Animation II: Soft Object Animation

Watt and Watt Ch.17

Soft Object Animation

Animation I: skeletal animation

forward kinematics $x=f(\theta)$

inverse kinematics $\theta=f^{-1}(x)$

Curves and Surfaces I&II: parametric representation

smooth surfaces (Bezier, B-spline, NURBS)

How can we animate the surface based on the skeleton?

- shape changes at each frame
- surface deforms based on skeleton model
- link surface shape(space) to animation (time)
- produces realistic animation

Polygonal Mesh Animation

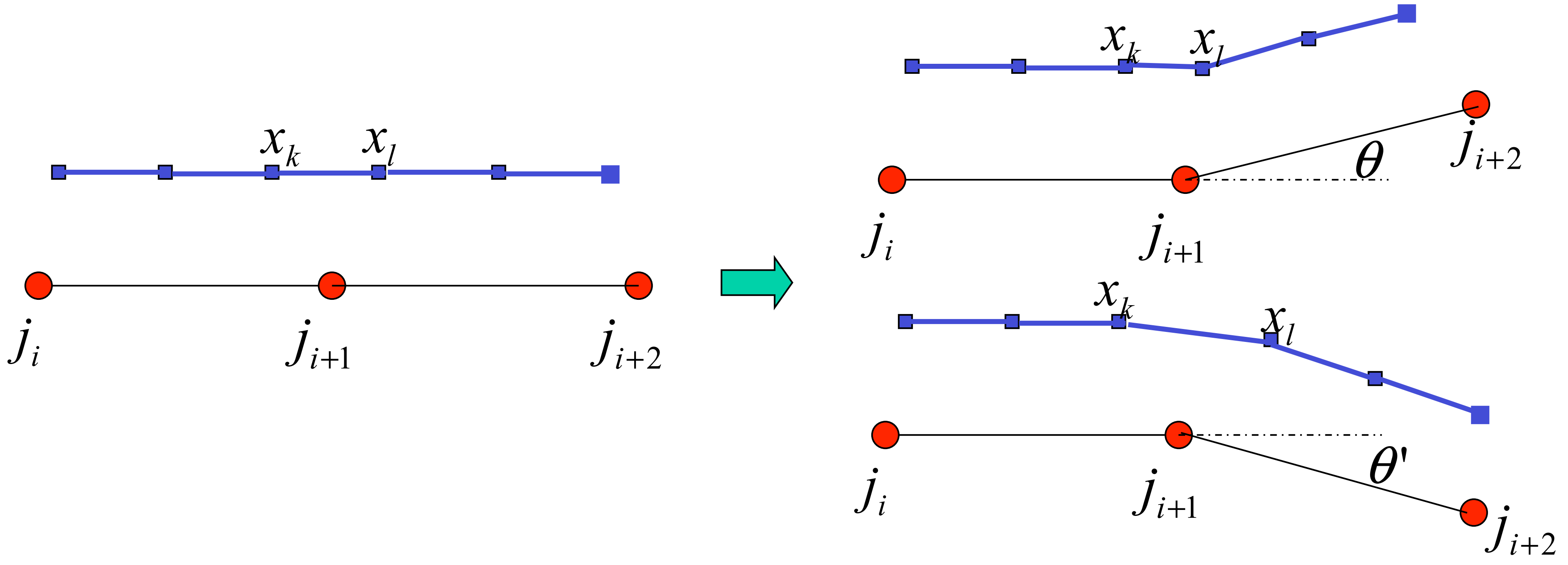
Change vertex positions as a function of time x(t)
Connectivity (topology) of mesh remains constant

To achieve smooth deformation vertices must be moved together(not independently)

Problems occur if vertices cross over when deformations are applied

Rigid Mesh Animation

Each vertex position is animated according to the underlying articulation



Rigid Animation of a mesh vertex:

Vertex rigidly attached to ith segment :

$$x_k = \left(\prod_{r=0}^{i} E_r \right) x_{k0}$$

E_r is the transformation (rotation) for the rth segment

x_{k0} is the default position of point x_k without any transformation $E_r = I$

Similarly for a vertex attached to the i + 1th segment :

$$x_l = \left(\prod_{r=0}^{i+1} E_r \right) x_{l0} = \left(\prod_{r=0}^{i} E_r \right) E_{i+1} x_{l0}$$

ie the change in vertex position between segements i and i + 1 is
determined by the transform of joint i + 1 E_{i+1}

Problems:

- self-interesection of mesh
- mesh collapse
- unrealisitic surface deformation

Simple Non-Rigid Mesh Animation

Vertex weighting :

$$x_k = \sum_{i=0}^{n} w_i \left(\prod_{r=0}^{i} E_r \right) x_{k0}$$

Weighted average of transformations of point x_{k0} by transforms of each joint

n - number of joints in the kinematic

w_i - is the weight for the ith point in the kinematic chain

$$\sum_{i=0}^{n} w_i = 1$$

is a convex sum of the tranforms for each joint

(ie resulting point is inside convex hull)

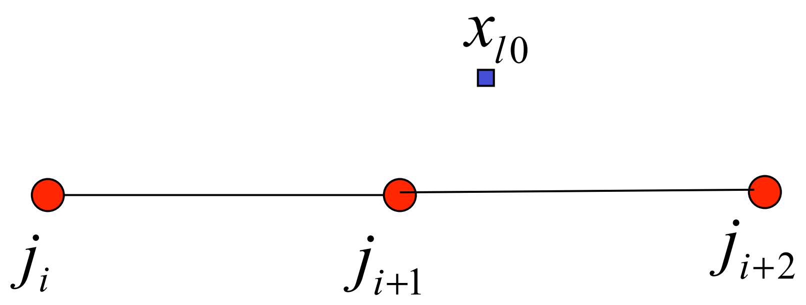
Typically, point just depends on transforms for 2 joints

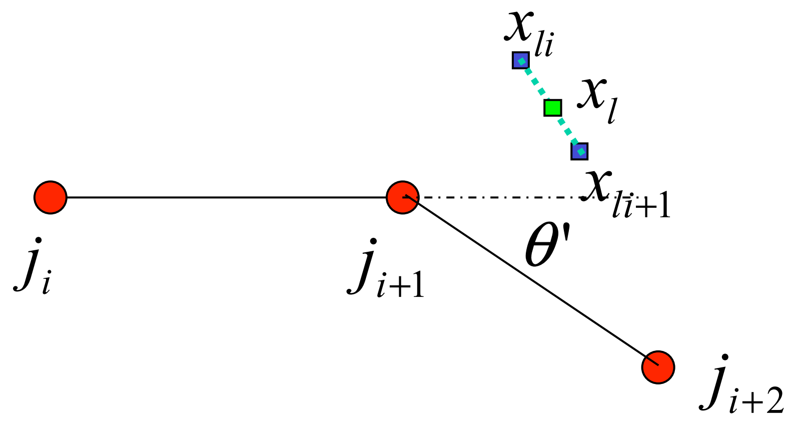
$w_i \neq 0, w_{i+1} \neq 0, w_j = 0$ for $\mathrm{j} \neq \mathrm{i}, \mathrm{i}+1$

$$x_l = w_i \left(\prod_{r=0}^{i} E_r \right) x_{l0} + w_{i+1} \left(\prod_{r=0}^{i+1} E_r \right) x_{l0}$$

$$= \left(\prod_{r=0}^{i} E_r \right) (w_i + w_{i+1} E_r) x_{l0}$$

Therefore, x_l is the weighted sum of the tranformed points for x_{l0} rigidly attached to joint i and rigidly attached to joint i+1 where $w_i + w_{i+1} = 1$





Vertex weighting:

- Fast/Simple (only small additional cost to rigid)
- Supported by many animation packages
- How to set weights?
- Works for simple chains of joints (knee, elbow)
- For complex joints (shoulder, hip) there may be no single acceptable set of weights
- incorrect weights result in visible artifacts (mesh collapse, popping of vertices)

Mesh Morphing

Interpolation of mesh between a set of pre-defined default shapes

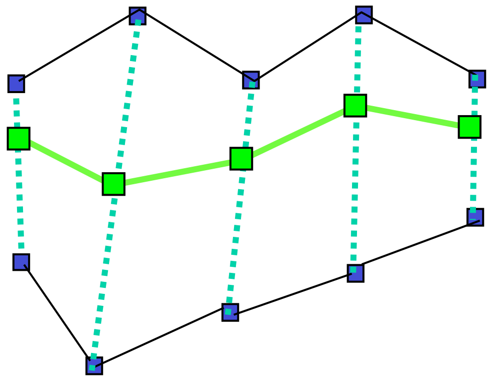
Pre-define a set of mesh M_i $i = 0 \ldots m$

Morphed mesh is a linear combination of the pre-defined meshes:

$$M = \sum_{i=0}^{m} \alpha_i M_i \quad \text{where} \quad \sum_{i=0}^{m} \alpha_i = 1$$

giving the morphed position of the ith vertex:

$$x_l = \sum_{i=0}^{m} \alpha_i x_{li}$$



x_{li} is the position of vertex x_l on the lth mesh

α_i is the blending factor

Morphing is widely used for face animation

- set of default meshes are defined for expressions etc.
- can also be used for skeletal animation with blending factors dependent on joint angles

Problems with deforming polygonal meshes:

- As vertices are deformed they may move apart
- reduces sampling resolution
- ‘3D spatial aliasing’ of surface representation
- most noticable at silhouette edges of object
  (edges of polygons visible)
- smooth animation using polygonal meshes requires
  high resolution mesh in surface areas that are deformed
  (expensive representation and animation)
- adaptive subdivision of mesh during animation

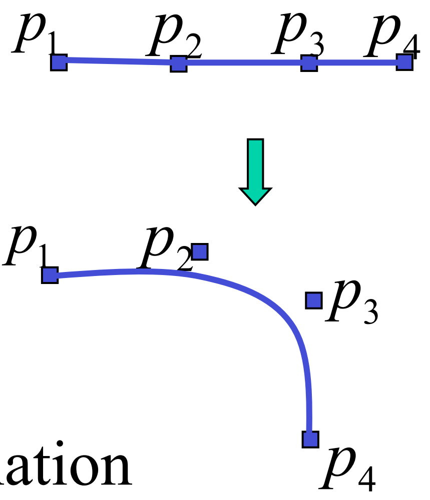
Animation of Parametric Surfaces

Change control point positions
Obtain new smooth surface by blending new control point positions

For example cubic surface is computed by blending 16 control points

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u) b_j(v) p_{ij}$$

Animation changes the control points
- changes coefficients of basis functions
- results in a new smooth representation



How to animate control points?
- smooth surface may not be desired deformation
- spatial aliasing will result if there are insufficient control points

Deformation Independent of Surface Representation

Define a deformation function on the space (volume) in which the surface is defined

- independent of representation
- applied to polygon mesh or parametric representation

Deformation function:

$$x' = f(x)$$

or for animation over time: $x'(t) = f(x,t)$

Consider two approaches:

(i) Non-linear global deformation

(ii) Free-form deformations FFD

Non-Linear Global Deformations

Apply a single deformation x' =f(x) to entire object

Tapering :

$$x' = Tx = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} rx \\ ry \\ z \end{bmatrix}$$

where r = f(z)

r = constant => scale

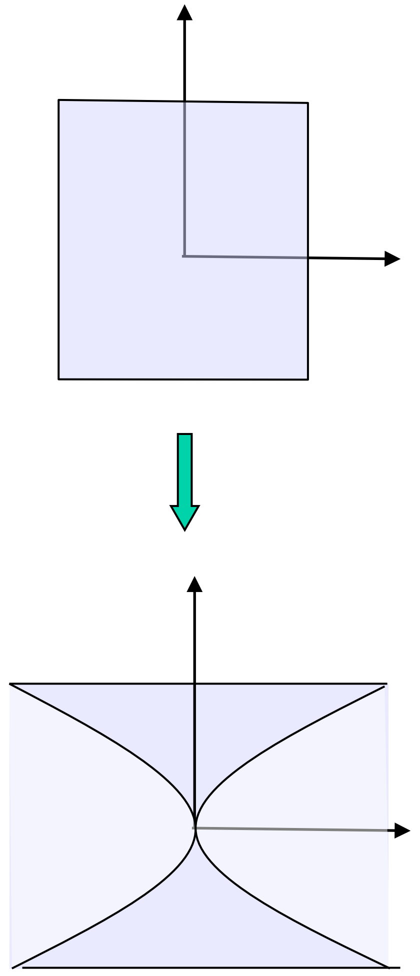
r = z => linear taper

$r = z^2$ => quadratic taper

Apply trasformation to all points on object

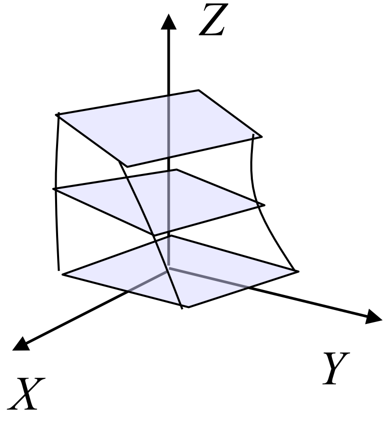
(mesh verticies or control points)

results in a global deformation



Twisting :

$$x' = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Rotation about z - axis

$$\theta = f(z)$$

$f(z)$ rate of twist about z - axis

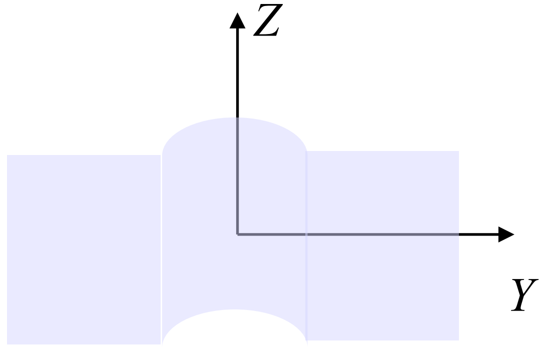
Bend : comprises a bent region + a region which is rigidly transformed

For a bend along the y - axis between $y_{\min} \leq y \leq y_{\max}$

and the centre of the bend at $y = y_0$

Bend angle $\theta = \frac{1}{\mathrm{r}}(y_b - y_0)$ where r is the bend radius

$$y_b = \begin{cases} y_{\min} & y \leq y_{\min} \\ y & y_{\min} \leq y \leq y_{\max} \\ y_{\max} & y_{\max} \leq y \end{cases}$$



The deformation is given by :

$$x' = x$$

$$y' = \begin{cases} -\sin\theta(z-r) + y_0 + \cos\theta(y - y_{\min}) & y \leq y_{\min} \\ -\sin\theta(z-r) + y_0 & y_{\min} \leq y \leq y_{\max} \\ -\sin\theta(z-r) + y_0 + \cos\theta(y - y_{\max}) & y_{\max} \leq y \end{cases}$$

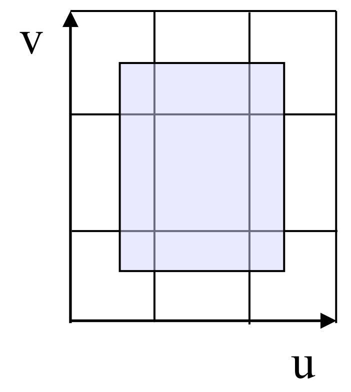
$$z' = \begin{cases} \cos\theta(z-r) + r + \sin\theta(y - y_{\min}) & y \leq y_{\min} \\ \cos\theta(z-r) + r & y_{\min} \leq y \leq y_{\max} \\ \cos\theta(z-r) + r + \sin\theta(y - y_{\max}) & y_{\max} \leq y \end{cases}$$

Free-form deformation

General transformation

- object is embedded in a space (volume) which is deformed
- smooth deformation of the space produces smooth deformation of the object
- may be global or local

Example: Represent plane as a bicubic bezier
define point in the plane in (u, v) coordinates
define a regular grid of 16 points p_{ij}



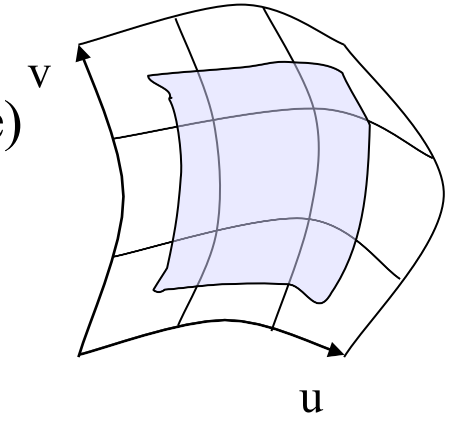
Bicubic patch $$Q(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}B_i(u)B_j(v)p_{ij}$$

Shape Deformation:

(i) Define the 2D shape in the regular u, v coordinates
(ie each vertex of polygonal shape has a (u, v) coordinate)

(ii) Change control point positions p_{ij}
result in a distortion of the u, v coordinates

(iii) Deform the 2D shape by computing the new location
according to the distorted u, v coordinate of each point
on the shape



Deformation based on a parametric representation of a surface patch extends directly to parametric representation of a volume

Example : Tricubic Bezier hyperpatch (volume)
volumetric latice represented by 64 points p_{ijk}

Tricubic hyperpatch $$Q(u,v,w) = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{p} B_i(u)B_j(v)B_k(w)\,p_{ijk}$$

$B_i(u), B_j(v), B_k(w)$ are Bernstein Bezier polynomials of degree 3

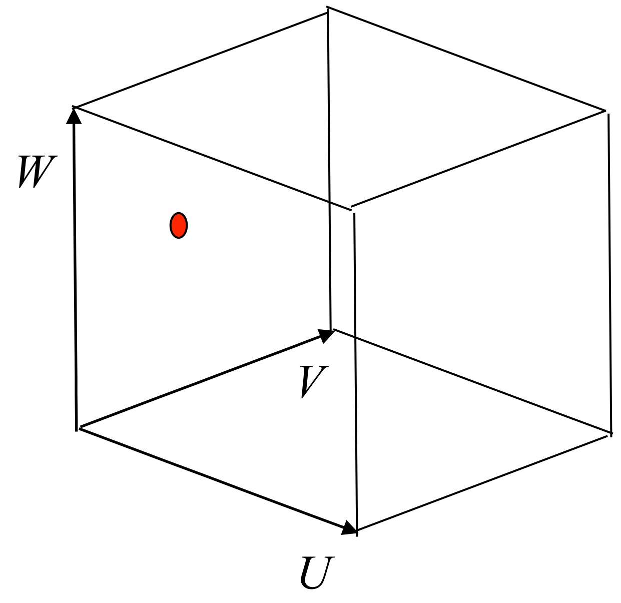
3D Shape Deformation :

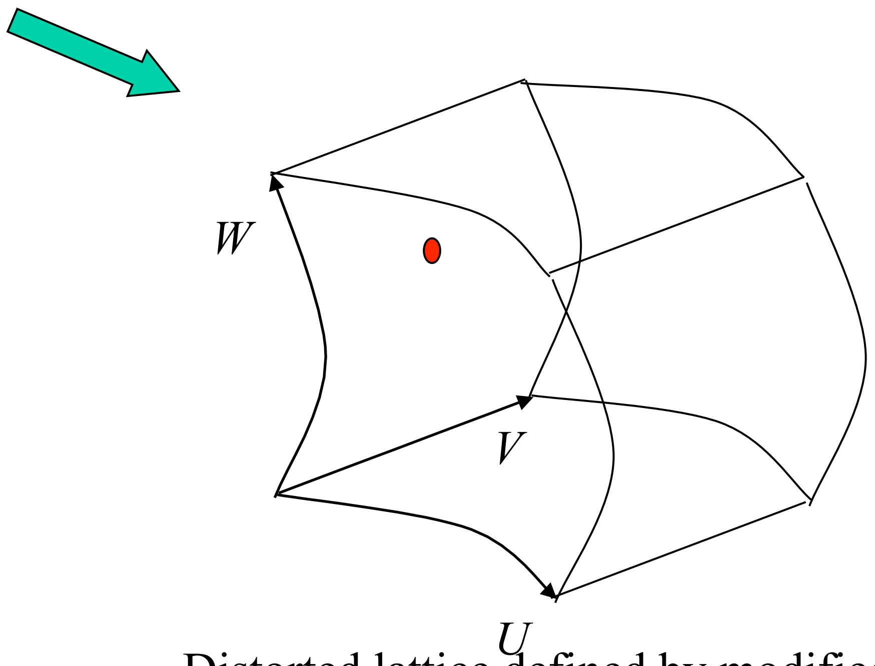
(i) Define shape within a regular lattice (u, v, w)

(ii) Change control point p_{ijk}

(iii) Compute deformed shape from (u, v, w) coordinates of points on the shape using the tricubic hyperpatch

Regular lattice defined by 64 control points





Distorted lattice defined by modified control point positions
- point location is computed from (u,v,w)

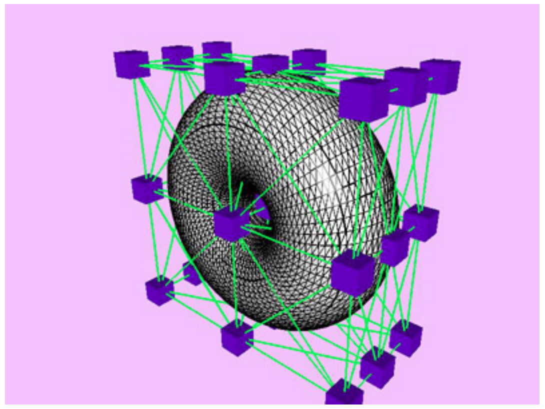
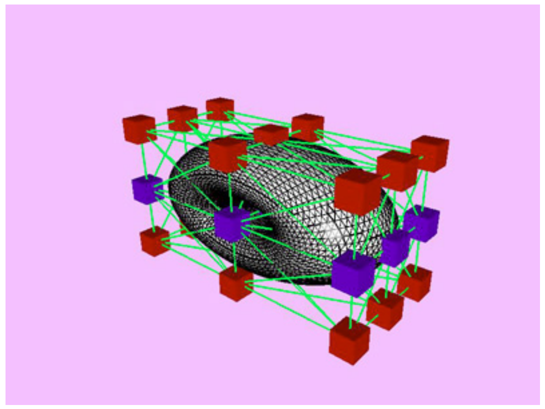
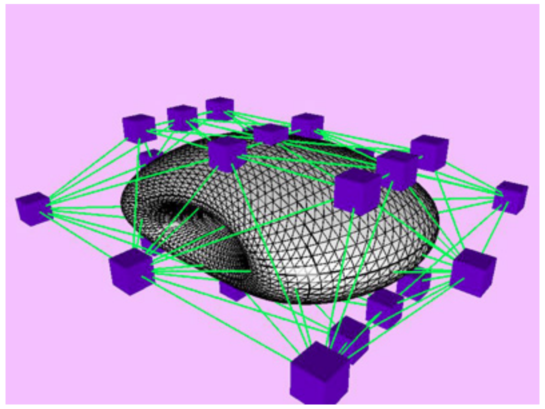
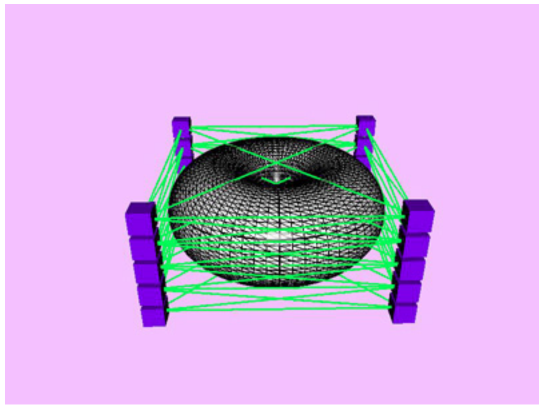
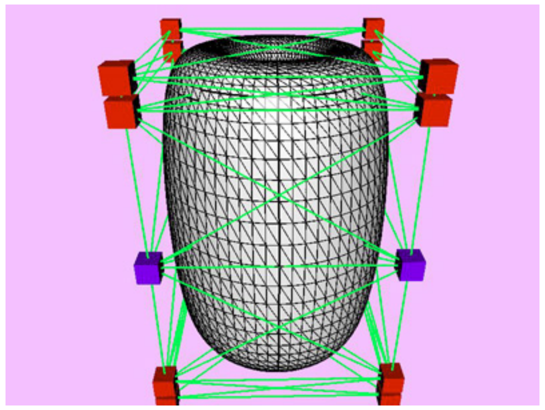
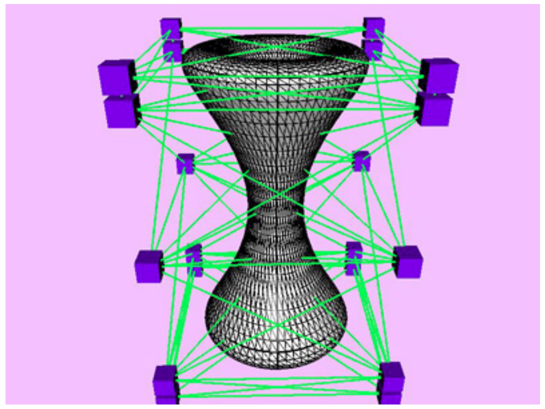
FFD Block

- Muliple hyperpatches connected together
- regular lattice of control points on 3 orthogonal UVW axes
- for an *lxmxn* block of hyperpatches
  have an array of $(3l+1)x(3m+1)x(3n+1)$ control points
- Enforce continuity between patches via control points
- Problems of 3D aliasing occur if surface resolution is
  similar to FFD resolution

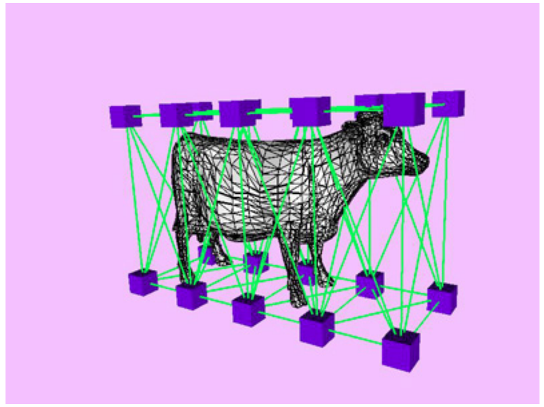
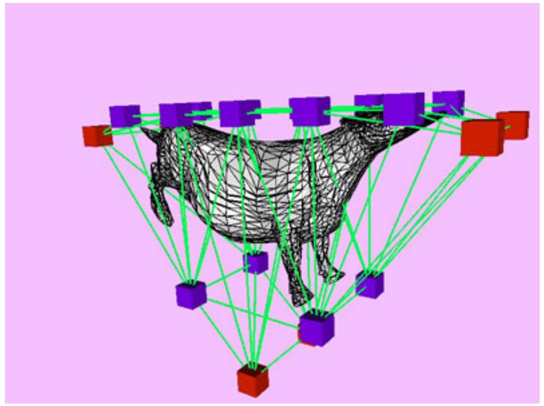
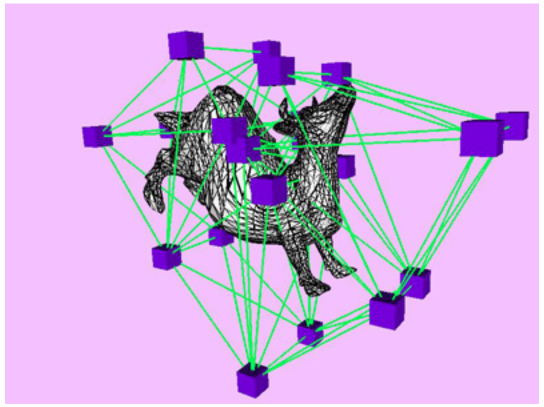
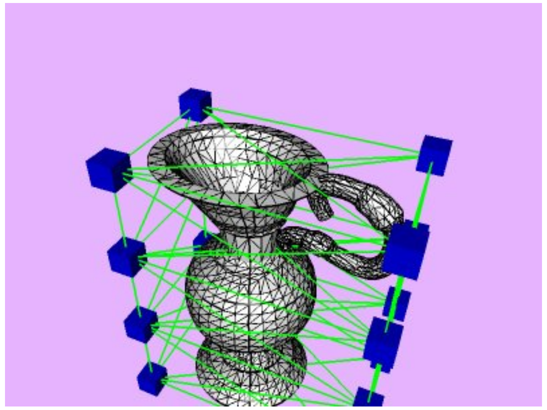
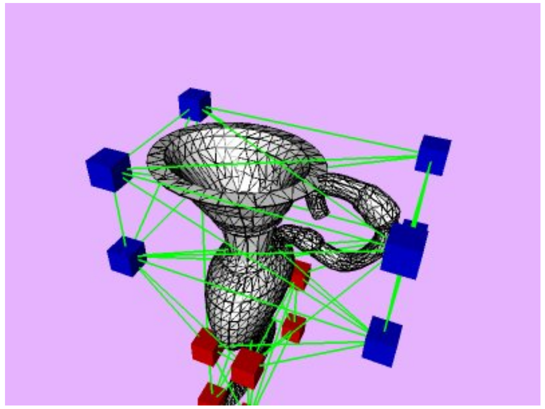
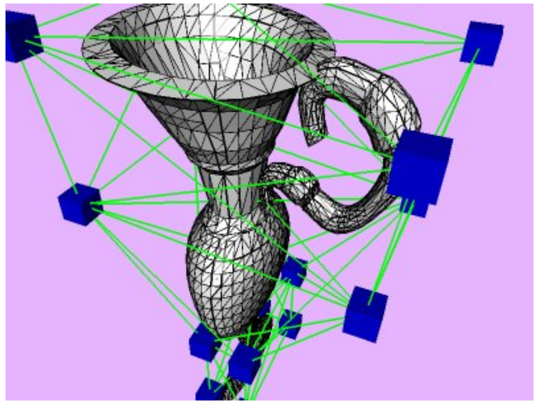
Extended FFD (EFFD)

- Generalises FFD to irregular latices
- FFD restricted to regular UVW grid
- EFFD extends to non - parallelapiped latices (FFD is a subset)
  Latices include : cylindrical, hexagonal
- control points of the hyperpatches are merged to give the
  required control lattice structure

Example FFD

(Volume Preserving FFD Horota ACM Solid Mod.'99)

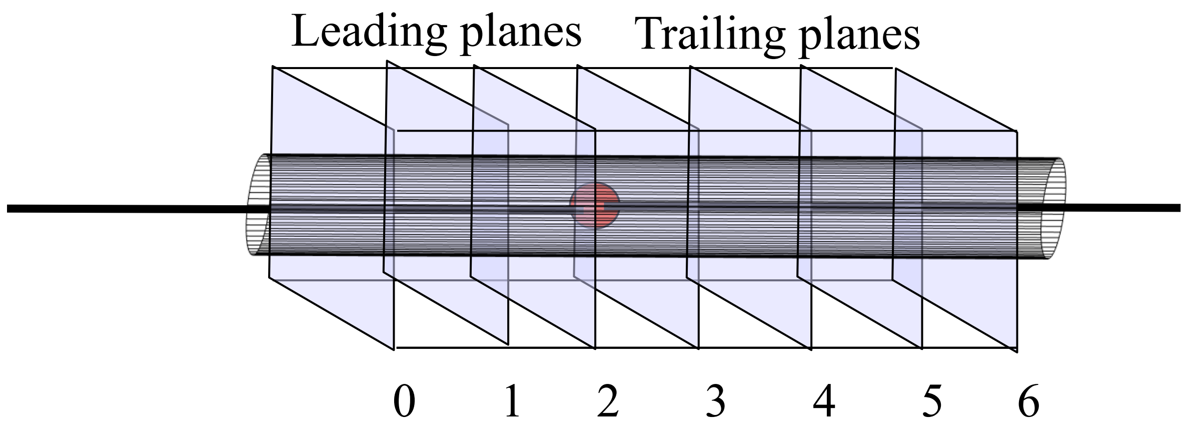







FFD for Deformation of Articulated Structures

Apply FFD to deform surface based on articulation

- Change FFD control point position based on articulation
- modify object surface based on FFD hyperpatch (Bezier)

Joint Based Deformation: use 2 overlapping FFD block one either side of joint



0,1 & 5,6 are adjoining planes - remain rigid to ensure continuity
2,3,4 midplanes - bend to give a smooth surface deformation

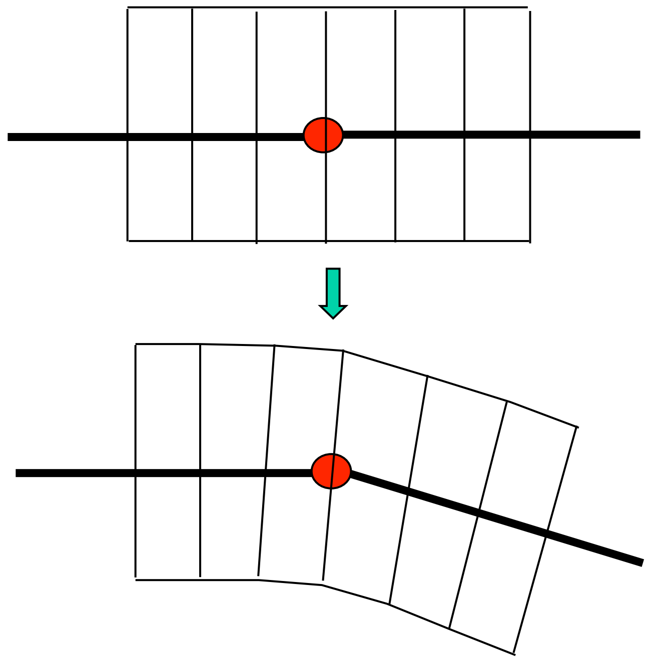
Layered model structure:

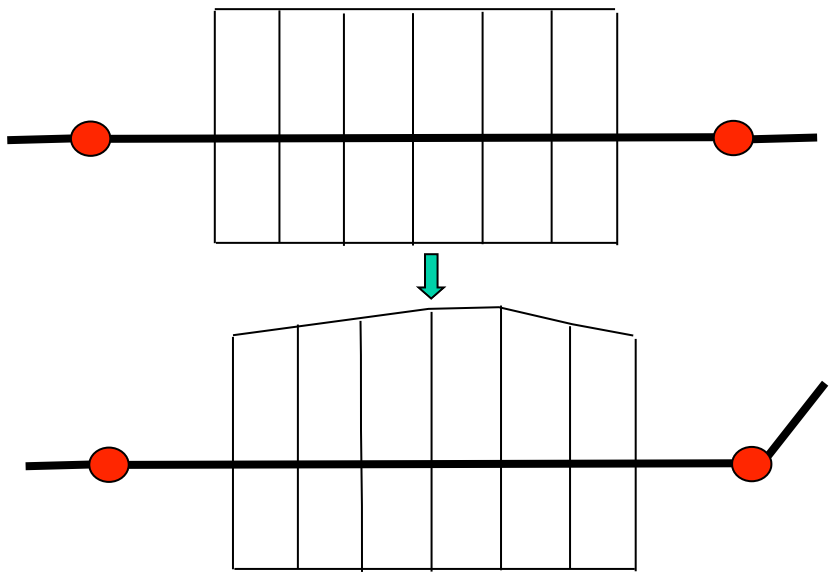
(1) Articulation structure (skeleton) controlled by joint angle parameters

(2) FFD layer (muscle) controlled by skeleton gives non-rigid deformation of volume surrounding the skeleton

(3) Surface (skin) polygonal mesh or parametric surface deformed by FFD layer

- Chadwick'89
- widely used for Soft Object Animation to simulate muscles/skin deformation
- animation of FFD layer controlled to preserve volume results in bulging





FFD Control Points must be animated non - rigidly to avoid intersection

For deformation around a joint

$$\text{let } \theta(\alpha) = \alpha \frac{\theta_j}{2}$$

$$\alpha = \frac{(u - u_{\max})}{(u_j - u_{\max})}$$

the angle decreases along the axis from $\frac{\theta_j}{2}$ at the joint u_j

to 0 at $u_{\max}$

Layered skeletal models using FFD to control surface deformation

- widely used for character animation
- realistic skin/muscle deformation
- smooth surface deformation based on hyperpatch (Bezier,B-Spline....)

Summary - Soft Object Animation

Control surface deformation
- skeletal animation
- want smooth deformation (no collapse/self-intersection)

Direct surface Deformation

Mesh
- vertex weighting
- morphing
- 3D aliasing + mesh collapse

Parametric surface
- control point animation gives new smooth surface

Space Deformation
- Global functions over entire space (twist,bend)
- Free-form Deformation
  - Hyperpatch (tricubic Bezier) gives smooth deformation
  - Layered animation change hyperpatch control points from skeleton