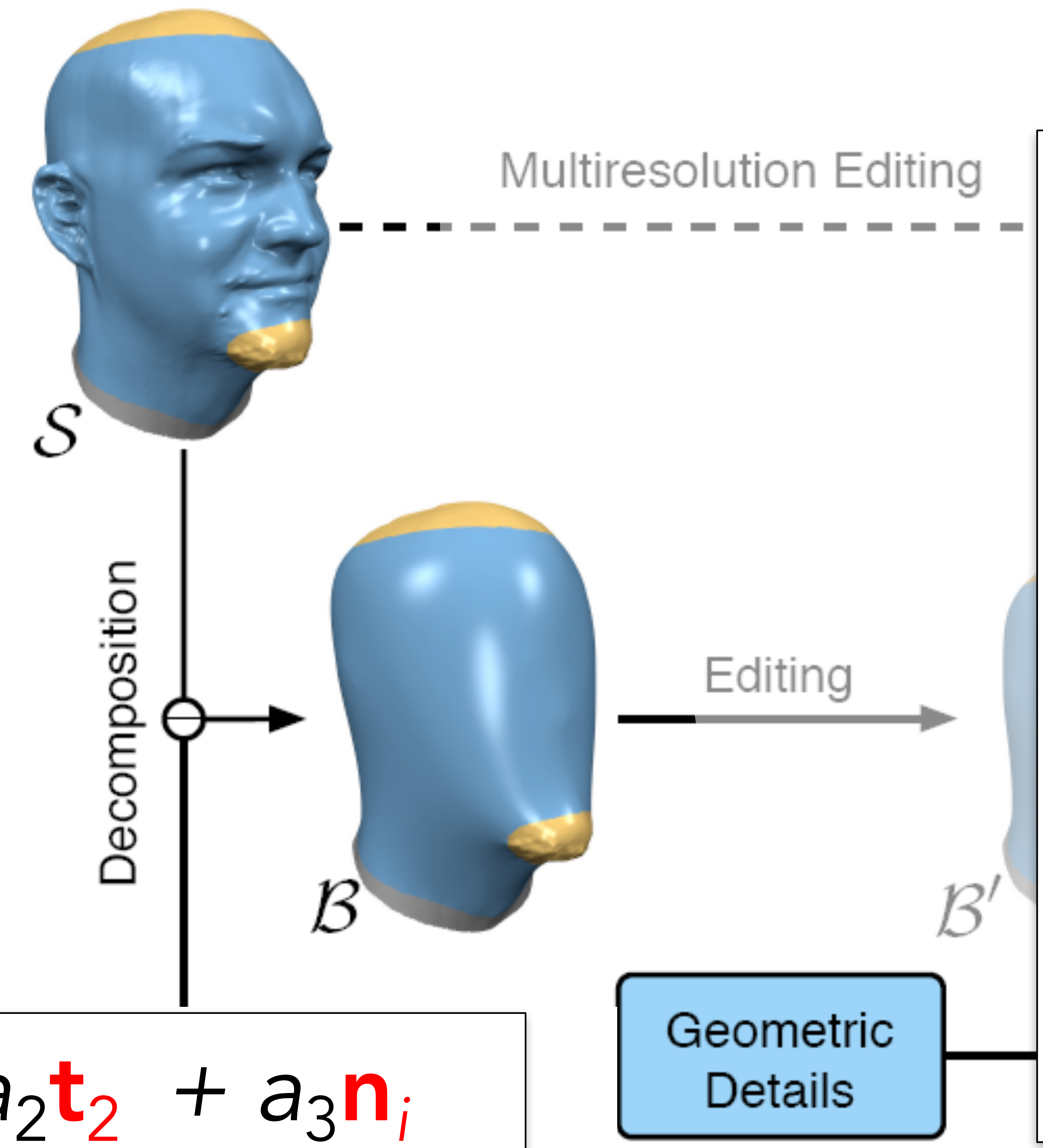
Smooth base surface: defined by Laplacian smoothing of the input mesh (1998) or the bi-Laplacian surface (2004): solve

$$\min_{\mathbf{x}'} \ \mathbf{x}'^T (L_w M^{-1} L_w)\, \mathbf{x}'$$

constraints: handles

Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



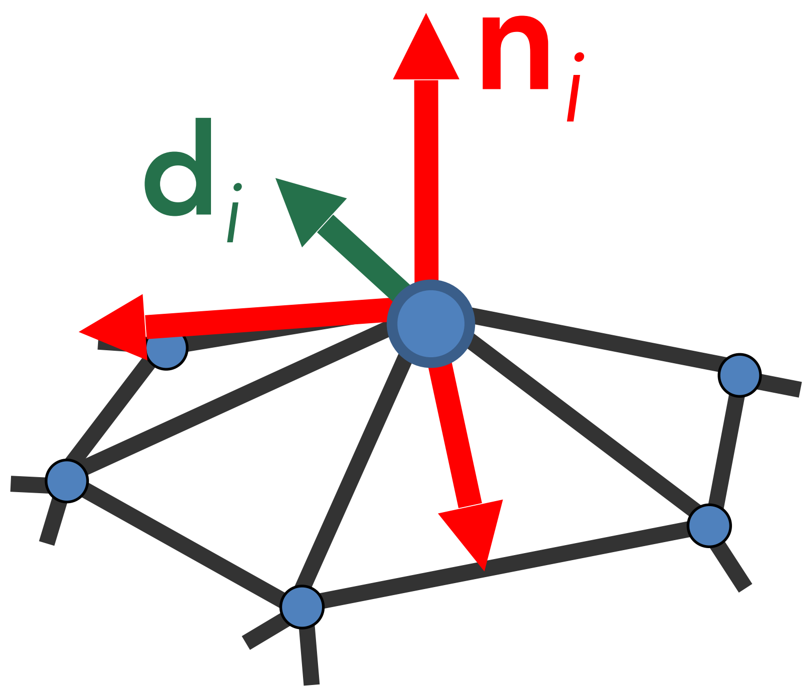
$$\mathbf{d}_i = a_1\mathbf{t}_1 + a_2\mathbf{t}_2 + a_3\mathbf{n}_i$$

Geometric details: simply the difference vectors between base surface B and input S

For each vertex, compute

$$\mathbf{d}_i = \mathbf{v}_i - \mathbf{b}_i$$

Represent $\mathbf{d}_i$ in the **local frame** of $\mathbf{b}_i$



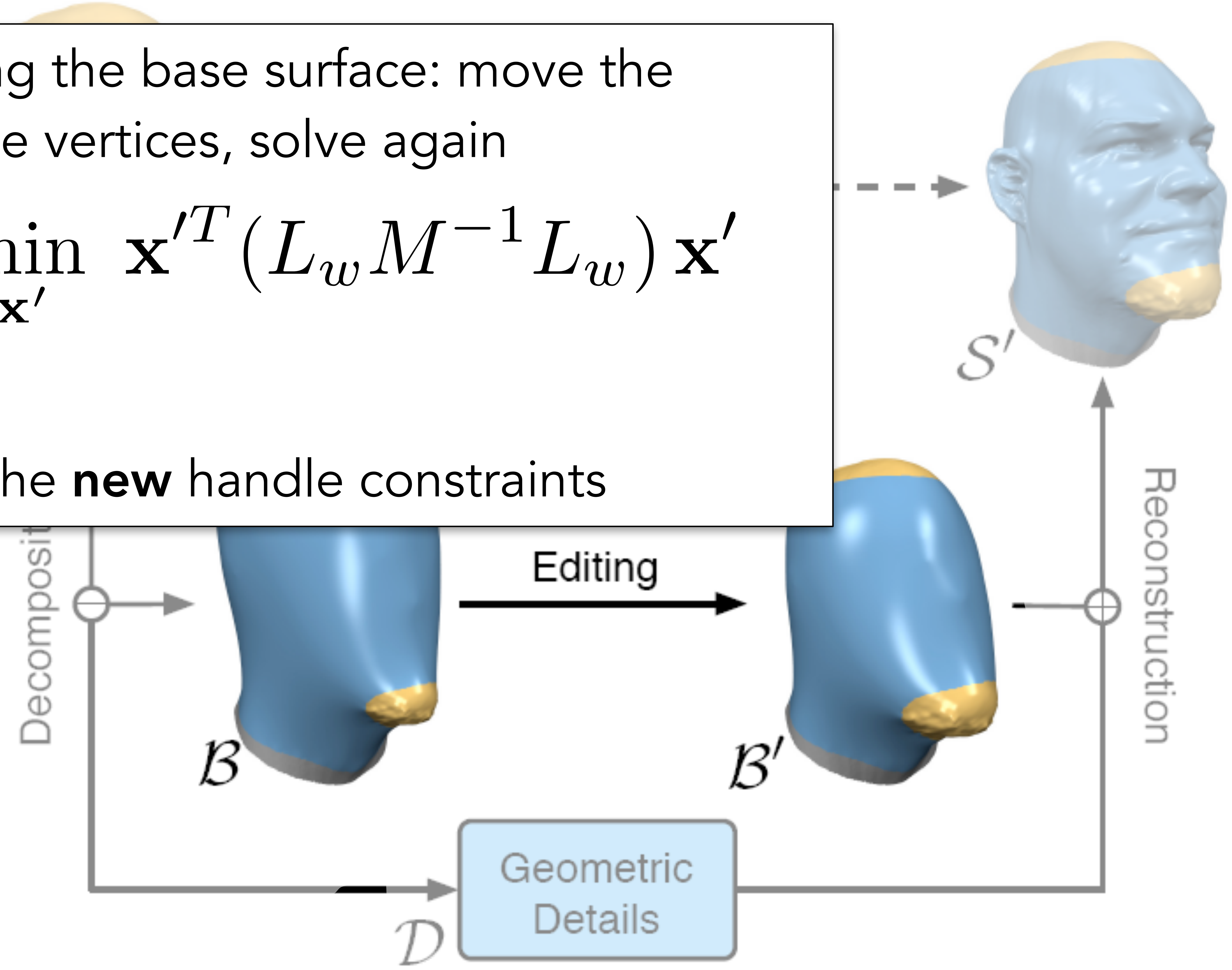
Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004

Editing the base surface: move the handle vertices, solve again

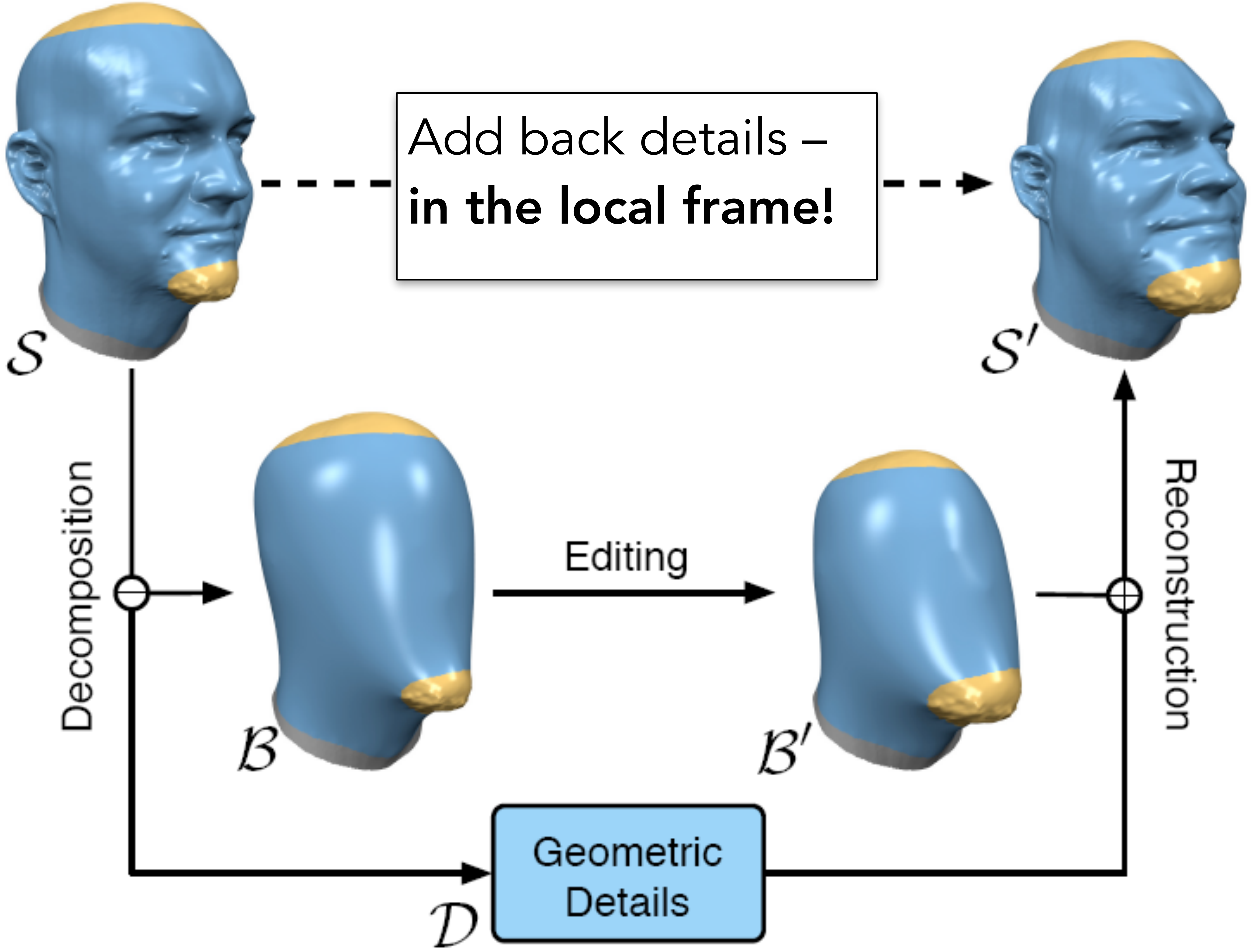
$$\min_{\mathbf{x}'} \; \mathbf{x}'^T \left(L_w M^{-1} L_w \right) \mathbf{x}'$$

with the **new** handle constraints



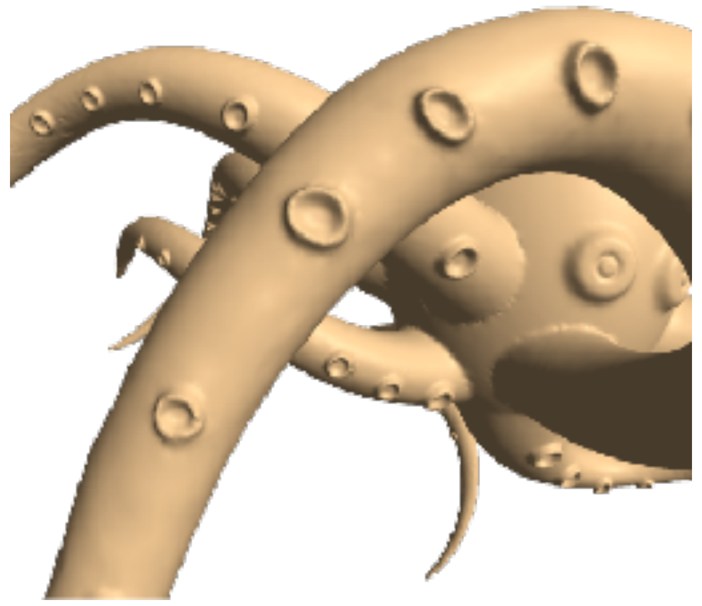
Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004

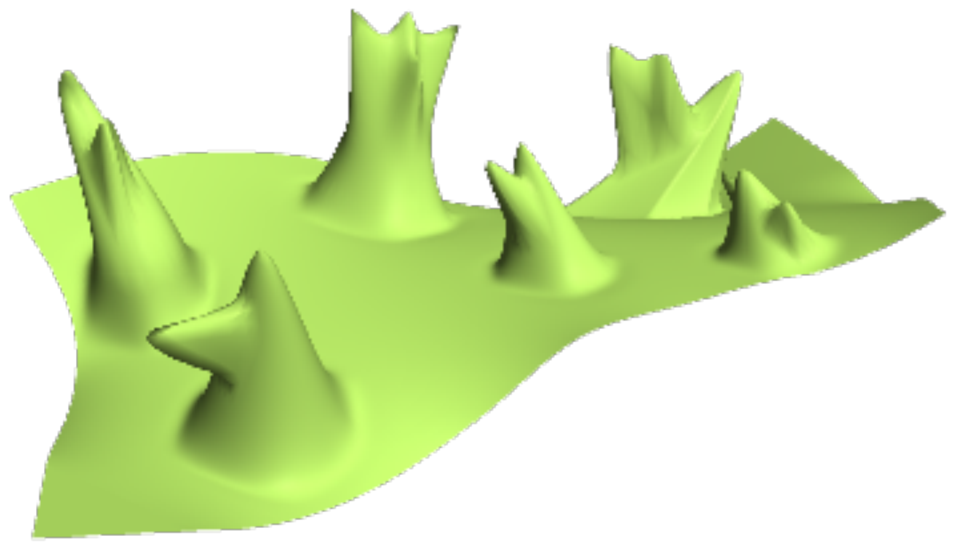


Discussion

- Pros:
  - Fast! Linear solve for the base surface deformation, and then add back displacements
  - Intuitive, easy to implement
- Cons:
  - works only for small height fields



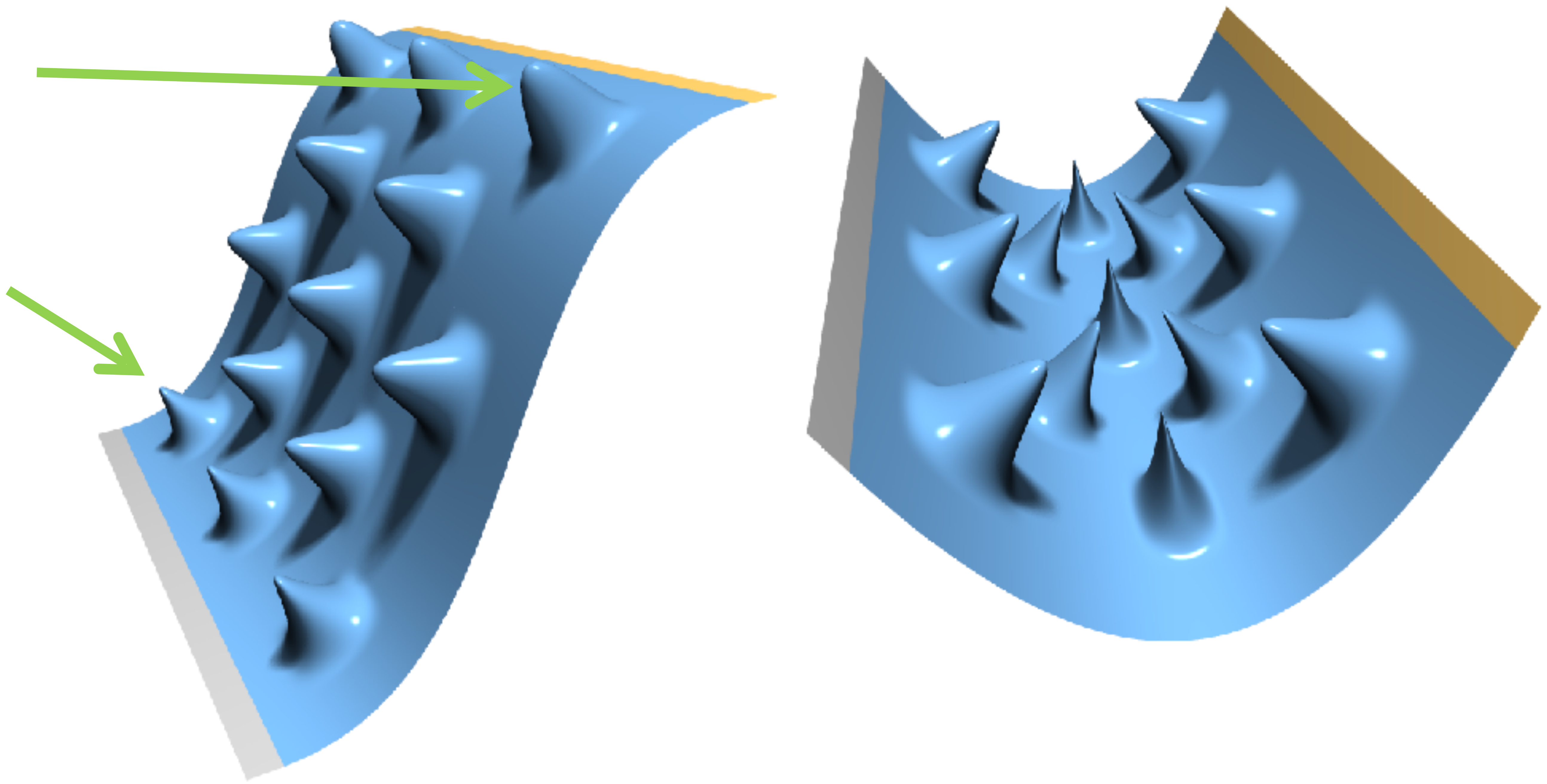
almost a height field



not a height field

Discussion

- If detail vectors are too big we get **overshooting** and **self-intersections**, especially in concave cases



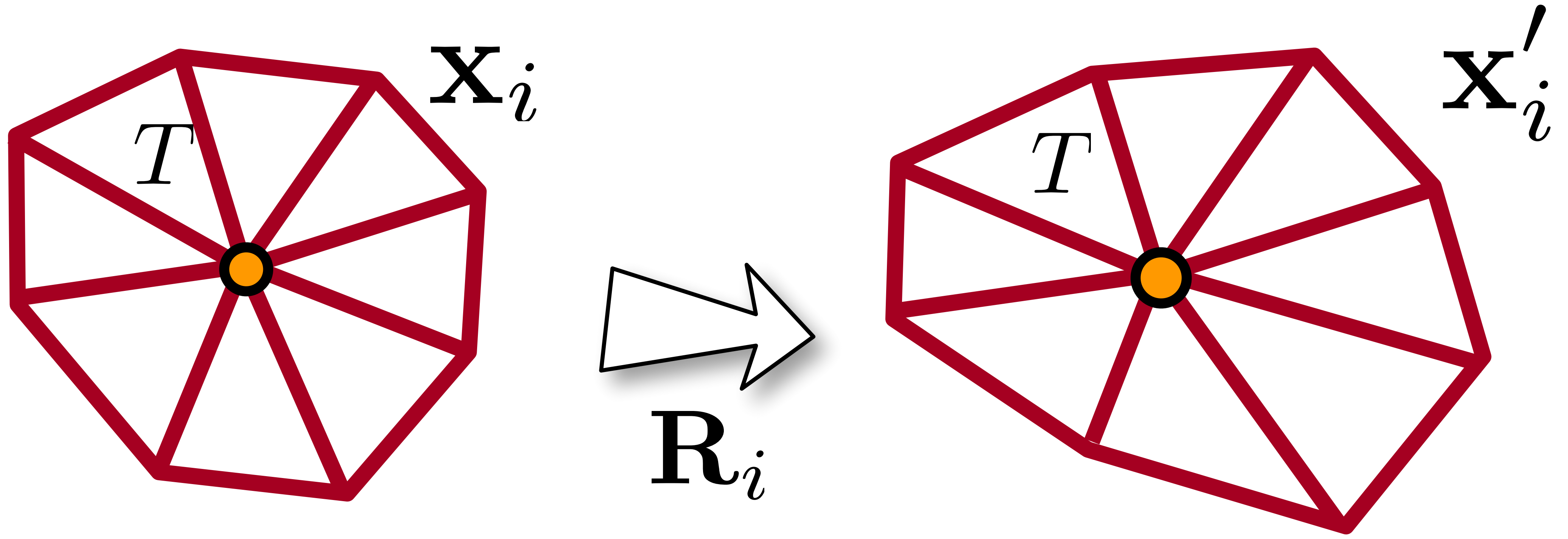
Approach 2:
As-Rigid-As-Possible Mesh Editing

Demo

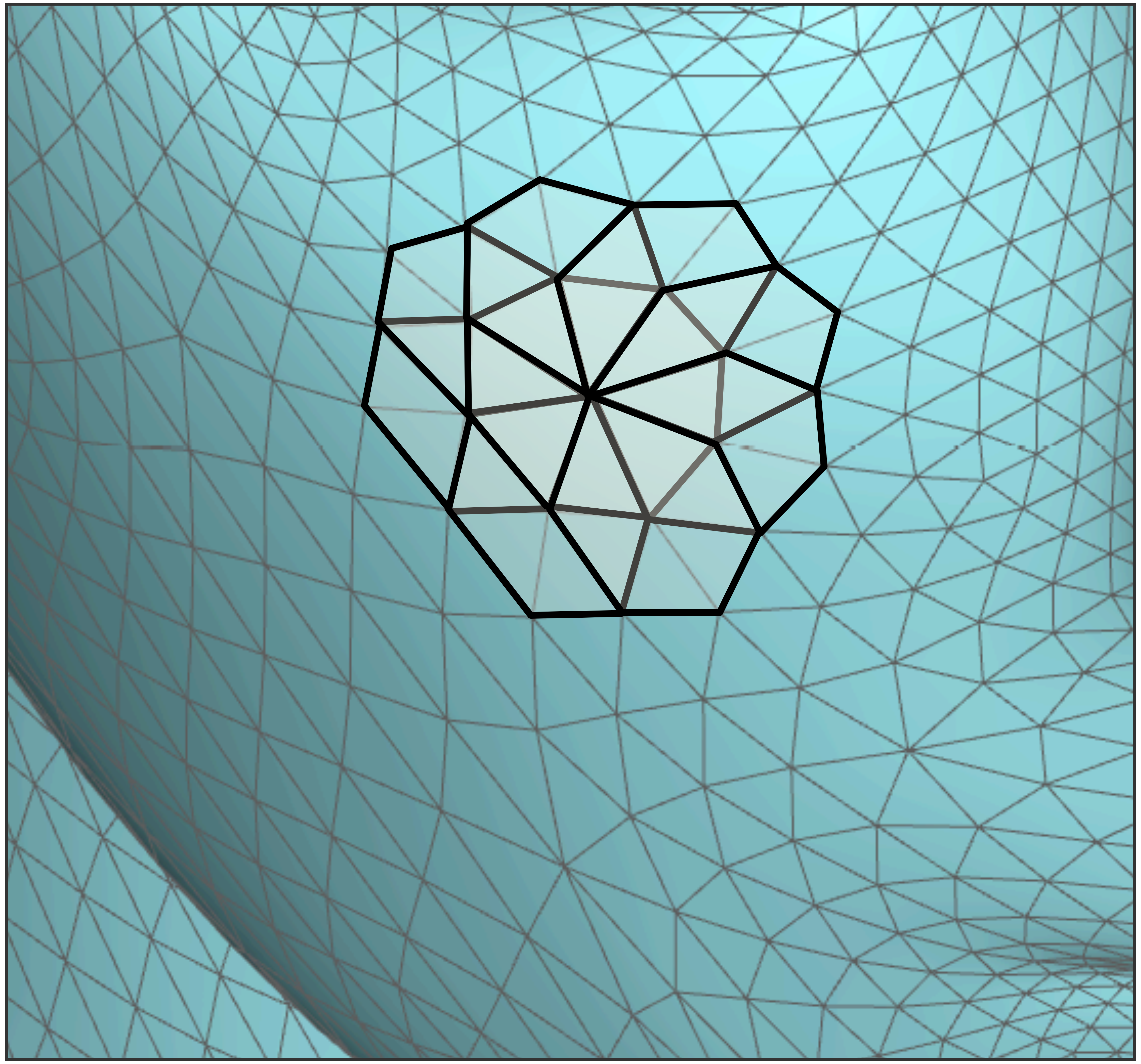
- https://libigl.github.io/libigl-python-bindings/tut-chapter3/

As-Rigid-As-Possible Deformation

- Preserve shape of cells covering the surface
- Ask each cell *i* to transform **rigidly** by best-fitting rotation $\mathbf{R}_i$

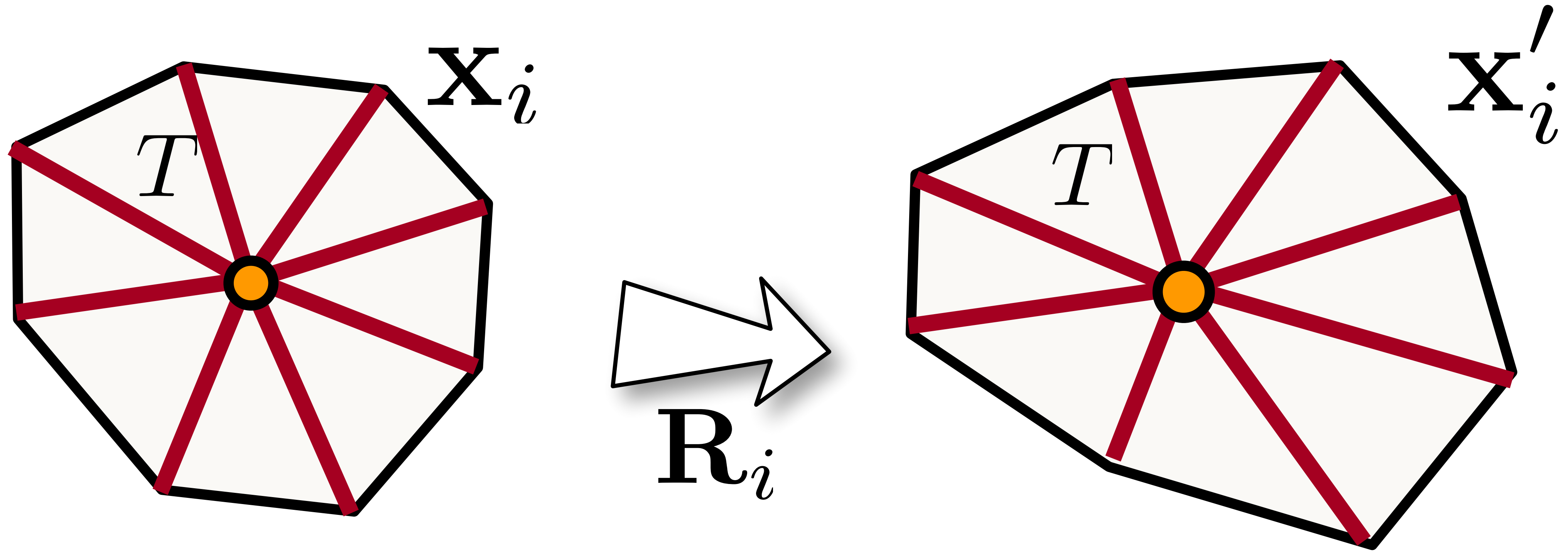


$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$



As-Rigid-As-Possible Deformation

- Optimal $\mathbf{R}_i$ is uniquely defined by $\mathbf{x}_i\ \mathbf{x}'_i$

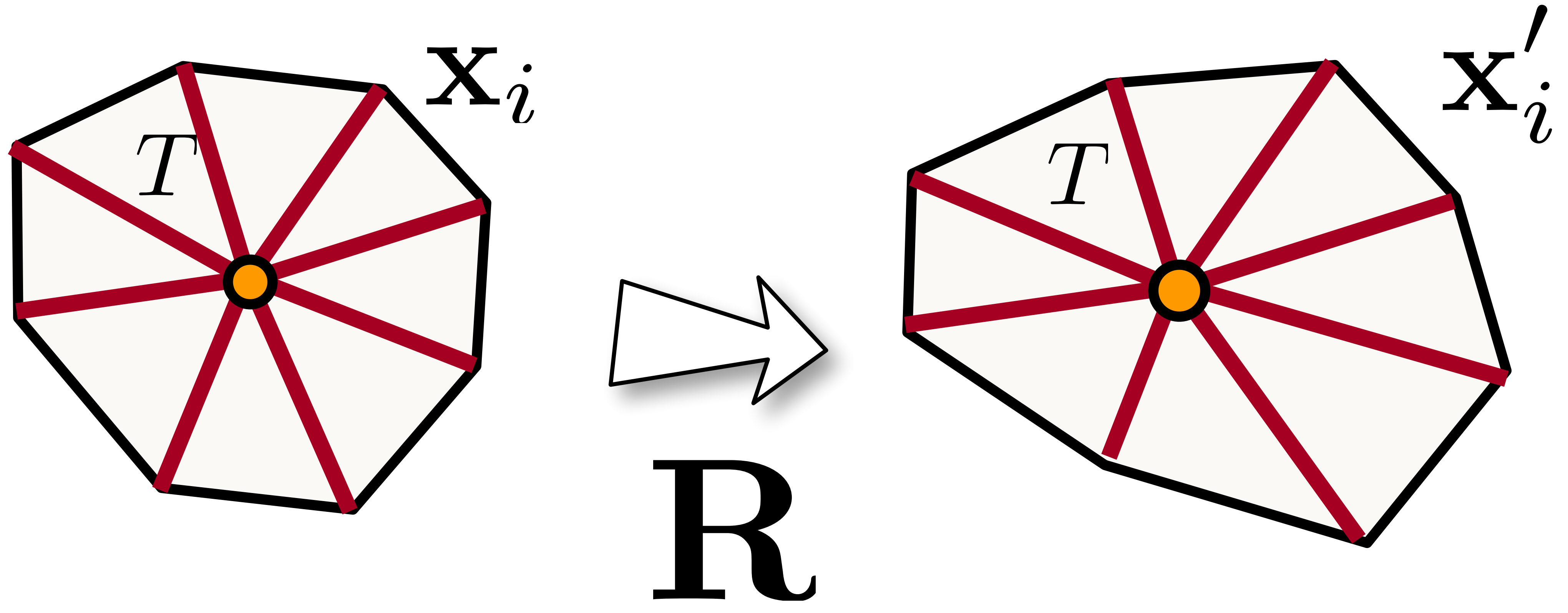


$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

- so-called shape-matching problem, solved by a 3x3 SVD

R$_i$ is a nonlinear function of **x**

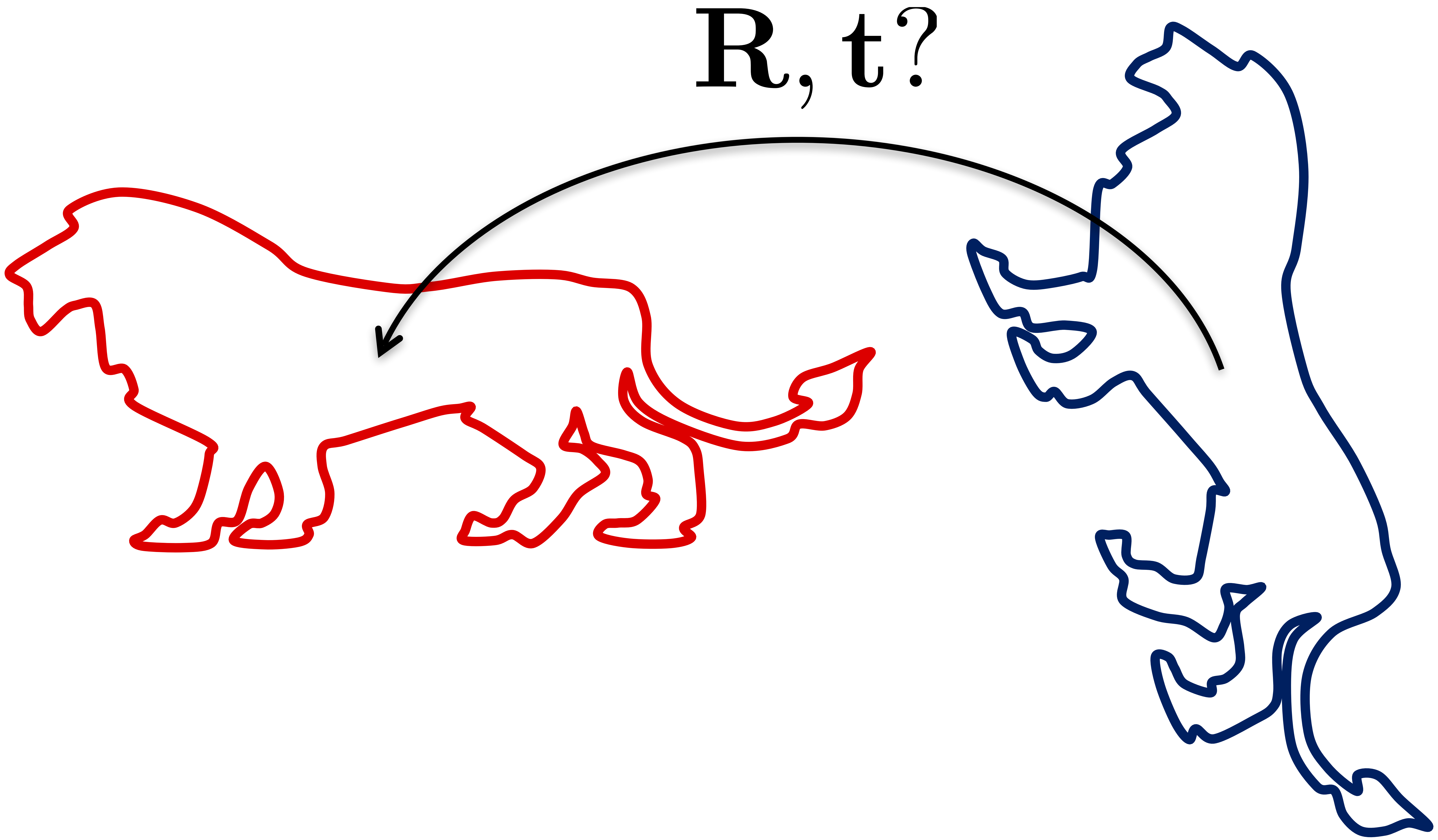
Optimal Rotation



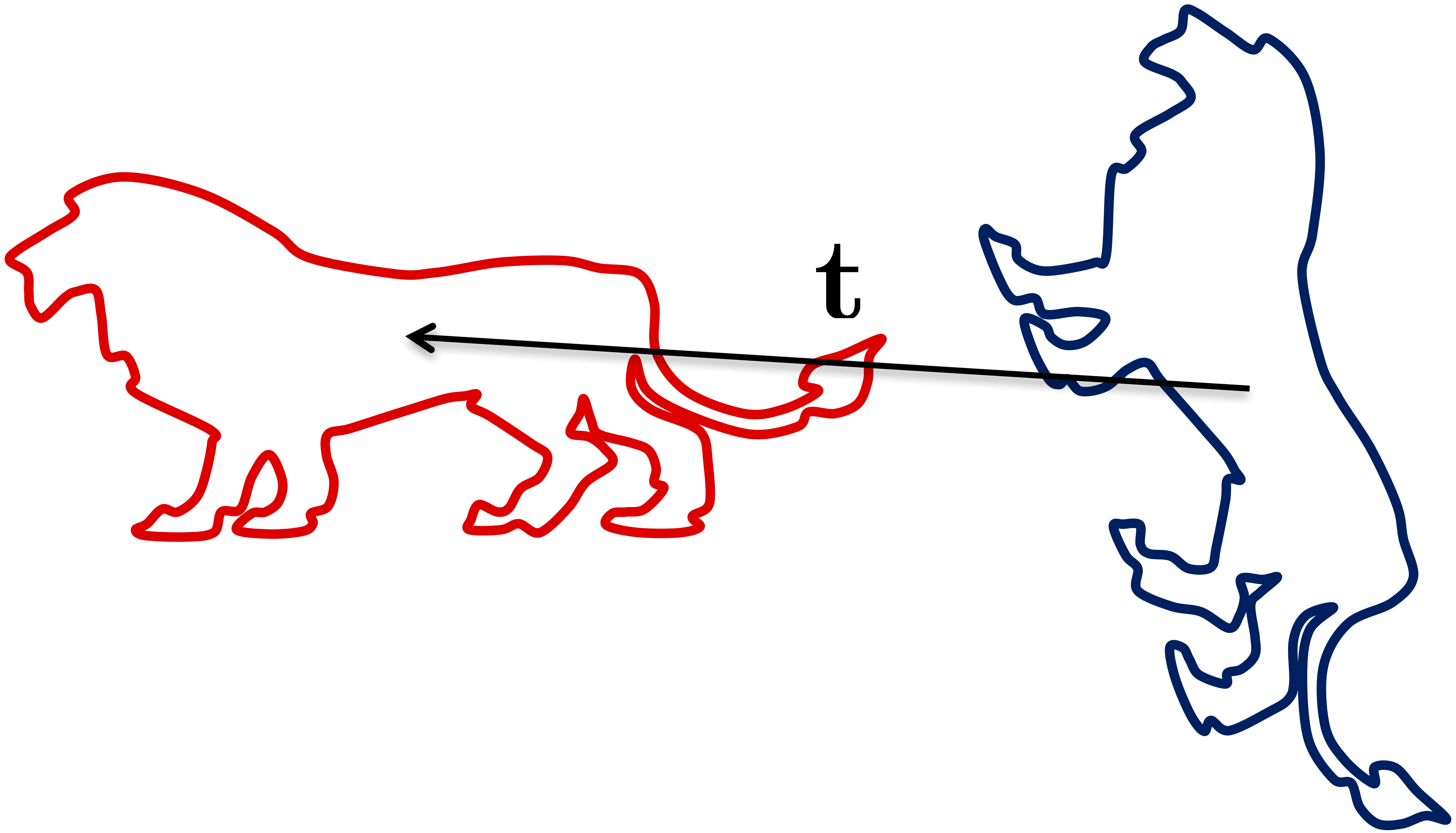
$$\min_{\mathbf{R}\in SO(3)} \sum_{T\in \text{Cell}_i} \sum_{(j,k)\in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

Rotation group

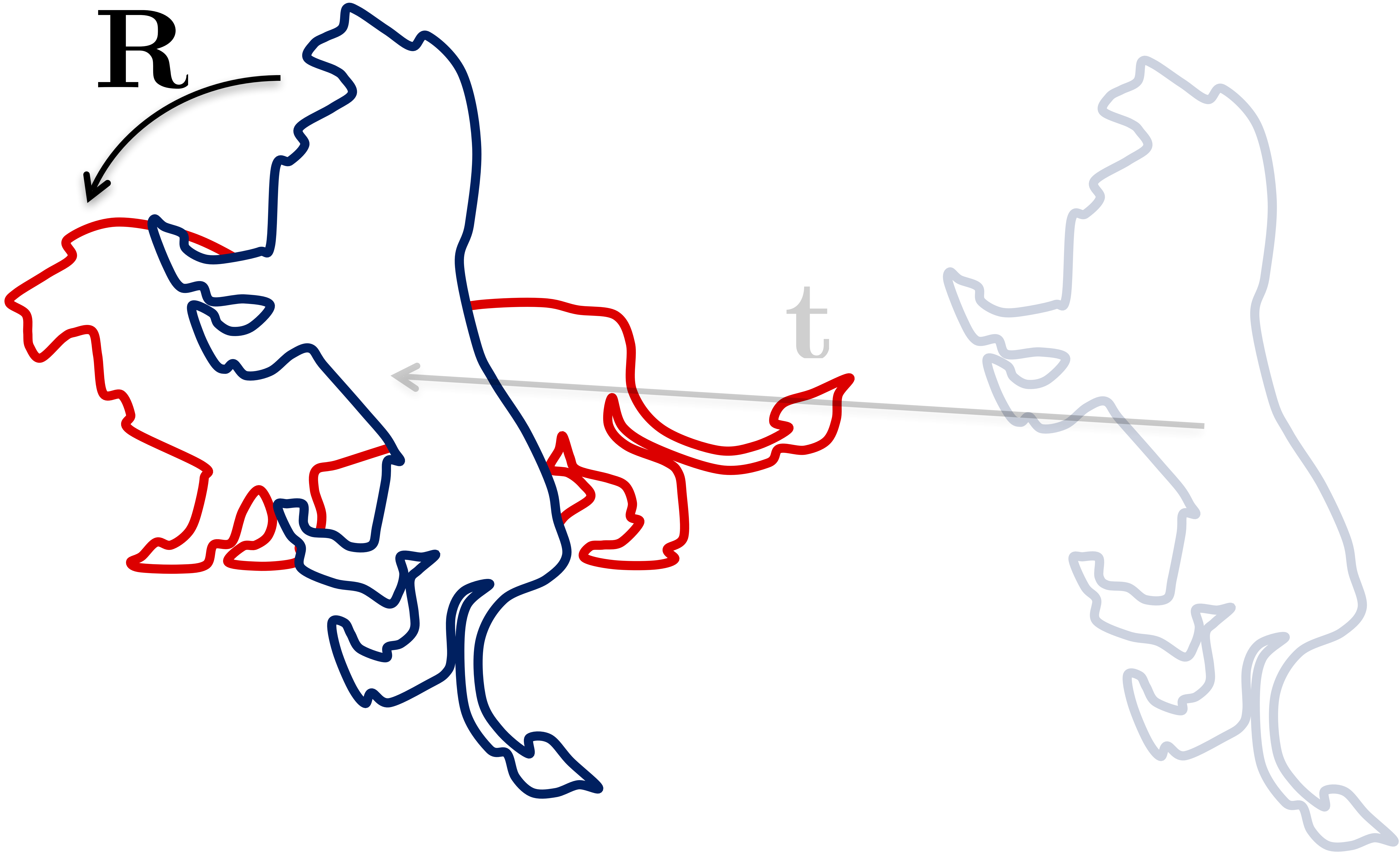
Shape Matching Problem



Shape Matching Problem

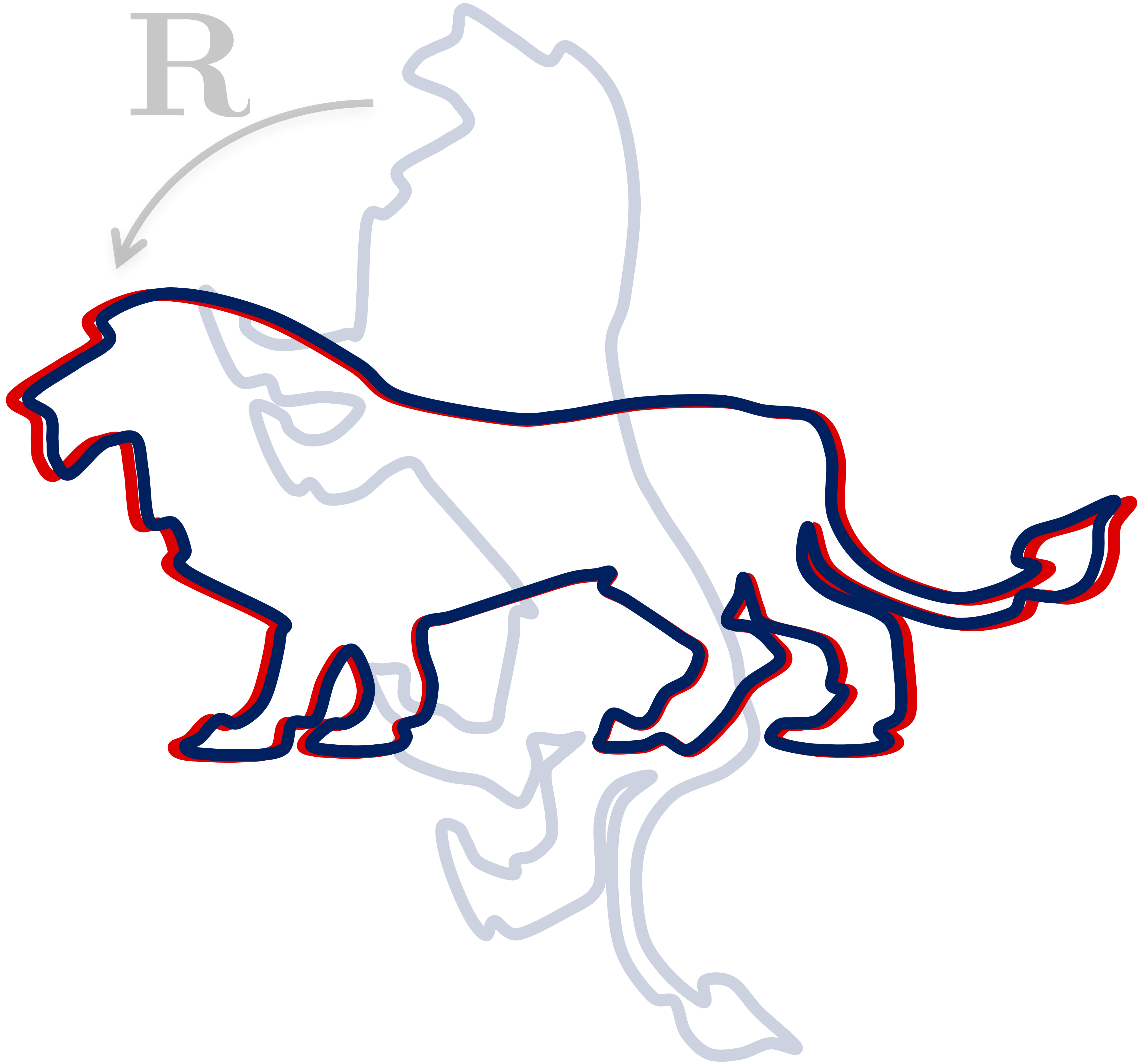


Shape Matching Problem



University of Victoria | Computer Science

Shape Matching Problem



Shape Matching Problem

- Align two point sets

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \text{ and } \mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$$

- Find a translation vector **t** and rotation matrix **R** so that

$$\sum_{i=1}^{n} \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\|^2 \text{ is minimized}$$

Shape Matching – Solution

- Solve for translation first (w.r.t. **R**, **p**, and **q**)

$$\frac{\partial}{\partial \mathbf{t}} \sum_{i=1}^{n} \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\|^2 = \sum_{i=1}^{n} 2\left((\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\right) \stackrel{!}{=} 0$$

$$\mathbf{R}\sum_{i=1}^{n} \mathbf{p}_i + \sum_{i=1}^{n} \mathbf{t} - \sum_{i=1}^{n} \mathbf{q}_i = 0$$

$$\mathbf{t} = \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i\right)}_{\bar{\mathbf{q}}} - \mathbf{R}\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i\right)}_{\bar{\mathbf{p}}}$$

Point sets $\{\mathbf{q}_i\}$ and $\{\mathbf{R}\mathbf{p}_i\}$ have the same center of mass

Finding the Rotation $\mathbf{R}$

- To find the optimal **R**, we bring the centroids of both point sets to the origin

$$\mathbf{v}_i = \mathbf{p}_i - \bar{\mathbf{p}}, \qquad \mathbf{v}'_i = \mathbf{q}_i - \bar{\mathbf{q}}$$

- We want to find **R** that minimizes

$$\sum_{i=1}^{n} \|\mathbf{R}\mathbf{v}_i - \mathbf{v}'_i\|^2$$

Finding the Rotation **R**

$$\sum_{i=1}^{n} \|\mathbf{R}\mathbf{v}_i - \mathbf{v}_i'\|^2 = \sum_{i=1}^{n} (\mathbf{R}\mathbf{v}_i - \mathbf{v}_i')^T (\mathbf{R}\mathbf{v}_i - \mathbf{v}_i') =$$

$$= \sum_{i=1}^{n} \left(\mathbf{v}_i^T \underbrace{\mathbf{R}^T \mathbf{R}}_{\mathbf{I}} \mathbf{v}_i - \mathbf{v}_i'^T \mathbf{R}\mathbf{v}_i - \mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' + \mathbf{v}_i'^T \mathbf{v}_i' \right)$$

These terms do not depend on **R**,
so we can ignore them in the minimization

Finding the Rotation $\mathbf{R}$

$$\underset{\mathbf{R}\in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} \left(-\mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i - \mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' \right) = \underset{\mathbf{R}\in SO(3)}{\operatorname{argmax}} \sum_{i=1}^{n} \left(\mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i + \underbrace{\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i'} \right) =$$

$$= \boxed{\underset{\mathbf{R}\in SO(3)}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i}$$

$$\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' = \left(\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' \right)^T = \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i$$

Finding the Rotation $\mathbf{R}$

$$\sum_{i=1}^{n} \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i = tr\left(\mathbf{V}'^T \mathbf{R} \mathbf{V}\right)$$

$$\underbrace{\begin{bmatrix} \mathbf{v}_1'^T \\ \mathbf{v}_2'^T \\ \vdots \\ \mathbf{v}_n'^T \end{bmatrix}}_{\mathbf{V}'^T} \begin{bmatrix} \mathbf{R} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}}_{\mathbf{V}} = \begin{bmatrix} \mathbf{v}_1'^T \\ \mathbf{v}_2'^T \\ \vdots \\ \mathbf{v}_n'^T \end{bmatrix} \begin{bmatrix} \mathbf{R}\mathbf{v}_1 & \mathbf{R}\mathbf{v}_2 & \cdots & \mathbf{R}\mathbf{v}_n \end{bmatrix}$$

University of Victoria | Computer Science

Finding the Rotation $\mathbf{R}$

$$\sum_{i=1}^{n} \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i = tr\left(\mathbf{V}'^T \mathbf{R} \mathbf{V}\right)$$

$$\begin{bmatrix} \mathbf{v}_1'^T \\ \mathbf{v}_2'^T \\ \vdots \\ \mathbf{v}_n'^T \end{bmatrix} \begin{bmatrix} \mathbf{R}\mathbf{v}_1 & \mathbf{R}\mathbf{v}_2 & \cdots & \mathbf{R}\mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1'^T \mathbf{R}\mathbf{v}_1 & & & \\ & \mathbf{v}_2'^T \mathbf{R}\mathbf{v}_2 & & \\ & & \ddots & \\ & & & \mathbf{v}_n'^T \mathbf{R}\mathbf{v}_n \end{bmatrix}$$

Finding the Rotation $\mathbf{R}$

Take a look at the Matrix Cookbook!

- Find **R** that maximizes

$$tr\left(\mathbf{V'}^T\mathbf{R}\mathbf{V}\right) = tr\left(\mathbf{R}\mathbf{V}\mathbf{V'}^T\right)$$

- SVD: $\mathbf{V}\mathbf{V'}^T = \mathbf{U}\boldsymbol{\Sigma}\tilde{\mathbf{U}}^T$

$$tr\left(\mathbf{R}\mathbf{V}\mathbf{V'}^T\right) = tr\left(\underbrace{\mathbf{R}\,\mathbf{U}}\underbrace{\boldsymbol{\Sigma}\tilde{\mathbf{U}}^T}\right) = tr\left(\boldsymbol{\Sigma}\,\underbrace{\tilde{\mathbf{U}}^T\mathbf{R}\mathbf{U}}_{\text{orthonormal matrix}}\right)$$

University of Victoria | Computer Science

Finding the Rotation **R**

- We want to maximize

$$tr\left(\Sigma\,\mathbf{M}\right)$$

M: orthonormal matrix
all entries ≤ 1

$$\begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} \begin{bmatrix} m_{11} & \cdots & \\ \vdots & m_{22} & \vdots \\ & \cdots & m_{33} \end{bmatrix}$$

$$tr\left(\Sigma\,\mathbf{M}\right) = \sum_{i=1}^{3} \sigma_i\, m_{ii} \leq \sum_{i=1}^{3} \sigma_i$$

Finding the Rotation **R**

$$tr\left(\Sigma\,\mathbf{M}\right) = \sum_{i=1}^{3} \sigma_i\, m_{ii} \leq \sum_{i=1}^{3} \sigma_i$$

- Our best shot is m_{ii} = 1, i.e. to make **M = I**

$$\begin{aligned} \mathbf{M} = \tilde{\mathbf{U}}^T\mathbf{R}\mathbf{U} &\overset{!}{=} \mathbf{I} \\ \mathbf{R}\mathbf{U} &= \tilde{\mathbf{U}} \\ \boxed{\mathbf{R} = \tilde{\mathbf{U}}\mathbf{U}^T} \end{aligned}$$

Summary of Rigid Alignment

- Translate the input points to the centroids

$$\mathbf{v}_i = \mathbf{p}_i - \bar{\mathbf{p}}, \qquad \mathbf{v}'_i = \mathbf{q}_i - \bar{\mathbf{q}}$$

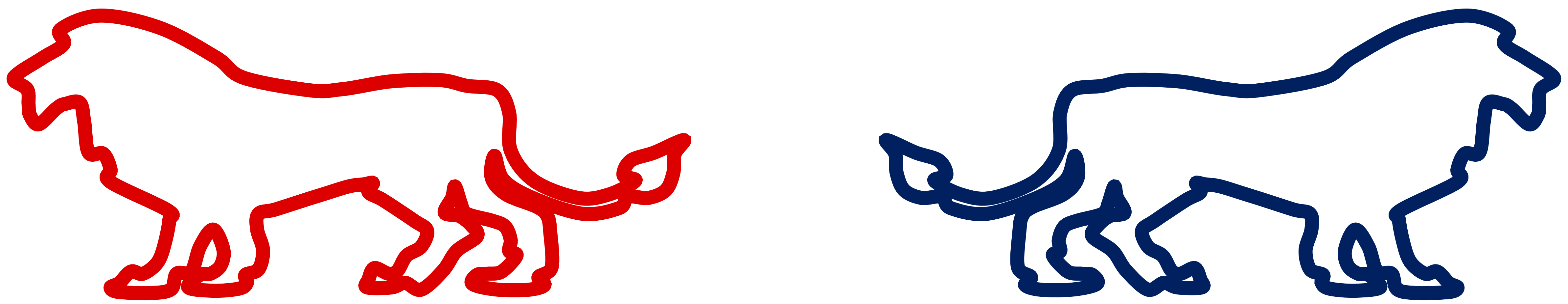
- Compute the "covariance matrix" $\mathbf{V}\mathbf{V}'^T$

- Compute its SVD: $\mathbf{V}\mathbf{V}'^T = \mathbf{U}\boldsymbol{\Sigma}\tilde{\mathbf{U}}^T$

- The optimal orthonormal **R** is $\mathbf{R} = \tilde{\mathbf{U}}\mathbf{U}^T$

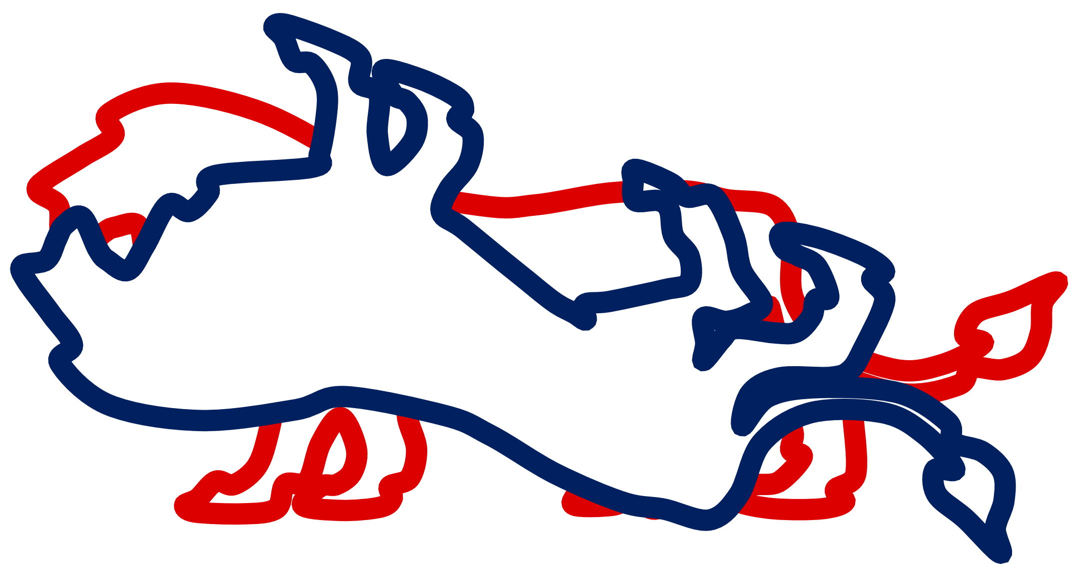
Sign Correction

- It is possible that $\det(\tilde{\mathbf{U}}\mathbf{U}^T) = -1$: sometimes reflection is the best orthonormal transform



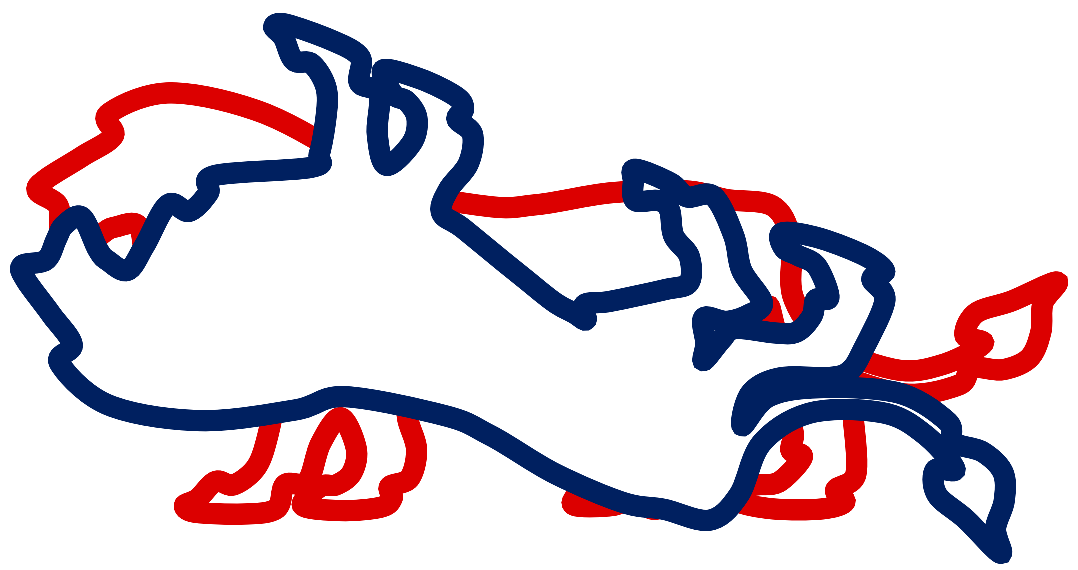
Sign Correction

- It is possible that $\det(\tilde{\mathbf{U}}\mathbf{U}^T) = -1$: sometimes reflection is the best orthonormal transform



Sign Correction

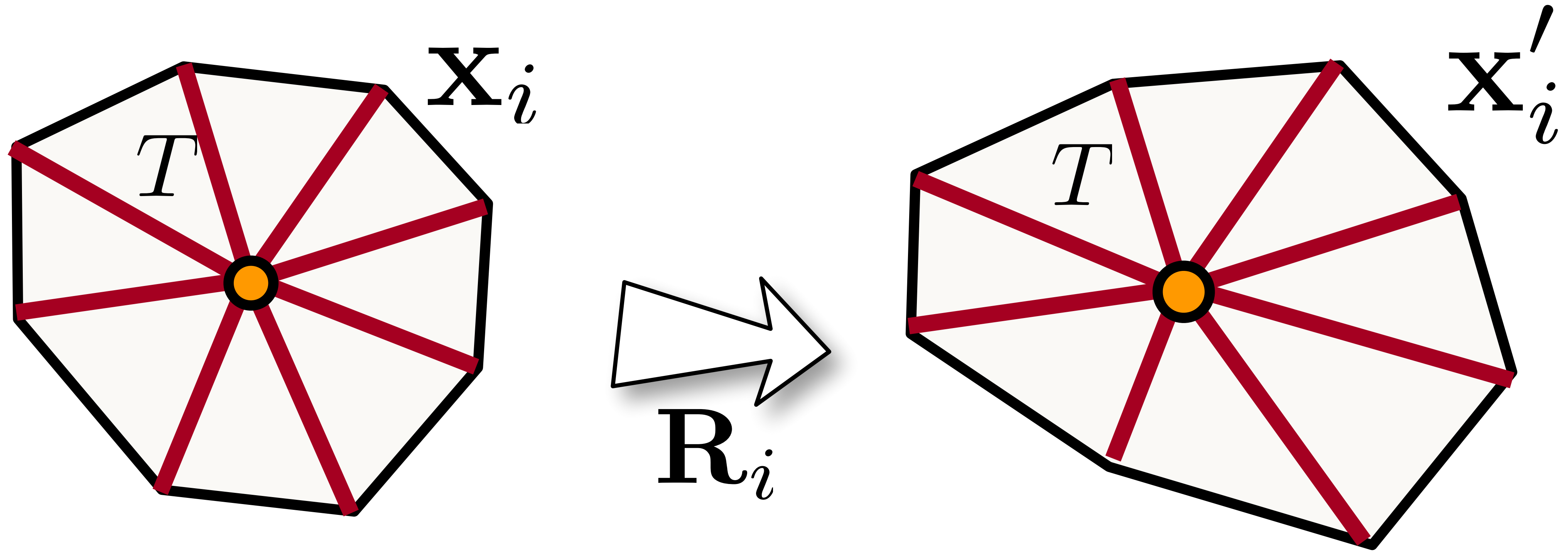
- To restrict ourselves to rotations only:
  take the last column of **U** (corresponding to the smallest singular value) and invert its sign.



- Why? See http://igl.ethz.ch/projects/ARAP/svd_rot.pdf

As-Rigid-As-Possible Deformation

- Optimal $\mathbf{R}_i$ is uniquely defined by $\mathbf{x}_i$ $\mathbf{x}'_i$



$$\min \sum_{T \in \text{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

- so-called shape-matching problem, solved by a 3x3 SVD

$\boxed{\mathbf{R}_i \text{ is a nonlinear function of } \mathbf{x}}$

As-Rigid-As-Possible Deformation

- Total ARAP energy: sum up for all the cells *i*

$$\sum_i \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

- Treat **x** and **R** as separate sets of variables
- Simple **local-global** iterative optimization process
  - Decreases the energy at each step

As-Rigid-As-Possible Deformation

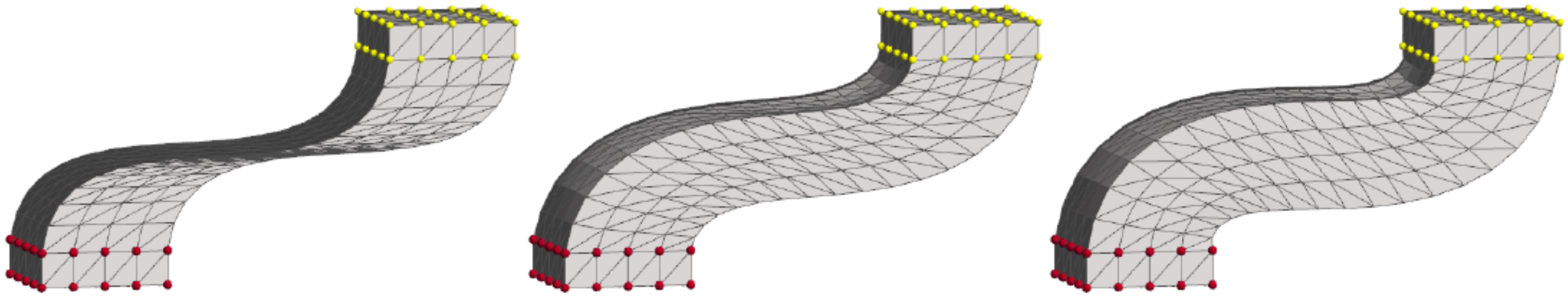
- Total ARAP energy: sum up for all the cells *i*

$$\sum_i \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \left\| (\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i (\mathbf{x}_j - \mathbf{x}_k) \right\|^2$$

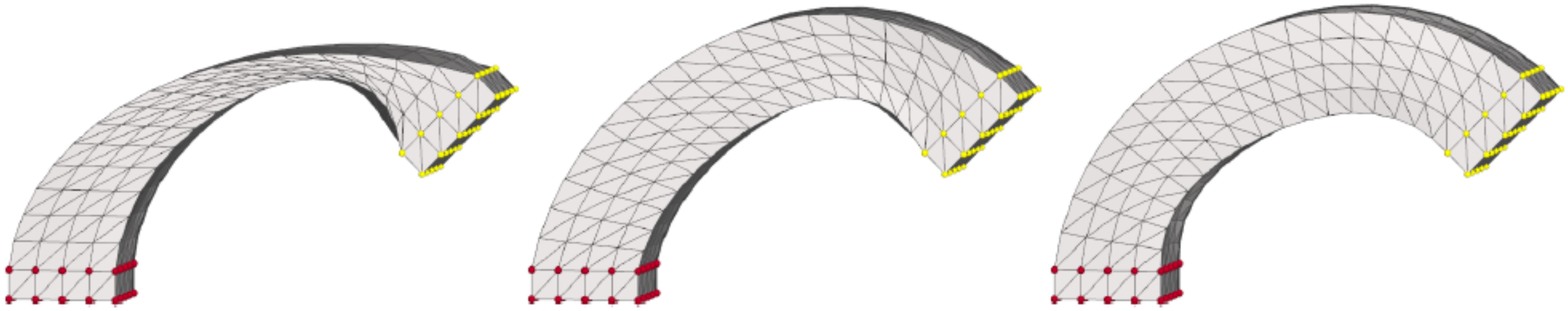
- Local step: keep **x'** fixed, find optimal $\mathbf{R}_i$ per cell *i*
- Global step: keep $\mathbf{R}_i$ fixed, solve for **x'** ➡ $\mathbf{L}\mathbf{x}' = \mathbf{b}$
quadratic minimization problem
  - The matrix **L** stays fixed, can pre-factorize

Initial Guess

- Can use naïve Laplacian editing



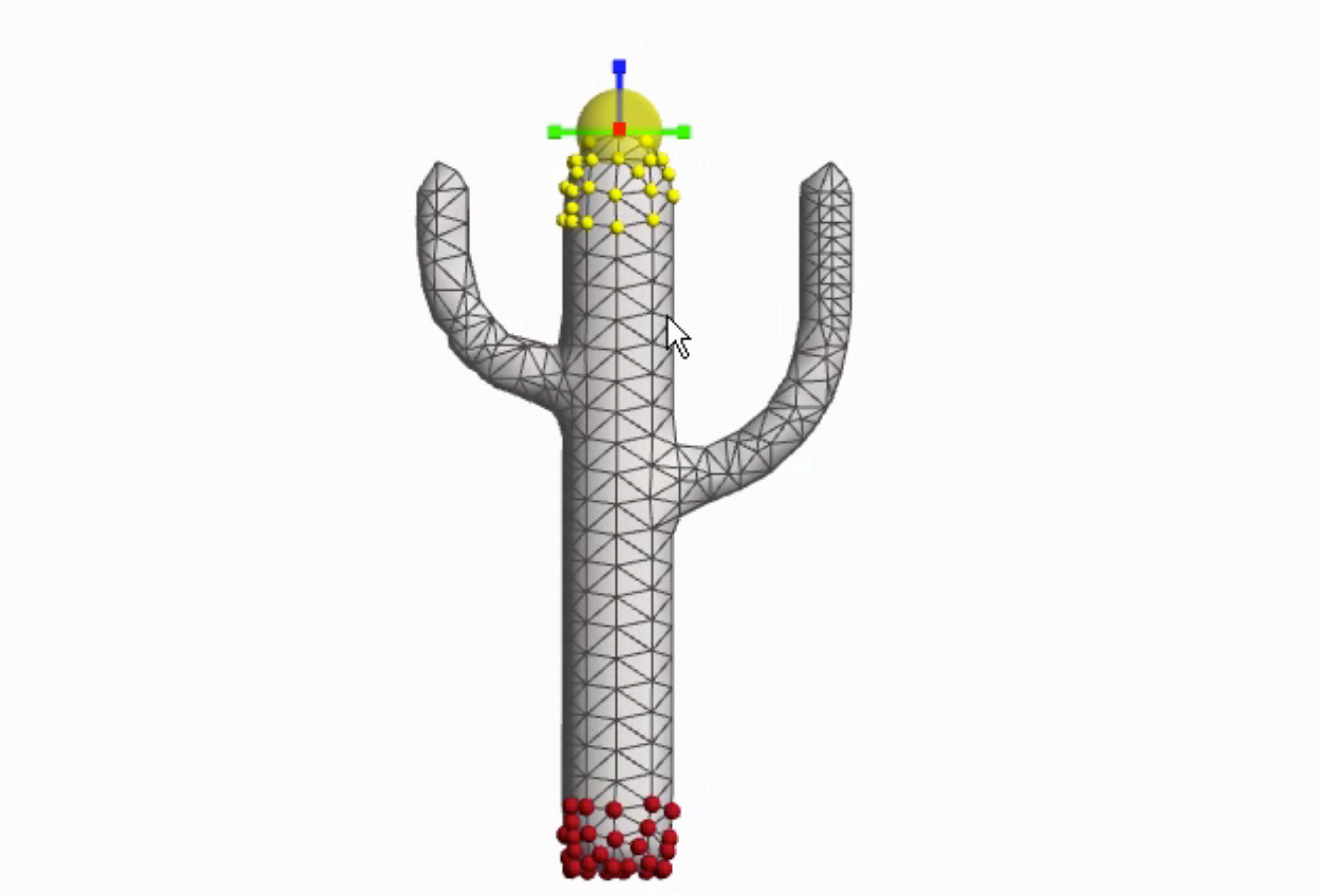
initial guess | 1 iteration | 2 iterations



initial guess | 1 iterations | 4 iterations

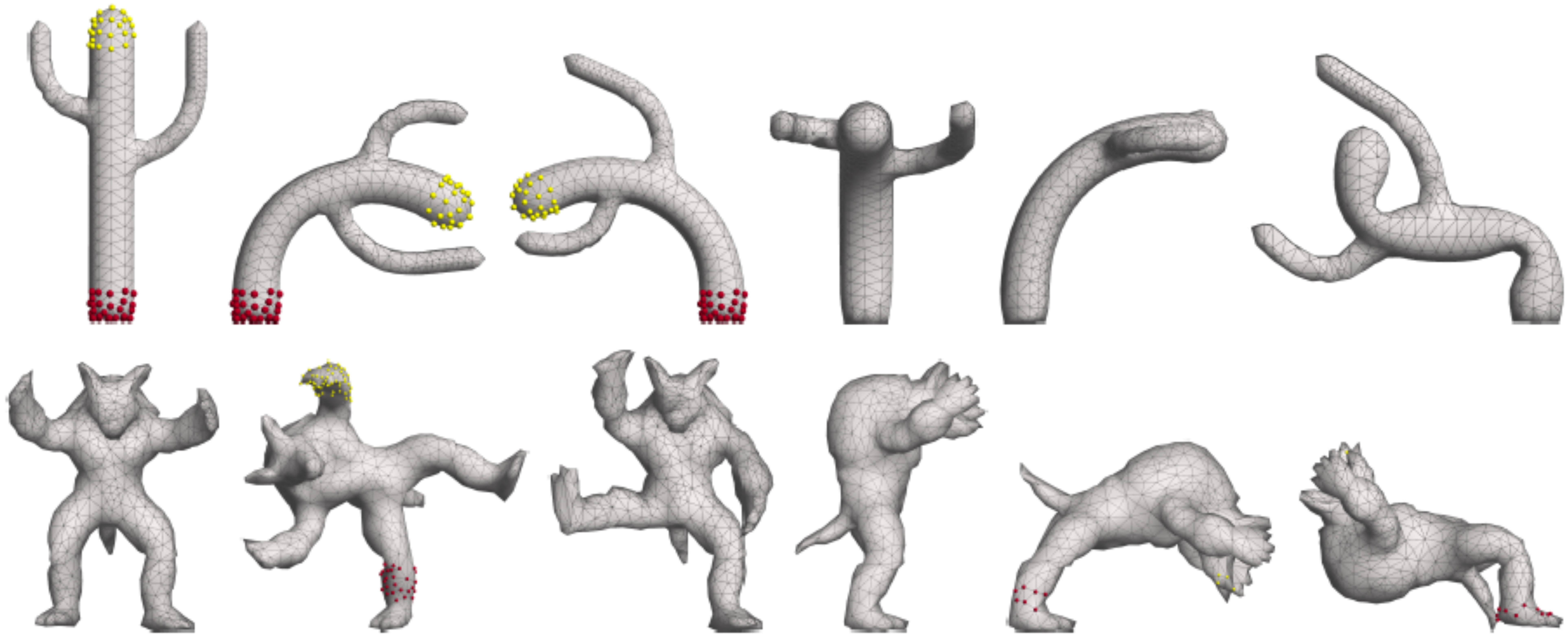
Initial Guess

- Can also use the previous frame
- Replace all handle vertex positions by the currently prescribed ones
- Fast convergence

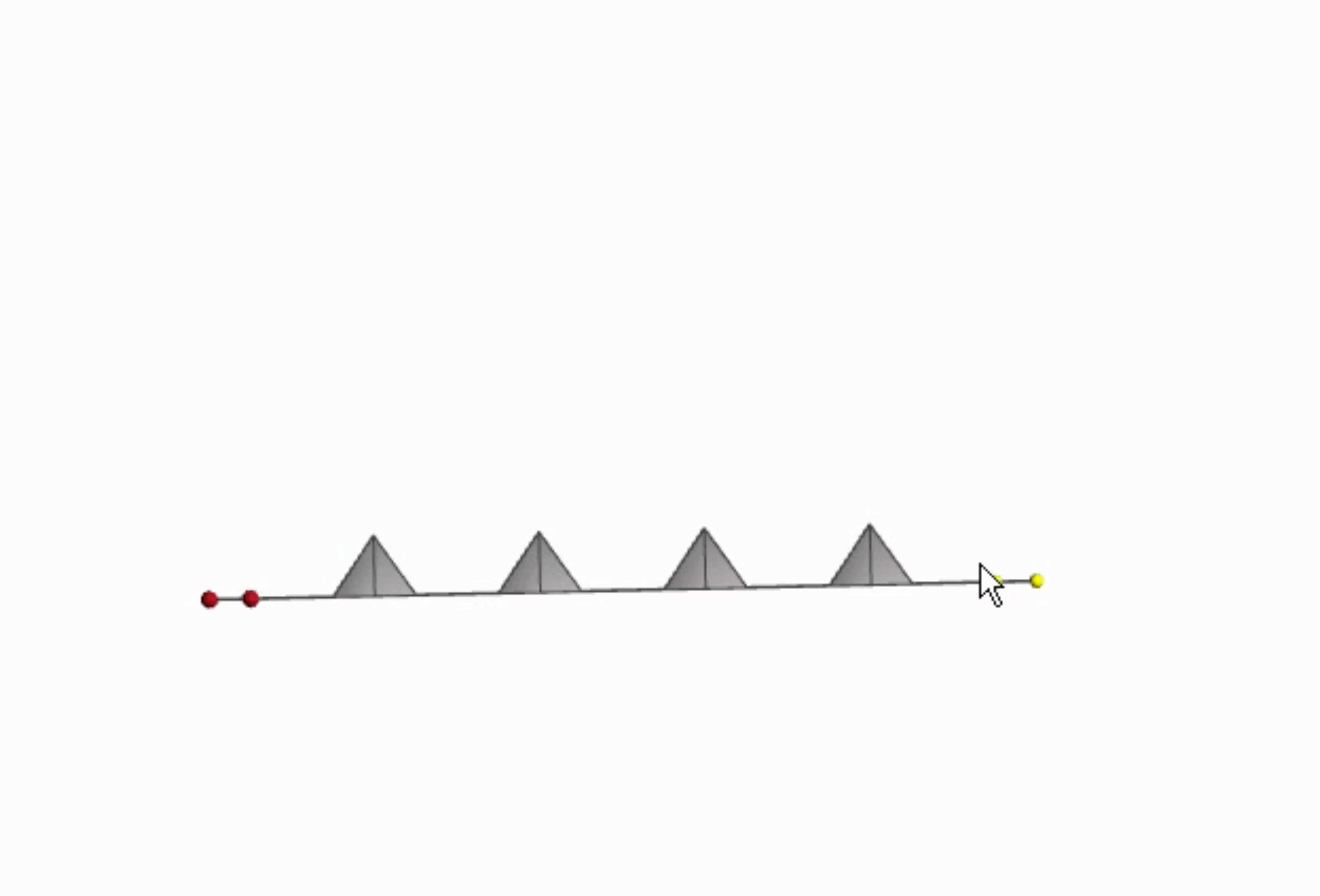


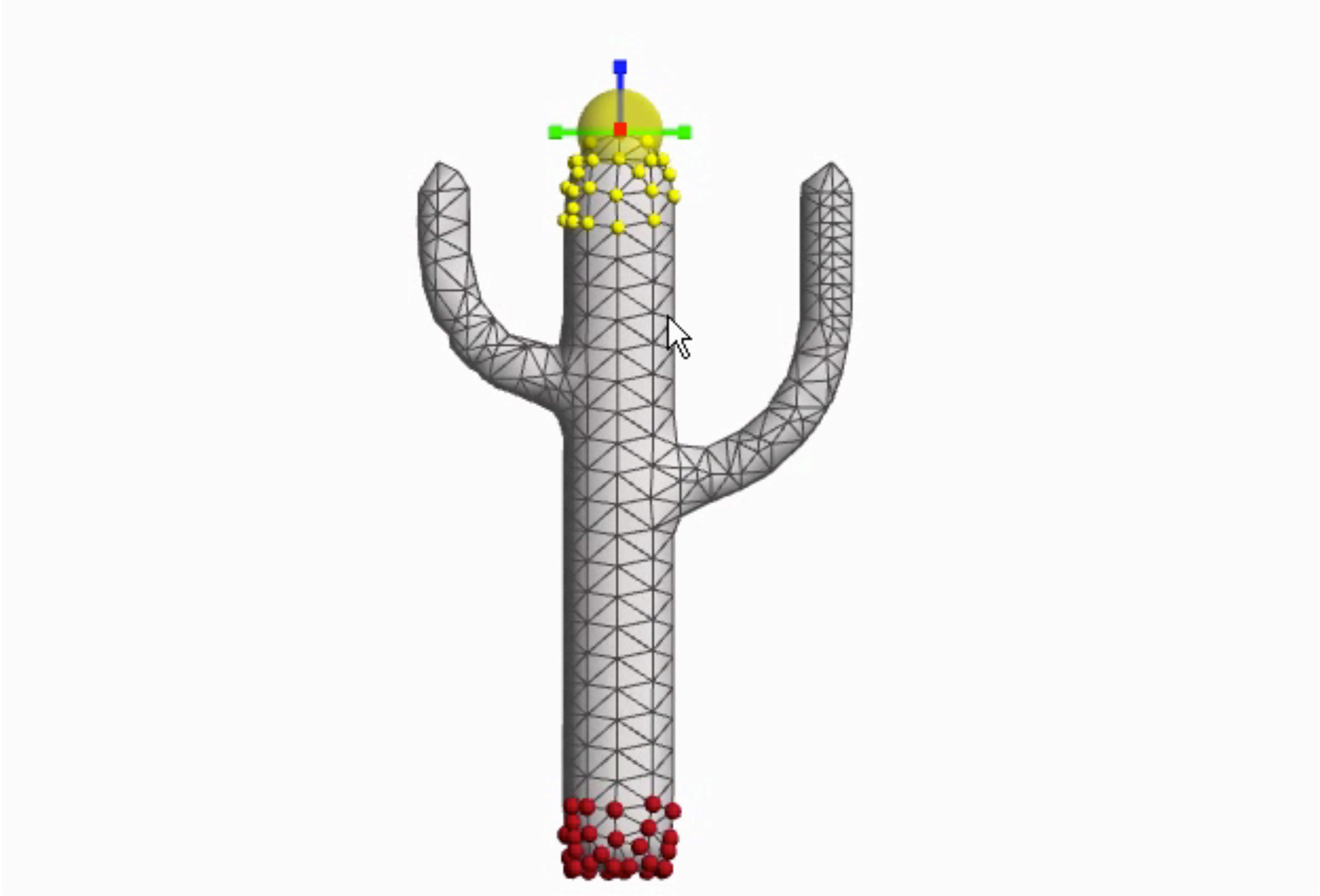
Large Rotations

- Use previous frame as the initial guess



Examples





Discussion

- Nonlinear deformation that models a elastic shell
- Simple to implement, no parameters to tune except number of iterations
- Each step is guaranteed to not increase the energy
- Each iteration is relatively cheap, no matrix re-factorization necessary

Discussion

- Works fine on small meshes
- On larger meshes: slow convergence
  - Each iteration is more expensive
  - Need more iterations because the conditioning of the system becomes worse as the matrix grows
- Material stiffness depends on the cell size
  - lots of wrinkles for fine meshes when using 1-rings as cells

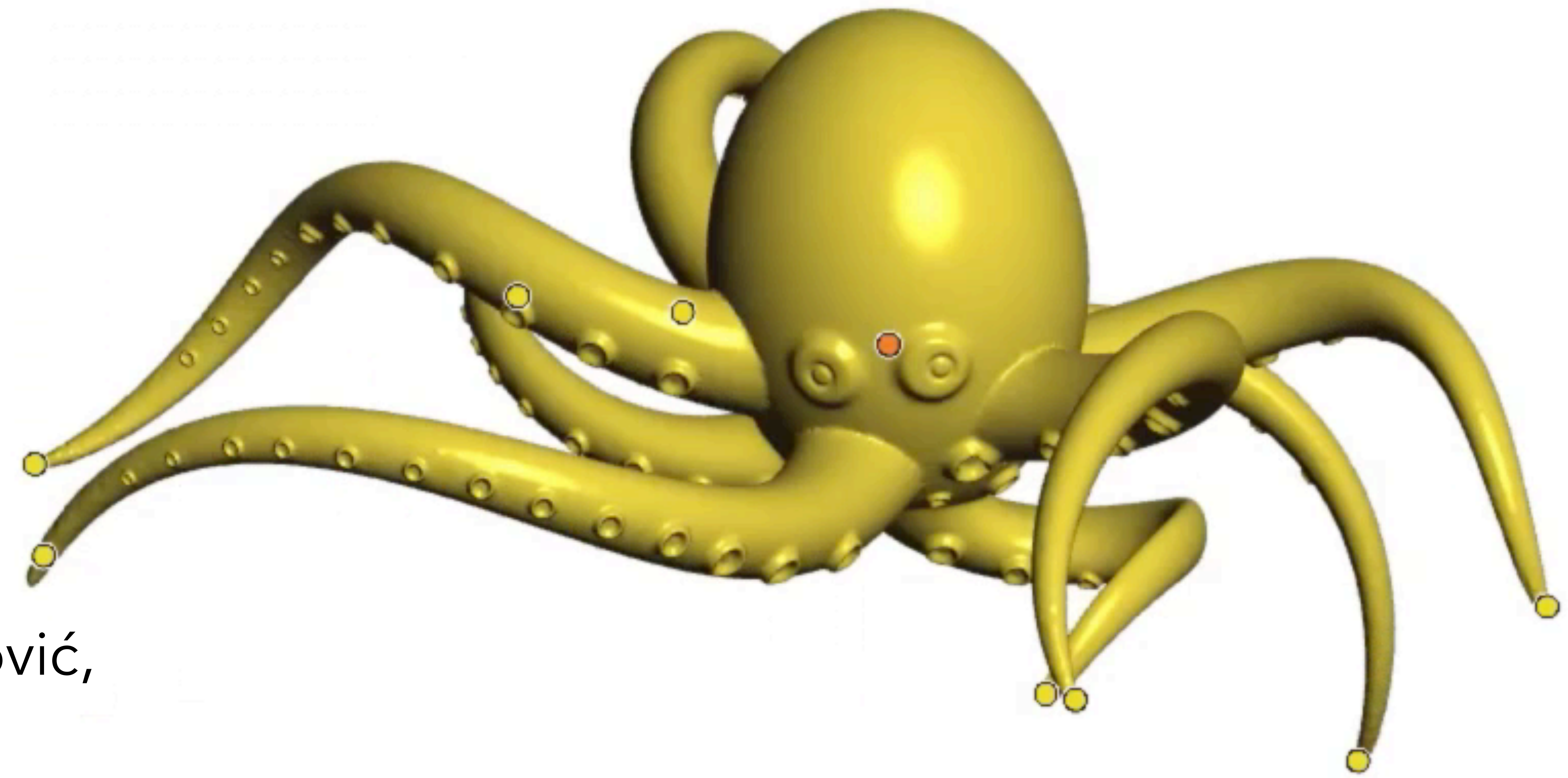
Acceleration using Subspace Techniques

- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. "Fast Automatic Skinning Transformations," 2012.

Acceleration using Subspace Techniques

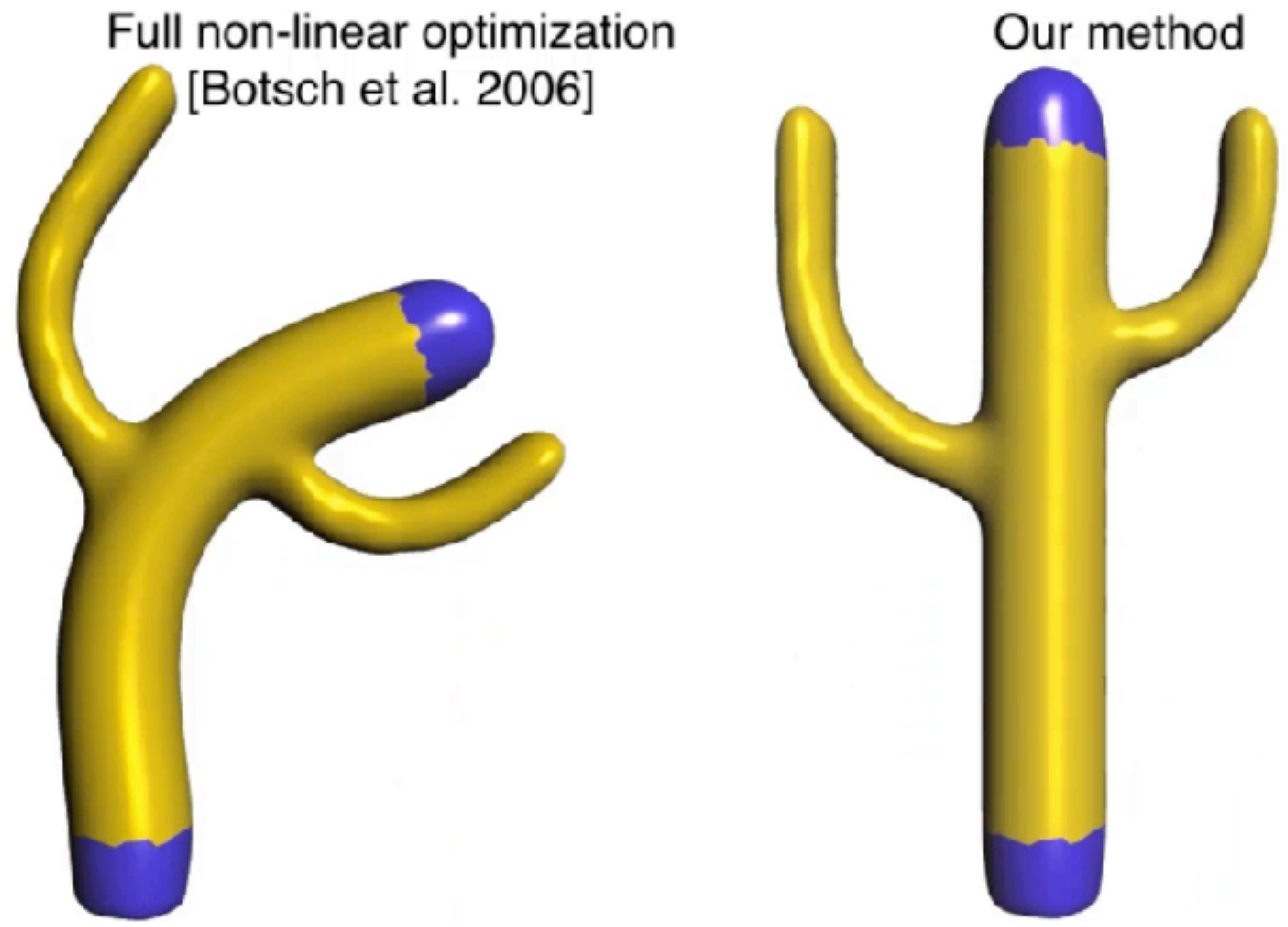
- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization



Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. "Fast Automatic Skinning Transformations," 2012.

Acceleration using Subspace Techniques

- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization



Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. [“Fast Automatic Skinning Transformations,” 2012.]

Summary on Surface-Based Deformation

Summary on Surface-Based Deformation

- Find a mesh that optimizes some objective functional and satisfies modeling constraints

$$\mathbf{x}_{\text{def}} = \underset{\mathbf{x}'}{\operatorname{argmin}}\, E(\mathbf{x}') \quad s.t. \quad \mathbf{x}'_i = \mathbf{c}_i$$

Linear Methods

- Triangle gradient methods (2004-2005)

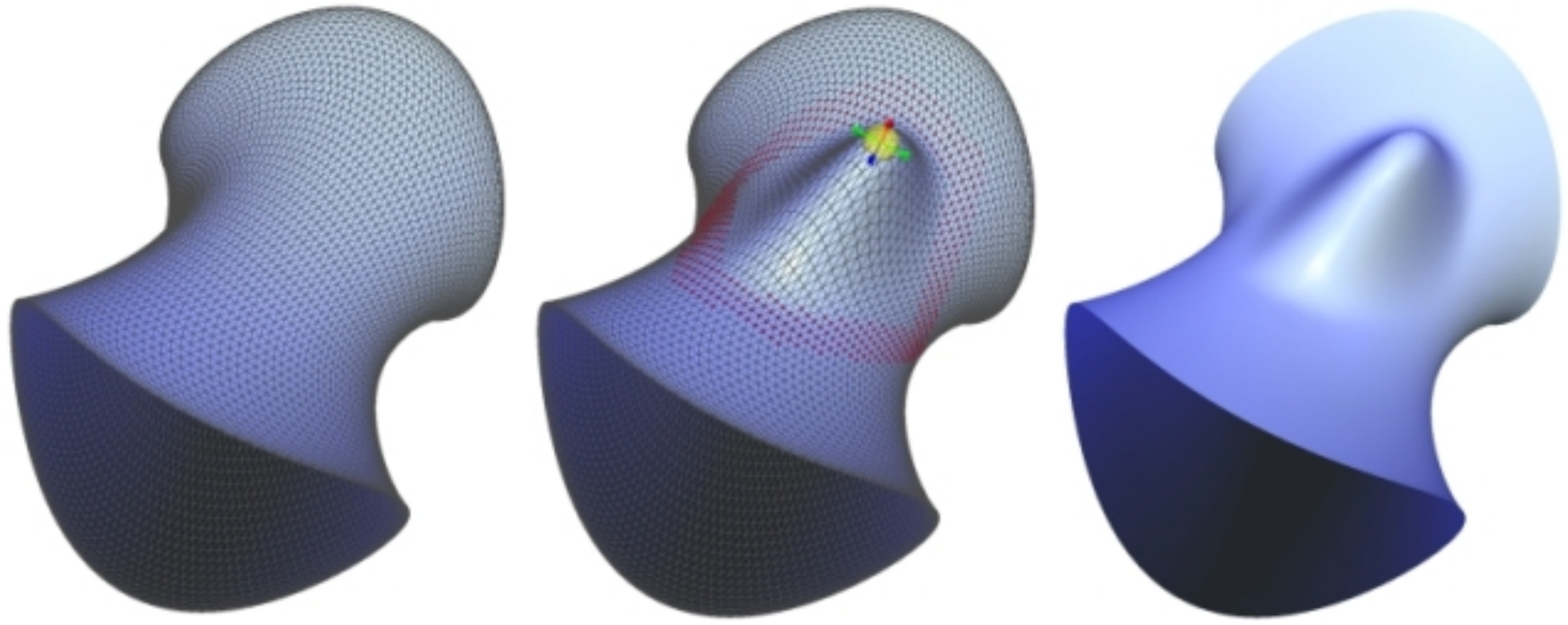
  http://dl.acm.org/citation.cfm?id=1015774

  http://www-ui.is.s.u-tokyo.ac.jp/~takeo/research/rigid/

- Laplacian surface editing (2004-2005)

  http://dl.acm.org/citation.cfm?id=1015772

  http://igl.ethz.ch/projects/Laplacian-mesh-processing/Laplacian-mesh-editing/



Nonlinear Methods

- As Rigid as Possible surface modeling

  http://igl.ethz.ch/projects/ARAP/

  http://dl.acm.org/citation.cfm?id=1778775

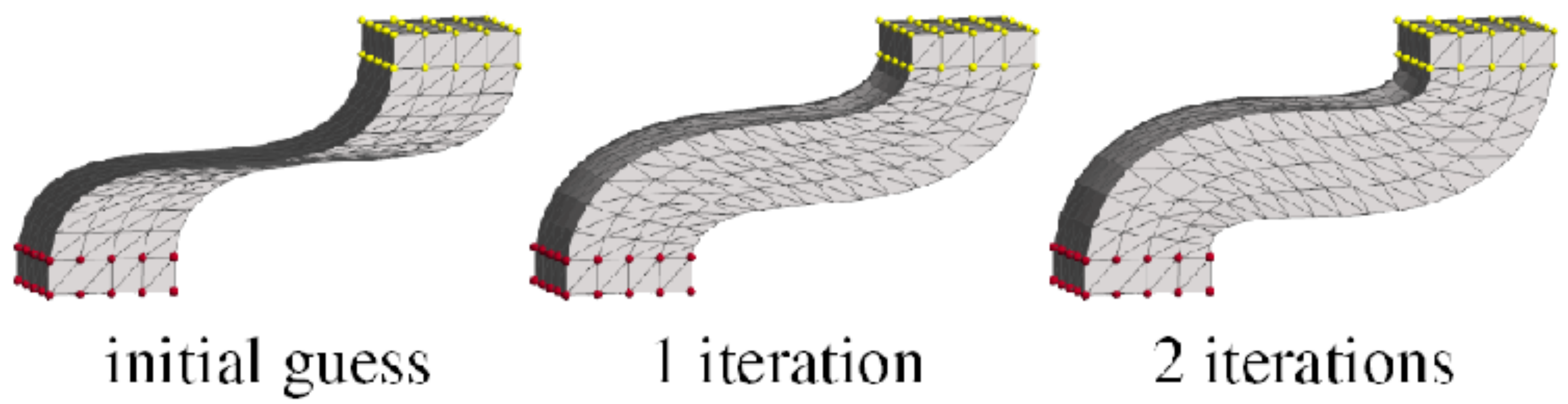
  http://igl.ethz.ch/projects/fast/

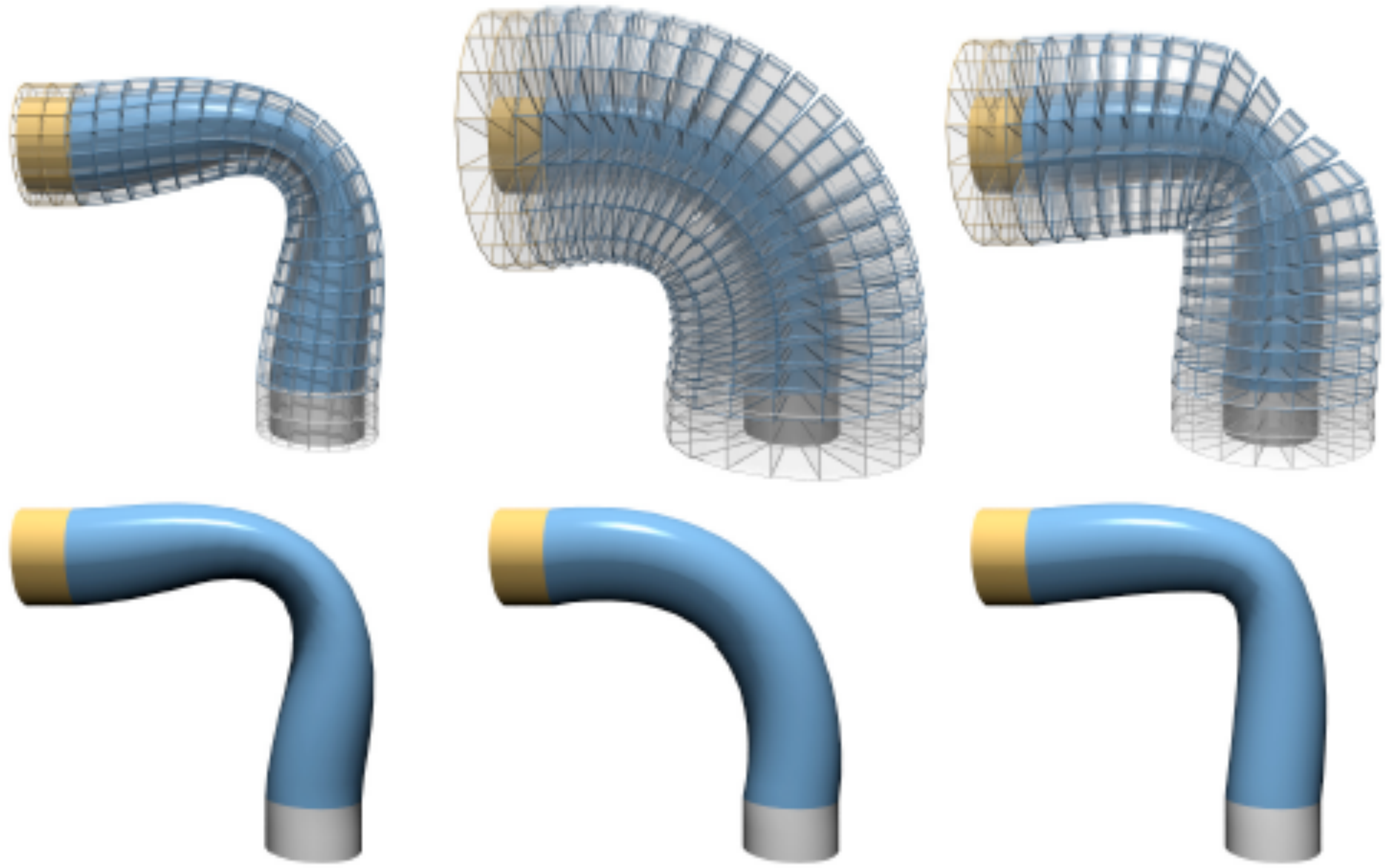
- PriMo

  http://dl.acm.org/citation.cfm?id=1281959

- Mesh Puppetry

  http://dl.acm.org/citation.cfm?id=1275808.1276479





University of Victoria | Computer Science

Surface-based Deformations

- Objective functional expressed in the mesh elements (vertices)
- Complexity depends on the mesh resolution
- Linear methods:
  - Solve a global linear system on the mesh
  - Usually suffer from some artifacts
- Nonlinear methods
  - Fewer artifacts but slower, and harder to implement

References

- Polygon Mesh Processing, Chapter 9