Dual Laplacian Manipulation for Triangular Meshes

Ligang Liu
Zhejiang University, China

Sep. 15, 2007

Strobl, Austria

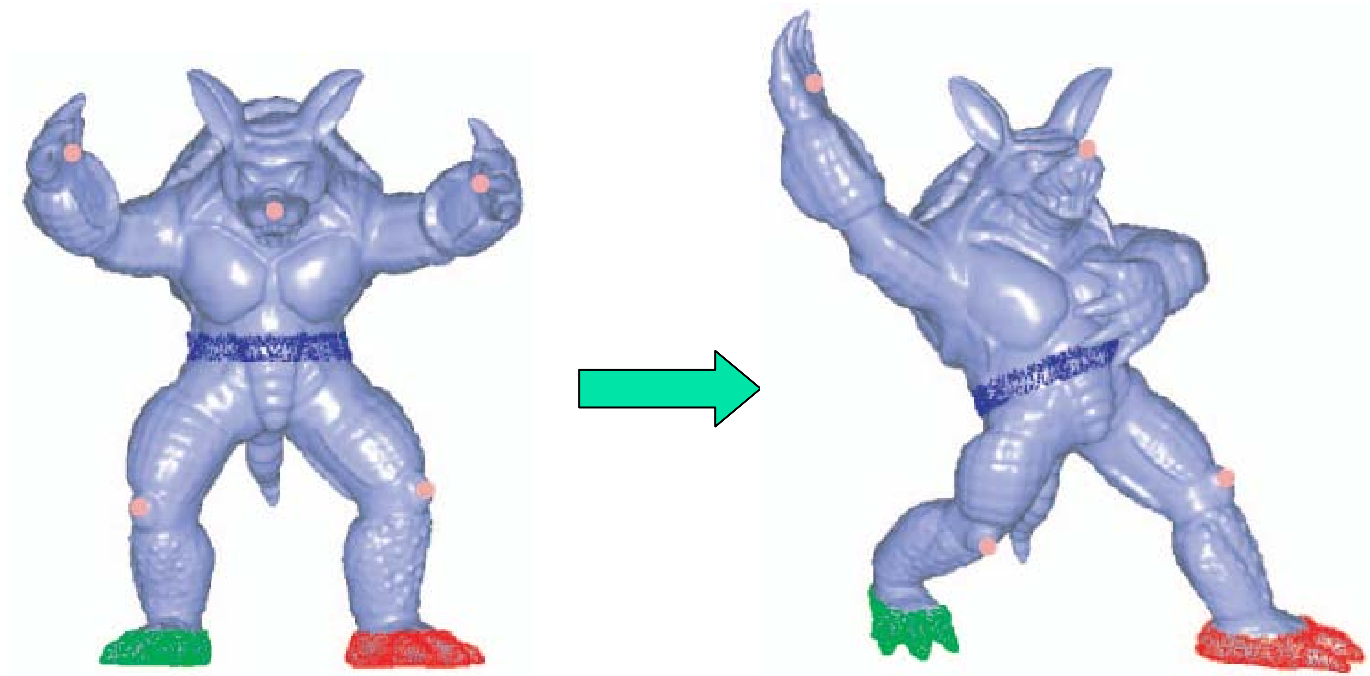
Outline

- Problem
- Linear iterative framework
- Dual mesh editing
- Dual mesh morphing
- Other manipulations
- Conclusions

Problem

Shape Deformation

- To deform/edit the surface as you imagine in your mind



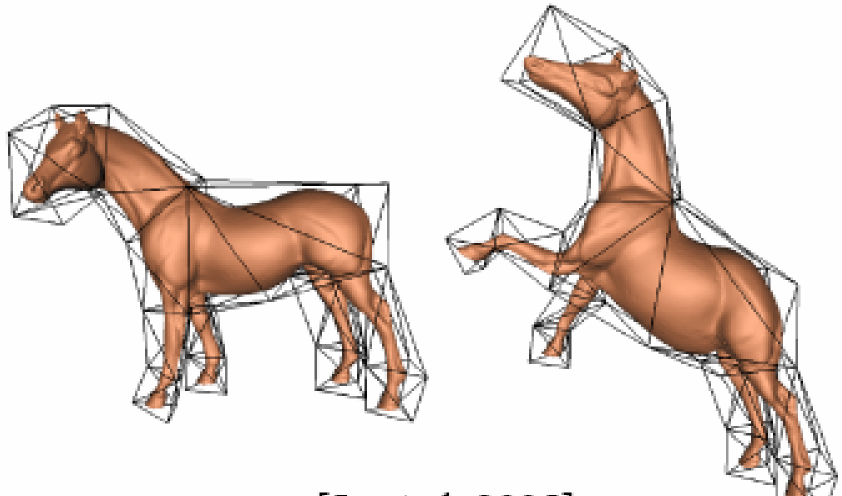
Input Output

Related Work

- Free-form deformation
  - [Sederberg and Parry, 1988]
  - [Lazarus et al., 1994]
  - [Ju et al., 2005]
  - [Pauly et al., 2006,2007]
- Multi-resolution editing
  - [Eck et al., 1995]
  - [Kobbelt et al., 1998]
  - [Xu et al., 2006]
- Differential surface editing
  - [Alexa, 2003]
  - [Sorkine et al., 2004]
  - [Yu et al., 2004]
  - [Sheffer and Krayevoy, 2004]

Free-form Deformation (Embedded Deformation)

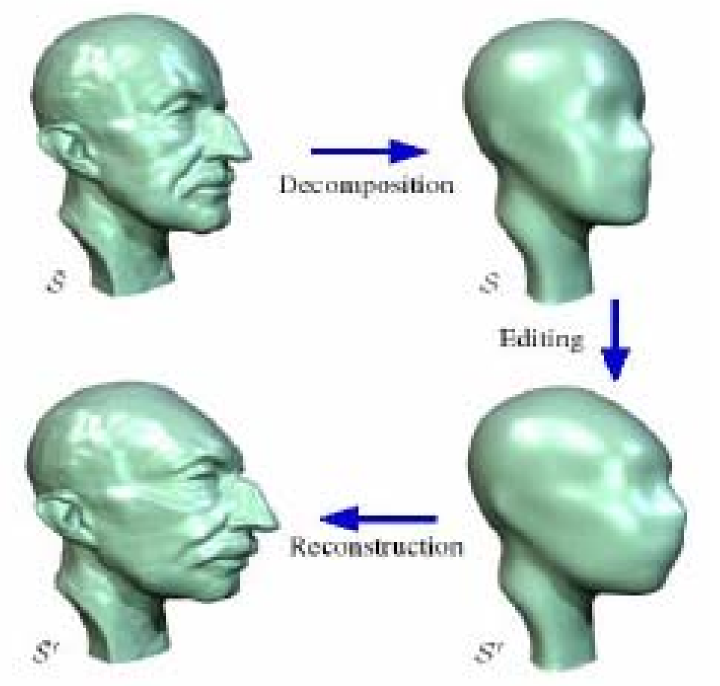
- Manipulated by proxy mesh
- Preserving
  - Parameters of vertices
- Pros
  - Simple, intuitive
- Cons
  - Loss of details



[Ju et al. 2005]

Multi-resolution Editing

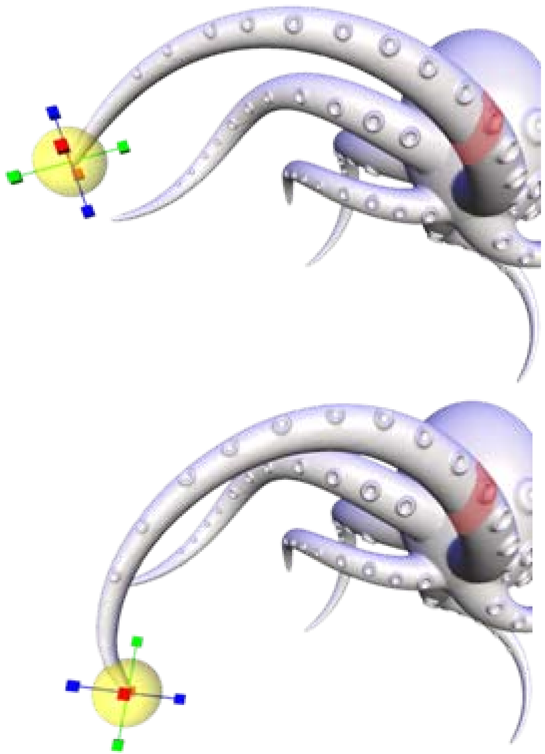
- Manipulated by simplified mesh
- Preserving
  - Detail encoding
- Pros
  - Scalable
- Cons
  - Unstable



[Botsch & Kobbelt 2003]

Differential Surface Editing

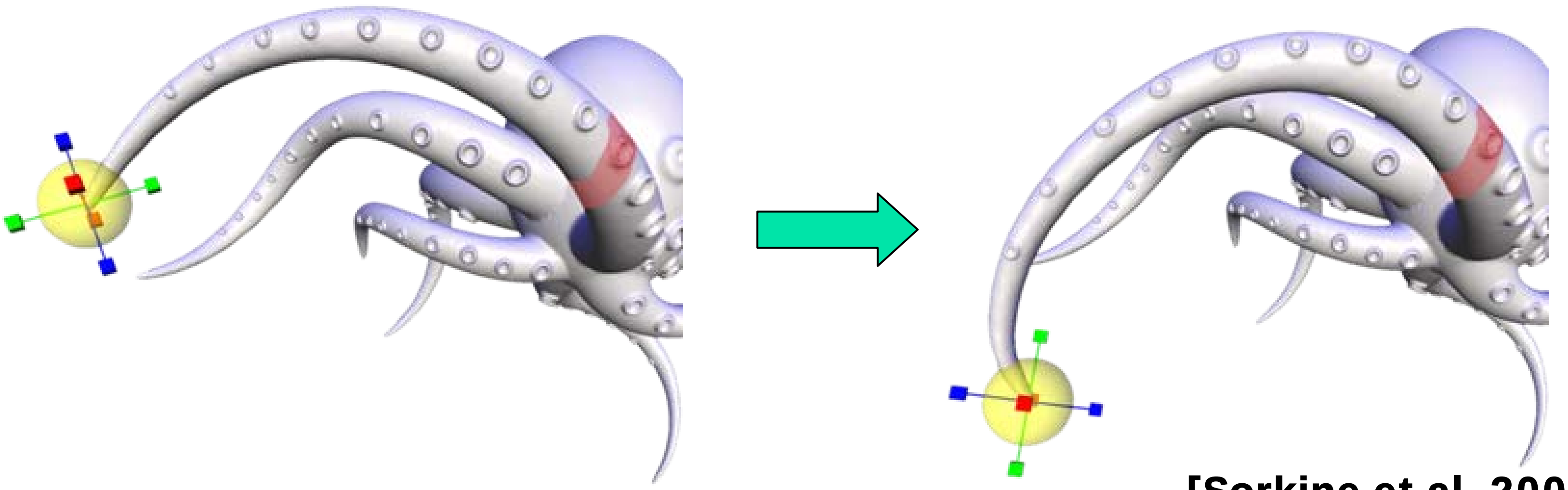
- Manipulated by user handles
- Preserving
  - Differential information
- Pros
  - Detail preserving
- Cons
  - Computational cost



[Sorkine et al. 2004]

Handle-based User Interfaces

- Select some part as handle
- Drag and move the handle
- Deform the surface

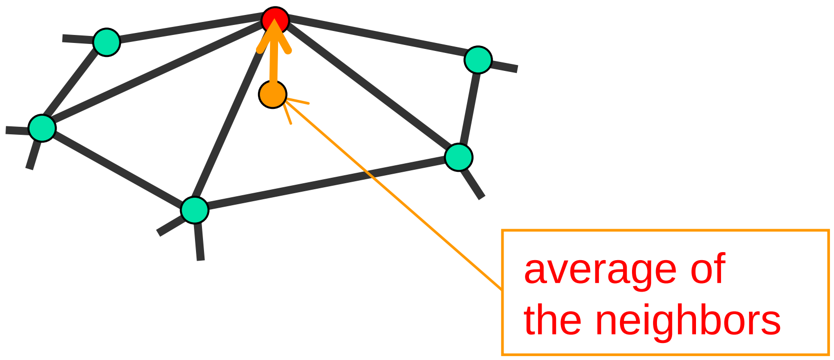


[Sorkine et al. 2004]

Laplace Coordinates (LC) or Laplace Vector (LV)

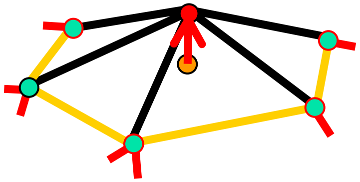
- Differential coordinates are defined by the discrete Laplacian operator:

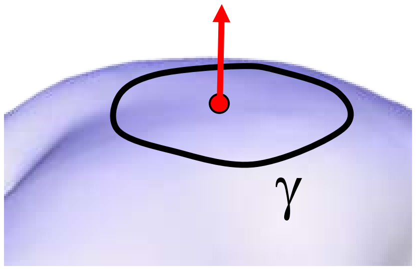
$$\delta_i = v_i - \sum_{j \in N(i)} w_j v_j$$



View of Differential Geometry

- Discretization of Laplace-Beltrami operator





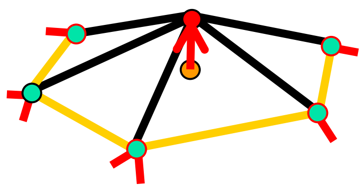
$$\boldsymbol{\delta}_{\mathbf{i}} = \frac{1}{d_i} \sum_{\mathbf{v} \in N(i)} \left(\mathbf{v}_{\mathbf{i}} - \mathbf{v} \right)$$

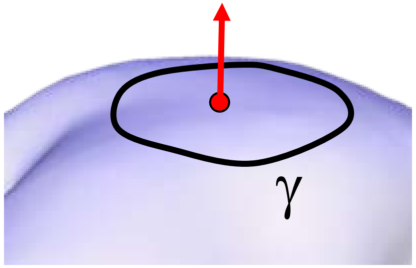
$$\frac{1}{len(\gamma)} \int_{\mathbf{v} \in \gamma} \left(\mathbf{v}_{\mathbf{i}} - \mathbf{v} \right) ds$$

$$\lim_{len(\gamma) \to 0} \frac{1}{len(\gamma)} \int_{\mathbf{v} \in \gamma} \left(\mathbf{v}_{\mathbf{i}} - \mathbf{v} \right) ds = H(\mathbf{v}_{\mathbf{i}}) \mathbf{n}_{\mathbf{i}}$$

Geometric Meaning

- LCs represent the **local** detail / local shape description
  - The direction approximates the normal
  - The size approximates the mean curvature





Laplacian Surface Editing

[Sorkine et al. 2004]

- Compute differential representation

$$\Delta = L(V)$$

- Pose modeling constraints

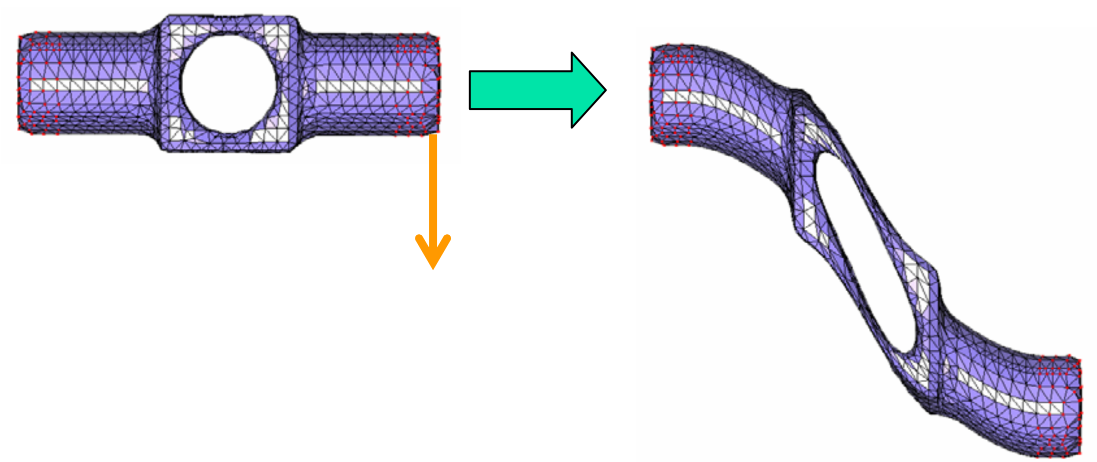
$$\mathbf{v}_i' = \mathbf{u}_i, \quad i \in C$$

- Reconstruct the surface in a least-squares sense

$$\tilde{V}' = \arg\min_{V'} \left(\left\| L(V') - \Delta \right\|^2 + \sum_{i \in C} \left\| \mathbf{v}_i' - \mathbf{u}_i \right\|^2 \right)$$

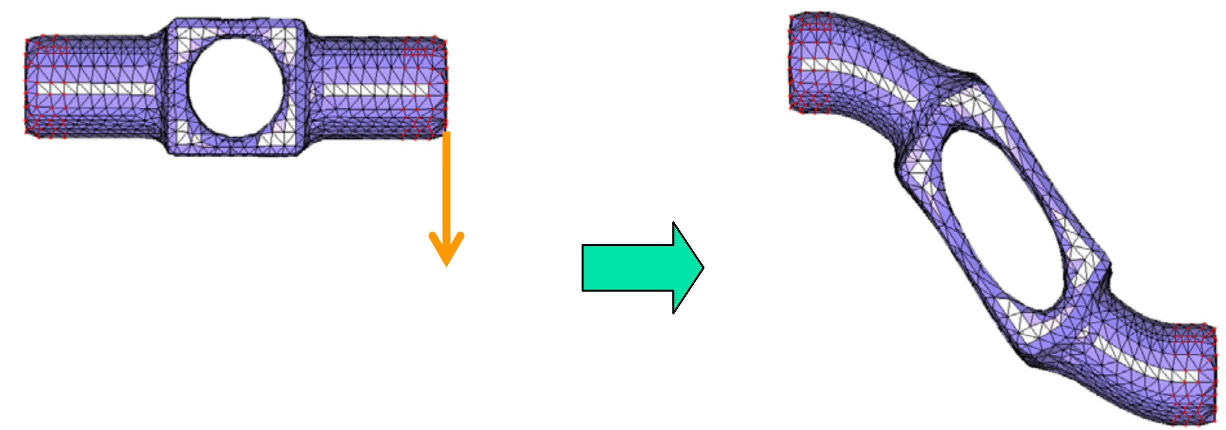
Transformation Problem

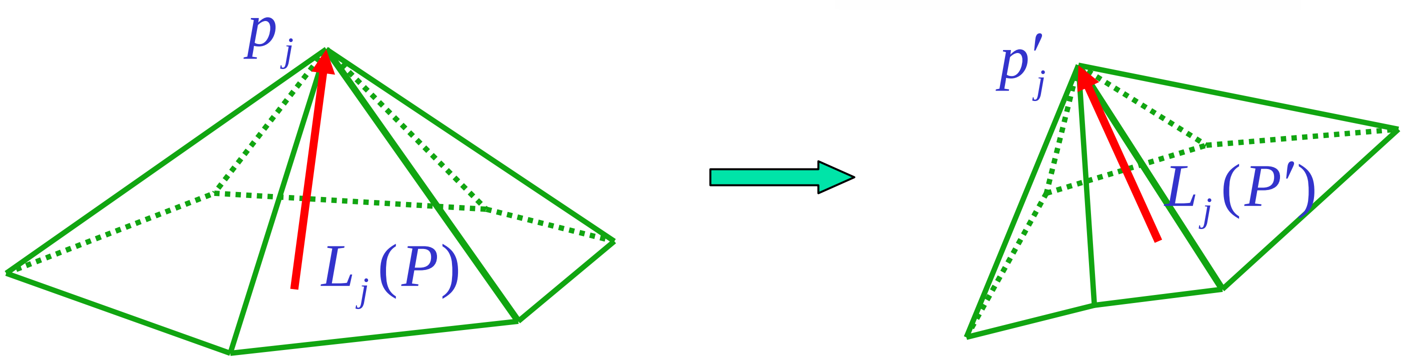
- The LCs are encoded in global coordinate system
  - Local structures of deformed surface may be rotated
  - Minimizing changes from LCs of original mesh is not appropriate
- Large distortion and stretch!



Transformation Problem

- The LCs should be properly reoriented





Transformation Problem

- Several researches attempted to solve it
  - [Lipman et al. 2004] used an intermediate reconstructed surface to guess the new orientation of the LCs
  - [Sorkine et al. 2004] employed an implicitly defined transformation onto each LC
  - [Yu et al. 2004] propagated the changes in the rotation and scaling of the handles to all the unconstrained vertices
  - [Zayer et al. 2004] propagated the transformations along harmonic field
  - [Lipman et al. 2005] encodes the vertex difference in local frames
  - [Sheffer and Krayevoi 2004] proposed pyramid coordinates to encode local features
- These methods only solve the problem partially
  - Have their limitations
  - Do not measure the quality of deformation

Transformation Problem

- Basically a chicken-and-egg problem
  - Do not know the deformed mesh before solving the linear system
  - Solving the linear system needs the properly reoriented LCs, which depend on the deformed mesh
- Can not be solved satisfactorily using only linear system as direct solvers

Linear Iterative Framework

[Joint work with Au et al.]

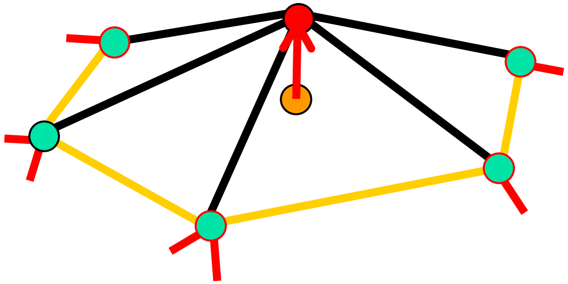
Observations

- The deformed mesh should have
  - Similar triangle shapes as the original mesh
    - Preserve parameterization information (i.e., shapes of local features)
    - Shape distortion causes undesired shearing and stretching
  - Similar local feature sizes as the original mesh
    - Preserve geometry information (i.e., sizes of local features)

Observations on LCs

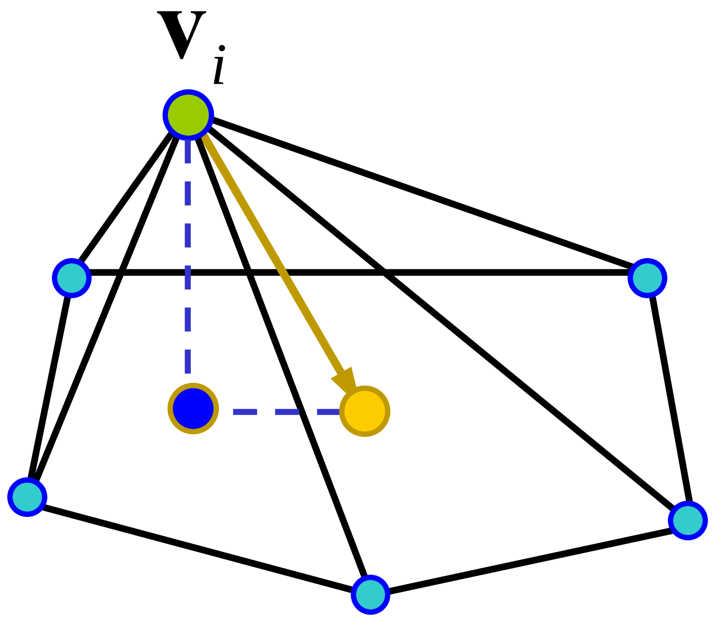
- Parameterization information
  - Captured by the coefficients of the Laplacian operator
- Geometry information
  - Captured by the magnitudes of LCs

$$\boxed{\boldsymbol{\delta}_i} = \sum_{j \in Neigh(i)} \boxed{\omega_j} (\mathbf{v}_i - \mathbf{v}_j)$$



Observations on LCs

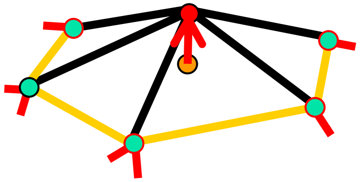
- LCs have both normal and tangential components

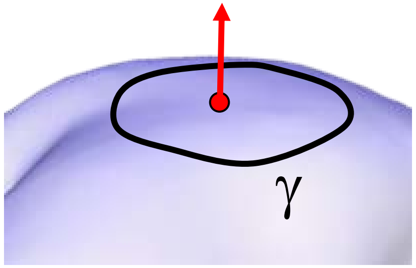


Weights

- Choices of weights of LCs affect the approximation quality of the surface normal

$$\boldsymbol{\delta}_i = \sum_{j \in Neigh(i)} \boxed{\omega_j} (\mathbf{v}_i - \mathbf{v}_j)$$





Cotangent Weight Scheme

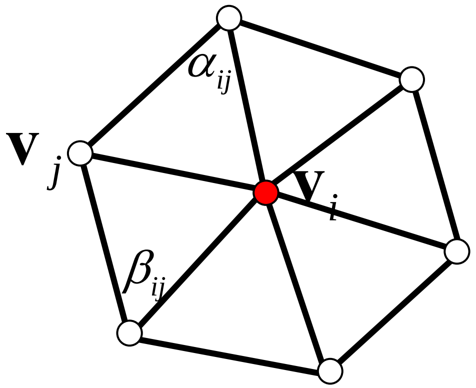
[Meyer et al. 2002]

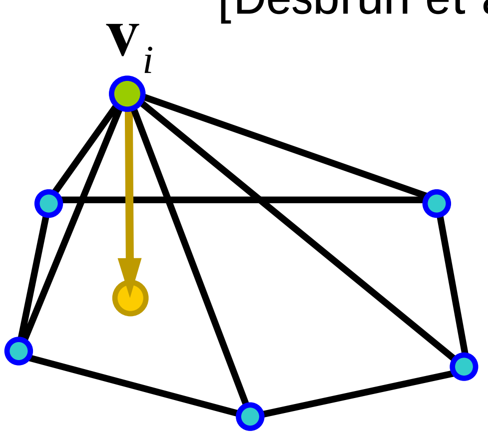
- Geometry dependent
  - Making LCs in local normal direction
  - Reduce tangential shift!

$$\boldsymbol{\delta}_i = \sum_{j \in Neigh(i)} \boxed{(\cot \alpha_{ij} + \cot \beta_{ij})} (\mathbf{v}_i - \mathbf{v}_j)$$

$$= 4 Area_i \kappa_i \mathbf{n}_i$$

[Desbrun et al. 1999]





Curvature Flow LCs

- Curvature flow LCs approximate the integrated mean curvature normal

$$\ell_i = \sum_{j \in i^*} \boxed{(\cot \alpha_j + \cot \beta_j)}(v_j - v_i)$$

$$= \boxed{4 Area_i \kappa_i} \widetilde{n}_i$$

Magnitudes –
geometry information

Coefficients –
parameterization information

Goals

- Minimize the difference of both parameterization and geometry information
  - Minimize shape distortion
- But, they are **non-linear** in the vertex positions
  - Single linear solver cannot obtain satisfactory solution

Alternating Iterations

- Given the LVs (fixed), compute the vertex positions using the weights of the original mesh
  - Keep the parameterization information
- Update the LVs so that they have the same magnitude of the original mesh
  - Keep the geometry information

Iteration: Step 1

- Update the vertex positions
  - Solve the linear system using the current LCs and original Laplacian operator

A is always the original Laplacian matrix

$$\boxed{A}V^k = b^k$$

$$V^k = \left(A^T A\right)^{-1} A^T$$

Enforce similar local parameterization as the original mesh

Iteration: Step 2

- Update the Laplacian vector
  - Compute the LCs using current vertex positions
  - Orient the LCs so that they point consistently to the same side of mesh as original LCs
  - Set the magnitude of the resulting LCs to be the same as original LCs

$$\boldsymbol{\delta}^k = L(V^k),$$

$$\mathbf{n}^{k+1} = Orient(\mathbf{n}^k),$$

$$\boldsymbol{\delta}^{k+1} = \boxed{d} \cdot \mathbf{n}^{k+1},$$

d is always the original magnitude of LCs

Enforce same scale local geometry as the original mesh

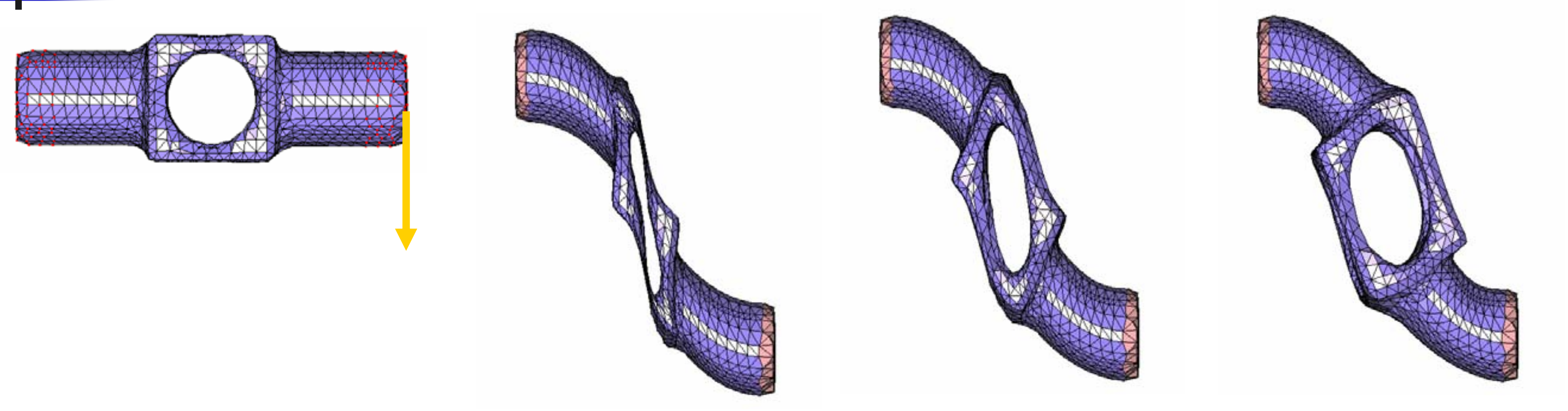
Normal Adjustments

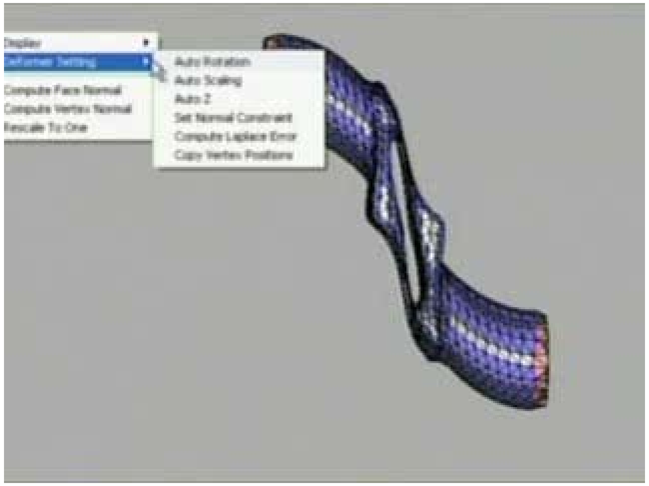
- Why adjust normal direction in the iteration?
  - The computed curvature normal may change between pointing inward or outward during editing
    - The local 1-ring structure might be convex or concave
  - Should be consistent with the corresponding original curvature normal

An Linear Iterative Framework

- Given the LVs (fixed), compute the vertex positions using the weights of the original mesh
  - Keep the parameterization information
- Update the LVs so that they have the same magnitude of the original mesh
  - Keep the geometry information

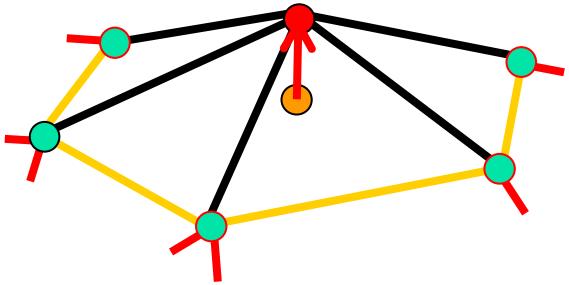
Experimental Result

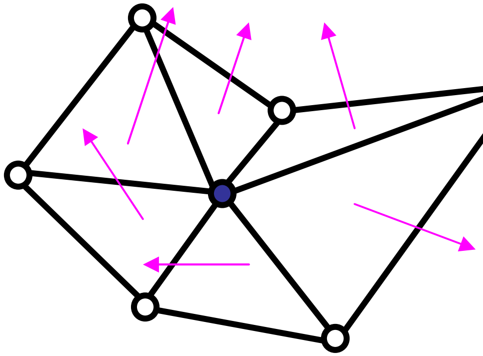




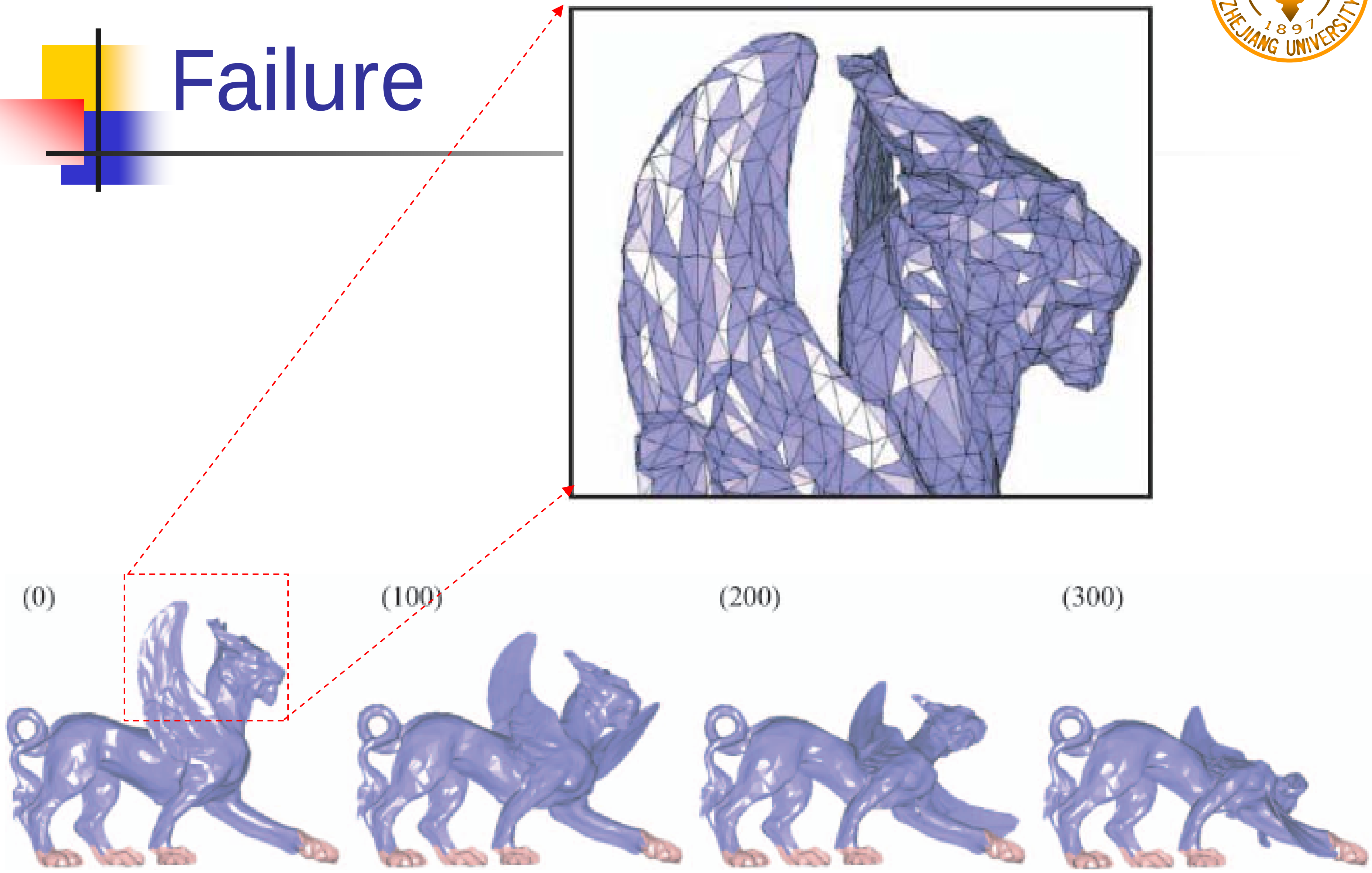
Drawbacks

- Might fail to converge
  - Poor sampling quality
  - Irregular connectivity
- The 1-ring neighbors are not coplanar
  - LCs have tangential components
  - The normal judgment is not reliable





Failure



Key to Solve the Problem

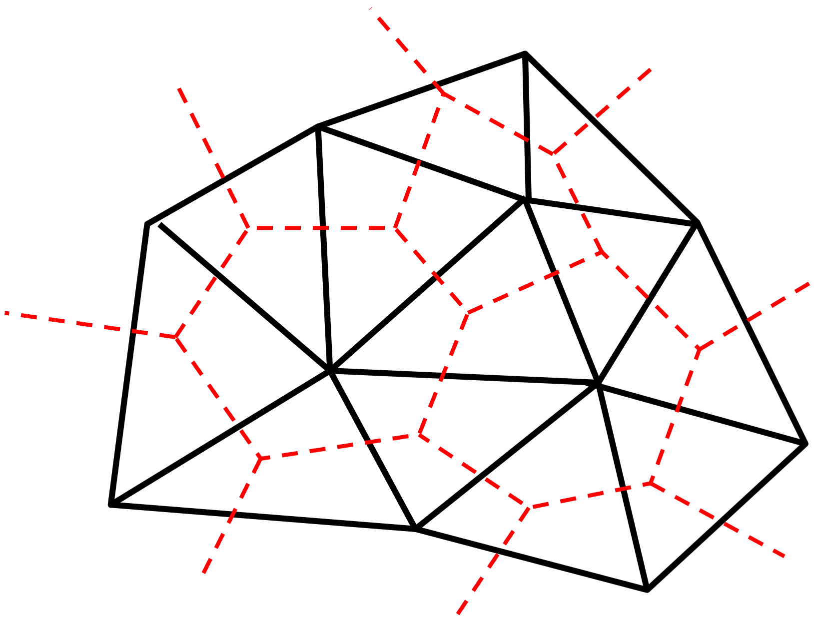
- Need to encode local geometry in the normal direction
- Eliminate the tangent shift component!

Dual Laplacian Editing

[Joint work with Au et al.]

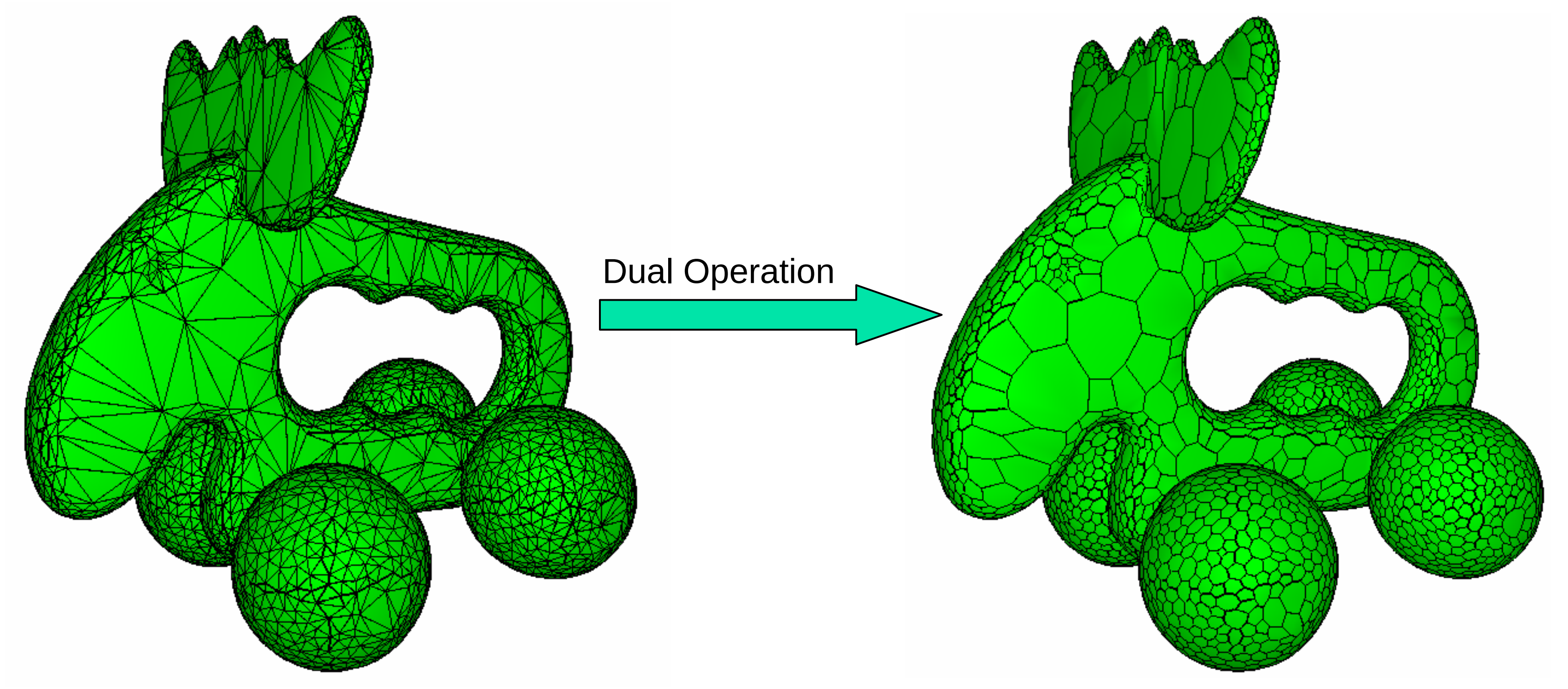
Dual Operator

Primary Mesh	Dual Mesh
Face	Vertex
Edge	Edge
Vertex	Face



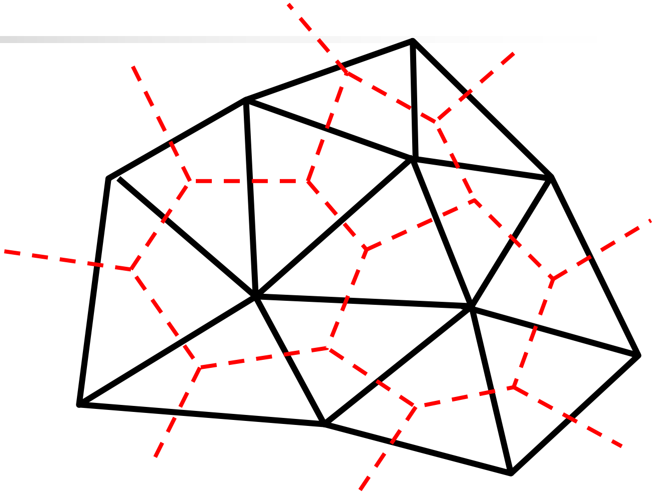
Dual Mesh

- For a triangular mesh, the valence of every vertex of its dual mesh is always 3
  - 1-ring structure of each vertex is simple and stable (always coplanar!)



Dual Mesh Generation

- Primary vertex $\mathbf{V}$
- Dual vertex $\bar{\mathbf{V}}$



$$\bar{\mathbf{V}} = D\mathbf{V}$$

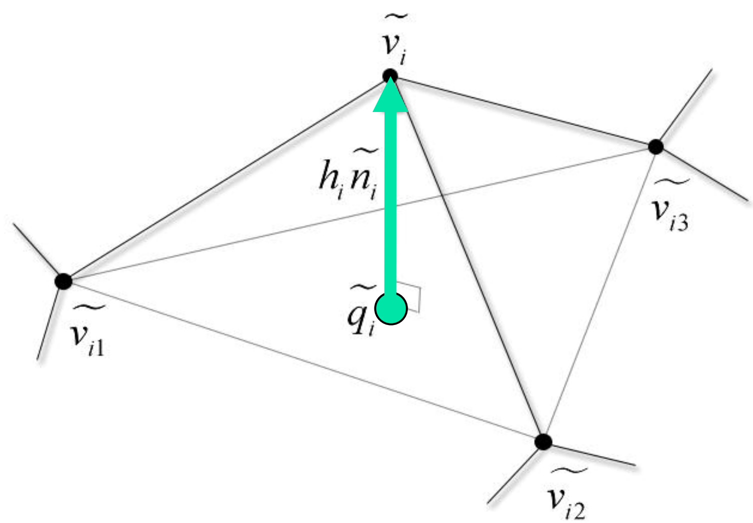
- Generally, $D(\bar{\mathbf{V}}) = D^2\mathbf{V} \neq \mathbf{V}$
  - Might introduce some errors in dual operator
- [Taubin 2001]: Dual Mesh Resampling

Dual Laplacian Coordinates

$$\tilde{\mathbf{v}}_i = \tilde{\mathbf{q}}_i + h_i \boxed{\tilde{\mathbf{n}}_i} = \sum_{j\in\{1,2,3\}} \tilde{w}_{i,j}\tilde{\mathbf{v}}_{i,j} + h_i\tilde{\mathbf{n}}_i$$

Uniquely defined!

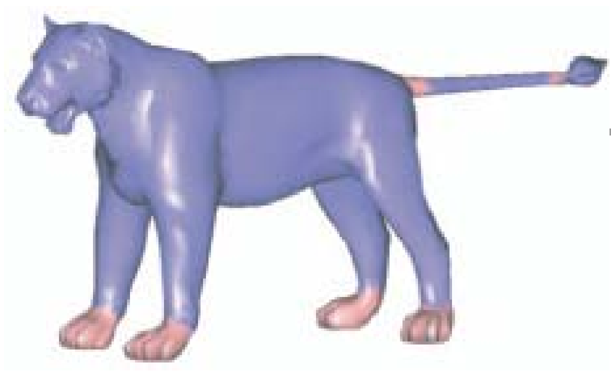
- Parameterization information
  - Footpoint $\tilde{q}_i = \tilde{w}_{i,1}\mathbf{v}_{i,1} + \tilde{w}_{i,2}\mathbf{v}_{i,2} + (1 - \tilde{w}_{i,1} - \tilde{w}_{i,2})\mathbf{v}_{i,3}$
- Geometry information
  - Height $\tilde{h}_i$
- The encoding $(\tilde{w}_{i,1}, \tilde{w}_{i,1}, \tilde{h}_i)$
  - Uniquely defined
  - Rotation-invariant



Dual Laplacian Editing

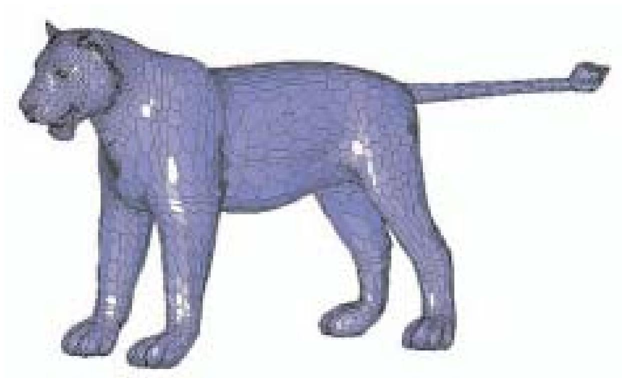
- Perform alternating iterations on dual mesh
  - Update the dual vertex positions
    - Keep the parameterization information
  - Update the dual Laplacian coordinates
    - Keep the geometry information
- Always convergent due to its stable 1-ring structures
  - Fast convergent

Dual Laplacian Editing



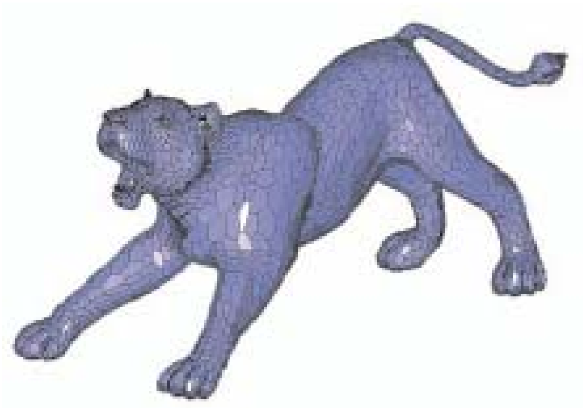
Original mesh

Edited mesh



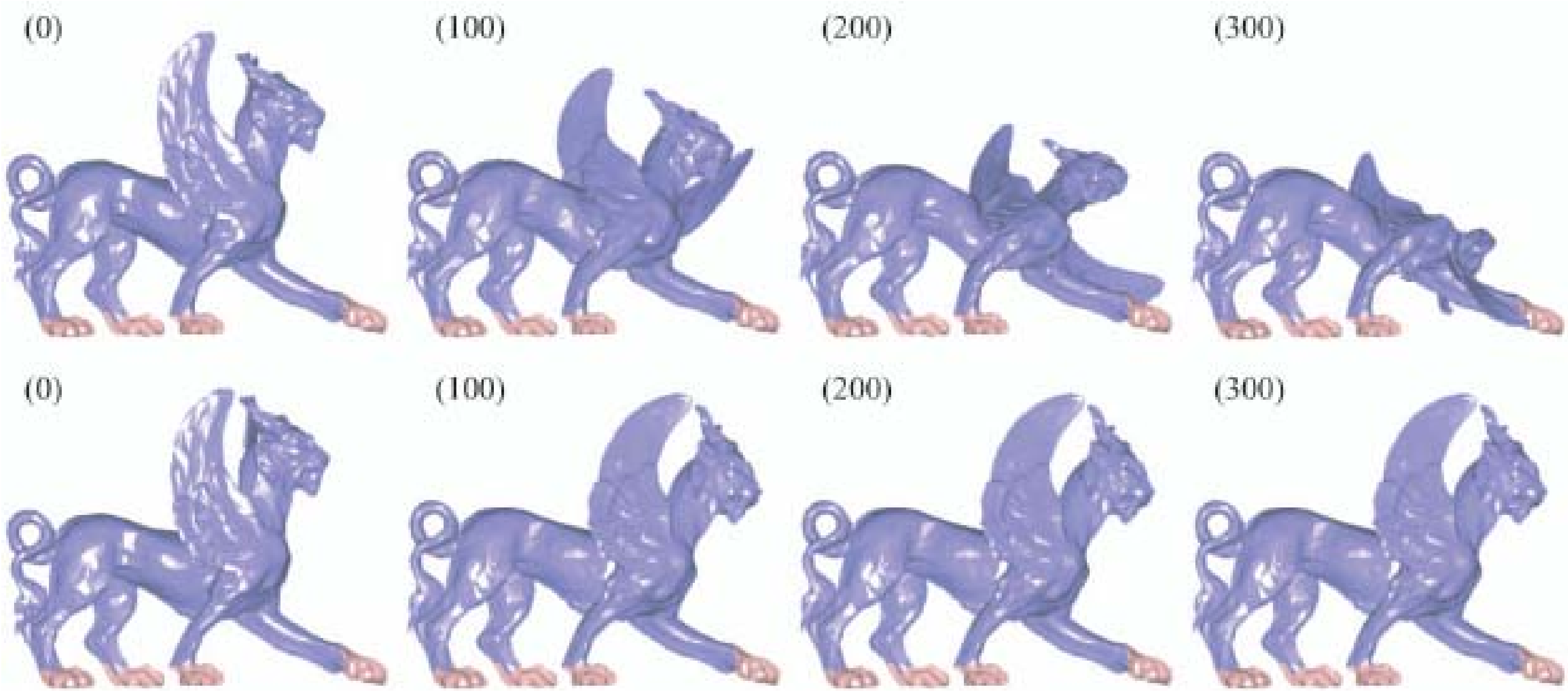
Dual mesh

Dual Laplacian Editing



Edited dual mesh

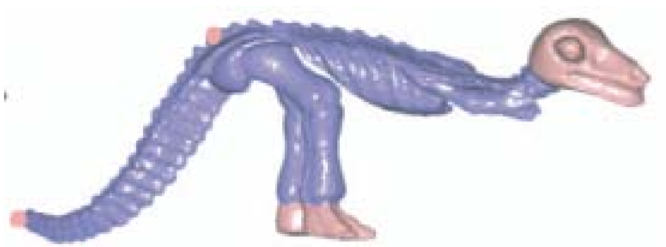
Experimental Result



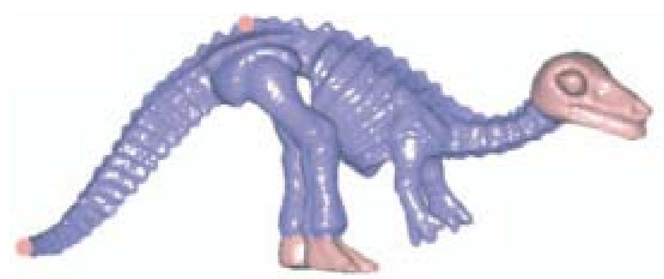
Dual Laplacian Editing

Reorienting the Dual LCs

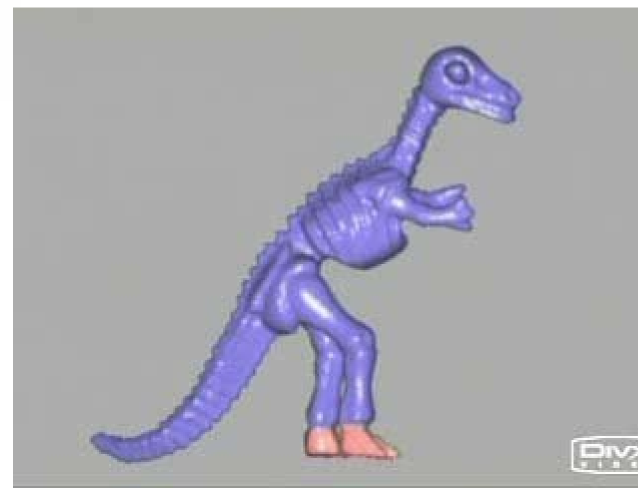
- Translational handles



Without reorientation



With reorientation

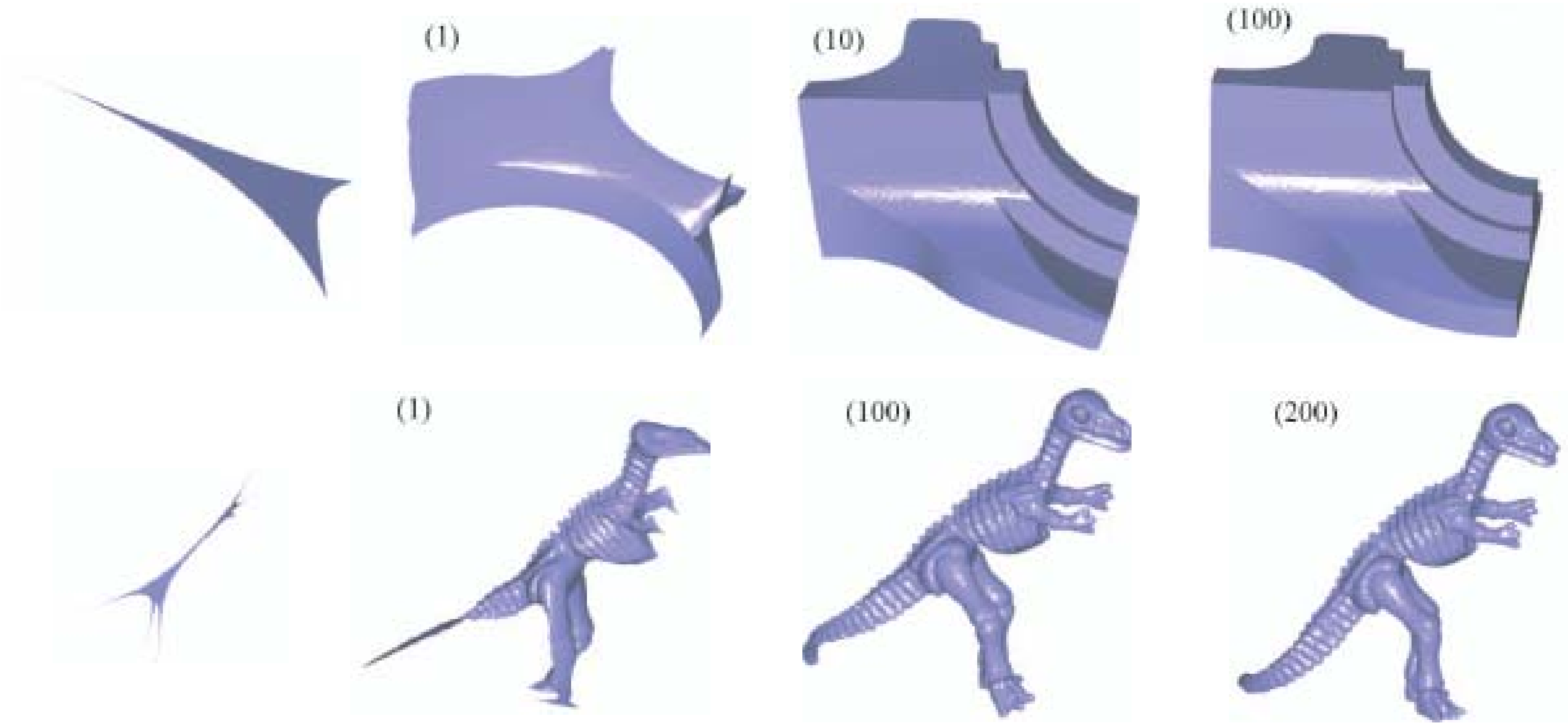


Examples

- Fine details

Robust

- Zero vectors as initial value



Implementation

- Solving the sparse linear system
  - Factorization of normal equation
  - Back-substitution at each steps
- 10-20 iterations for all examples

Model	Number of vertices	Factorization	Back-substitution
Lion	5000	0.360	0.016
Feline	4176	0.28	0.016
Dinosaur	14000	1.63	0.047
Skull	20002	3.38	0.078
Armadillo	60000	31.7	0.43

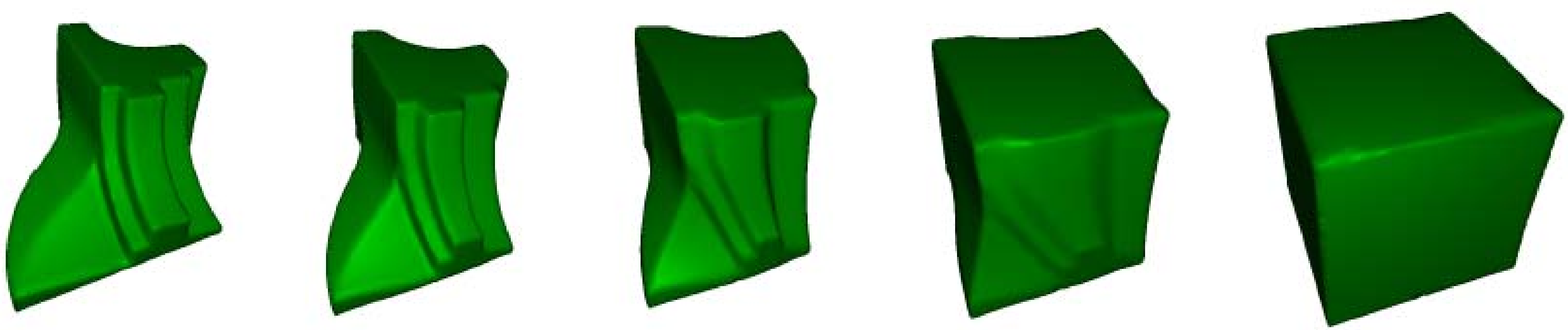
Pentium IV, 2.0GHz, 512M

Dual Laplacian Morphing

[Joint work with Hu et al.]

Mesh Morphing

- Input
  - Source mesh and target mesh
- Output
  - generate a sequence of intermediate meshes which gradually change from the source mesh to the target one

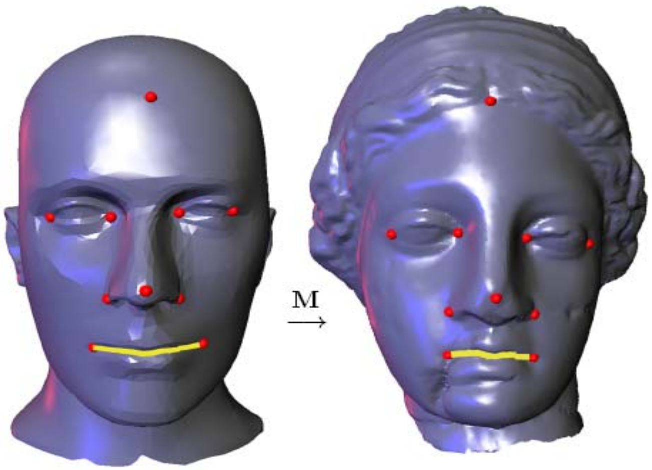


source

target

Two Subproblems

- Vertex *Correspondence* Problem
  - To find a correspondence between vertices of the two shapes
- Vertex *Path* Problem
  - To find paths that the corresponding vertices traverse during the morphing process



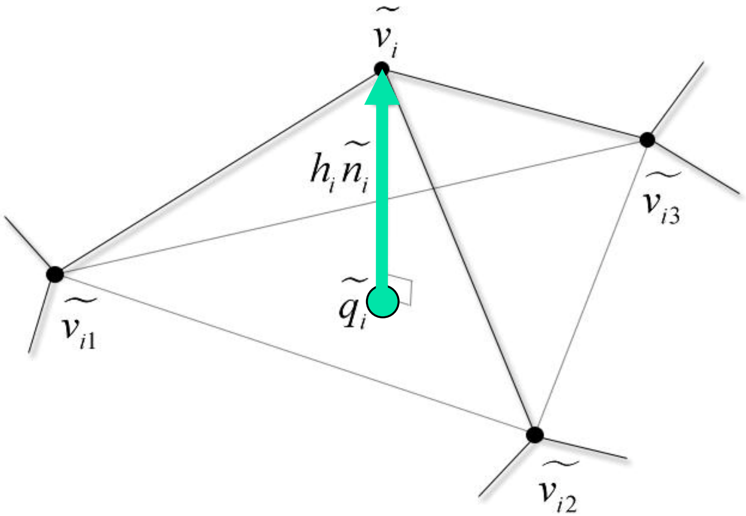
Motivation

- Novel path interpolation solution
  - Assume the vertex correspondence between meshes has been established
- Goal
  - Avoid shrinkage and kink in intermediate meshes

Basic Idea

- Both parameterization and geometry information are linearly interpolated
- The interpolated intrinsic information are used to construct the intermediate shapes

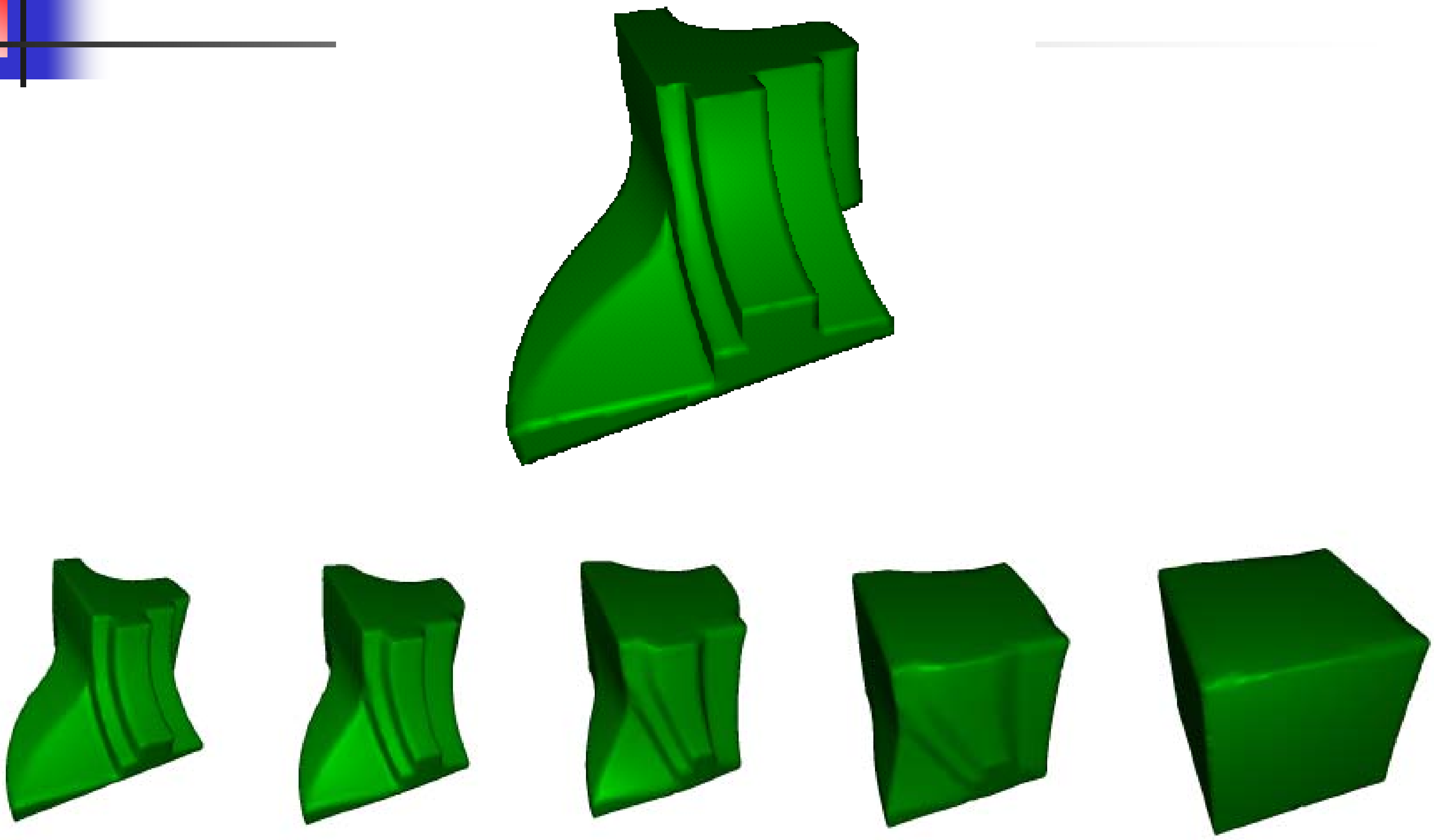
$$\{(\tilde{w}_{i,1}, \tilde{w}_{i,1}, \tilde{h}_i)\}$$



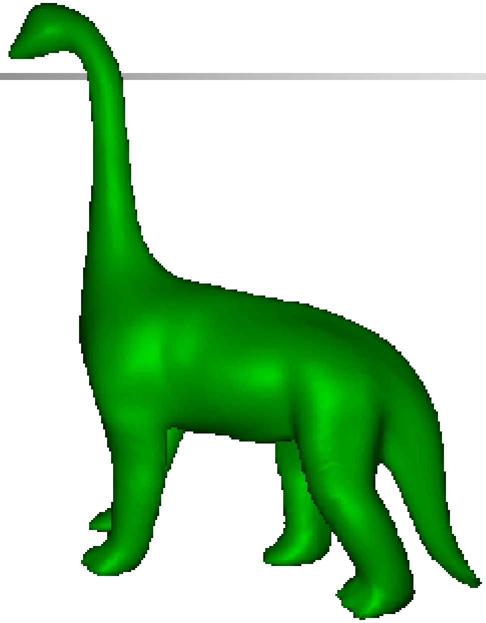
Dual Laplacian Morphing

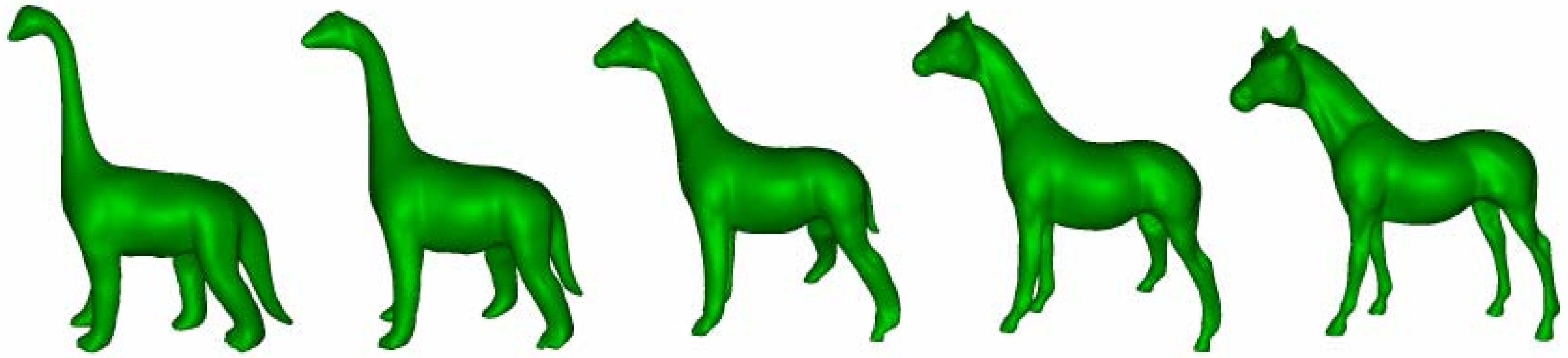
- Interpolating the intrinsic information of the two meshes
  - Parameterization and geometry information
- Performing on the dual meshes
  - Stable
- Reconstruction from intrinsic information
  - Non-linear process: linear iterative approximation

Fandisk & Cube

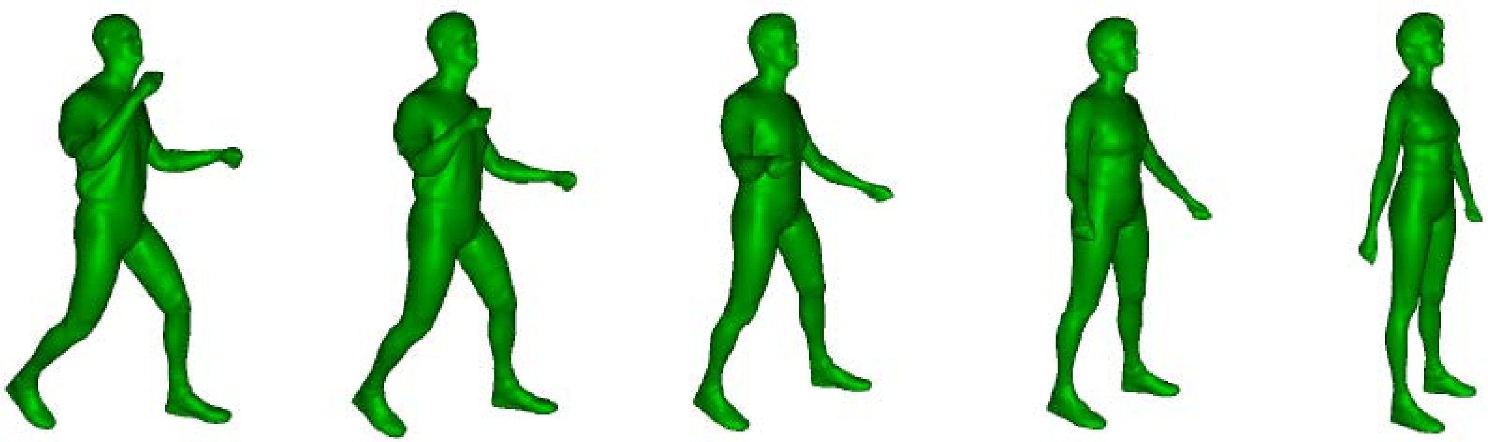


Dinosaur & Horse

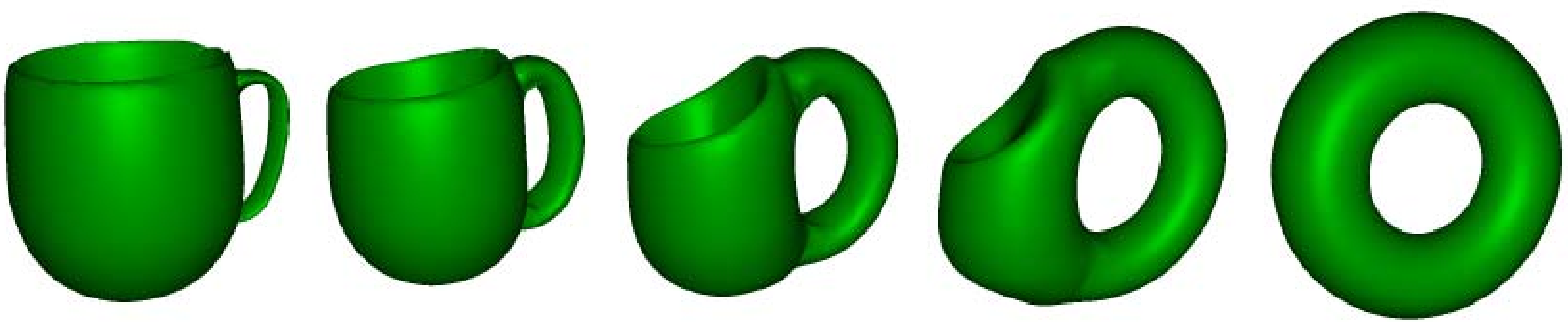




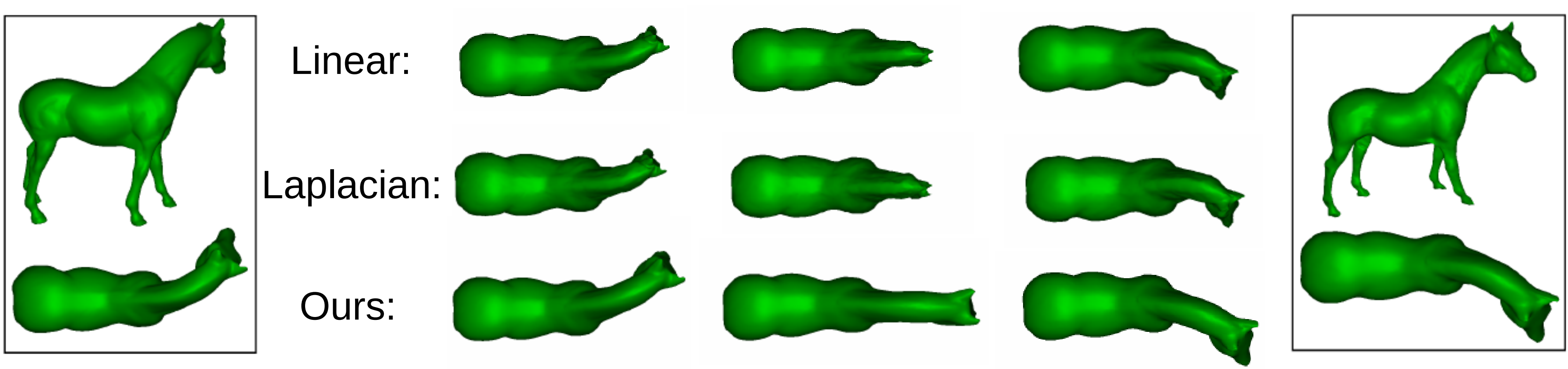
Man & Woman



Mug & Torus (High genus)



Comparisons



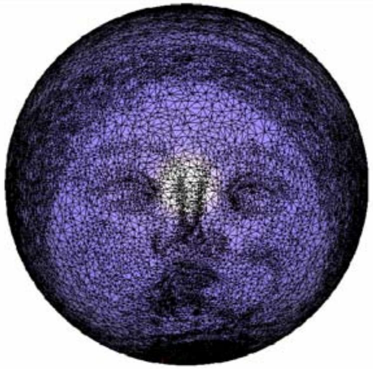
Dual Laplacain Morphing

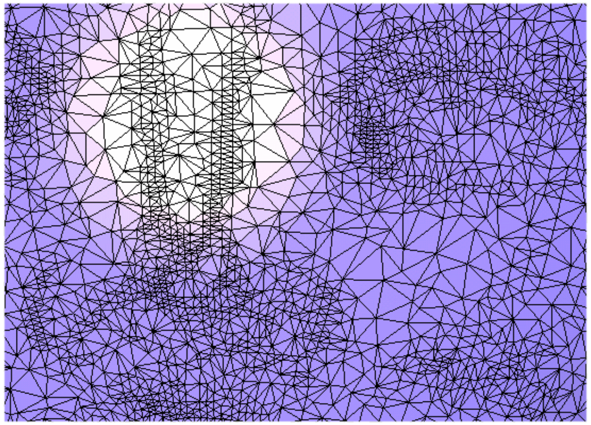
- Reconstruct the in-between shapes by dual Laplacian coordinates
- Construct the intermediate meshes by using an iterative framework
- Avoid the shape shrinkages and kinks

Other Manipulations

Spherical Parameterization

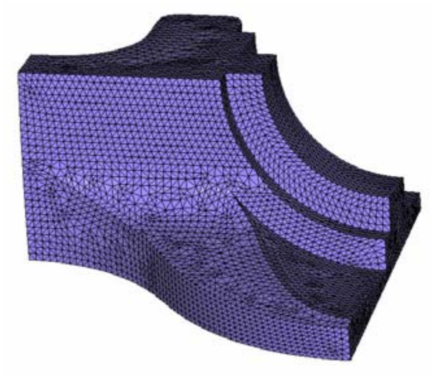
- Assign constant mean curvatures to all vertices
  - No overlapped

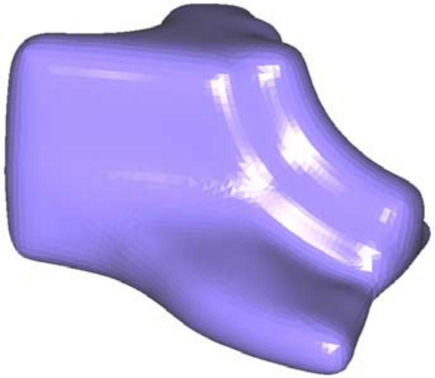


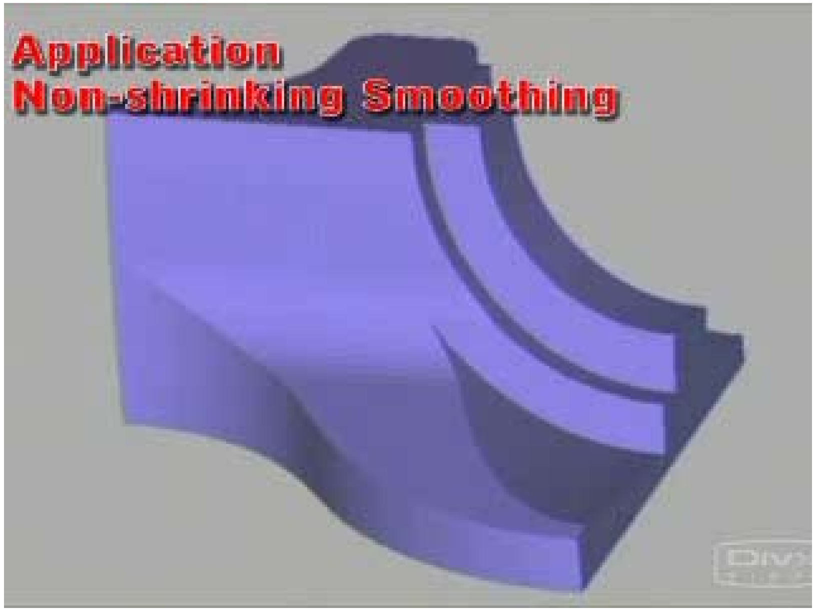


Smoothing

- Filtering curvature field
  - No shrinking effect







Conclusion

- Dual Laplacian processing
  - Mean curvature flow
    - Intrinsic information
  - Perform on dual domain
    - Stable solution
  - Linear iteration framework
    - Fast
  - Many applications

Thank you!