Linear Rotation-invariant Coordinates for Meshes

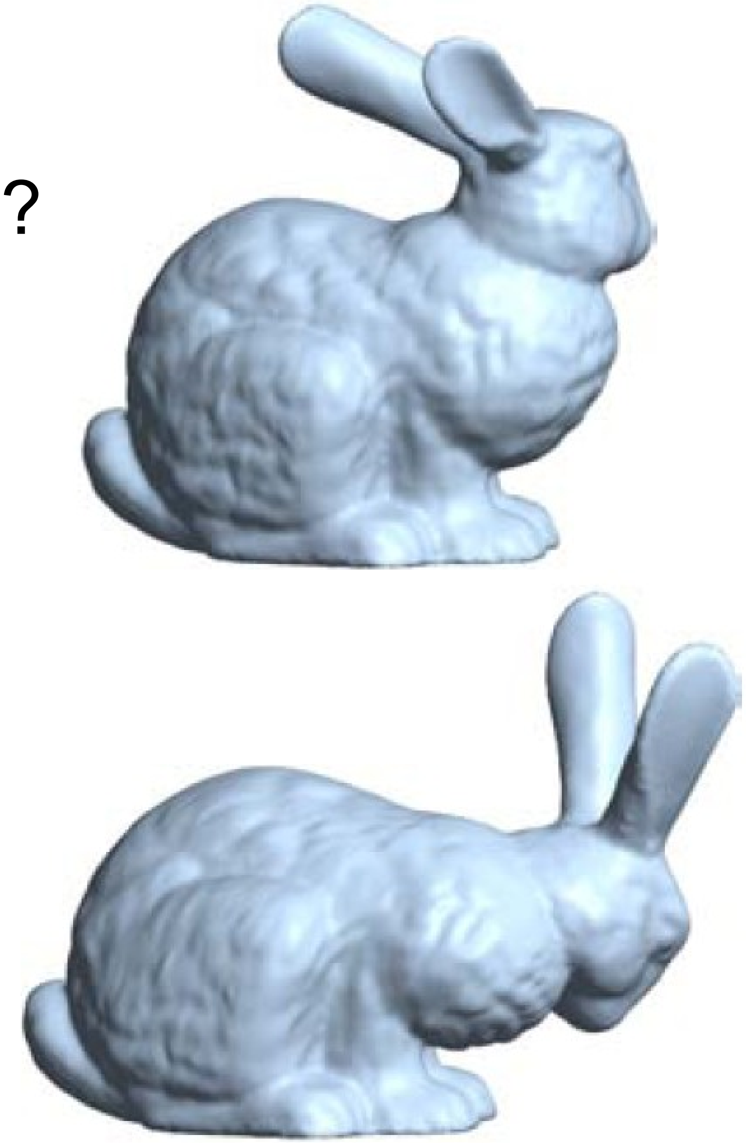
Yaron Lipman, Olga Sorkine, David Levin, Daniel Cohen-Or

Tel Aviv University

ACM SIGGRAPH 2005

Outline

- Motivation
  - What's the problem, anyway?
  - Some approaches
    - Multiresolution methods
    - Local frames
- Method
  - Discrete differential forms
- Results



Problem

- Mesh requires local deformations
- Macro changes must preserve micro detail
- Deformations must be smooth and “intuitive”
  - Preserve distance, area, curvature etc as far as possible
- Interactive process: select a handle and transform it – the rest of the model should follow

Previous work

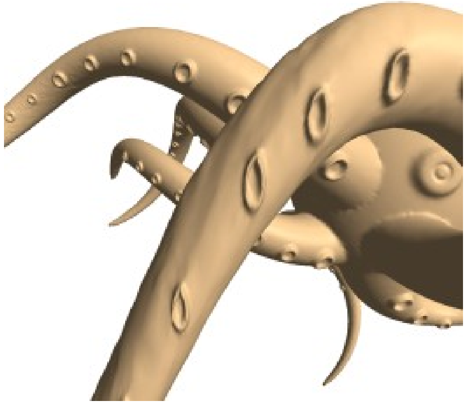
- Multiresolution methods
  - Break up model into various levels of detail
  - Transform one level while keeping others invariant
    - Zorin et al. '97
    - Kobbelt et al. '99
    - Guskov et al. '99
    - Botsh and Kobbelt '04
  - Problem: requires manual setting of level thresholds

Previous Work

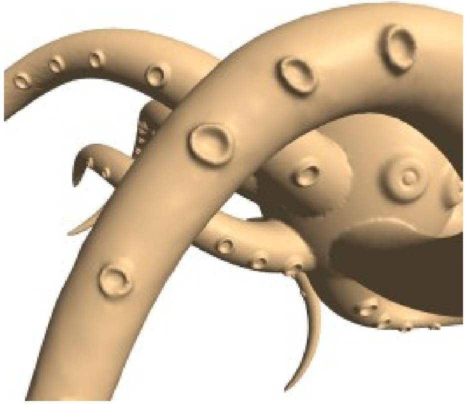
- Preserve differential properties
  - Explicit thresholding not required
  - One approach: consider global coordinates but ensure that local frames are correctly updated
    - Implicitly include local frame data in Laplacian fitting scheme [Sorkine et al. 2004]
    - Propagate user-defined transformation of “handle” [Yu et al. 2004, Zayer et al. 2005]
    - Heuristically approximate local rotations [Lipman et al. 2004]
  - Problem: local quantities not rotation invariant, so patches must be... umm... patched

We need...

- A way to store the mesh that is invariant under all rigid transformations
- Why?
  - When a joint is bent, the limbs (pseudo-rigid) should preserve their properties although their spatial orientation has changed
- So global coordinates, in fact anything that stores absolute or relative position ***vectors***, are out

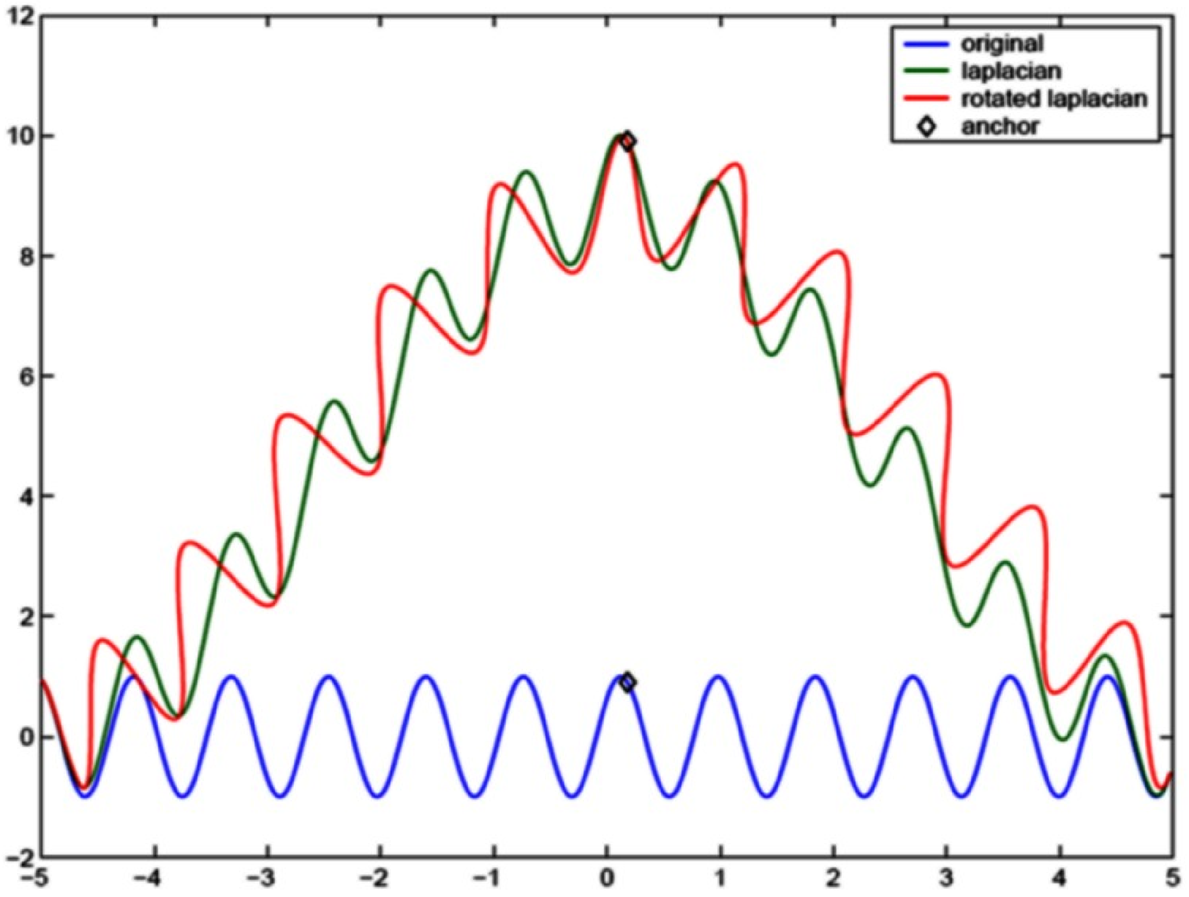


Wrong



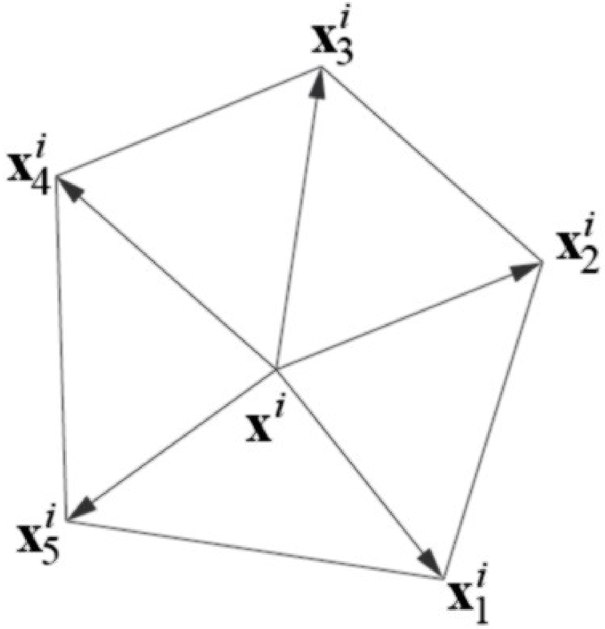
Right

Example: disaster recovery

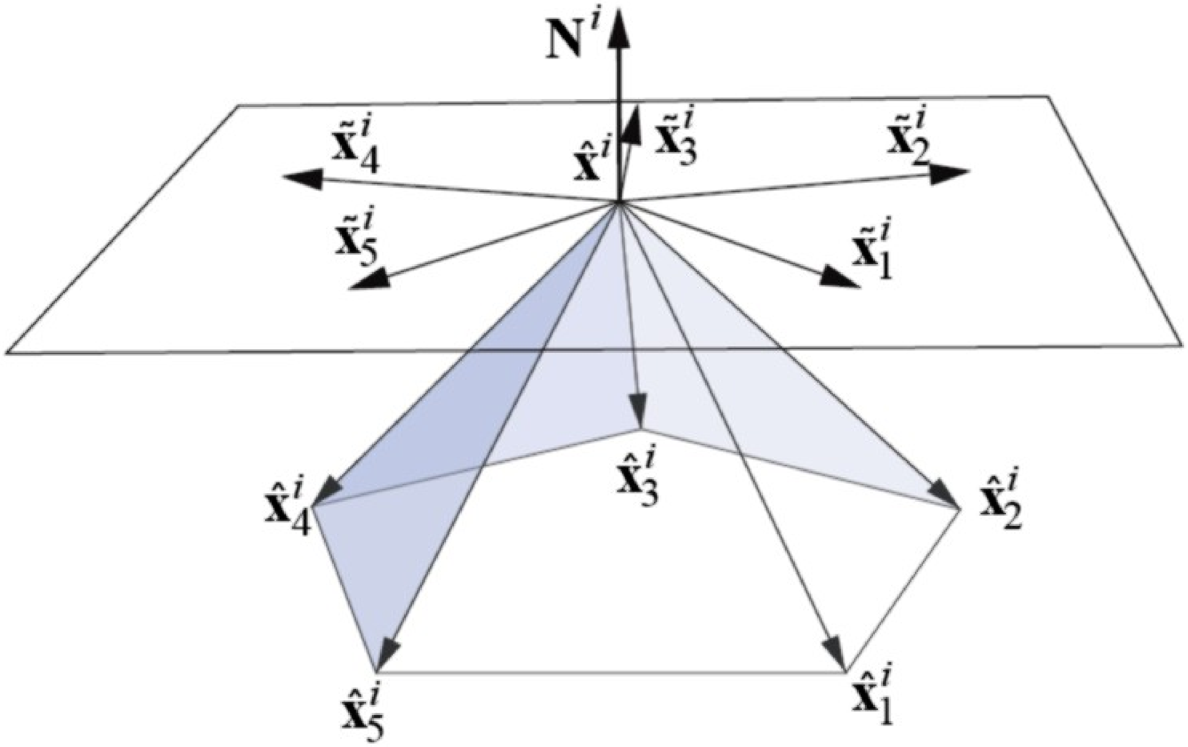


Ideally, we shouldn't need this

Discrete forms



parameter space



embedded manifold

Discrete forms

Parametrization $\mu = \mu_1 \, \mathbf{x}^i_k + \mu_2 \, \mathbf{x}^i_{k+1} \in \triangle^i_k$

1st and 2nd forms $\widetilde{I}^{\,i}(\cdot)\ ,\ \widetilde{II}^{\,i}(\cdot) : \bigcup_{k=1}^{d_i-1} \triangle^i_k \longrightarrow \mathbb{R}$

$$\widetilde{I}^{\,i}(\mu) = \langle \mu , \mu \rangle_{\mathbb{R}^3} = \mu_1^2 \boxed{\widetilde{g}^{\,i}_{k,k}} + 2\,\mu_1\,\mu_2 \boxed{\widetilde{g}^{\,i}_{k,k+1}} + \mu_2^2 \boxed{\widetilde{g}^{\,i}_{k+1,k+1}}$$

$\langle \widetilde{\mathbf{x}}^i_k , \widetilde{\mathbf{x}}^i_k \rangle_{\mathbb{R}^3}$

$\langle \widetilde{\mathbf{x}}^i_k , \widetilde{\mathbf{x}}^i_{k+1} \rangle_{\mathbb{R}^3}$

$$\widetilde{II}^{\,i}(\mu) := \mu_1 \, \langle \hat{\mathbf{x}}^i_k , \, \mathbf{N}^i \rangle_{\mathbb{R}^3} + \mu_2 \, \langle \hat{\mathbf{x}}^i_{k+1} , \, \mathbf{N}^i \rangle_{\mathbb{R}^3} = \mu_1 \boxed{\widetilde{L}^i_k} + \mu_2 \boxed{\widetilde{L}^i_{k+1}}$$

Also $\boxed{O^i_k} := \operatorname{sign}\left(\det(\widetilde{\mathbf{x}}^i_k , \, \widetilde{\mathbf{x}}^i_{k+1} , \, \mathbf{N}^i)\right)$ to store direction of normal

Discrete forms

- First DF – geometry in tangent plane
  - quadratic in each triangle
  - C^0 continuity between adjacent triangles
- Second DF – geometry perp. to tangent plane
  - linear (“height above tangent plane”)
- The coefficients (DFC's) depend on:
    - vertex angles
    - edge lengths
  - This is nice – preserve these quantities and you're likely to preserve size, curvature etc

Key Point #1: Local Reconstruction

- The geometry at each vertex (upto a rigid transformation) can be computed from the discrete form coefficients ($g_{k,k}$, $g_{k,k+1}$, L_k, O_k)
  - locally, we can compute:
    - the neighbours of a vertex
    - the normal at the vertex
  - Basic idea: fix the vertex, an edge incident on it, and the normal (rigidity); now generate neighbours successively

$$\begin{tikzcd} \mathbf{x}_1 \arrow[r, "{\substack{g_{11}, g_{12}, g_{22}, \\ L_2, O_1}}"'] & \mathbf{x}_2 \arrow[r, "{\substack{g_{22}, g_{23}, g_{33}, \\ L_3, O_2}}"'] & \mathbf{x}_3 & \ldots \end{tikzcd}$$

Key Point #2: Global Reconstruction

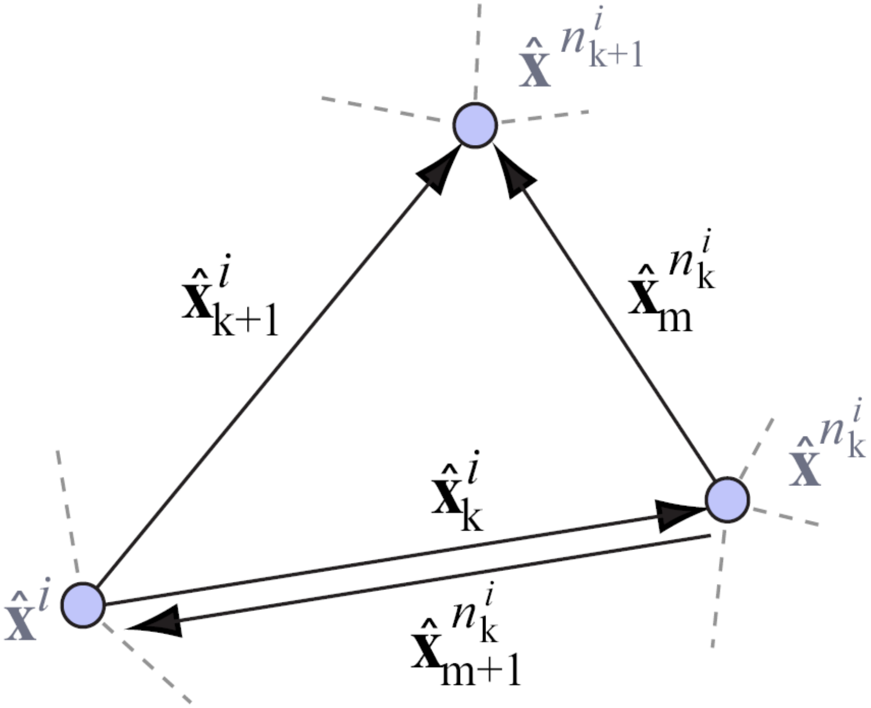
- Discrete form coefficients uniquely define the entire mesh (again upto a RT)
- Basic idea: discrete surface equations:

$$\begin{aligned}
\mathbf{b}_1^j &= \left(\Gamma_{j,1}^{i,1}+1\right)\mathbf{b}_1^i+\Gamma_{j,1}^{i,2}\mathbf{b}_2^i+A_{j,1}^i\mathbf{N}^i \\
\mathbf{b}_2^j &= \Gamma_{j,2}^{i,1}\mathbf{b}_1^i+\left(\Gamma_{j,2}^{i,2}+1\right)\mathbf{b}_2^i+A_{j,2}^i\mathbf{N}^i \\
\mathbf{N}^j &= \Gamma_{j,3}^{i,1}\mathbf{b}_1^i+\Gamma_{j,3}^{i,2}\mathbf{b}_2^i+\left(A_{j,3}^i+1\right)\mathbf{N}^i
\end{aligned}$$

  - Now dealing with $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{N}$, not direct vertex geometry, but the two are equivalent

Key Point #2: Global Reconstruction

- The coefficients in the discrete surface eqns are functions of the DFC's
  - How?
    - the eqns express one frame in terms of an adjacent frame
    - local frame at one vertex, by construction, has info about neighbours
    - do some algebra and voila!



$\hat{\mathbf{x}}^{n^i_{k+1}}$
$\hat{\mathbf{x}}^i_{k+1}$
$\hat{\mathbf{x}}^{n^i_k}_{m}$
$\hat{\mathbf{x}}^{n^i_k}$
$\hat{\mathbf{x}}^i_k$
$\hat{\mathbf{x}}^i$
$\hat{\mathbf{x}}^{n^i_k}_{m+1}$

Key Point #2: Global Reconstruction

- Put all disc. surface eqns together to get a
    - huge,
    - overdetermined
    - sparse
    - linear

  system in b's and N's
- Solve to get local frame at each vertex
- Could fix a vertex, generate its neighbours, and branch out until all vertices are covered. But this amplifies errors.

Key Point #2: Global Reconstruction

- Better: if the frame of vertex i has been determined, then the tangent-plane projections of all its neighbours can be found:

  $\widetilde{\mathbf{x}}_1^i = \mathbf{b}_1^i (\widetilde{g}_{1,1}^i)^{1/2}$, and cyclically generate others

- Now set up another linear system as:

$$\hat{\mathbf{x}}^j - \hat{\mathbf{x}}^i = \widetilde{\mathbf{x}}_k^i + \widetilde{L}_k^i \mathbf{N}^i = \langle \widetilde{\mathbf{x}}_k^i, \mathbf{b}_1^i \rangle \mathbf{b}_1^i + \langle \widetilde{\mathbf{x}}_k^i, \mathbf{b}_2^i \rangle \mathbf{b}_2^i + \widetilde{L}_k \mathbf{N}^i$$

- Solve this to get the vertex positions

Back to deformations...

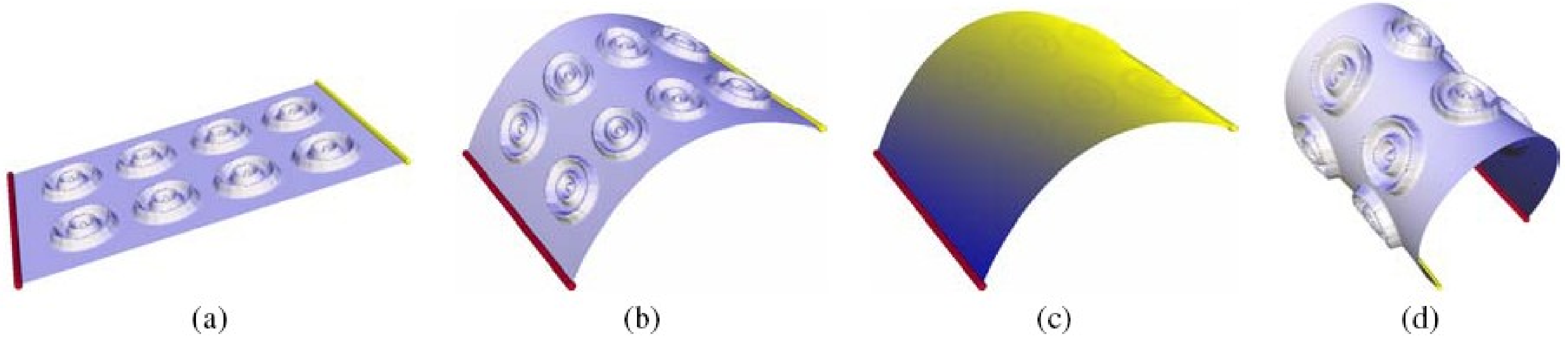
- Observe: surface eqns form linear system, and so do the position difference eqns.
- Add constraints on frames and positions as required:
  - Linear transformation (e.g. rotation, scale): surface (frame) eqns
  - Translation: position eqns
- Now overdetermined augmented system may not have a solution!

Deformations

- Get least squares solutions of linear systems

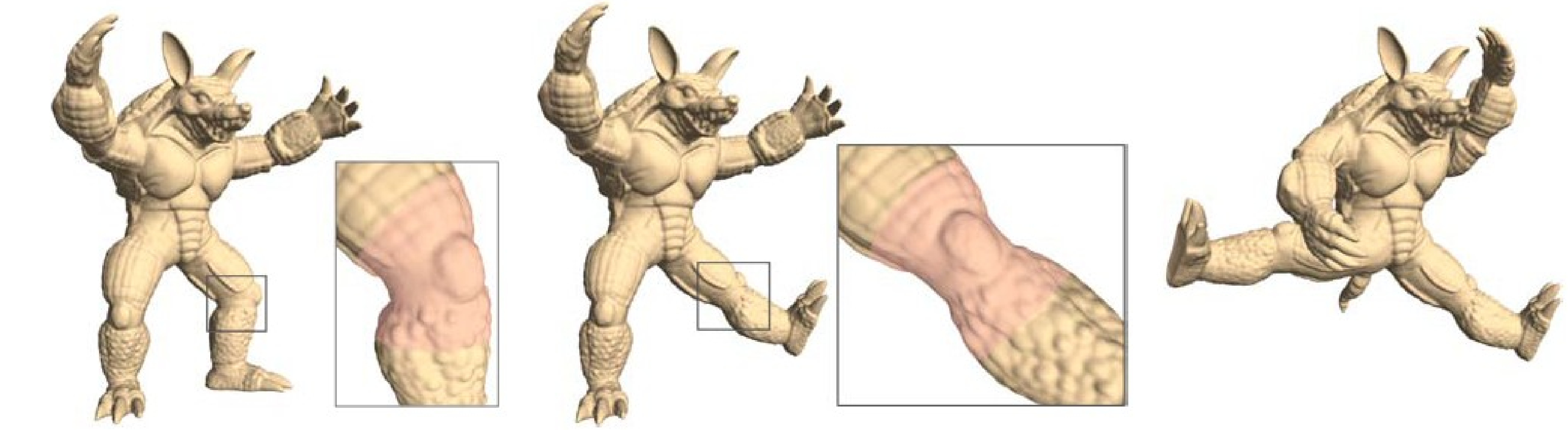
  (recall: construct normal eqns: $Ax = B \longrightarrow A^T Ax = A^T B$)

- This usually does the job, assuming deformations are not too large

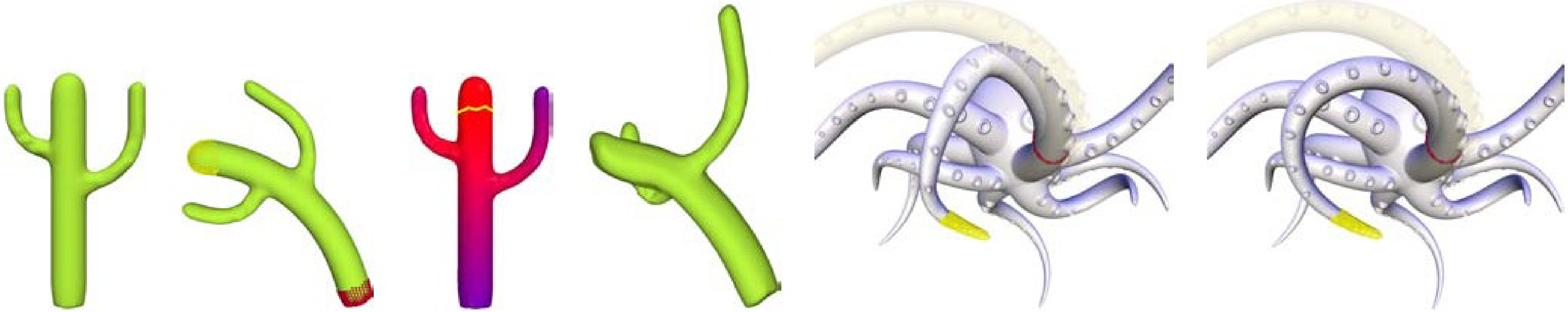


(a) (b) (c) (d)

More examples

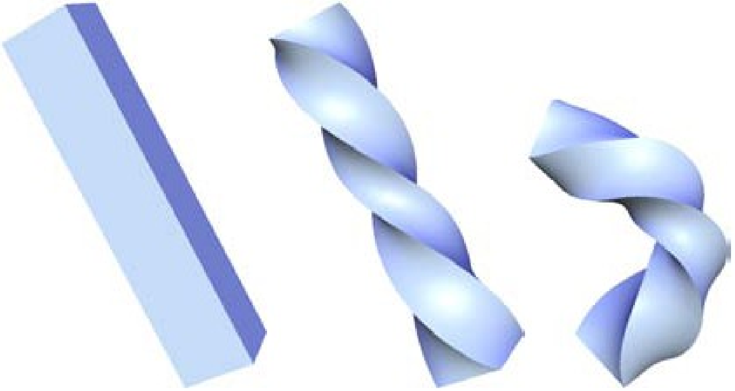


(a) (b) (c)



(a) (b) (c) (d) (e) (f)

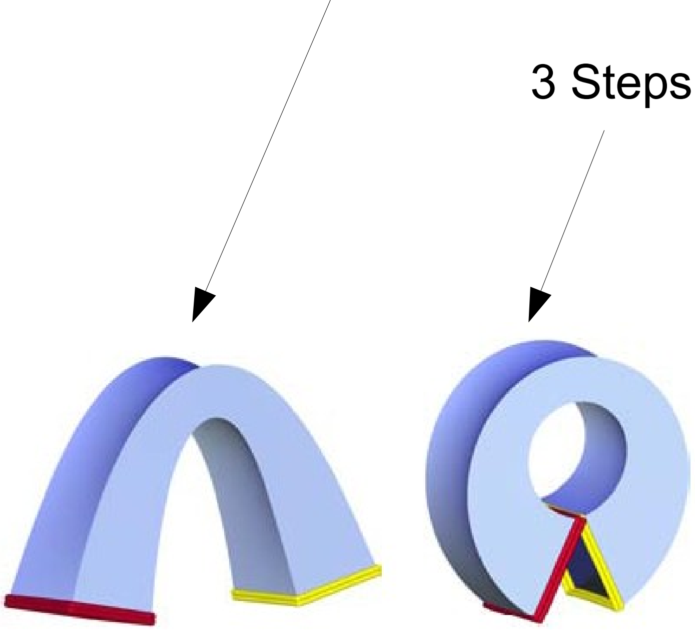
More examples



Sharp edges are preserved

170°

3 Steps to 315°



Connections

- Differential geometry: fundamental forms
  - first $E = \mathbf{x}_u . \mathbf{x}_u, \quad F = \mathbf{x}_u . \mathbf{x}_v, \quad G = \mathbf{x}_v . \mathbf{x}_v,$
  - second $L = \mathbf{N} . \mathbf{x}_{uu}, \quad M = \mathbf{N} . \mathbf{x}_{uv}, \quad N = \mathbf{N} . \mathbf{x}_{vv}.$
- Characterize tangential and normal properties
- Approximate local reconstruction
  - Bonnet theorem ensures surface can be reconstructed given the fundamental forms which hold Gauss-Codazzi-Mainardi conditions
  - Two stages: i) changes in local frames, ii) frames are integrated

Connections

- Given k_g (geodesic curv.), k_n (normal curv.), t_r (relative torsion) and a curve parametrized by s:

$$\frac{d}{ds}\begin{pmatrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \mathbf{N}\end{pmatrix} = \begin{pmatrix}0 & k_g & k_n\\ -k_g & 0 & t_r\\ -k_n & -t_r & 0\end{pmatrix}\begin{pmatrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \mathbf{N}\end{pmatrix}$$

- Compare with discrete surface eqns
- So at least intuitively, this method preserves the curvature quantities as much as possible