G22.3033-008, Spring 2010

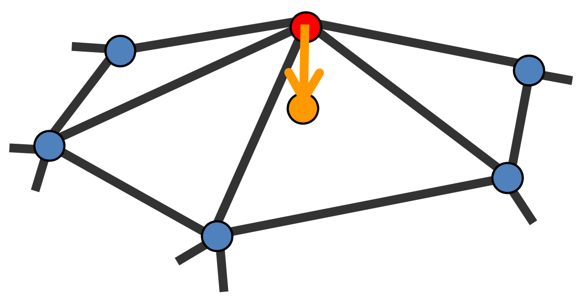
Geometric Modeling

Surface deformation using differential coordinates

Recap

Differential coordinates

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



$$\boldsymbol{\delta}_i = \frac{1}{A_i} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} w_{ij}\left(\mathbf{v}_j - \mathbf{v}_i\right) \approx -H\mathbf{n}$$

Recap

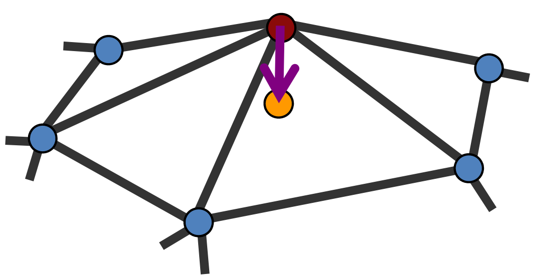
Differential coordinates

- 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

Recap

Laplacian matrix

- The transition between *xyz* and δ is linear:



$$\boldsymbol{\delta}_i = \sum_{j \in N(i)} w_{ij} \left(\mathbf{v}_i - \mathbf{v}_j \right)$$

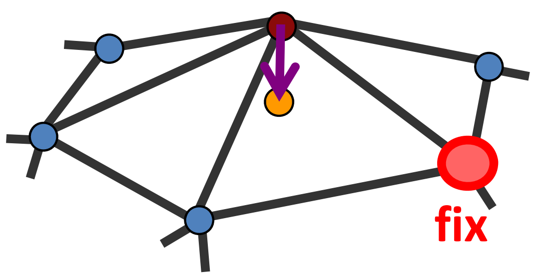
$$\mathbf{L} \; \mathbf{v_x} = \delta_{\mathbf{x}}$$

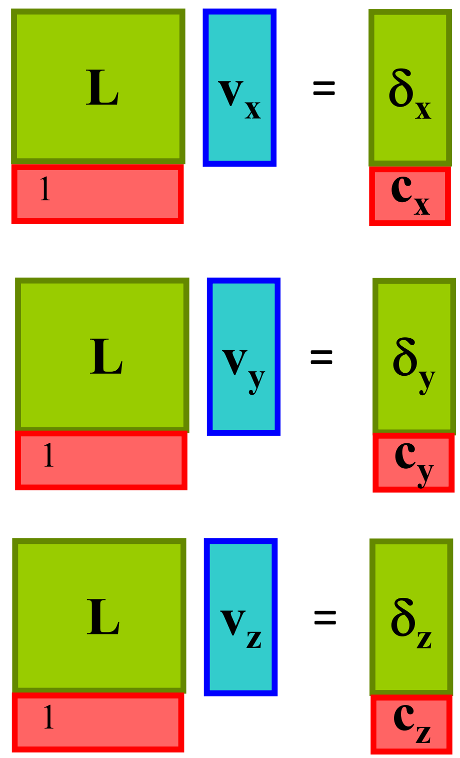
$$\mathbf{L} \; \mathbf{v_y} = \delta_{\mathbf{y}}$$

$$\mathbf{L} \; \mathbf{v_z} = \delta_{\mathbf{z}}$$

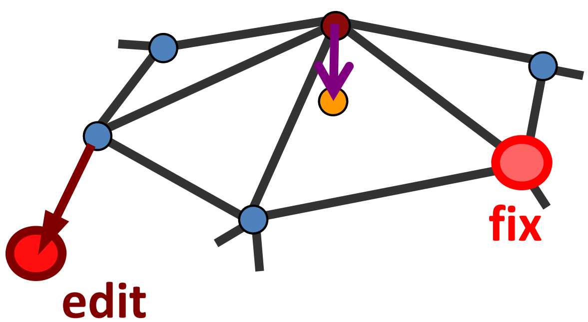
Properties of the Laplacian matrix

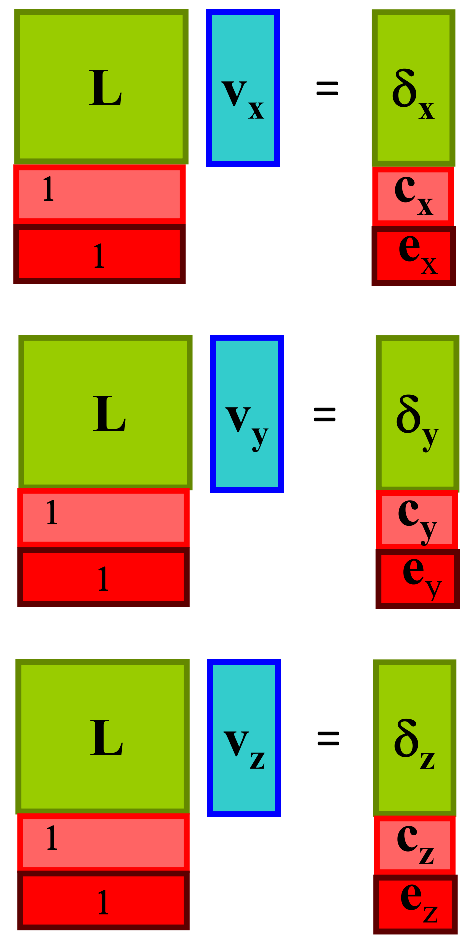
- $\mathrm{rank}(L) = \mathrm{n} - \mathrm{c}$ $(\mathrm{n} - 1$ for connected meshes)
- We can reconstruct the xyz geometry from δ ***up to translation***



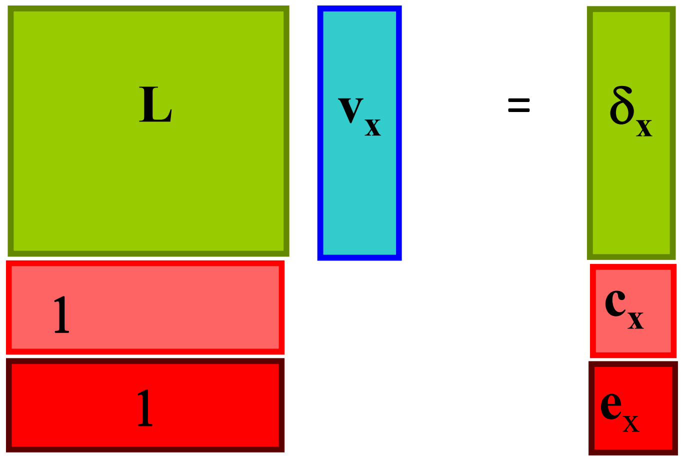


Reconstruction





Reconstruction



$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\left\| L\mathbf{x} - \boldsymbol{\delta}_x \right\|^2 + \sum_{s=1}^{k} \left| x_k - c_k \right|^2 \right)$$

… and the same for y and z

Reconstruction

$$\begin{bmatrix} \mathbf{L} \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{v_x} \end{bmatrix} = \begin{bmatrix} \delta_x \\ \mathbf{c_x} \\ \mathbf{e_x} \end{bmatrix}$$

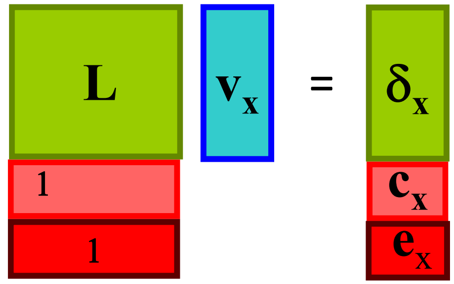
$$\mathbf{A} \quad \mathbf{x} = \mathbf{b}$$

Normal Equations:

$$\mathbf{A^T A} \quad \mathbf{x} = \mathbf{A^T b}$$

$$\mathbf{x} = \underbrace{(\mathbf{A^T A})^{-1} \; \mathbf{A^T}}_{\text{compute once}} \mathbf{b}$$

Details I left out



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

Normal Equations:

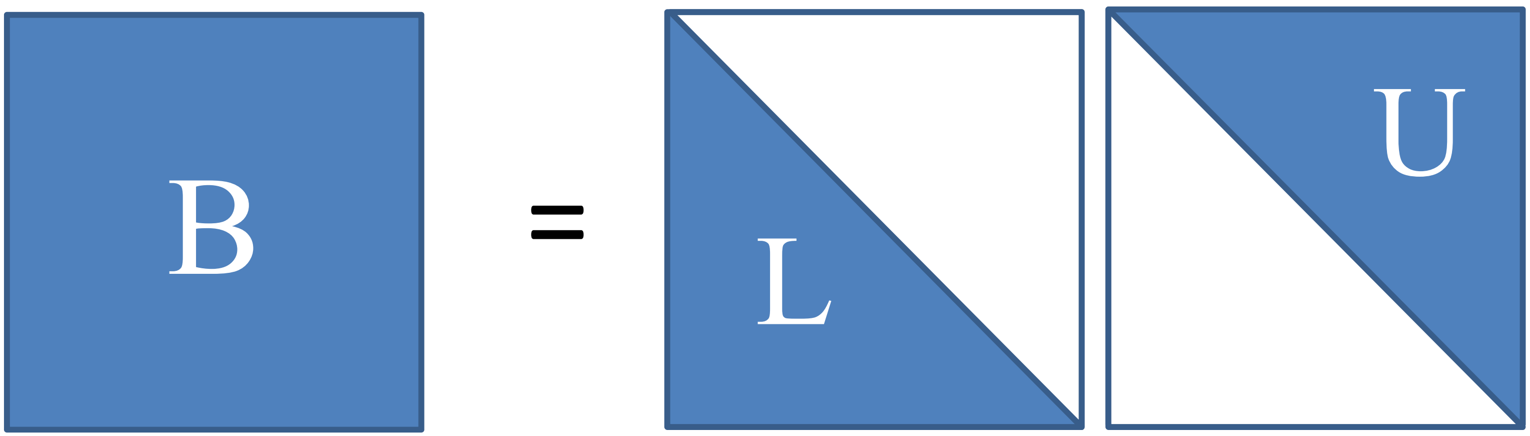
$$\mathbf{A}^T\mathbf{A} \quad \mathbf{x} \quad = \quad \mathbf{A}^T \mathbf{b}$$

$$\mathbf{x} \quad = \quad \underbrace{(\mathbf{A}^T\mathbf{A})^{-1} \quad \mathbf{A}^T}\mathbf{b}$$

Actually, we won't compute the inverse (dense matrix, expensive). Instead we will factor $\mathbf{A}^T\mathbf{A} = \mathbf{M}\mathbf{M}^T$, $\mathbf{M}$ is sparse and *triangular*

Matrix factorization

LU decomposition



$$\mathrm{B}\mathbf{x} = \mathbf{b}$$
$$\mathrm{L}(\mathrm{U}\mathbf{x}) = \mathbf{b}$$

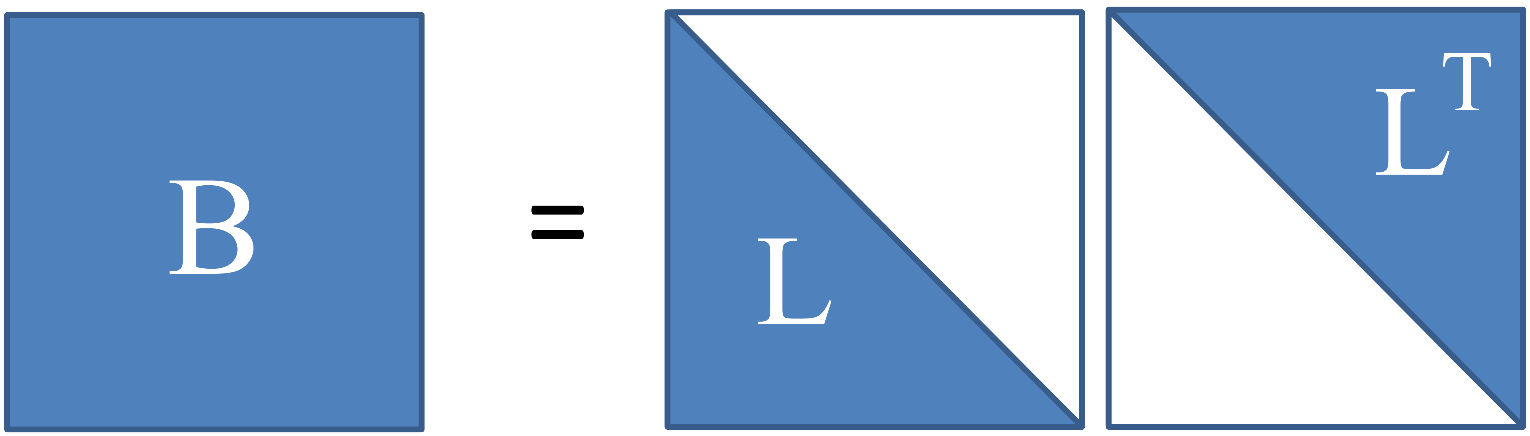
$$\Longrightarrow$$

$$\left.\begin{aligned} \mathrm{L}\mathbf{y} &= \mathbf{b} \\ \mathrm{U}\mathbf{x} &= \mathbf{y} \end{aligned}\right\}$$

This is backsubstitution. If L, U are sparse it is very fast. The hard work is computing L and U

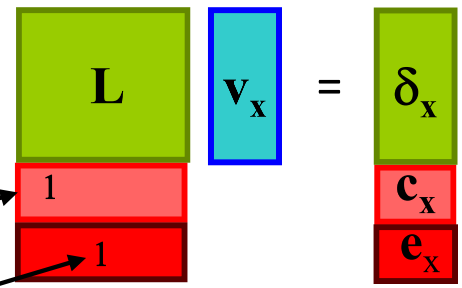
Matrix factorization

Cholesky decomposition



Cholesky factor exists if B is positive definite. It is even better than LU because we save memory.

Details I left out



These should actually be high weights to ensure interpolation of the constraints. Or better yet, we can substitute the constraints directly into the LS system

$$A \quad x \quad = \quad b$$

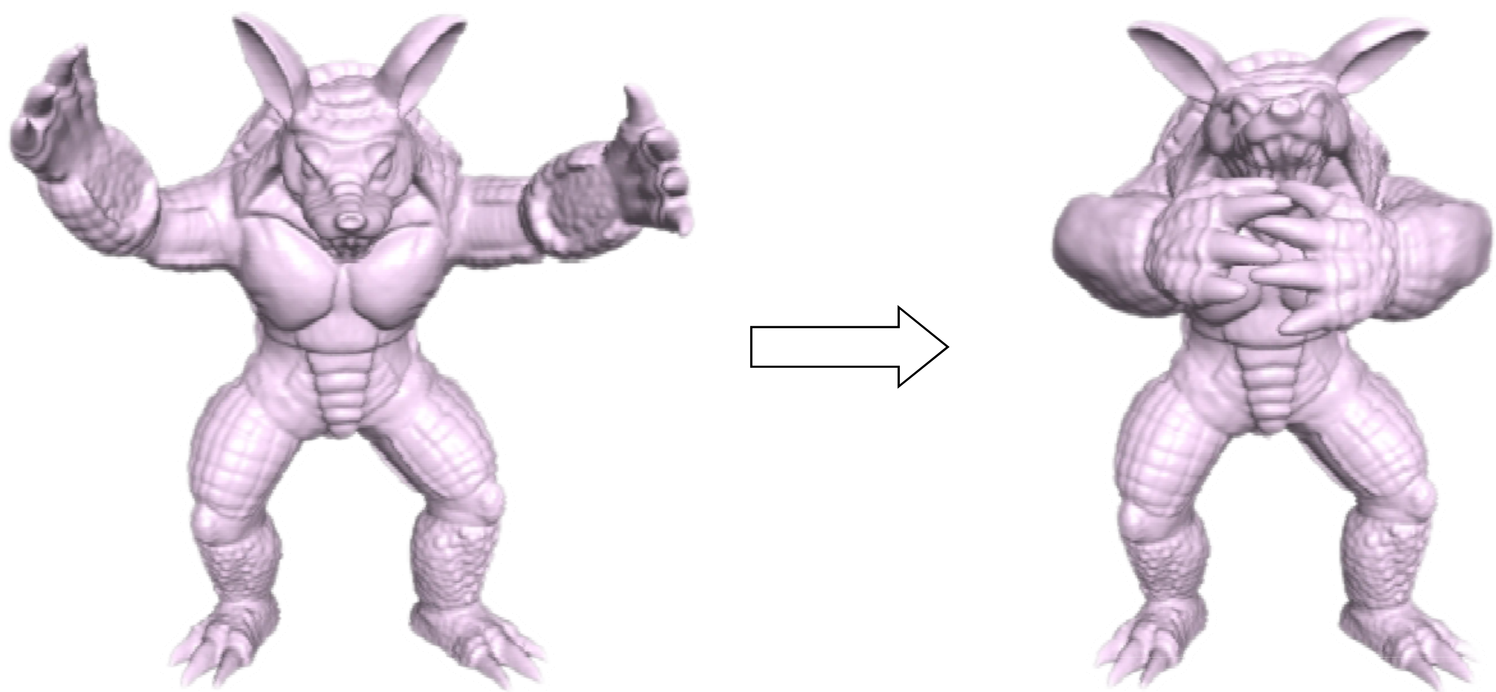
Normal Equations:

$$A^T A \; x = A^T b$$

$$x = (A^T A)^{-1} \; A^T b$$

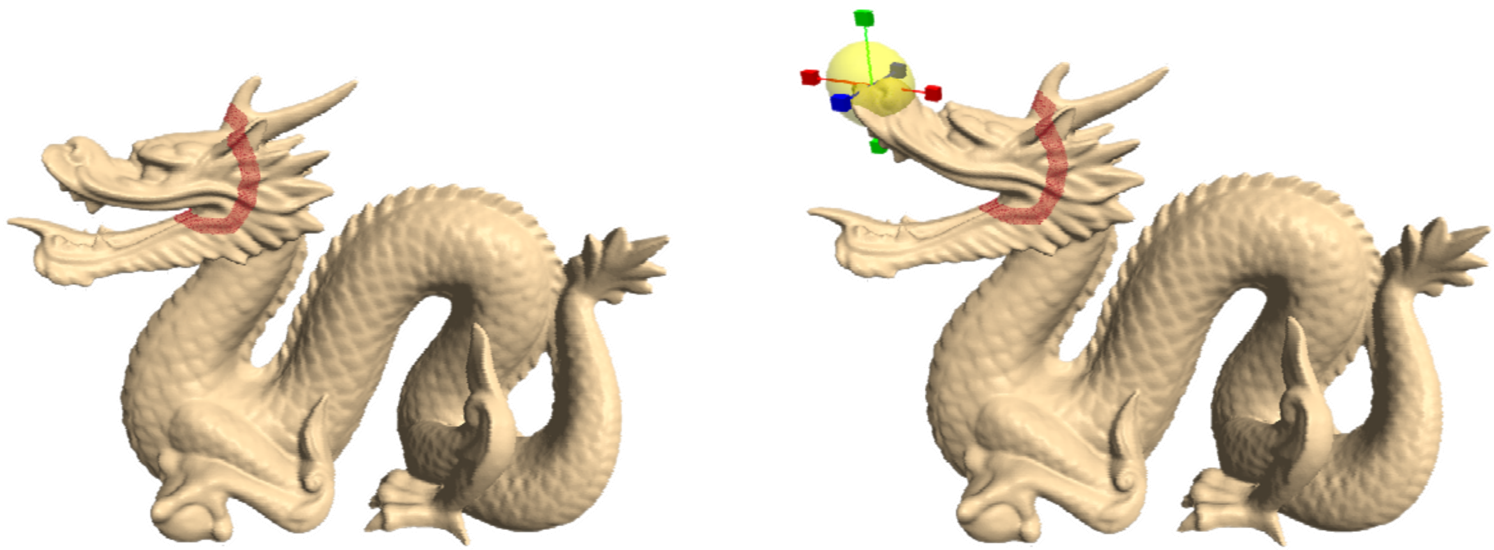
Differential coordinates for editing

- Intrinsic surface representation
- Allows various surface editing operations that preserve local surface details (normals, mean curvature)



Why differential coordinates?

- Local detail representation – enables detail preservation through various modeling tasks
- Representation with sparse matrices
- Efficient linear reconstruction

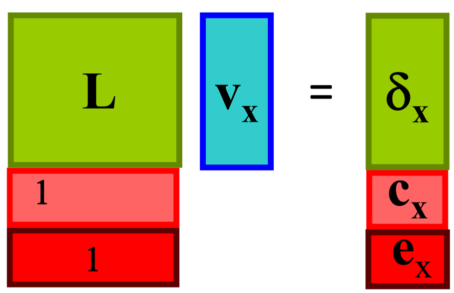


Editing framework

- The spatial constraints will serve as modeling constraints
- Solve the reconstruction equation every time the modeling constraints are changed

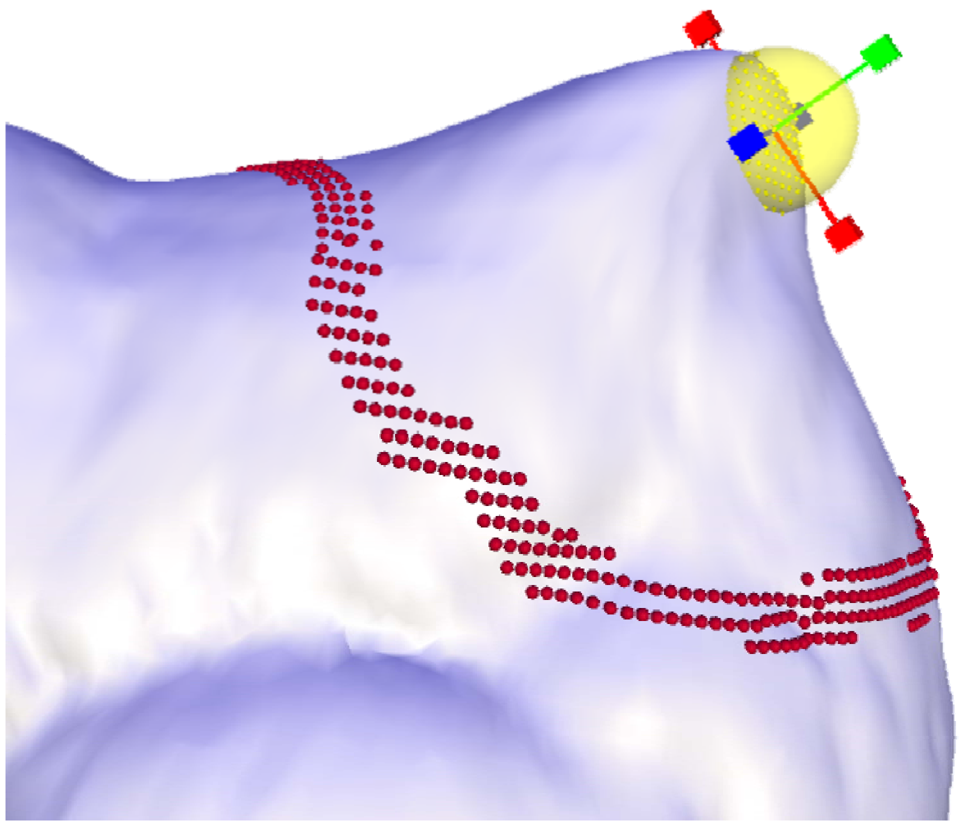
Detail constraints: $L\mathbf{x} = \boldsymbol{\delta}$

Modeling constraints: $x_j = c_j, \quad j \in \{j_1, j_2, \ldots j_k\}$



Editing framework

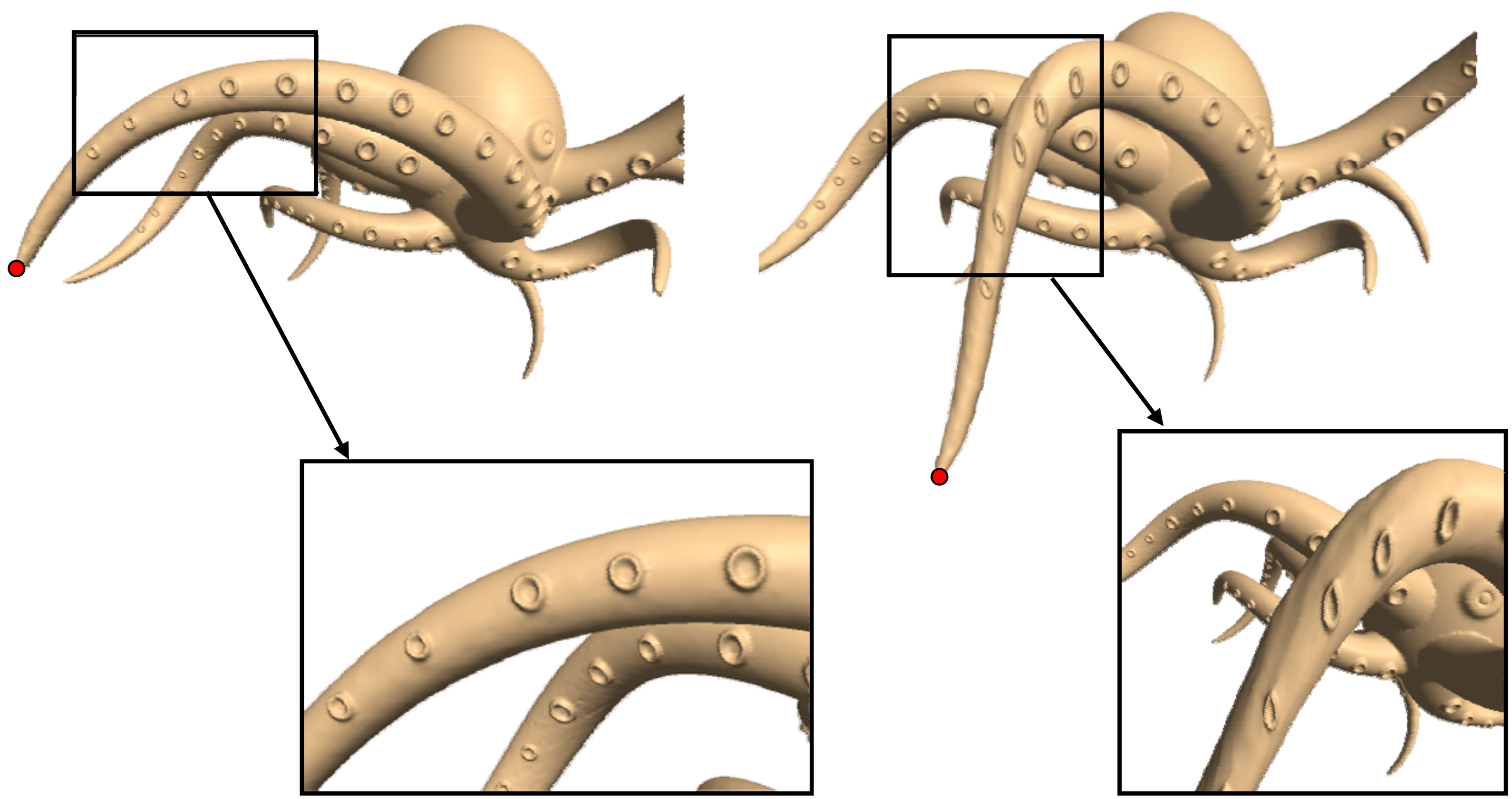
- ROI is bounded by a belt (static anchors)
- Manipulation through handle(s)



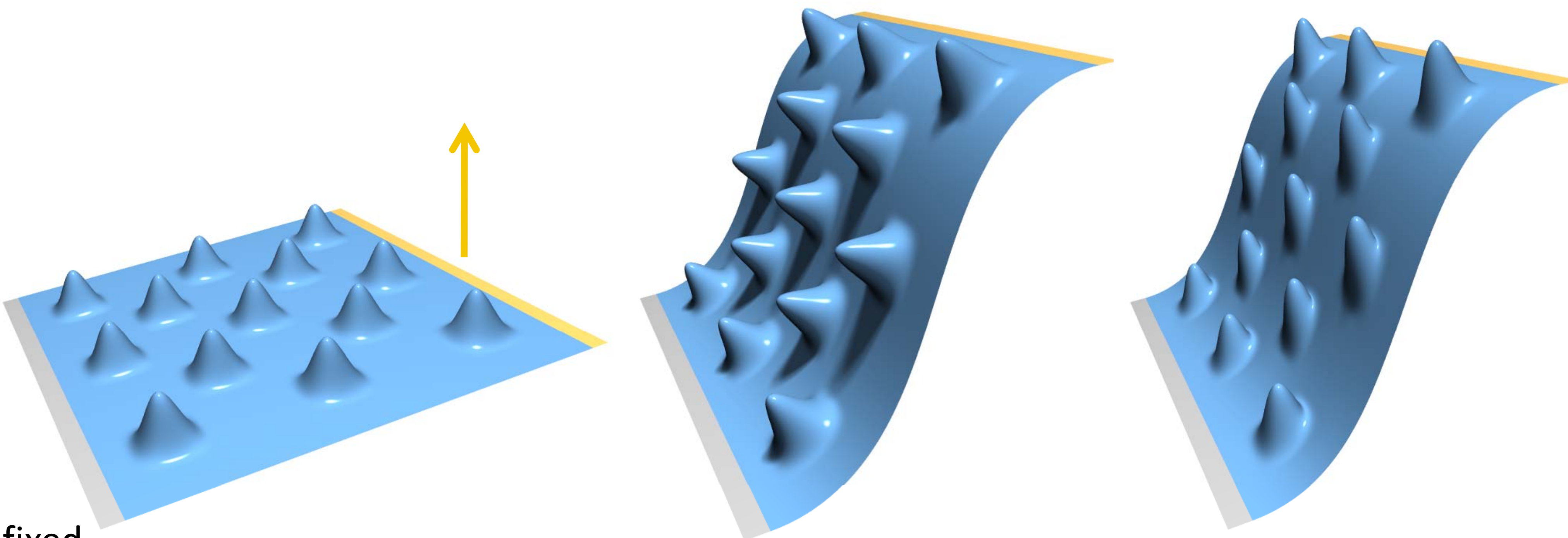
Fundamental problem: invariance to transformations

$$\boxed{\Delta_M \mathbf{p} = -H\mathbf{n}}$$

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



Fundamental problem: invariance to transformations

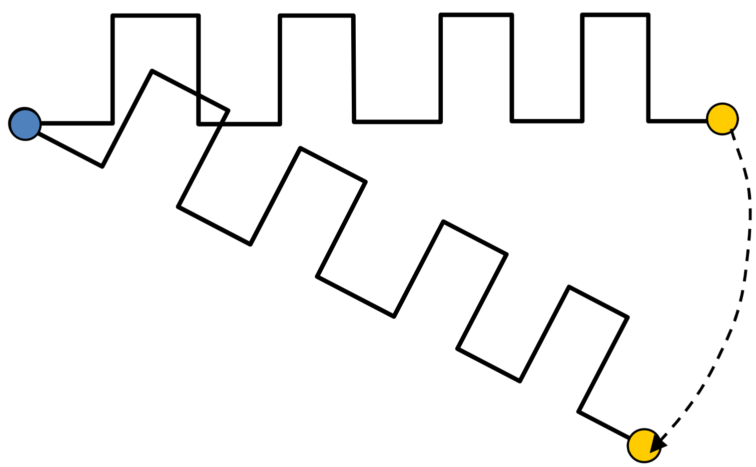


fixed vertices

input | intuitive expectation | actual result

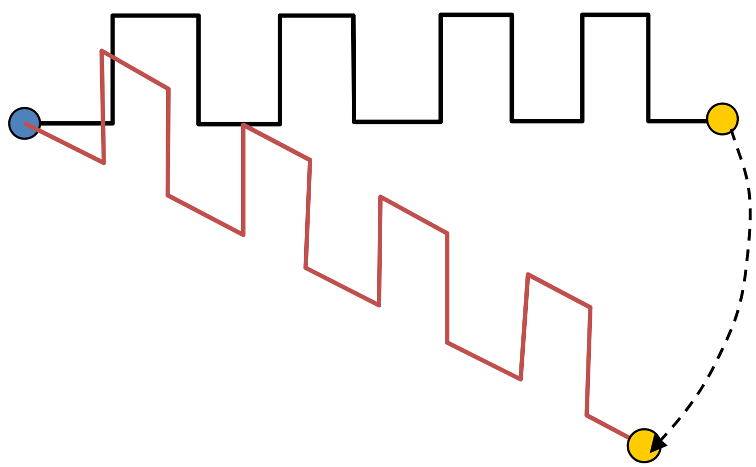
Fundamental problem: invariance to transformations

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



Fundamental problem: invariance to transformations

- Similar problem with the Great Wall of China...

Energy functional

- We posed this minimization problem (under handle constraints):

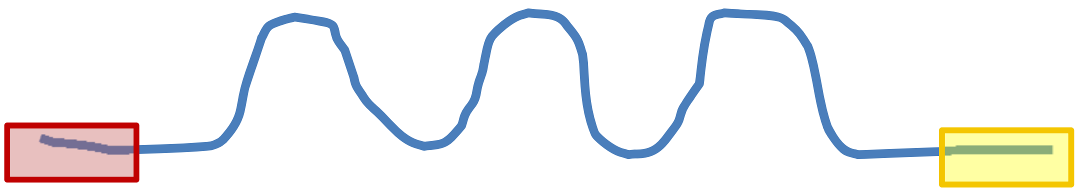
$$\arg\min_{\mathbf{x}} \left\| \Delta\mathbf{x} - \Delta\mathbf{x}_{org} \right\|^2$$

- But the rotated version of the original shape *is not a minimizer*. Need a rigid-invariant energy!

Fixing local rotations

Multiresolution framework

input

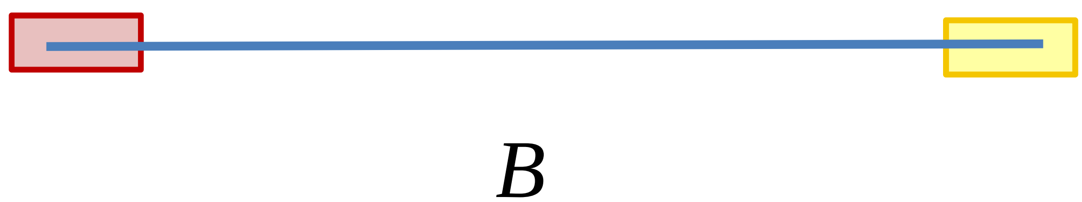


S

Fixing local rotations

Multiresolution framework

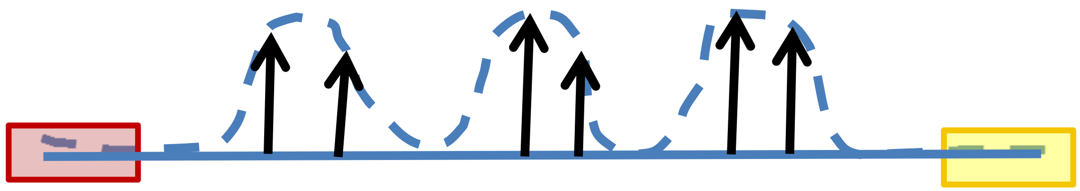
Smooth base surface



Fixing local rotations

Multiresolution framework

Details – displacement vectors

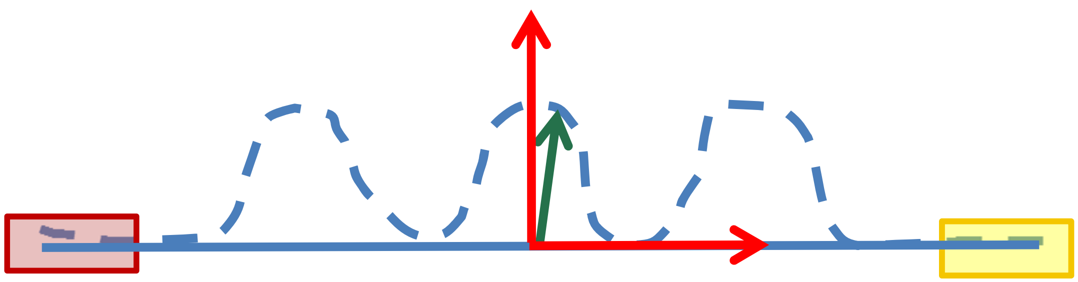


$$S - B$$

Fixing local rotations

Multiresolution framework

Encode details in the local frame of B

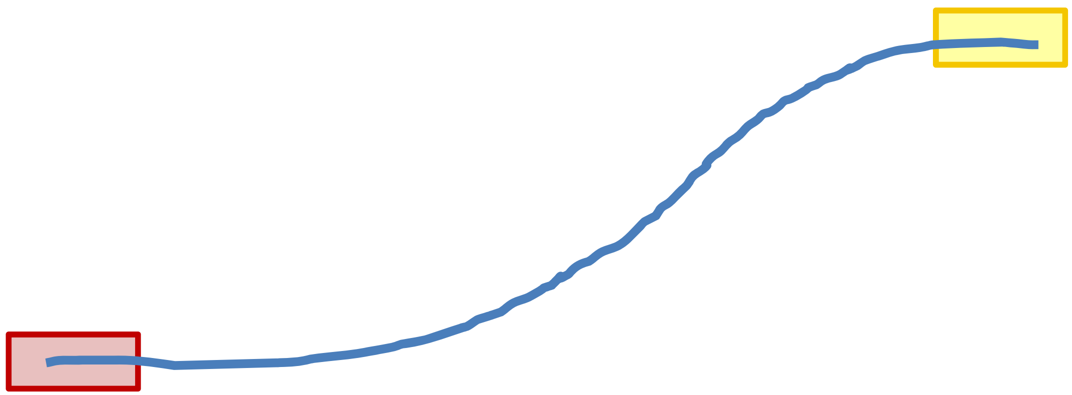


$$\mathbf{d}_i = a_1 \mathbf{t}_i + a_2 \mathbf{n}_i$$

Fixing local rotations

Multiresolution framework

Deform smooth base surface

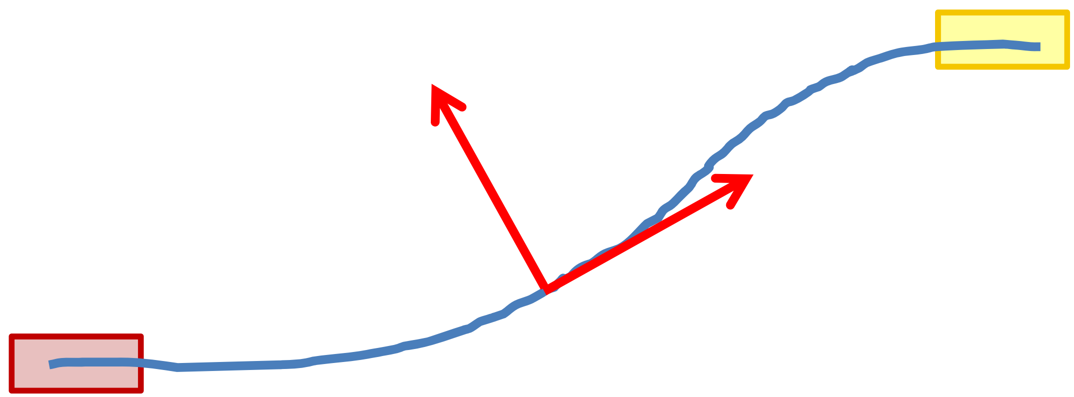


B'

Fixing local rotations

Multiresolution framework

Local frames on B'

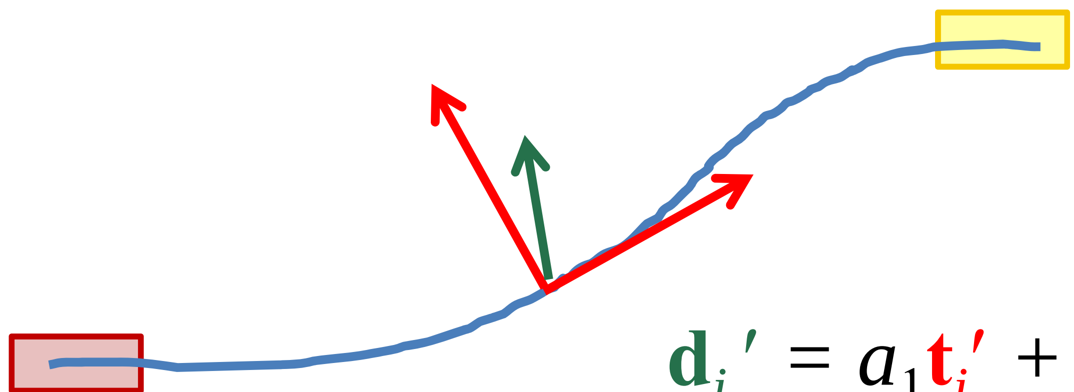


B'

Fixing local rotations

Multiresolution framework

Add details back – in local frame!



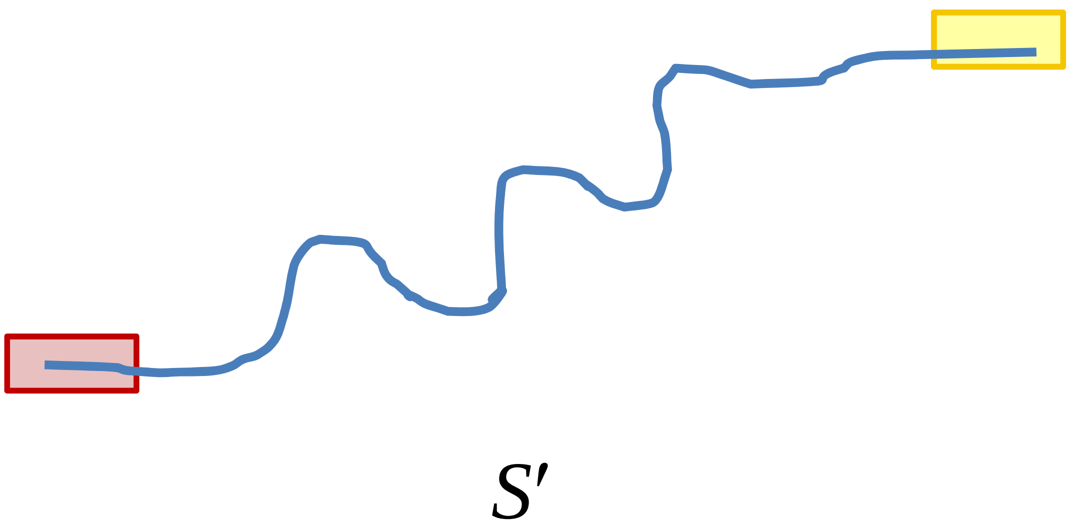
$$\mathbf{d}_i' = a_1\mathbf{t}_i' + a_2\mathbf{n}_i'$$

B'

Fixing local rotations

Multiresolution framework

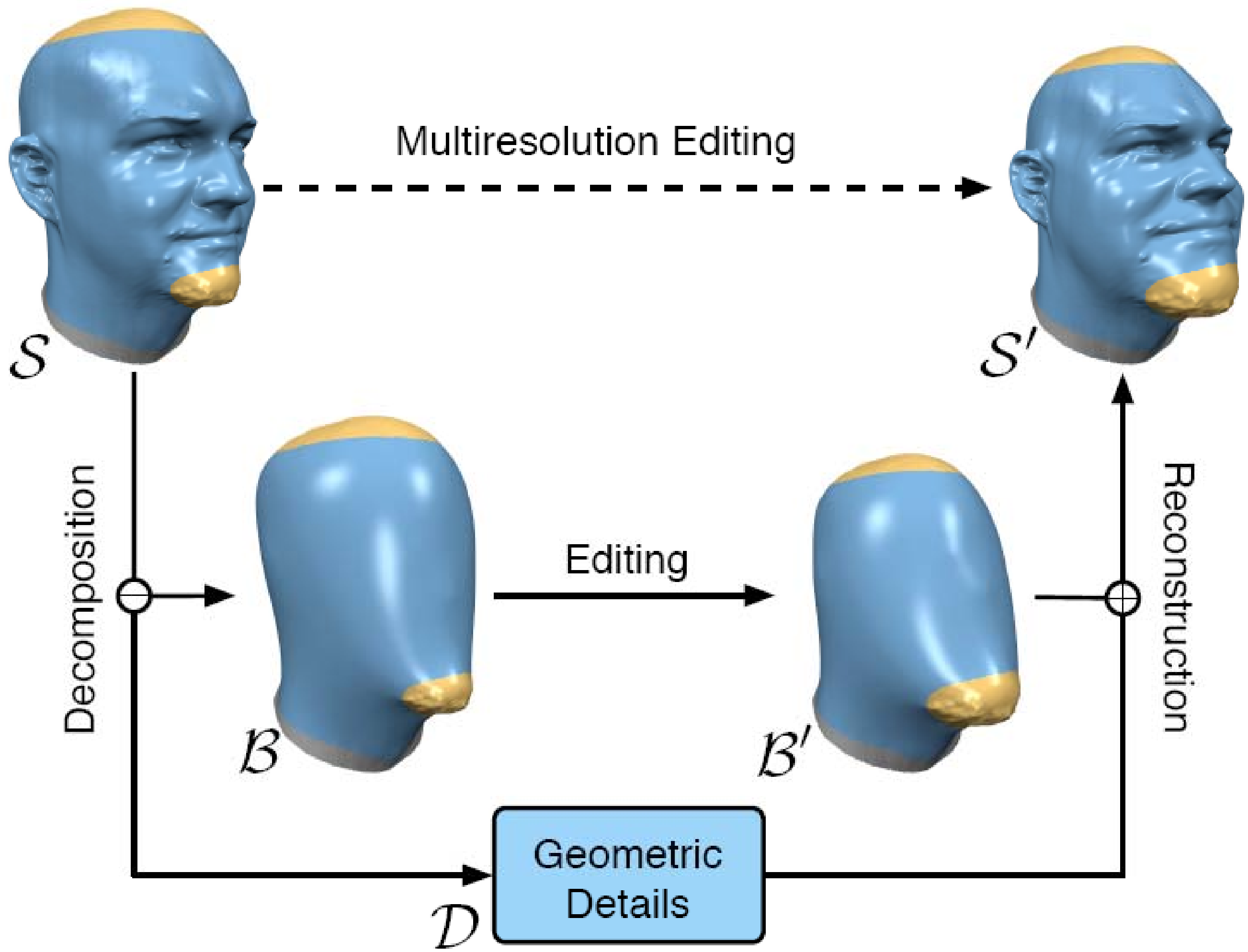
Displace the vertices to get the result



Fixing local rotations

Multiresolution framework

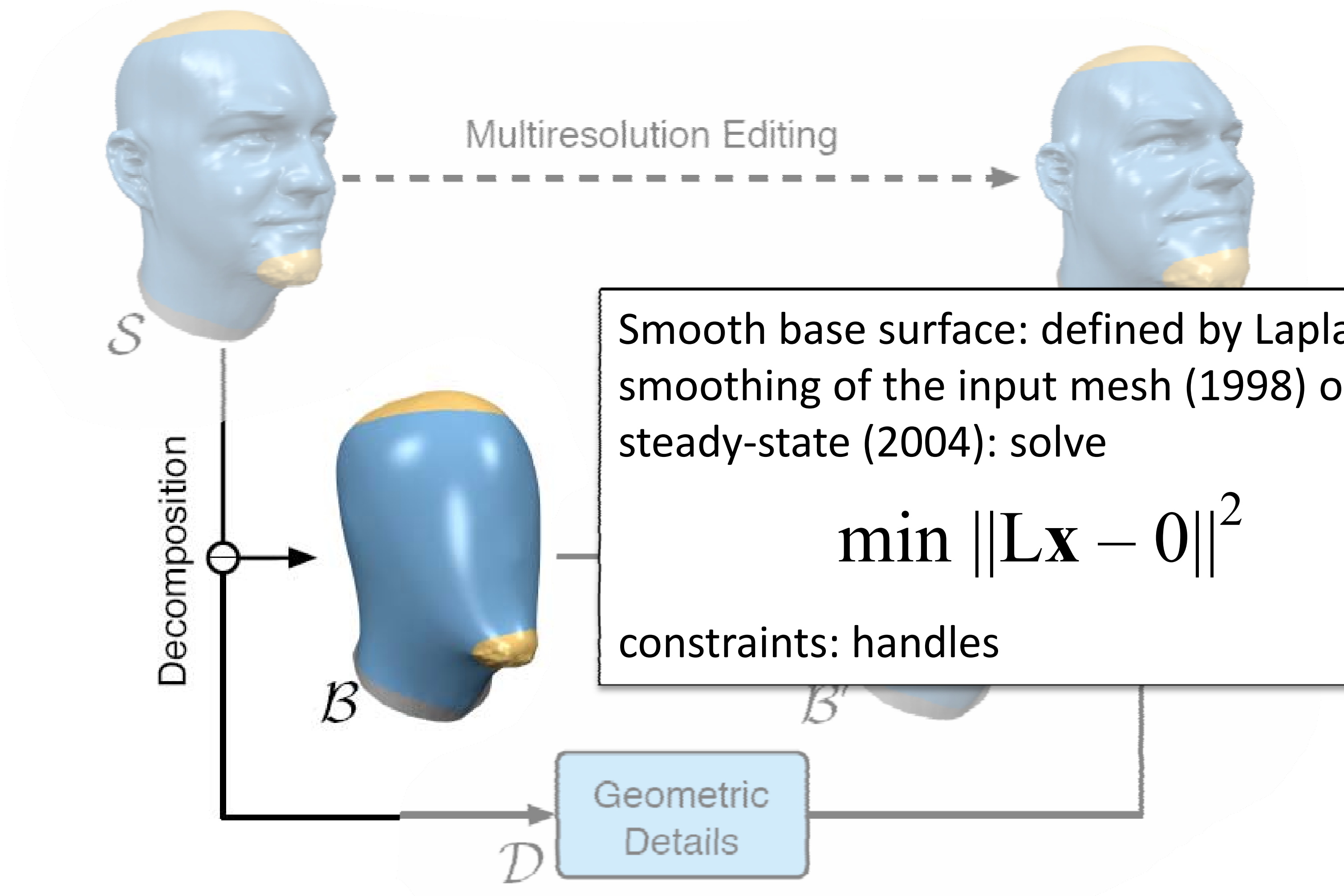
- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



Fixing local rotations

Multiresolution framework

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



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

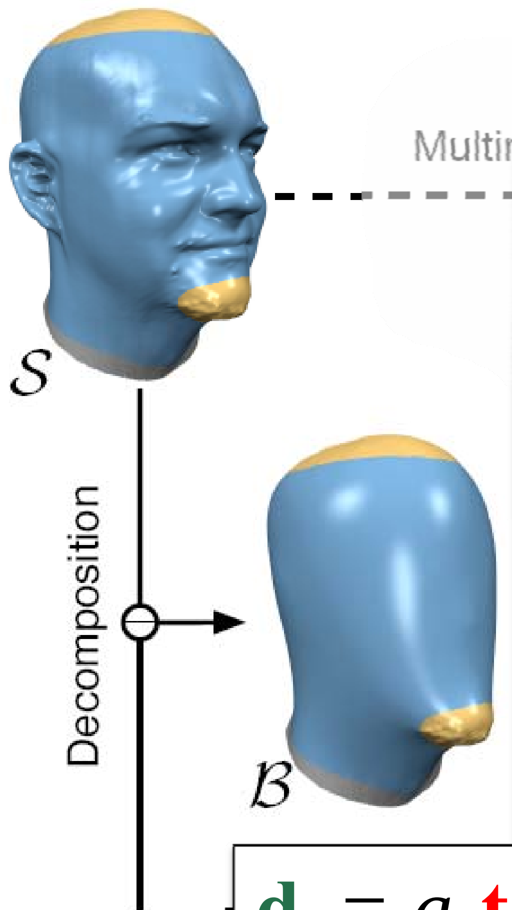
$$\min \|L\mathbf{x} - 0\|^2$$

constraints: handles

Fixing local rotations

Multiresolution framework

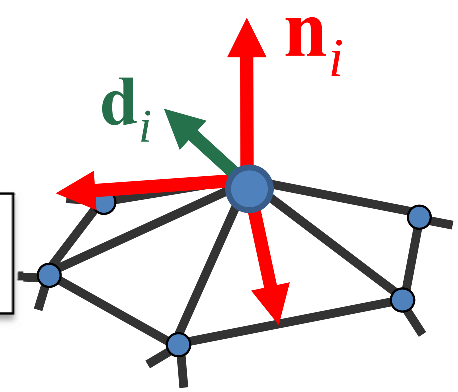
- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



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 $\mathbf{b}_i$



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

Fixing local rotations

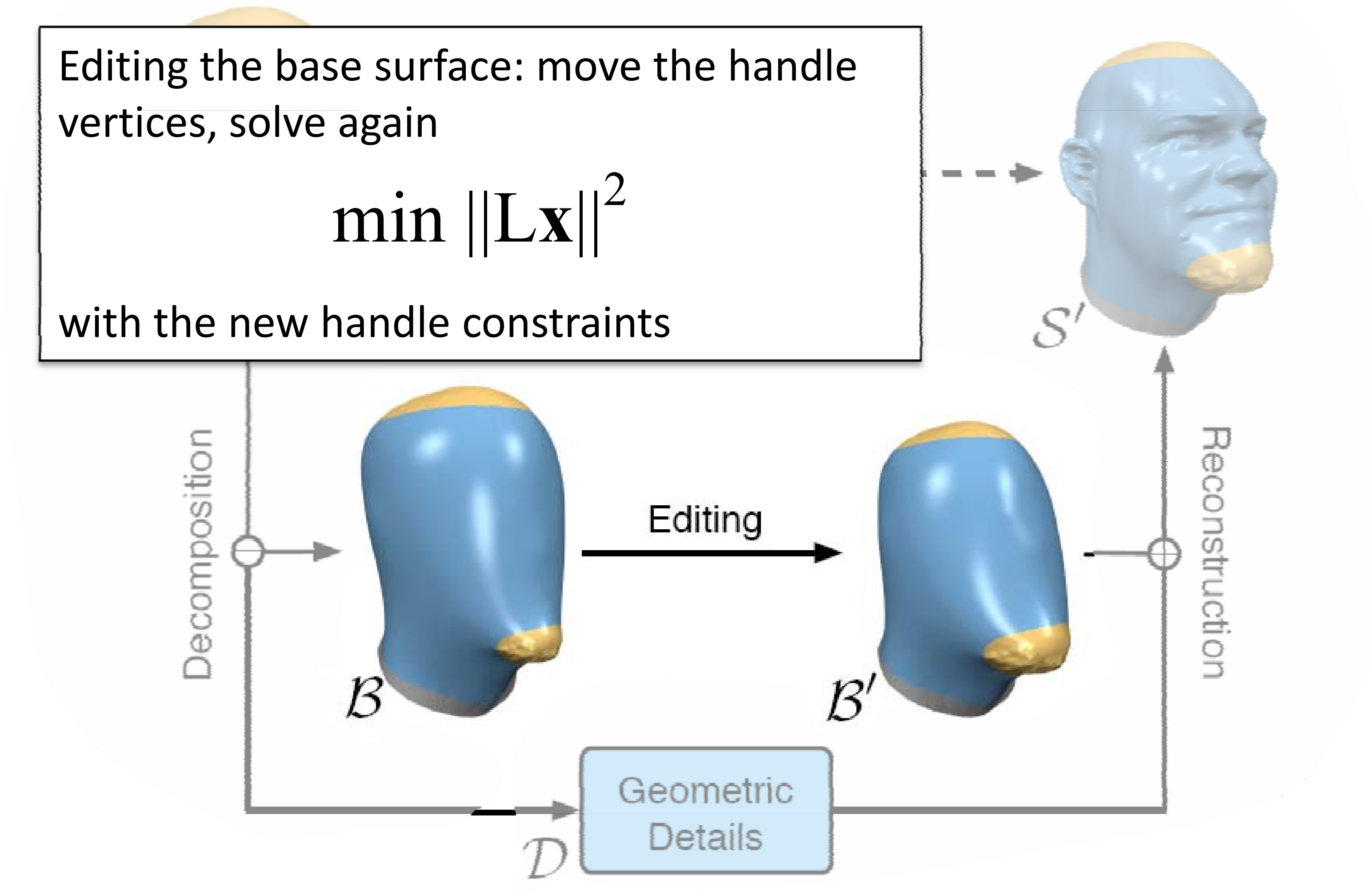
Multiresolution framework

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

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

$$\min \|\mathrm{L}\mathbf{x}\|^2$$

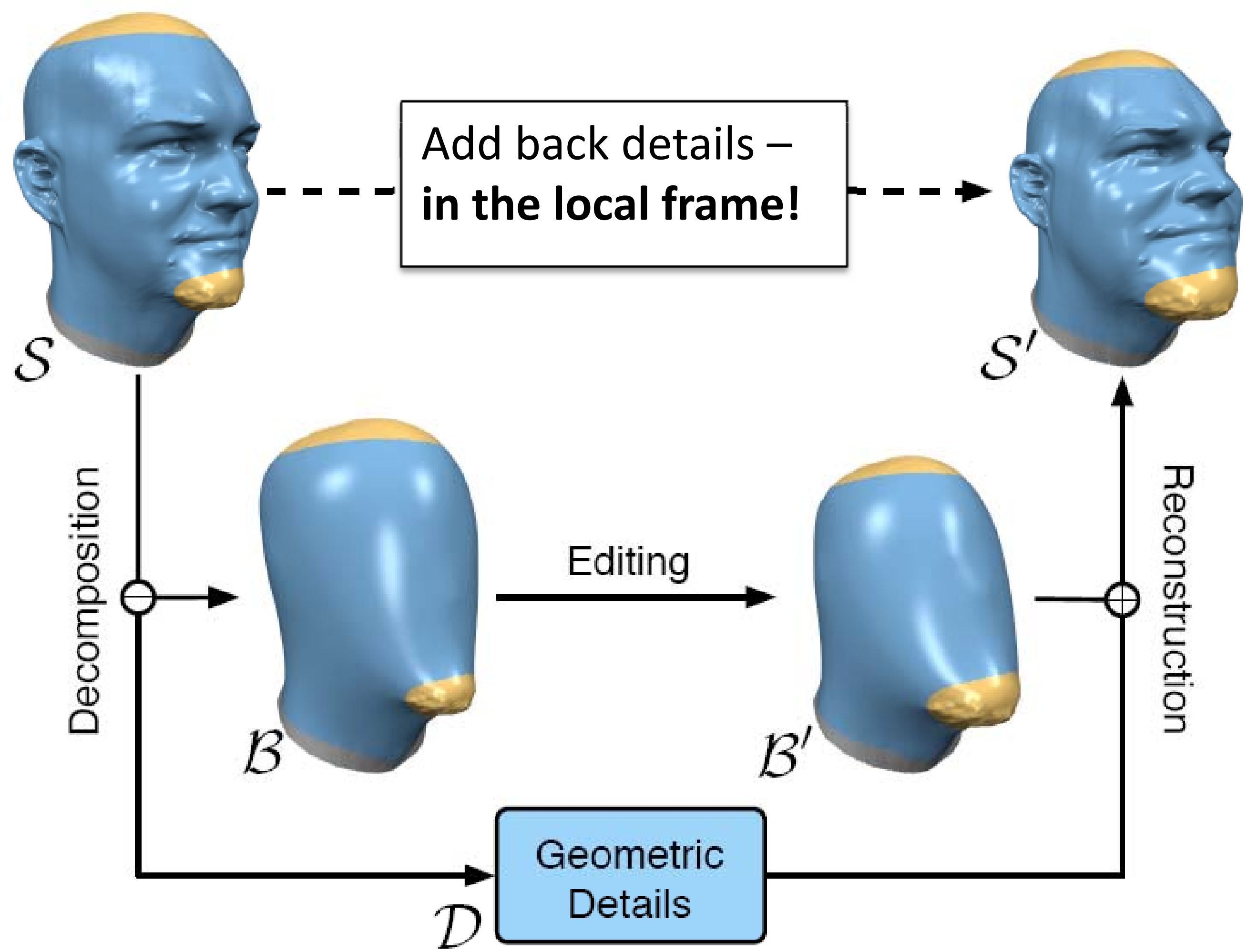
with the new handle constraints



Fixing local rotations

Multiresolution framework

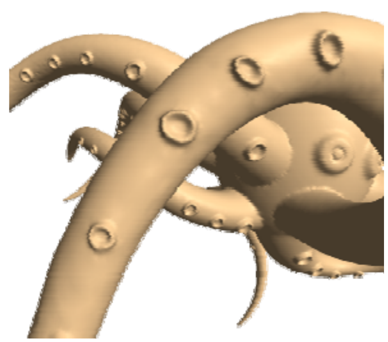
- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



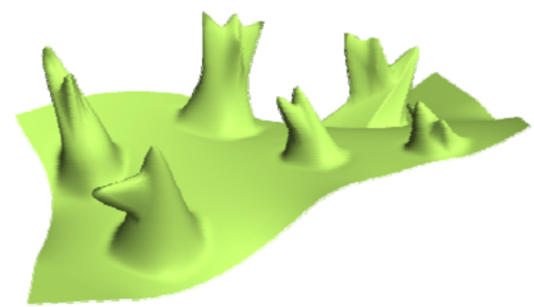
Multiresolution framework

Discussion

- Advantages:
  - Fast! Linear solve for the base surface deformation, and then add back displacements
  - Intuitive, easy to implement
- Problem: works only for small height fields (details vectors are small)



almost a height field

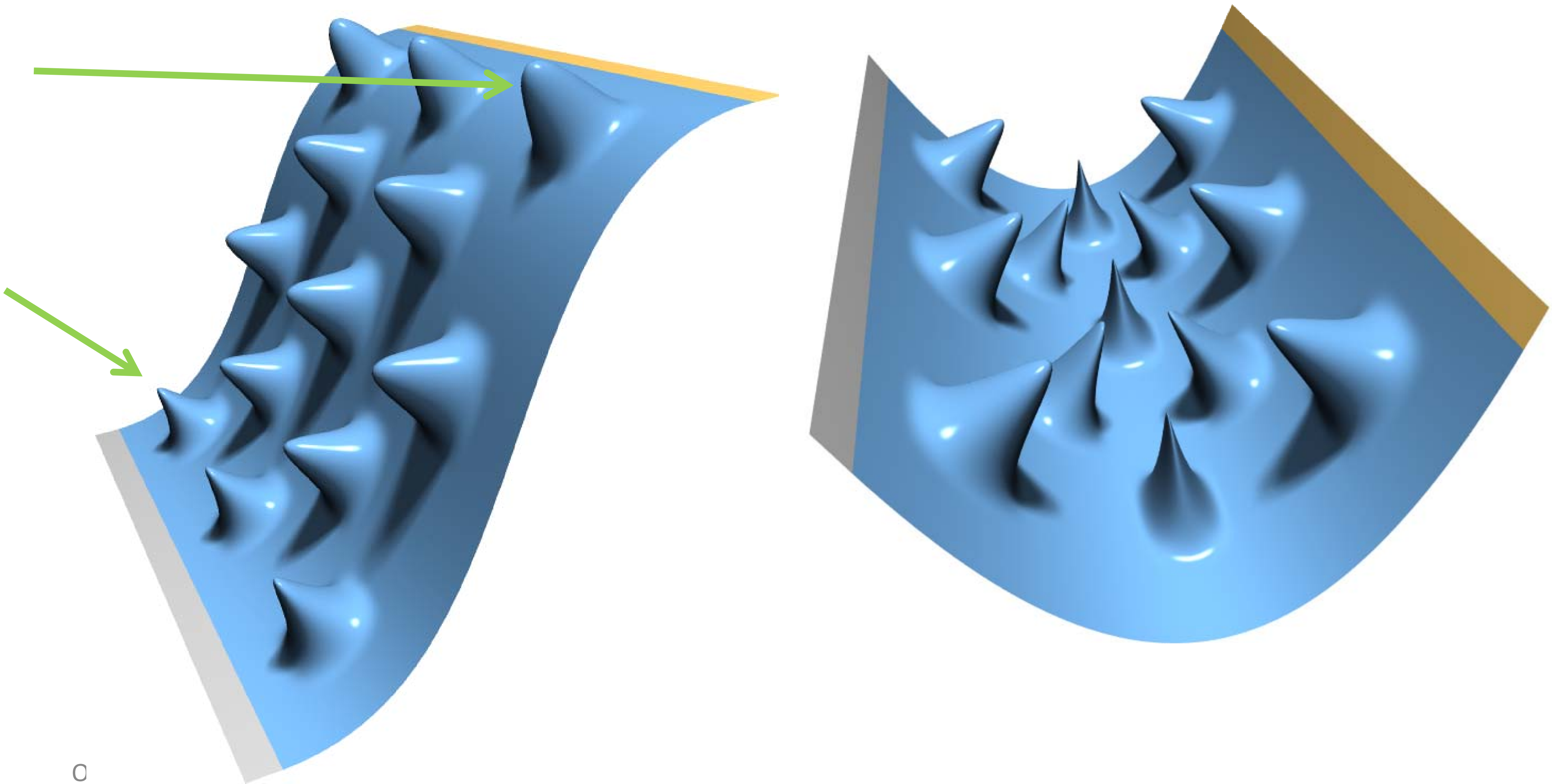


not a height field

Multiresolution framework

Discussion

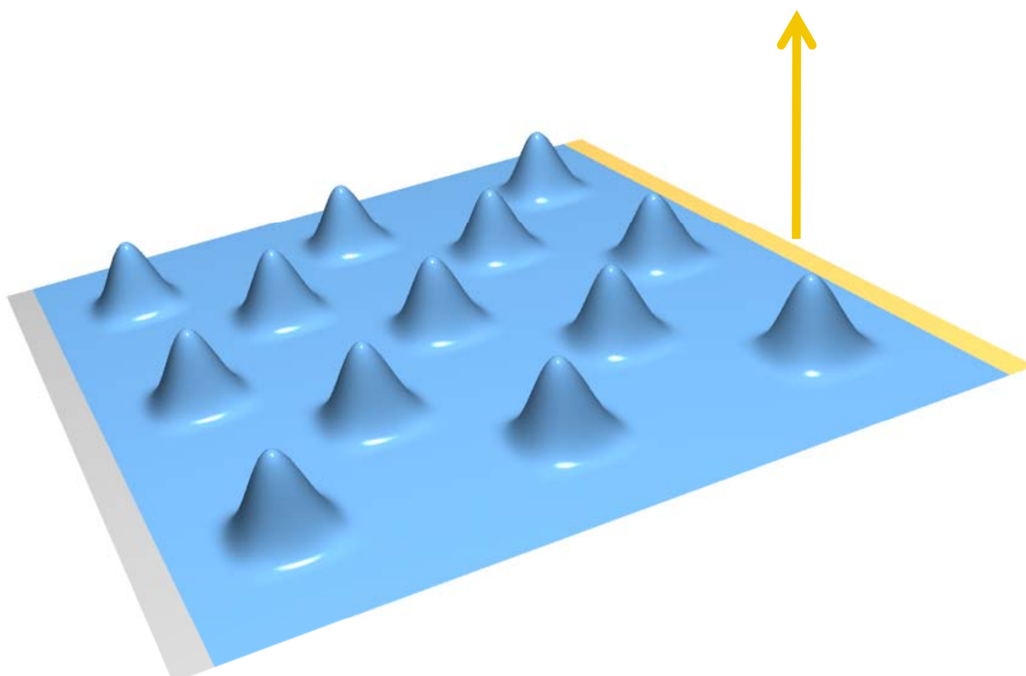
- Problem: If detail vectors are too big we get overshooting and **self-intersections**, especially in concave cases



0

Local rotations – single res. solutions

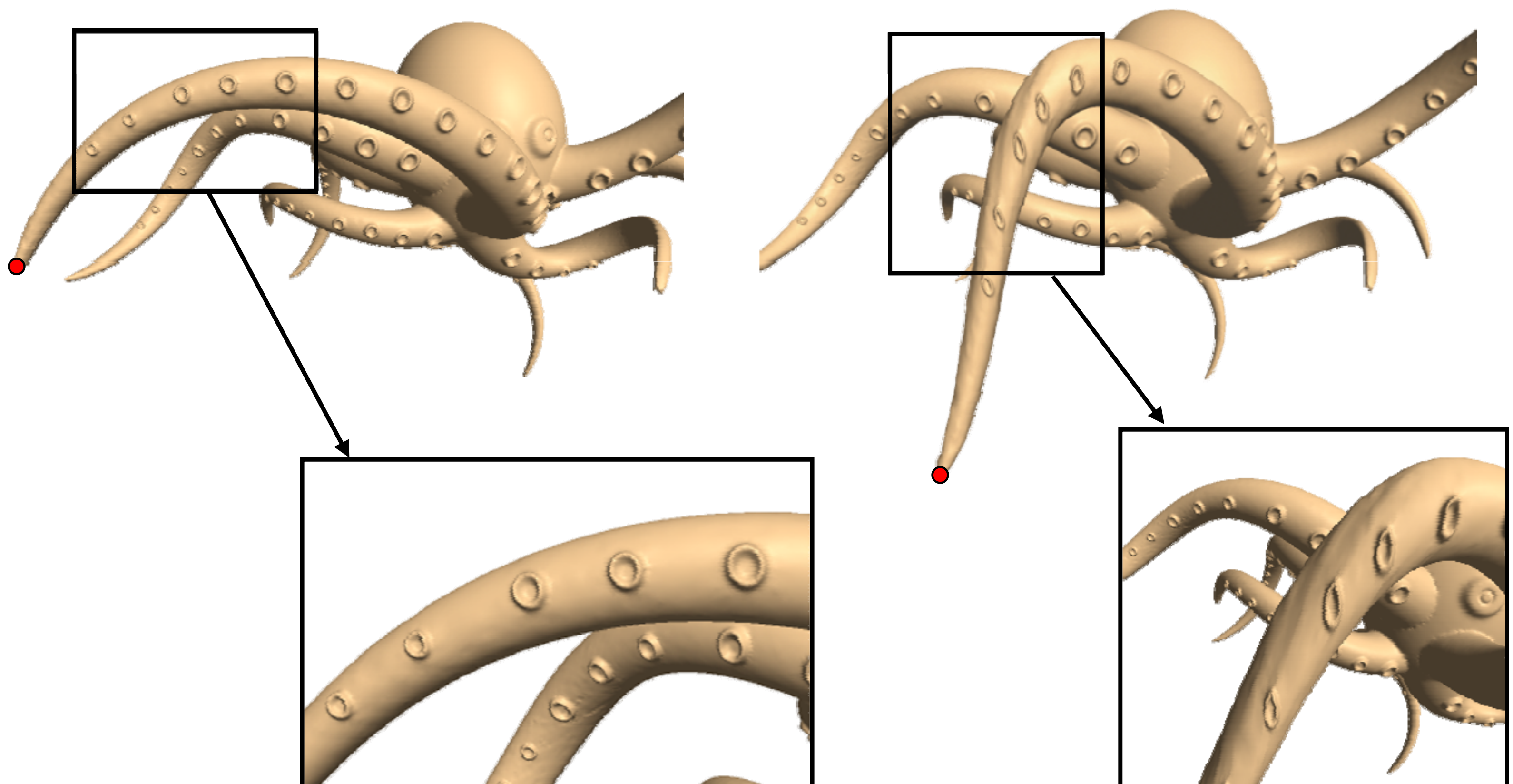
- Come up with a rotation field on the surface based on the modeling constraints
- Rotate the differential coordinates; solve



Estimation of rotations

Lipman et al. 2004

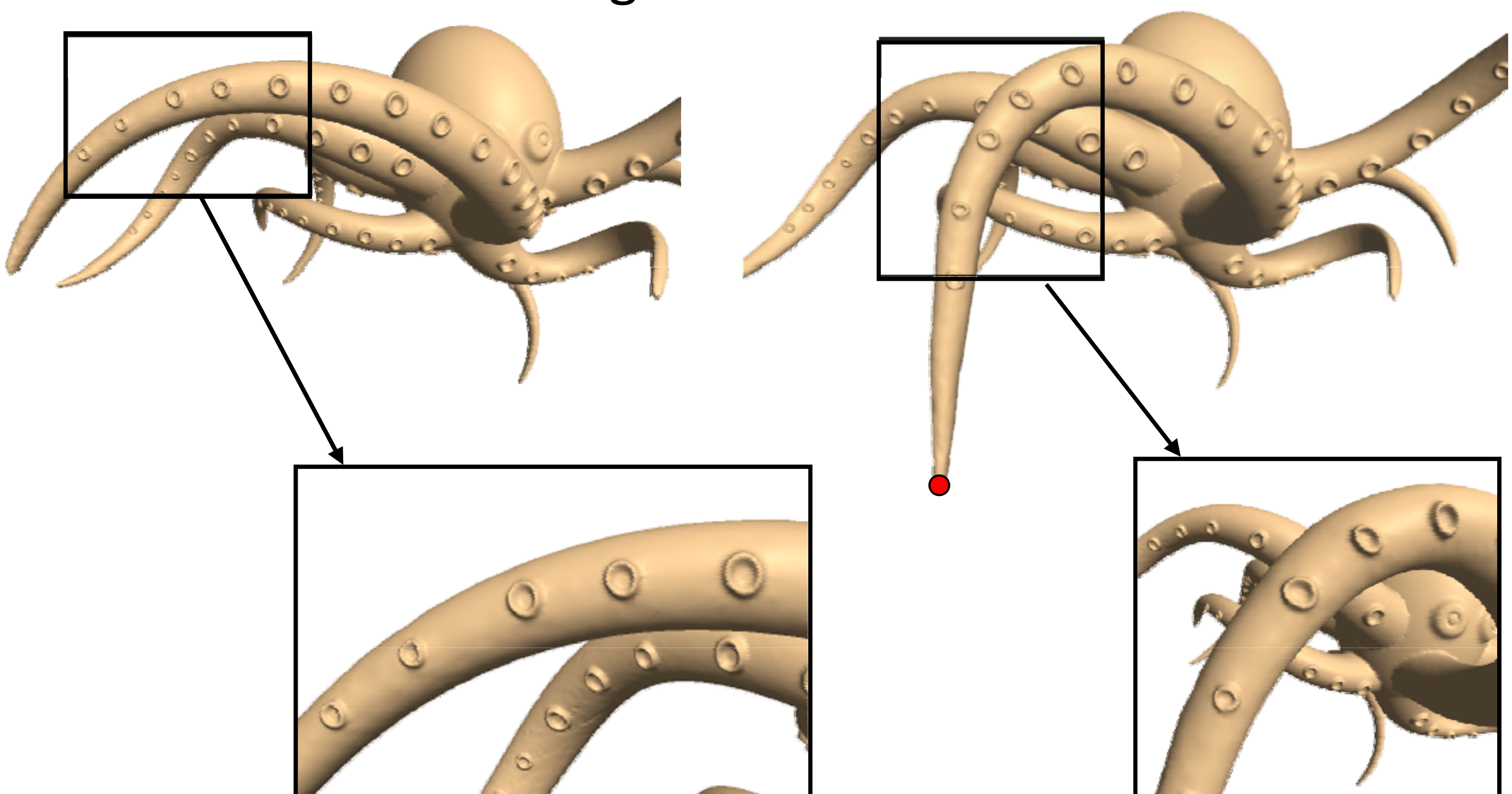
- Reconstruct the surface with the original Laplacians δ (naïve Laplacian editing)
- Compute smoothed local frames, estimate rotation



Estimation of rotations

Lipman et al. 2004

- Reconstruct the surface with the original Laplacians δ (naïve Laplacian editing)
- Compute smoothed local frames, estimate rotation
- Rotate the δ's and reconstruct again



Estimation of rotations

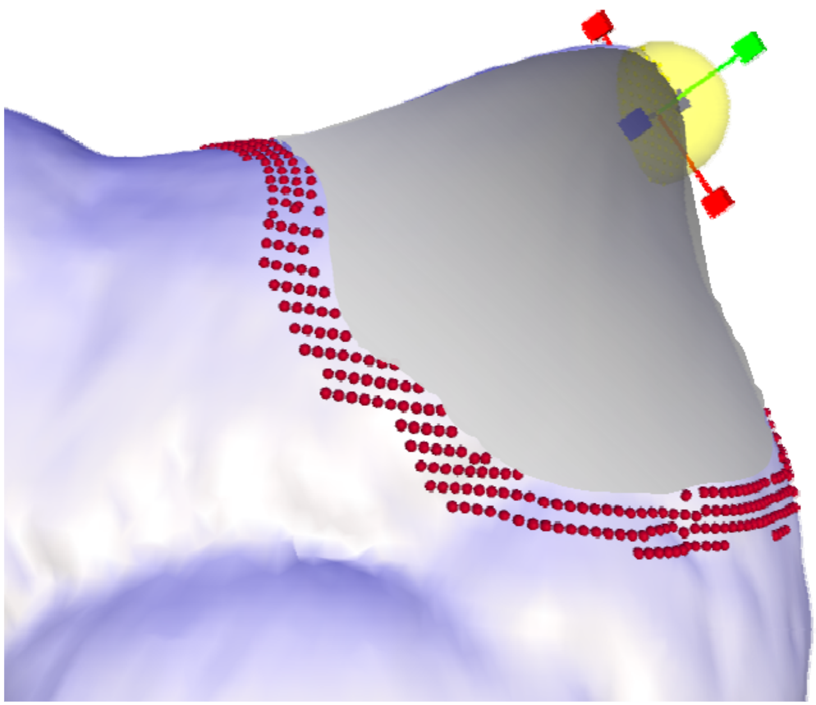
Lipman et al. 2004

- Advantages:
  - Sparse linear solve
  - Less or no self-intersections thanks to global optimization (no more local displacements that fight each other)
- Disadvantages:
  - Heuristic estimation of the rotations
  - Speed depends on the support of the smooth local frame estimation operator; for highly detailed surfaces it must be large
  - Unclear how much we need to smooth (what is detail?)

Rotation propagation

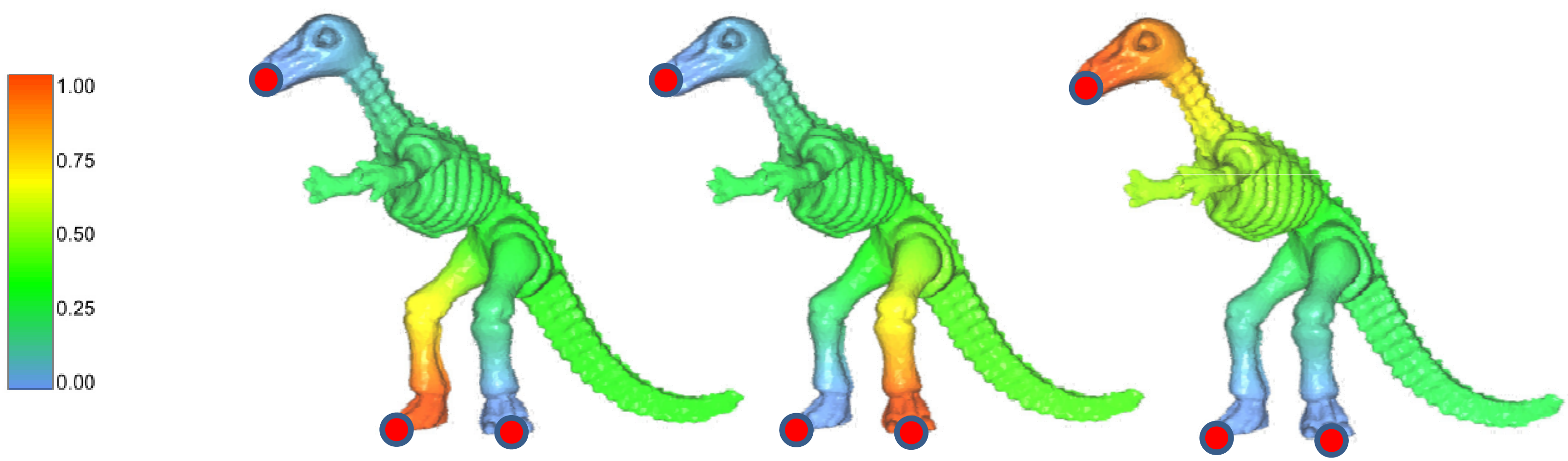
[Yu et al. SIGGRAPH 2004][Zayer et al. EG 2005][Lipman et al. SIGGRAPH 2005]

- Assume more user input: the user also specifies handle rotation
- The rotation is diffused to the rest of the ROI



Rotation propagation

- Geodesic distance [Yu et al. 2004]
- Harmonic field [Zayer et al. 2005]
- Optimization [Lipman et al. 2005, 2006]



Harmonic field

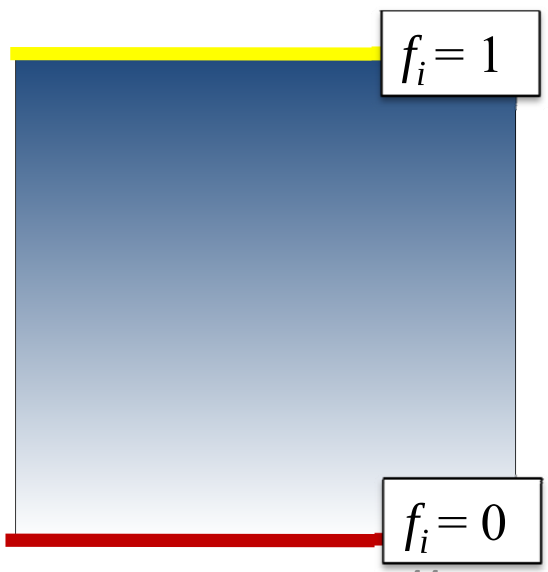
Harmonic fields on meshes

- Scalar function, attains 1 on the active handle, 0 on the static handles
- Smooth in-between, no local extrema
- Solve:

$$\Delta_M \mathbf{f} = 0$$

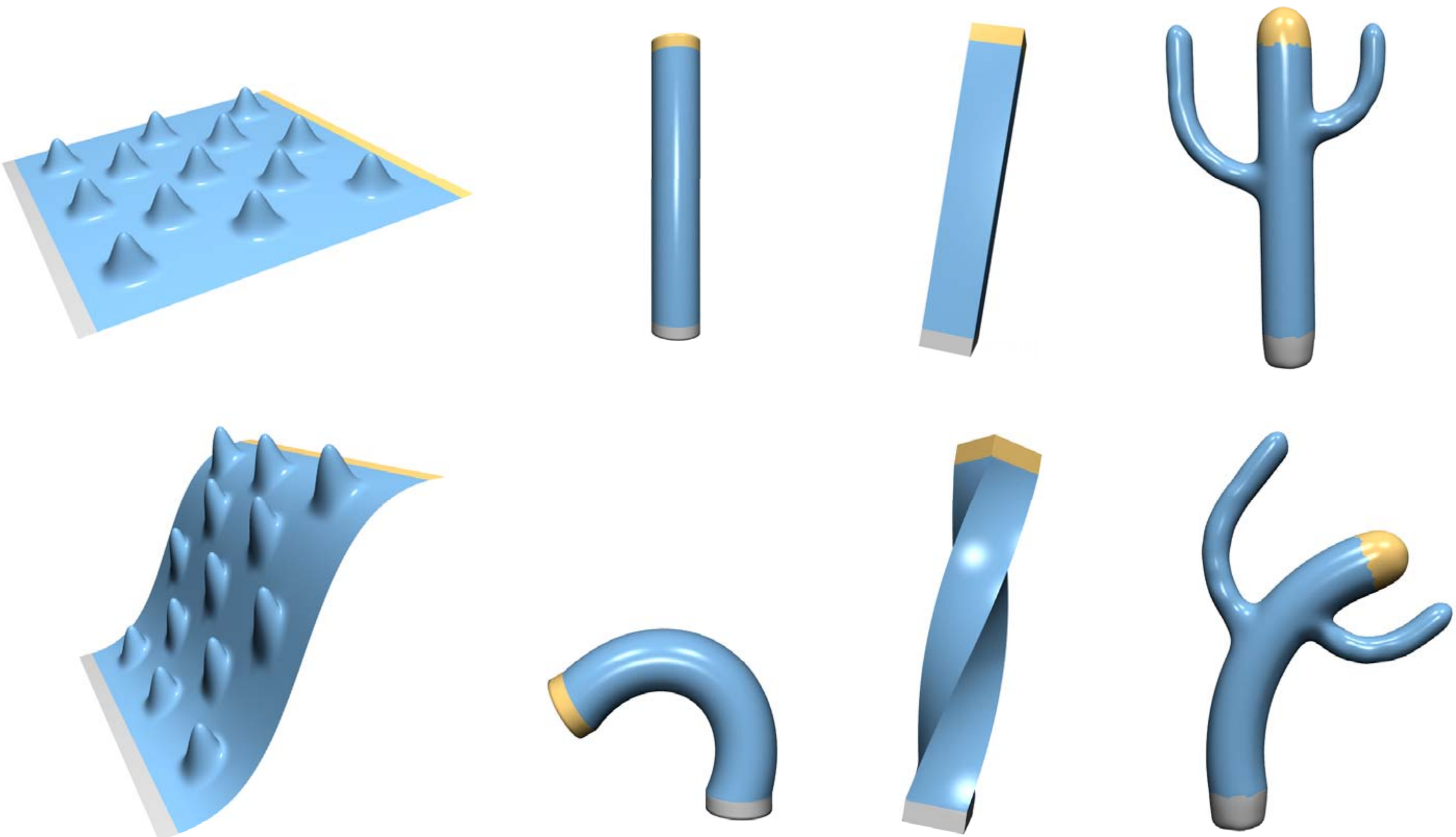
with constraints $f_i = 1$ on active handle, $f_i = 0$ on static handle

Example: in this simple case, the harmonic field is a just a linear ramp



Rotation propagation w/harmonic fields

Examples

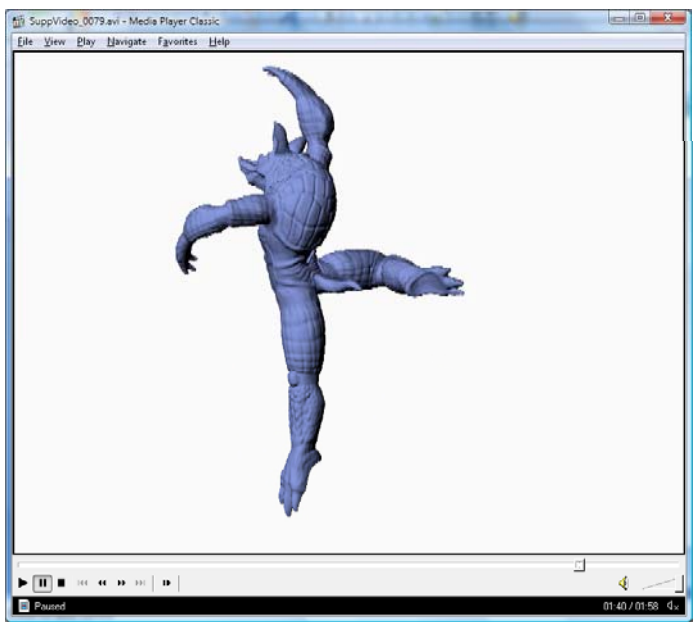


Why does this happen?

Rotation propagation w/harmonic fields

Examples

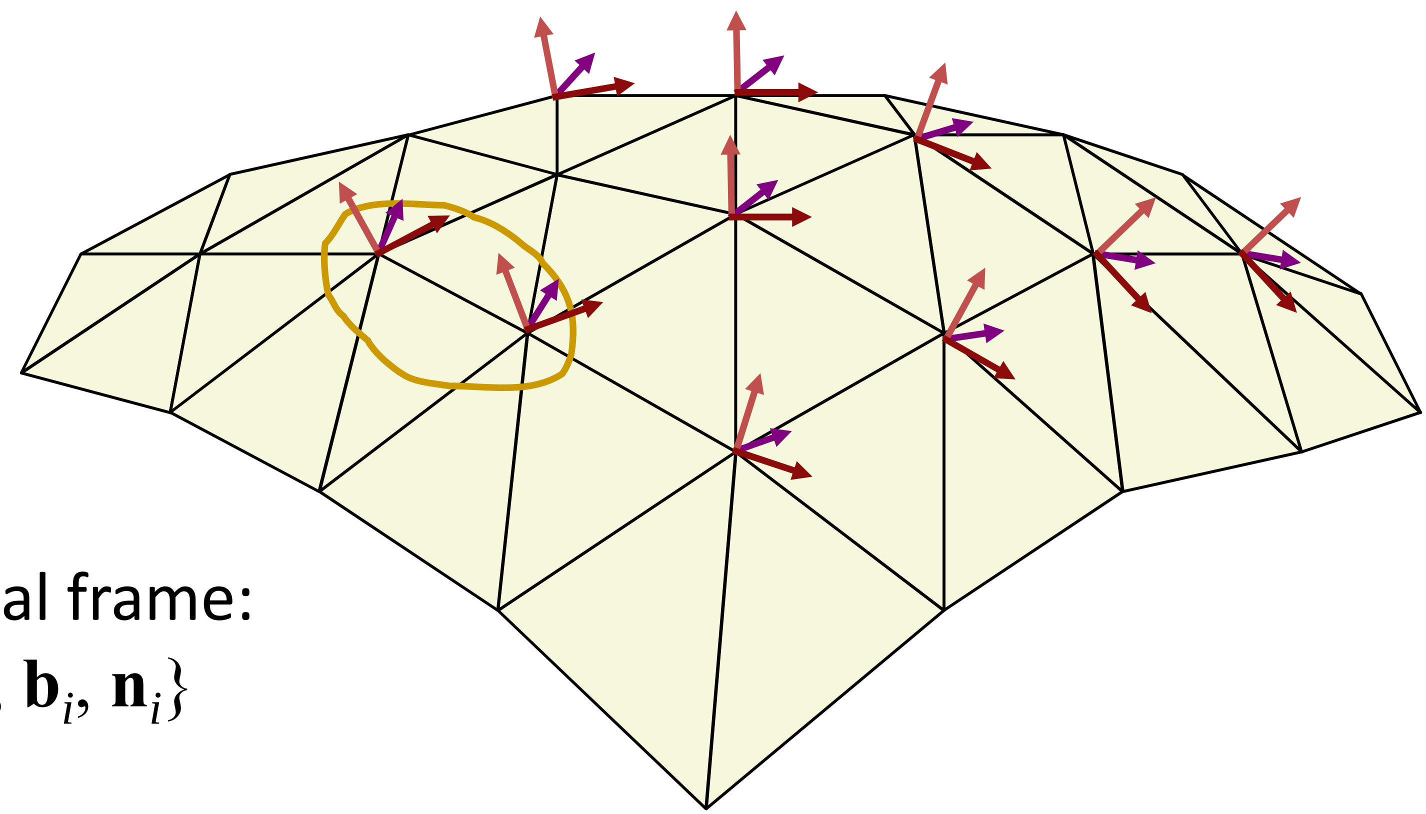
- If rotations are provided and consistent with the desired transformation, this works well
- However, the method is translation-insensitive (doesn't generate rotations when there are none provided)



Optimization of rotation propagation

Lipman et al. 2005

- Keep a local frame at each vertex
- Prescribe changes to some selected frames (rotation/scaling)



Local frame:

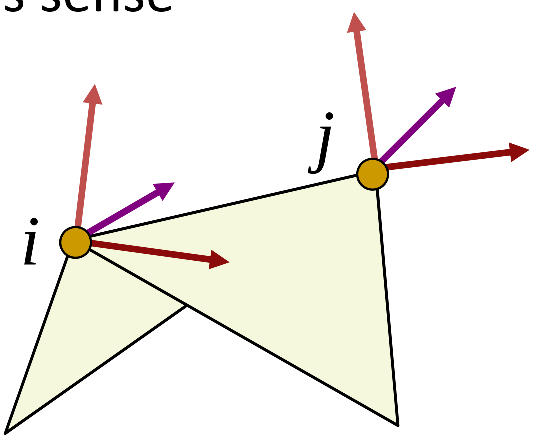
$\{\mathbf{a}_i, \mathbf{b}_i, \mathbf{n}_i\}$

Optimization of rotation propagation

Lipman et al. 2005

- Reconstruction:
  - Encode the differences between adjacent frames – the numbers α β γ for each edge…
  - Solve for the new frames in least-squares sense

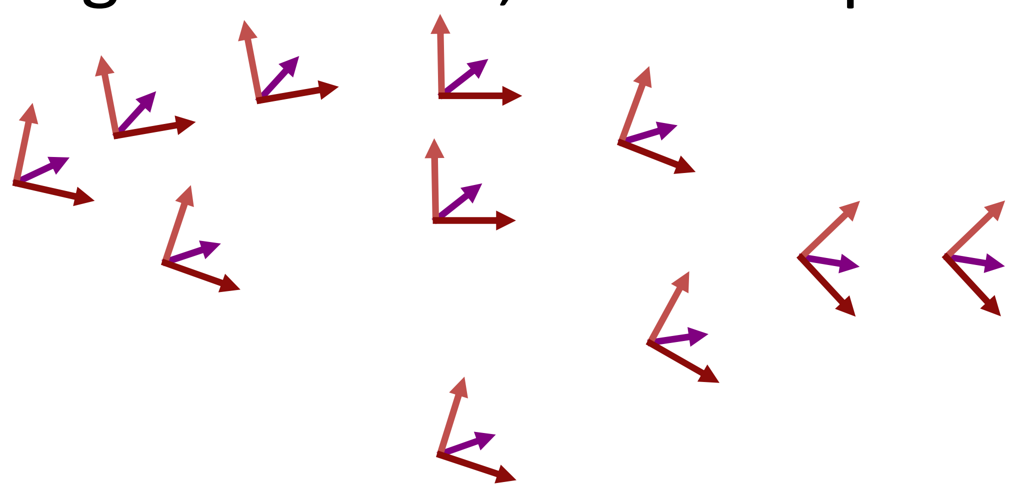
$$
\begin{aligned}
\mathbf{a}_i - \mathbf{a}_j &= \alpha_1 \mathbf{a}_i + \alpha_2 \mathbf{b}_i + \alpha_3 \mathbf{n}_i \\
\mathbf{b}_i - \mathbf{b}_j &= \beta_1 \mathbf{a}_i + \beta_2 \mathbf{b}_i + \beta_3 \mathbf{n}_i \\
\mathbf{n}_i - \mathbf{n}_j &= \gamma_1 \mathbf{a}_i + \gamma_2 \mathbf{b}_i + \gamma_3 \mathbf{n}_i \\
&\dots\dots \\
&\mathit{constraints}
\end{aligned}
$$



Optimization of rotation propagation

Lipman et al. 2005

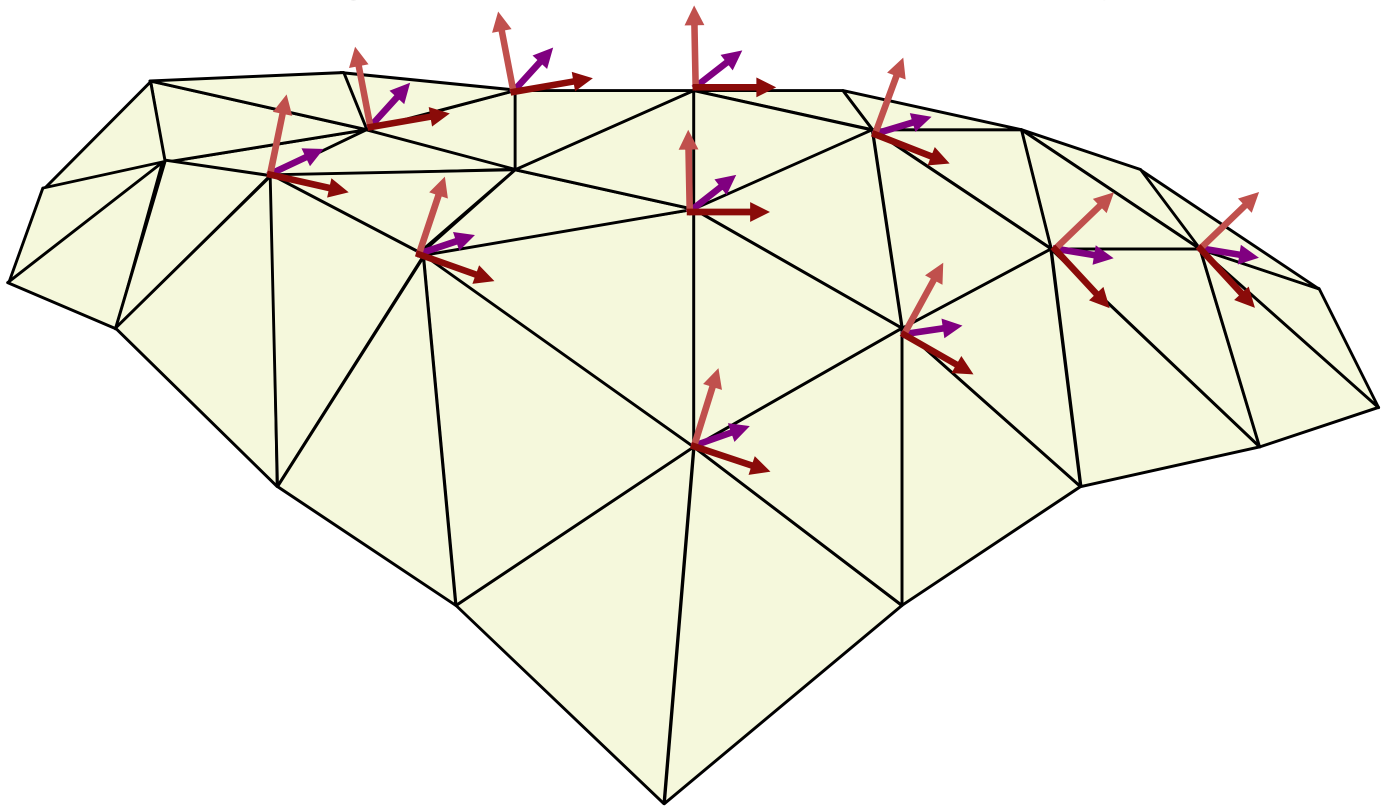
- Reconstruction:
  - After having the frames, solve for positions



Optimization of rotation propagation

Lipman et al. 2005

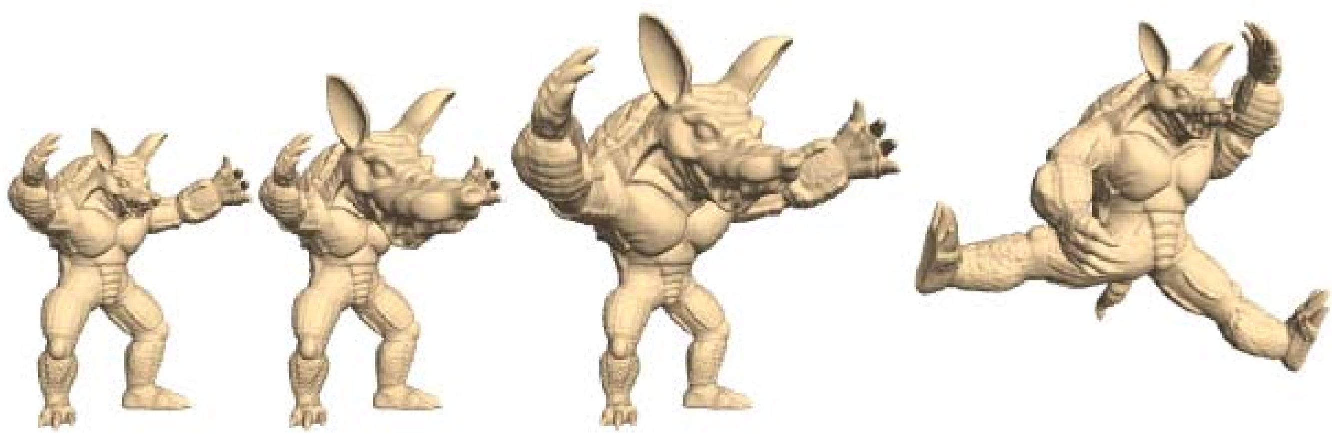
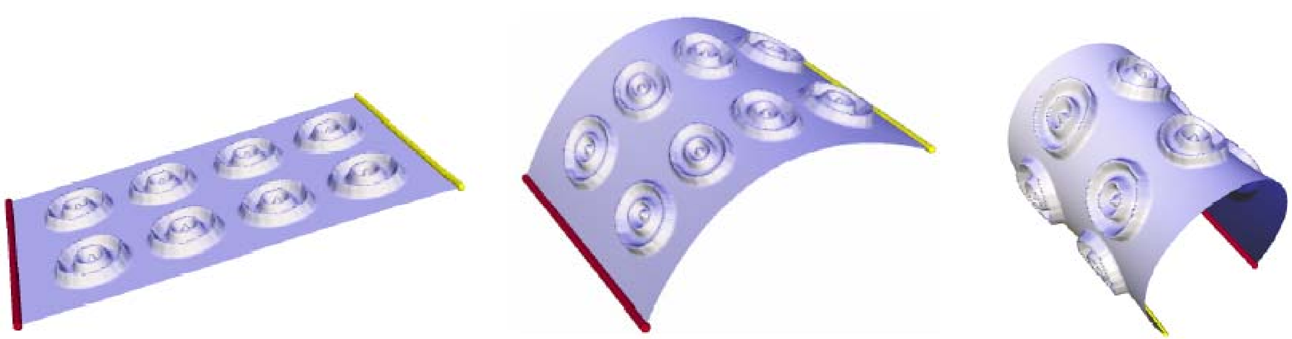
- Reconstruction:
  - After having the frames, solve for positions



Optimization of rotation propagation

Lipman et al. 2005

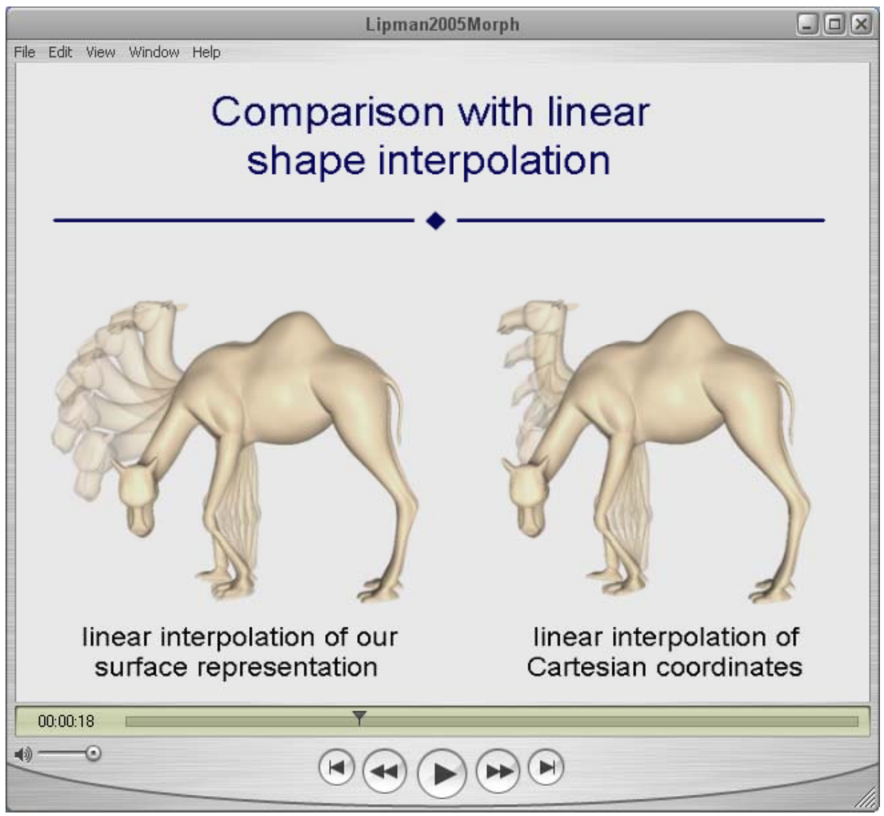
- Some results


Optimization of rotation propagation

Lipman et al. 2005

- Can use this representation for shape interpolation



Implicit definition of transformations

Sorkine et al. 2004

- The idea: solve for local transformations AND the edited surface simultaneously!
- Estimate the local transformations T_i from the eventual solution

$$\tilde{V}' = \arg\min_{V'} \left(\sum_{i=1}^{n} \left\| L(\mathbf{v}'_i) - T_i(\boldsymbol{\delta}_i) \right\|^2 + \sum_{j \in C} \left\| \mathbf{v}'_j - \mathbf{c}_j \right\|^2 \right)$$

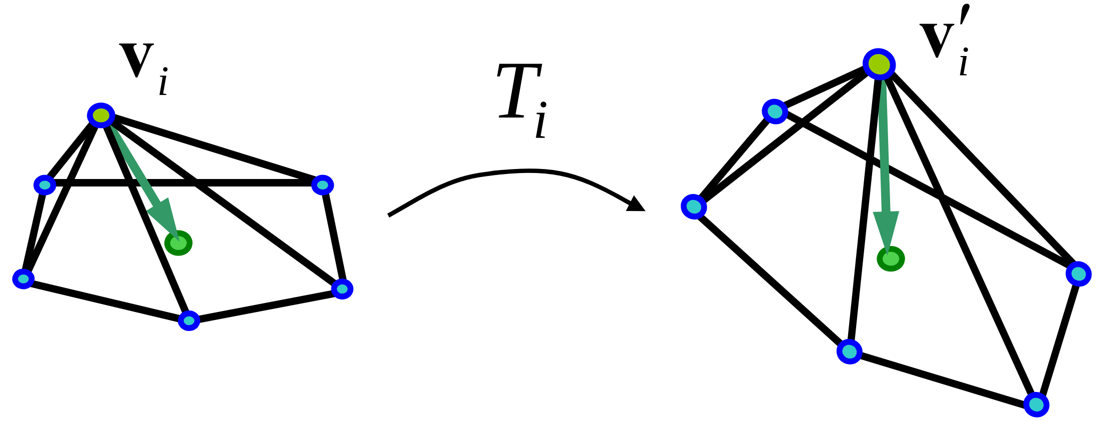
Transformation
of the local frame

Defining T_i

$$\tilde{V}' = \arg\min_{V'} \left(\sum_{i=1}^{n} \left\| L(\mathbf{v}'_i) - T_i(\boldsymbol{\delta}_i) \right\|^2 + \sum_{j \in C} \left\| \mathbf{v}'_j - \mathbf{c}_j \right\|^2 \right)$$

- How to formulate T_i?
  - Based on the local (1-ring) neighborhood
  - Linear dependence on the unknown $\mathbf{v}'_i$

$$\mathbf{v}'_i = T_i \, \mathbf{v}_i$$
$$\mathbf{v}'_{j1} = T_i \, \mathbf{v}_{j1}$$
$$\vdots$$
$$\mathbf{v}'_{jk} = T_i \, \mathbf{v}_{jk}$$



Defining T_i

- First attempt: define T_i simply by solving

$$T_i = \arg\min_{T_i} \sum_{j=1}^{k} \left\| \mathbf{v}'_{i_j} - T_i \mathbf{v}_{i_j} \right\|^2$$

$$\Downarrow$$

$$\begin{pmatrix} & & \\ & T_i & \\ & & \end{pmatrix} = \begin{pmatrix} | & | & & & | \\ \mathbf{v}'_i & \mathbf{v}'_{j1} & \cdots & \cdots & \mathbf{v}'_{jk} \\ | & | & & & | \end{pmatrix} \begin{pmatrix} | & | & & & | \\ \mathbf{v}_i & \mathbf{v}_{j1} & \cdots & \cdots & \mathbf{v}_{jk} \\ | & | & & & | \end{pmatrix}^{+}$$

Defining T_i

- Plug the expressions for T_i into the least-squares reconstruction formula:

$$\tilde{V}' = \arg\min_{V'} \left(\sum_{i=1}^{n} \left\| L(\mathbf{v}'_i) - T_i(\boldsymbol{\delta}_i) \right\|^2 + \sum_{j \in C} \left\| \mathbf{v}'_j - \mathbf{c}_j \right\|^2 \right)$$

Linear combination of the unknown $\mathbf{v}'_i$

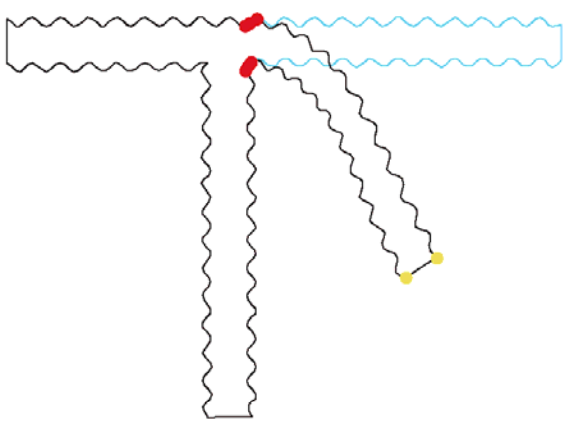
But: we didn't solve anything since T_i is arbitrary affine transformation, i.e. admits distorting shears

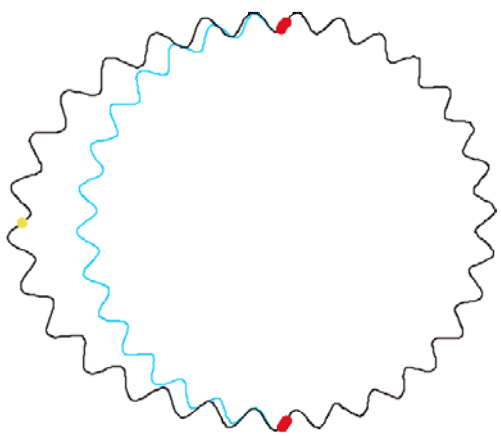
Constraining T_i

- Rotation + scale (i.e., similarity) is easy in 2D:

$$T_i = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & d_x \\ -\sin\theta & \cos\theta & d_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} w & a & t_x \\ -a & w & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

- Can edit 2D curves:



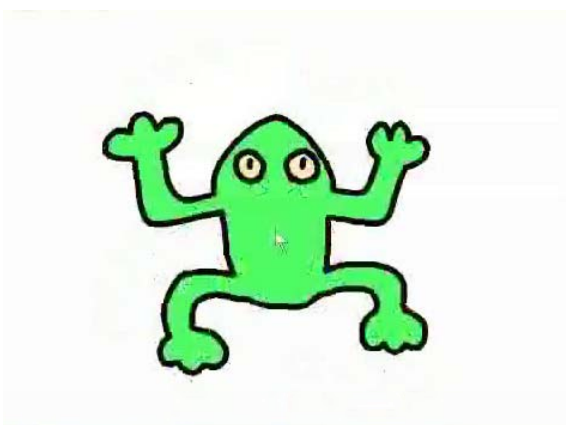


Constraining T_i

- Rotation + scale (i.e., similarity) is easy in 2D:

$$T_i = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & d_x \\ -\sin\theta & \cos\theta & d_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} w & a & t_x \\ -a & w & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

- Applied in [Igarashi et al. 05] for 2D shape manipulation:

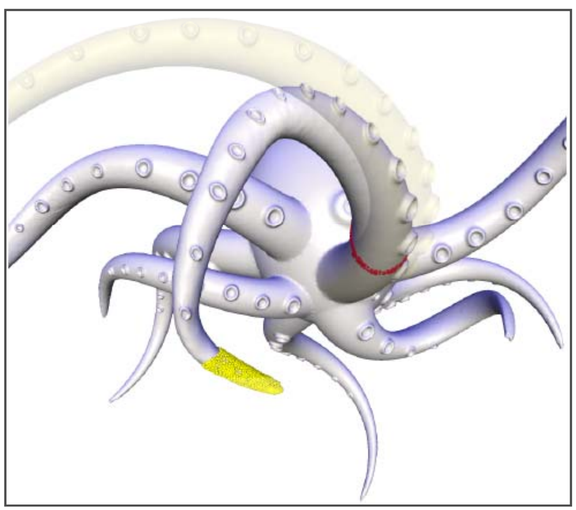


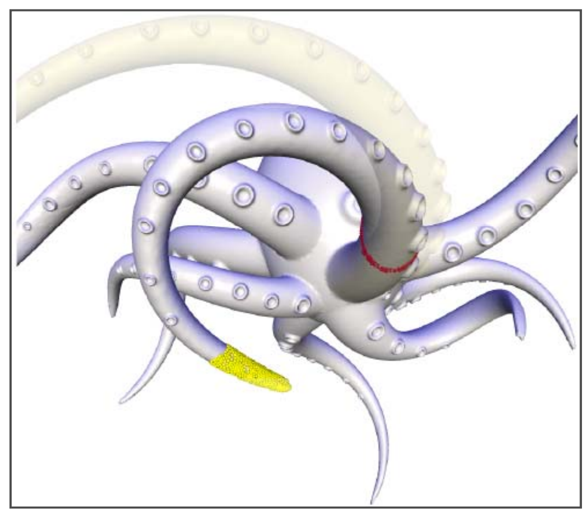
Defining the transformations T_i

- In 3D: have to linearize rotations

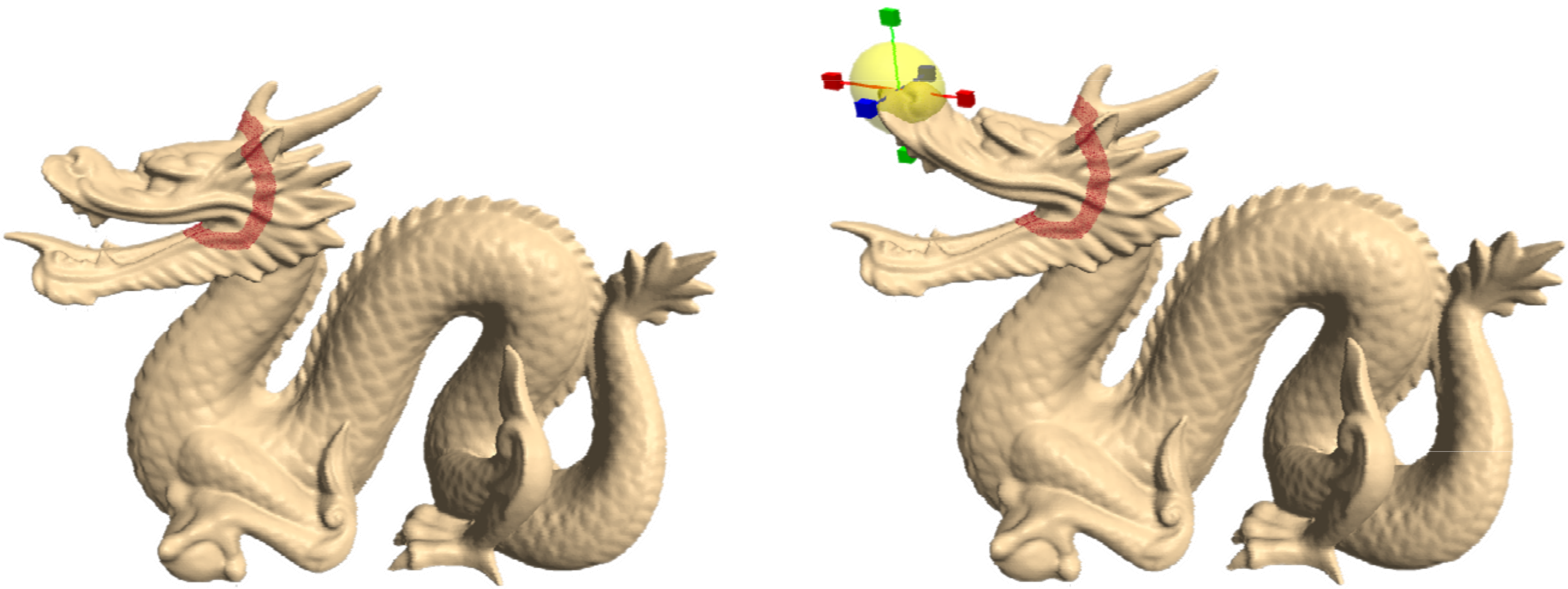
$$T_i = \begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Works well for moderate rotations, problems with large rotation angles

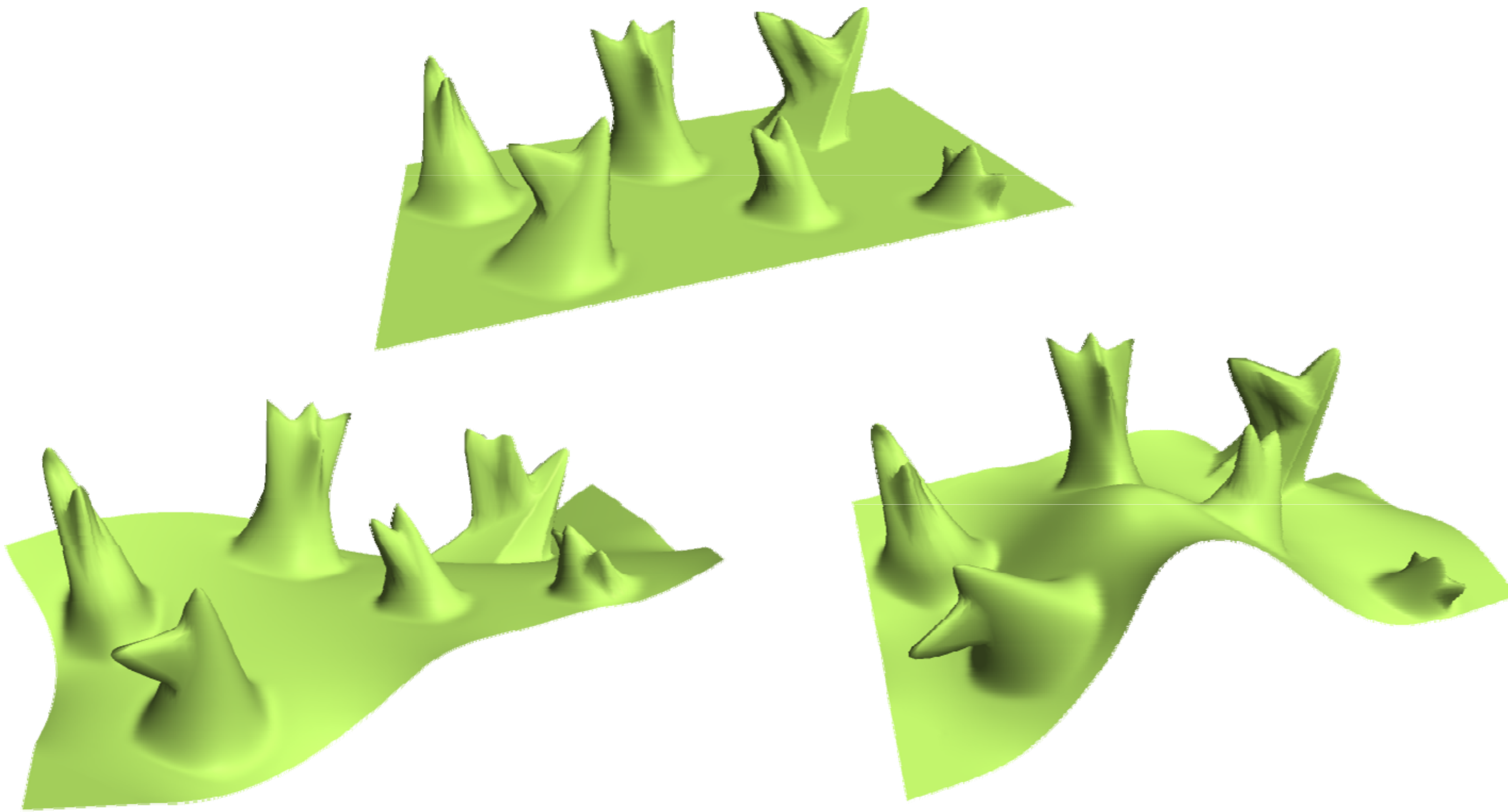




Laplacian editing results

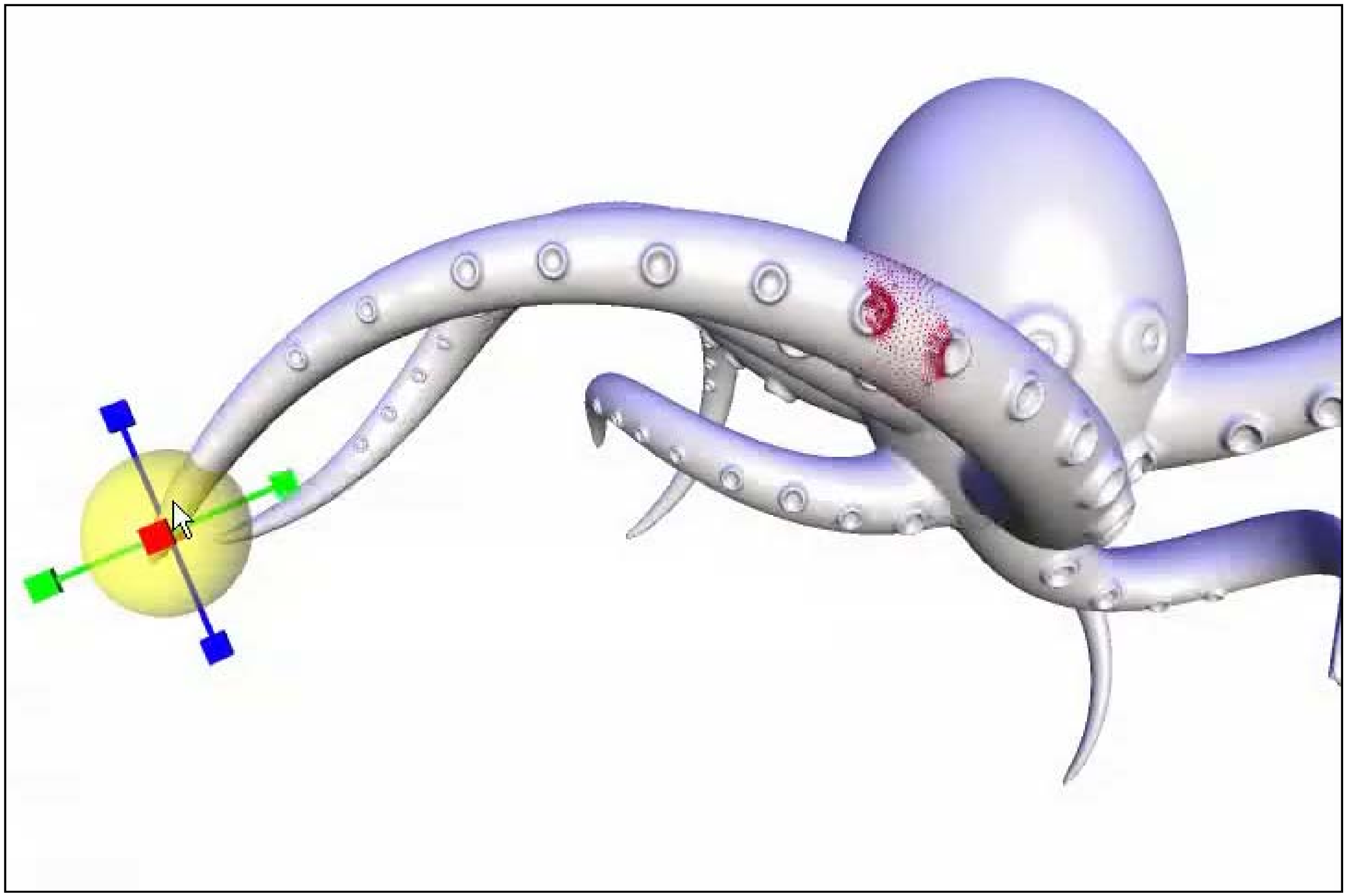


Laplacian editing results



Laplacian editing results

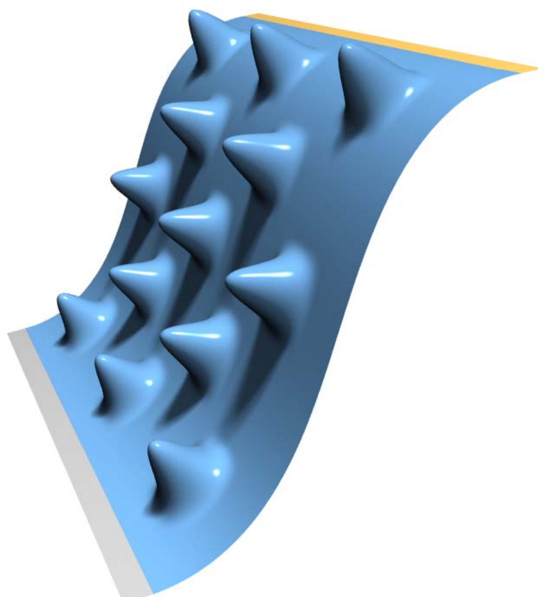
Laplacian editing results



Linear deformation methods

Summary

- Involve **linear** global optimization (efficient)
- Suffer from artifacts because of **local rotations**
- The relationship between the translation of a handle and the local rotation is inherently **nonlinear**



Nonlinear surface-based deformations

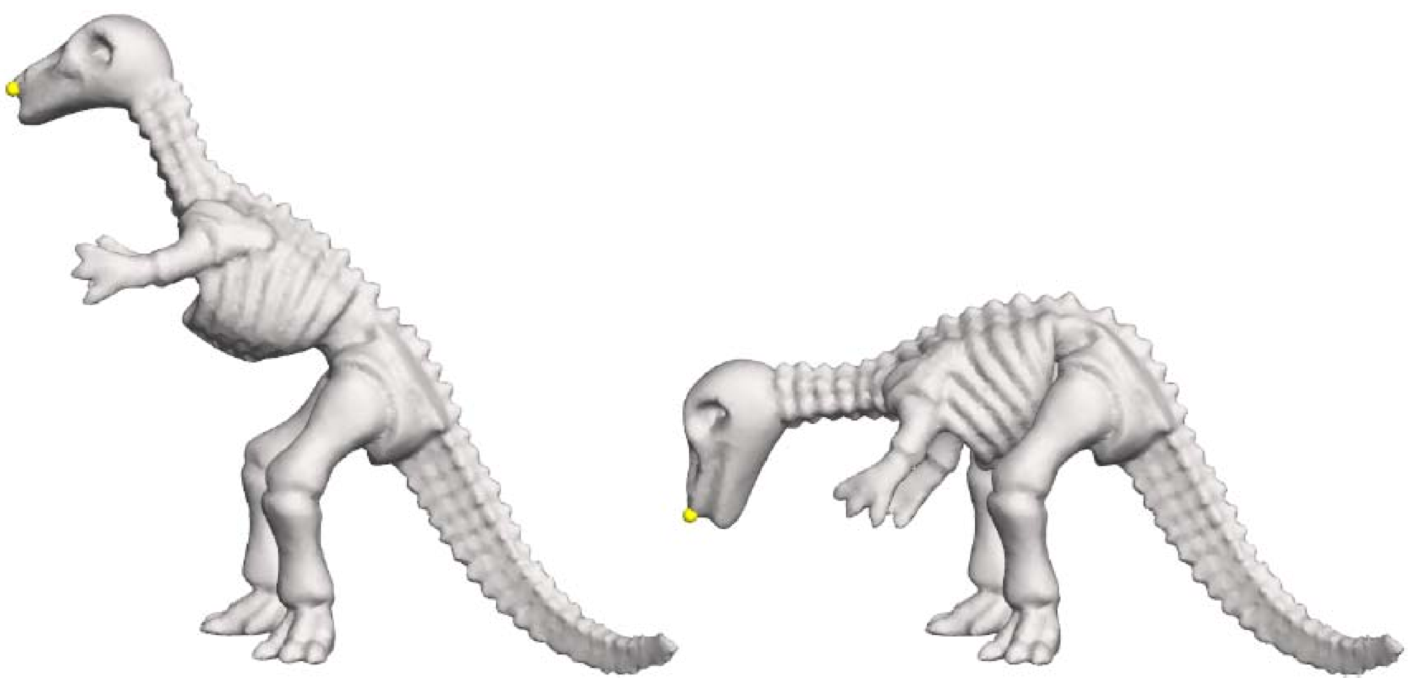
- Formulate a nonlinear functional that handles local rotations properly
- Still need an efficient method to minimize

$$\mathbf{p}' = \arg\min_{\mathbf{p}'} \mathrm{E}(\mathbf{p}, \mathbf{p}')$$

As-rigid-as-possible surface deformation

Sorkine and Alexa 2007

- Smooth effect on the large scale
- As-rigid-as-possible effect on the small scale (preserves details)



Modeling ARAP detail preservation

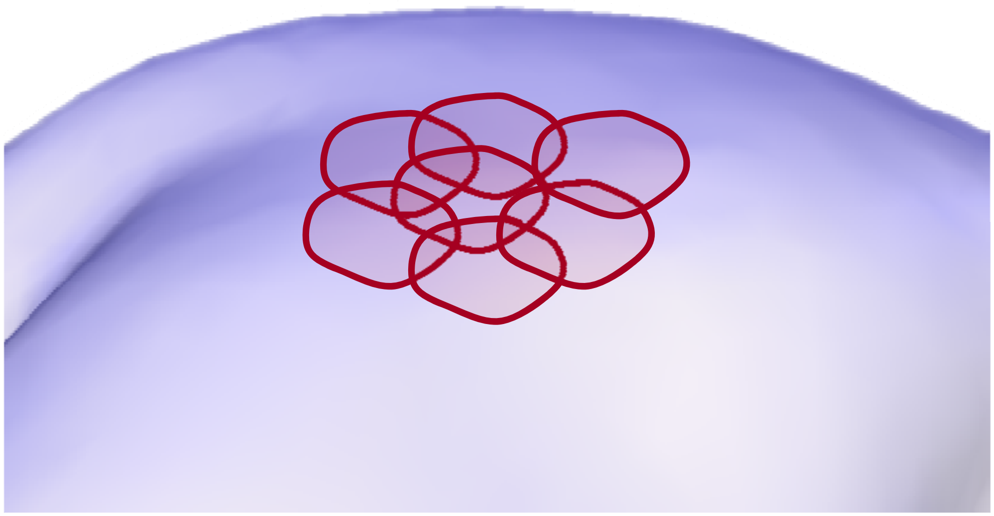
- Previous work: Laplacian editing and its variants

$$\min_{\mathbf{v}'} \sum_{i=1}^{n} \left\| L(\mathbf{v}'_i) - R_i \boldsymbol{\delta}_i \right\|^2 \qquad s.t.\ \mathbf{v}'_j = \mathbf{c}_j,\ j \in C$$

- Concentrated on making the optimization linear by “inventing” the right rotations or optimizing their linearized version

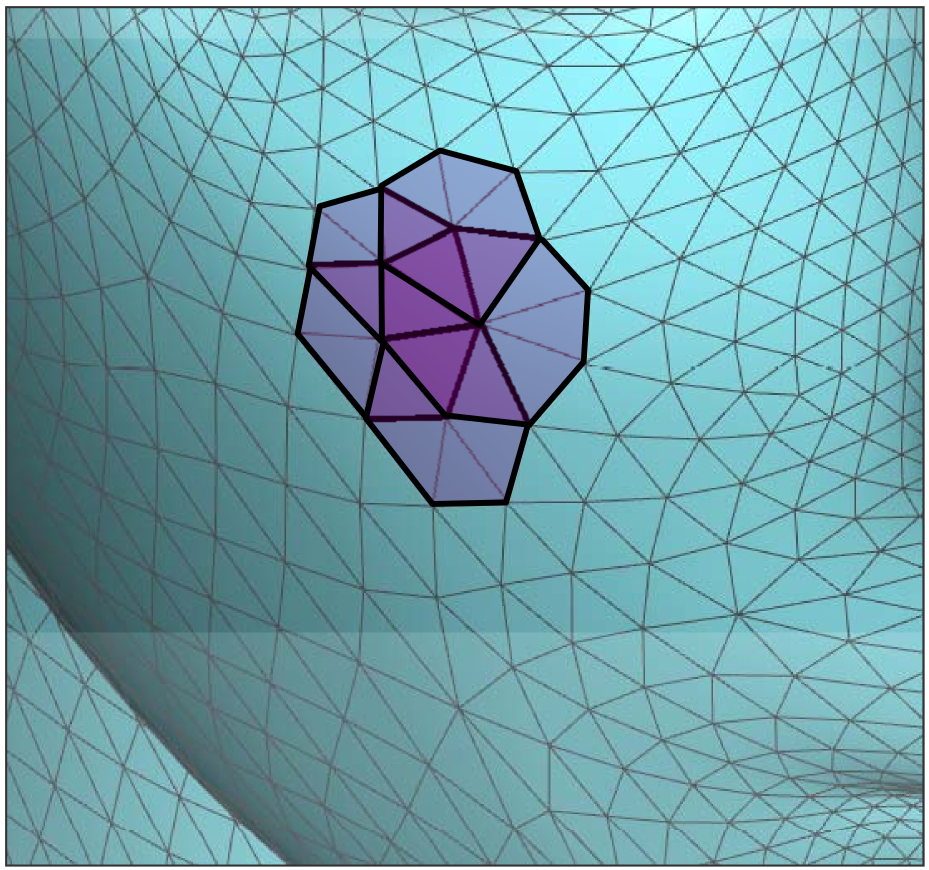
Direct ARAP modeling

- We actually may want to preserve the shapes of cells covering the surface



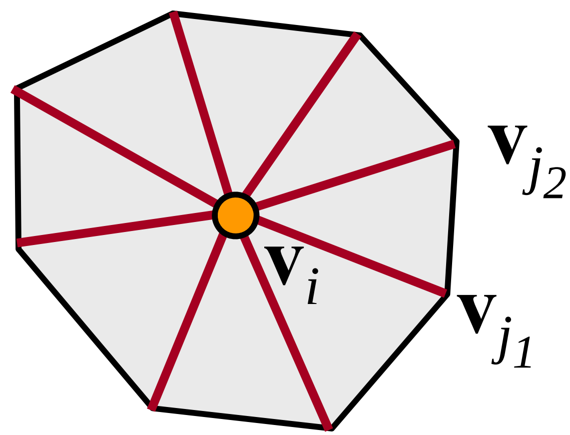
Direct ARAP modeling

- Let’s look at cells on a mesh



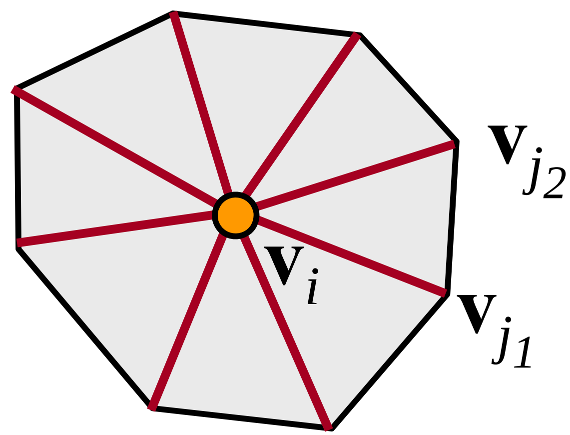
Direct ARAP modeling

- Ask all the star edges to transform rigidly, then the shape of the cell is preserved



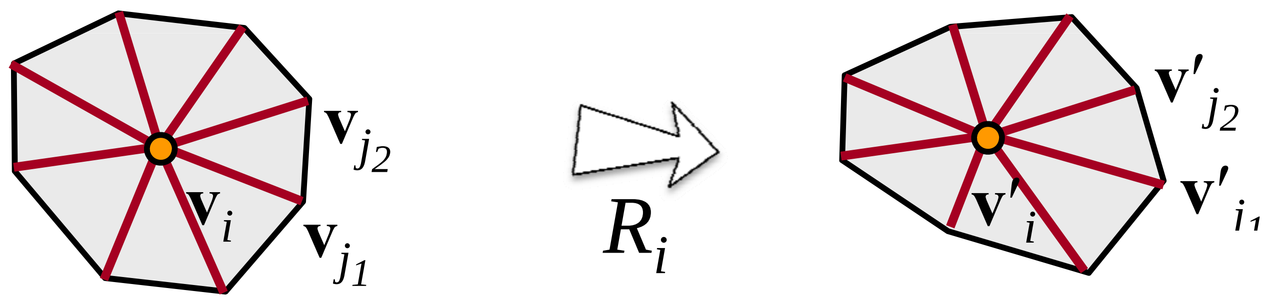
Direct ARAP modeling

- Cell energy: $\min \sum_{j \in N(i)} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - R_i(\mathbf{v}_i - \mathbf{v}_j) \right\|^2$



Direct ARAP modeling

- If $\mathbf{v}$, $\mathbf{v}'$ are known then R_i is uniquely defined



- It's the shape matching problem!
  - Build covariance matrix $\mathrm{S} = \mathrm{V}\mathrm{V}'^{\mathrm{T}}$
  - SVD: $\mathrm{S} = \mathrm{U}\Sigma\mathrm{P}^{\mathrm{T}}$
  - $R_i = \mathrm{U}\mathrm{P}^{\mathrm{T}}$

R_i is a non-linear function of $\mathbf{v}'$

Direct ARAP modeling

- Can formulate overall energy of the deformation:

$$\min_{\mathbf{v}'} \sum_{i=1}^{n} \sum_{j \in N(i)} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - R_i (\mathbf{v}_i - \mathbf{v}_j) \right\|^2$$

$$s.t.\ \mathbf{v}'_j = \mathbf{c}_j,\ j \in C$$

Energy minimization

- Alternating iterations
  - Given initial guess $\mathbf{v}'_0$, find optimal rotations R_i
    - This is a per-cell task! We already showed how to define R_i when $\mathbf{v}$, $\mathbf{v}'$ are known
  - Given the R_i (fixed), minimize the energy by finding new $\mathbf{v}'$

$$\min_{\mathbf{v}'} \sum_{i=1}^{n} \sum_{j \in N(i)} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - R_i(\mathbf{v}_i - \mathbf{v}_j) \right\|^2$$

Energy minimization

- Alternating iterations
  - Given initial guess $\mathbf{v}'_0$, find optimal rotations R_i
    - This is a per-cell task! We already showed how to define R_i when $\mathbf{v}$, $\mathbf{v}'$ are known
  - Given the R_i (fixed), minimize the energy by finding new $\mathbf{v}'$

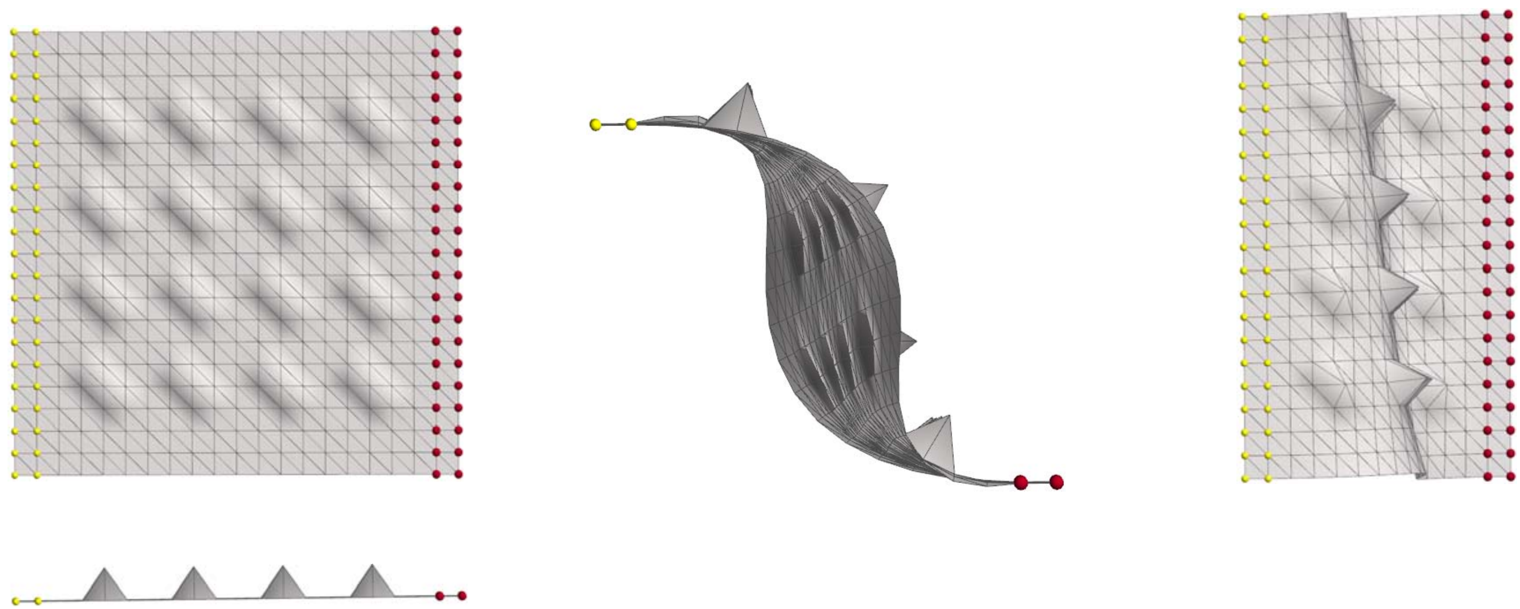
$$L\mathbf{v}' = \mathbf{b}$$

The big advantage

- Each iteration decreases the energy (or at least guarantees not to increase it!)
- The matrix L stays fixed!
  - Precompute Cholesky factorization
  - Just back-substitute each iteration (+ the SVD computations)

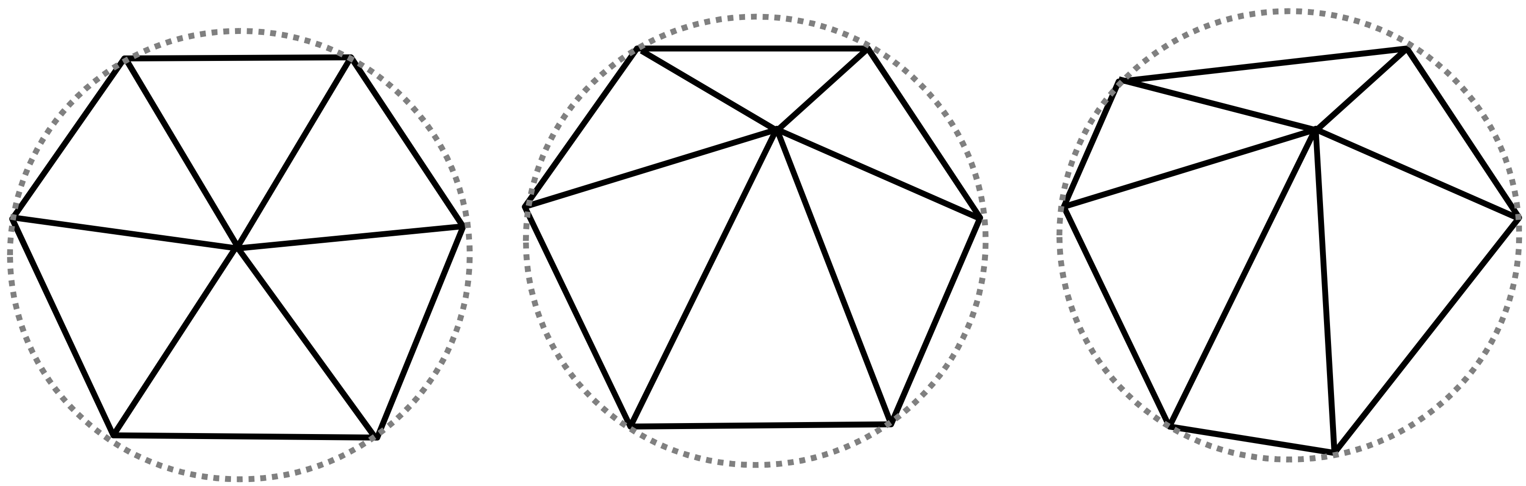
The importance of proper weighting

- If we use uniform Laplacian L



The importance of proper weighting

- The problem: need to compensate for varying shapes of the 1-ring

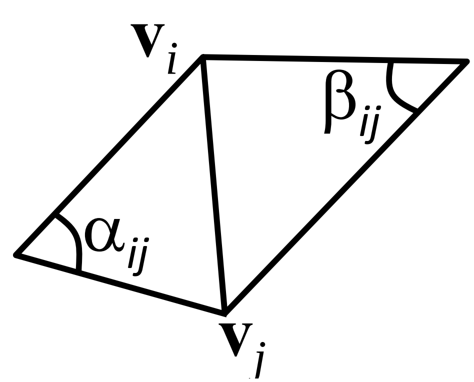


$$E_{cell} = \sum_{j \in N(i)} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - R_i(\mathbf{v}_i - \mathbf{v}_j) \right\|^2$$

Use cotan weights

- Add cotangent weights [Pinkall and Polthier 93]

$$E_{cell} = \sum_{j \in N(i)} w_{ij} \left\| (\mathbf{v}_i' - \mathbf{v}_j') - R_i (\mathbf{v}_i - \mathbf{v}_j) \right\|^2$$

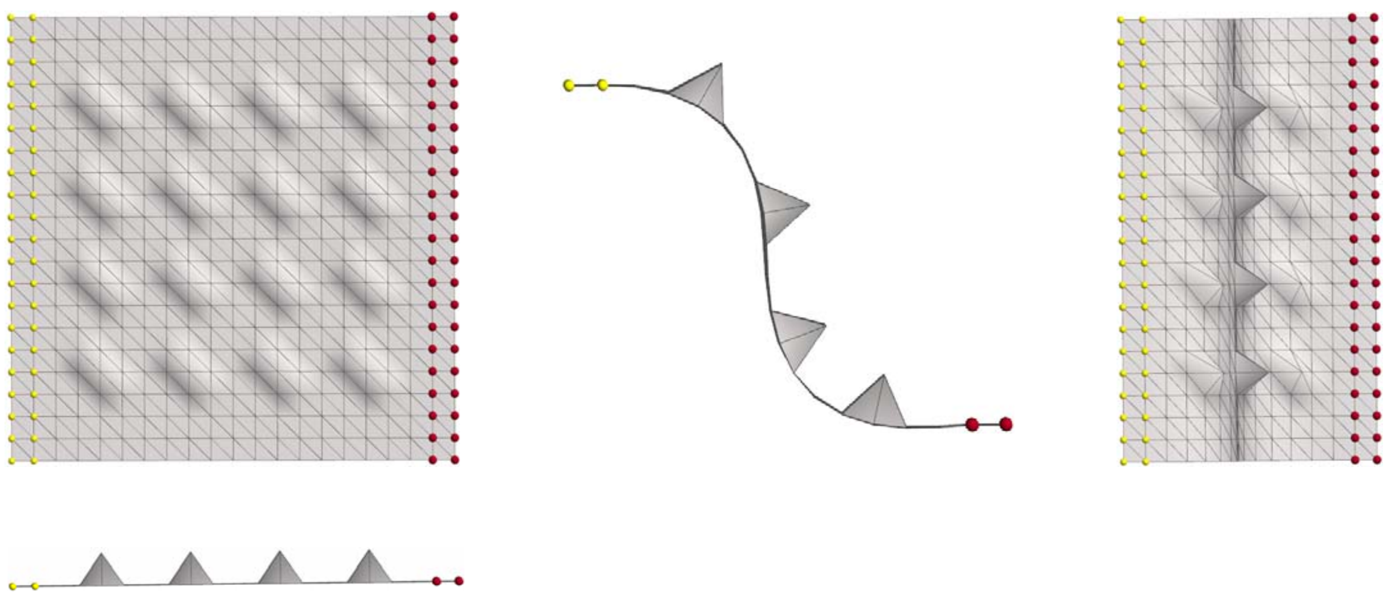


$$w_{ij} = \frac{1}{2}\left(\cot \alpha_{ij} + \cot \beta_{ij}\right)$$

Use cotan weights

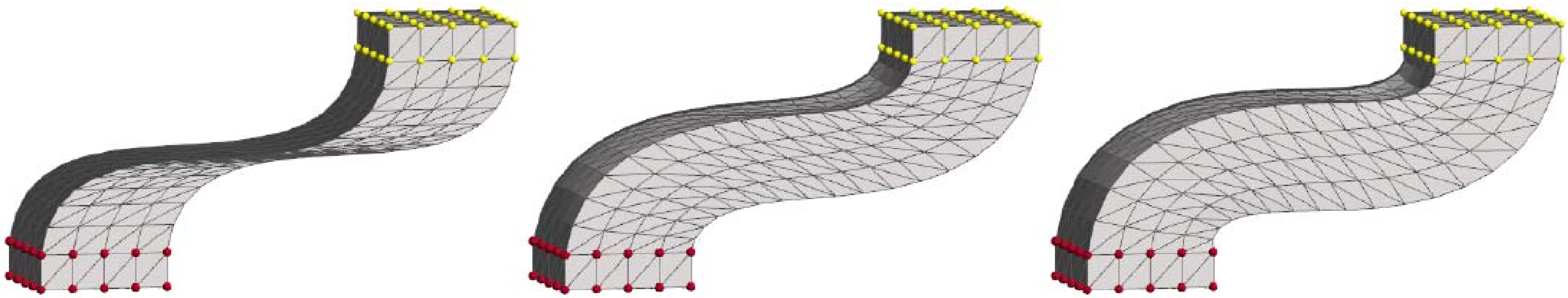
- This gives symmetric results

$$E_{cell} = \sum_{j \in N(i)} w_{ij} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - R_i (\mathbf{v}_i - \mathbf{v}_j) \right\|^2$$

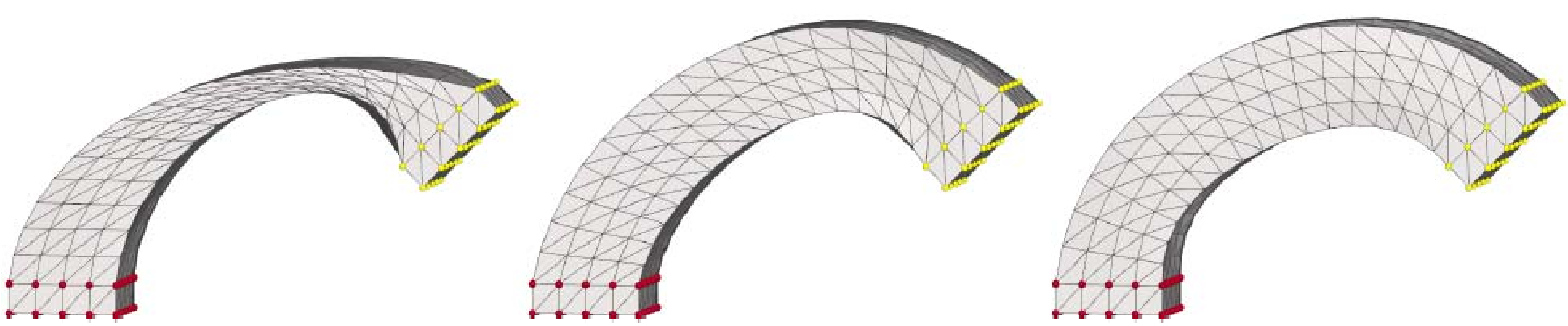


Results

- Can start from naïve Laplacian editing as initial guess and iterate



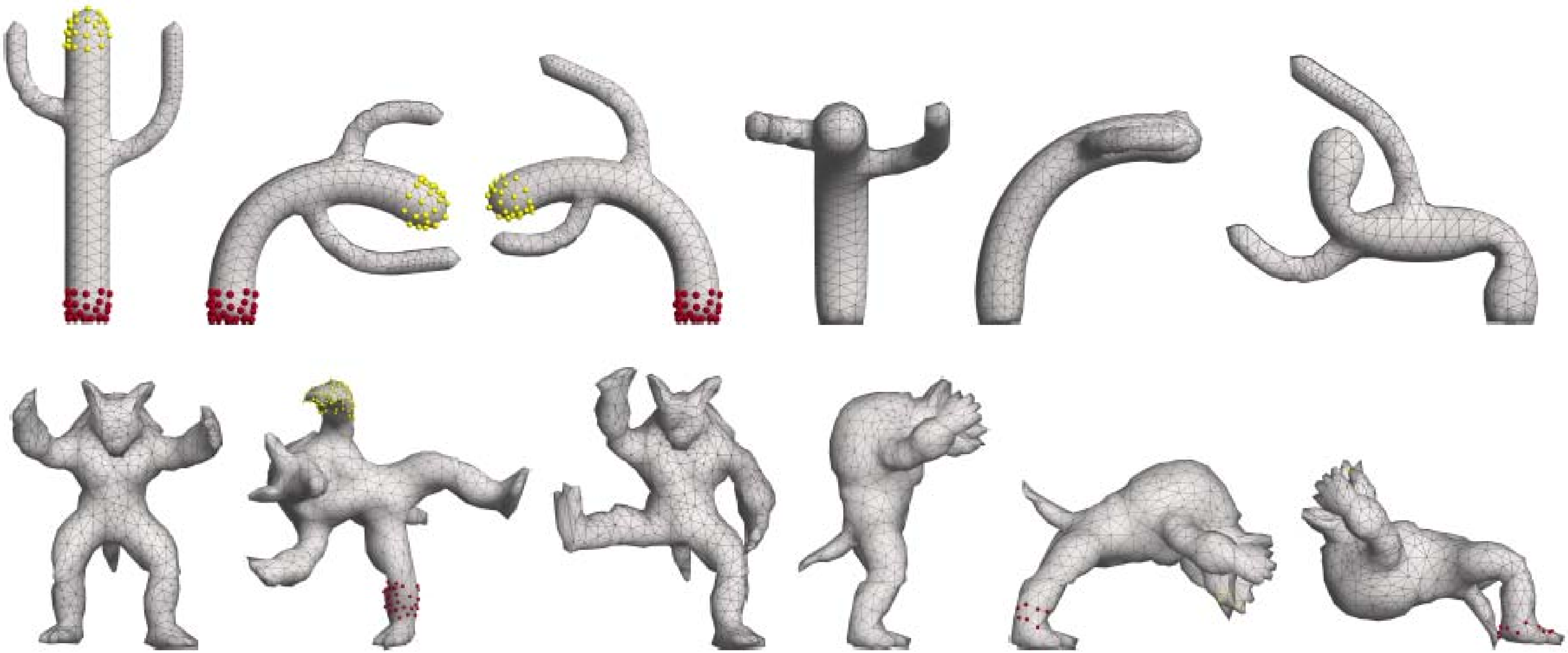
initial guess 1 iteration 2 iterations



initial guess 1 iterations 4 iterations

Results

- Faster convergence when we start from the previous frame



Issues

- Works fine on small meshes
- On larger meshes: slow convergence
  - Each iteration is more expensive of course
  - Need more iterations because the conditioning of the system becomes worse as the matrix grows
- Implement multi-res strategy?
- Also: material stiffness depends on the 1-ring size (lots of wrinkles for fine meshes)

More issues

- This technique is good for preserving edge length (relative error very small)
- No notion of volume, however
  - Essentially, thin shells for the poor
- Can extend to volumetric meshes