Free-Form Deformation and Other Deformation Techniques

Deformation

Deformation

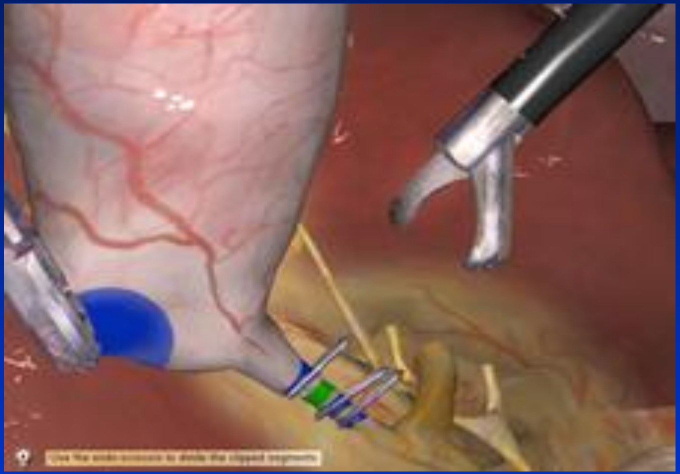
Basic Definition

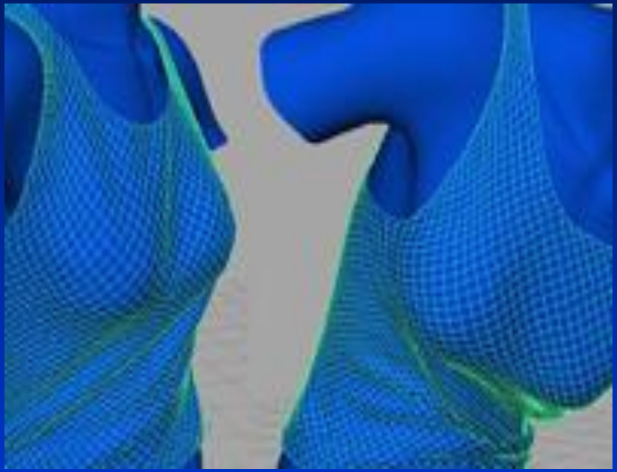
- Deformation: **A transformation/mapping of the positions of every particle in the original object to those in the deformed body**
- **Each particle represented by a point p is moved by $\phi(\cdot)$:**

$$p \rightarrow \phi(t, p)$$

where p represents the original position and $\phi(t, p)$ represents the position at time t

Deformation Applications



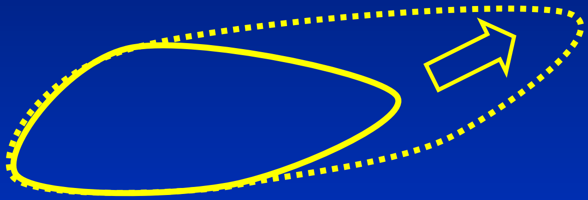


Deforming Objects

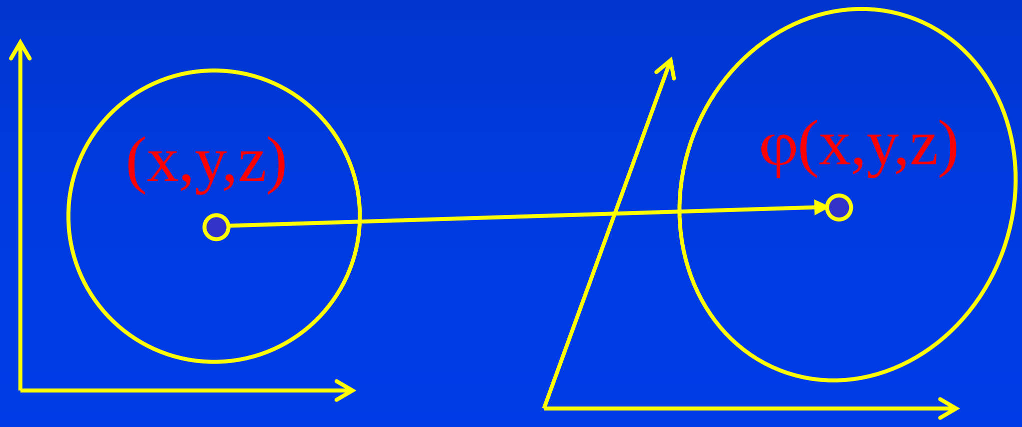
- Changing an object's shape
  - Usually refers to non-simulated algorithms
  - Usually relies on user guidance
- Easiest when the number of faces and vertices of a shape is preserved, and the shape topology is not changed either
  - Define the movements of vertices

Deformation

- Modify Geometry



- Space Transformation

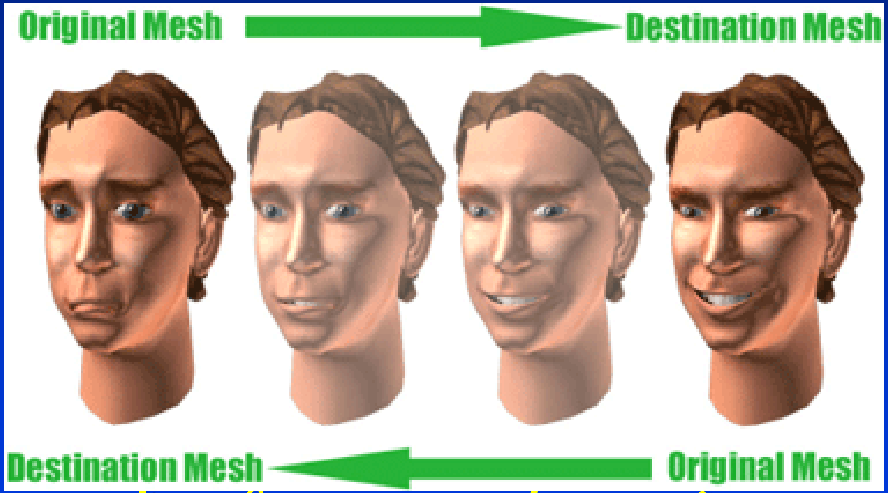


Defining Vertex Functions

- If vertex *i* is displaced by *(x, y, z)* units
  - Displace each neighbor, *j*, of *i* by
    - $(x, y, z) * f(i, j)$
- $f(i,j)$ is typically a function of distance
  - Euclidean distance
  - Number of edges from *i* to *j*
  - Distance along surface (i.e., geodesics)

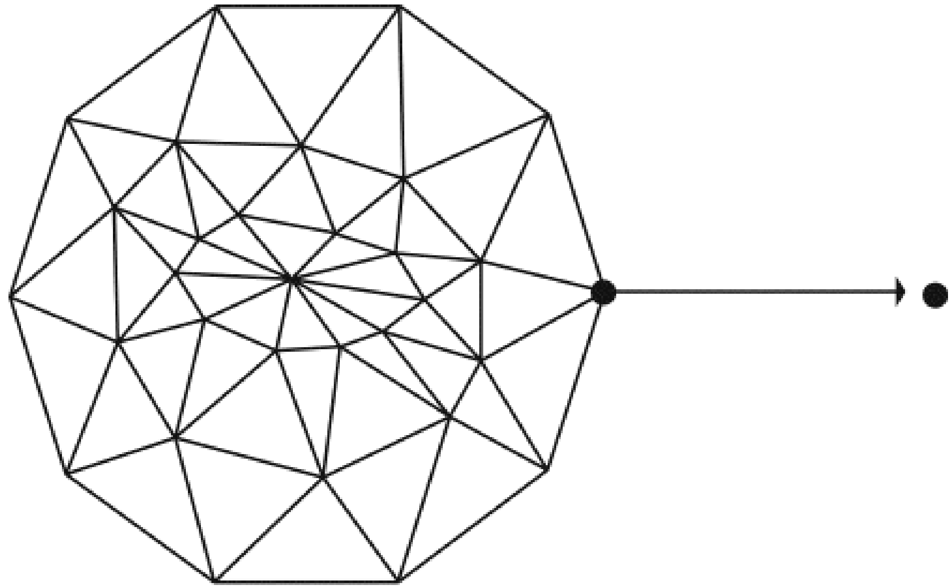
Moving Vertices

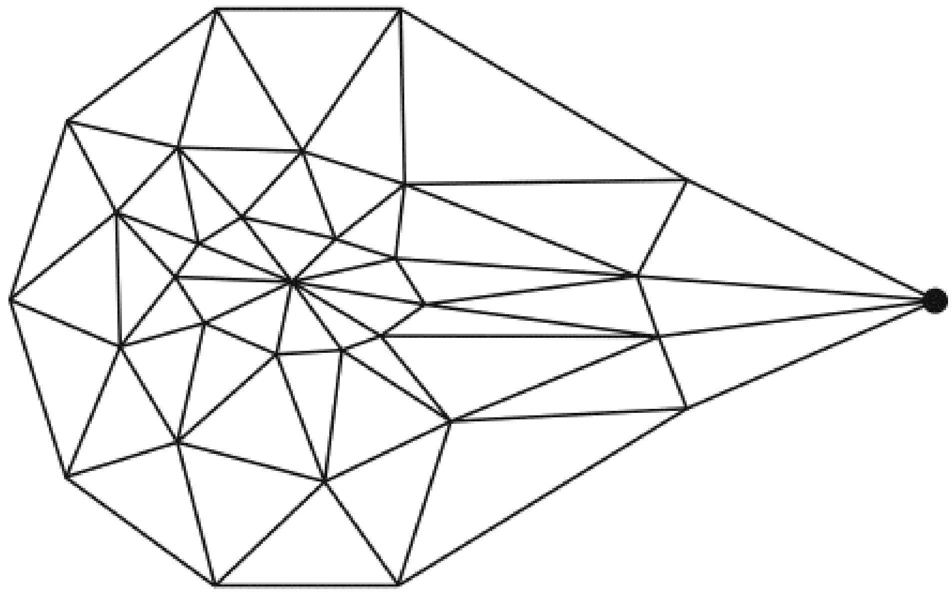
- Time consuming to define the trajectory through space of all vertices



- Instead, control a few *seed* vertices which in turn affect nearby vertices

Warping


Displacement of seed vertex


Attenuated displacement propagated to adjacent vertices

Vertex Displacement Function

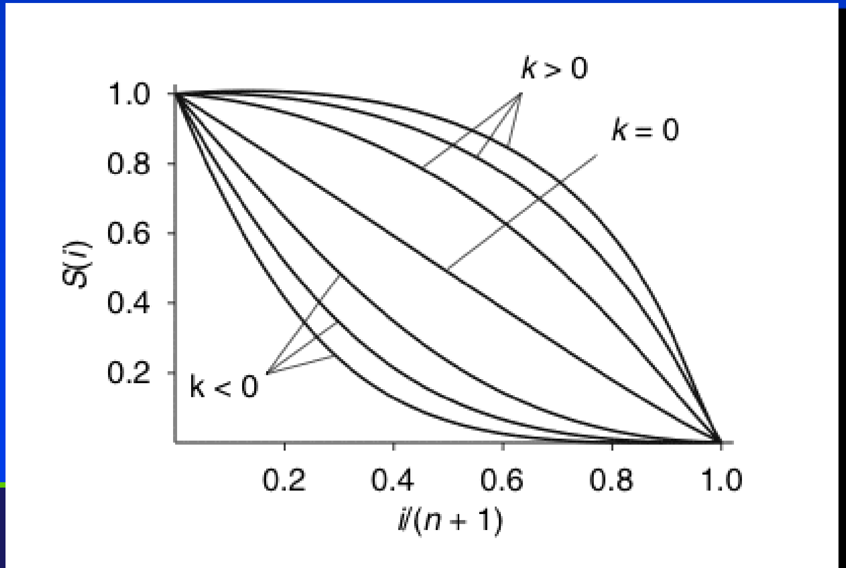
- i is the (shortest) number of edges between i and j
- n is the max number of edges affected
- (k=0) = linear; (k<0) = rigid; (k>0) = elastic

$$f(i) = 1.0 - \left(\frac{i}{n+1}\right)^{k+1} \quad ; k \geq 0$$

$$f(i) = \left(1.0 - \left(\frac{i}{n+1}\right)\right)^{-k+1} \quad ; k < 0$$

Warping effects by using power functions

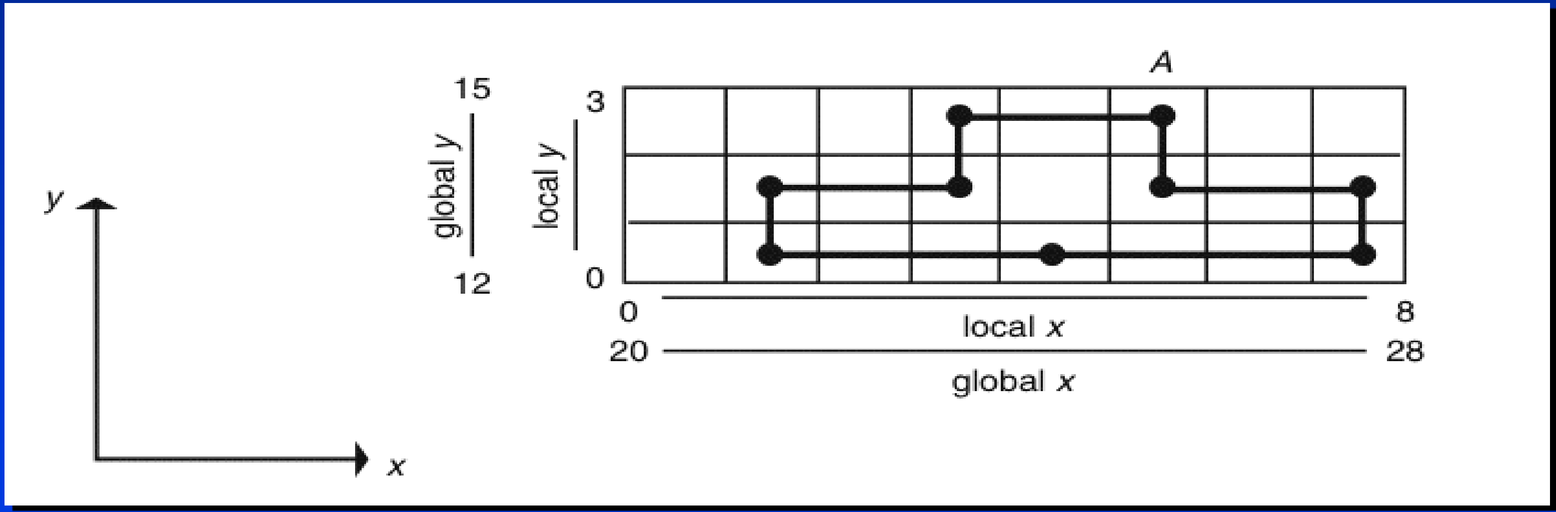
For attenuating warping effects



2-D Grid Deformation

- 1974 film “*Hunger*”
- Draw object on grid
- Deform grid points
- Use bilinear interpolation to re-compute vertex positions on deformed grid

2D Grid-based Deformation

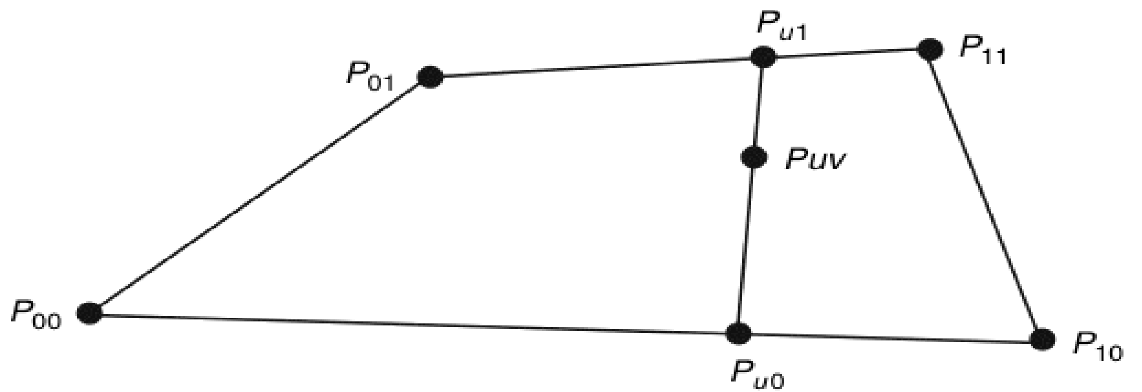


Assumption

Easier to deform grid points than object vertices

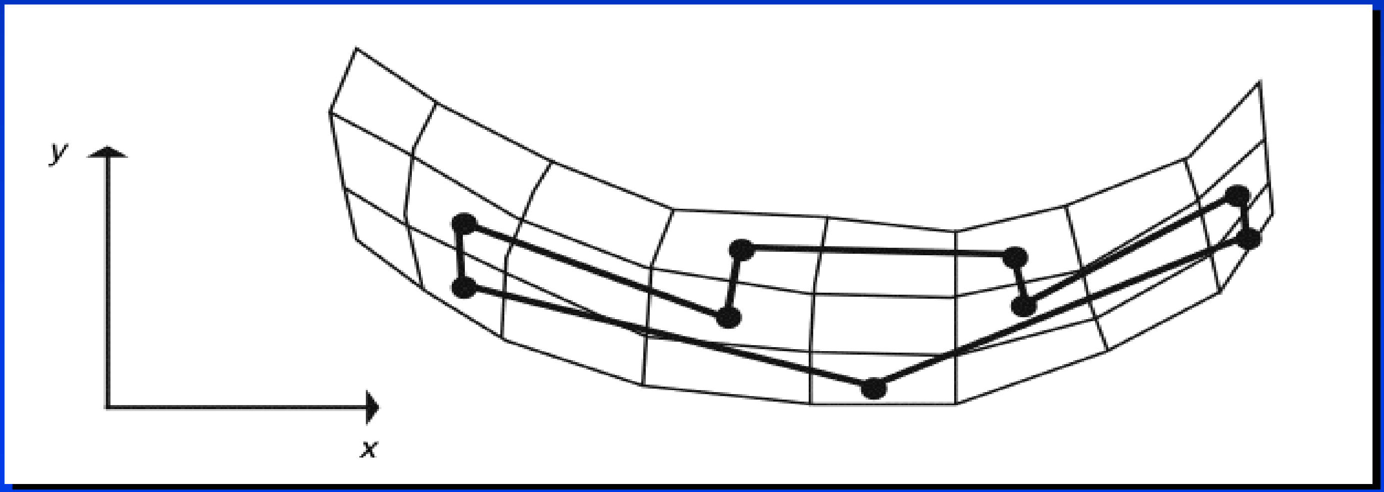
2D Grid-based Deformation

$$
\begin{aligned}
P_{u0} &= (1-u)P_{00} + uP_{10} \\
P_{u1} &= (1-u)P_{01} + uP_{11} \\
P_{uv} &= (1-v)P_{u0} + vP_{u1} \\
&= (1-u)(1-v)P_{00} + (1-v)uP_{01} + u(1-v)P_{10} + uvP_{11}
\end{aligned}
$$



Inverse bilinear mapping (determine u,v from points)

2D Grid-based Deformation



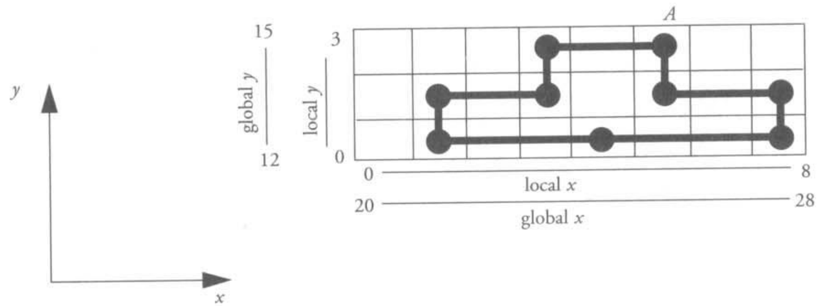


Figure 3.57 Initial 2D coordinate grid

$$
\begin{aligned}
Pu0 &= (1-u)\cdot P00 + u\cdot P10 \\
Pu1 &= (1-u)\cdot P01 + u\cdot P11 \\
Puv &= (1-v)\cdot Pu0 + v\cdot Pu1 \\
&= (1-u)\cdot(1-v)\cdot P00 + (1-u)\cdot v\cdot P01 + u\cdot(1-v)\cdot P10 + u\cdot v\cdot P11
\end{aligned}
$$

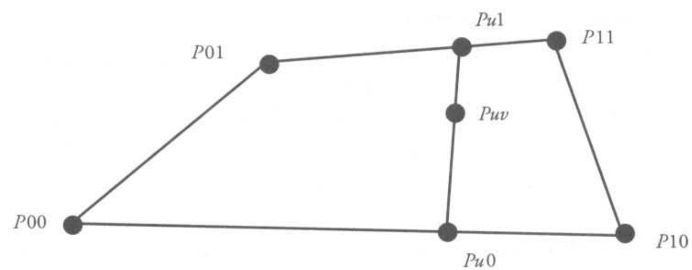


Figure 3.58 Bilinear interpolation

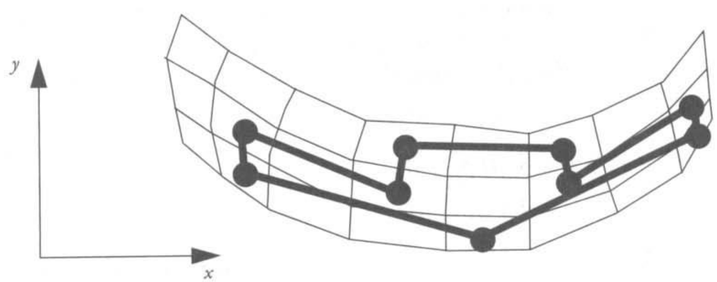
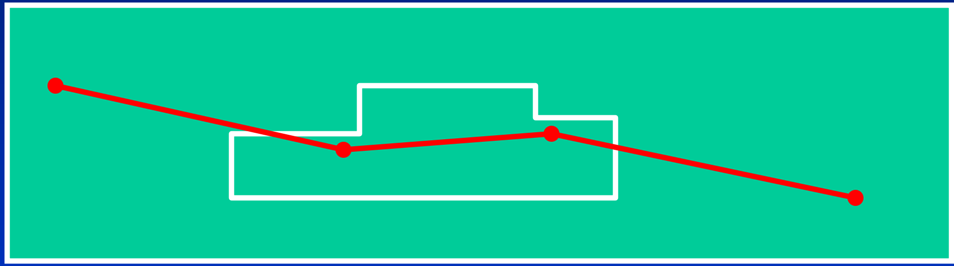


Figure 3.59 2D grid deformation

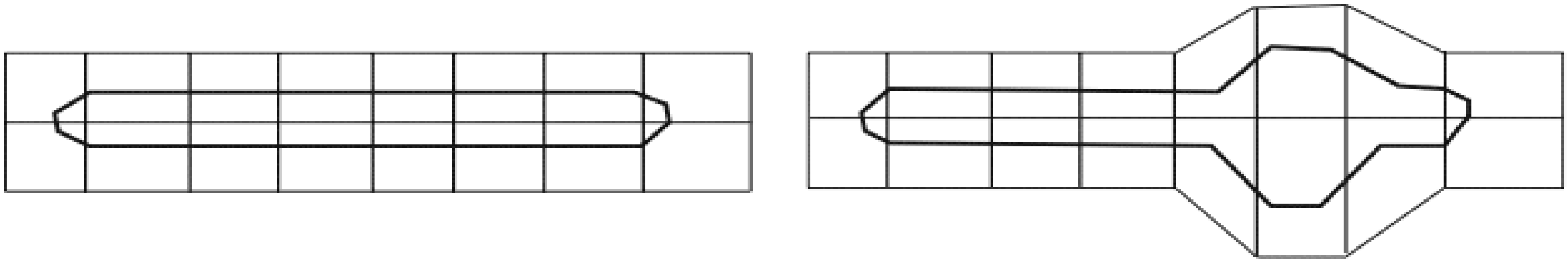
Polyline Deformation (Skeleton)

- Draw a piecewise linear line (polyline) passing through the geometry

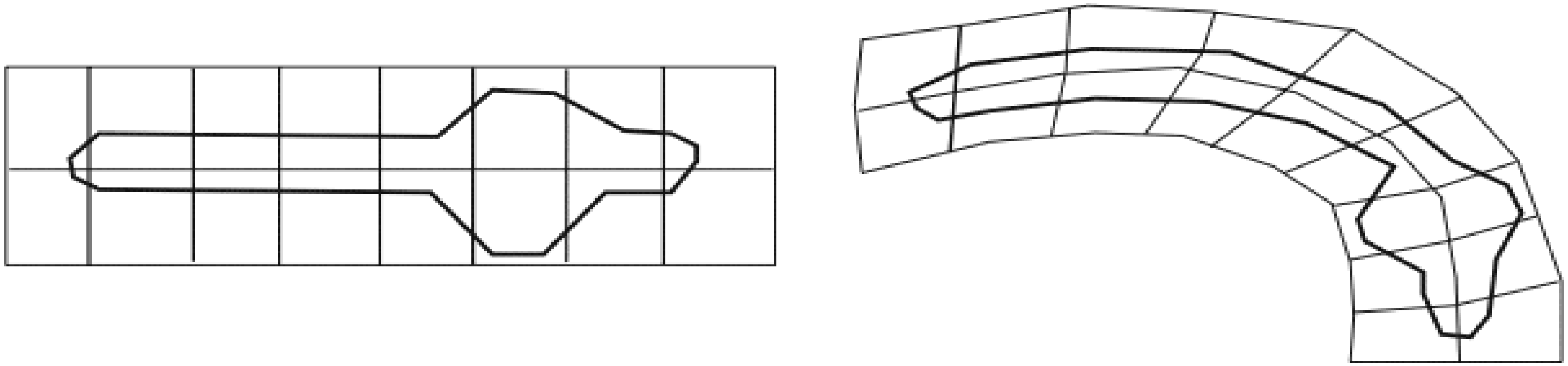


- For each vertex compute
  - Closest polyline segment
  - Distance to segment
  - Relative distance along this segment
- Deform polyline and re-compute vertex positions
- The earlier version of skeleton-based deformation

Bulging & Bending

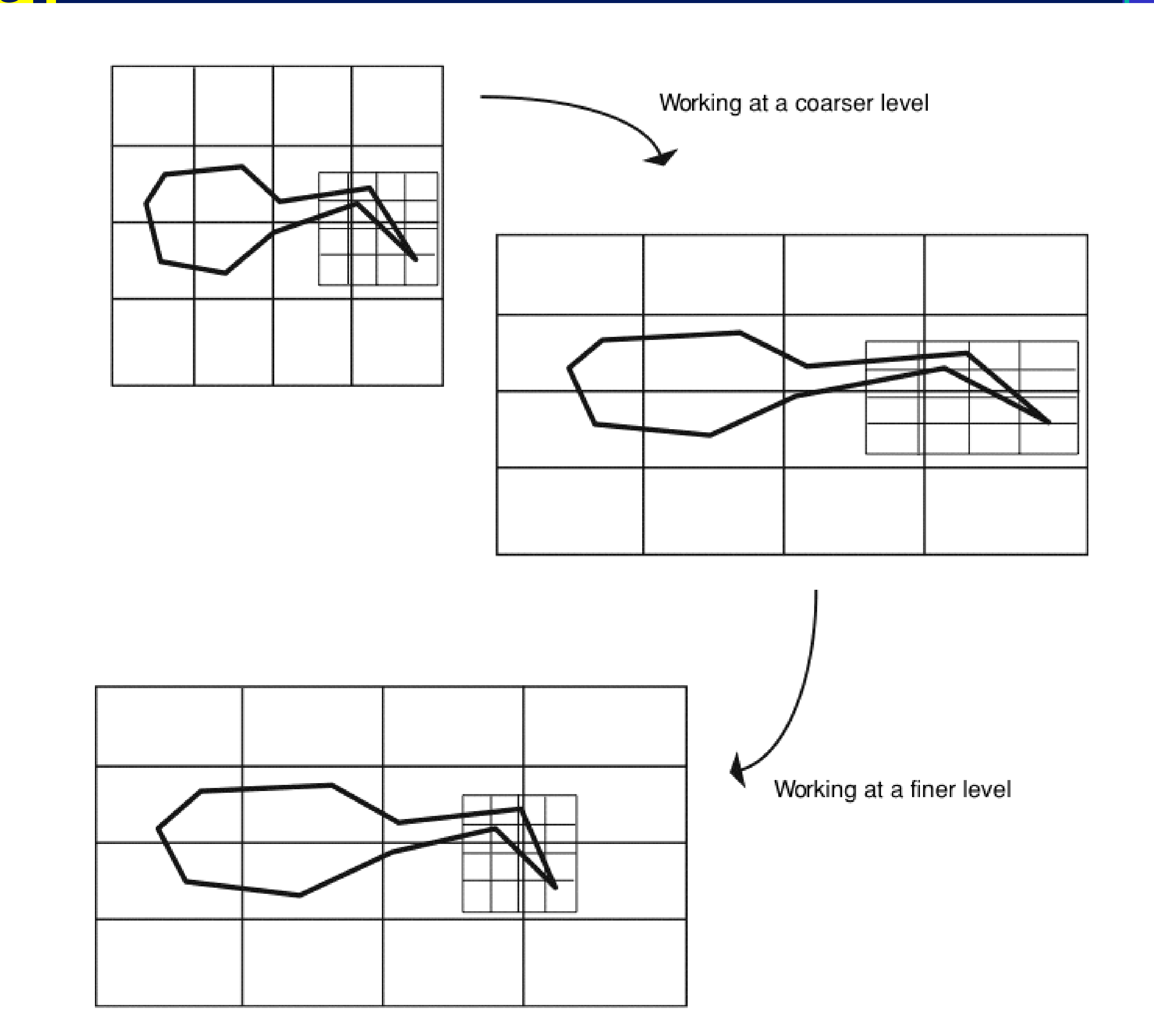


Bulging

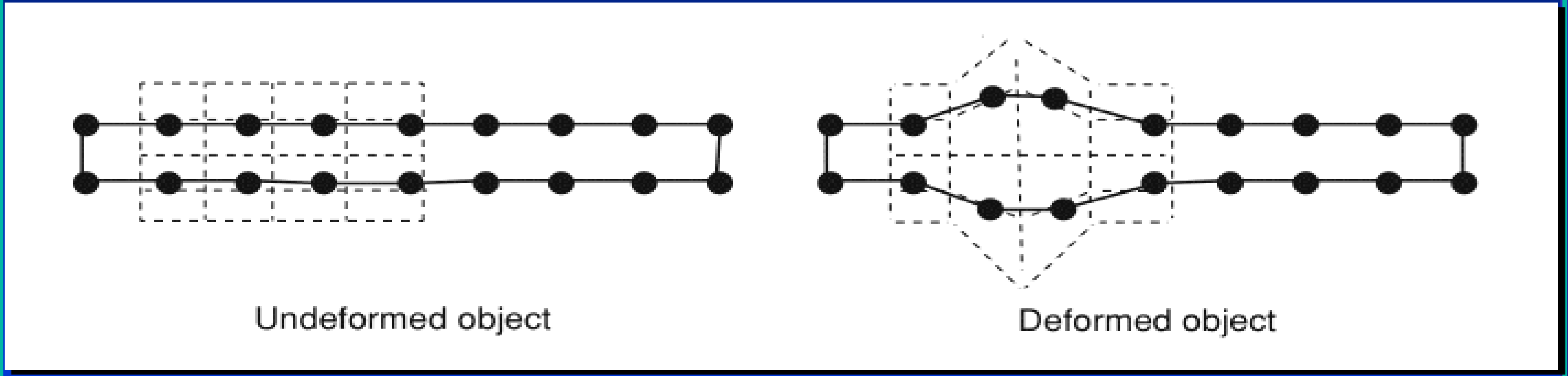


Bending

Hierarchical



FFDs – as tools to design shapes



Object Modification/Deformation

- **Modify the vertices directly**
  - **Vertex warping**
- **OR**
- **Modify the space the vertices lie in**
  - **2D grid-based deformation**
  - **Skeletal bending**
  - **Global transformations**
  - **Free-form deformations**

Global Deformations

- Alan Barr, SIGGRAPH ’84
- A 3x3 transformation matrix affects all vertices
  - P’=M(P) .dot. P
- M(P) can taper, twist, bend…

Global Transformations

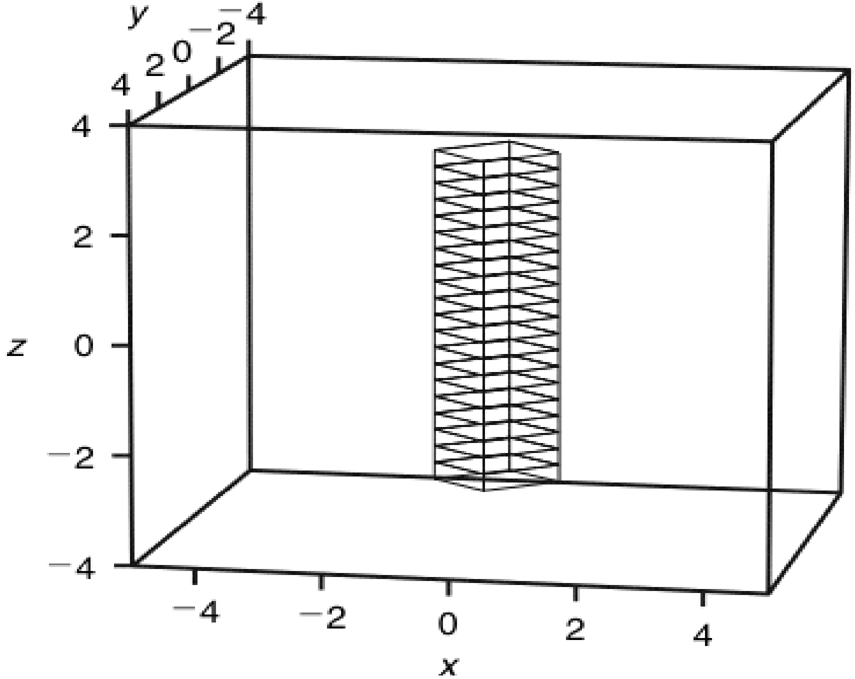
$$p' = Mp$$

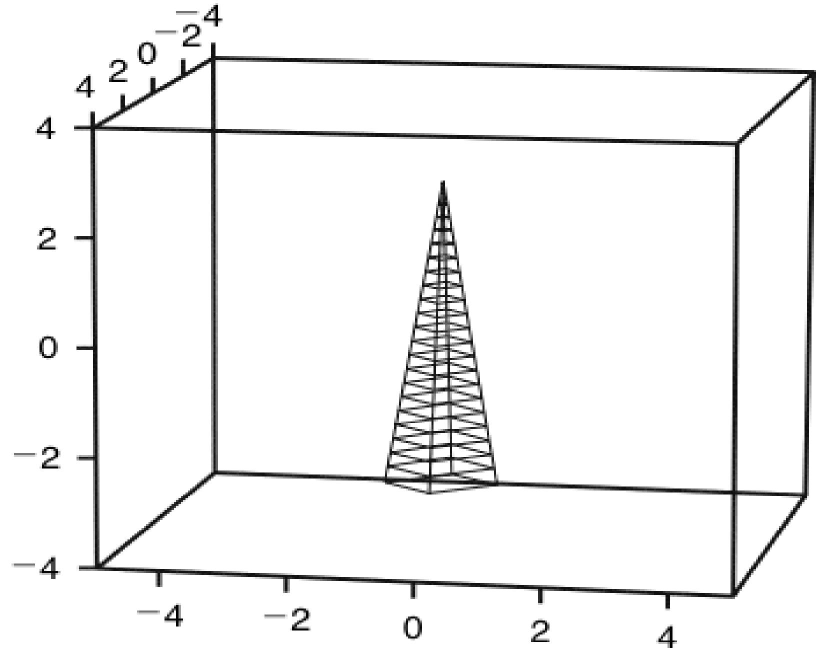
Commonly-used linear transformation of space

$$p' = M(p)\,p$$

In Global Transformations, Transform is a function of where you are in space

Global Transformations





$$s(z) = \frac{(maxz - z)}{(maxz - minz)}$$

$$x' = s(z)x$$
$$y' = s(z)y$$
$$z' = z$$

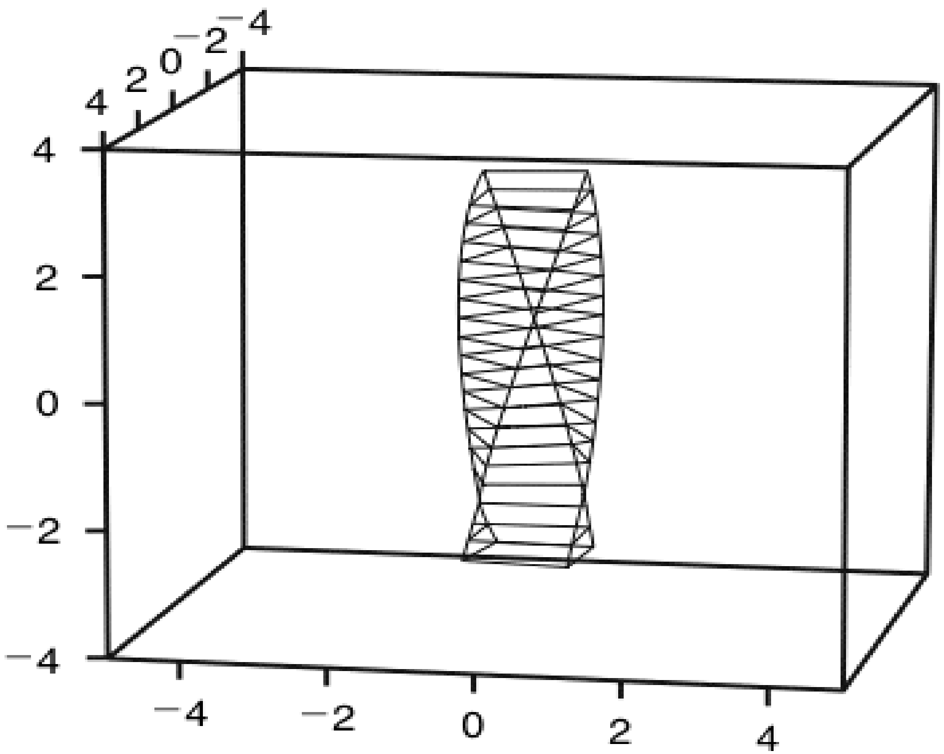
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s(z) & 0 & 0 \\ 0 & s(z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$P' = M(p)p$$

Original object

Tapered object

Global Transformations



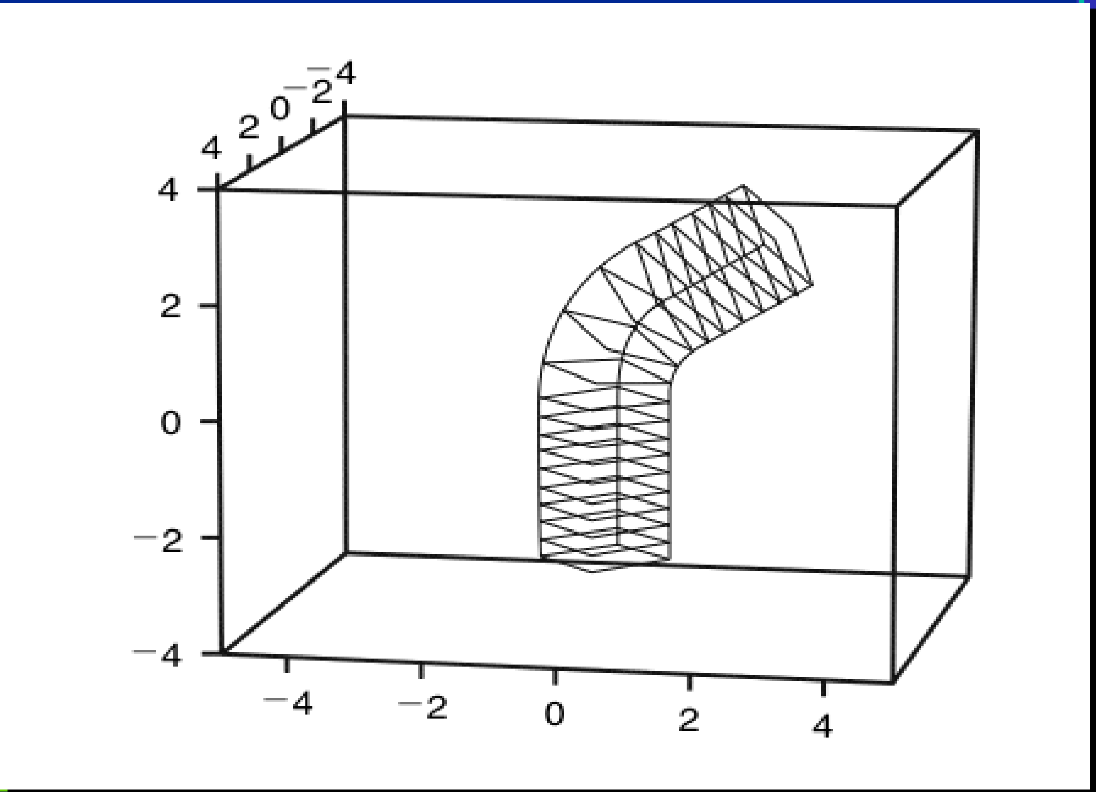
$$
\begin{aligned}
k &= \text{twist factor} \\
x' &= x\cos(kz) - y\sin(kz) \\
y' &= x\sin(kz) + y\cos(kz) \\
z' &= z
\end{aligned}
$$

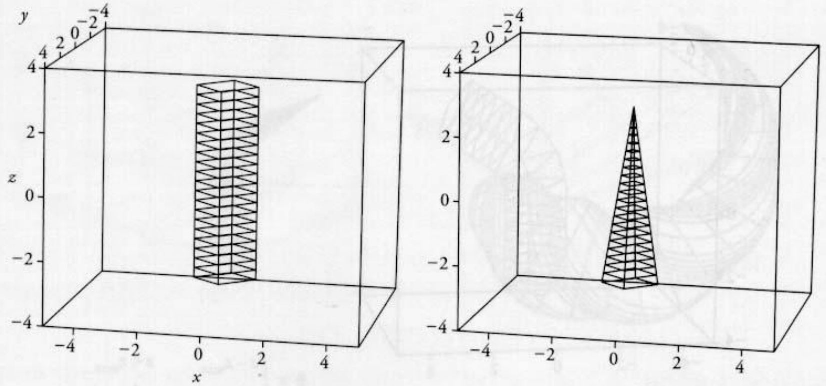
Global Transformations

z above z_{min}: rotate

z between z_{min} ,z_{max} :
Rotate from 0 to

z below z_{min}: no rotation





$$s(z) = \frac{(maxz - z)}{(maxz - minz)}$$

$$x' = s(z) \cdot x$$
$$y' = s(z) \cdot y$$
$$z' = z$$

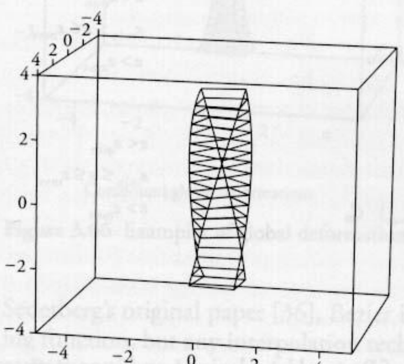
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s(z) & 0 & 0 \\ 0 & s(z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$P' = M(P) \cdot P$$

Original object

Tapered object

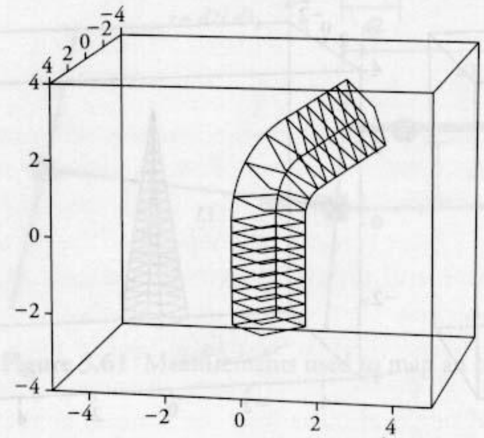
Figure 3.63 Global tapering



k = twist factor

$$x' = x \cdot \cos(k \cdot z) - y \cdot \sin(k \cdot z)$$
$$y' = x \cdot \sin(k \cdot z) + y \cdot \cos(k \cdot z)$$
$$z' = z$$

Figure 3.64 Twist about an axis



$$\theta = \begin{cases} z - z_{min} & z < z_{max} \\ z_{max} - z_{min} & \text{otherwise} \end{cases}$$

$$C_\theta = \cos\theta$$
$$S_\theta = \sin\theta$$
$$R = y_0 - y$$

$(z_{min} : z_{max})$—bend region

(y_0, z_{min})—center of bend

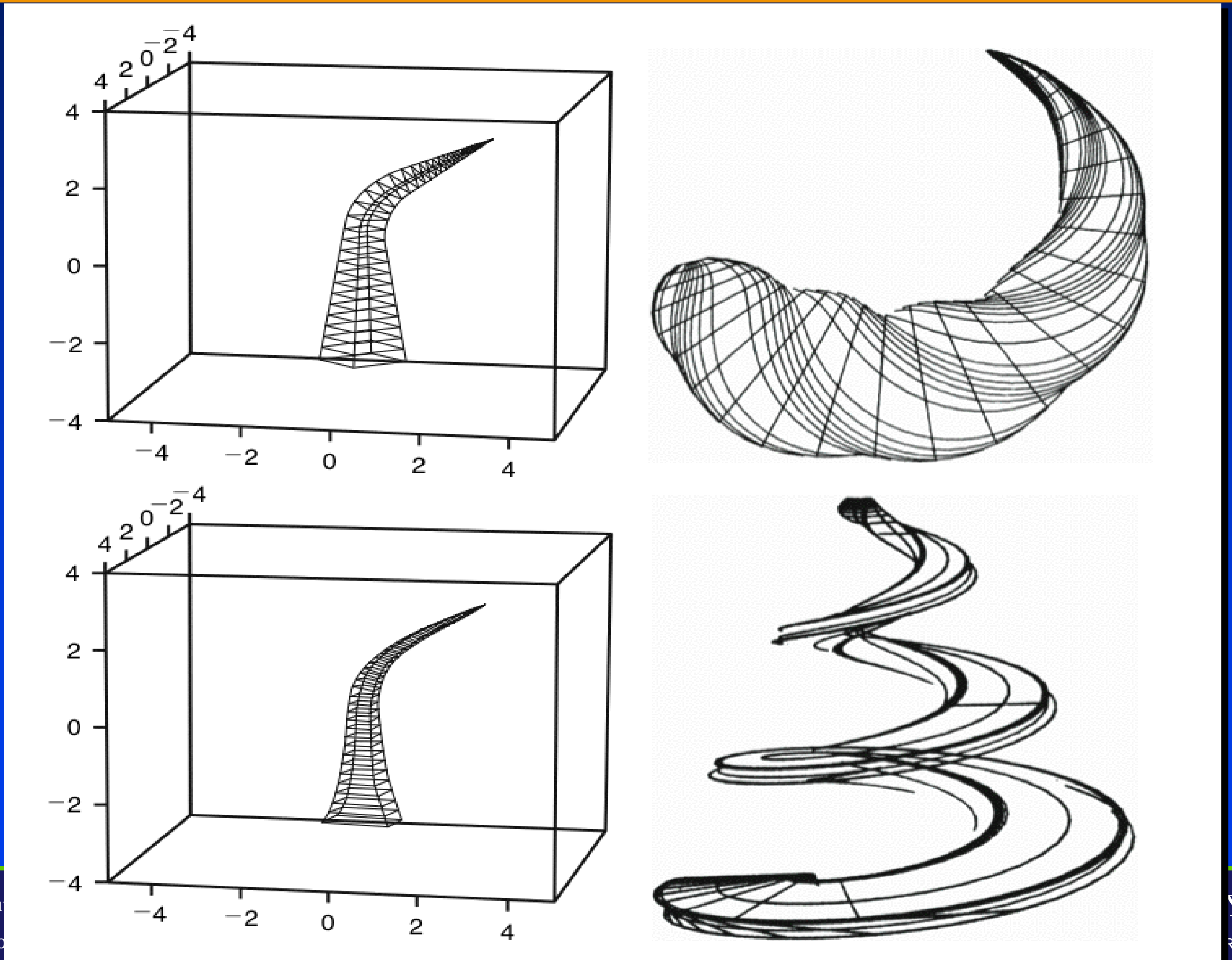
$$x' = x$$

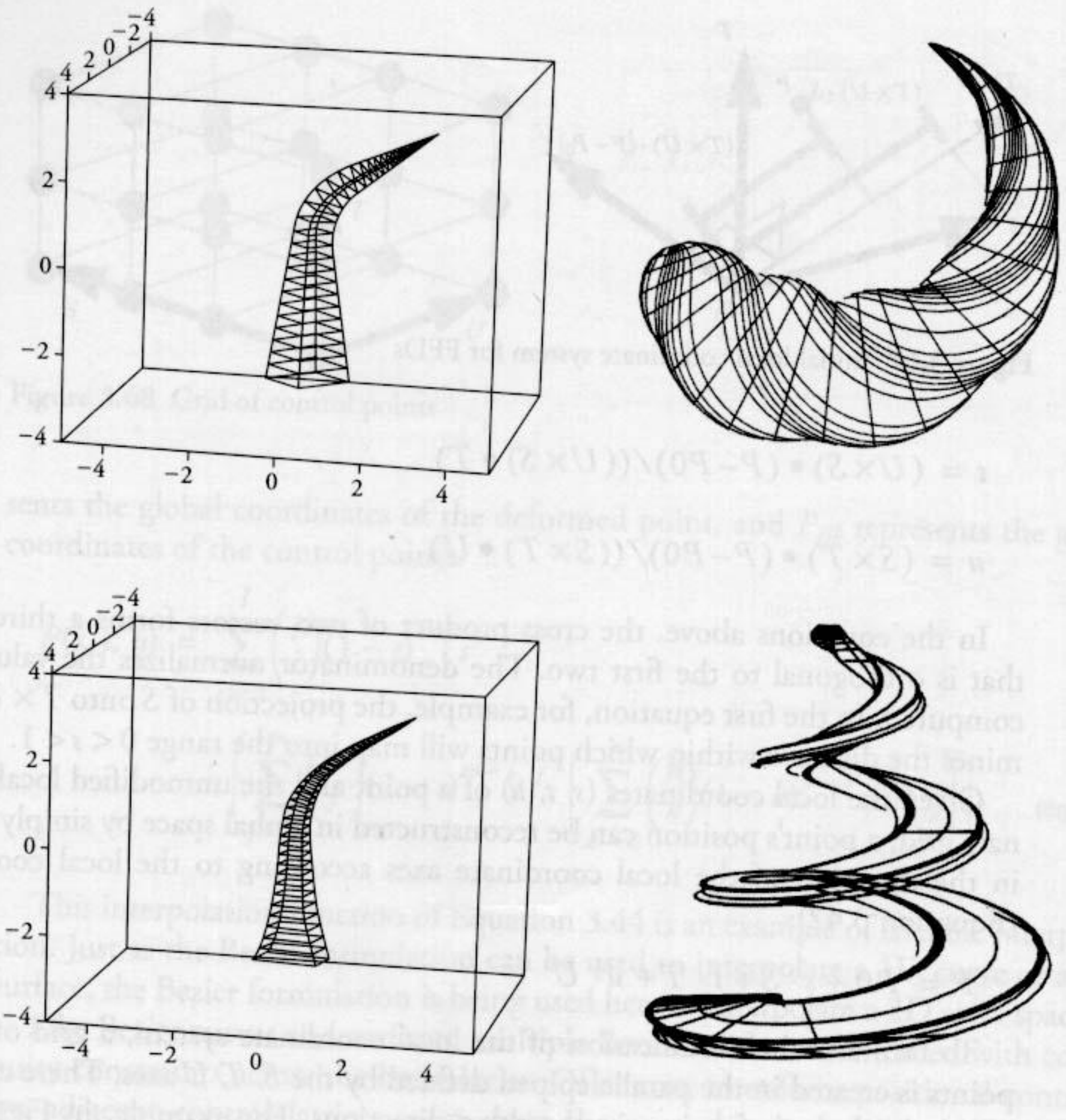
$$y' = \begin{cases} y & z < z_{min} \\ y_0 - (R \cdot C_\theta) & z_{min} \le z \le z_{max} \\ y_0 - (R \cdot C_\theta) + (z - z_{max}) \cdot S_\theta & z > z_{max} \end{cases}$$

$$z' = \begin{cases} z & z < z_{min} \\ z_{min} + (R \cdot S_\theta) & z_{min} \le z \le z_{max} \\ z_{min} + (R \cdot S_\theta) + (z - z_{max}) \cdot C_\theta & z > z_{max} \end{cases}$$

Figure 3.65 Global bend operation

Compound Global Transformations





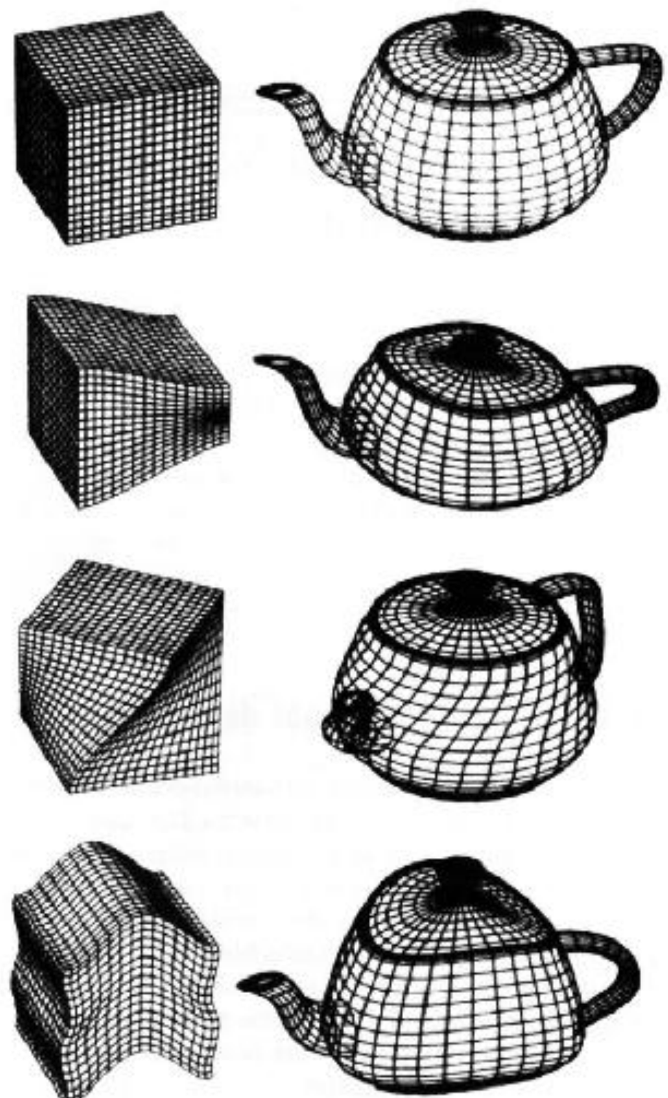
Compound global deformations

Examples from Barr [2]

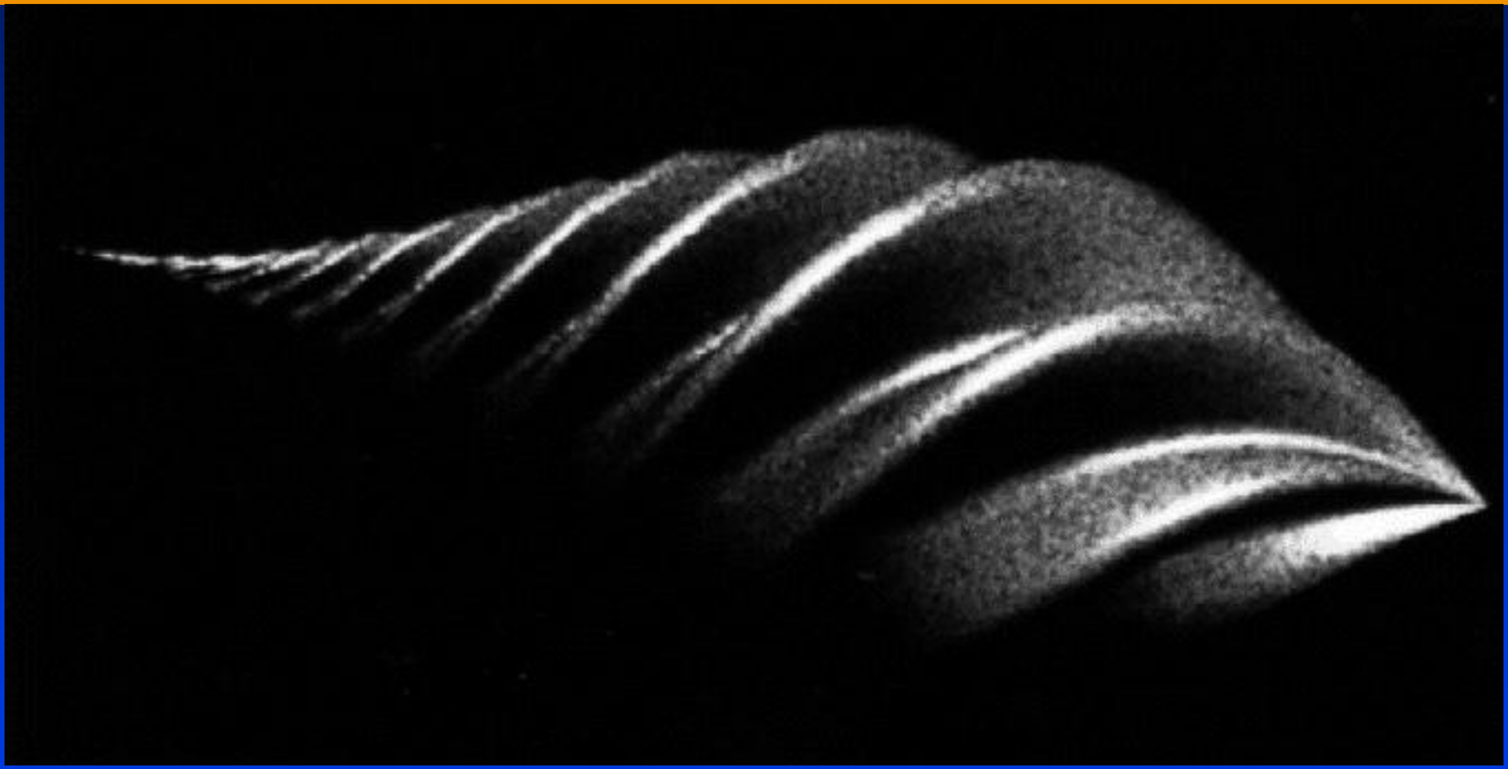
Figure 3.66 Examples of global deformations

Nonlinear Global Deformation

- original
- tapering
- twisting
- bending



Nonlinear Global Deformation



Good for modeling [Barr 87]

Animation is harder

Space Warping

- Deformation the object by deforming the space it is residing in
- Two main techniques:
- Nonlinear deformation
- Free Form Deformation (FFD)

Independent of object representation

Nonlinear Global Deformation

- Objects are defined in a local object space
- Deform this space by using a combination of:
- Non-uniform scaling
- Tapering
- Twisting
- Bending

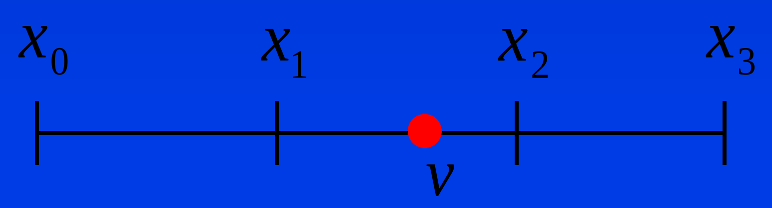
What is “Free-Form”?

- Parametric surfaces are free-form surfaces.
- The flexibility in this technique of deformation allows us deform the model in a free-form manner.
  - ✓ Any surface patches
  - ✓ Global or local deformation
  - ✓ Continuity in local deformation
  - ✓ Volume preservation

Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices

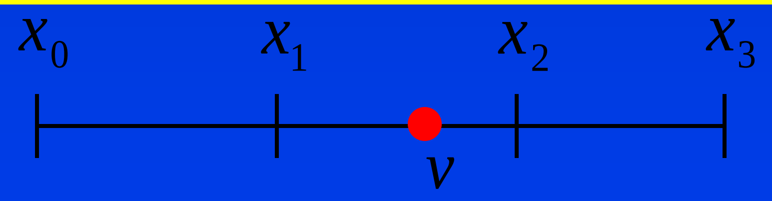
$$v = \sum_i w_i x_i$$



Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices
- Assume x_i are equally spaced and use Bernstein basis functions

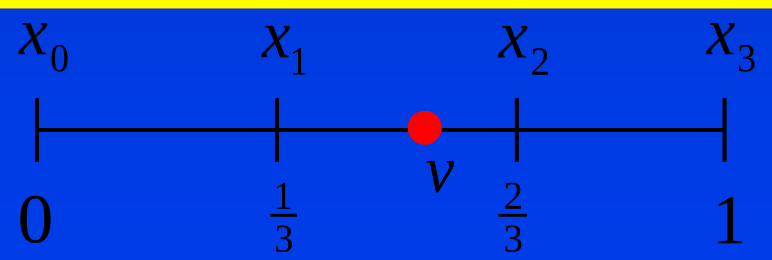
$$v = \sum_i w_i x_i = \sum_i \binom{d}{i} (1-t)^{d-i} t^i x_i$$



Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices
- Assume x_i are equally spaced and use Bernstein basis functions

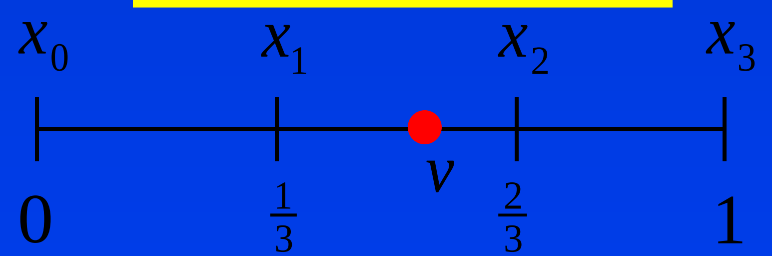
$$v = \sum_i w_i x_i = \sum_i \binom{d}{i} (1-t)^{d-i} t^i x_i = t$$



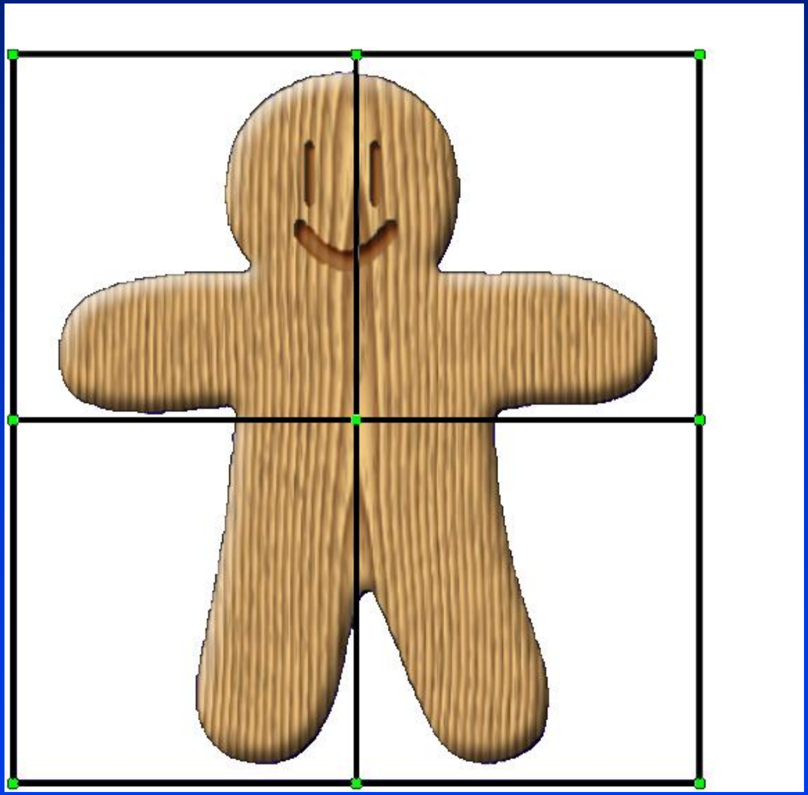
Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices
- Assume x_i are equally spaced and use Bernstein basis functions

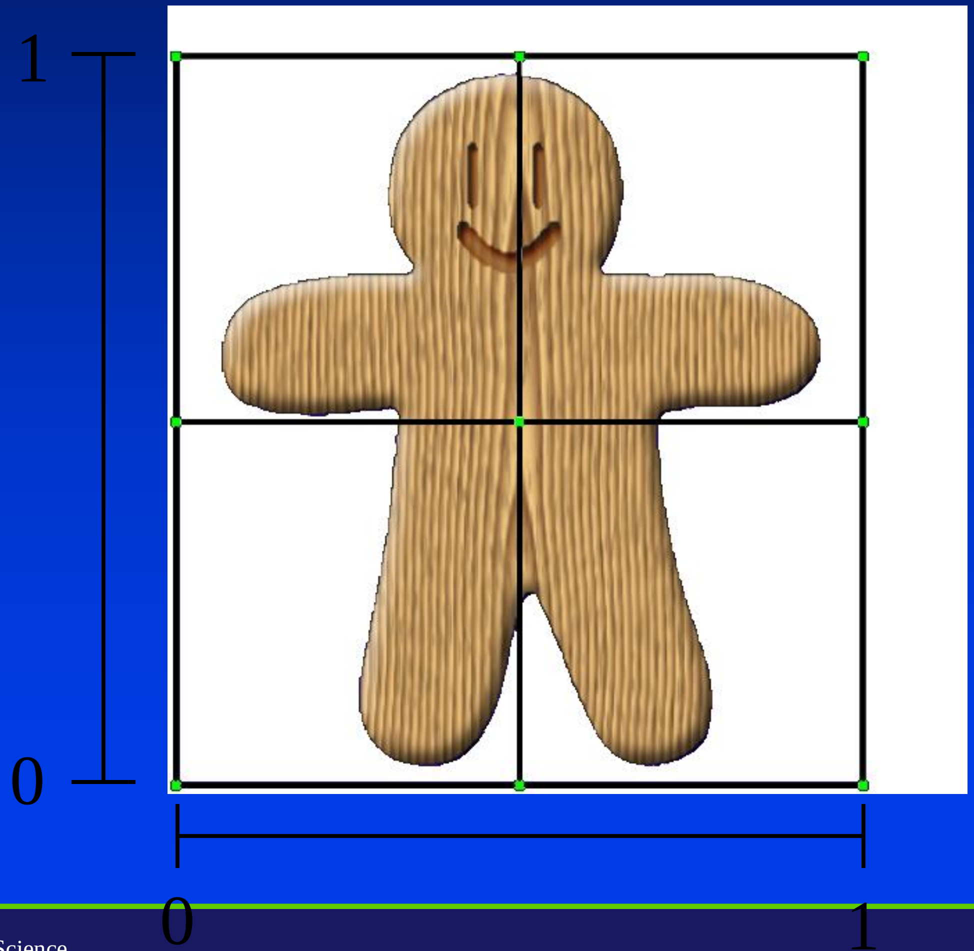
$$w_i = \binom{d}{i}(1-v)^{d-i}v^i$$



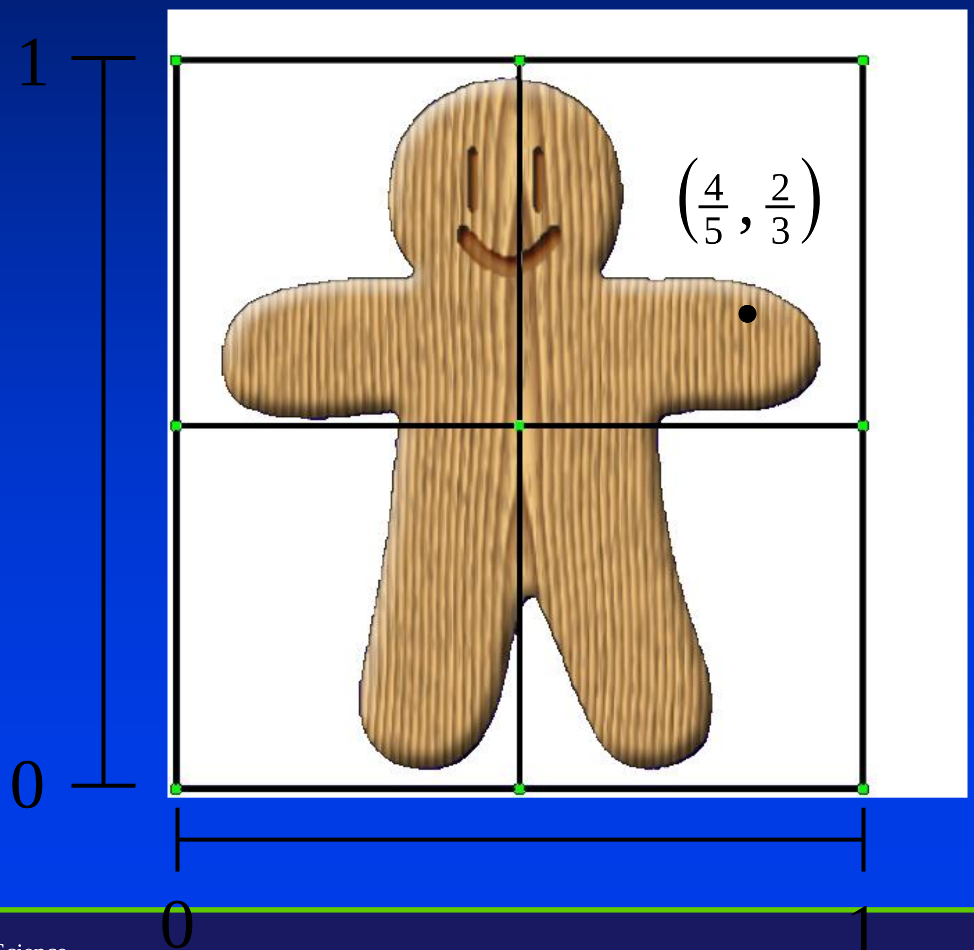
2D Example



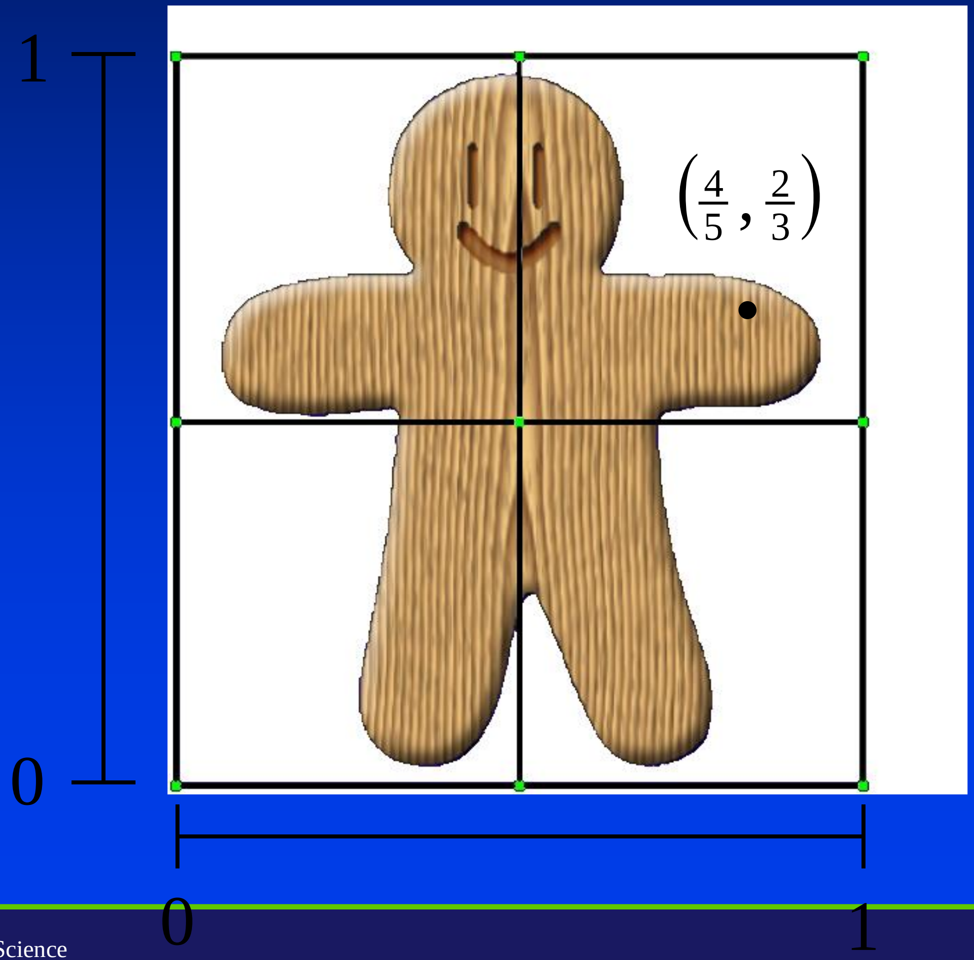
2D Example



2D Example



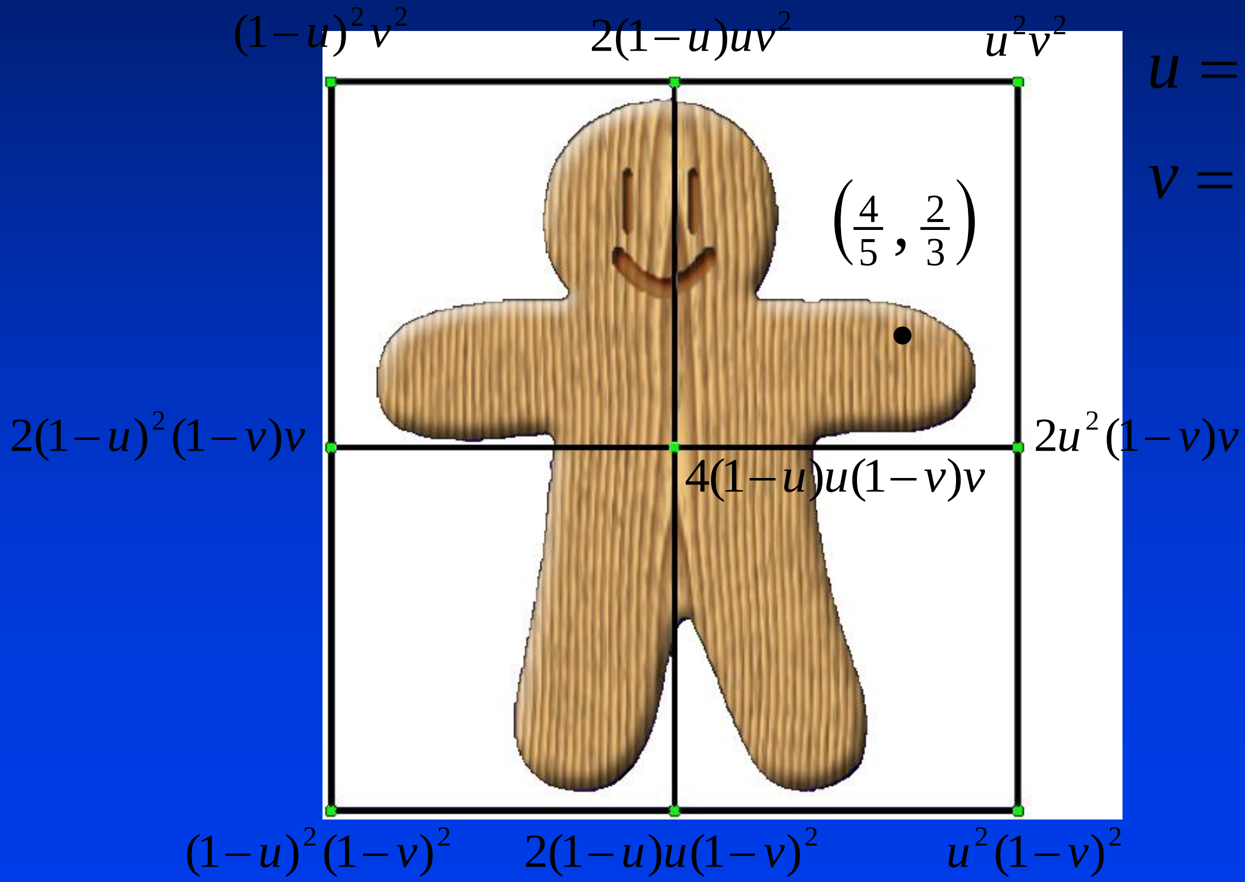
2D Example



$u = \frac{4}{5}$

$v = \frac{2}{3}$

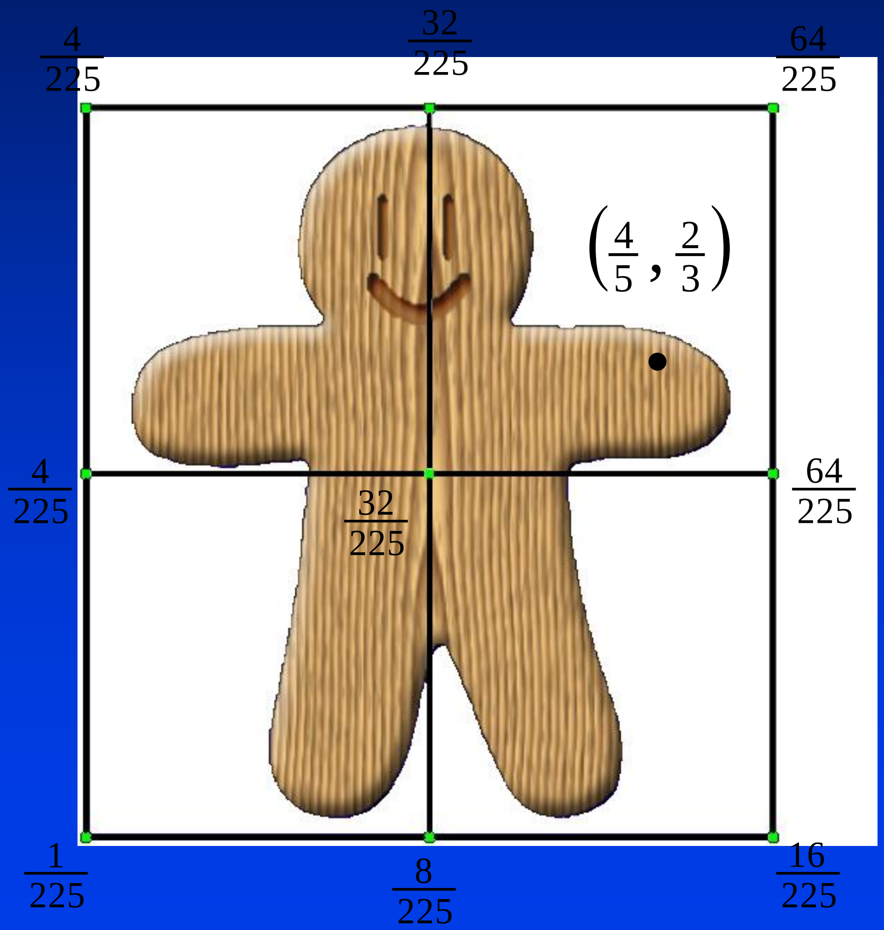
2D Example



$u = \frac{4}{5}$

$v = \frac{2}{3}$

2D Example

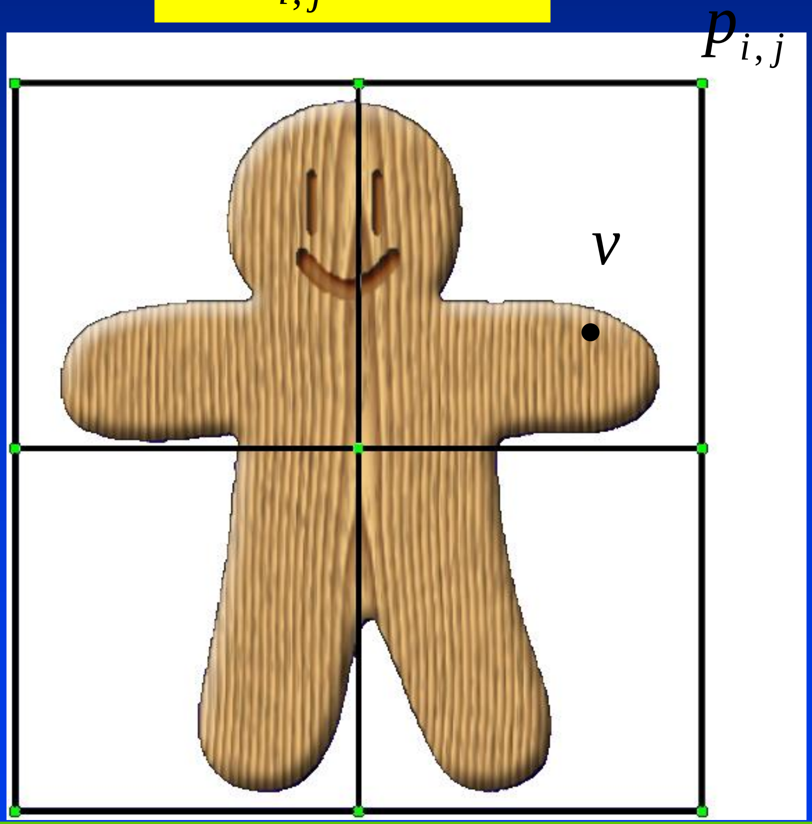


$$u = \frac{4}{5}$$

$$v = \frac{2}{3}$$

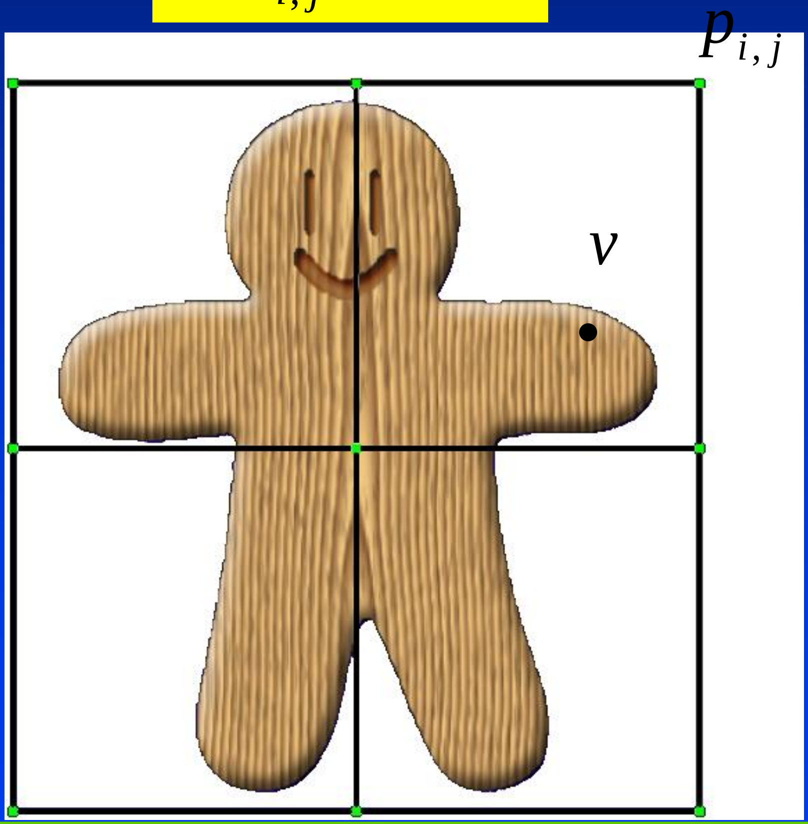
Applying the Deformation

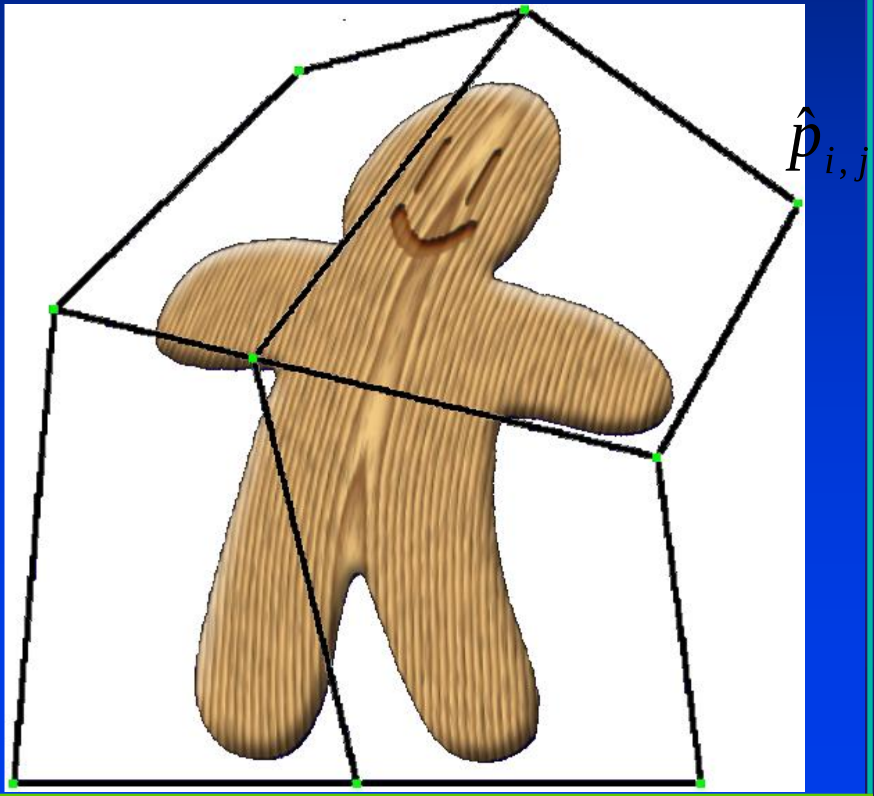
$$v = \sum_{i,j} w_{i,j}\, p_{i,j}$$



Applying the Deformation

$$v = \sum_{i,j} w_{i,j} p_{i,j}$$

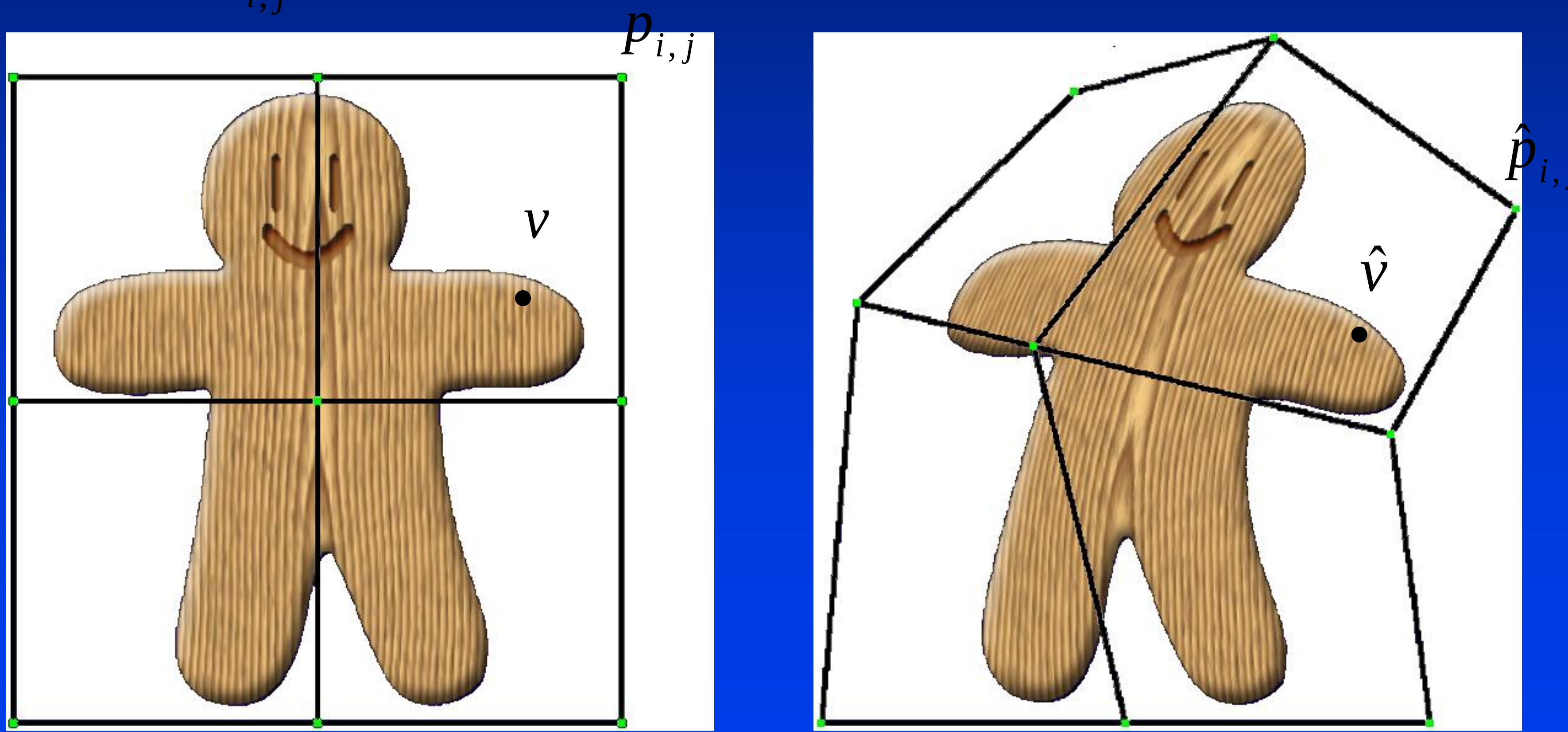




Applying the Deformation

$$v = \sum_{i,j} w_{i,j} p_{i,j}$$

$$\hat{v} = \sum_{i,j} w_{i,j} \hat{p}_{i,j}$$



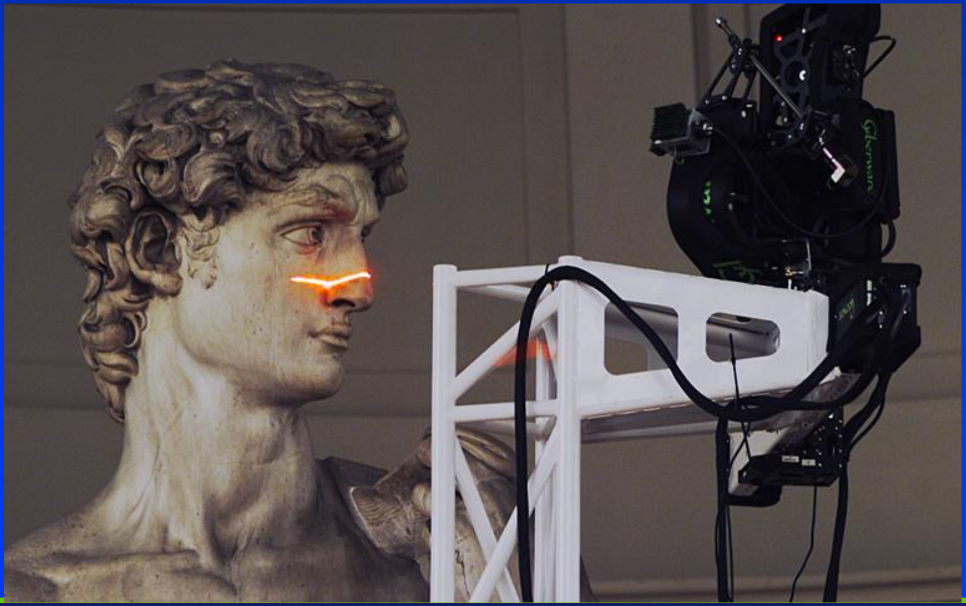
FFD Contributions

- Smooth deformations of arbitrary shapes
- Local control of deformation
- Performing deformation is fast

- Widely used
  - Game/Movie industry
  - Part of nearly every 3D modeling package

Challenges in Deformation

- Large meshes – millions of polygons
- Need efficient techniques for computing and specifying the deformation

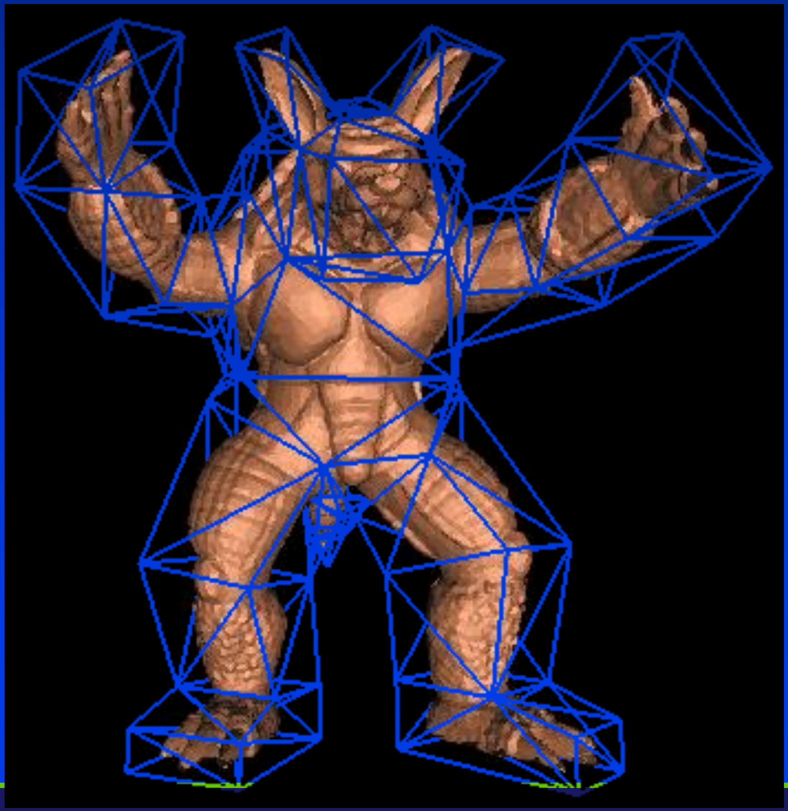


Deformation Handles

- Low-resolution auxiliary shape controls deformation of high-resolution model

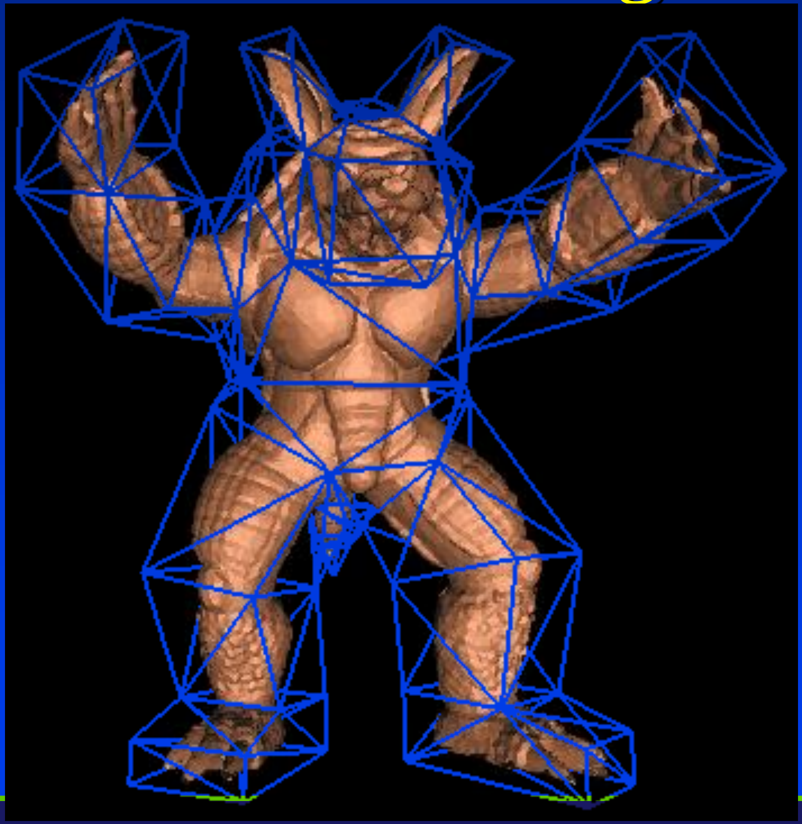
Deformation Handles

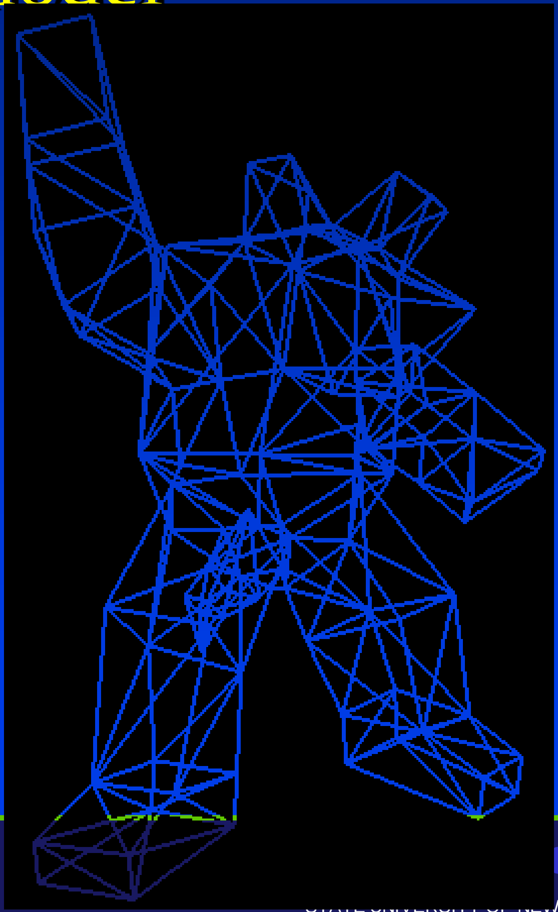
- Low-resolution auxiliary shape controls deformation of high-resolution model



Deformation Handles

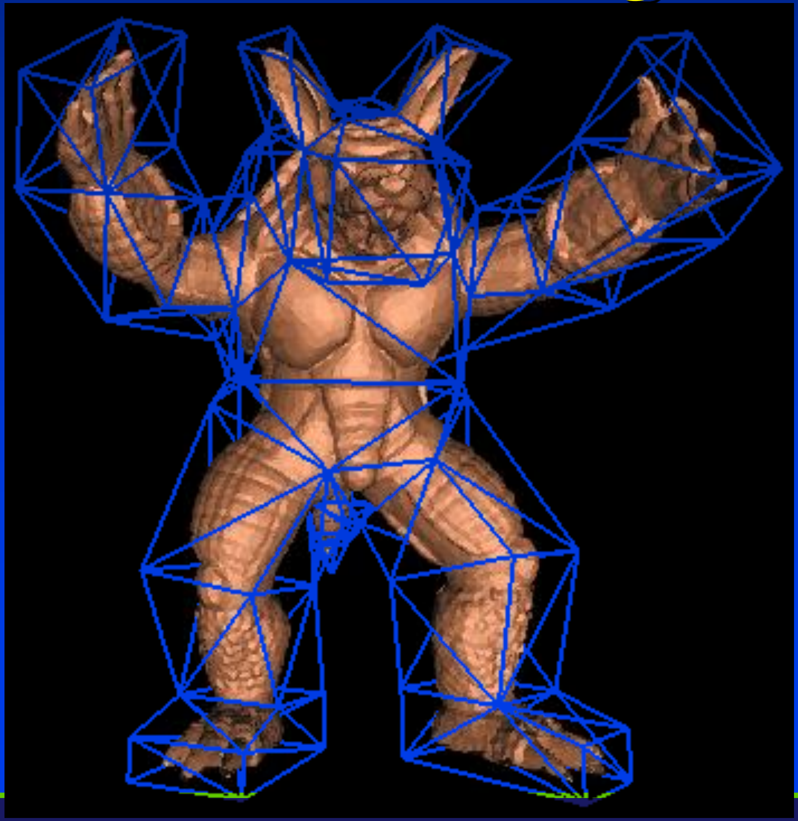
- Low-resolution auxiliary shape controls deformation of high-resolution model

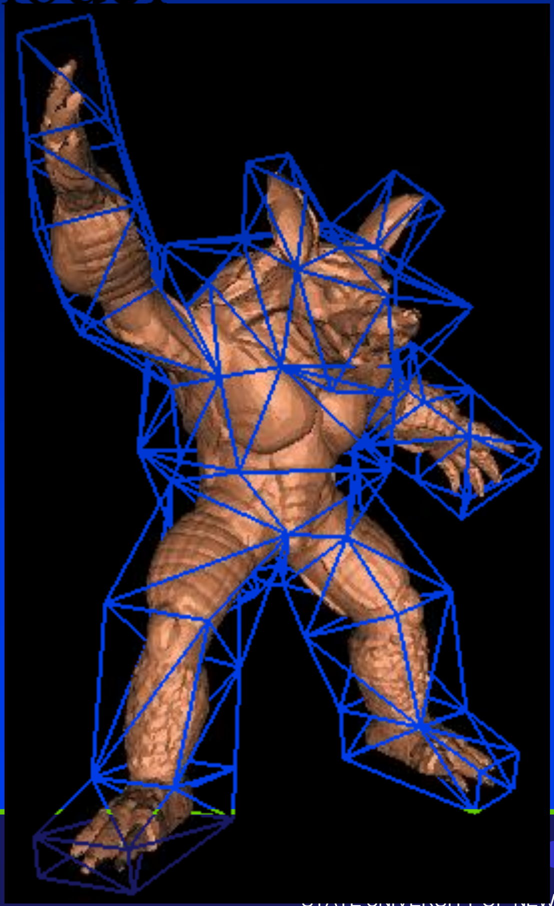




Deformation Handles

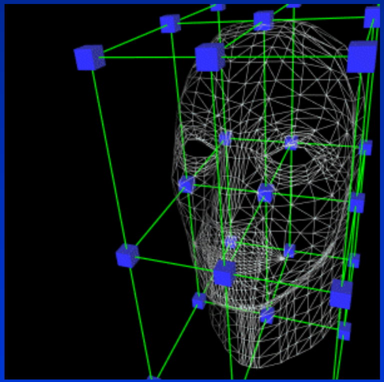
- Low-resolution auxiliary shape controls deformation of high-resolution model

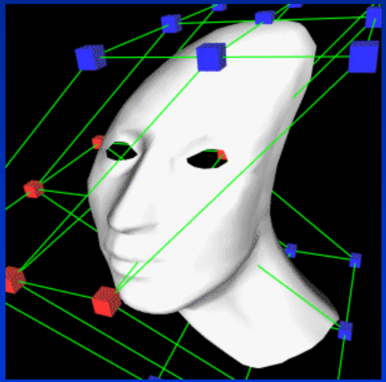




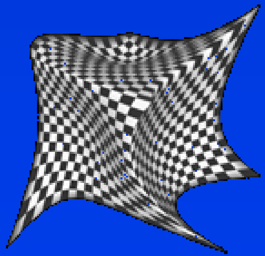
Free-Form Deformation (FFD)

- Sederberg, SIGGRAPH '86
- Place geometric object inside local coordinate space
- Build local coordinate representation



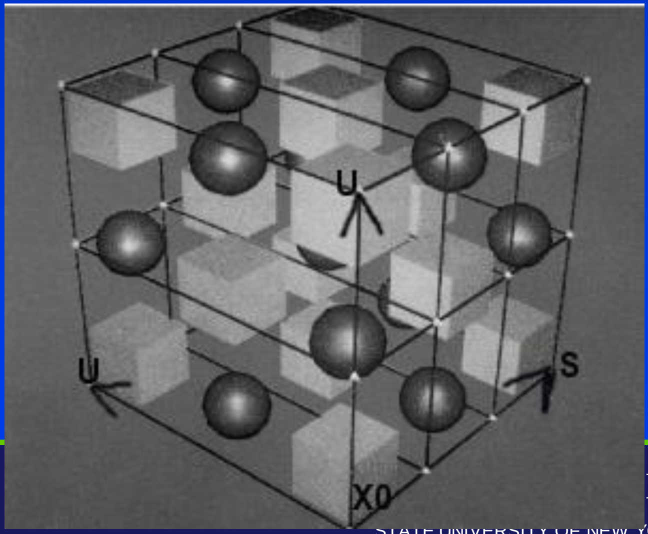


- Deform local coordinate space and thus deform geometry



Free-Form Deformation (FFD)

- Basic idea: deform space by deforming a lattice around an object
- The deformation is defined by moving the control points of the lattice
- Imagine it as if the object were enclosed by rubber
- The key is how to define
  - Local coordinate system
  - The mapping



Free-Form Deformation

- Similar to 2-D grid deformation
- Define 3-D lattice surrounding geometry
- Move grid points of lattice and deform geometry accordingly
- Local coordinate system is initially defined by three (perhaps non orthogonal) vectors

Trilinear Interpolation

- Let *S*, *T*, and *U* (with origin P_0 define local coordinate axes of bounding box that encloses geometry
- A vertex, P's, coordinates are:

$$s = (T \times U) \cdot \frac{P - P_0}{(T \times U) \cdot S}$$

$$t = (U \times S) \cdot \frac{P - P_0}{(U \times S) \cdot T}$$

$$u = (S \times T) \cdot \frac{P - P_0}{(S \times T) \cdot U}$$

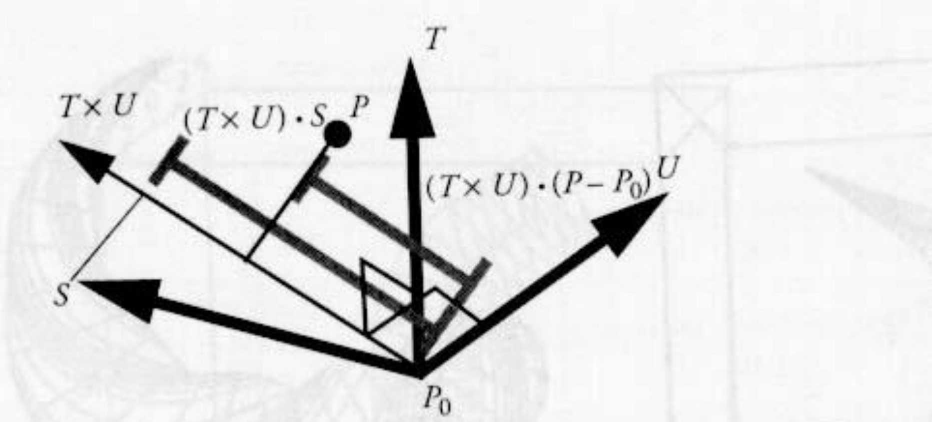


Figure 3.67 Initial local coordinate system for FFDs

Volumetric Control Points

- Each of S, T, and U axes are subdivided by control points
- A lattice of control points is constructed

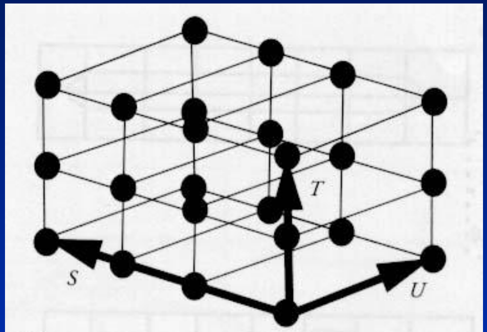


Figure 3.68 Grid of control points

- Bezier interpolation of move control points define new vertex positions

$$P = P_0 + s \cdot S + t \cdot T + u \cdot U$$

$$P_{ijk} = P_0 + \frac{i}{l} \cdot S + \frac{j}{m} \cdot T + \frac{k}{n} \cdot U$$

$$P(s,t,u) = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^i \cdot \left(\sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^j \cdot \left(\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^k P_{ijk} \right) \right)$$

Free-Form Deformation (FFD)

The lattice defines a Bezier volume

$$\mathbf{Q}(u,v,w) = \sum_{ijk} \mathbf{p}_{ijk} B(u) B(v) B(w)$$

Compute lattice coordinates

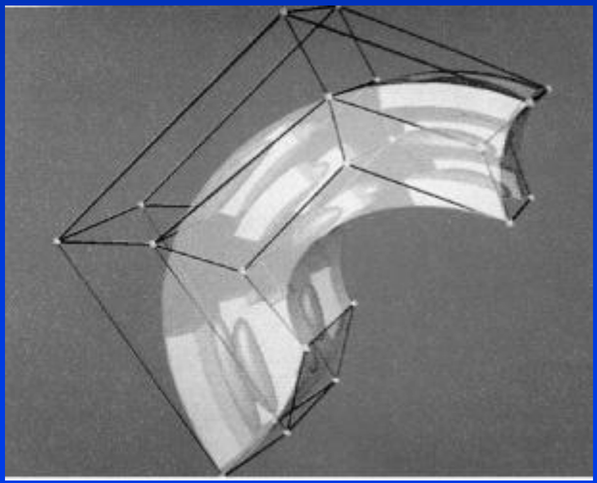
$$(u,v,w)$$

Move the control points

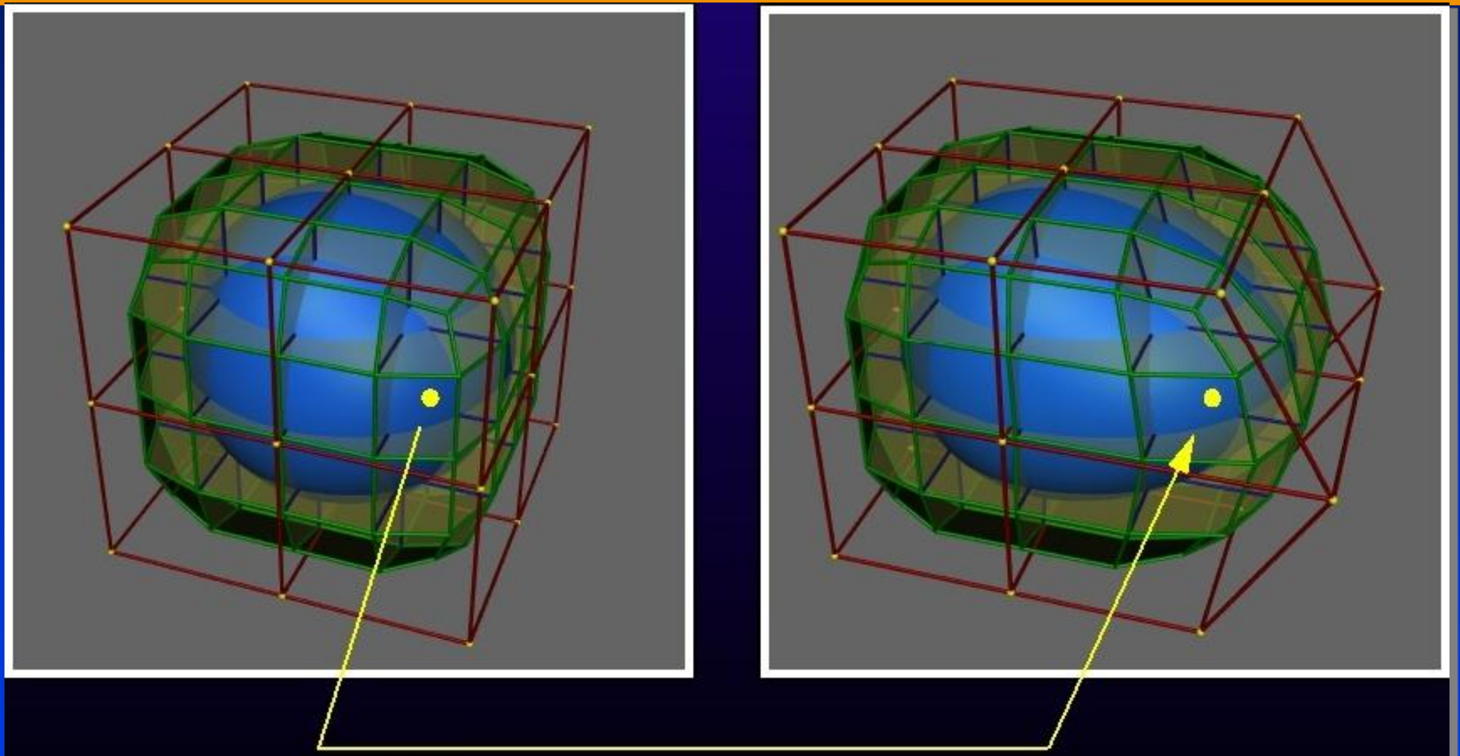
$$\mathbf{p}_{ijk}$$

Compute the deformed points

$$\mathbf{Q}(u,v,w)$$

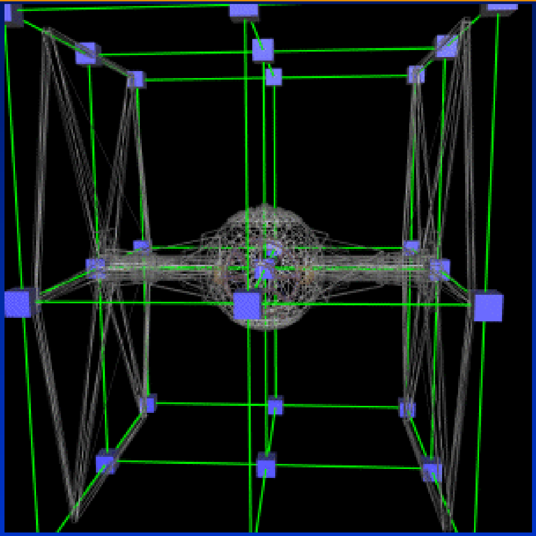


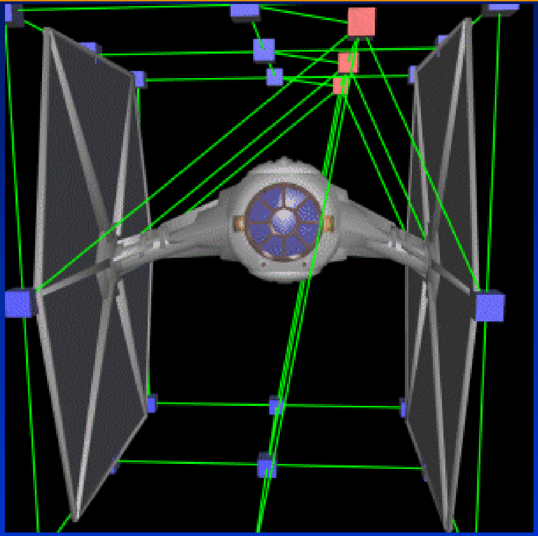
The FFD Process - Example

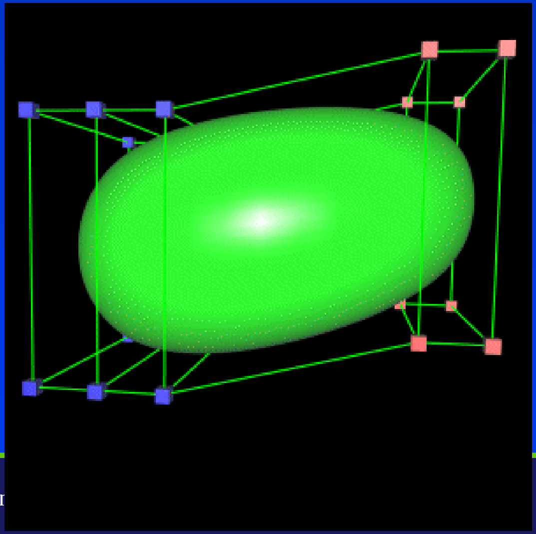


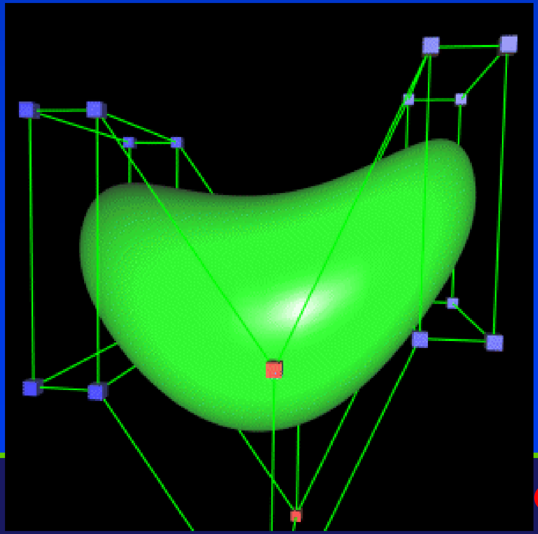
Point in a cell is repositioned within the corresponding cell in the deformed lattice, in the same relative position within the cell.

Examples









Smoothness of Deformation

- Constraining Bezier control points controls smoothness

C^{-1}

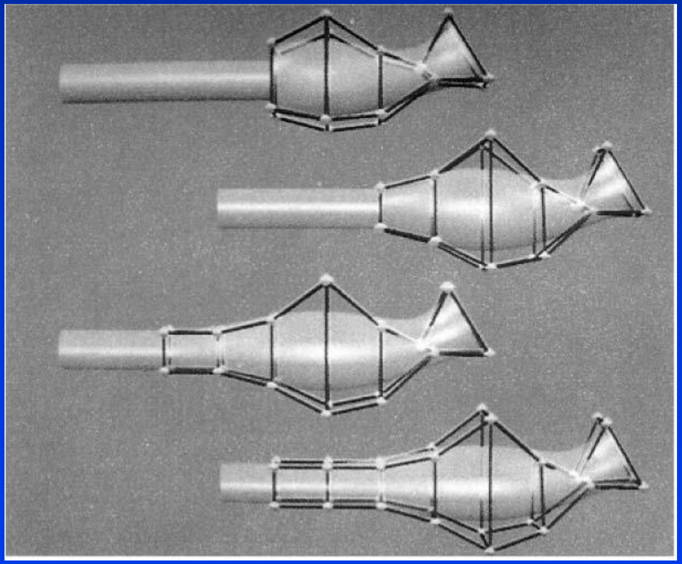
C^{0}

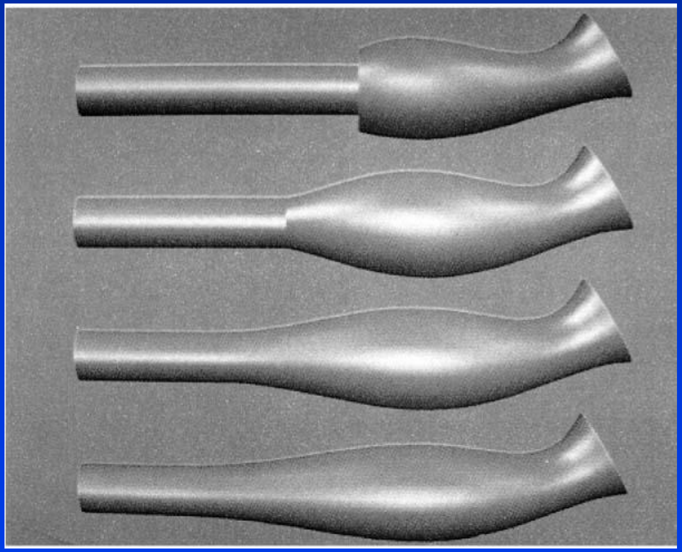
C^{1}

C^{2}

Smooth the deformed surface

Can be done by properly set the lattice position and (l, m, n) dimension



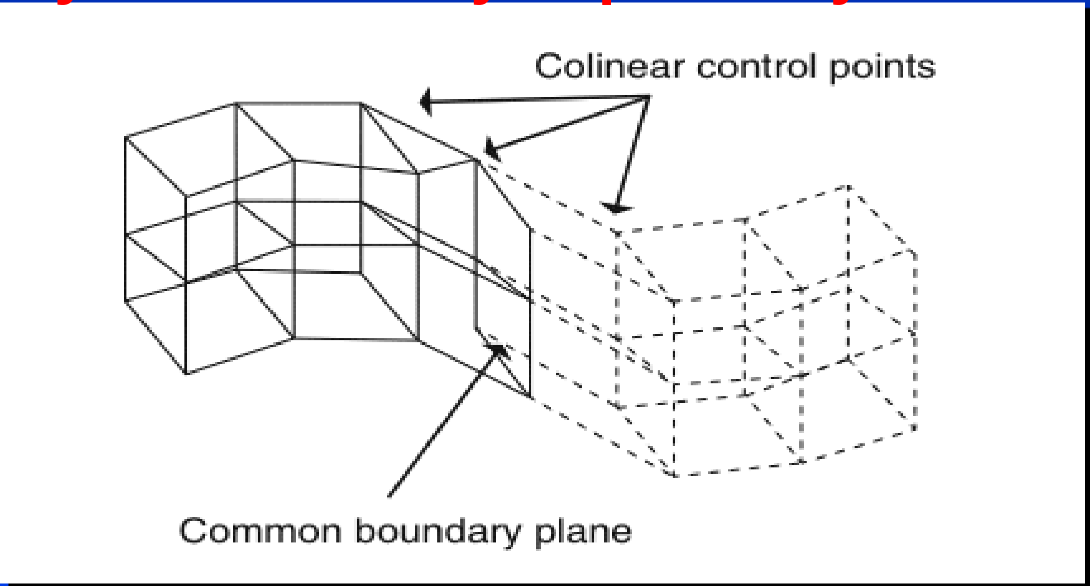


Department of Computer Science
Center for Visual Computing
STONY BROOK
STATE UNIVERSITY OF NEW YORK

Free-Form Deformations

- Continuities

As in Bezier curve interpolation
Continuity controlled by coplanarity of control points



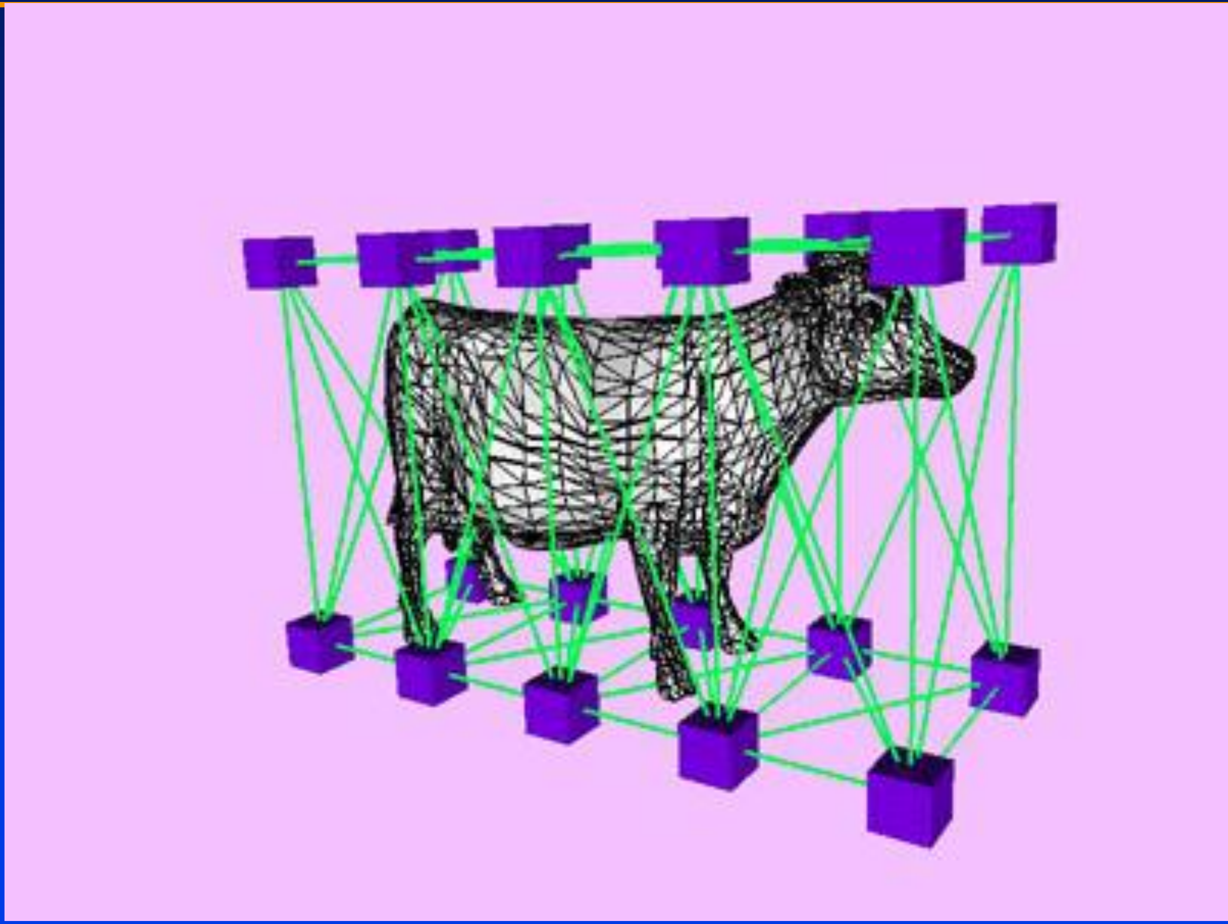
Volume Preservation

- Must ensure that the jacobian of the deformation is 1 everywhere

$$\left(\hat{x}, \hat{y}, \hat{z}\right) = \left(F(x,y,z), G(x,y,z), H(x,y,z)\right)$$

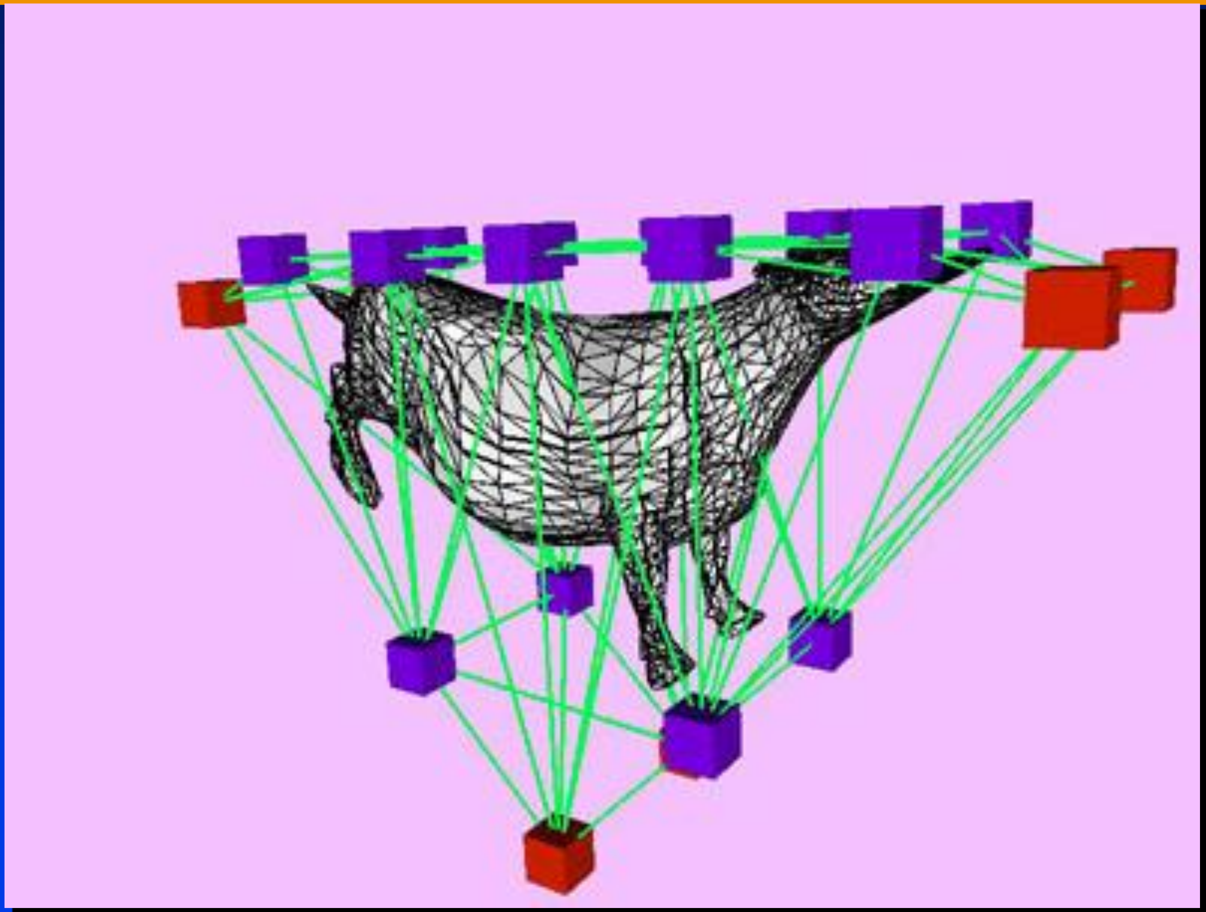
$$\begin{vmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial z} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} & \frac{\partial G}{\partial z} \\ \frac{\partial H}{\partial x} & \frac{\partial H}{\partial y} & \frac{\partial H}{\partial z} \end{vmatrix} = 1$$

FFD: Examples



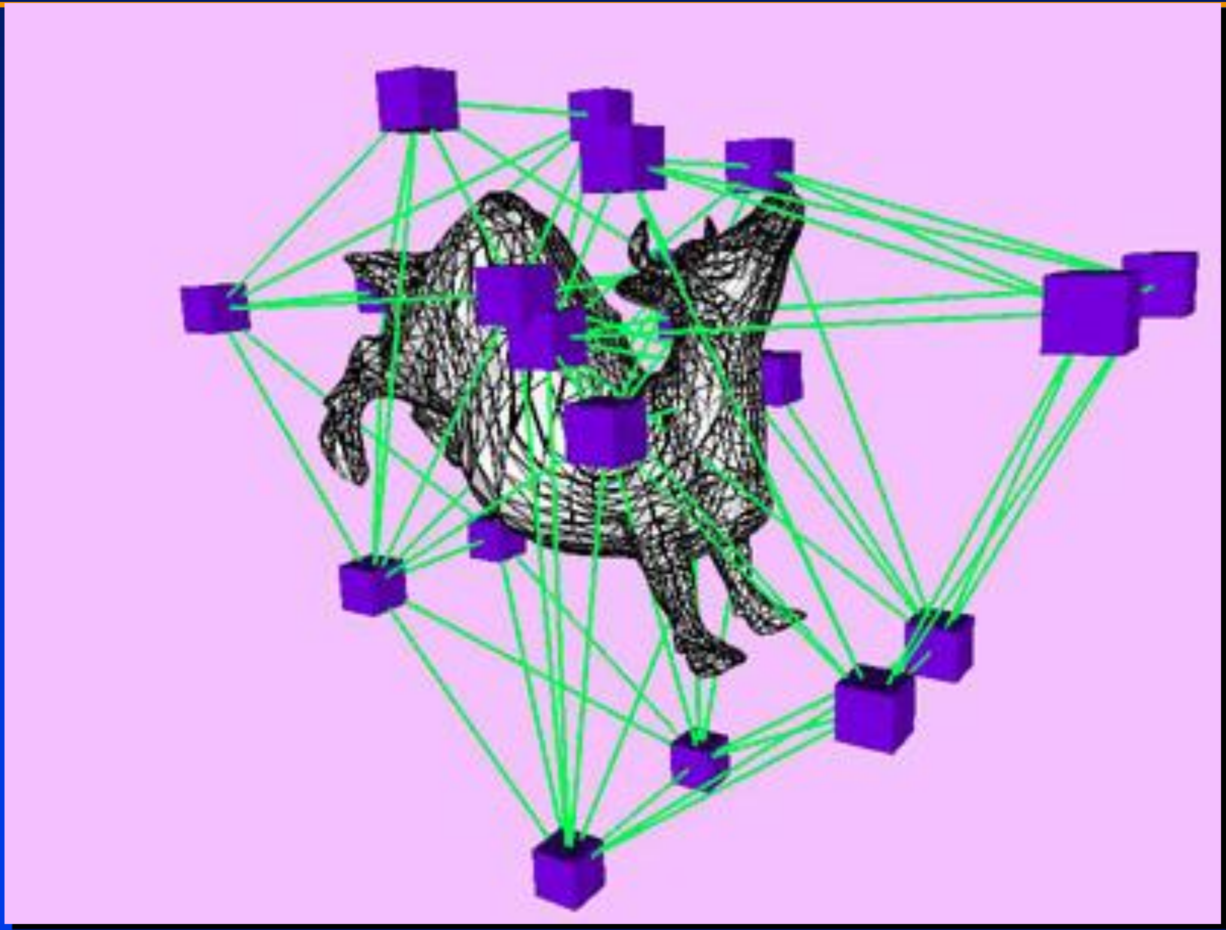
From "Fast Volume-Preserving Free Form Deformation
Using Multi-Level Optimization" appeared in ACM Solid Modelling '99

FFD: Examples



From “Fast Volume-Preserving Free Form Deformation Using Multi-Level Optimization” appeared in ACM Solid Modelling ‘99

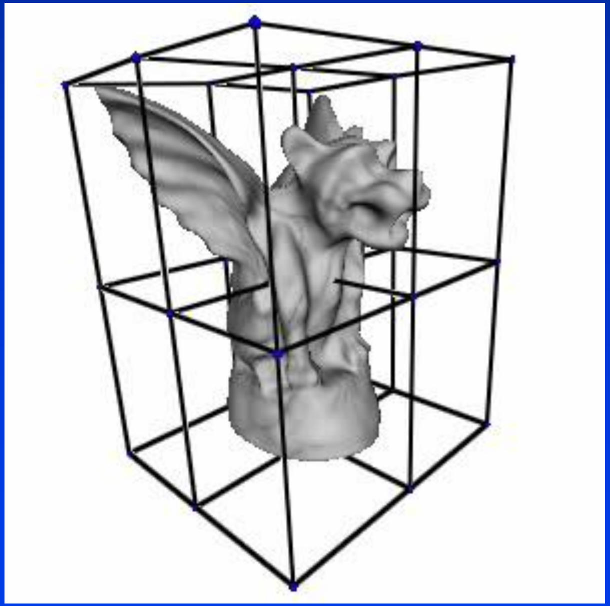
FFD: Examples



From “Fast Volume-Preserving Free Form Deformation Using Multi-Level Optimization” appeared in ACM Solid Modelling ‘99

Advantages

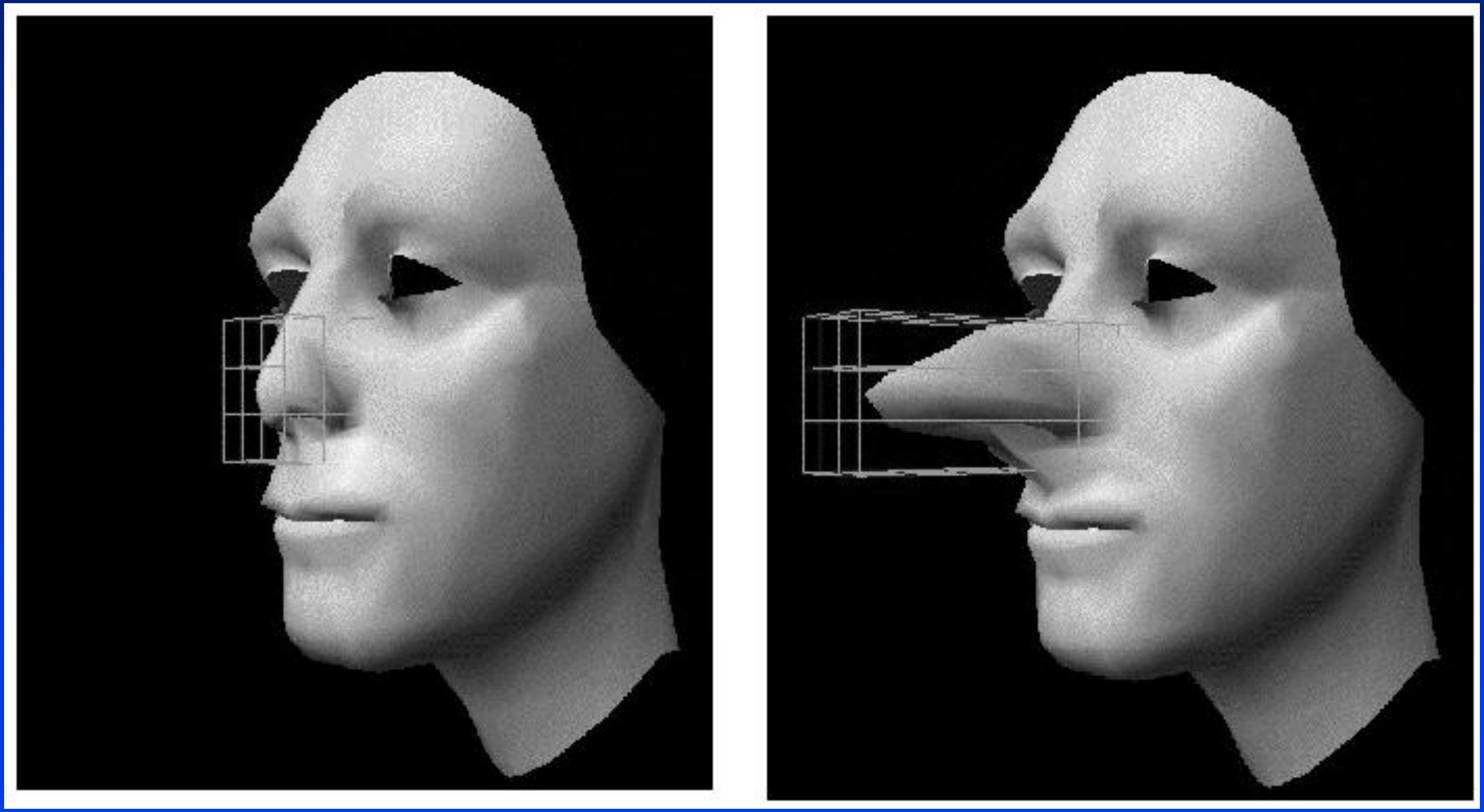
- Smooth deformation of arbitrary shapes
- Local control of deformations
- Computing the deformation is easy
- Deformations are very fast



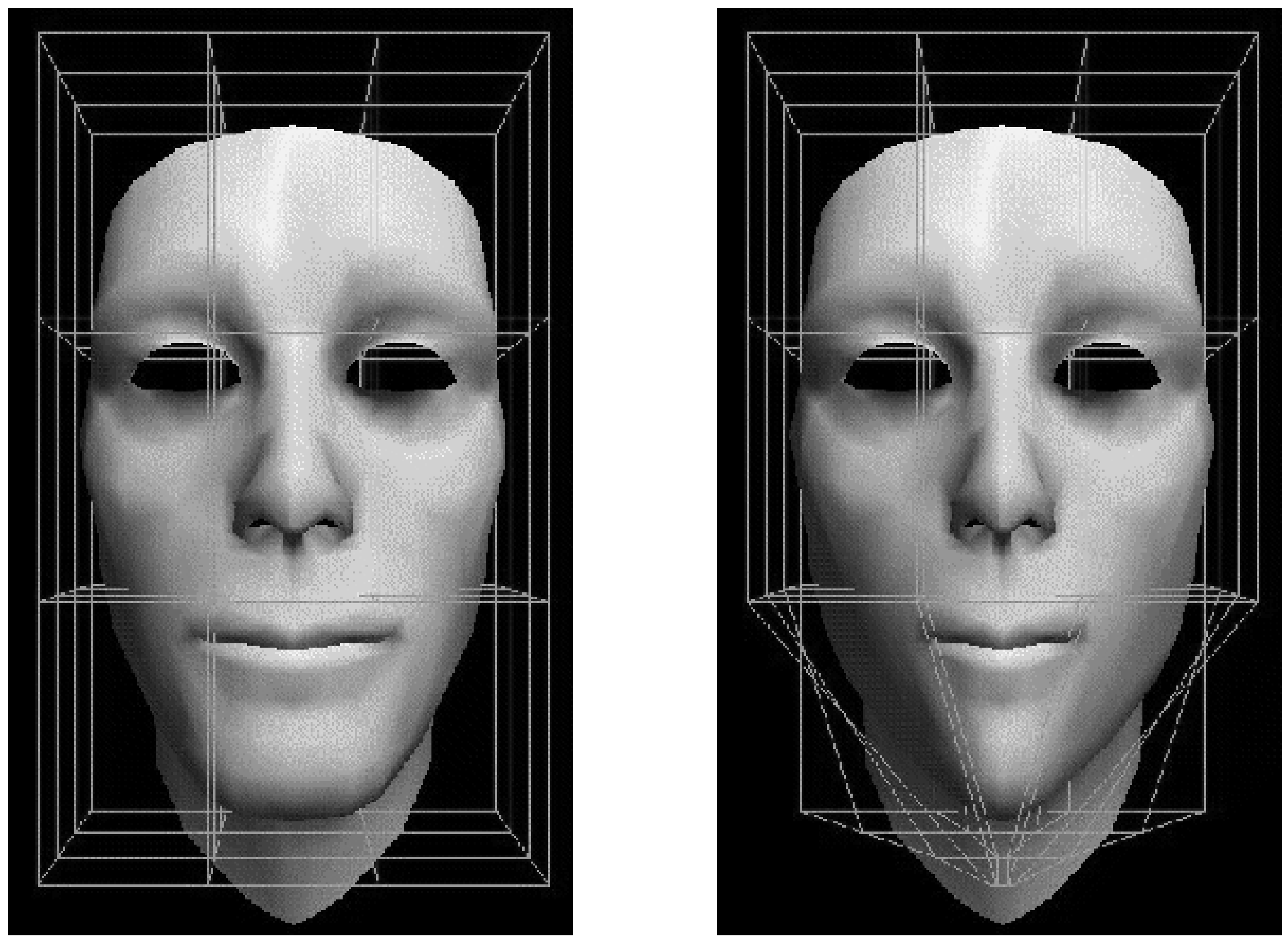
Disadvantages

- Must use cubical cells for deformation
- Restricted to uniform grid
- Deformation warps space.... not surface
  - Does not take into account geometry/topology of surface
- May need many FFD's to achieve a simple deformation

FFD Example



FFD Example

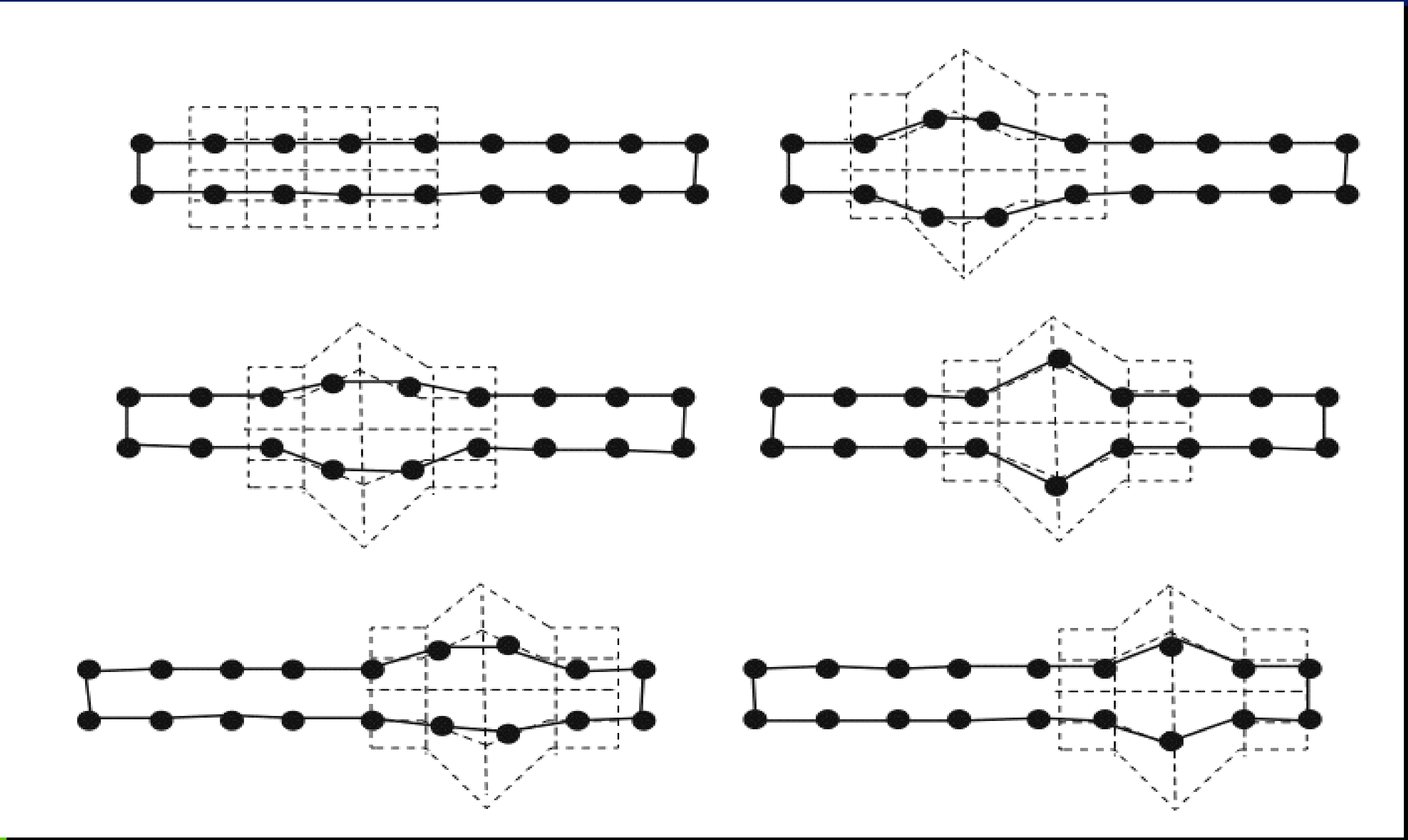


Free-Form Deformation

- Widely used deformation technique
- Fast, easy to compute
- Some control over volume preservation/smoothness

- Uniform grids are restrictive

FFD as a Animation Tool



Use FFDs to Animate

- Build control point lattice that is smaller than geometry
- Move lattice through geometry so it affects different regions in sequence
- Animate mouse under the rug, or subdermals (alien under your skin), etc.

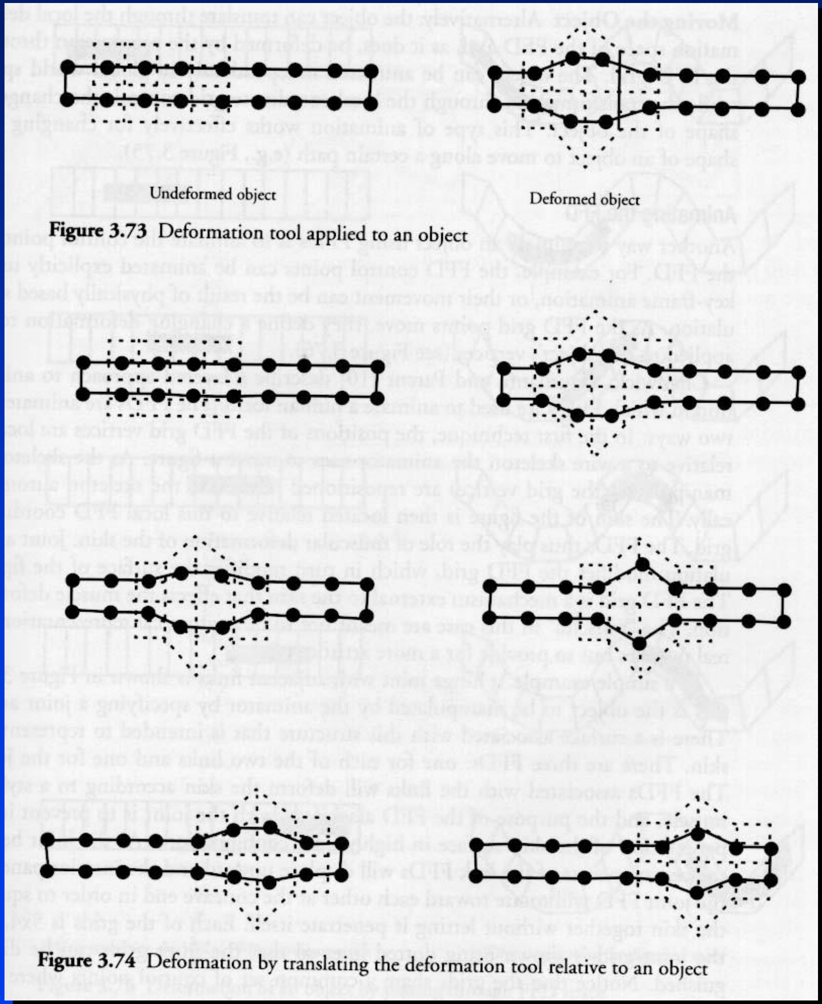
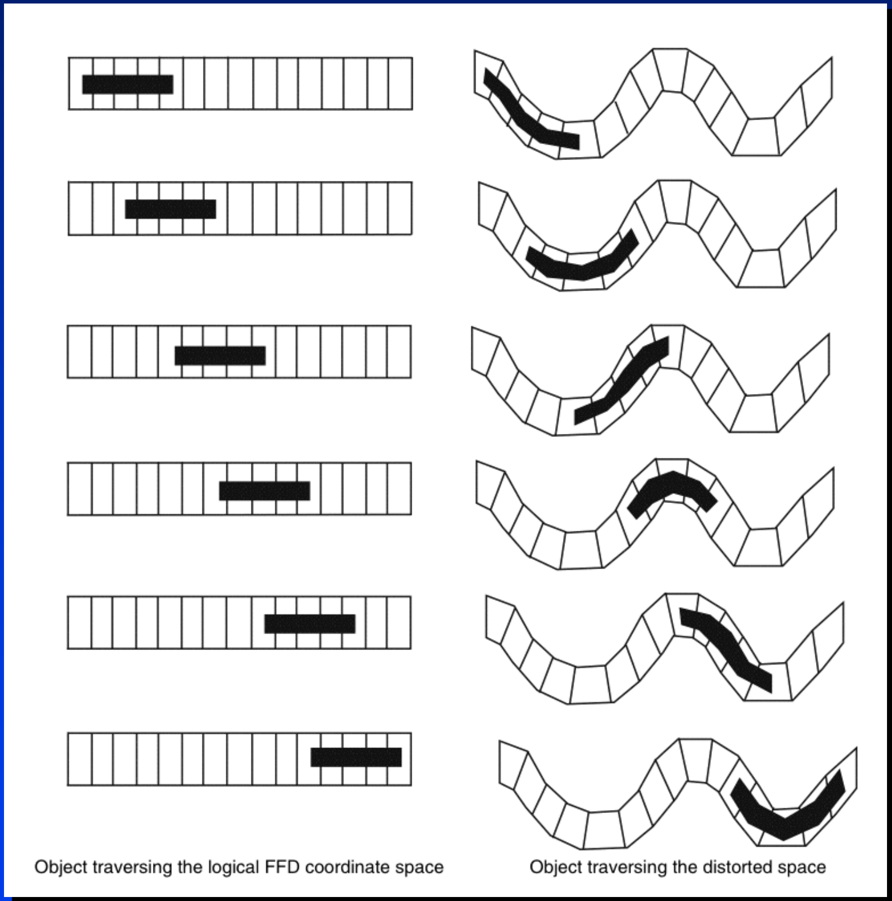


Figure 3.73 Deformation tool applied to an object

Figure 3.74 Deformation by translating the deformation tool relative to an object

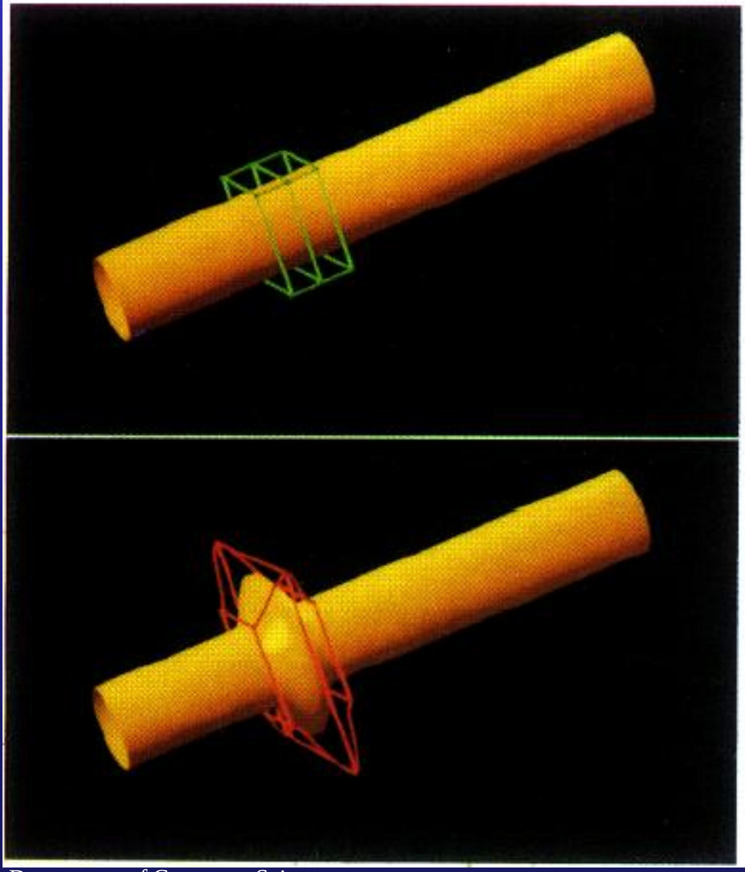
Use FFDs to Animate

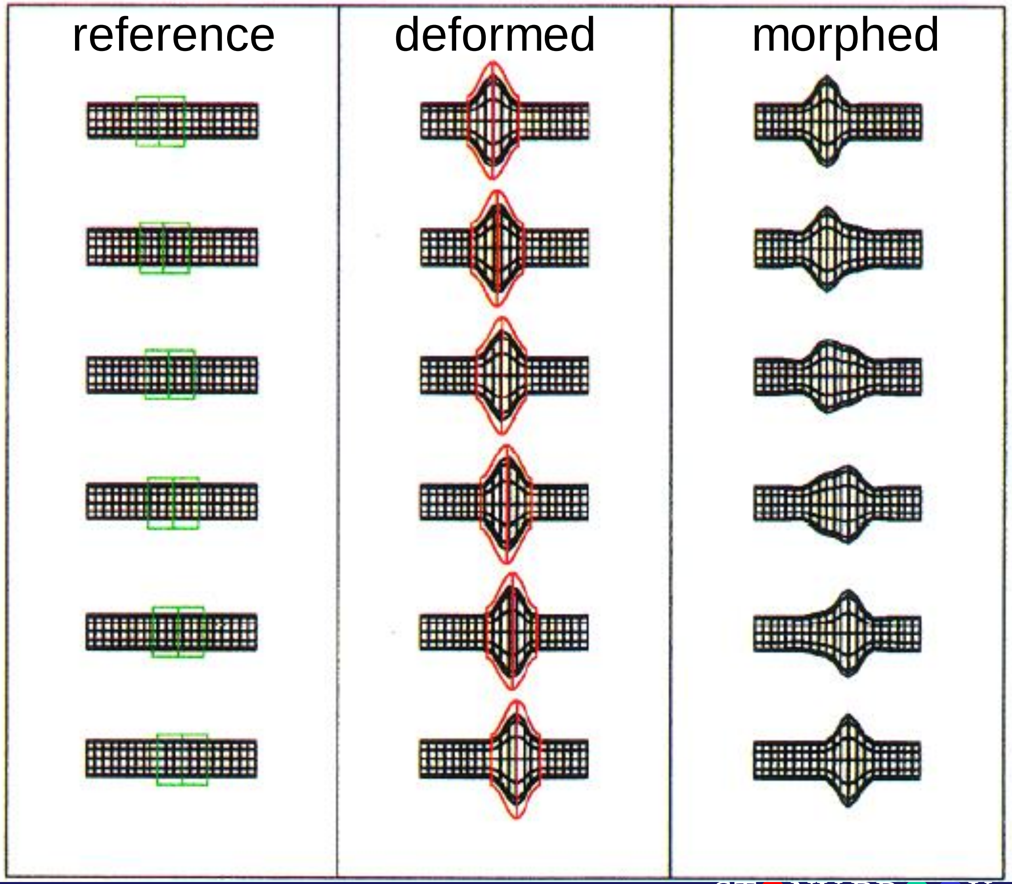
- Build FFD lattice that is larger than geometry
- Translate geometry within lattice so new deformations affect it with each move
- Change shape of object to move along a path



FFD Animation

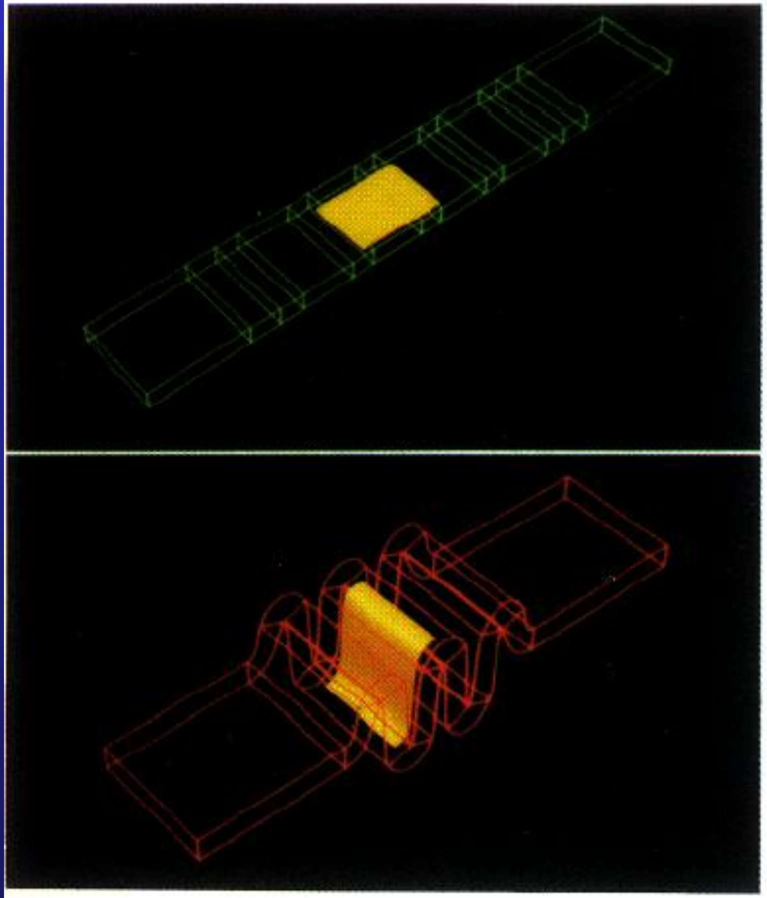
Animate a reference and a deformed lattice

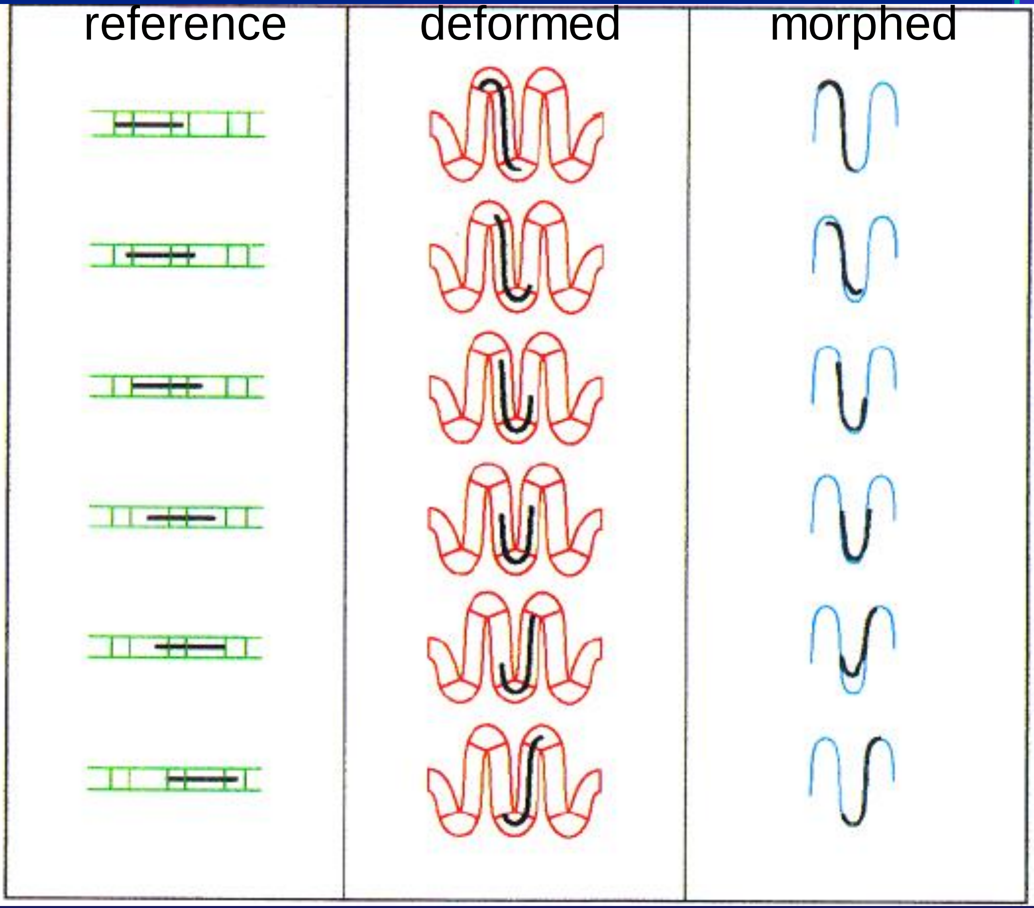




FFD Animation

Animate the object through the lattice



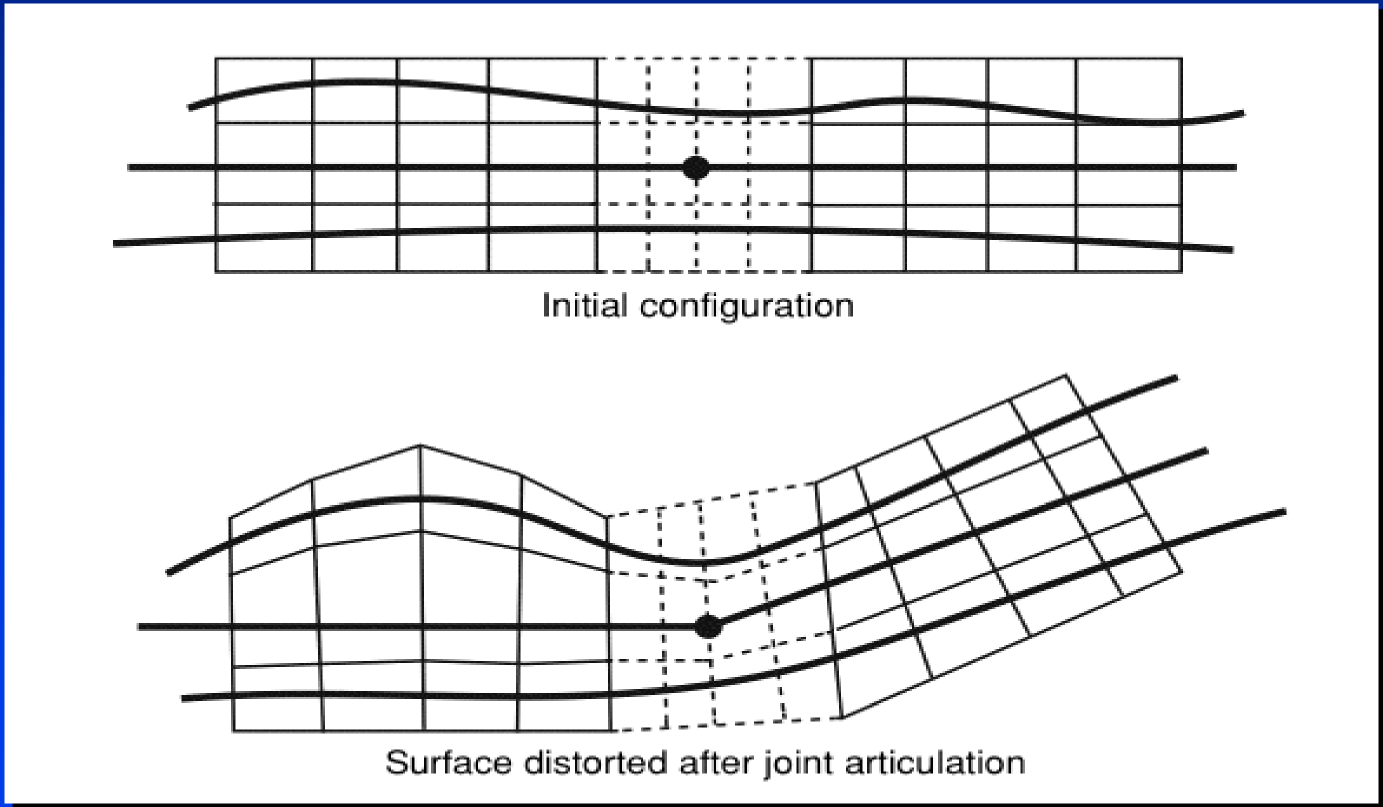


Animating the FFD

- Create interface for efficient manipulation of lattice control points over time
  - Connect lattices to rigid limbs of human skeleton
  - Physically simulate control points

Application: Skin, Muscle, and Bone Animation

Exo-muscular system
Skeleton -> changes FFD -> changes skin



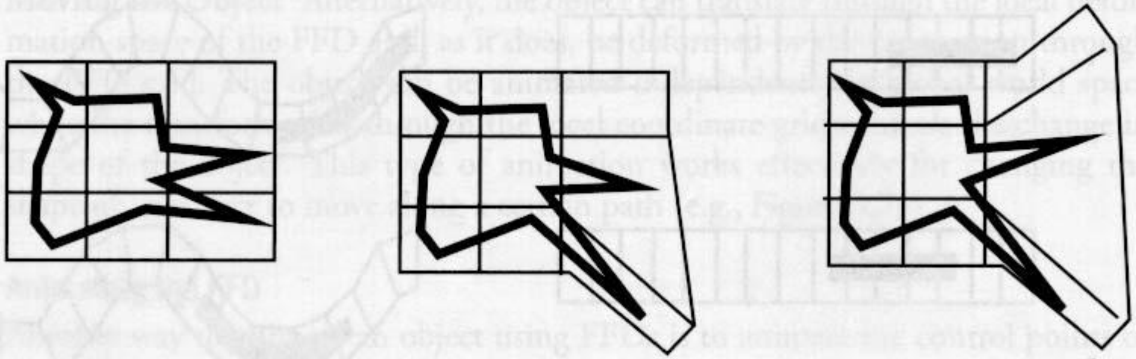


Figure 3.76 Using an FFD to animate a figure's head

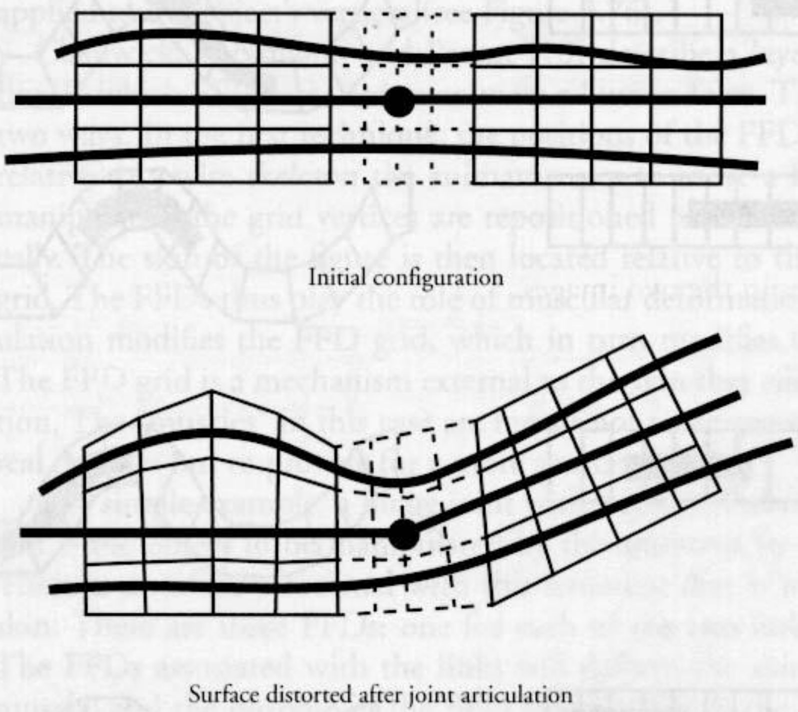
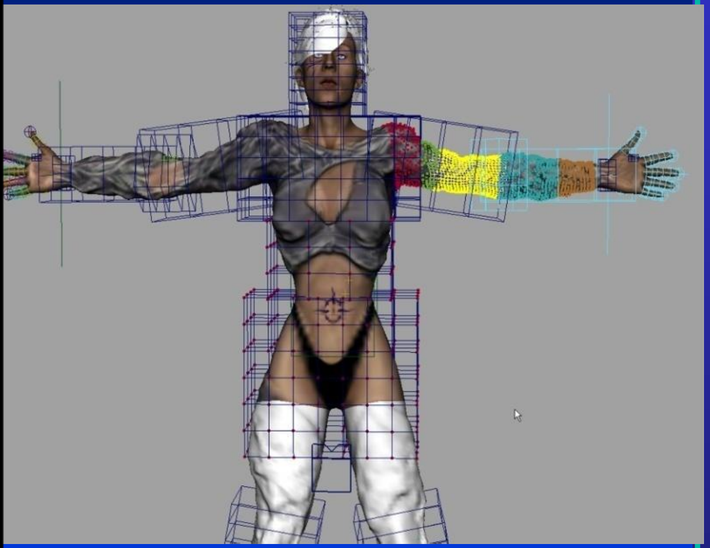
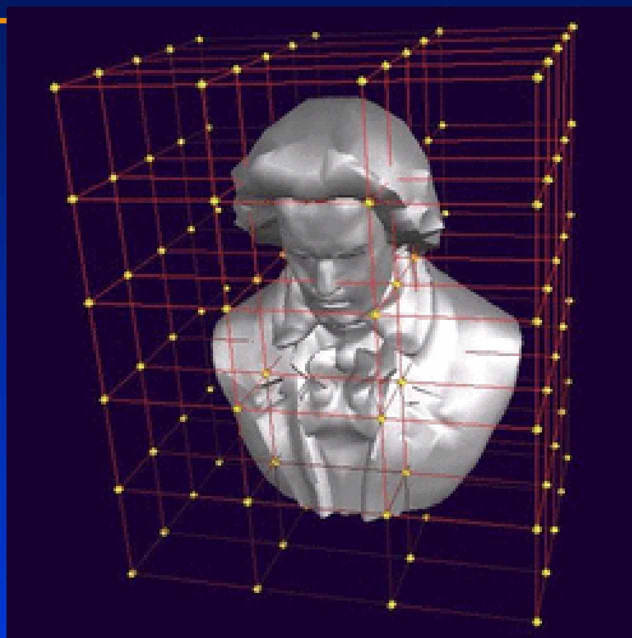


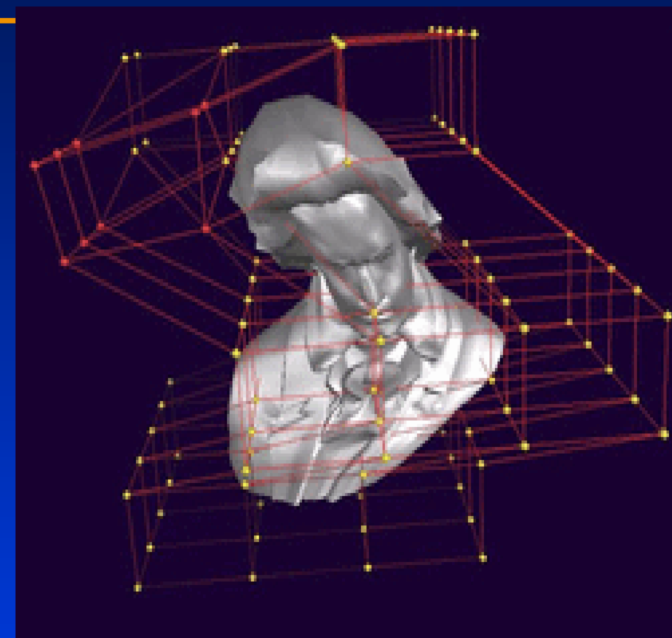
Figure 3.77 Using FFD to deform a surface around an articulated joint

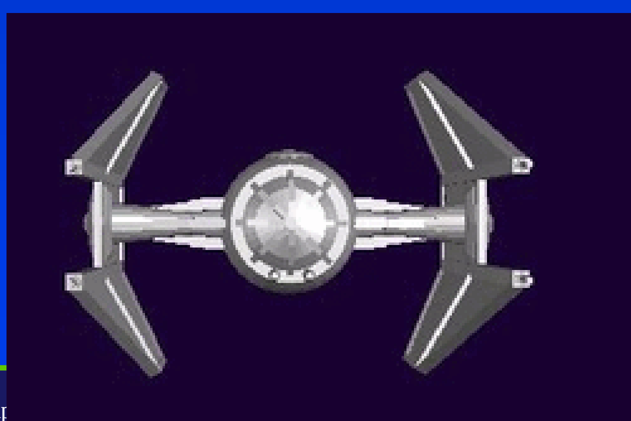
FFD for Human Animation

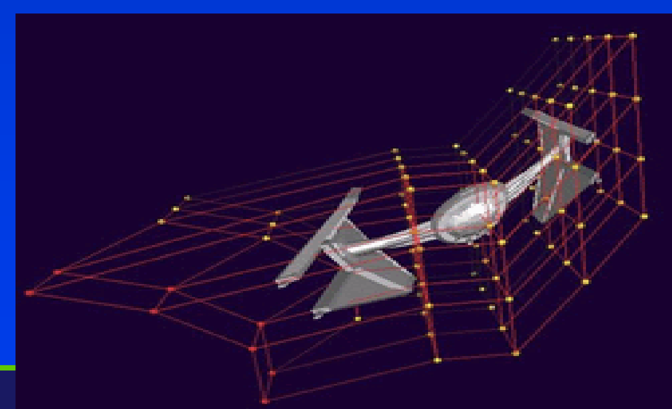


Free-Form Deformation

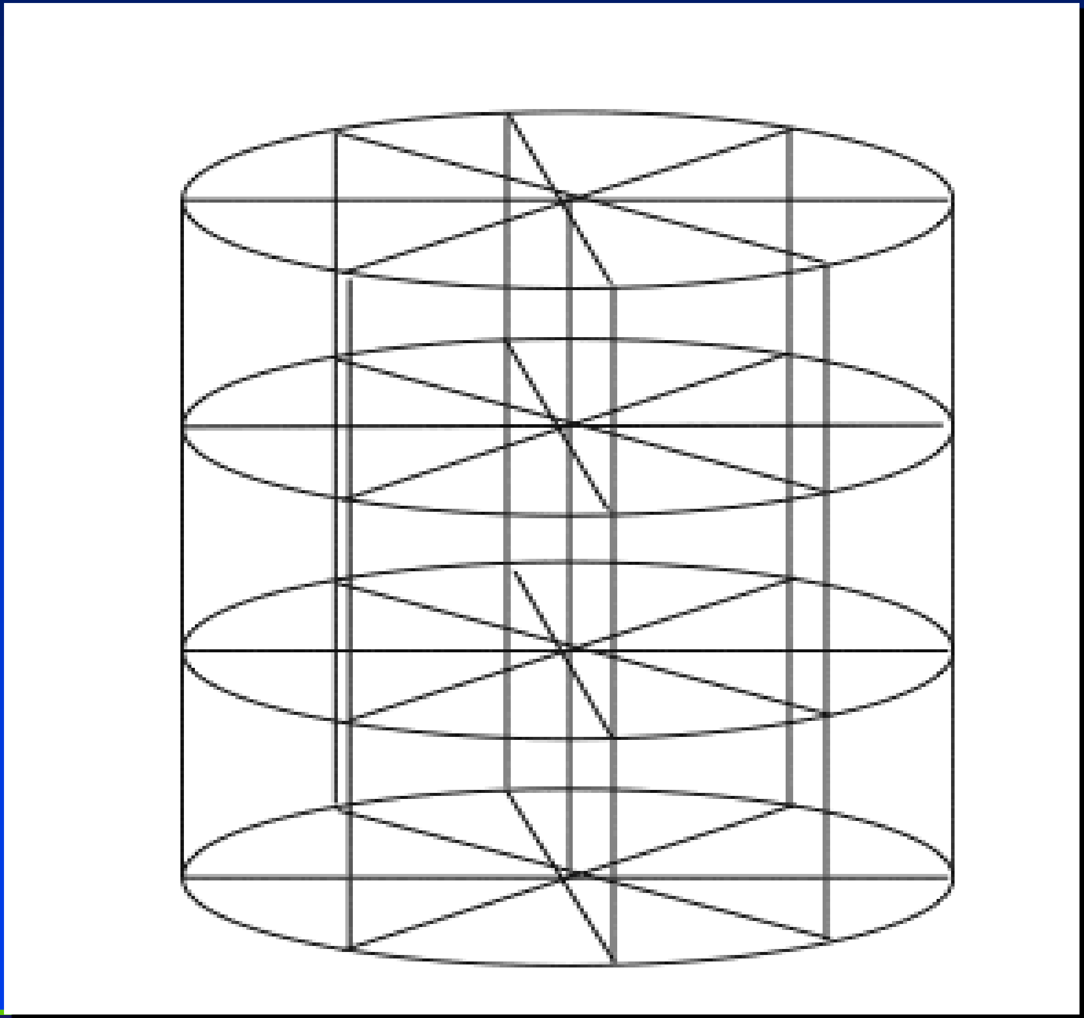




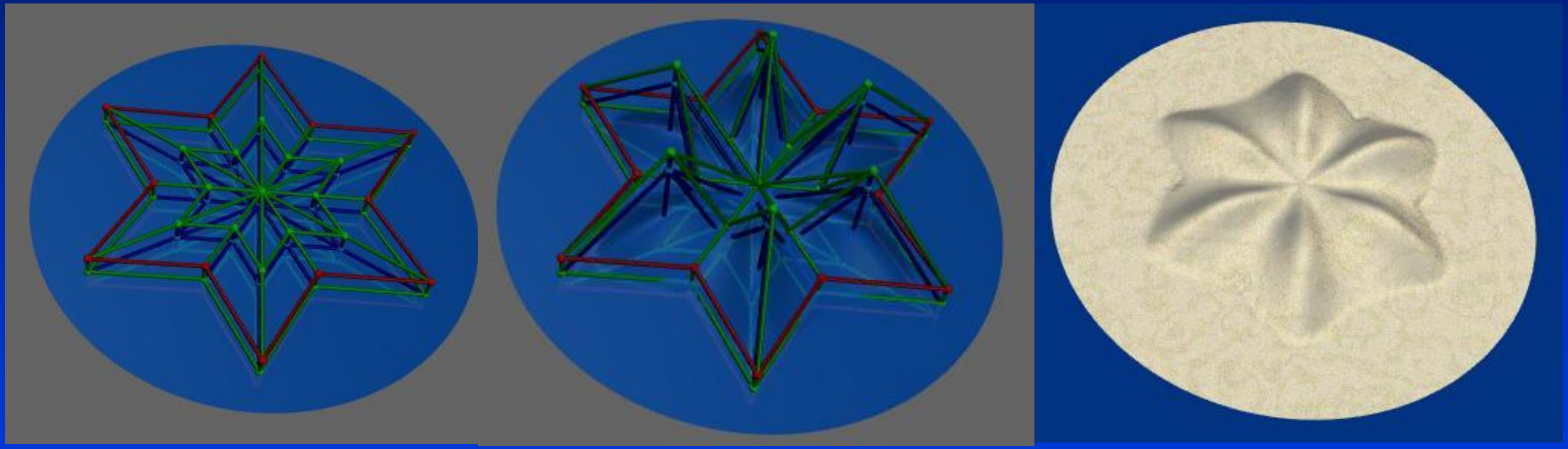




Non-Tensor-Product Grid Structure

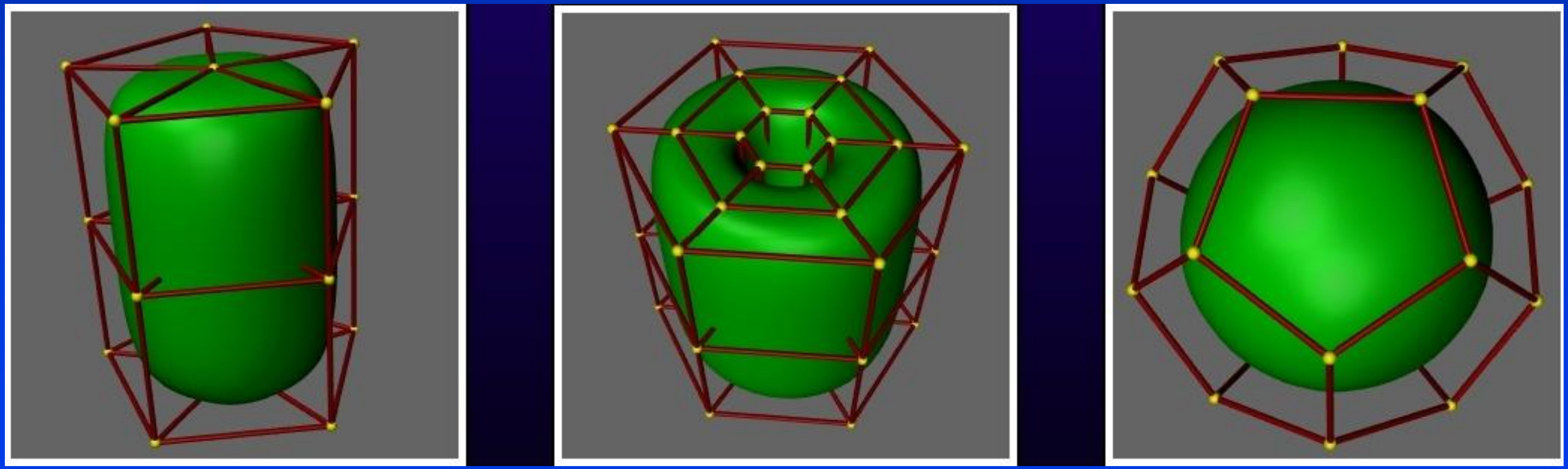


Arbitrary Grid Structure (Star-Shape)



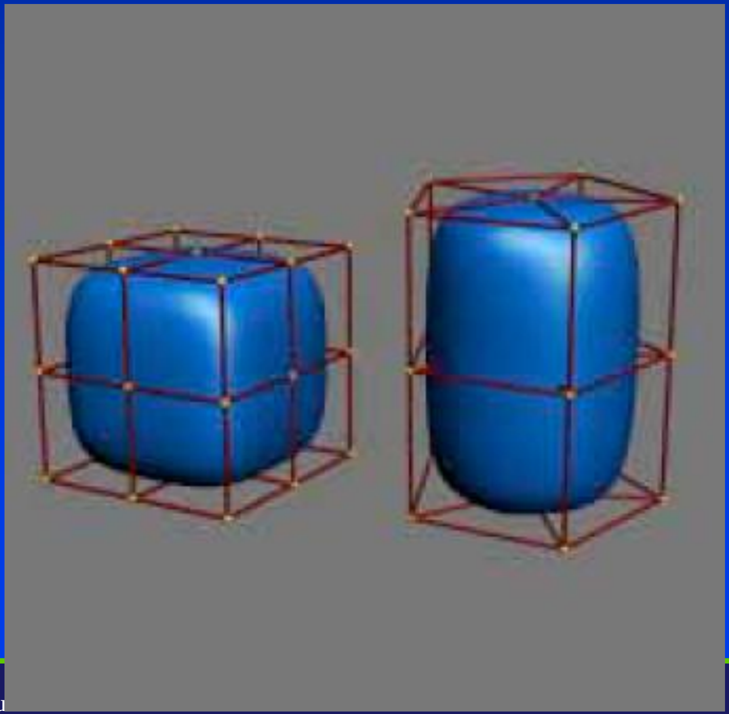
Volume defined by Arbitrary Lattices

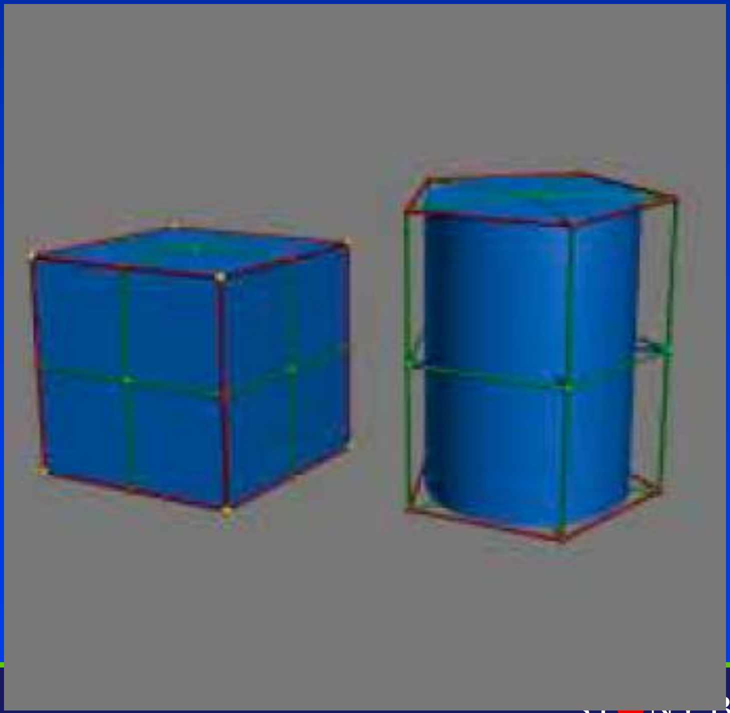
- The volumetric regions of space results from Catmull-Clark subdivision method.



Modified Refinement Rules

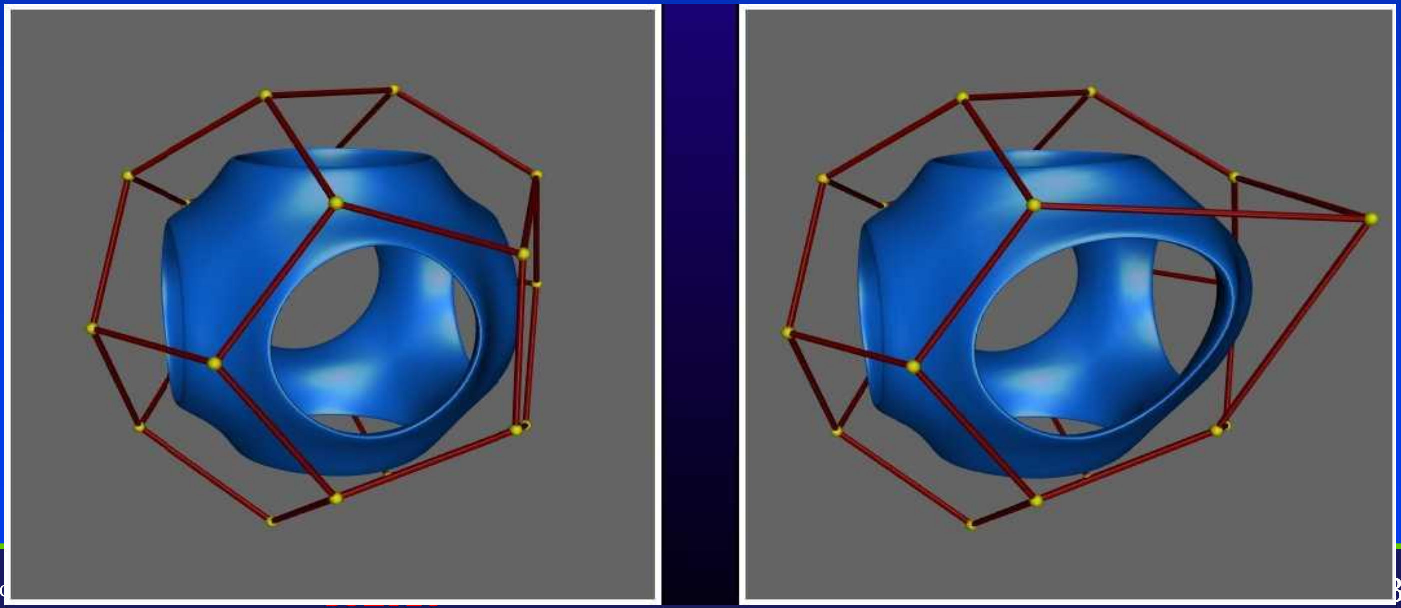
- Green: boundary edges.
- Red: sharp edges.
- Yellow: corner vertices





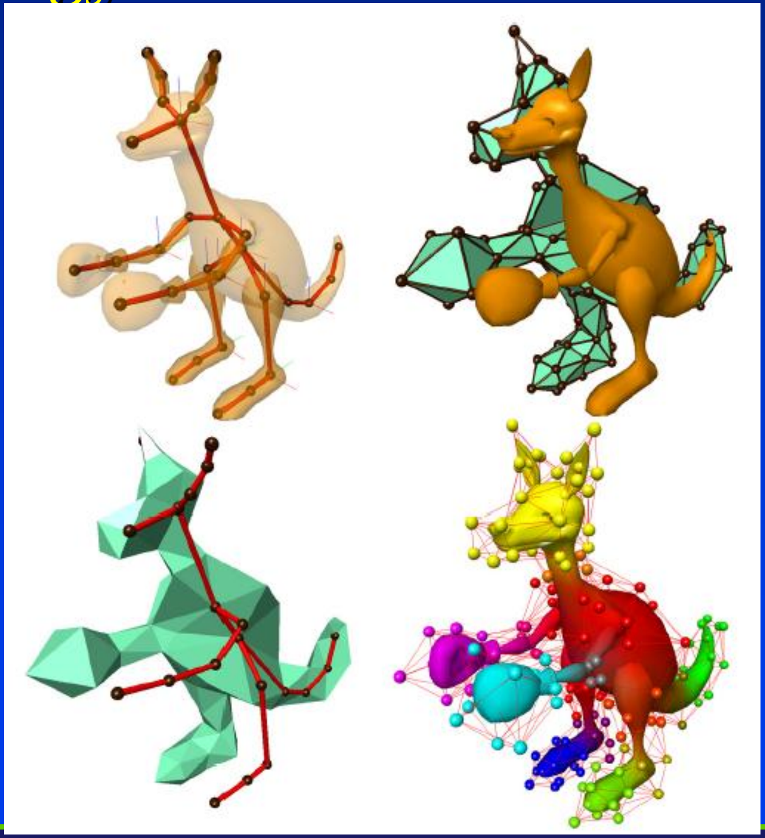
Arbitrary Topology

- Previous method can only handle a parallelepiped lattice.
- A new method allows lattices of arbitrary topology.

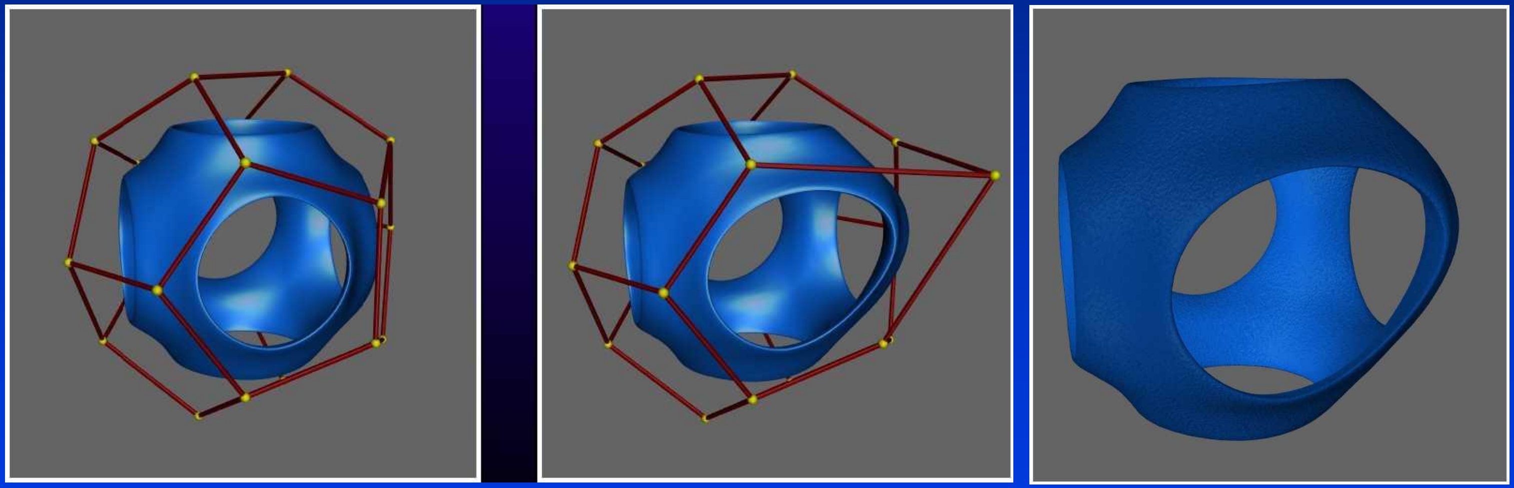


Arbitrary Topology FFDs

- The concept of FFDs was later extended to allow an arbitrary topology control volume to be used

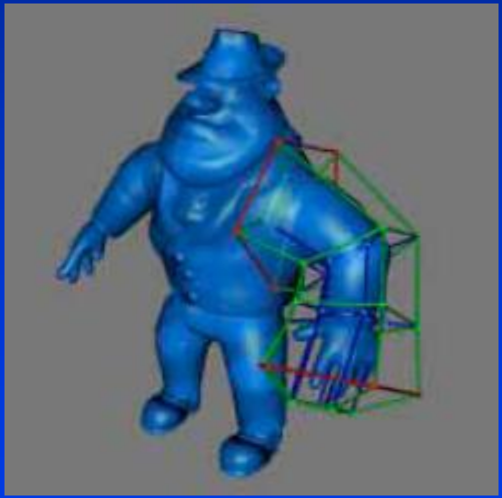
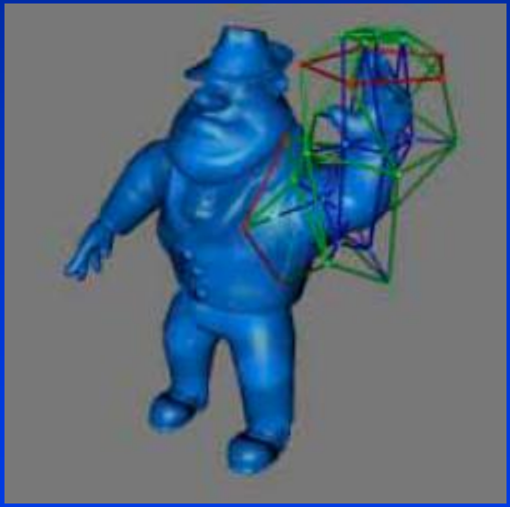


Results



Results

- Deform a monster’s arm

Direct Manipulation