Computer Graphics International 2006

Skeleton-Based Shape Deformation using Simplex Transformations

Han-Bing Yan, Shi-Min Hu and Ralph Martin

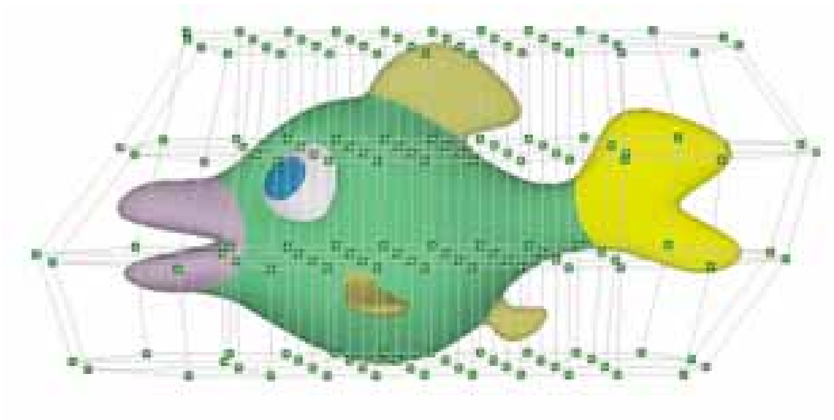
2006-6-29

Outline

- Motivation
- Introduction
- Mesh segmentation using skeleton
- Skeleton-based shape deformation
- Results
- Conclusions and future work

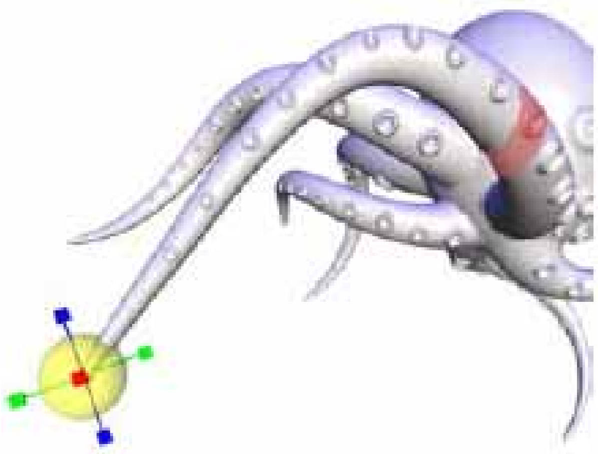
Motivation

- Applications of Meshes Deformation :
  - Animation & modelling
- Traditional methods [Sederberg et al. 1986]
  - FFD & other space warping methods



Motivation

- Differential methods
  - Laplacian coordinates [Alexa2003, Lipmann2004]
  - Possion-based gradient field method [Yu2004]



Motivation

- Skeleton-based Method (Skinning, Envelopes)
  - Each vertex is controlled by several bones

$$P' = \sum_{k=1}^{n} w_k \, P M_k$$

  - The final position of each vertex is a weighted sum of the vertex position in coordinate frames fixed to moving bones
  - Widely used in commercial software

Motivation

- Pros and Cons of Skeleton-based Method
  - + Gives *natural* control of shape feature deformation.
    - Animals have skeletons…
  - - Weights must be carefully selected
    - to retain original mesh features
    - to avoid flipping and self-intersection
  - - No weight selection method works well in all cases
  - - Weight selection is a tedious manual process

Motivation

- Traditional Skeleton-based method
  - Does not used vertex connectivity information directly, though mesh has provided this information.
- Target of our method
  - We want to develop a new skeleton-based deformation method which uses vertex connectivity information directly.

Our Method

- Key idea in our method
  - We use skeleton to drive *simplex*, not *vertex*:
    - Simplices contain shape connectivity information
- Input
  - Original image or shape (2D or 3D triangle mesh), Original skeleton , Deformed skeleton
- Output
  - Deformed mesh

Simplex Transformation

- Simplex Transformation
  - There exists a unique transformation between two (triangles, tetrahedra) in (2D, 3D)

$$v_i = Ru_i + T$$

$$R = VU^{-1}$$

$$V = [v_1 - v_3 \quad v_2 - v_3] \qquad V = [v_1 - v_4 \quad v_2 - v_4 \quad v_3 - v_4]$$

$$U = [u_1 - u_3 \quad u_2 - u_3] \qquad U = [u_1 - u_4 \quad u_2 - u_4 \quad u_3 - u_4]$$

2D 3D

Main Steps

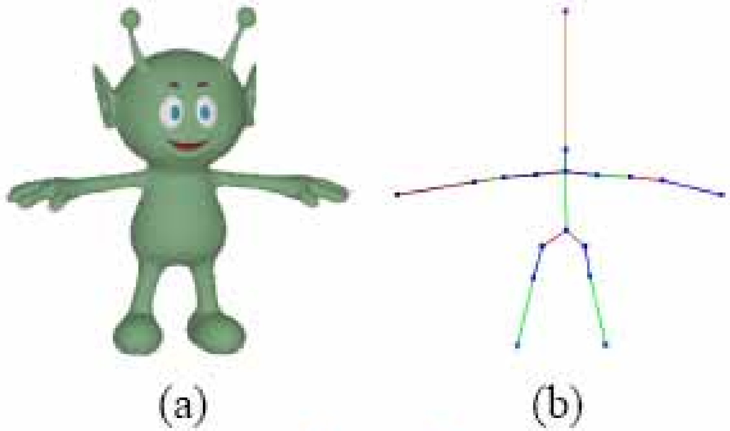
- Steps
  - Segment mesh using skeleton
    - i.e. allocate each simplex to a *controlling* bone
  - Calculate transformation matrix for each bone
  - Ensure connectivity between final simplices while keeping the transformation for each simplex as close as possible to that of its controlling bone

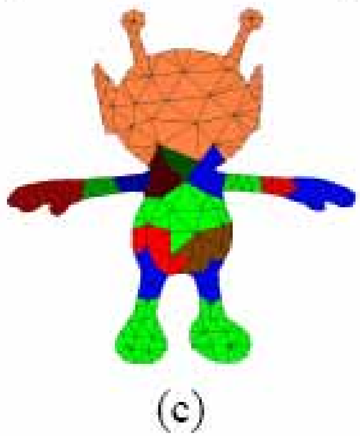
Mesh Segmentation

- Aim
  - ➢ Decide which bone controls each triangle
    - Each triangle is controlled by one bone
  - ➢ This is a segmentation problem
- Two possible approaches
  - ➢ Using Euclidean distance between bone and triangle
  - ➢ Using the shortest path distance on mesh between bone and triangle

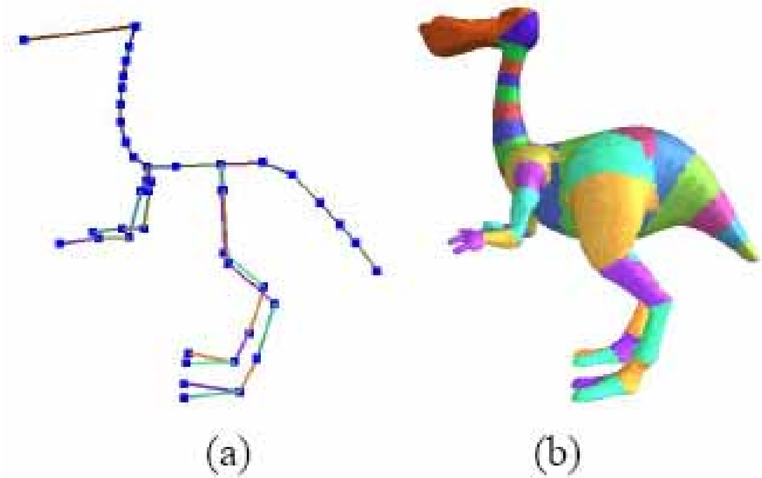
Mesh Segmentation

- We show some results first





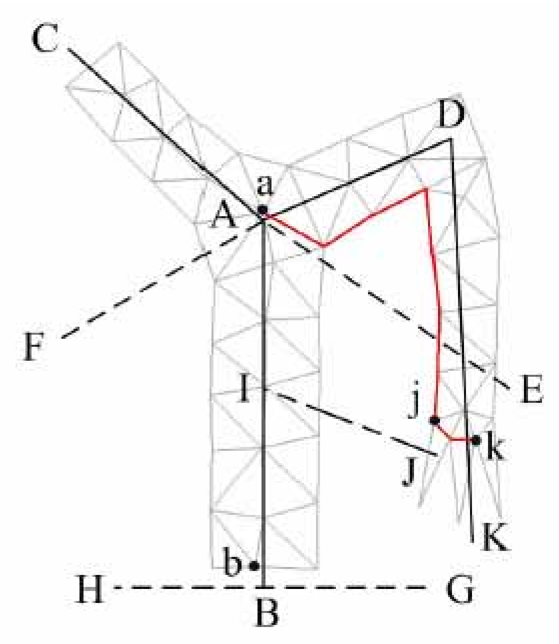
2D



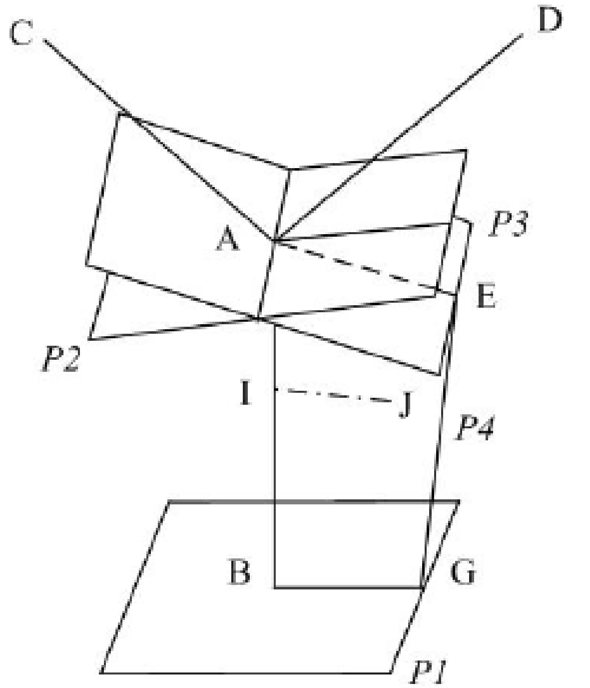
3D

Mesh Segmentation

- Define the *control domain* of each bone using *Range lines*(2D), *Range planes*(3D)



2D



3D

Mesh Segmentation

- Each vertex is in the control domain of at least one bone
- One vertex maybe in the control domain of several bones
- *Effective Distance:* d_{eff}
- *Effective Distance with Penalty*:

$$d_{effpen} = d_{eff} + n\delta$$

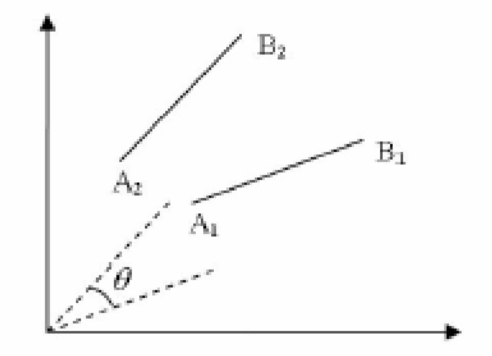
Mesh Segmentation

- If $\min d_{effpen} < \delta$, the bone with minimal *Effective Distance with Penalty* is the control bone of this triangle
- Otherwise, calculate the *shortest path distance* from the triangle center to the two joints of this bone. The bone with the minimal shortest path distance is the control bone of this triangle.

Transformation Matrix of Bones

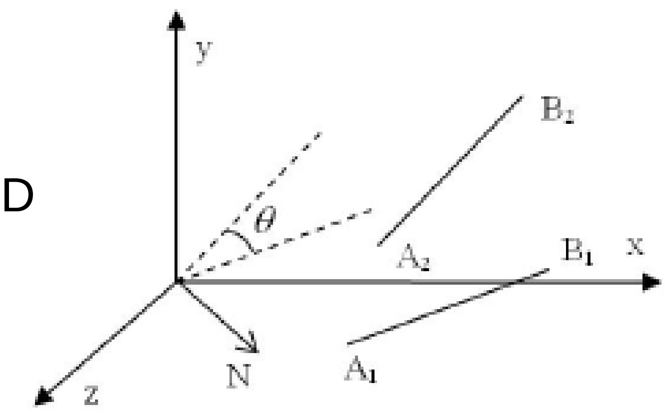
- Without scaling:

2D



$$R' = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

3D



$$R' = \begin{bmatrix} a^2 + (b^2 + c^2)\cos\theta & ab(1-\cos\theta) + c\sin\theta & ac(1-\cos\theta) - b\sin\theta \\ ab(1-\cos\theta) - c\sin\theta & b^2 + (a^2 + c^2)\cos\theta & bc(1-\cos\theta) + a\sin\theta \\ ac(1-\cos\theta) + b\sin\theta & bc(1-\cos\theta) - a\sin\theta & c^2 + (a^2 + b^2)\cos\theta \end{bmatrix}$$

Transformation Matrix of Bones

- Also allowing scaling of bones
  - Rotate the bone to the x axis, scale it along the x axis, then rotate it back.

$$S' = R_{S2} S R_{S1}$$

  - The transformation matrix of a bone is the product of a rotation matrix and a scaling matrix.

$$M' = S' R'$$

- Translation vector of bones

Simplices in 3D

- 2D Triangle Mesh
  - Simplices used are triangles themselves
- 3D Triangle Mesh
  - One vertex needs to be added to each triangle to create a simplex

$$v_4 = \frac{(v_1 + v_2 + v_3)}{3} + \frac{(v_2 - v_1) \times (v_3 - v_2)}{\sqrt{(v_2 - v_1) \times (v_3 - v_2)}}$$

Optimization

- We want to use the bone's transformation matrix for the simplices it controls
  - Problem: if every simplex uses the same transform as its controlling bone, there will be *gaps* between adjacent simplices which belong to different bones
- Solution
  - Let the simplices transform as much as possible like its controlling bone while enforcing vertex connectivity requirements

Optimization

- We optimise an error energy function

$$E = \sum_{i=1}^{n} A_i \left\| M_i - M_i^{'} \right\|_F^2$$

  - The variables are the vertex coordinates of the deformed mesh

- By minimizing the error energy function, we get a result which
  - Keep vertex connectivity requirements
  - follows the bones as much as possible

Optimization

- Linear Equation Solver
  - This quadratic optimization problem can be easily transformed to a *linear system*

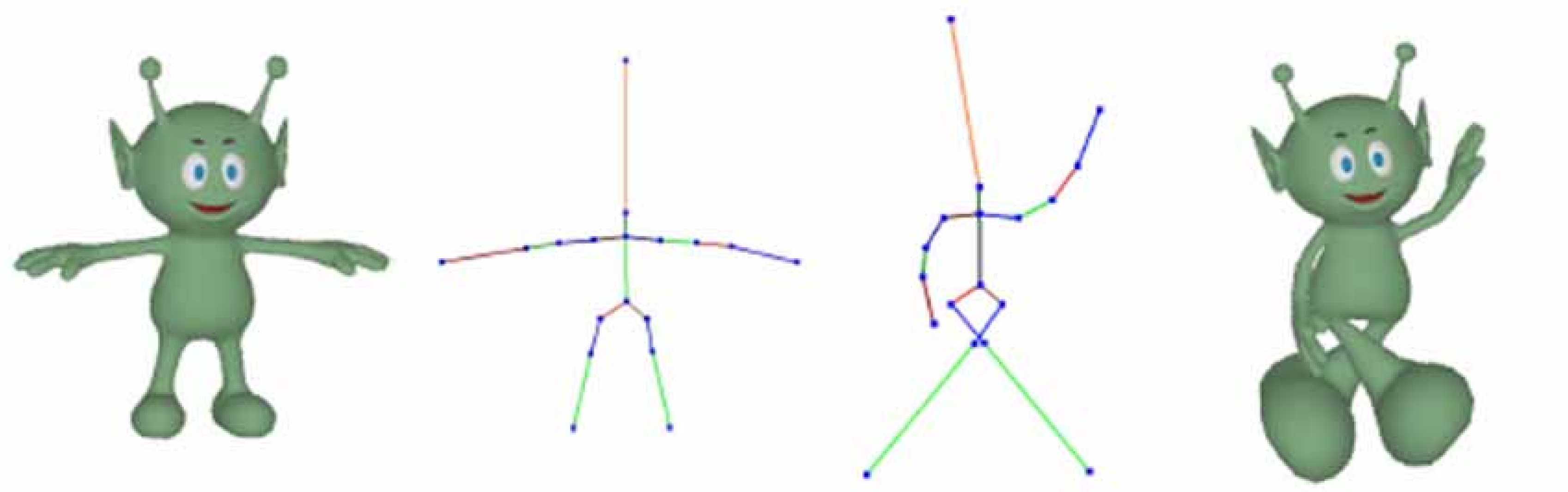
$$KV = d$$

  - *V* is the unknown vector of vertex coordinates
  - The x, y (and z in 3D) coordinates form *separate* linear subsytems

$$KX = d_x \qquad KY = d_y \qquad KZ = d_z$$

Results

- 2D



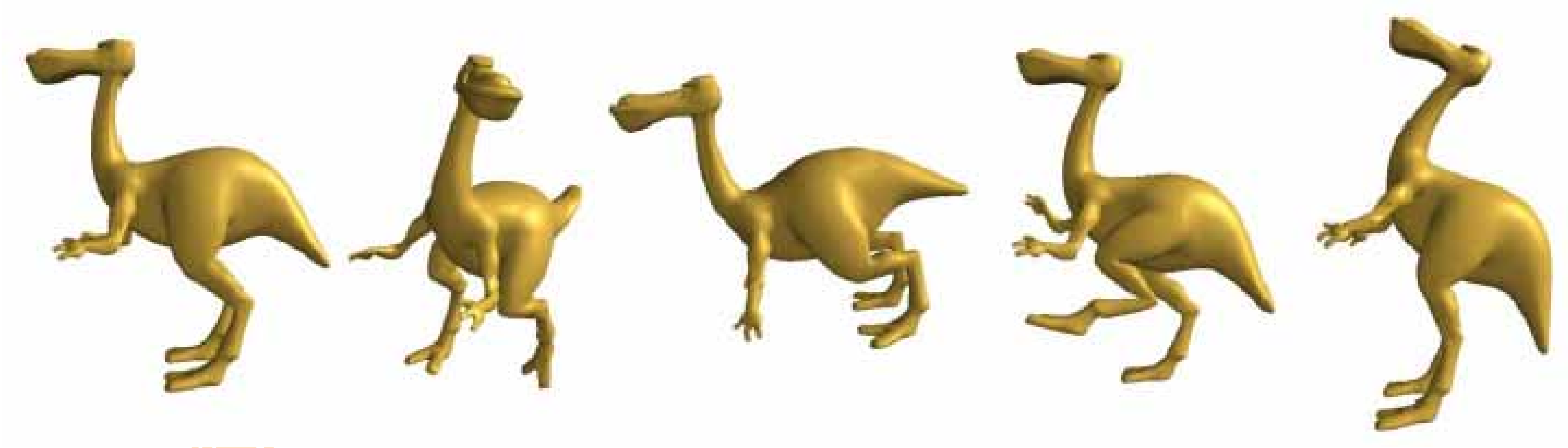
(a) (b) (c) (d)

Results

- 3D

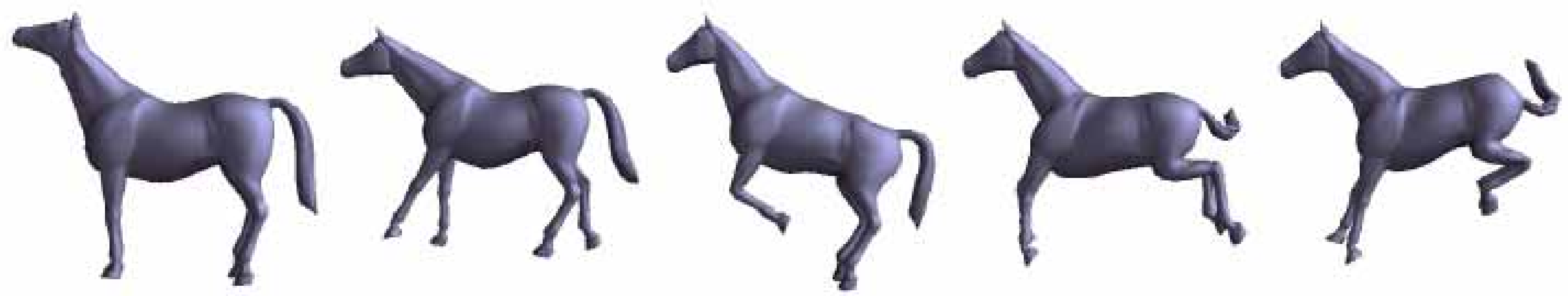
Results

- 3D



Results

- 3D



Extensions of this paper

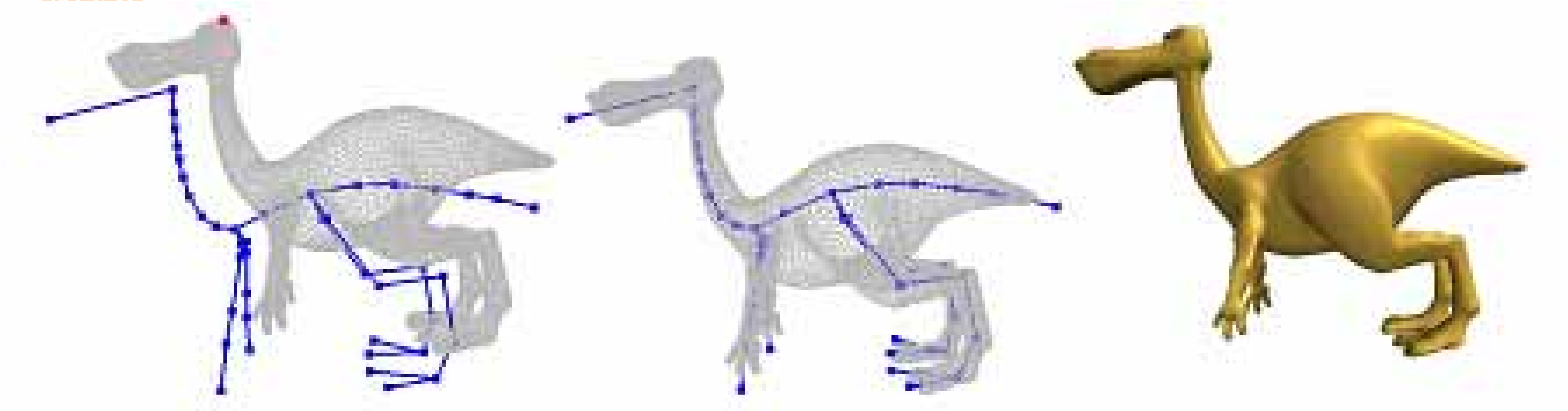
- Small issue with this method
  - ➢ At least one vertex must be *fixed* to avoid a matrix singularity
    - Different choice of fixed vertex will result in a different position for the deformed mesh
    - but it has the same shape
- Solution
  - ➢ Include the translation vector in the Error Energy Function

Extensions of this paper

- New Error Energy Function

$$E = \sum_{i=1}^{n} A_i \left(\| R - R' \|_F^2 + \beta \| T - T' \|_2^2 \right)$$

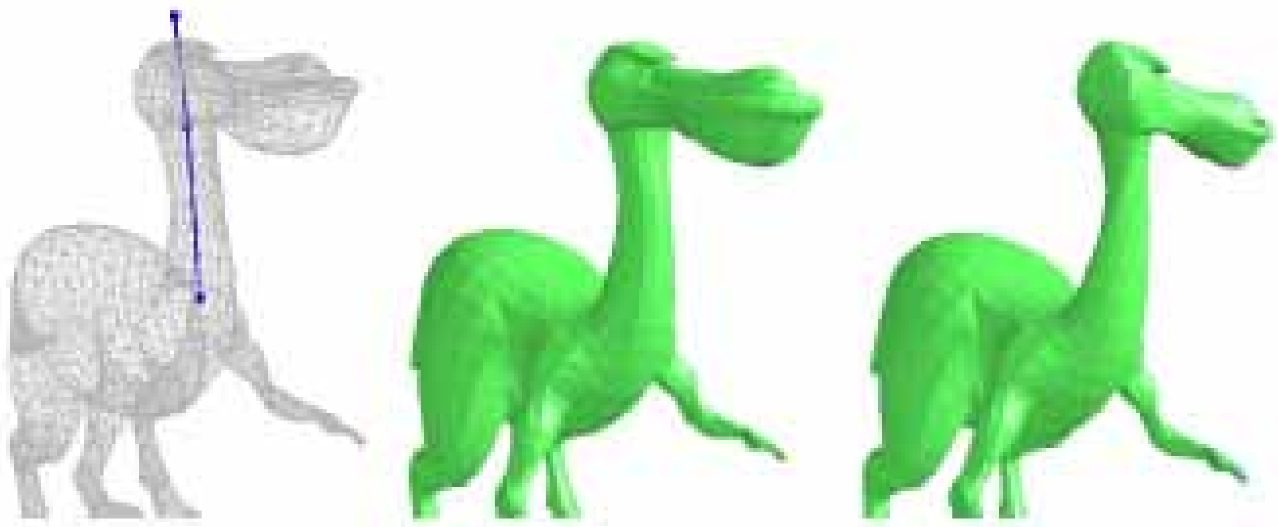
  - No vertex needs to be fixed, and the deformed mesh moves with the skeleton.



(a) (b) (c)

Extensions of this paper

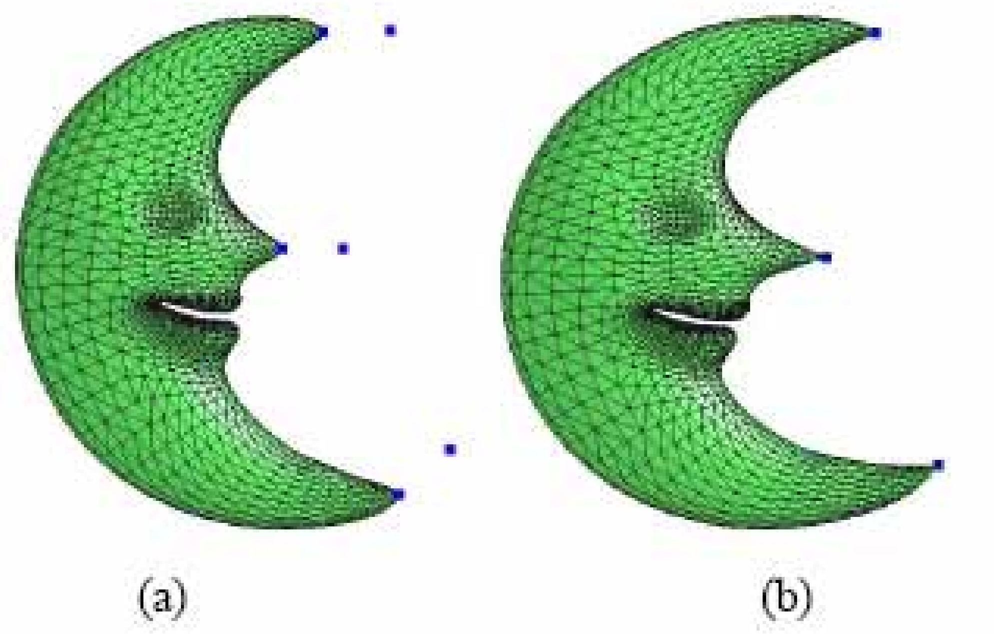
- Twist
  - The above method is very easily to extend to twist the mesh using bones or line segments.



(a) (b) (c)

Extensions of this paper

- Vertex constraint
  - Even we can deform the mesh without using skeleton, just using control vertices



Conclusion

- Pros and Cons
  - + Uses skeleton to drive simplices, not vertices
    - offering easy control
    - vertex connectivity information is used directly
  - + No weights need to be defined
  - + Experimental results are very good
  - + No flip or self-intersection happens in most cases
  - - Need to solve a linear equation
    - hence this method is slower than traditional methods

Future Work

- Other extensions
  - In the above method, the skeleton can be seen as a tool to modify the simplex transformation matrix
  - The skeleton can also be used to modify other local intrinsic attributes
    - Thus, other skeleton-based deformation methods can be developed: e.g. Laplacian coordinates, gradient fields, and so on.

Q & A