Mesh Shape Editing

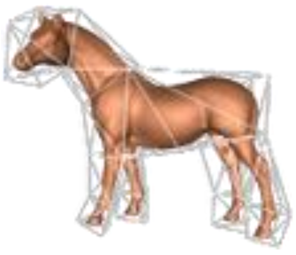
Computer Animation and Visualisation

Lecture 16

Taku Komura

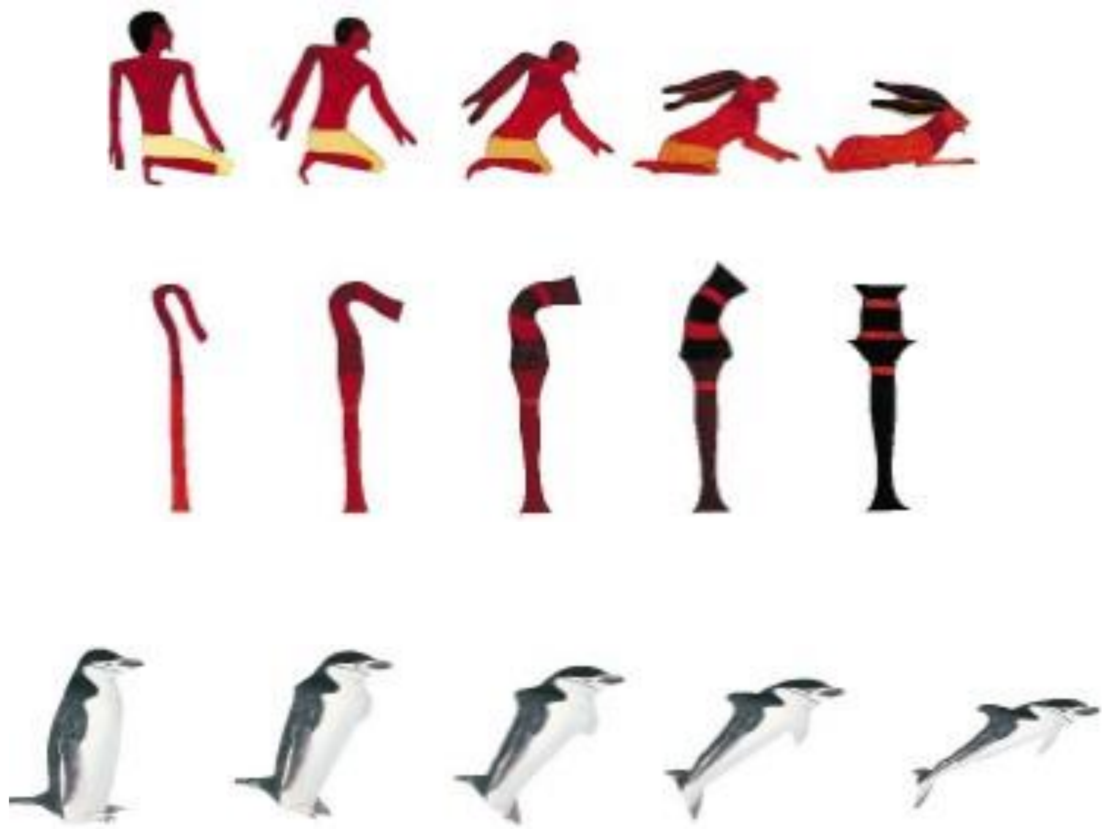
Today

- As-rigid-as possible shape interpolation
- Laplacian mesh editing
- Generalized Barycentric Coordinates



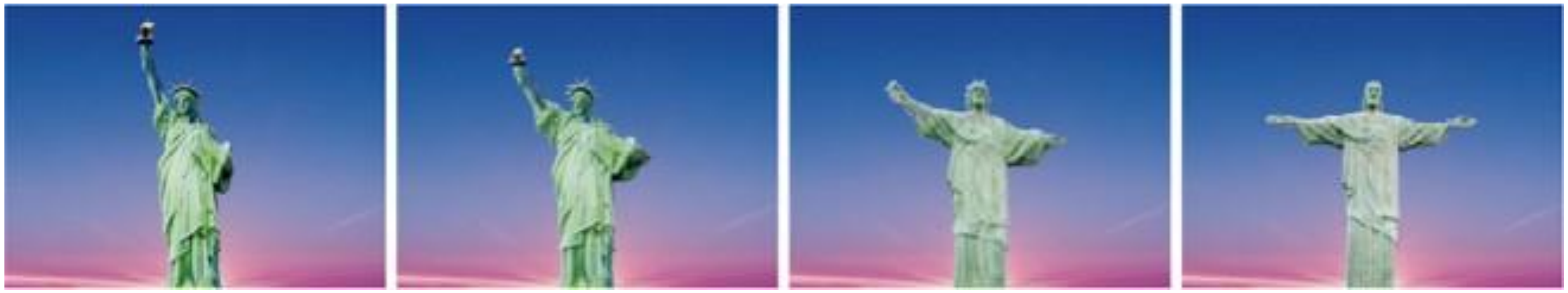
Interpolating Shapes

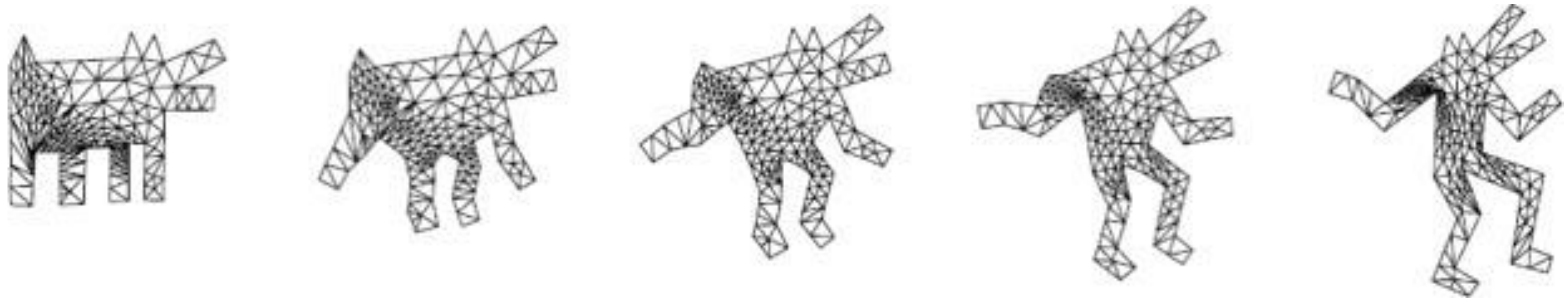
– Producing a smooth transition between two given shapes



As-Rigid-As Possible Shape Interpolation [Alexa '00]

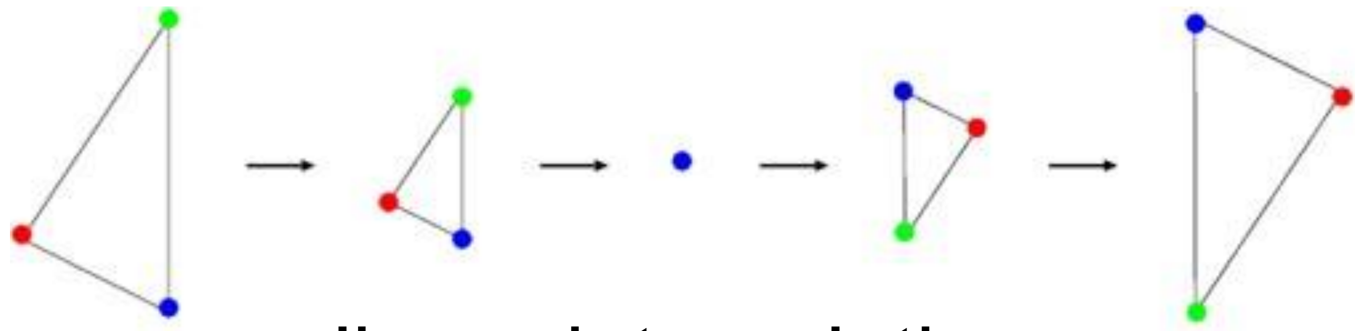
- Interpolating the shapes of the two polygons so that each triangle (2D) / tetrahedron (3D) transformation appears as rigid as possible



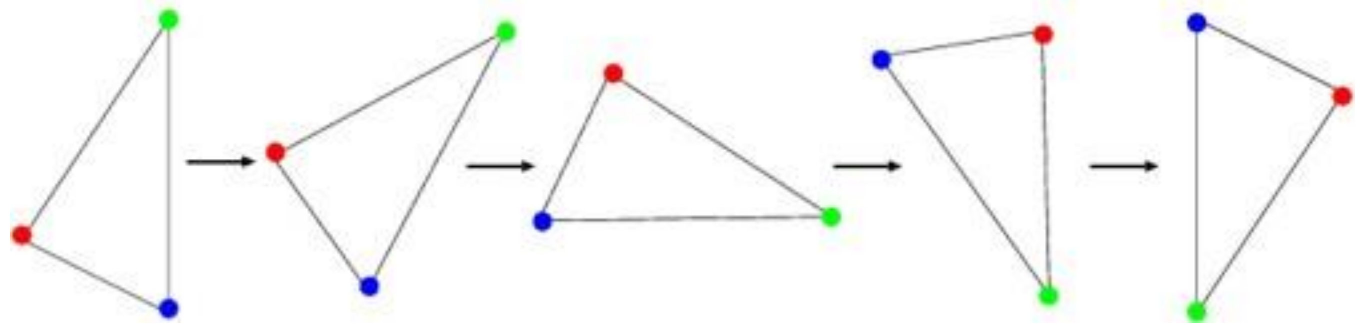


Interpolating the shapes of triangles

Represent the interpolation of triangles by rotation & scaling



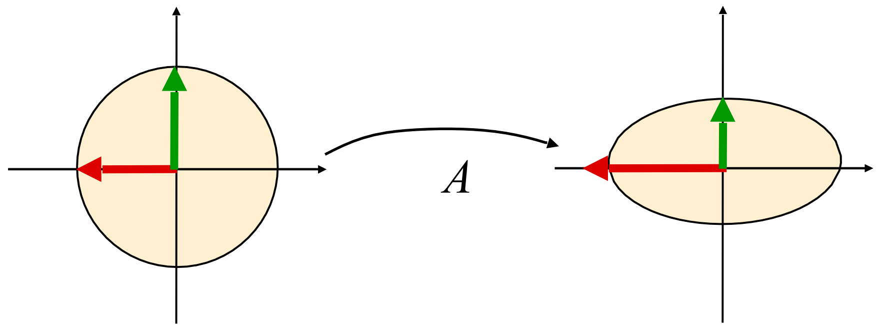
- linear interpolation



- rotation & scaling

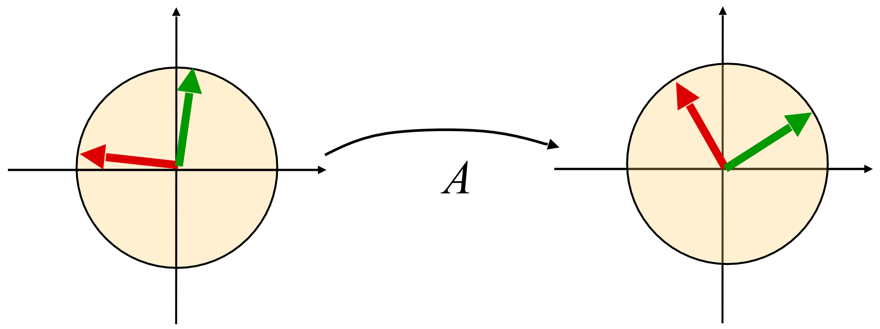
Background Knowledge: 2x2 matrix

Scaling



$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Rotation



$$R_\alpha = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

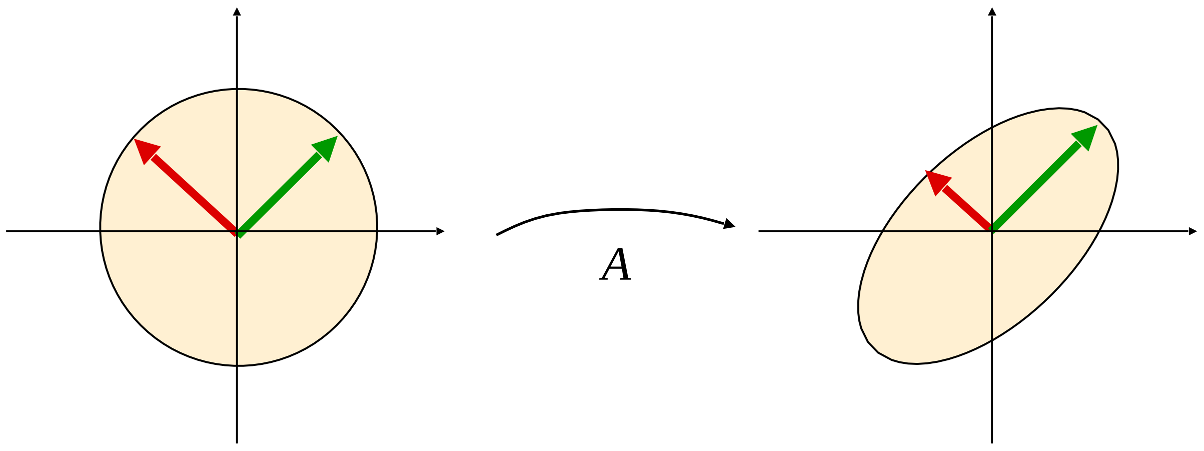
Transformation by 2x2 Matrix

If the matrix is symmetric, can be decomposed as

$\boxed{A = R\,\Lambda\,R^T, \quad R \text{ orthogonal},}$ where Λ scaling matrix

- this is called eigenvector decomposition

The eigenvectors of A are the axes of the ellipse



$$A = R \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} R^T$$

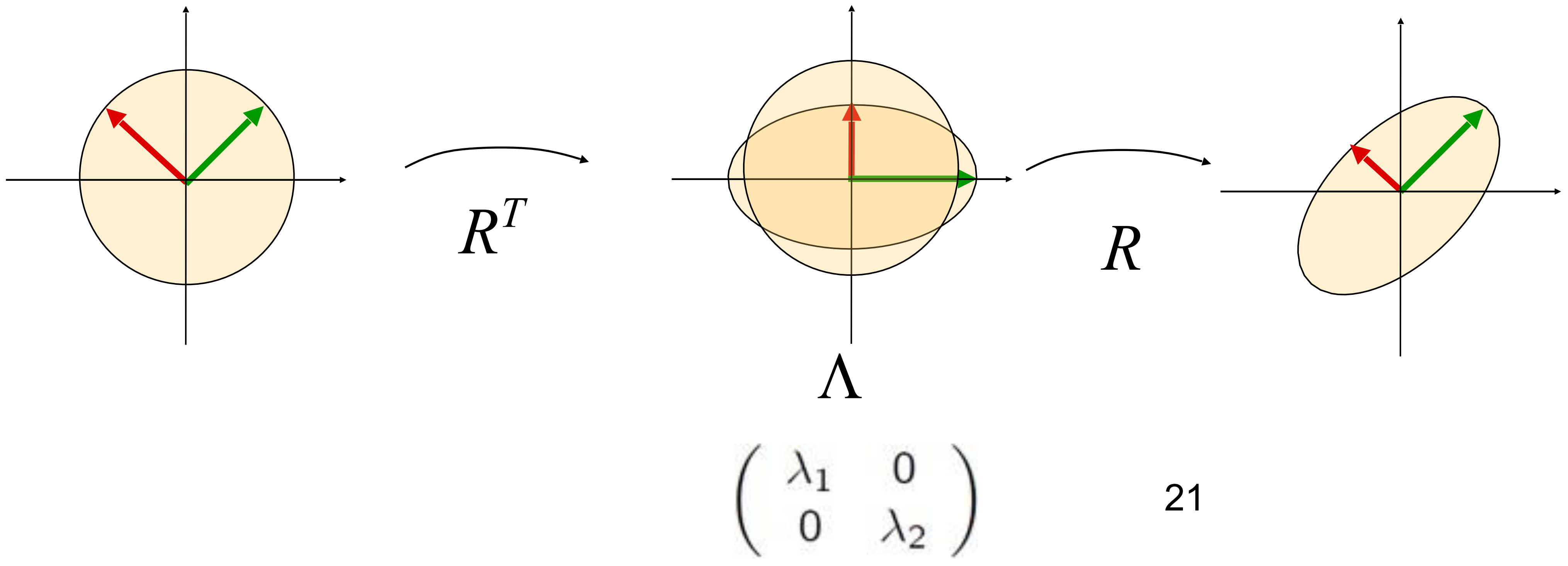
Geometric analysis of linear transformations

If the matrix is symmetric, can be decomposed as

$\boxed{A = R \Lambda R^T, \quad R \text{ orthogonal,}}$ where Λ scaling matrix

- this is called eigenvector decomposition

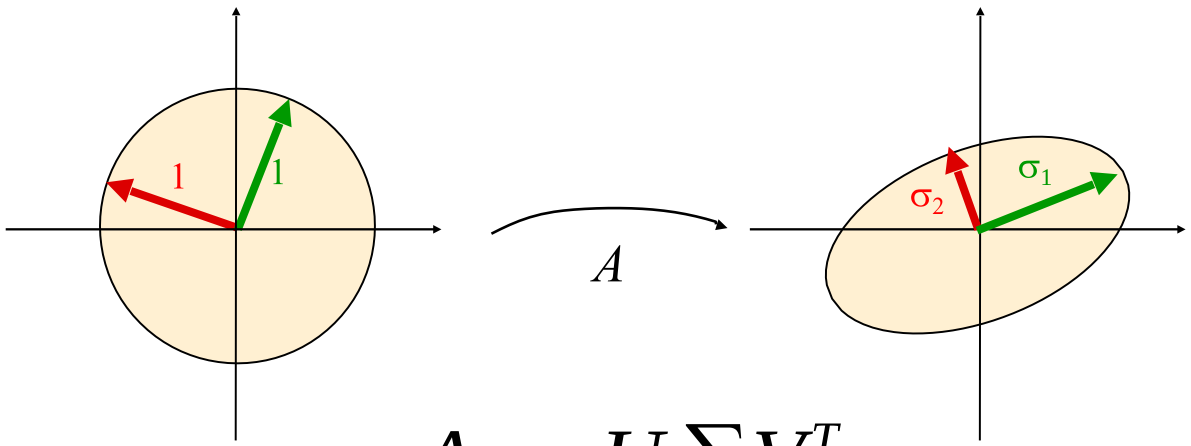
The eigenvectors of A are the axes of the ellipse



R^T
R
Λ
$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$

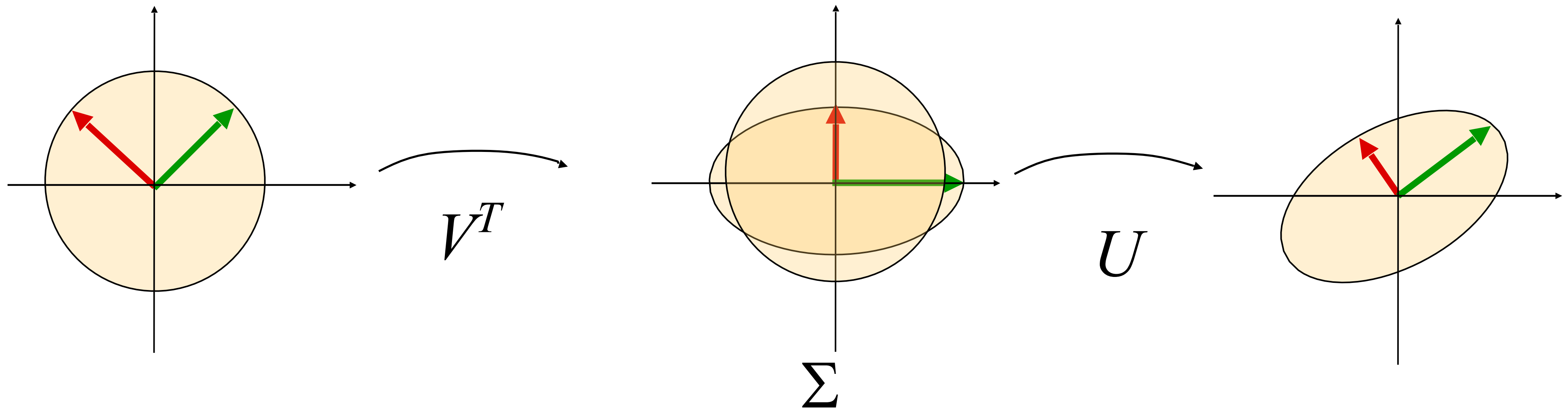
General linear transformations: Singular Value Decomposition (SVD)

In general A will also contain rotations, not just scales:



$$A = U \Sigma V^T$$

General Transformation : SVD



$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$$A = U \Sigma V^T$$

SVD operation

The columns of U are the eigenvectors of $A^{\mathrm{T}}A$
The columns of V are the eigenvectors of $A\,A^{\mathrm{T}}$
The diagnol values of $\sum$ are the square roots of the eigenvalues of $A\,A^{\mathrm{T}}$ or $A^{\mathrm{T}}A$

$$A = U \sum V^T$$

$$A^{\mathrm{T}}A = V\sum^T U^T U \sum V^T = V\sum^T \sum V^T$$

$$AA^{\mathrm{T}} = U\sum V^T \; V\sum^T U^T = U\sum \; \sum^T U^T$$

Back to ARAP Deformation
Least-Distorting Triangle-to-Triangle Morphing

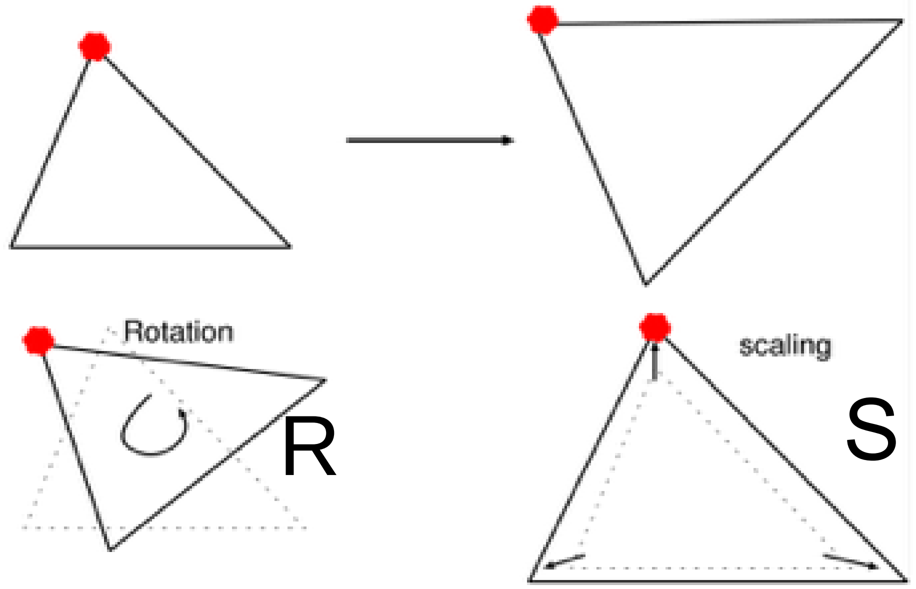
- Say the three vertices $\mathbf{P}=(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3)$ are morphed to $\mathbf{Q}=(\mathbf{q}_1,\mathbf{q}_2,\mathbf{q}_3)$

We want to compute a transformation that produces $[\mathbf{q}_2-\mathbf{q}_1\ \ \mathbf{q}_3-\mathbf{q}_1] = \mathbf{A}\,[\mathbf{p}_2-\mathbf{p}_1\ \ \mathbf{p}_3-\mathbf{p}_1]$

- We can apply SVD to compute the scaling and rotation part.
- $\mathbf{A} = \mathbf{R}\ \mathbf{S}$

rotation scaling

$$A = U\Sigma V^T = \overbrace{UV^T}^{\text{rotation}}\overbrace{V\Sigma V^T}^{\text{scaling}} = RS$$



Interpolation of the transformation matrix

The intermediate vertices will be computed by

V(t)=A(t) P

A is decomposed into the rotation part R(t) and scaling part and S.

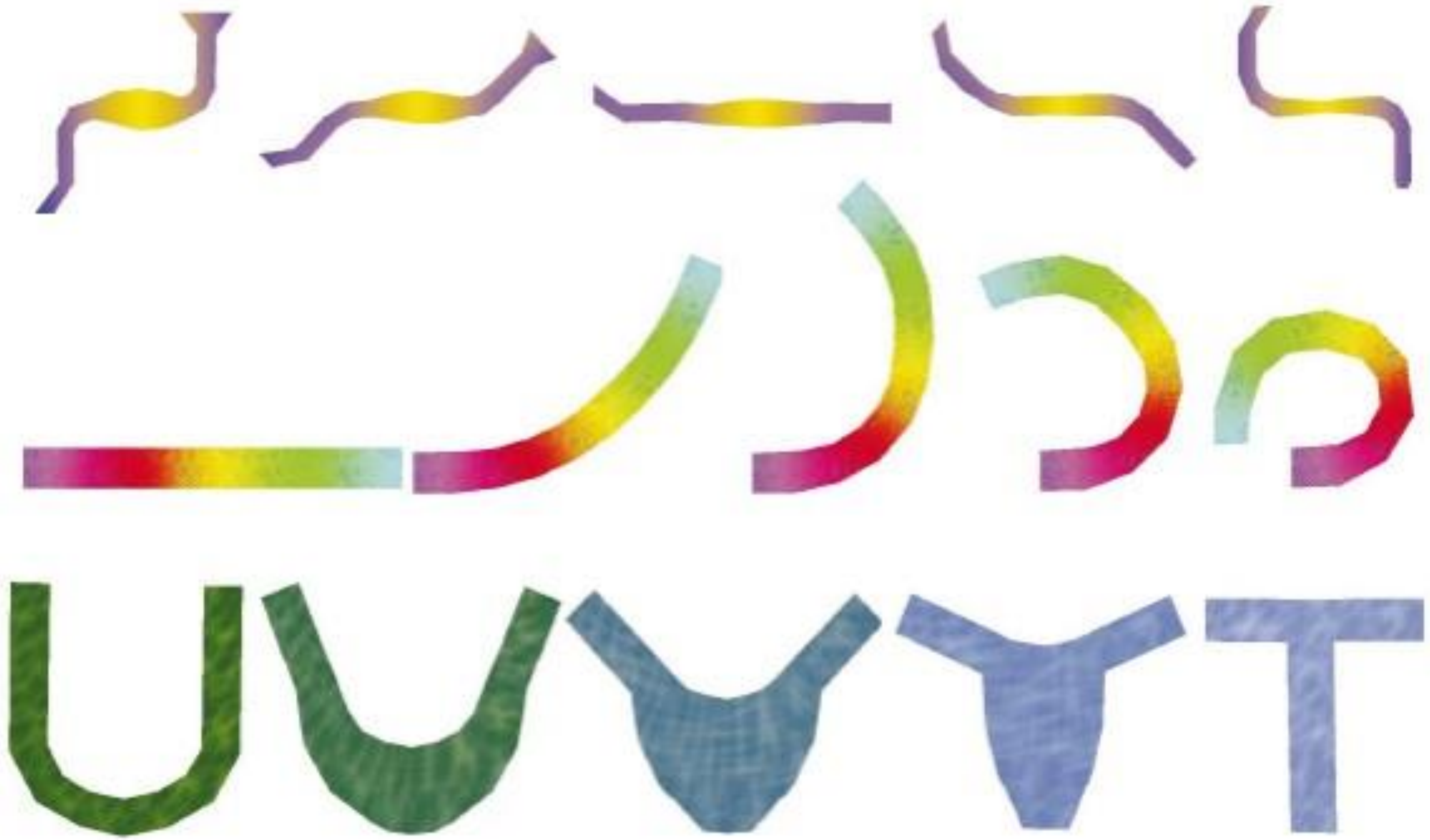
$$A_\gamma(t) = R_{t\gamma}((1-t)I + tS)$$

Keeping the transformation similar to $A_{\{i,j,k\}}$

- We cannot just interpolate all the triangles independently
- We need to move the vertices, not the triangles
- A vertex configuration that minimizes the error between A and B is computed

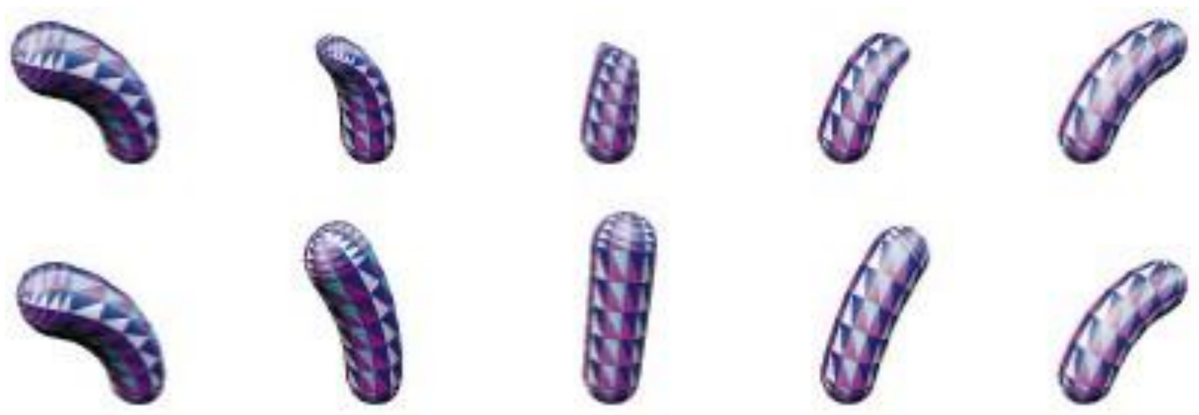
$$E_{V(t)} = \sum_{\{i,j,k\} \in \mathcal{T}} \left\| A_{\{i,j,k\}}(t) - B_{\{i,j,k\}}(t) \right\|^2$$

Some 2D results



3D objects

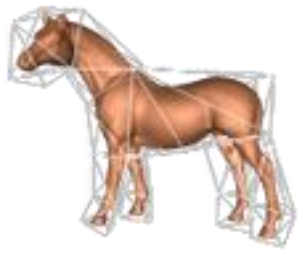
- The method is applicable to polyhedra
- Tetrahedralization is applied to polyhedra and the tetrahedra are morphed so that they are as rigid as possible



More results

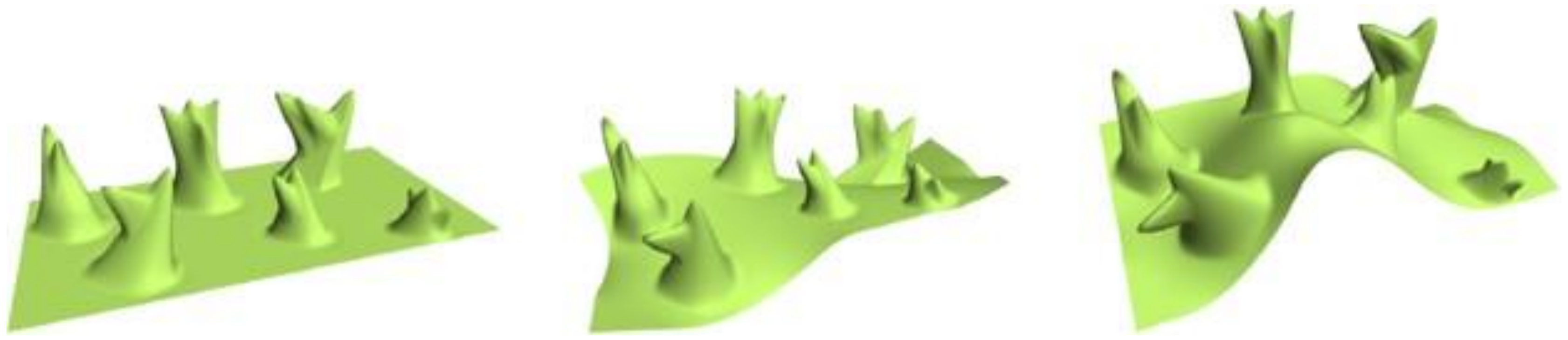
Today

- As-rigid-as possible shape interpolation
- **Laplacian mesh editing**
- Generalized Barycentric Coordinates



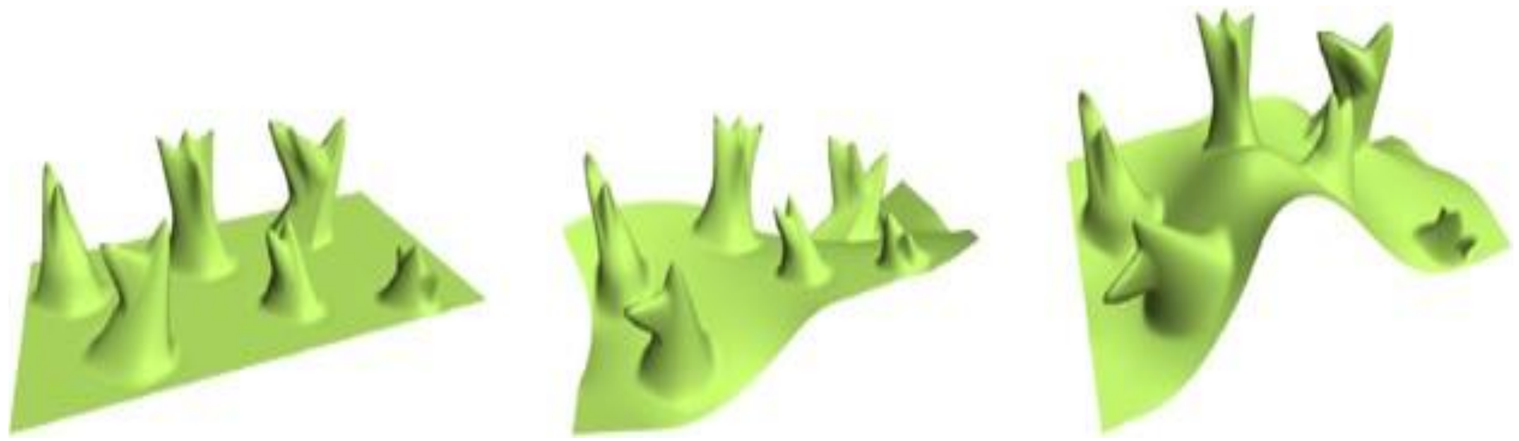
Editing shapes

- For animating rigid/articulated objects, we can use previous techniques like skinning
- Let’s think of animating cloths, clay, rubber, soft tissues, dolls, etc



Editing shapes

- Usually, it is easier to edit a given shape rather than modeling it from scratch
- What is important when editing shapes?
- Keeping the local information unchanged
  - Deforming as if the object is like rubber
  - See the image below



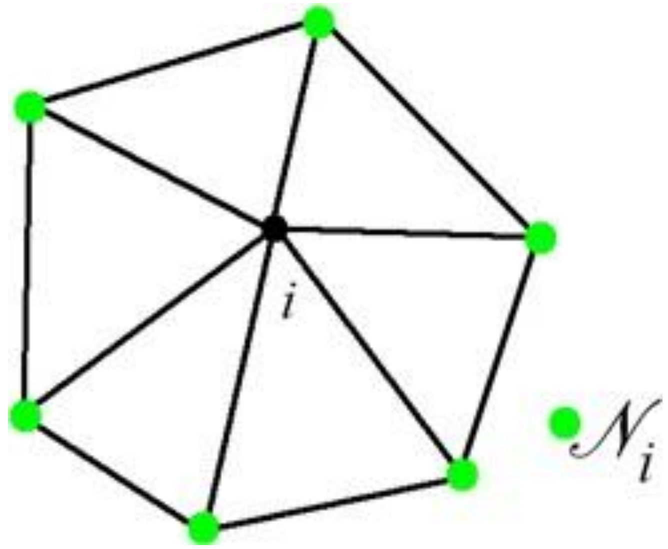
Surface Editing

- Must
  - Be fast (interactive rate)
  - Preserving the details

  - > Differential coordinates – Laplacian coordinates
    - Linear, invariant to scale, rotation [Sorkine ’04]
    - As-rigid-as-possible

Laplacian coordinates

- Assuming all polygons are split into triangles
- Every vertex **v***i* is surrounded by a group of vertices **N***i*
- Coordinate *i* will be represented by the difference between **v***i* and the average of its neighbors



- The transformation between the Laplacian coordinates and the original vertices is linear

$$\mathscr{L}(\mathbf{v}_i) = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \mathbf{v}_j .$$

Laplacian coordinates (2)

- Suppose the new position of the vertices are $\mathbf{v}'_i$
- We want to keep the Laplacian coordinates the same after the deformation

$$\delta_i = \mathscr{L}(\mathbf{v}_i).$$

$$E(V') = \sum_{i=1}^{n} \left\| T_i(V')\delta_i - \mathscr{L}(\mathbf{v}'_i) \right\|^2$$

- T is a homogeneous transformation of scaling, rotation, and translation

Laplacian coordinates (3)

- We also want to constrain the position of some points (keeping $\mathbf{v}_i$' close to $\mathbf{u}_i$)
- After all, we need to minimize the following function

$$E(V') = \sum_{i=1}^{n} \left\| T_i(V')\delta_i - \mathscr{L}(\mathbf{v}'_i) \right\|^2 + \sum_{i=m}^{n} \|\mathbf{v}'_i - \mathbf{u}_i\|^2.$$

- This becomes a simple quadratic optimization problem

Solving a quadratic problem

Rewriting this problem, it becomes like

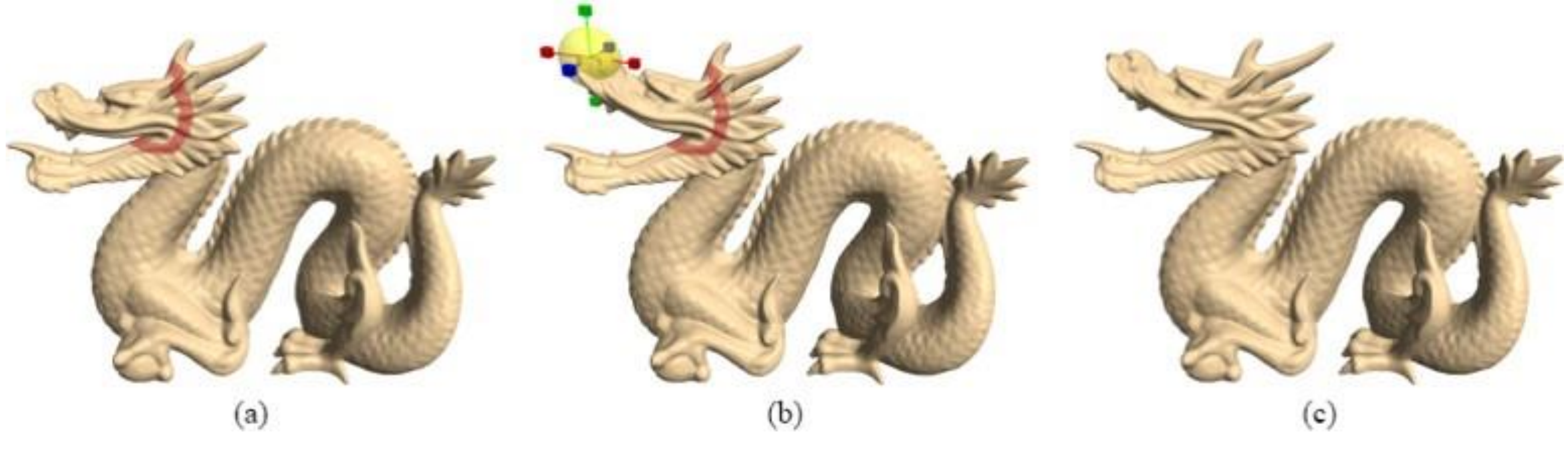
$$\min_{\mathbf{x}} \| \mathbf{c} - \mathbf{A}\,\mathbf{x} \|^2$$

The optimal x can be computed by solving

$$\mathbf{A}^T\mathbf{A}\,\mathbf{x} = \mathbf{A}^T\mathbf{c}$$

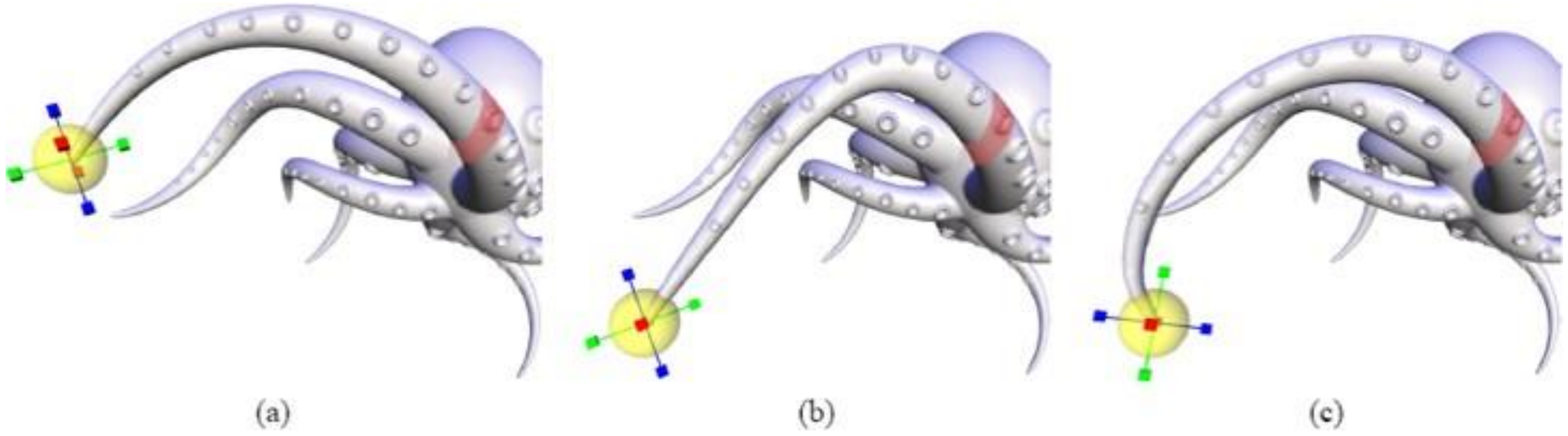
Editing Surfaces by Keeping the Details

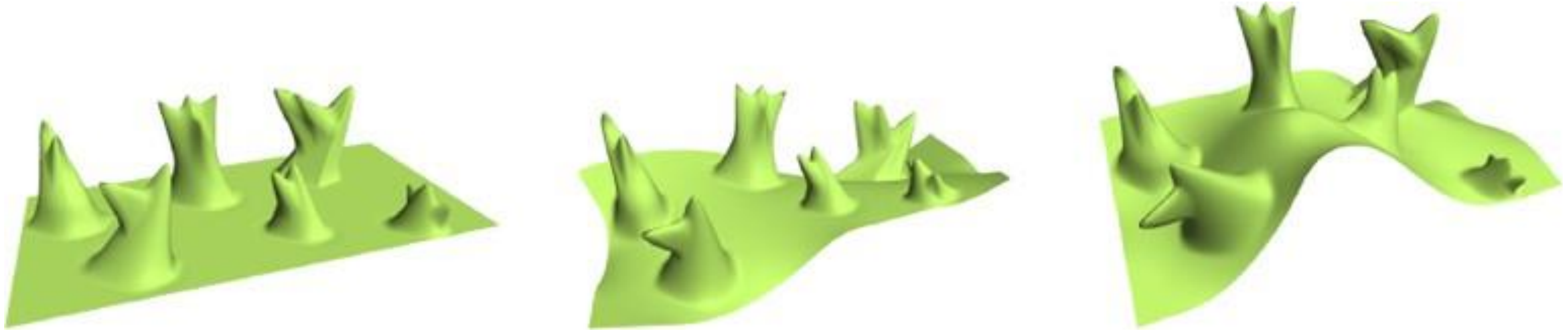
- The user specifies the region of interest (ROI) (the area to be edited)
- The user directly moves some of the vertices
- The rest of ROI is decided by minimizing the error function



Some more …

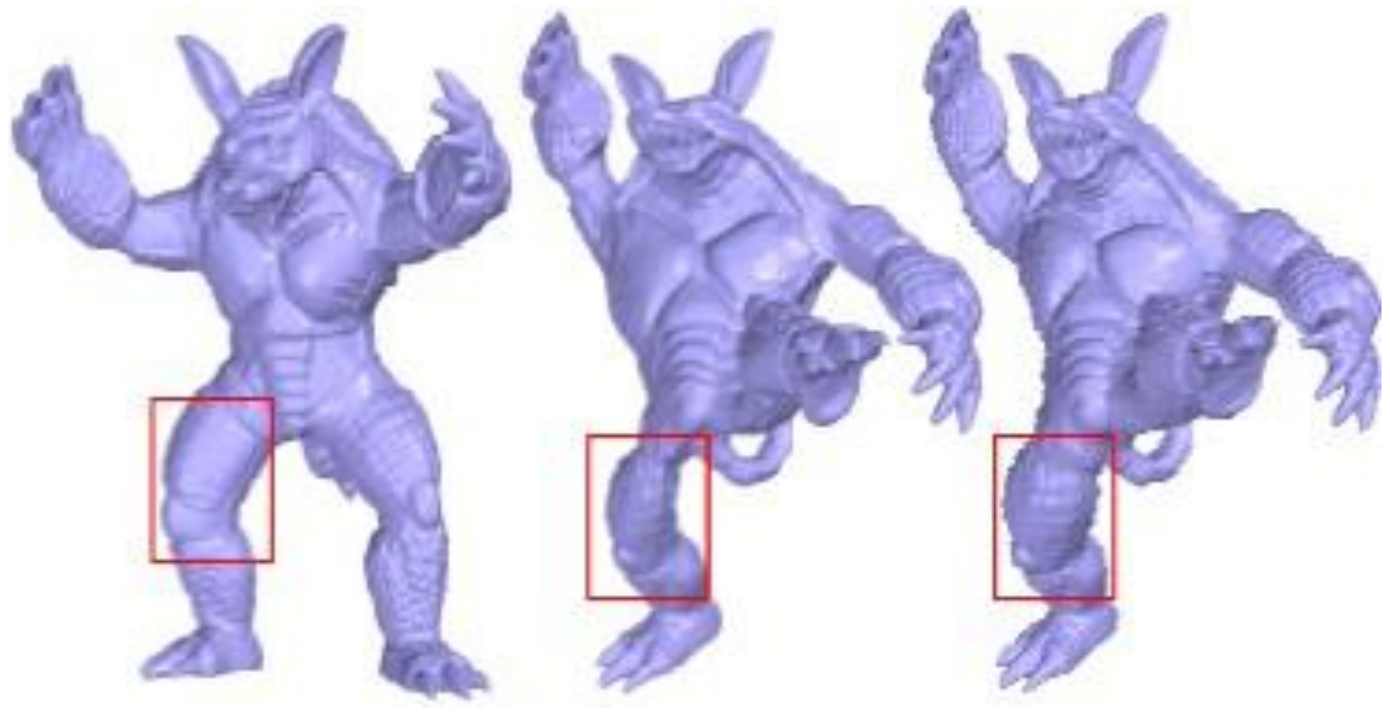
http://www.youtube.com/watch?v=Yn3P4EK8sYE&feature=channel_page





It is possible to add more constraints

- Sometimes the shape might shrink as we allow scaling for the transformation
- It might be better to keep the volume the same



Creating a Volumetric Graph

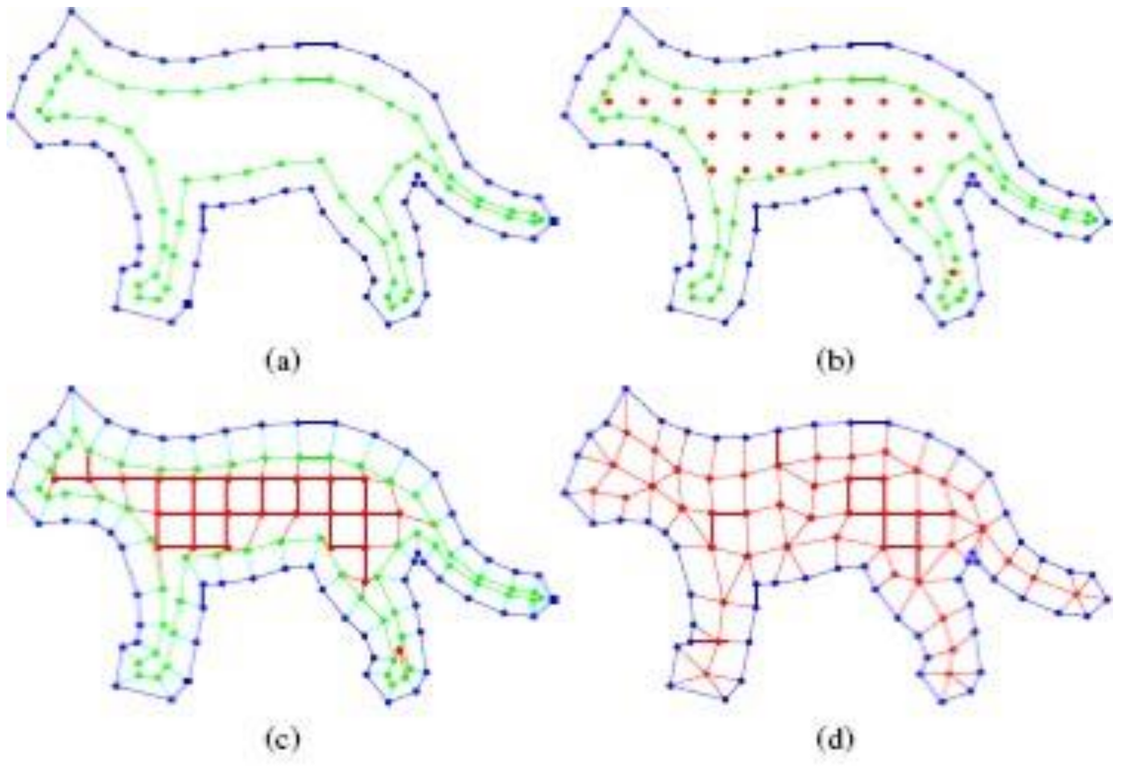


Figure 5: *Volumetric graph construction.*

- Add internal vertices and edges
- Compute and preserve the details for the internal structure

Some Results

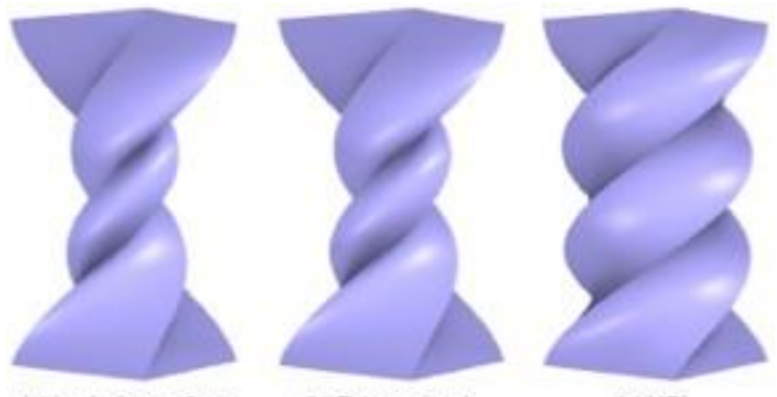


Figure 2: *Large twist deformation.*

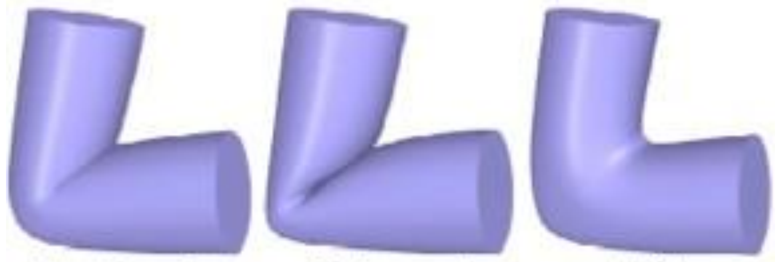
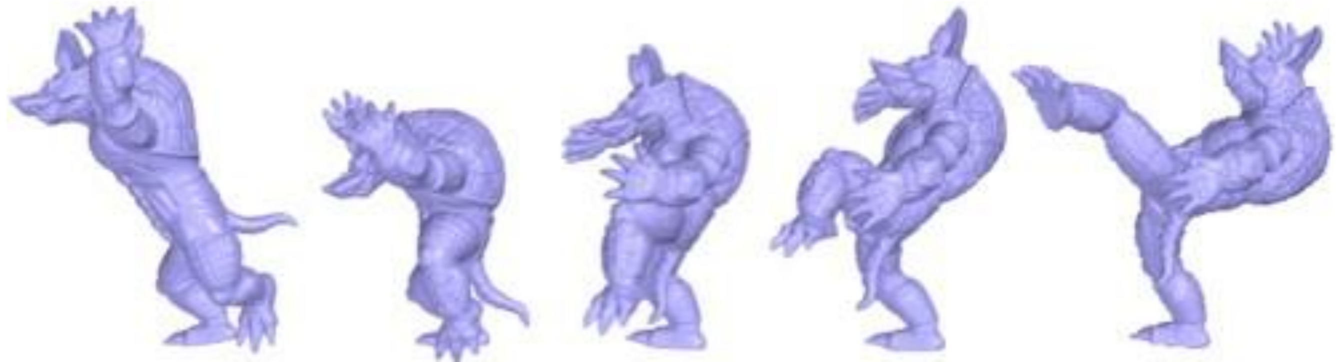
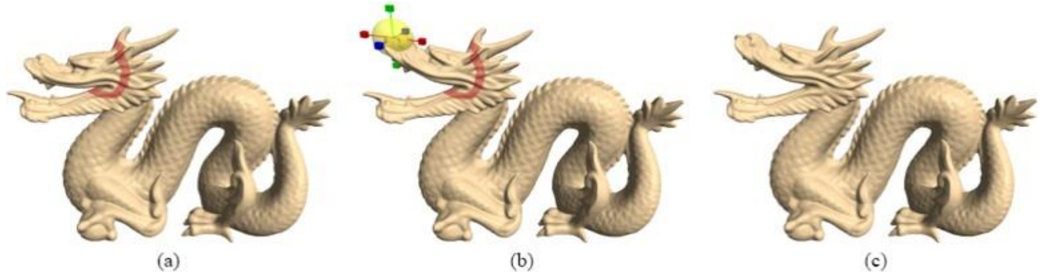


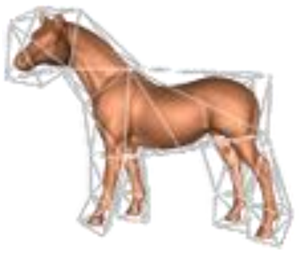
Figure 3: *Large bend deformation.*



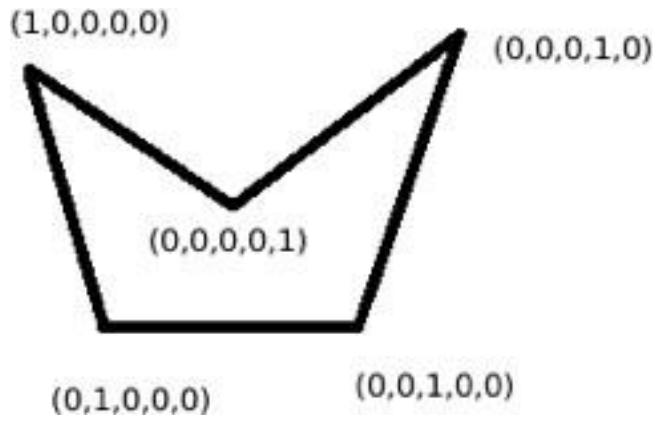
Today

- As-rigid-as possible shape interpolation
- Laplacian mesh editing
- **Generalized Barycentric Coordinates**





Barycentric Coordinates



Given a polygon with N vertices, the barycentric coordinates is a N dimensional vector

$$(\lambda_1, \dots, \lambda_N)$$

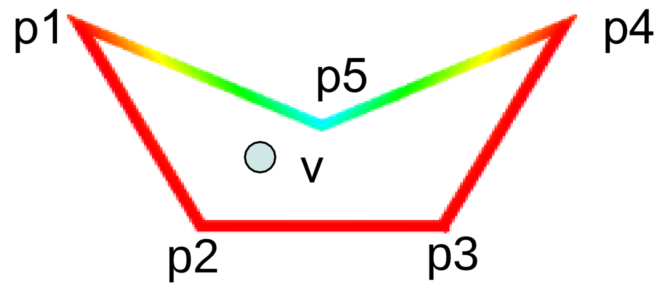
We can use them to represent a position in the space with respect to the N points of the polygon

We can also use them to interpolate attribute values defined at the N vertices of the polygon

Interpolation

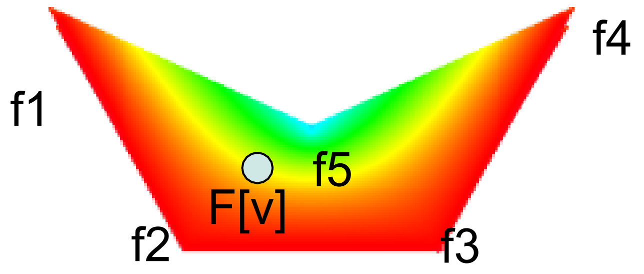
Geometric interpolation – derive the global coordinates for a position in parametric cell space

$$v = \sum_i \lambda_i p_i$$



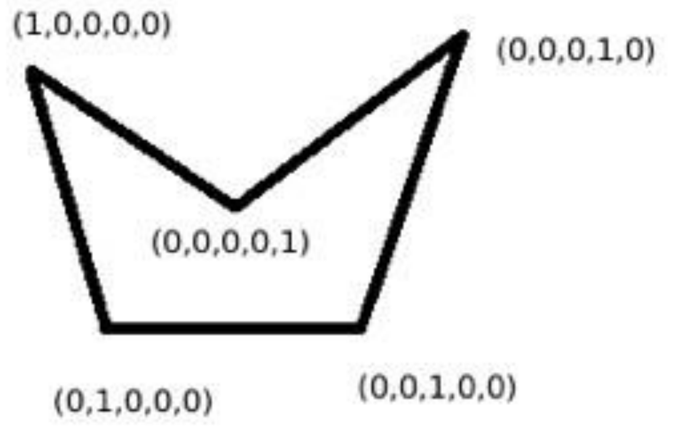
Attribute interpolation – derive the attribute value for a position defined in parametric cell space

$$f[v] = \sum_i \lambda_i f_i$$



Demo http://www.lidberg.se/math/shapetransforms/barycentric.html

Barycentric Coordinates

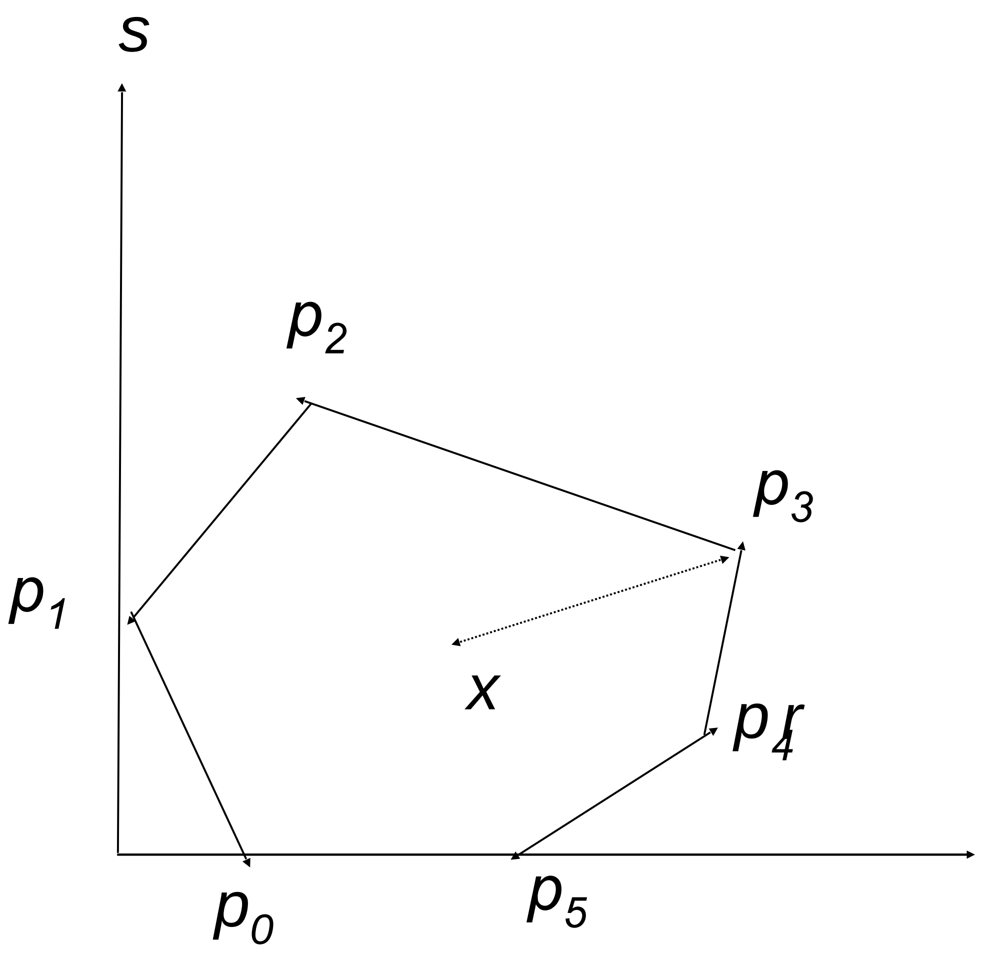


Requirements

– $\lambda_i = 1 \; and \; \lambda_j = 0 \;\; when \; p = p_i \; and \; i \neq j$

- The values must be bounded as follows:
- $\sum_i \lambda_i = 1, 0 \leq \; \lambda_i \leq 1$

How to compute the barycentric coordinates for each point?

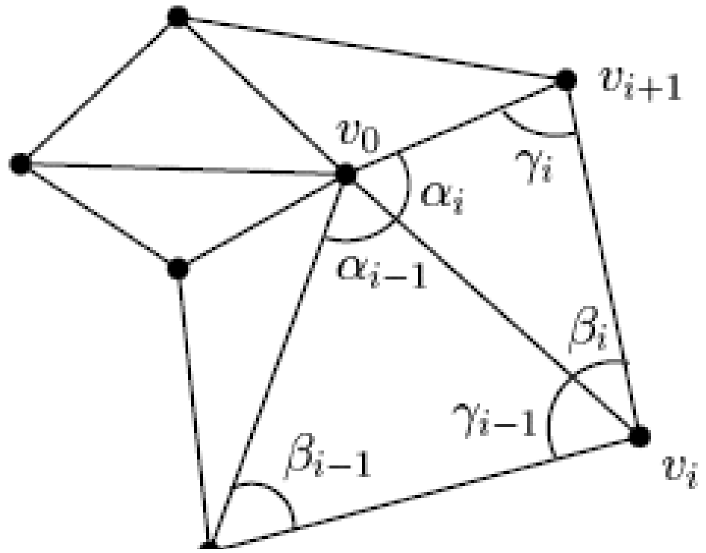


Mean Value Coordinates

A good and smooth barycentric coordinates that can smoothly interpolate the boundary values

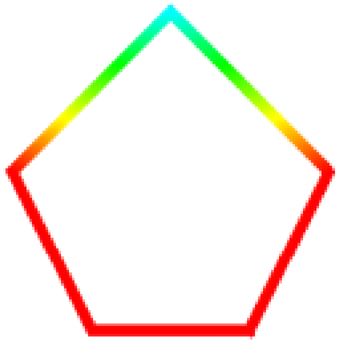
Also works well for concave polygons

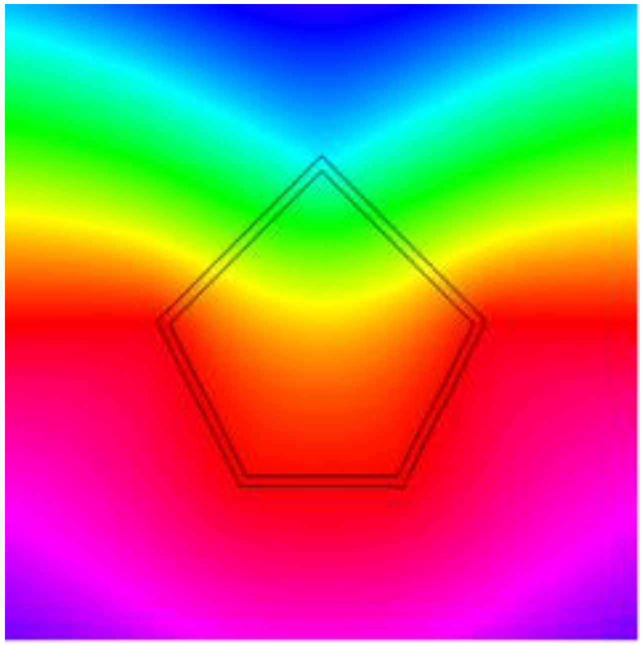
There is also a 3D version



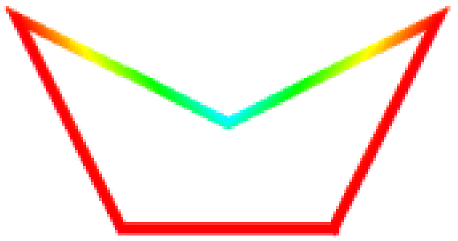
$$\lambda_i = \frac{w_i}{\sum_{j=1}^{k} w_j}, \qquad w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{||v_i - v_0||},$$

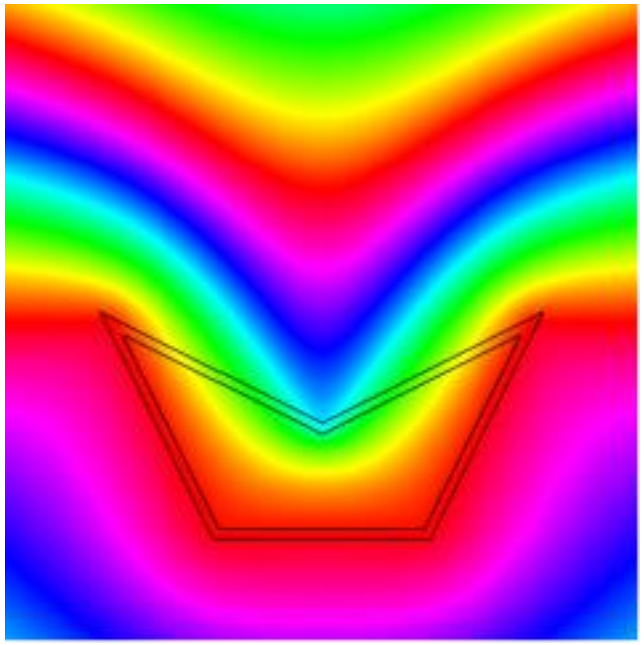
Previous Work


Input


Interpolation/extrapolation
by Mean Value Coordinates

Previous Work


Input


Interpolation/extrapolation
by Mean Value Coordinates

3D Mean Value Coordinates

For barycentric coordinates, we wish to compute weights *w*i that satisfies

$$v = \frac{\sum_i w_i p_i}{\sum_i w_i} \longrightarrow \sum_i w_i (p_i - v) = 0$$

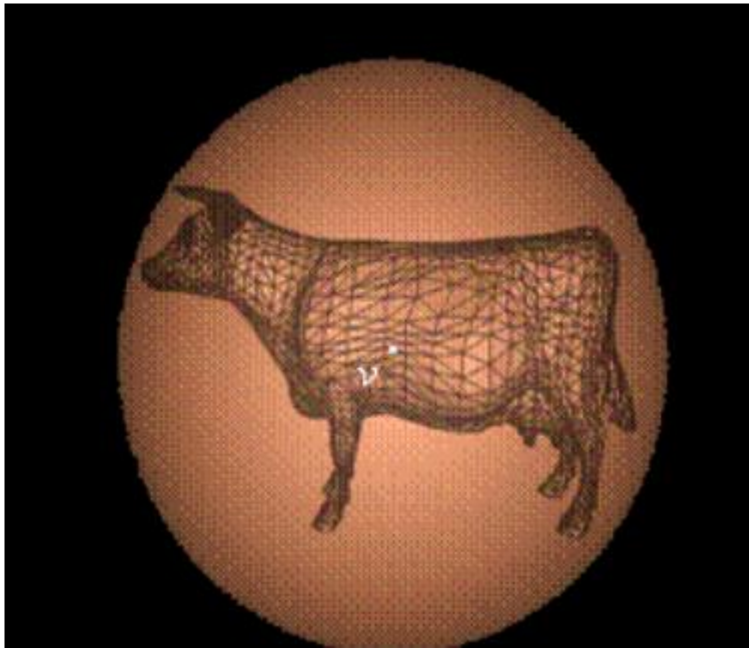
$$\lambda_i = \frac{w_i}{\sum_j w_j}$$

3D Mean Value Coordinates

For barycentric coordinates, we wish to compute weights wi that satisfies

$$v = \frac{\sum_i w_i p_i}{\sum_i w_i} \longrightarrow \sum_i w_i (p_i - v) = 0$$

Projecting the surface of the mesh onto a unit sphere whose center is at v

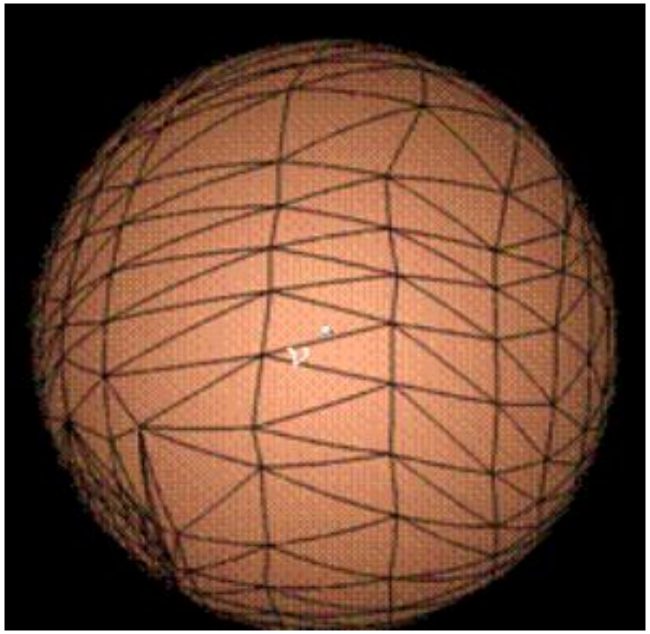


3D Mean Value Coordinates

For barycentric coordinates, we wish to compute weights wi that satisfies

$$v = \frac{\sum_i w_i p_i}{\sum_i w_i} \longrightarrow \sum_i w_i(p_i - v) = 0$$

Projecting the surface of the mesh onto a unit sphere whose center is at v

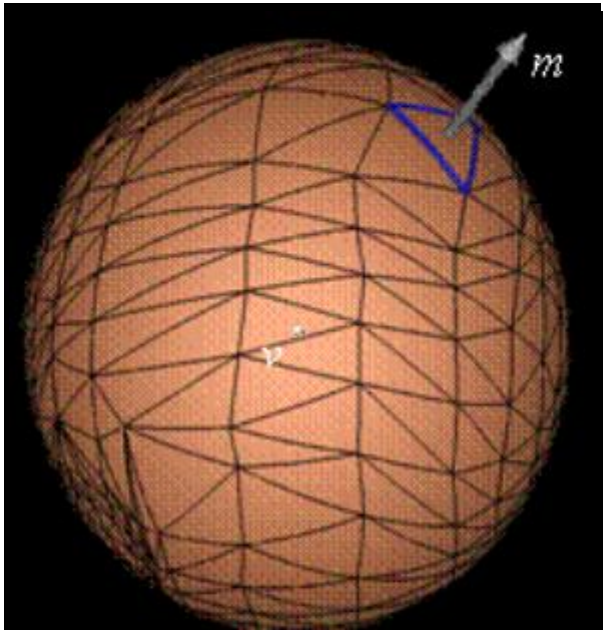


3D Mean Value Coordinates

For barycentric coordinates, we wish to compute weights wi that satisfies

$$v = \frac{\sum_i w_i p_i}{\sum_i w_i} \longrightarrow \sum_i w_i (p_i - v) = 0$$

Mean vector *m* (integral of unit normal over spherical triangle)



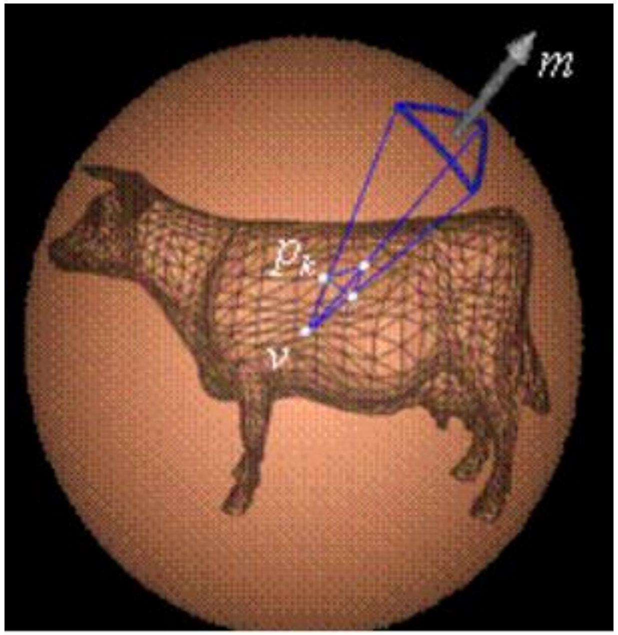
3D Mean Value Coordinates

For barycentric coordinates, we wish to compute weights wi that satisfies

$$v = \frac{\sum_i w_i p_i}{\sum_i w_i} \longrightarrow \sum_i w_i (p_i - v) = 0$$

Mean vector *m* (integral of unit normal over spherical triangle)

$$m = \sum_{k=1}^{3} u_k (p_k - v)$$



3D Mean Value Coordinates

For barycentric coordinates, we wish to compute weights wi that satisfies

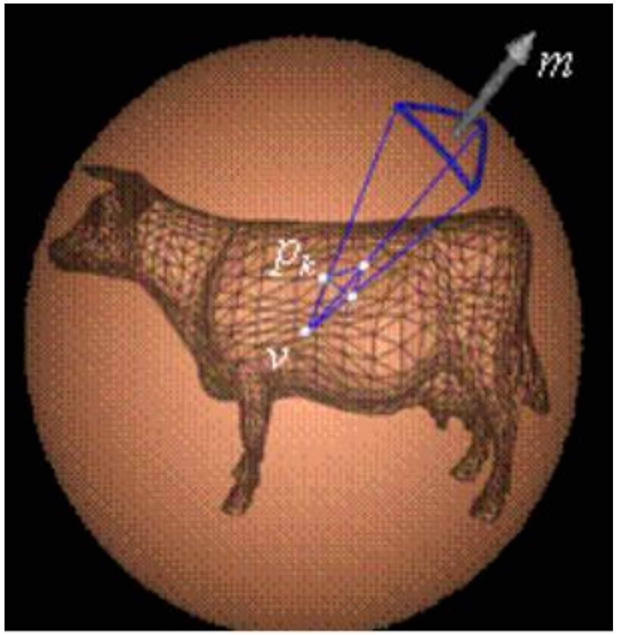
$$v = \frac{\sum_i w_i p_i}{\sum_i w_i} \longrightarrow \sum_i w_i (p_i - v) = 0$$

Mean vector m (integral of unit normal over spherical triangle)

$$m = \sum_{k=1}^{3} u_k (p_k - v)$$

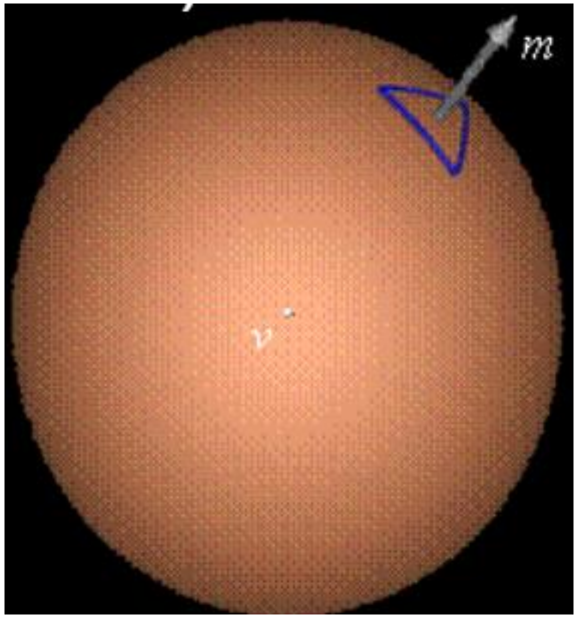
Stoke's theorem : $\sum_j m_j = 0$

$$\longrightarrow \sum_j \sum_k u_k^j (p_k - v) = 0$$



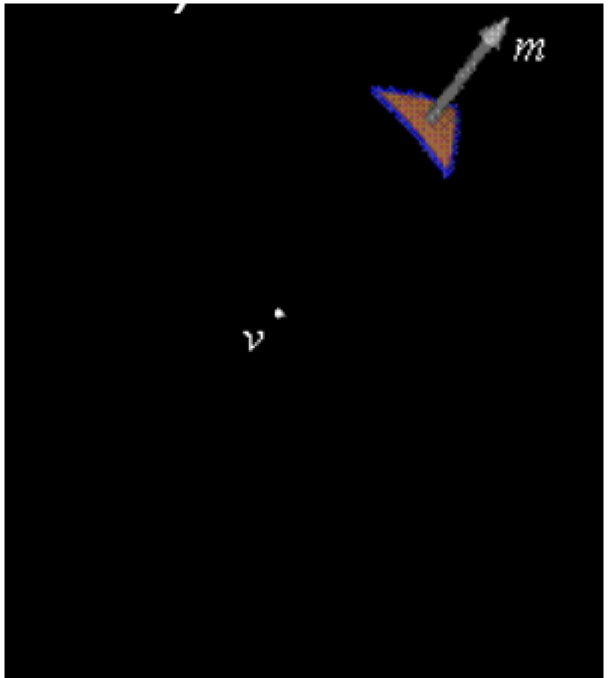
Computing the Mean Value Vector

Given the spherical triangle, compute the mean value vector *m* (integral of a unit normal)



Computing the Mean Value Vector

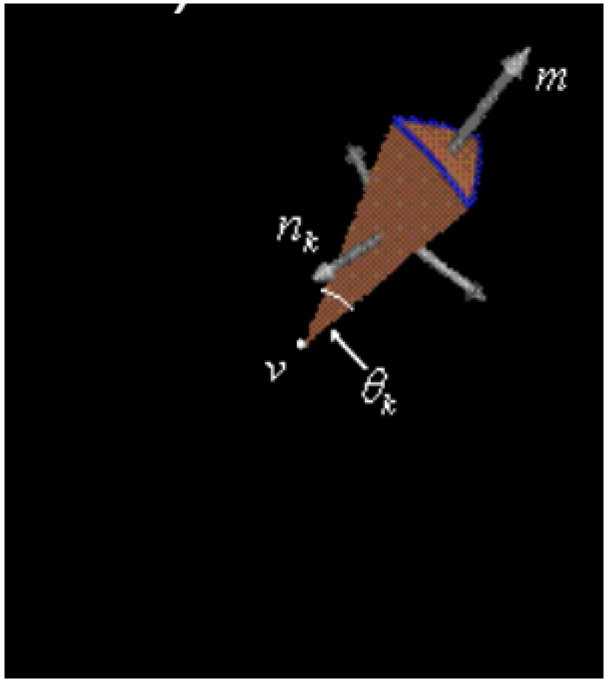
Given the spherical triangle, compute the mean value vector *m* (integral of a unit normal)



Computing the Mean Value Vector

Given the spherical triangle, compute the mean value vector *m* (integral of a unit normal)

Build wedge with face normal n_k



Apply Stoke's theorem to the wedges and the spherical triangle

$$\sum_{k=1}^{3} \frac{1}{2}\theta_k n_k + m = 0$$

Process of computing Mean Value Coordinates

Compute mean vector by solving for *m*

$$\sum_{k=1}^{3} \frac{1}{2}\theta_k n_k + m = 0$$

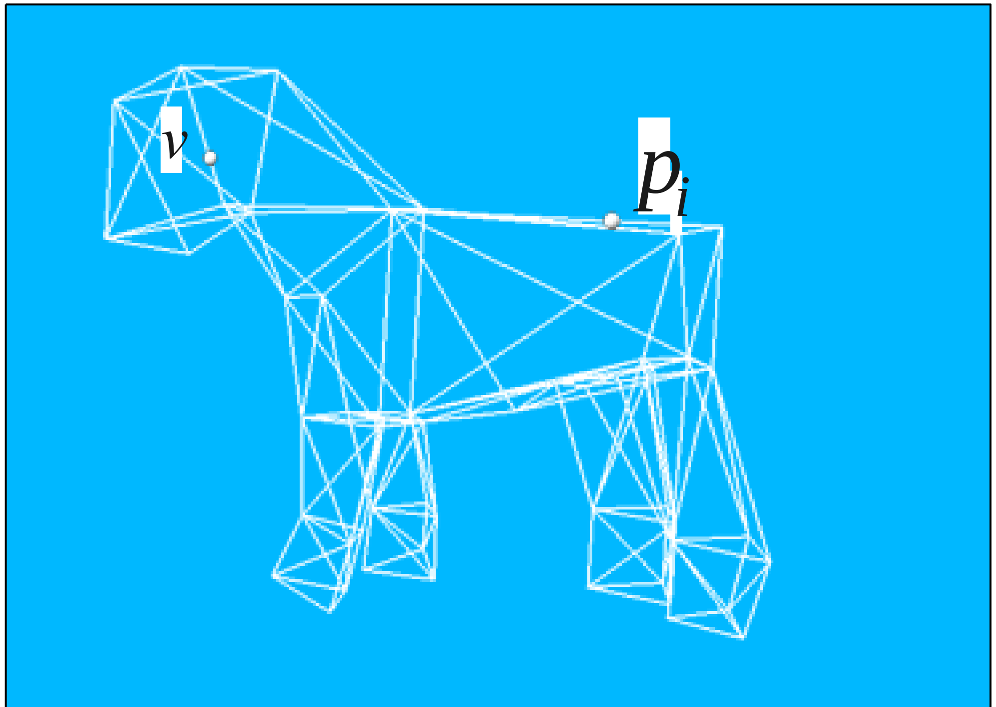
Calculate weights by solving for *u_k* in

$$m = \sum_{k=1}^{3} u_k(p_k - v)$$

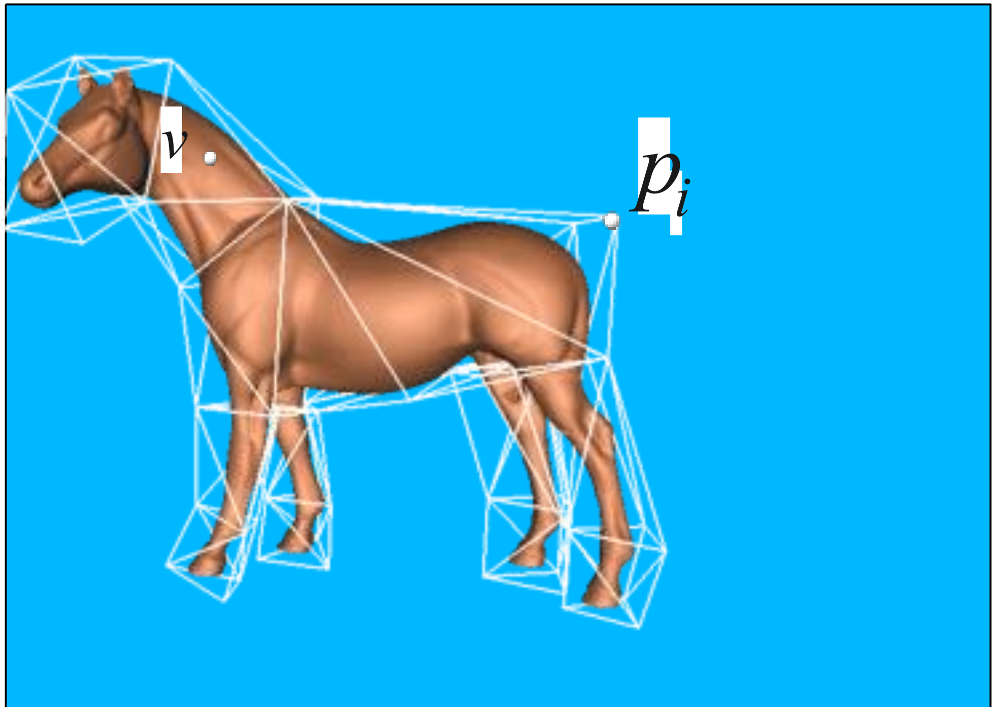
Sum over all triangles to compute what you want

$$f[v] = \frac{\sum_j \sum_{k=1}^{3} u_k^j f_k^j}{\sum_j \sum_{k=1}^{3} u_k^j}$$

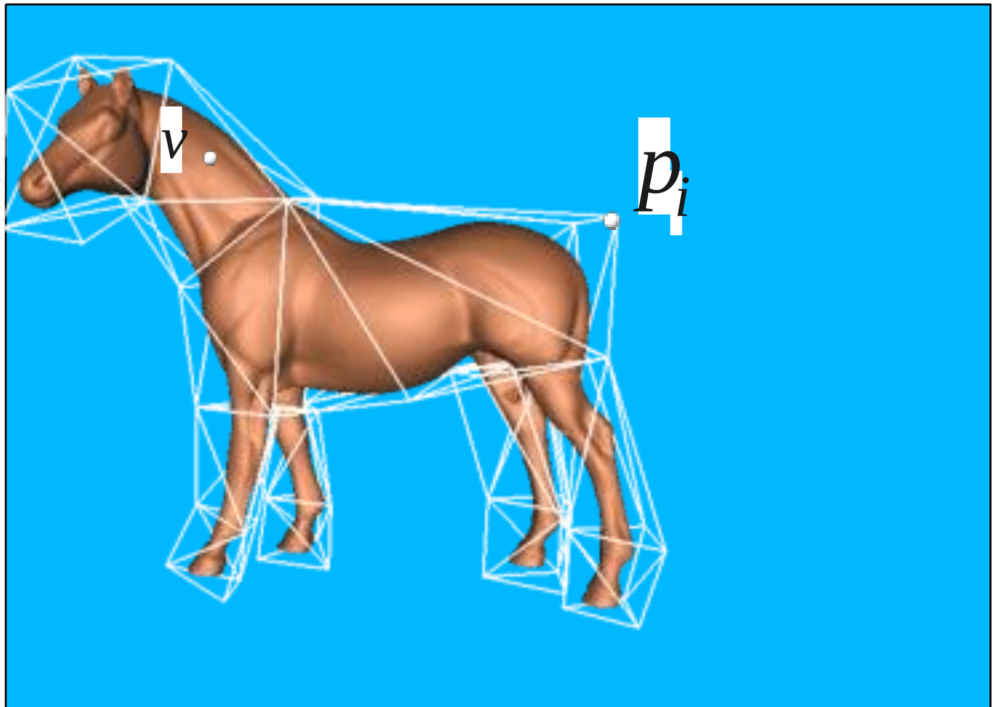
Application:
Surface Deformation

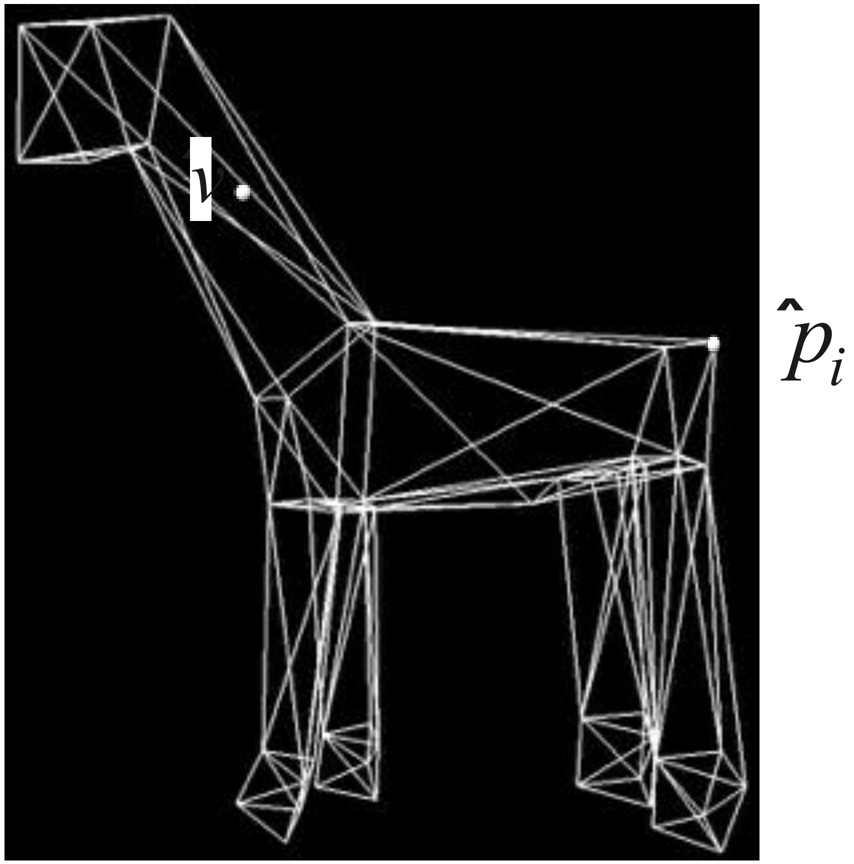


Application: Surface Deformation

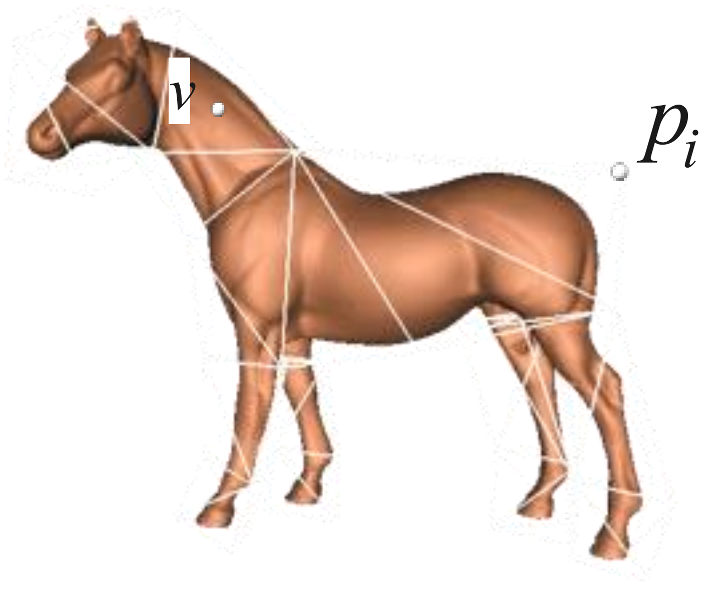


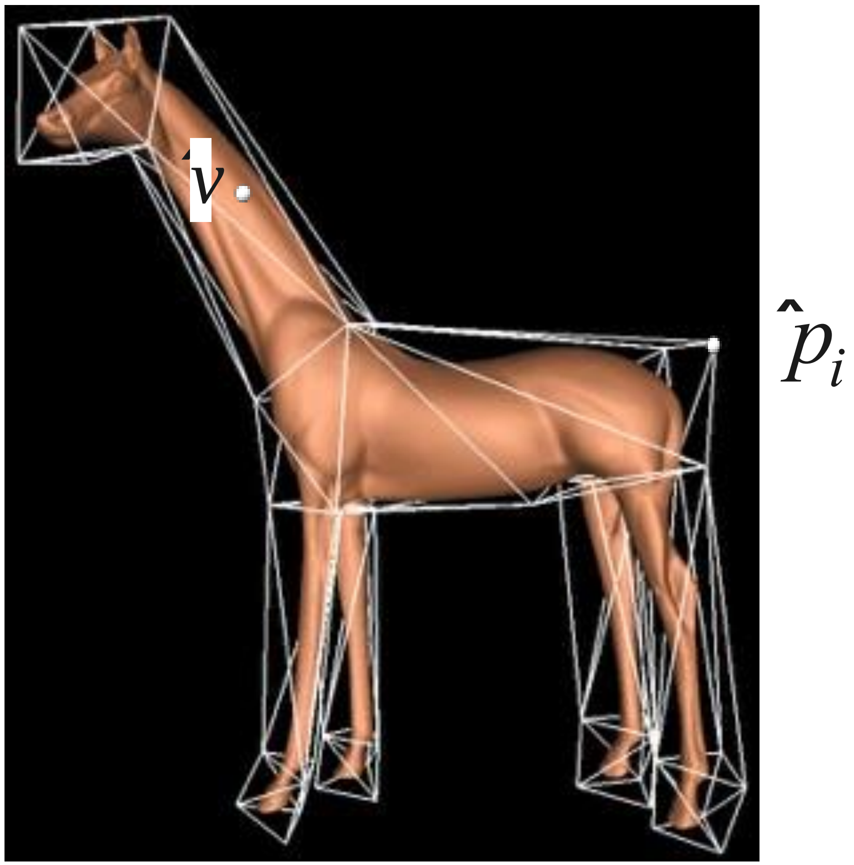
Application:
Surface Deformation





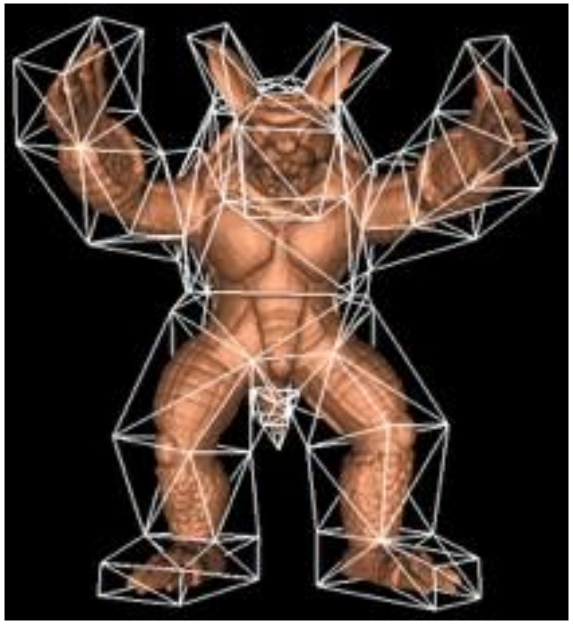
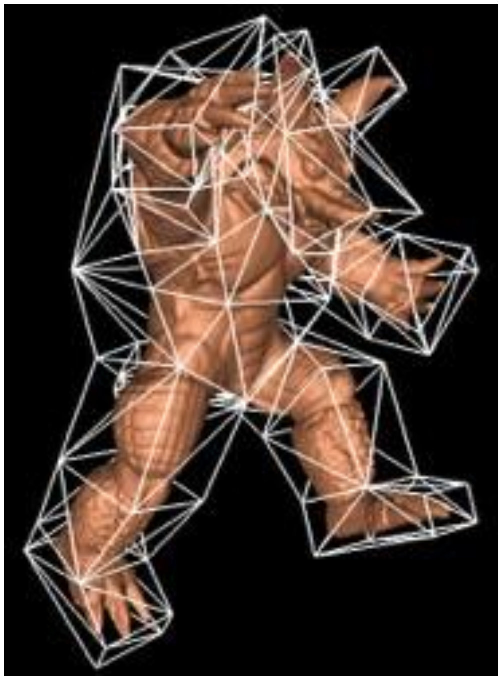
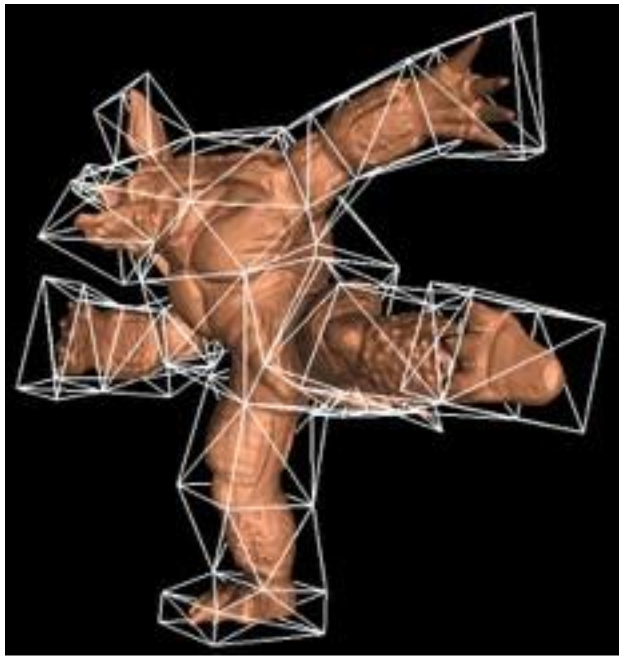
Application: Surface Deformation





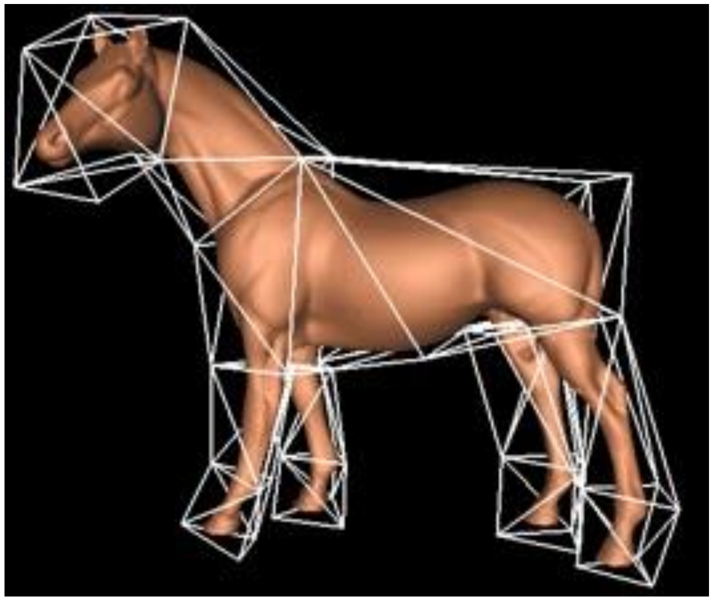
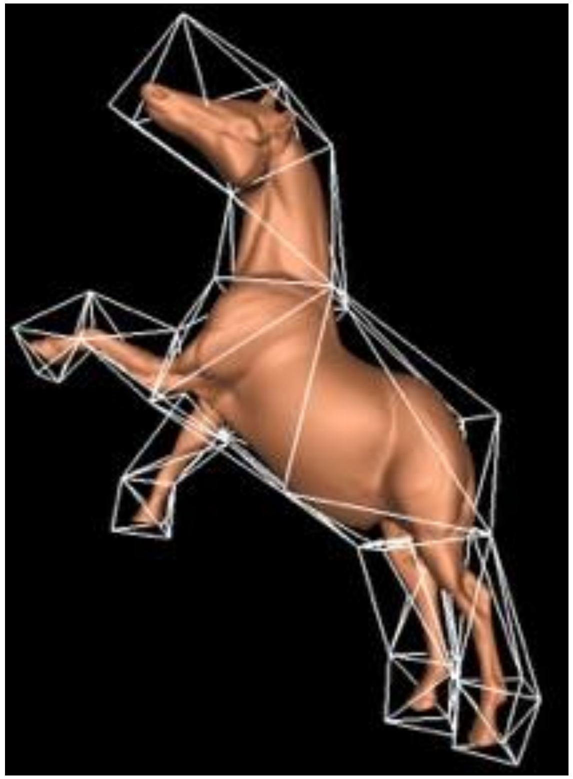
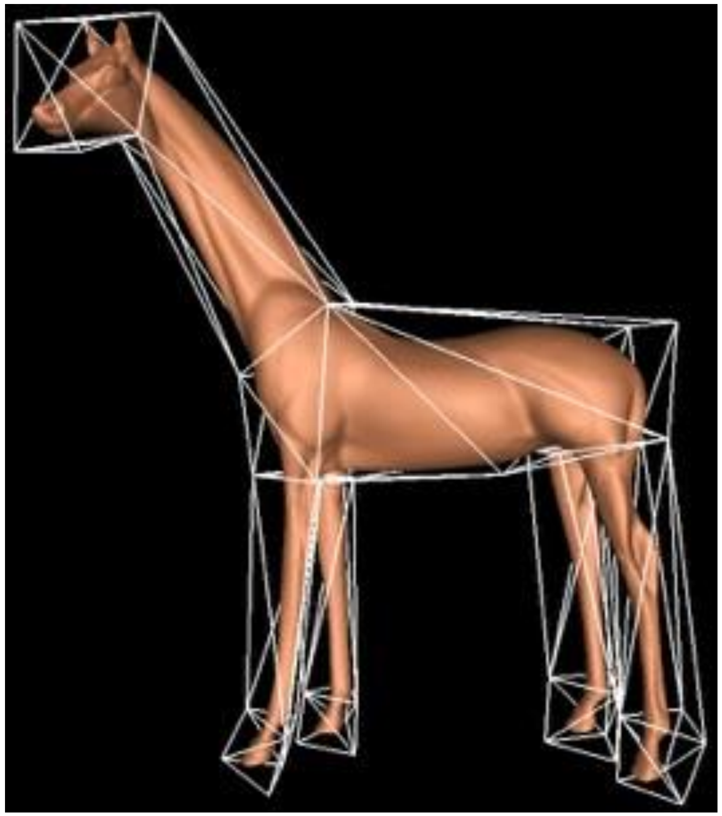
Applications
Surface Deformation

Control Mesh	Surface	Computing Weights	Deformation
216 triangles	30,000 triangles	1.9 seconds	0.03 seconds

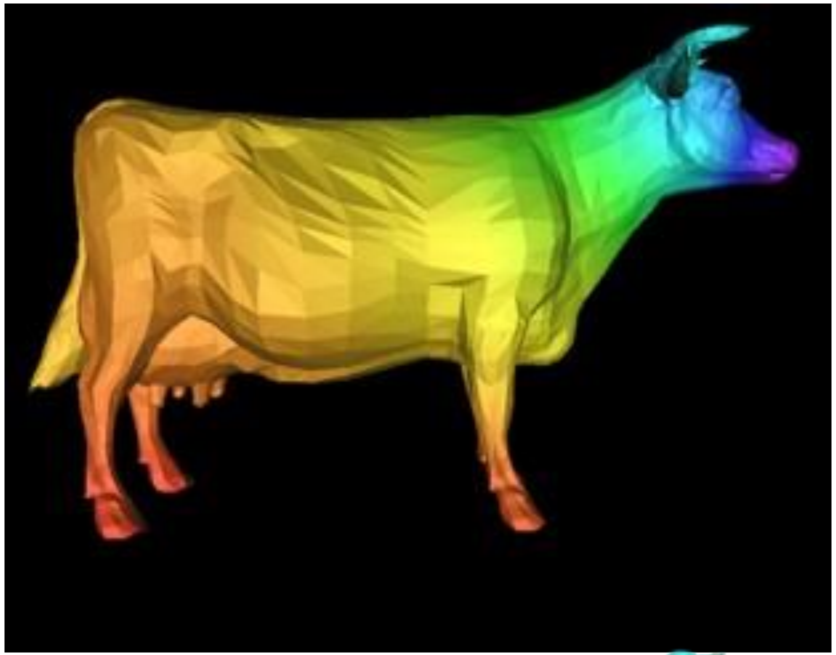
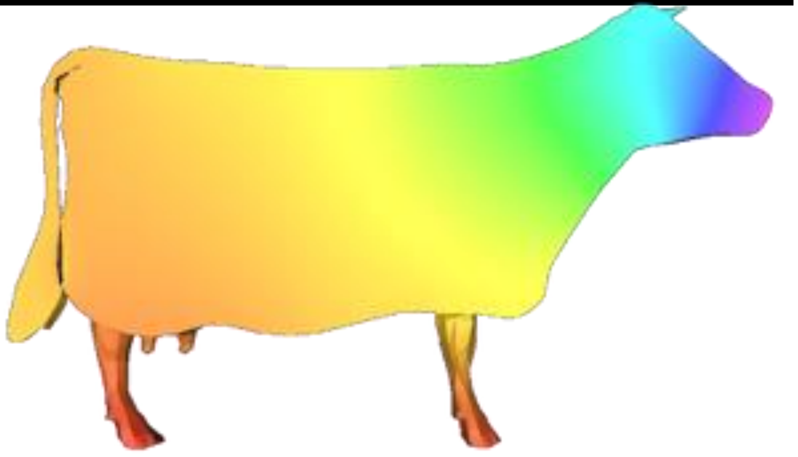




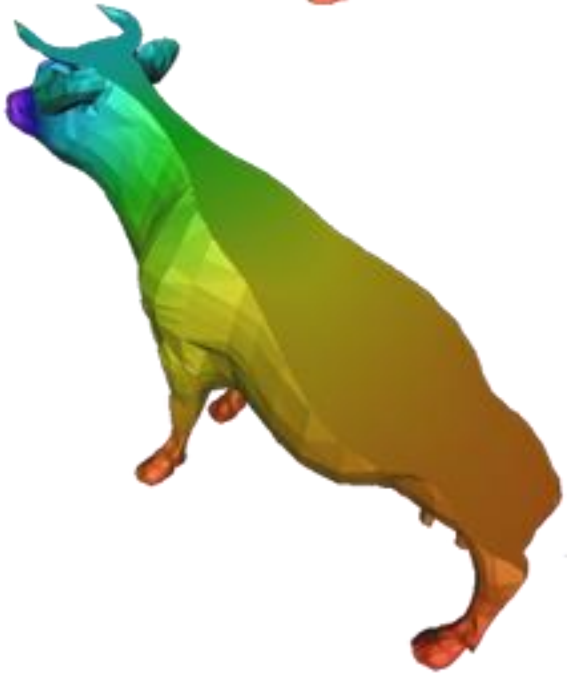
Applications
Surface Deformation

Control Mesh	Surface	Computing Weights	Deformation
98 triangles	96,966 triangles	3.3 seconds	0.09 seconds

Applications
Boundary Value Problems

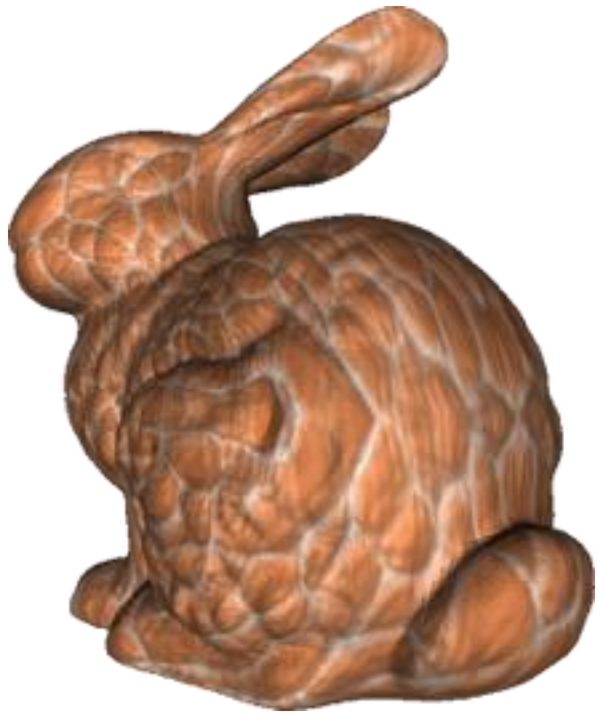
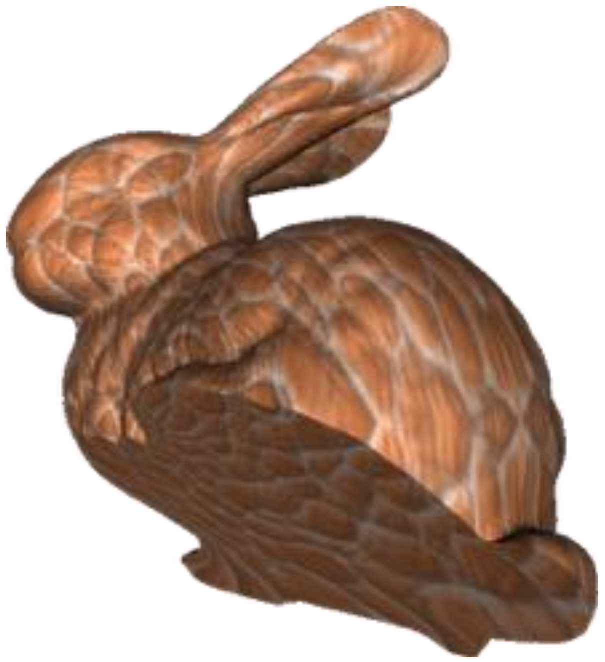





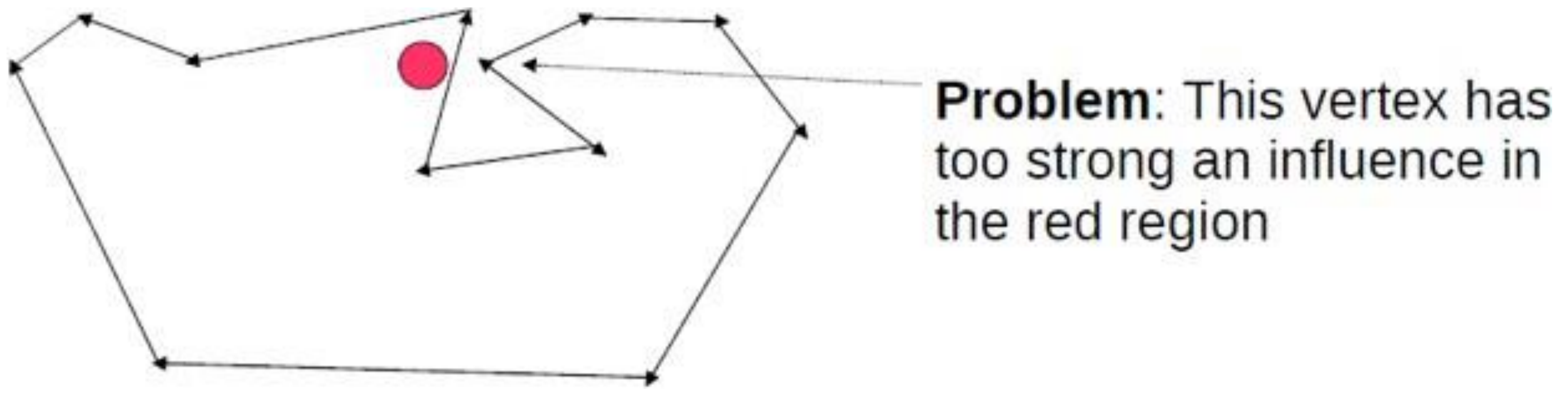
Applications
Volume Textures

Interpolating the texture coordinates

Extend texture to interior

Interpolation : problem polygon


Problem: This vertex has too strong an influence in the red region

http://www.youtube.com/watch?v=egf4m6zVHUI

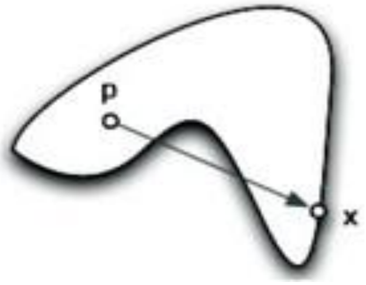
Shows the problem of mean value coordinates at 1:54

- **ill-formed polygon**
  - poor interpolation results based on point distance due to narrow concavity

Harmonic Coordinates

The problem with Mean Value Coordinates is that their values are affected by the Euclidean distance but not the distance that needs to be travelled

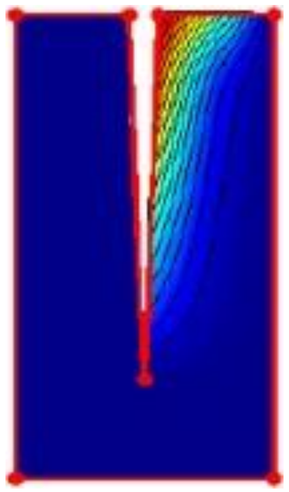
Affected by geometrically close points



Harmonic Scalar Field

– The value depends on the distance travelled inside the polygon

Not affected by the Euclidean distance but the geodesic distance

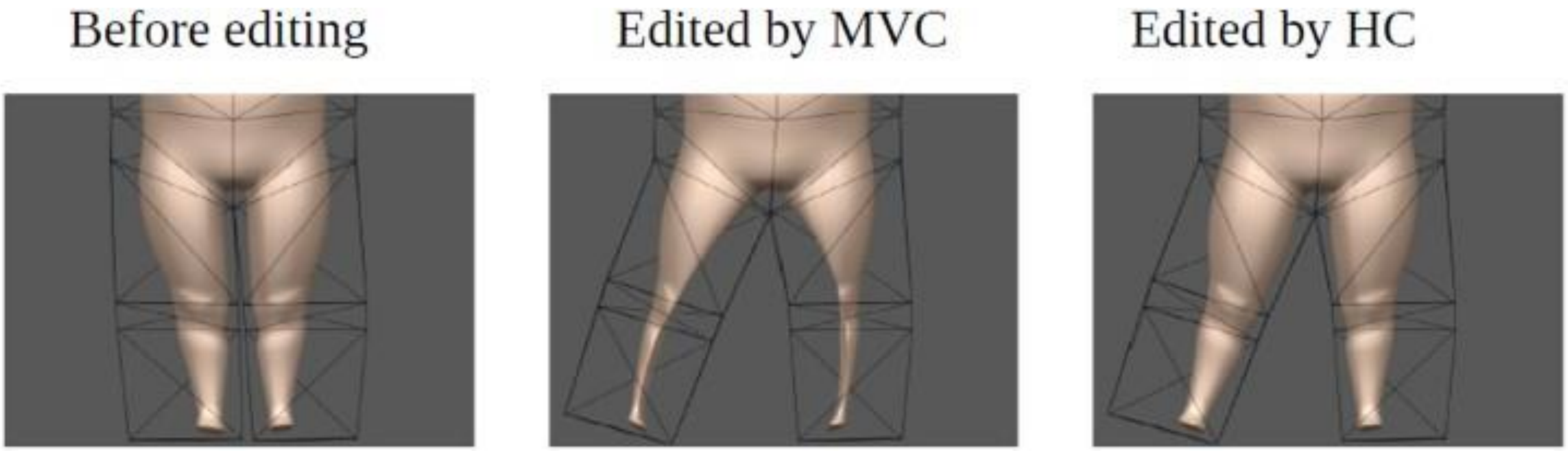


Harmonic Coordinates : Procedure

- For each vertex i of the cage, set the potential value of $\mathbf{v}_i$ to 1, and the rest to 0
- Compute the potential for all the points inside the the polygon (p_i) by solving a Laplace equation (as taught in Lecture 12)
- Normalize p_i : $\rho_i = \frac{p_i}{\sum_j p_j}$
- $(\rho_1, \ldots, \rho_n)$ are the harmonic coordinates
- The global position of the point can be expressed in the form

$\mathbf{P} = \rho_1 \mathbf{v}_1 + \rho_2 \mathbf{v}_2 + \ldots + \rho_n \mathbf{v}_n$

Harmonic Coordinates: comparison



Barycentric Coordinates Summary

Mean value coordinates

- Can handle concave objects well to some extent
- Values affected by geometrically close control points
- Defined outside the polygon too

Harmonic Coordinates

- Can handle concave objects well Only defined inside the polygon

Summary

- As-rigid-as-possible interpolation
  - Introducing the idea of changing shapes while preserving the rigidity of objects
- Laplacian coordinates
  - a linear shape editing scheme that is fast and preserves local details
- Generalized barycentric coordinates
  - Allow users to edit shapes using the lattice

Readings

- Alexa et al. “As-rigid-as-possible shape interpolation SIGGRAPH ’00
- Deformation Transfer for Triangle Mesh, Sumner et al. SIGGRAPH 2004
- Igarashi et al. “As-rigid-as-possible shape manipulation”, SIGGRAPH ‘05
- O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl and H.-P. Seidel, “Laplacian Surface Editing”, Eurographics Symposium on Geometry Processing (2004)
- Large Mesh Deformation Using the Volumetric Graph Laplacian

  Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, Heung-Yeung Shum. *ACM SIGGRAPH 2005*, 496-503.
- Harmoinc Coordinates for Character Articulation: Joshi et al. SIGGRAPH 2007
- Mean Value Coordinates for Closed Triangular Meshes, Ju et al. SIGGRAPH 2005

SVD operation

The columns of U are the eigenvectors of $A^{\mathrm{T}}A$
The columns of V are the eigenvectors of $A\,A^{\mathrm{T}}$
The diagnol values of $\sum$ are the square roots of the eigenvalues of $A\,A^{\mathrm{T}}$ or $A^{\mathrm{T}}A$

$$A = U \sum V^T$$

$$A^{\mathrm{T}}A = V{\sum}^T U^T U \sum V^T = V{\sum}^T \sum V^T$$

$$AA^{\mathrm{T}} = U \sum V^T\ V {\sum}^T U^T = U \sum\ {\sum}^T U^T$$

Mean Value Interpolation

Assuming P is a close form surface in R^3, let p[x] be a paremeterization of P, x are the parameters to define p and f[x] be a function defined over P.

We project p[x] onto a unit sphere Sv centered at v.

The mean value interpolant is defined as

$$\hat{f}(v) = \frac{\int_x \ w[x,v]f[x]dS_v}{\int_x \ w[x,v]dS_v}$$

where $w[x,y] = \frac{1}{|p[x]-v|}$

Mean Value Interpolation

Mean value interpolant satisfies all the requirements for barycentric coordinates but is continuous (not defined per vertex).

We compute the weights per triangle

This will be