Computer Animation
Algorithms and Techniques

Interpolation-based animation

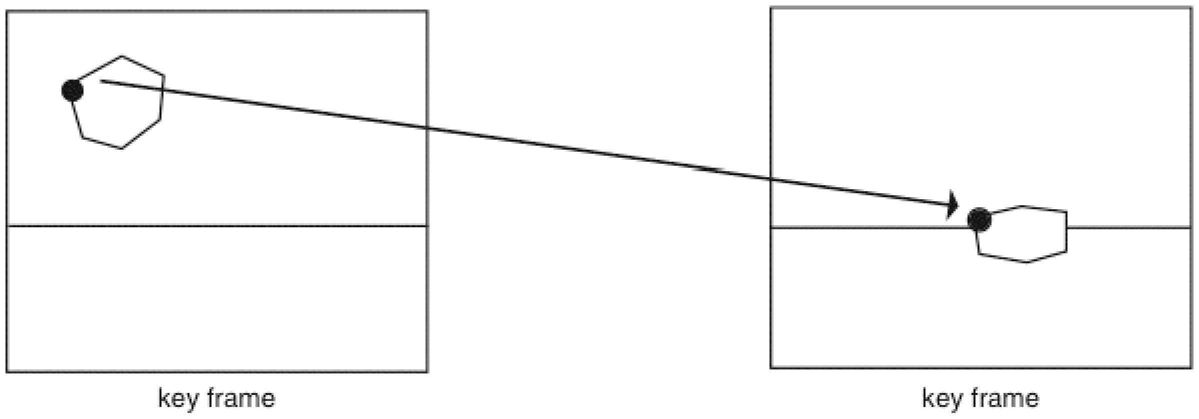
Interpolation based animation

Key-frame systems – in general

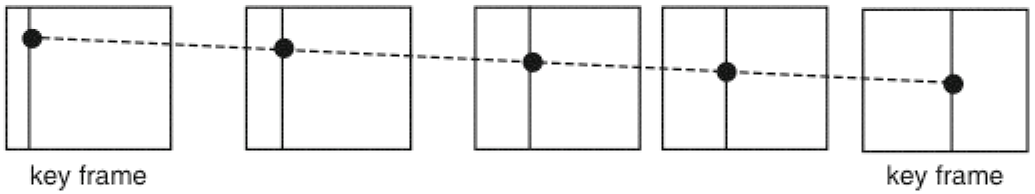
Interpolating shapes

- Deforming an single shape
- 3D interpolation between two shapes
- Morphing – deforming an image

Keyframing – interpolating values



Simple key frames in which each curve of a frame has the same number of points as its counterpart in the other frame



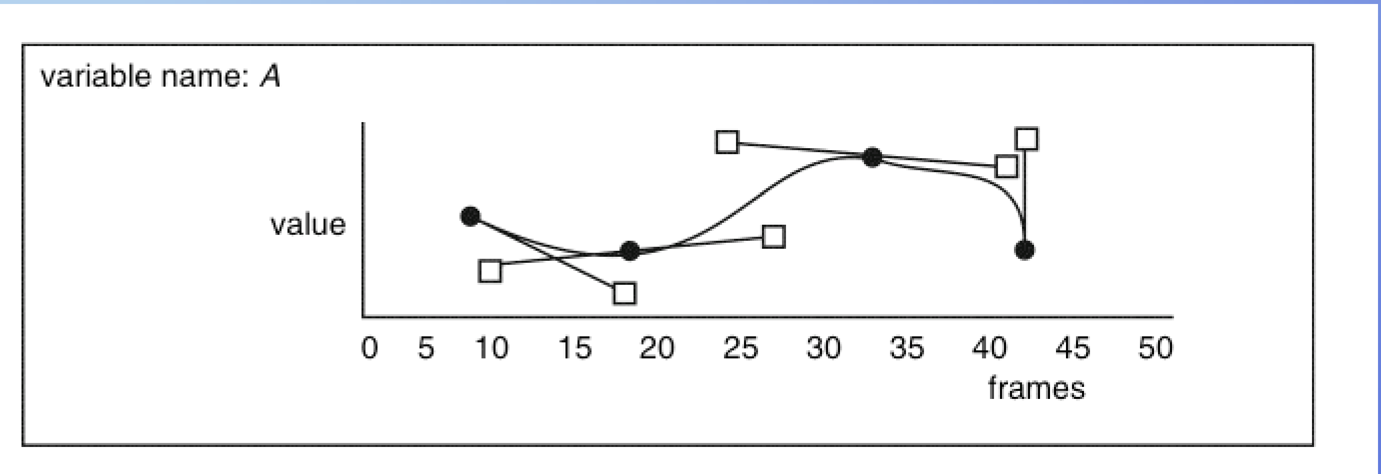
Keys and three intermediate frames with linear interpolation of a single point (with reference showing the progression of the interpolation in *x* and *y*)

Keyframing

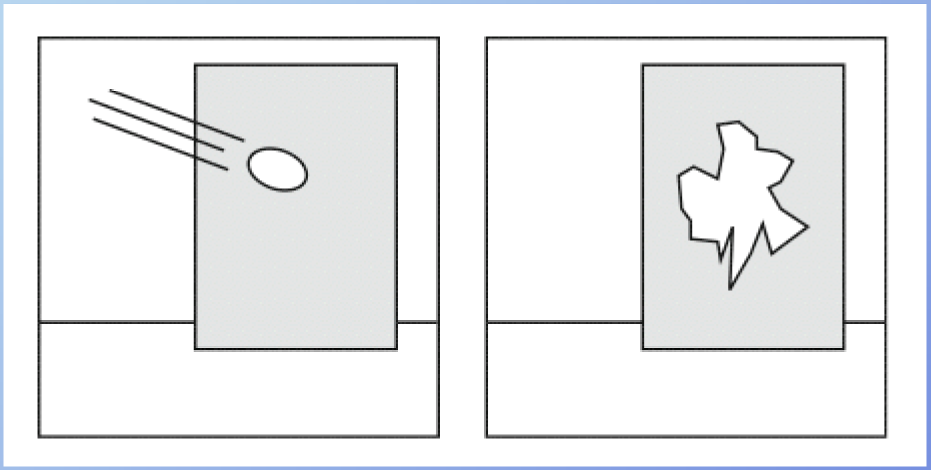
keys, in-betweens

track-based

Avars – articulation variables

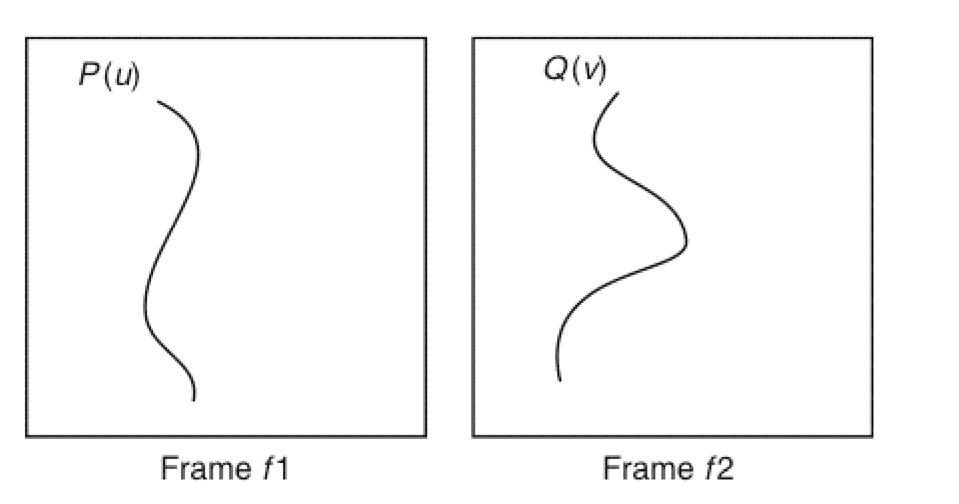


Keyframing curves



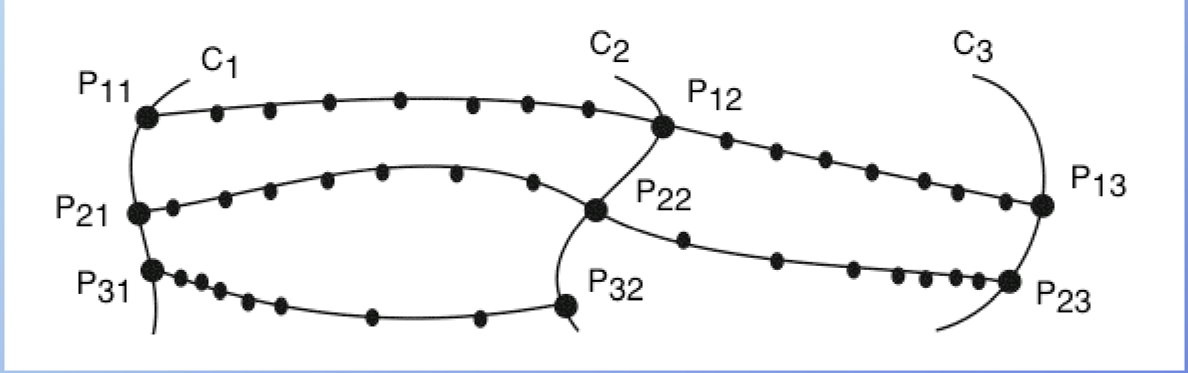
Time-Curve interpolation

Implement using surface patch technology



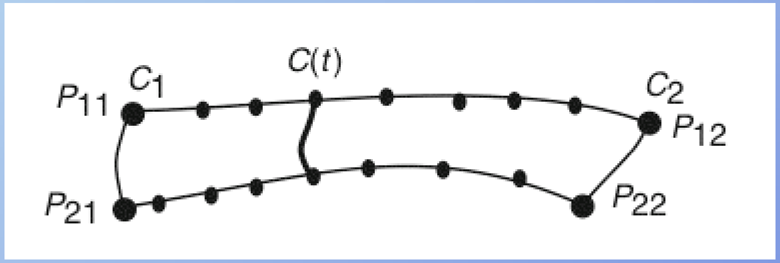
Time-Curve interpolation

Establish point correspondence



Time-Curve interpolation

Define time – space-curve “patches”



Interpolate in one dimension for curve (spatially)
Interpolate in other dimension temporally

Object interpolation

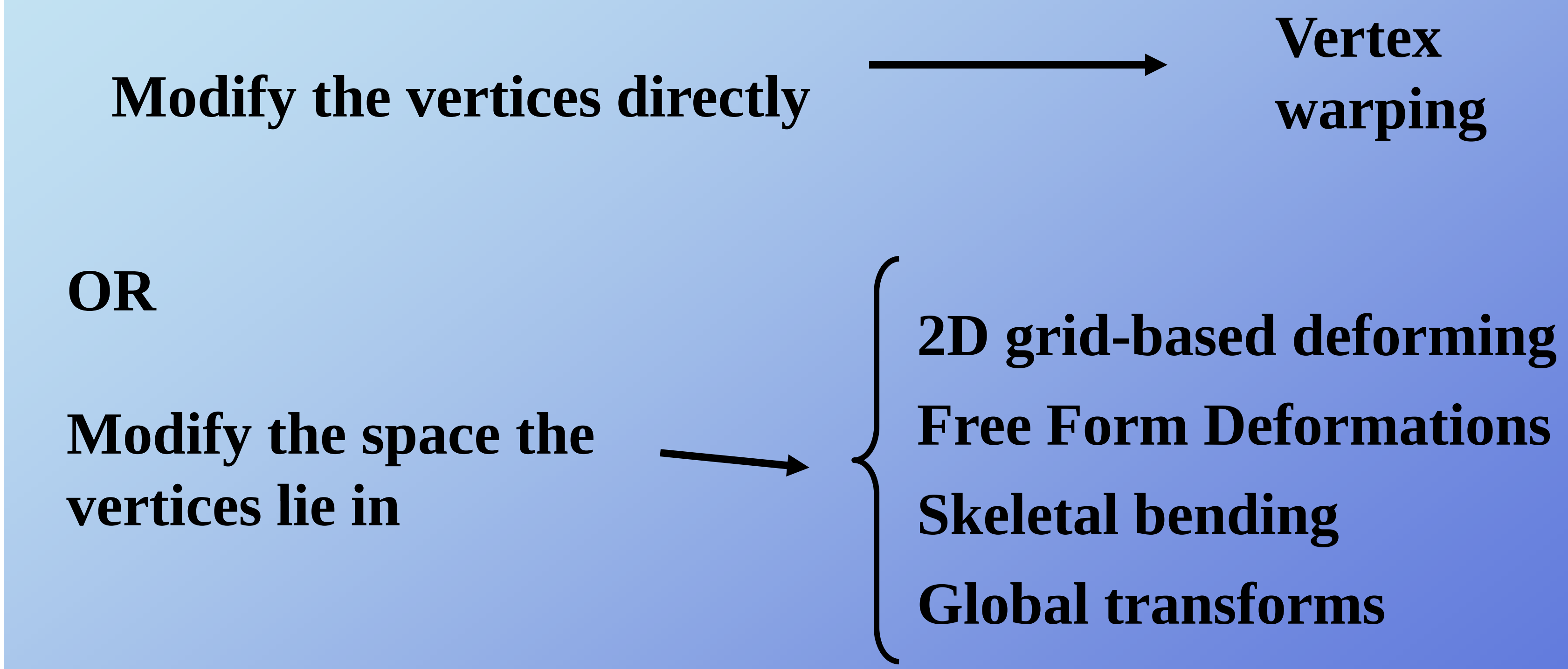
Correspondence problem
Interpolation problem

1. Modify shape of object interpolate vertices of different shapes

2. Interpolate one object into second object

3. Interpolate one image into second image

Object Modification



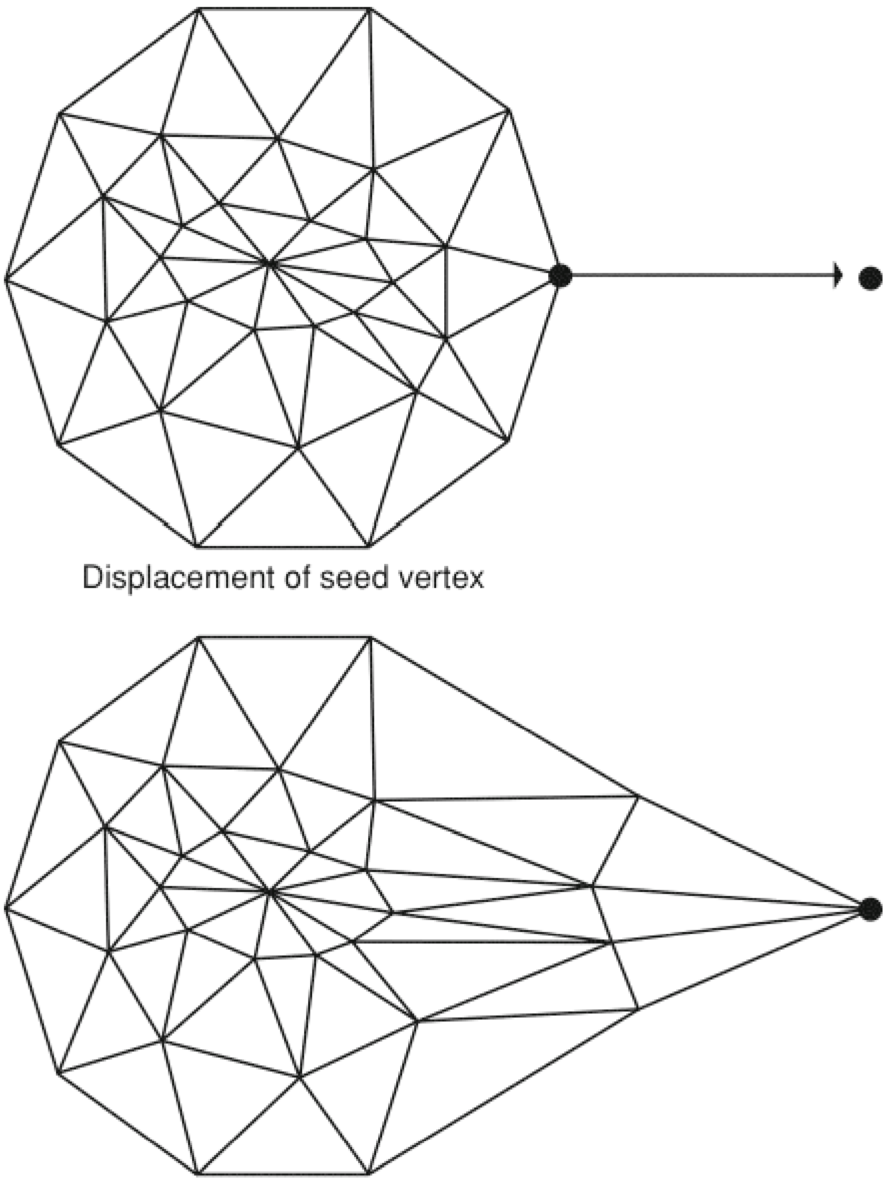
Modify the vertices directly → Vertex warping

OR

Modify the space the vertices lie in →

- 2D grid-based deforming
- Free Form Deformations
- Skeletal bending
- Global transforms

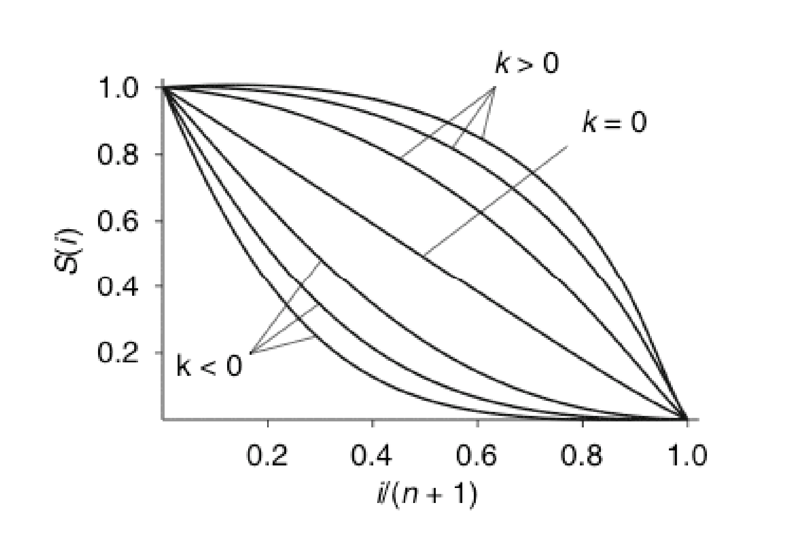
Warping



Displacement of seed vertex

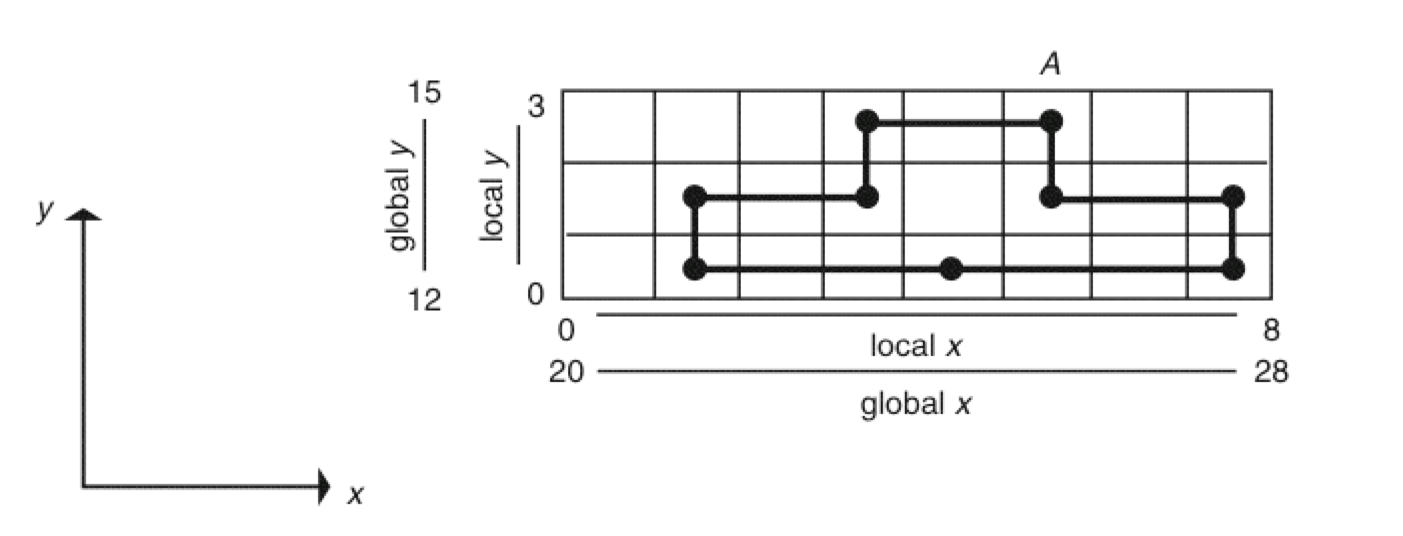
Attenuated displacement propagated to adjacent vertices

Power functions
For attenuating warping effects



$$S(i) = 1.0 - \left(\frac{i}{n+1}\right)^{k+1} \qquad k \geq 0$$
$$= \left(1.0 - \frac{i}{n+1}\right)^{-k+1} \qquad k < 0$$

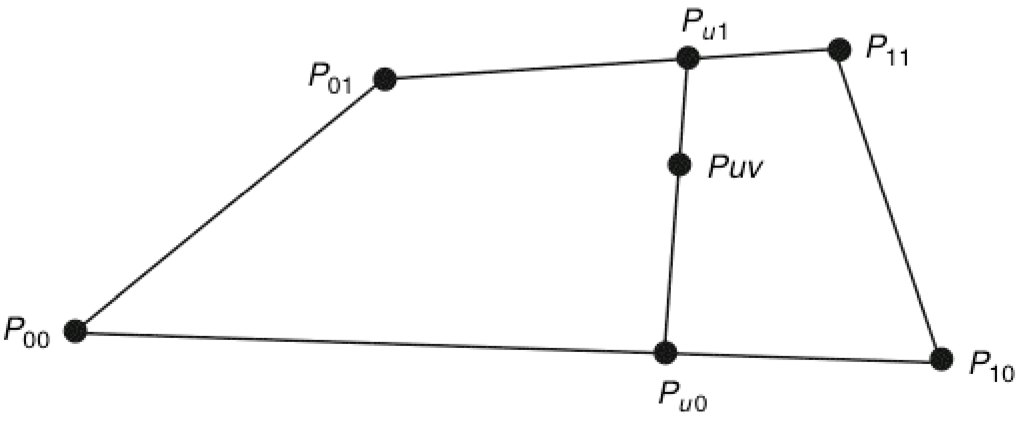
2D grid-based deforming



Assumption
Easier to deform grid points than object vertices

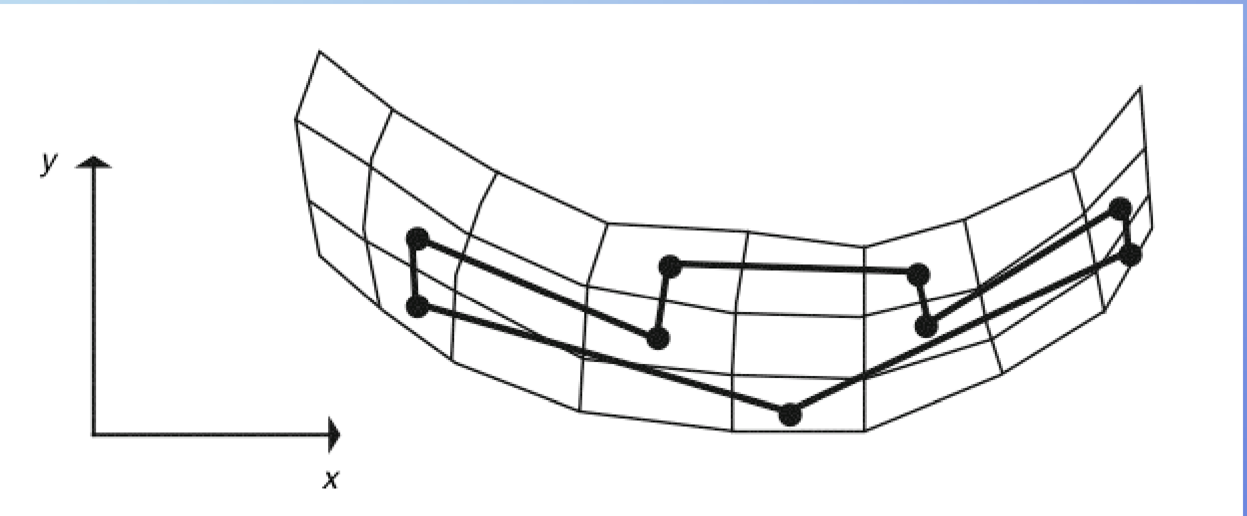
2D grid-based deforming

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

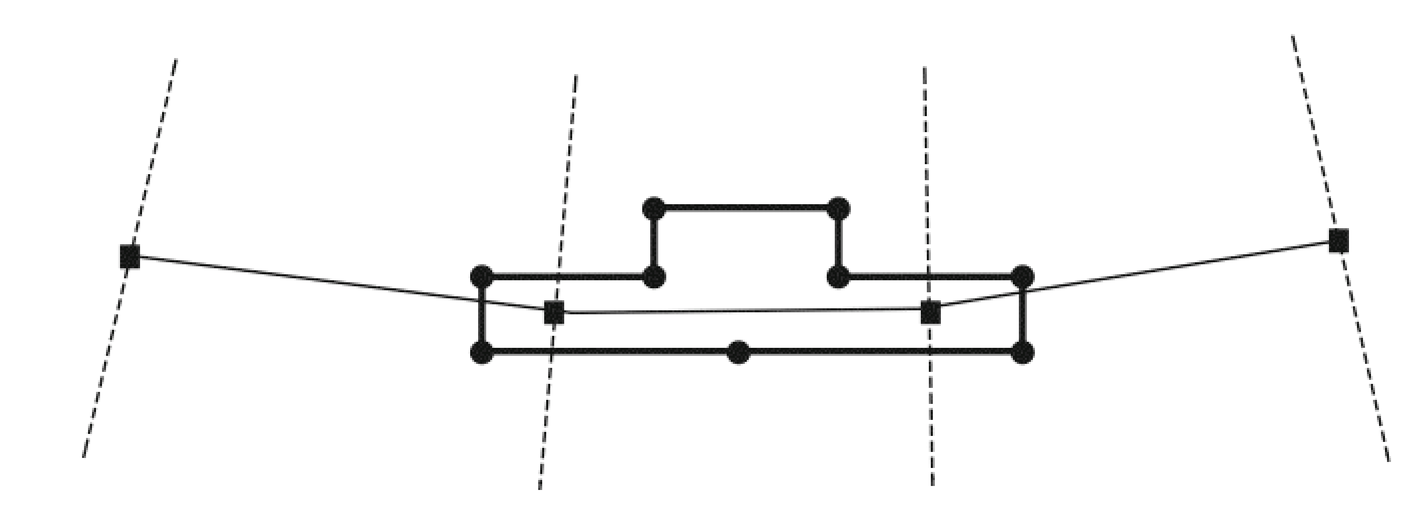


Inverse bilinear mapping (determine u,v from points)

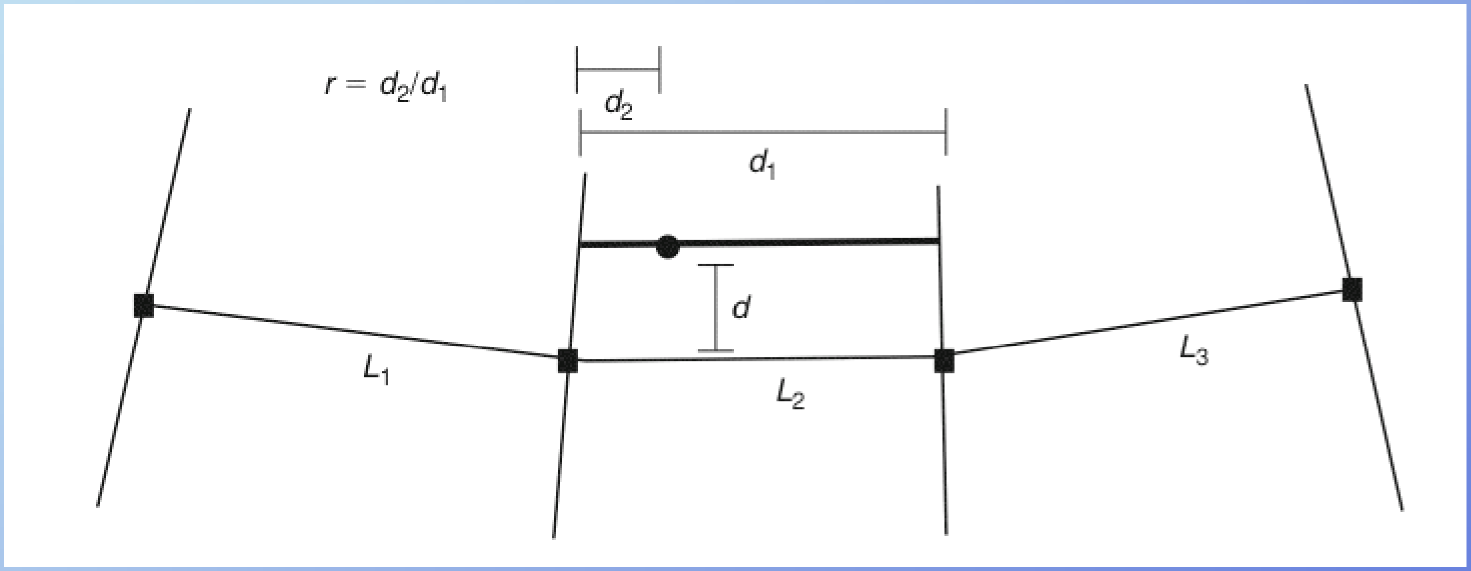
2D grid-based deforming



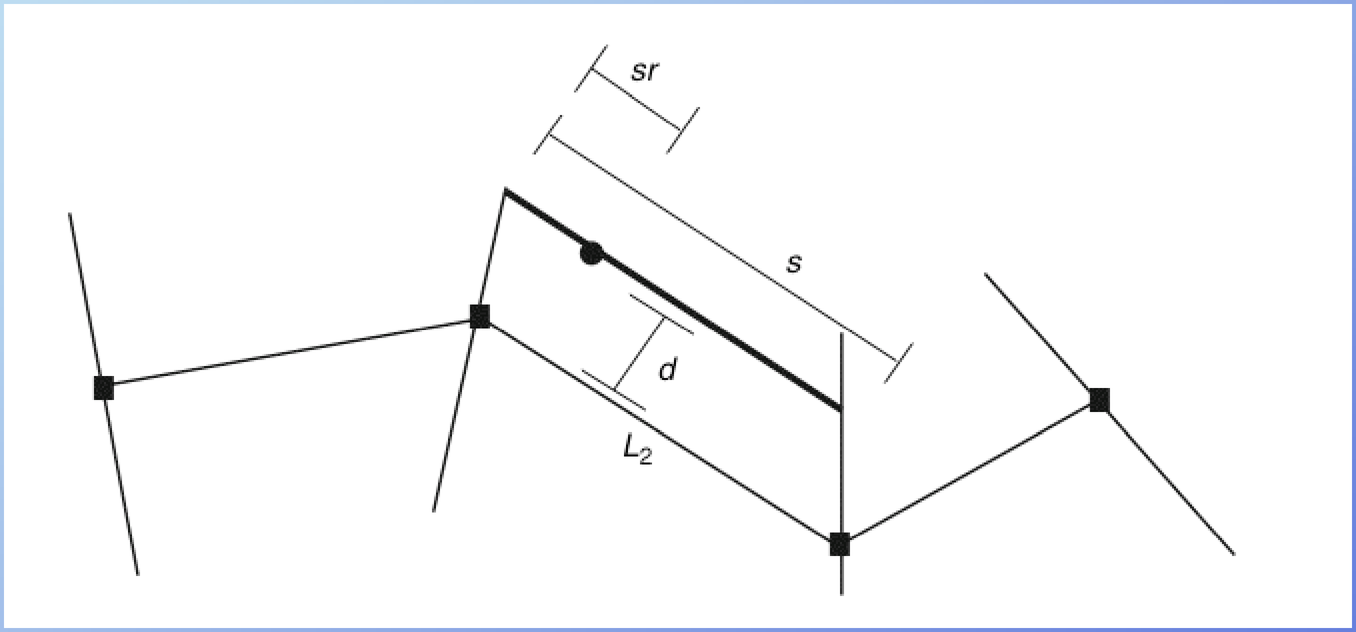
2D skeleton-based bending



2D skeleton-based bending



2D skeleton-based bending



Global Transformations

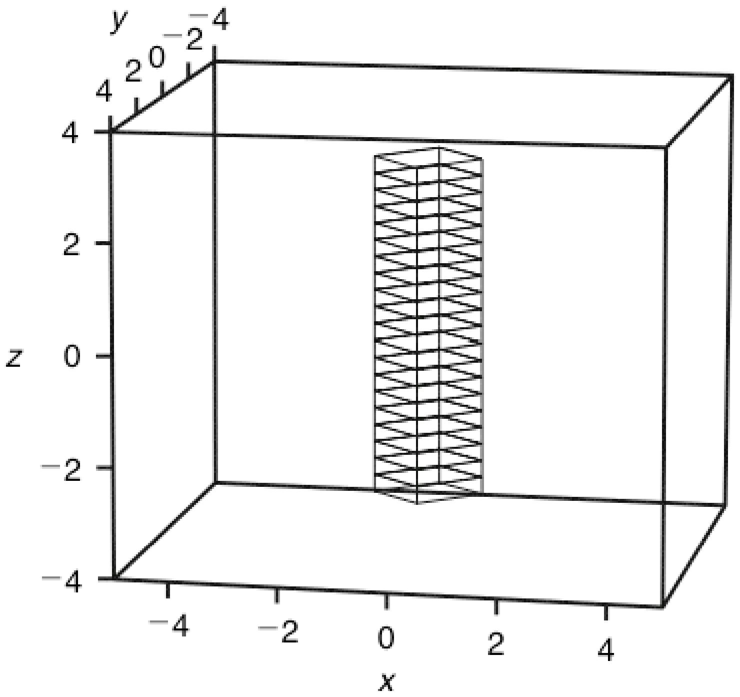
Common linear transform of space

$$p' = Mp$$

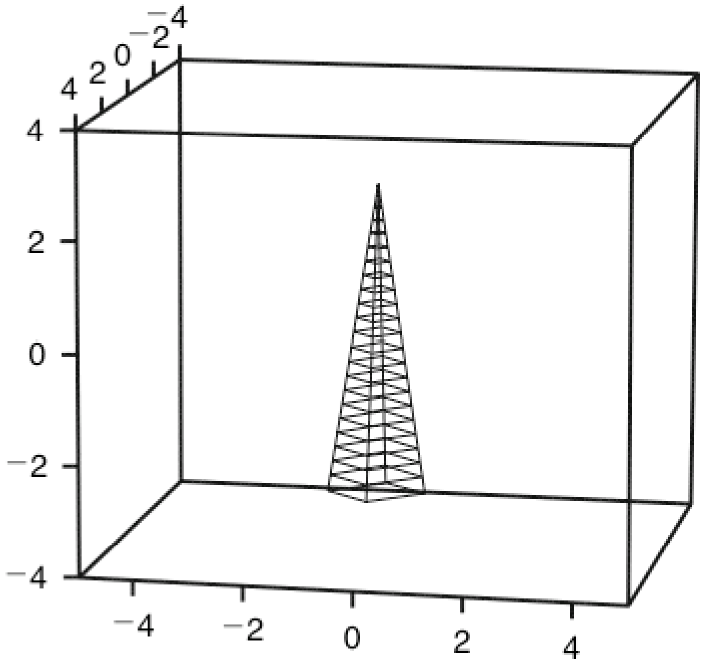
In GT, Transform is a function of where you are in space

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

Global Transformations



Original object



Tapered object

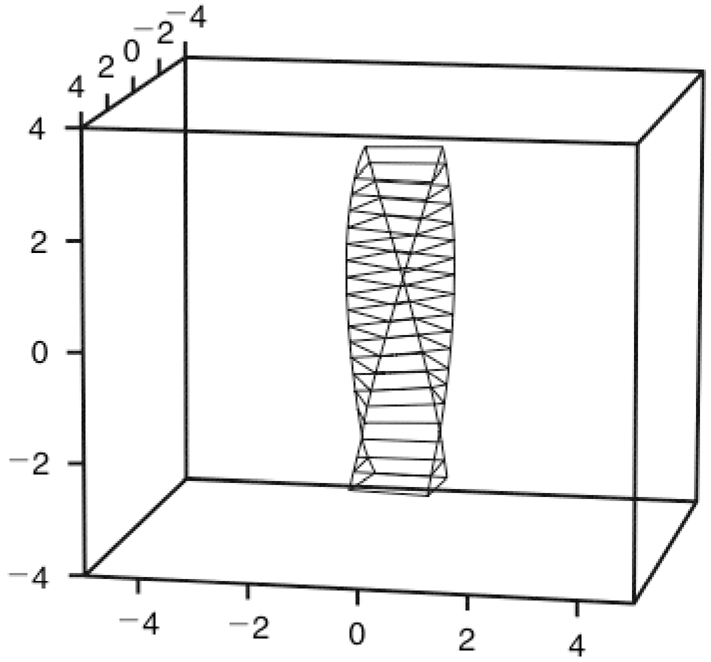
$$s(z) = \frac{(maxz - z)}{(maxz - minz)}$$

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

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

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

Global Transformations



k = twist factor

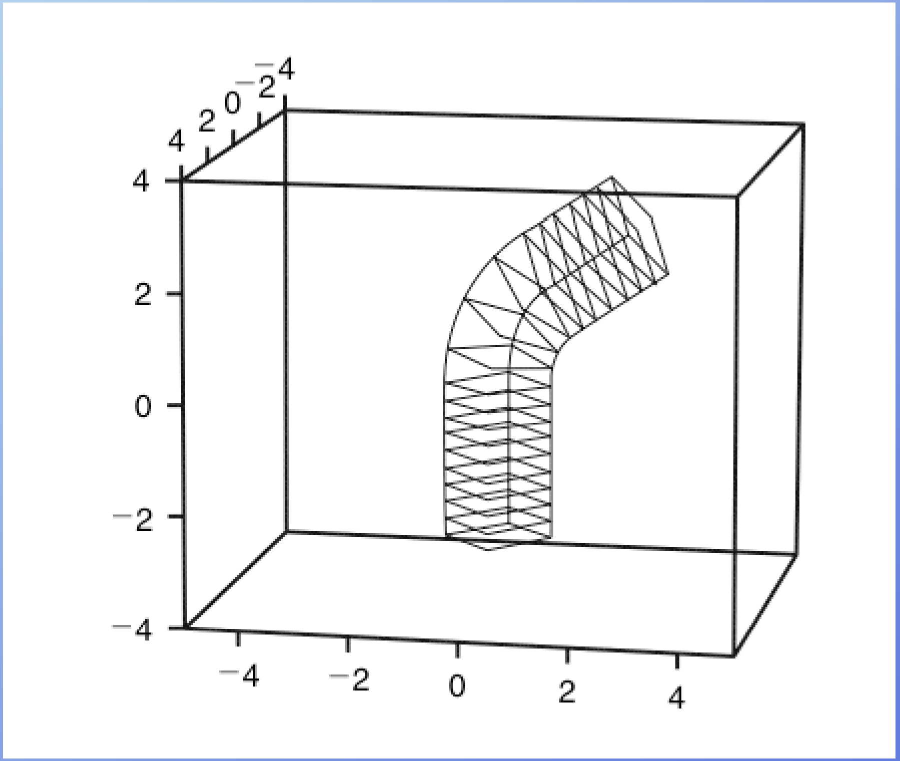
$$
\begin{aligned}
x' &= x\cos(kz) - y\sin(kz)\\
y' &= x\sin(kz) + y\cos(kz)\\
z' &= z
\end{aligned}
$$

Global Transformations

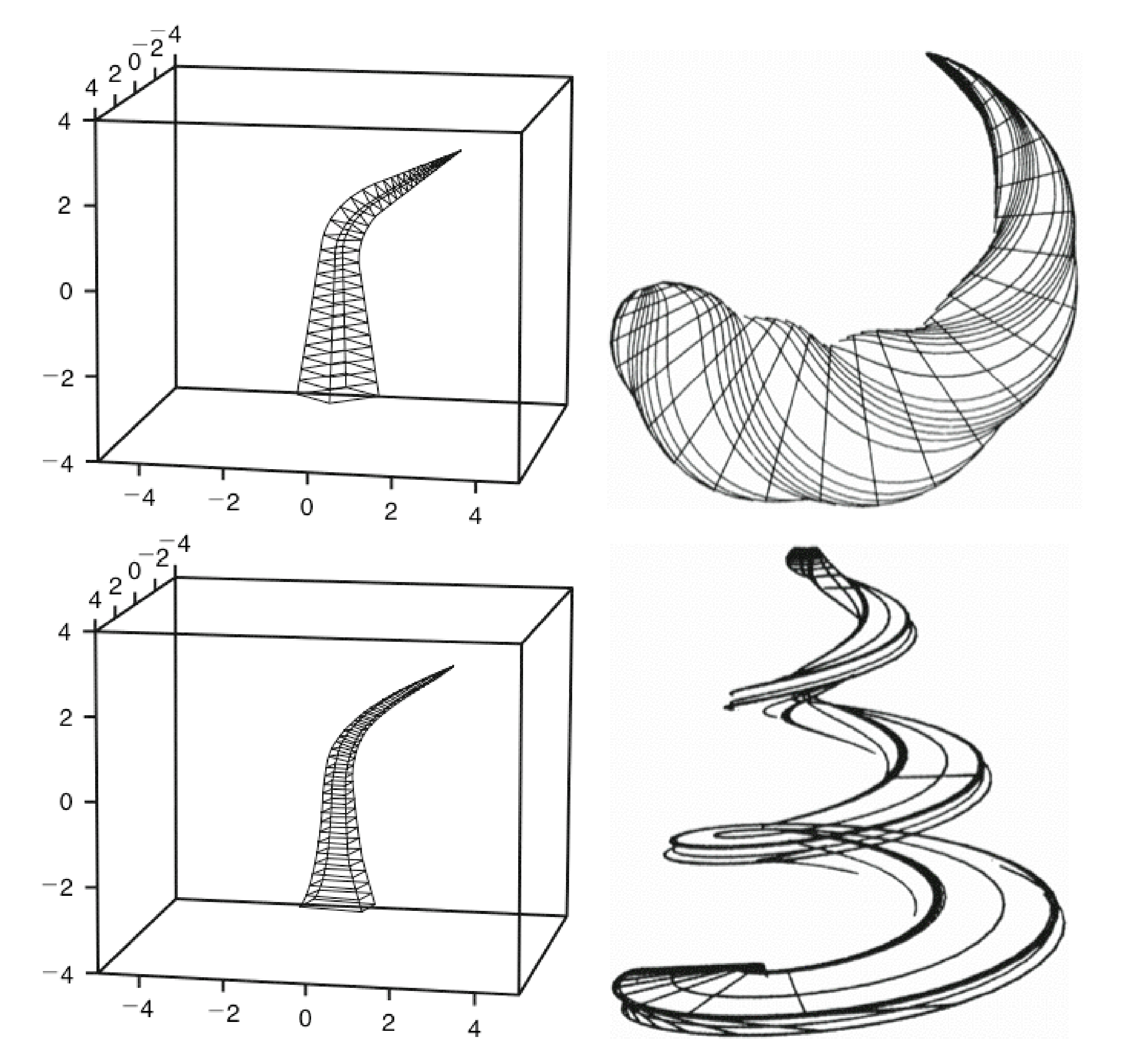
z above z_{min}: rotate θ

z between z_{min} ,z_{ma}x :
Rotate from 0 to θ

z below z_{min}: no rotation



Compound global transformations

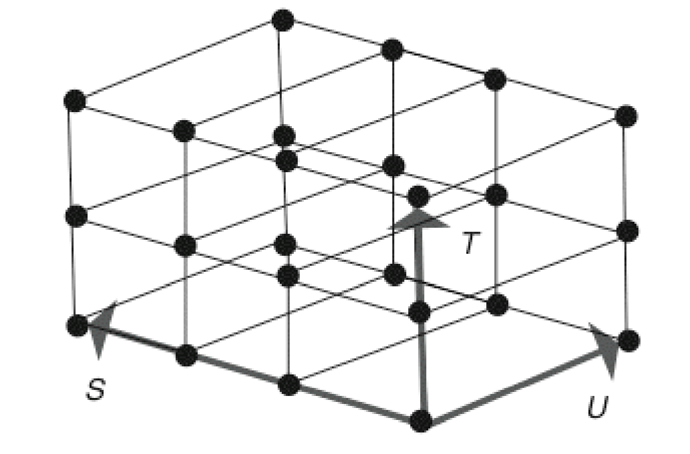


Free-Form Deformations (FFDs)

2D grid-based deforming	**FFDs**
2D grid	3D grid
bi-linear interpolation	tri-cubic interpolation

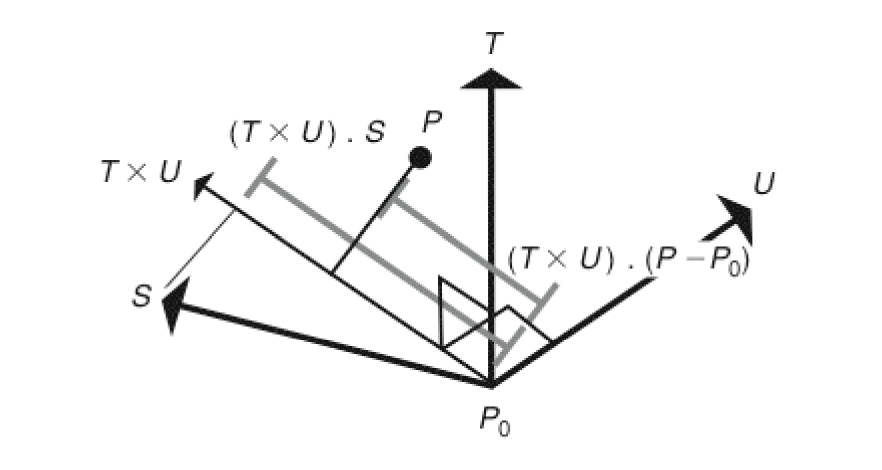
Free-Form Deformations

Embed object in rectilinear grid



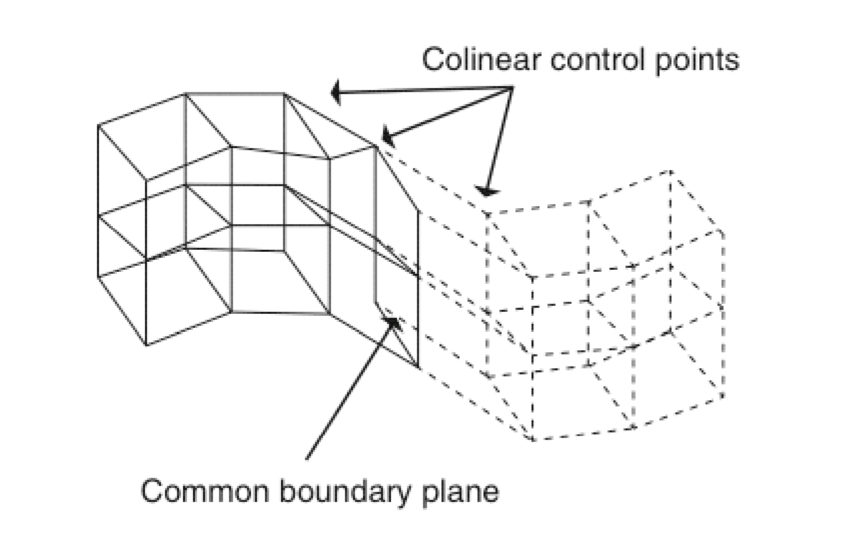
Free-Form Deformations

Register points in grid: cell x,y,z; (s,t,u)

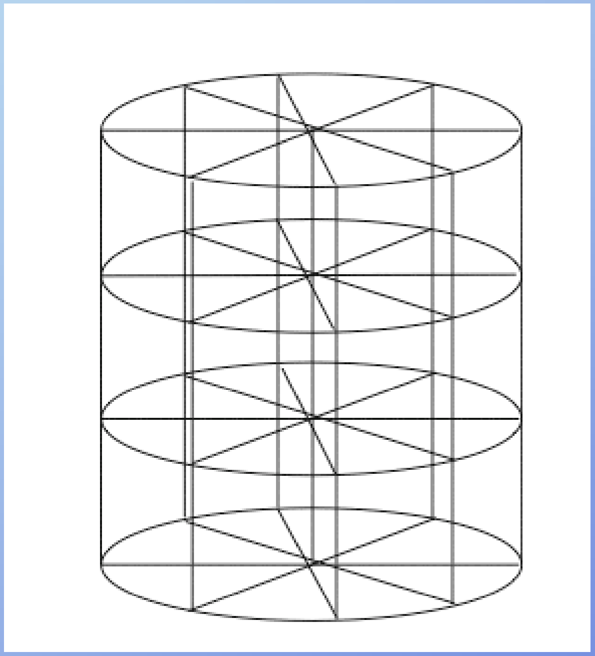


Free-Form Deformations

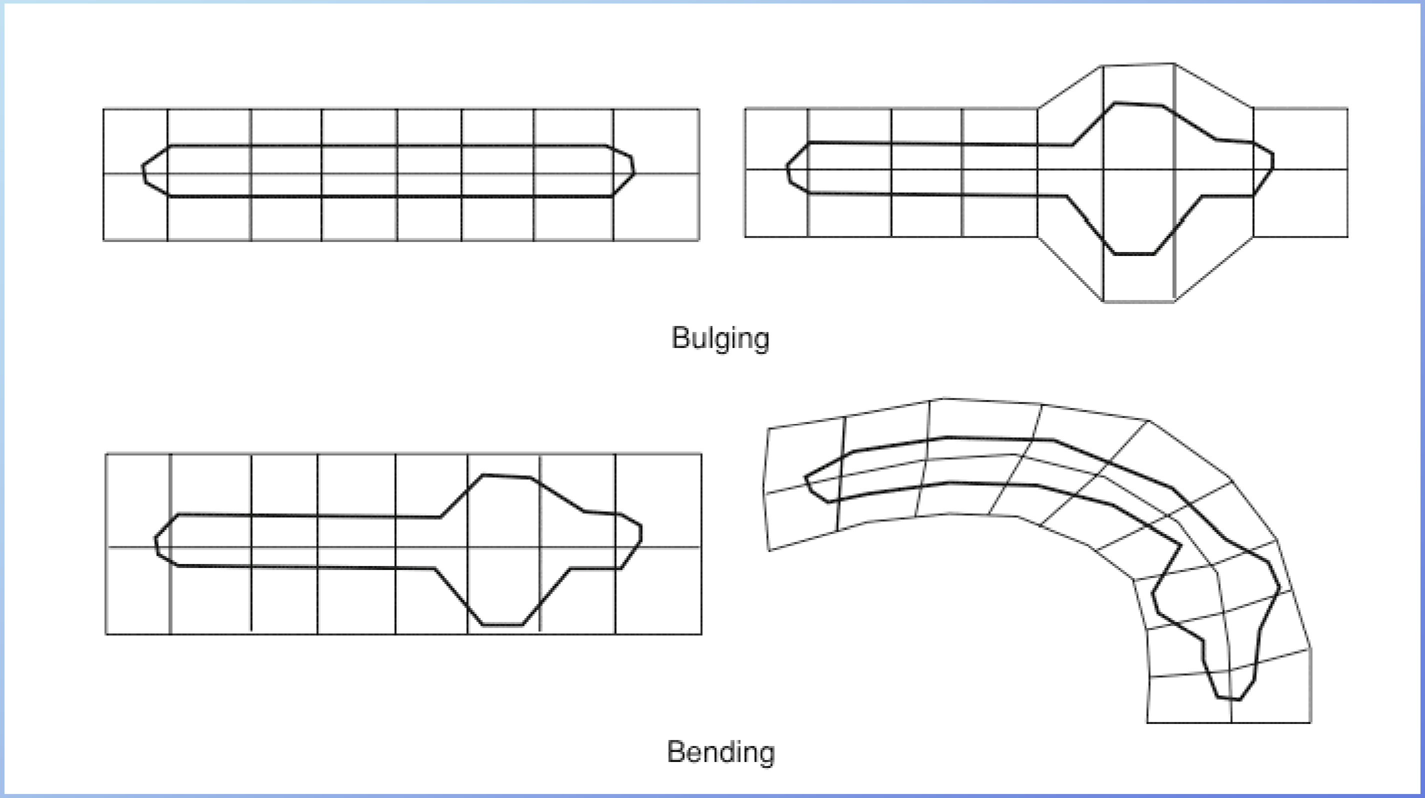
As in Bezier curve interpolation
Continuity controlled by coplanarity of control points



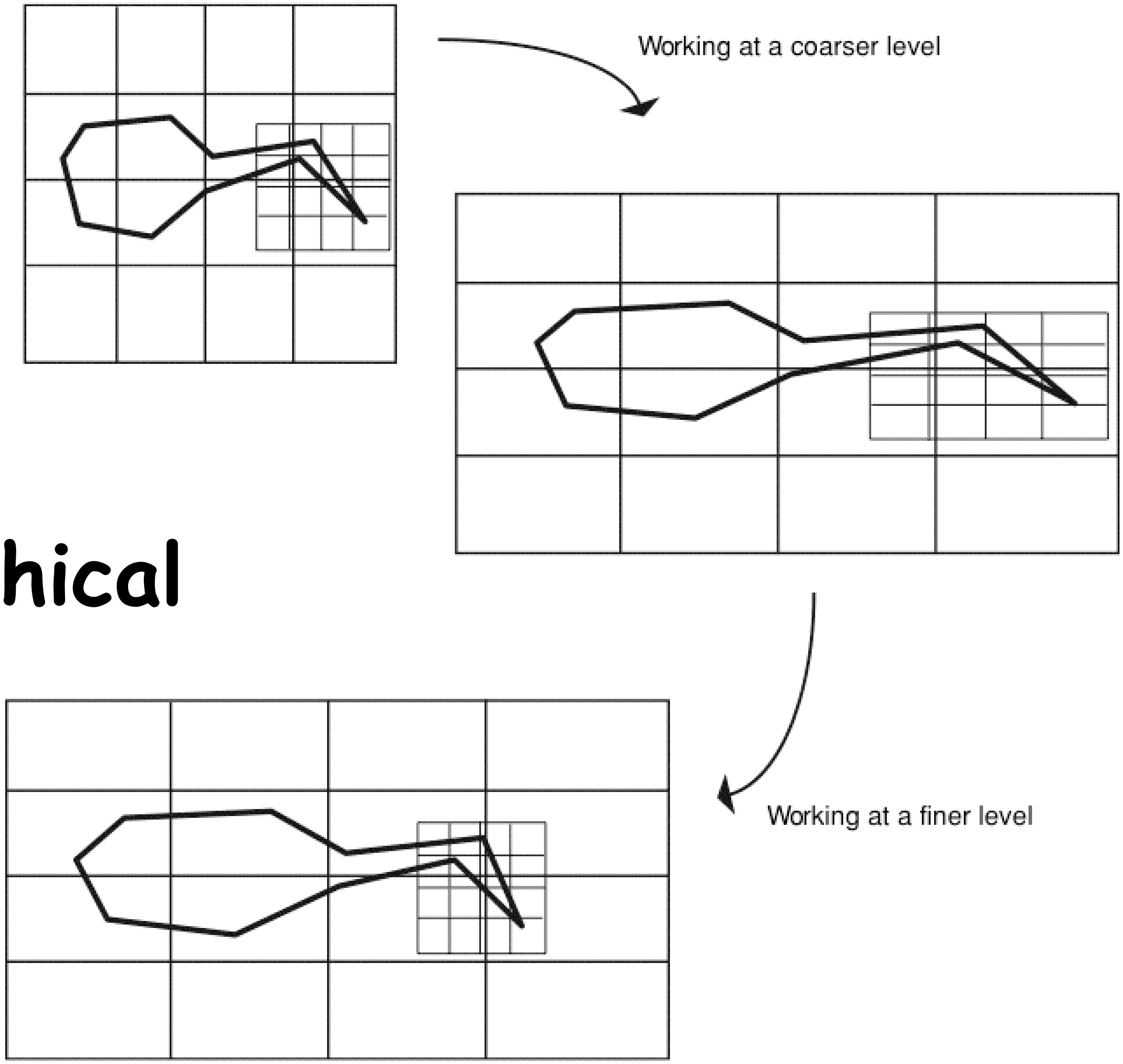
FFDs: alternate grid organizations



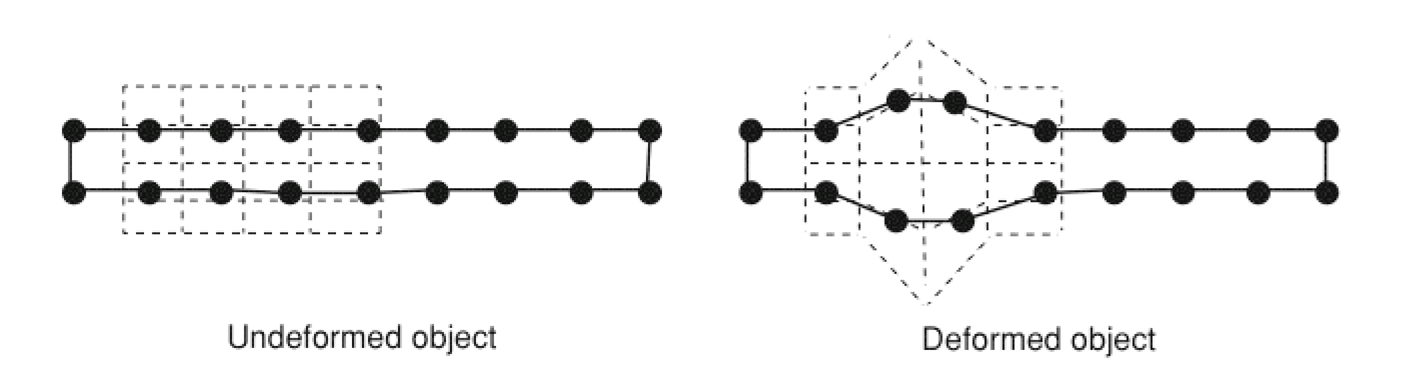
FFDs: bending



FFDs hierarchical

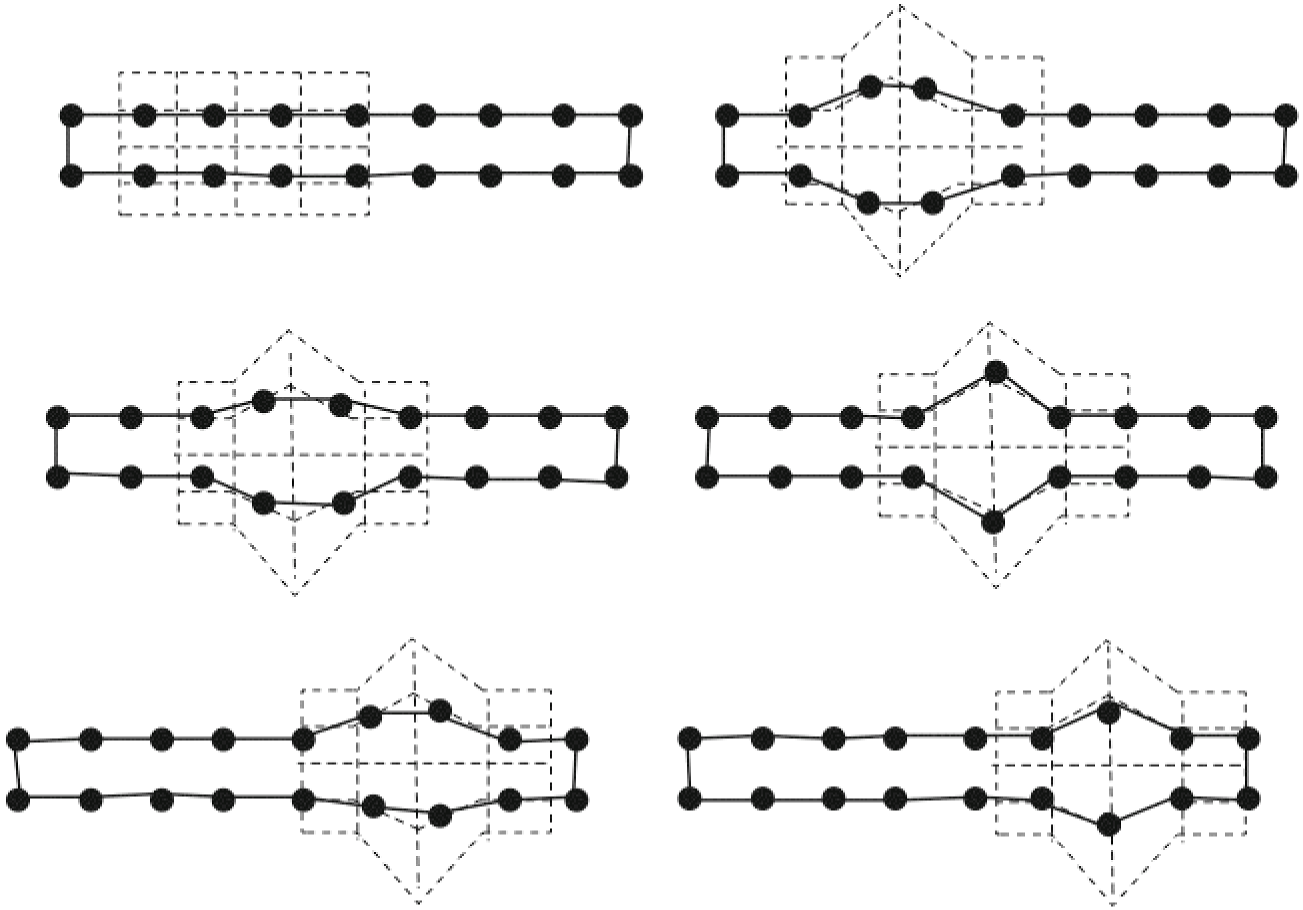


FFDs - as tools to design shapes



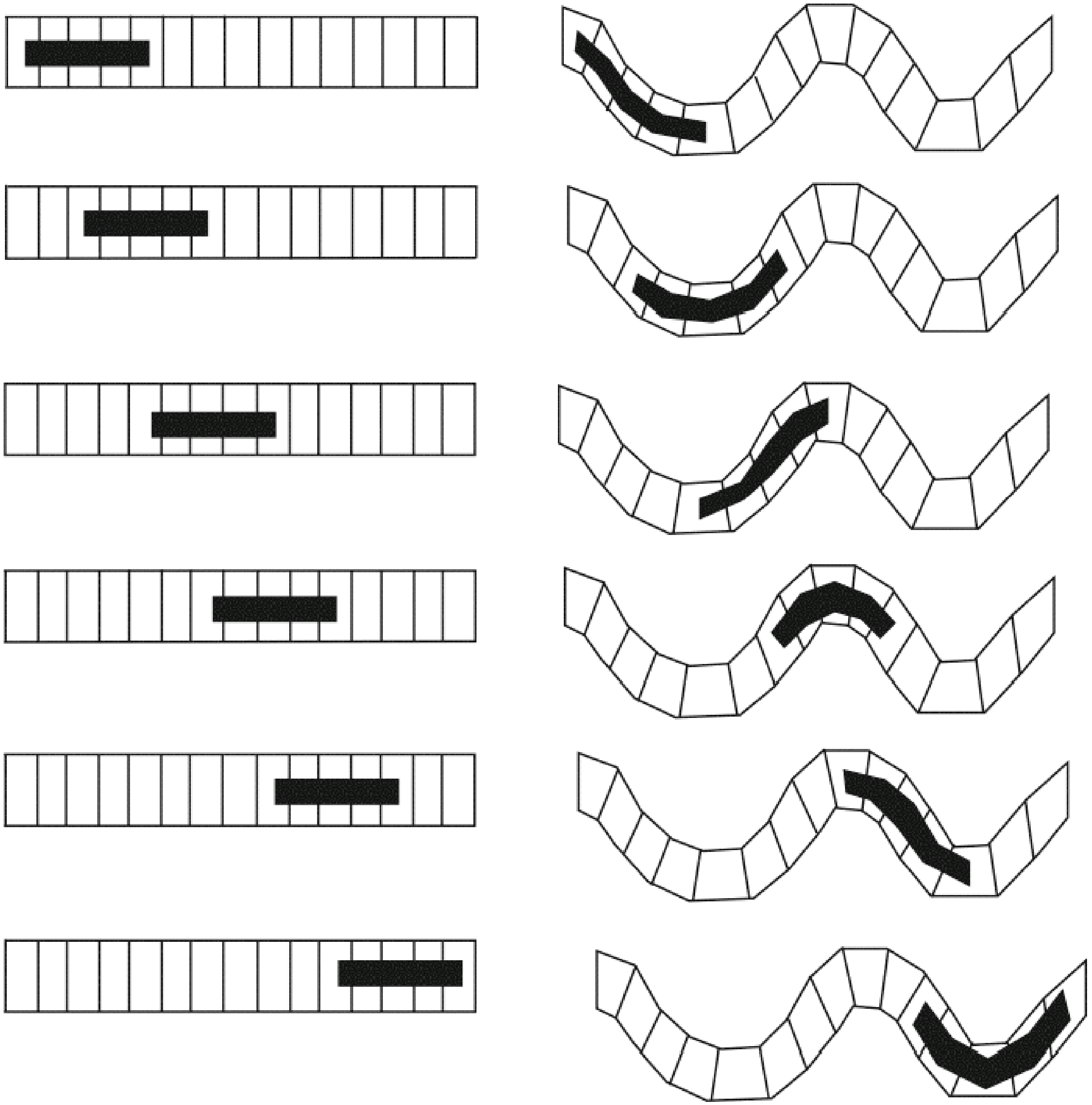
FFDs

Animate by passing over object



FFDs

Animate by passing object through FFD

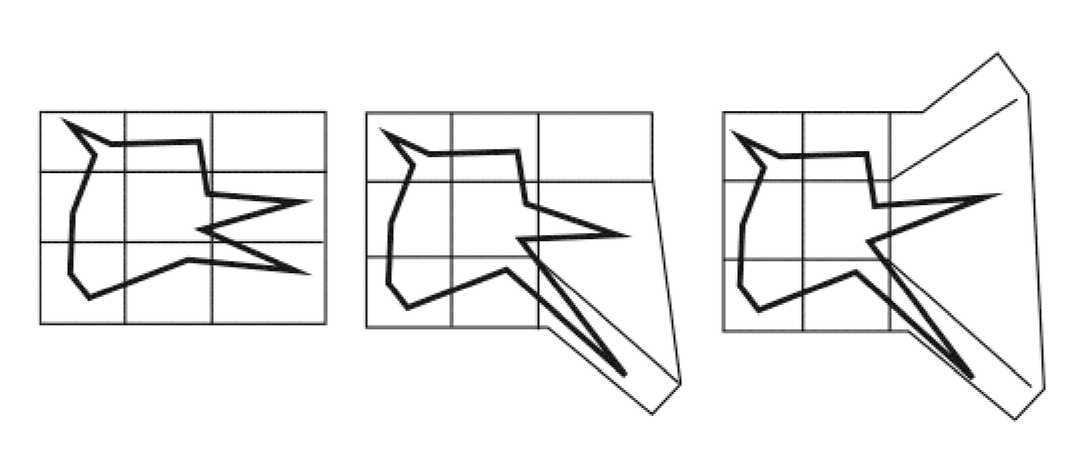


Object traversing the logical FFD coordinate space

Object traversing the distorted space

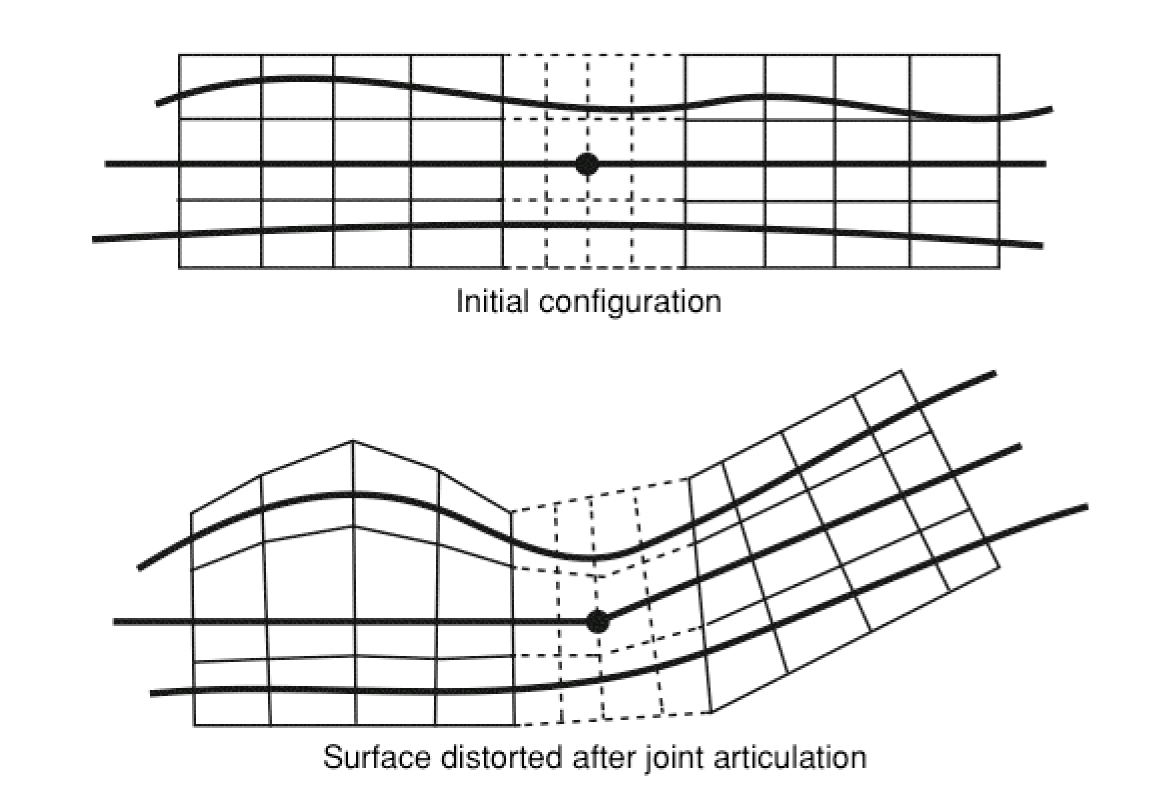
FFDs

Facial animation by manipulating FFD



FFDs

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



Interpolate between 2 objects

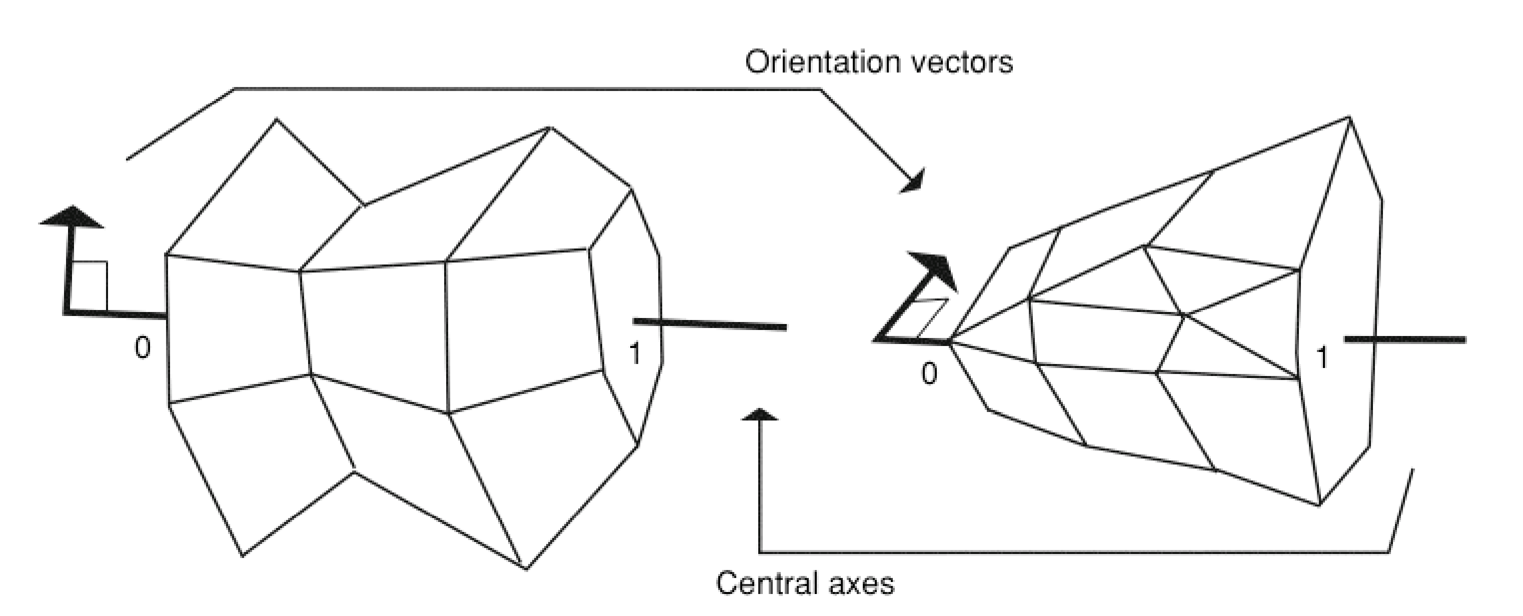
Correspondence problem: what part of one object to map into what part of the other object

How to handle objects of different genus?
Volumetric approaches with remeshing

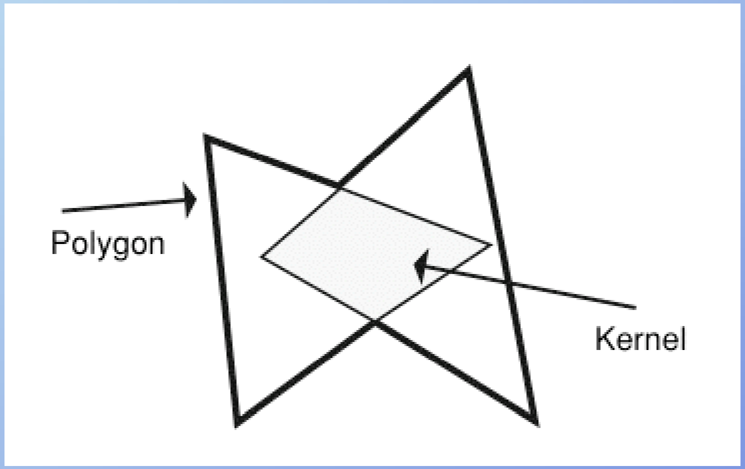
<u>Some surface-based approaches</u>
Slice along one dimension; interpolate in other two
Map both to sphere
Recursively divide into panels

Object interpolation

For cylinder-like objects

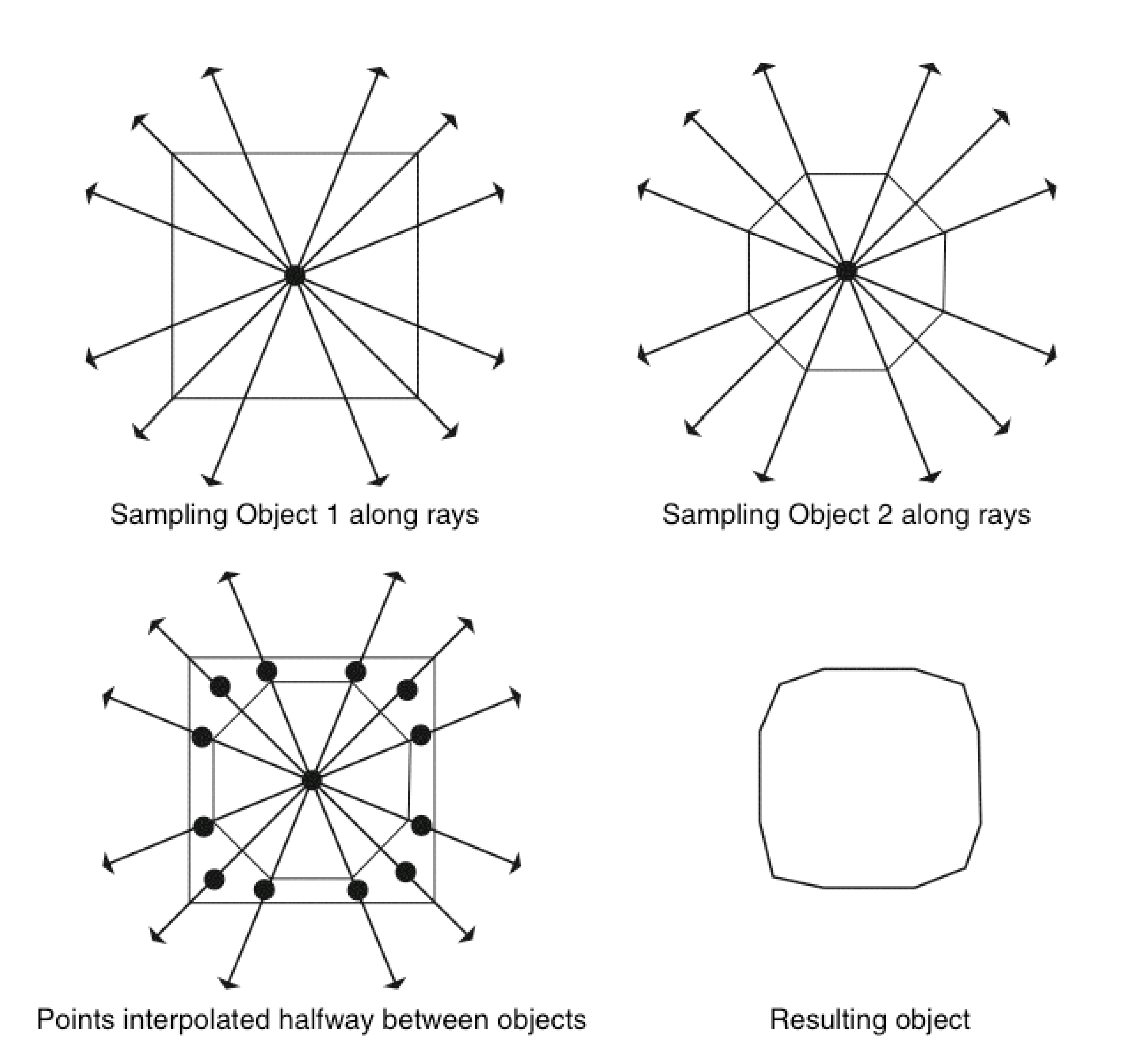


Radial mapping



**If central axis intersects polygonal slice inside kernel
Then simple radial mapping possible**

Object interp



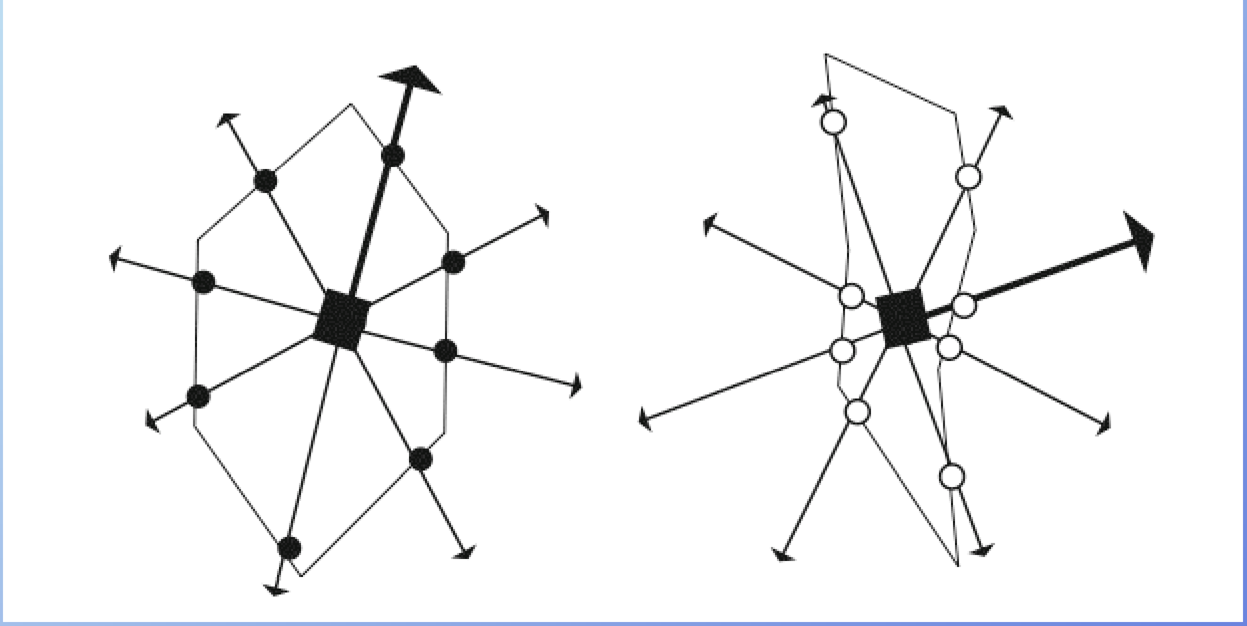
Sampling Object 1 along rays

Sampling Object 2 along rays

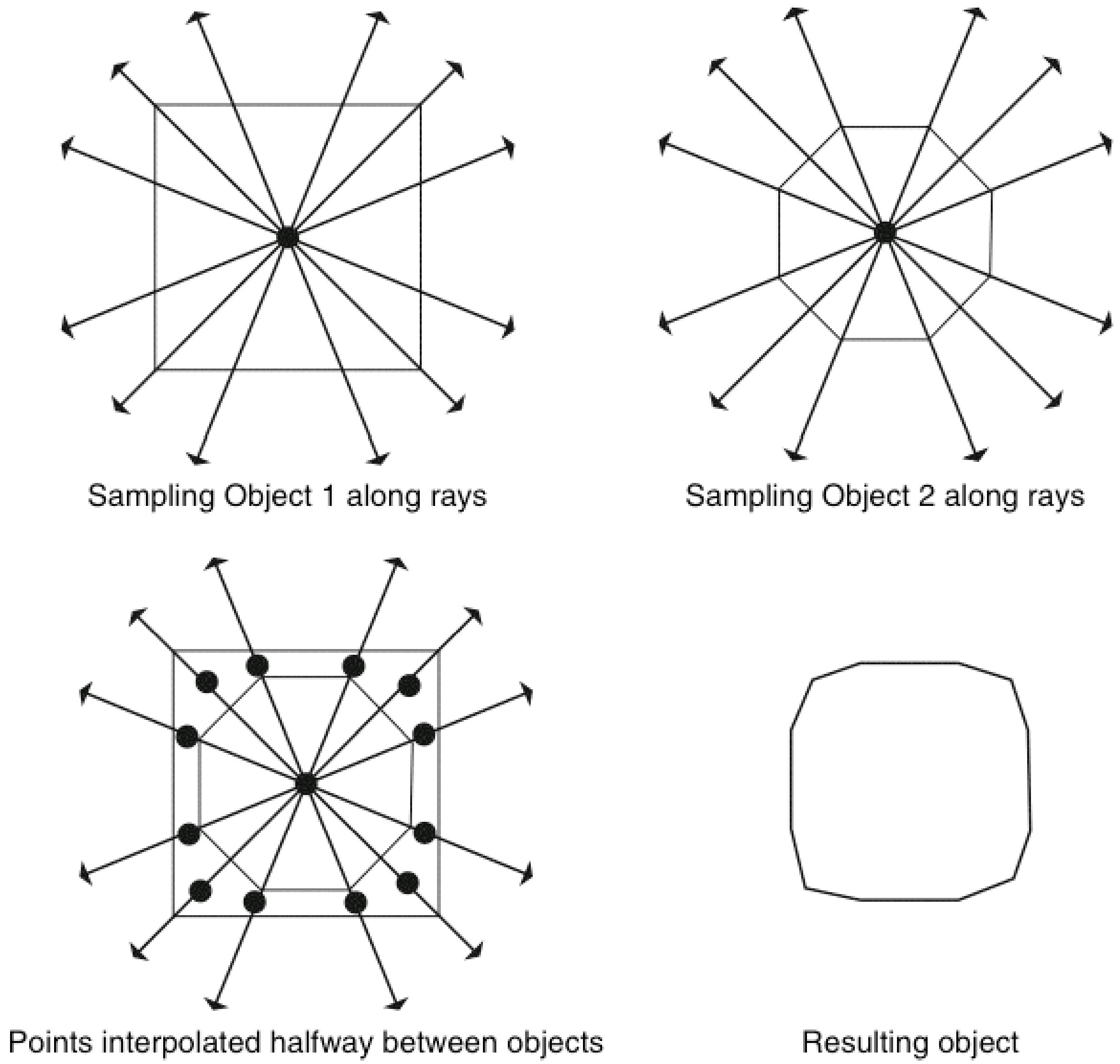
Points interpolated halfway between objects

Resulting object

Object interpolation



Object interp

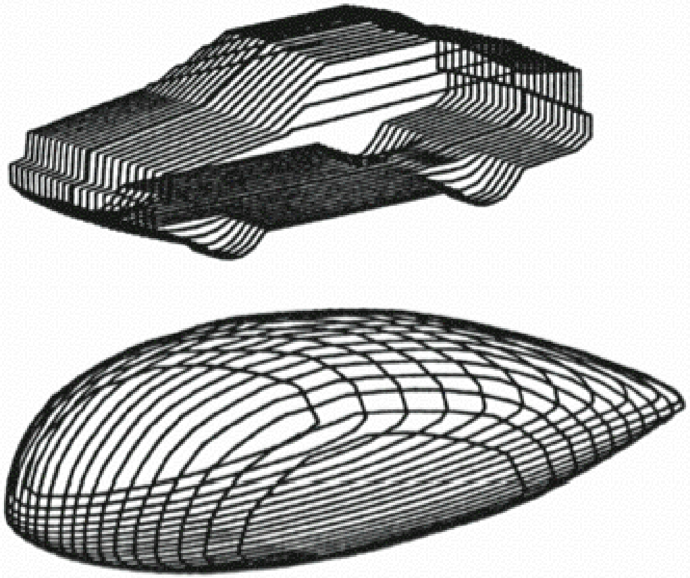


Sampling Object 1 along rays

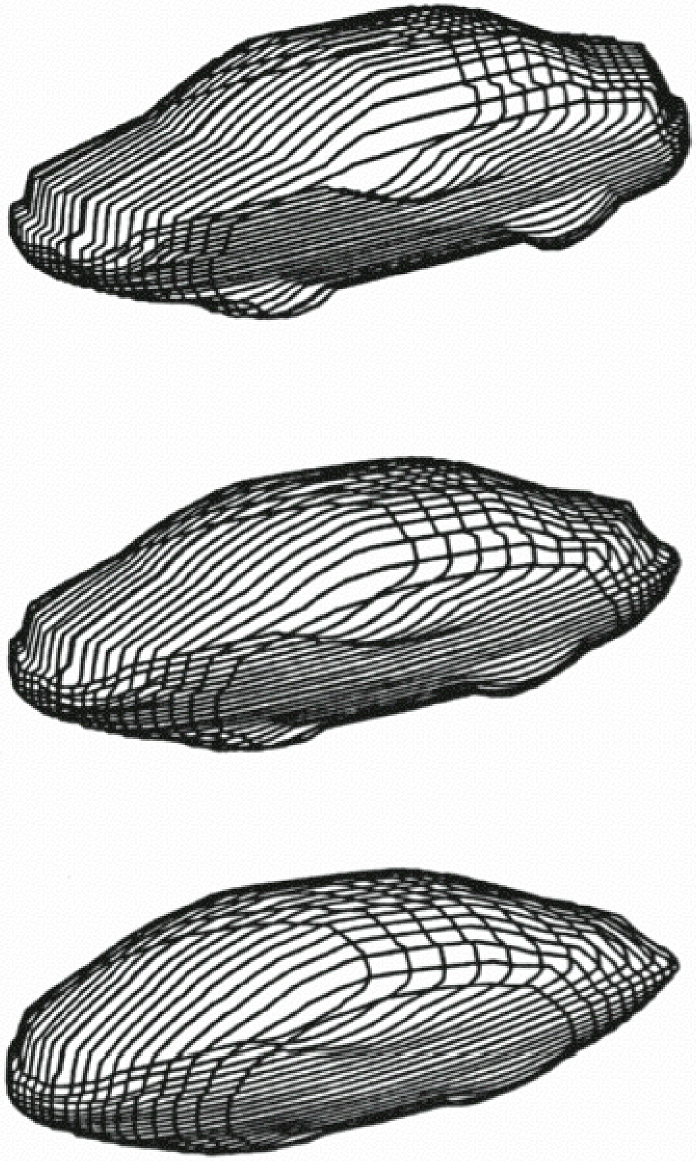
Sampling Object 2 along rays

Points interpolated halfway between objects

Resulting object



Original shapes sliced into contours



Interpolated shapes

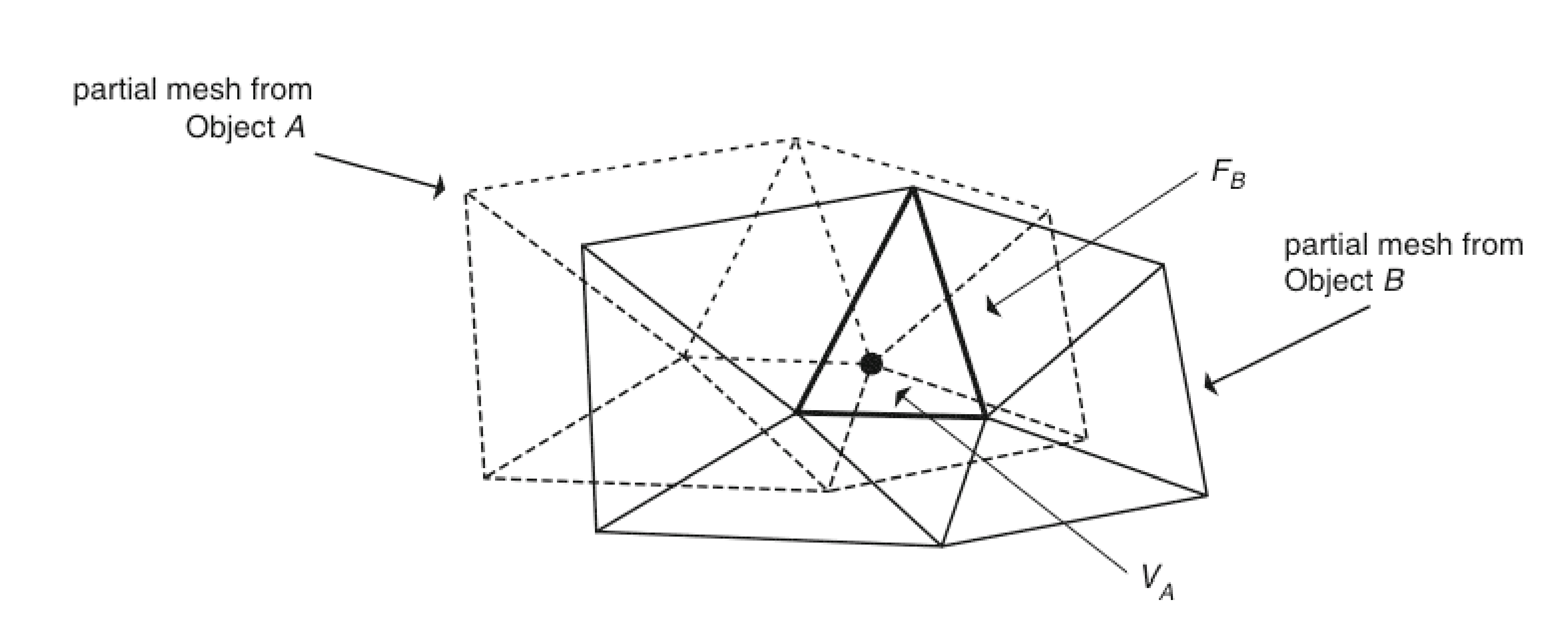
on

Object interpolation

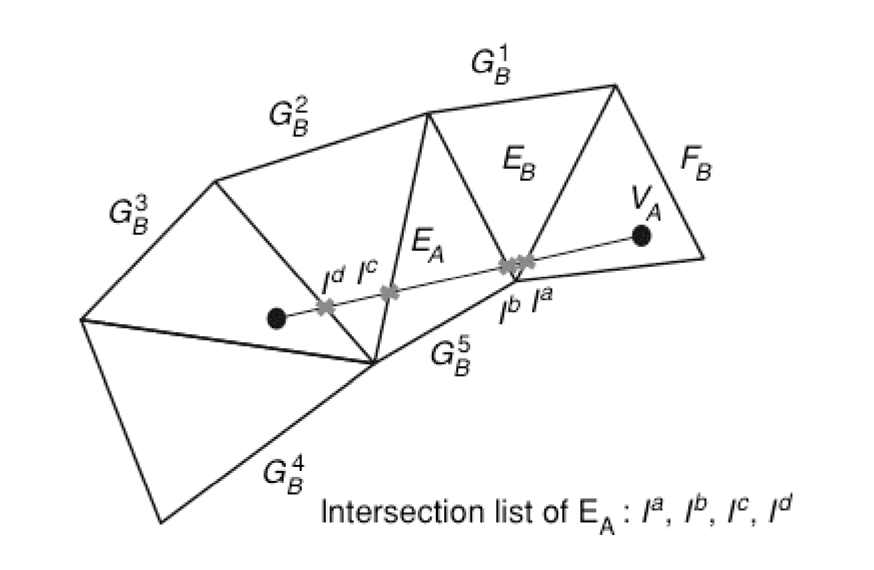
Spherical mapping to establish matching edge-vertex topology

1. Map to sphere
2. Intersect arc-edges
3. Retriangulate
4. Remap to object shapes
5. Vertex-to-vertex interpolation

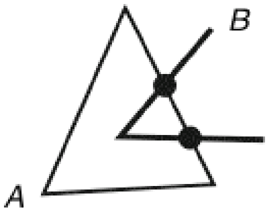
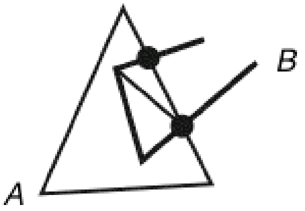
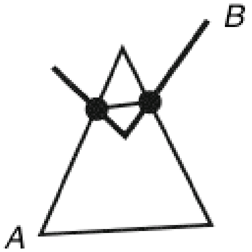
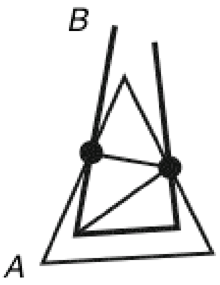
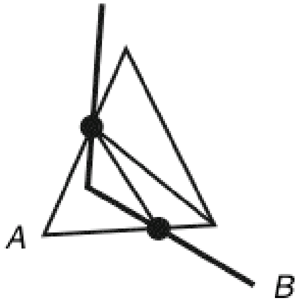
Map to sphere



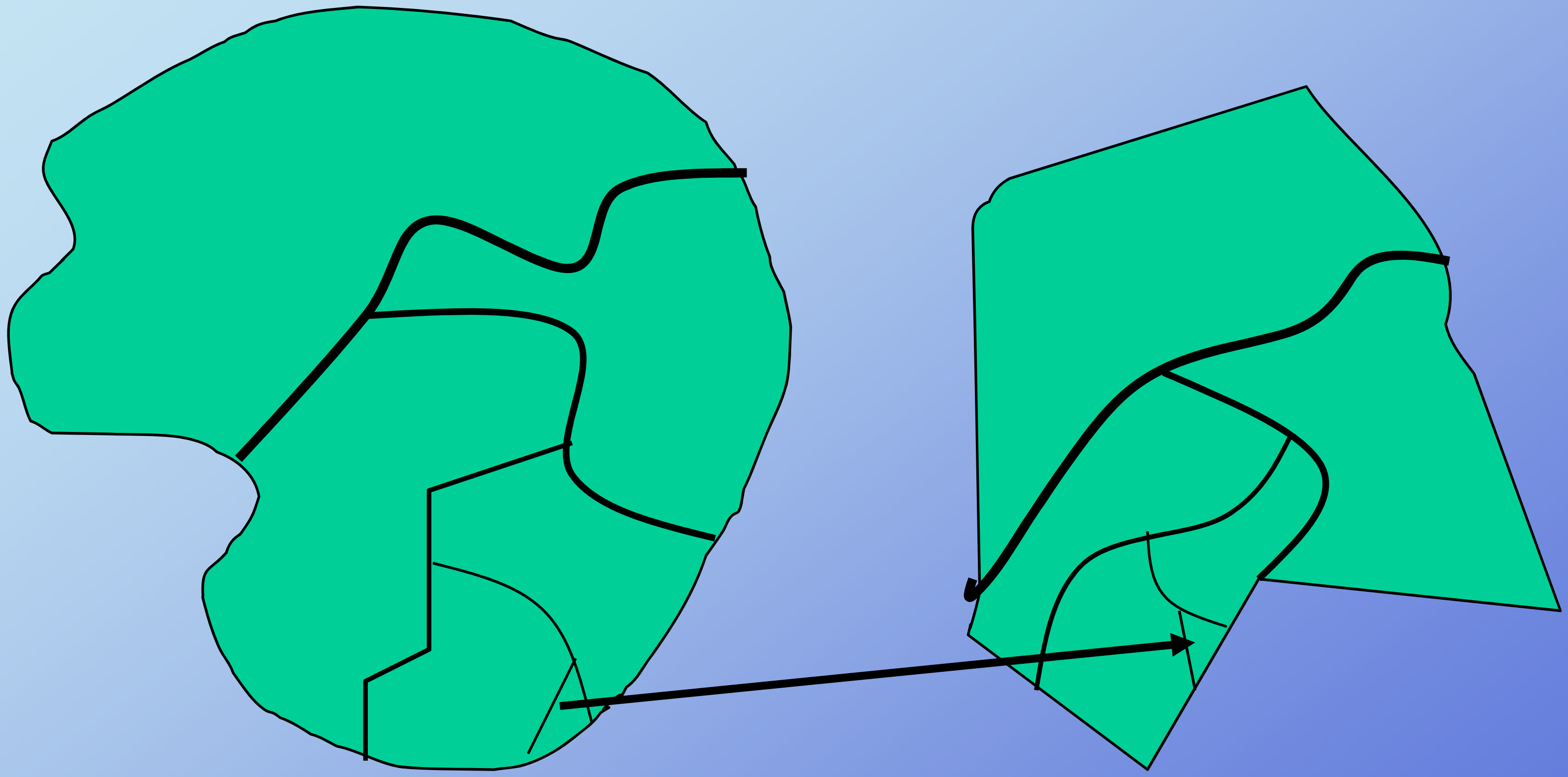
Object interpolation



Intersection list of E_A : I^a, I^b, I^c, I^d

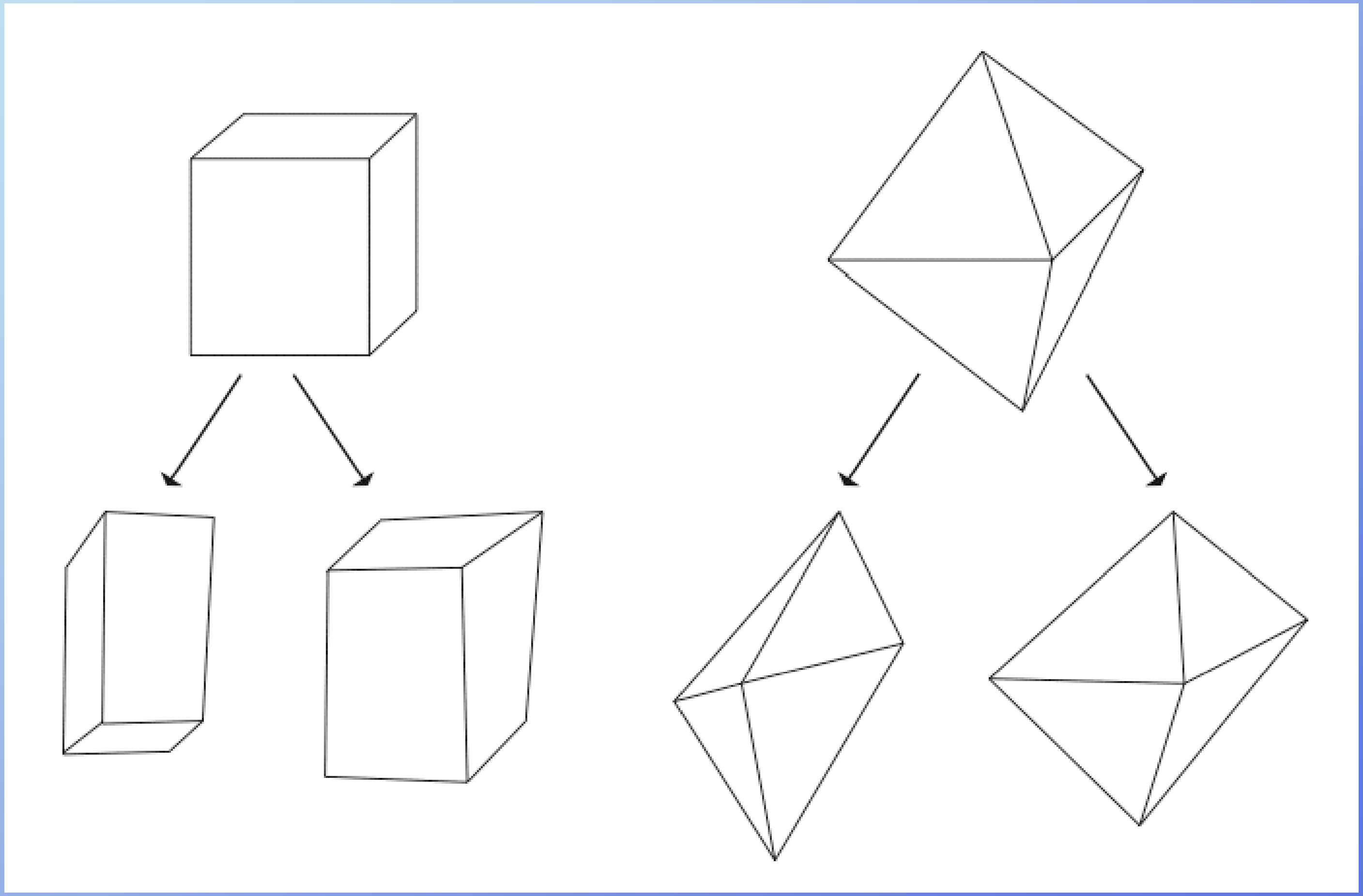
B's intersections connected inside *A* / Vertices of triangle *A* inside of *B*	By a point	By an edge
No vertices		
One vertex		
Two vertices		not possible with triangles

Object interpolation – recursive sheets

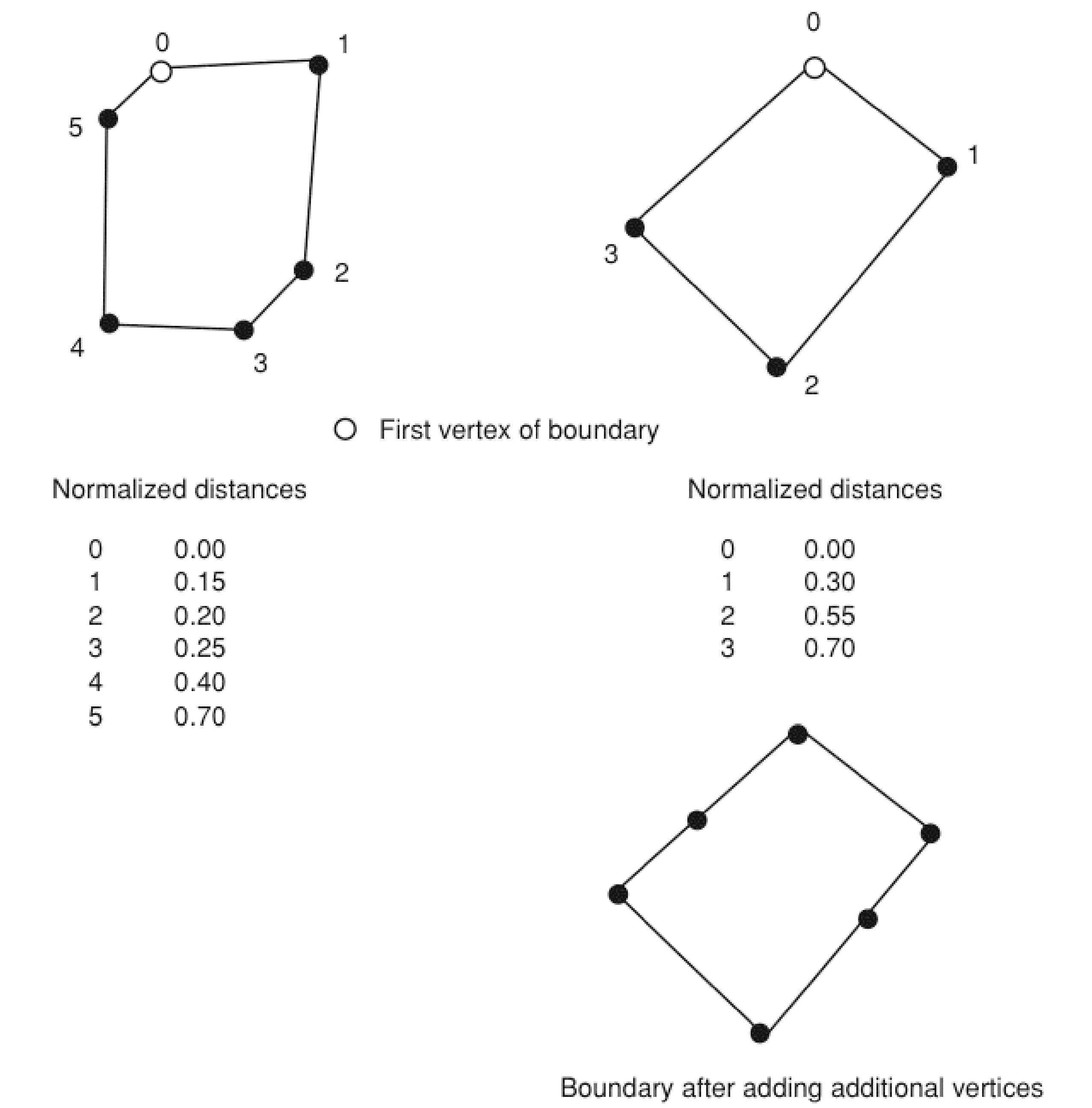


Continually add vertices to make corresponding boundaries have an equal number

Object interpolation



Object interp



○ First vertex of boundary

Normalized distances

0	0.00
1	0.15
2	0.20
3	0.25
4	0.40
5	0.70

Normalized distances

0	0.00
1	0.30
2	0.55
3	0.70

Boundary after adding additional vertices

Morphing

Image blending
Move pixels to corresponding pixels
Blend colors

Morphing

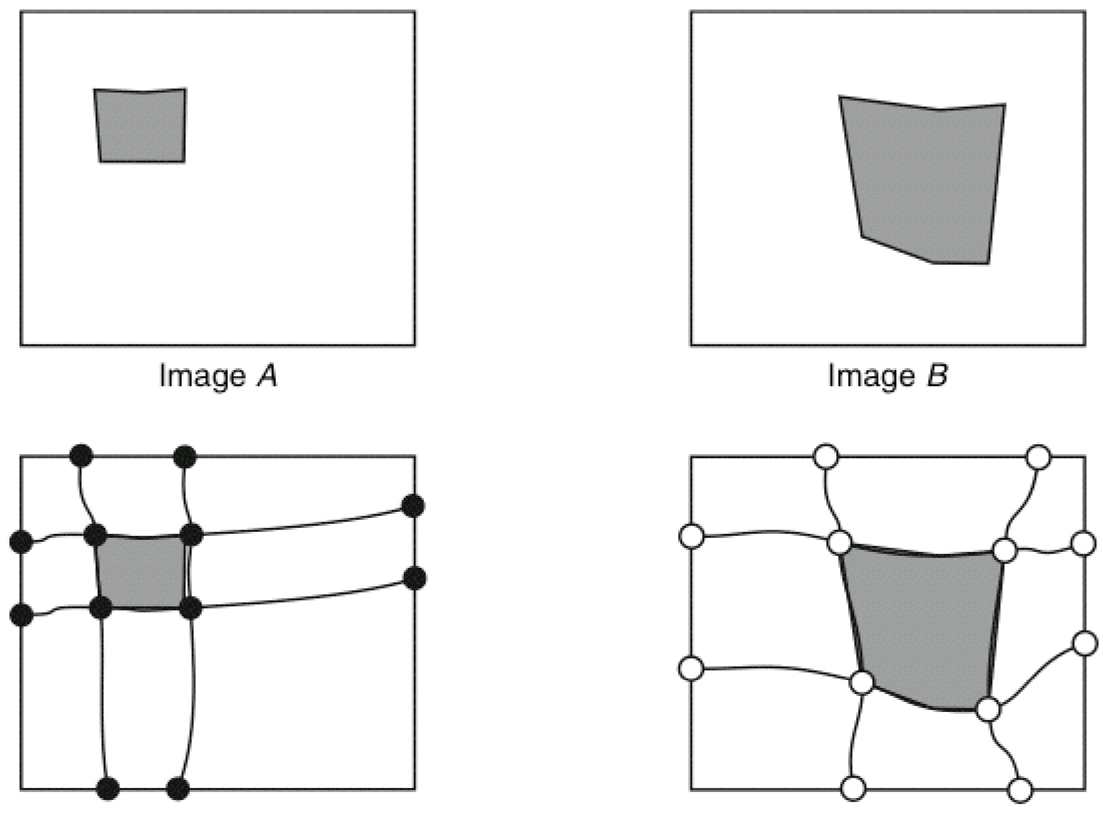


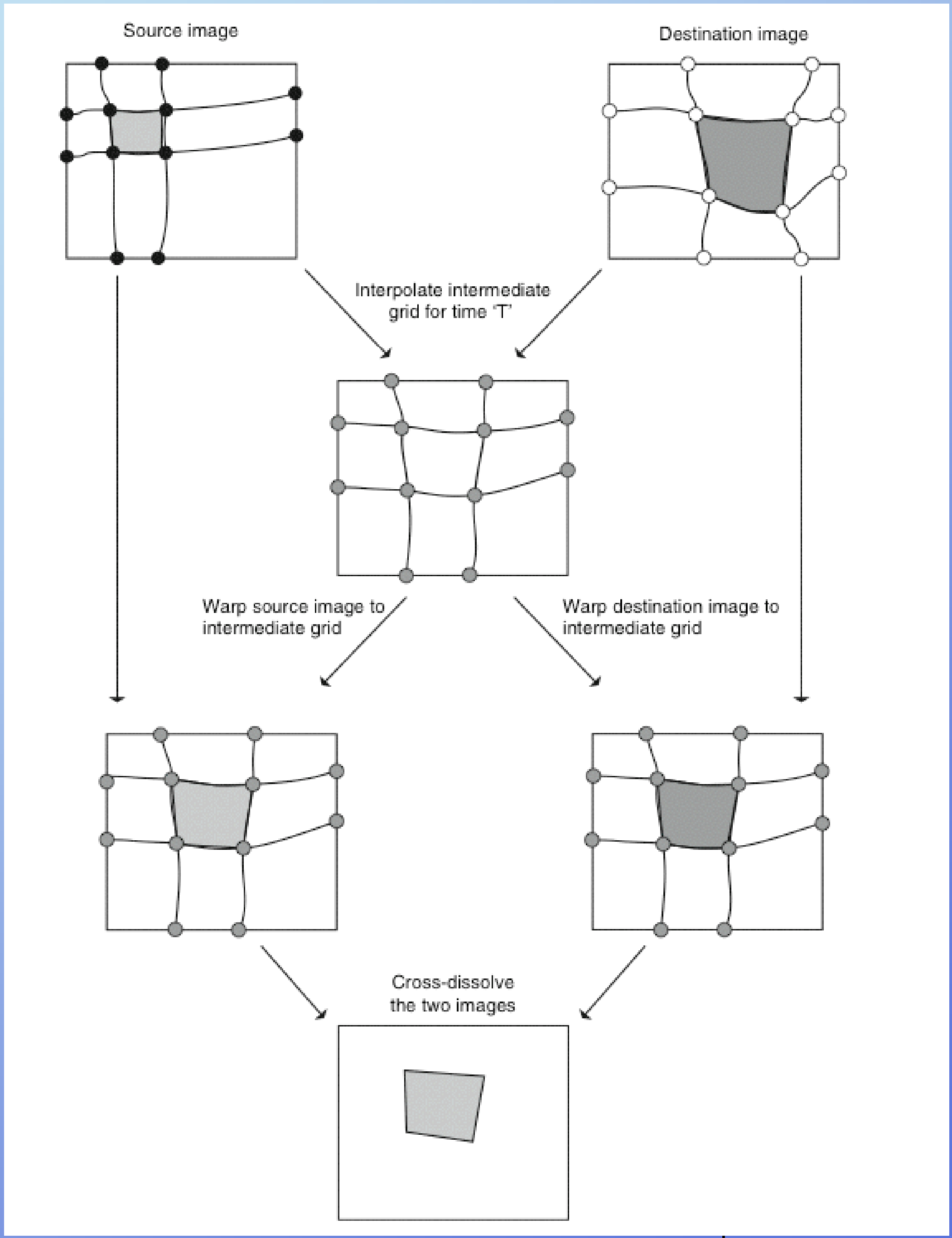
Image *A*

Image *B*

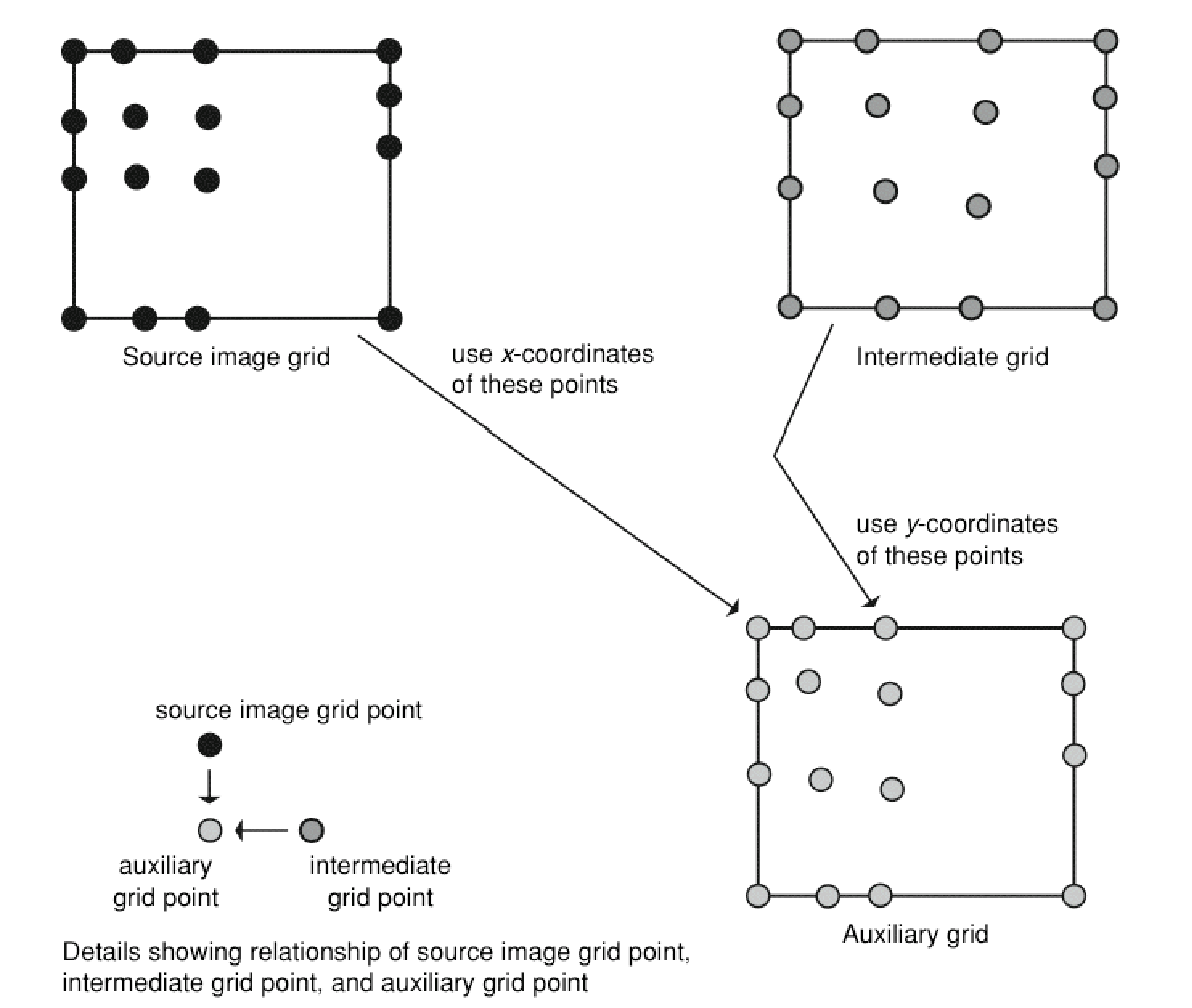
Image *A* with grid points and curves defined

Image *B* with grid points and curves defined

Morphing

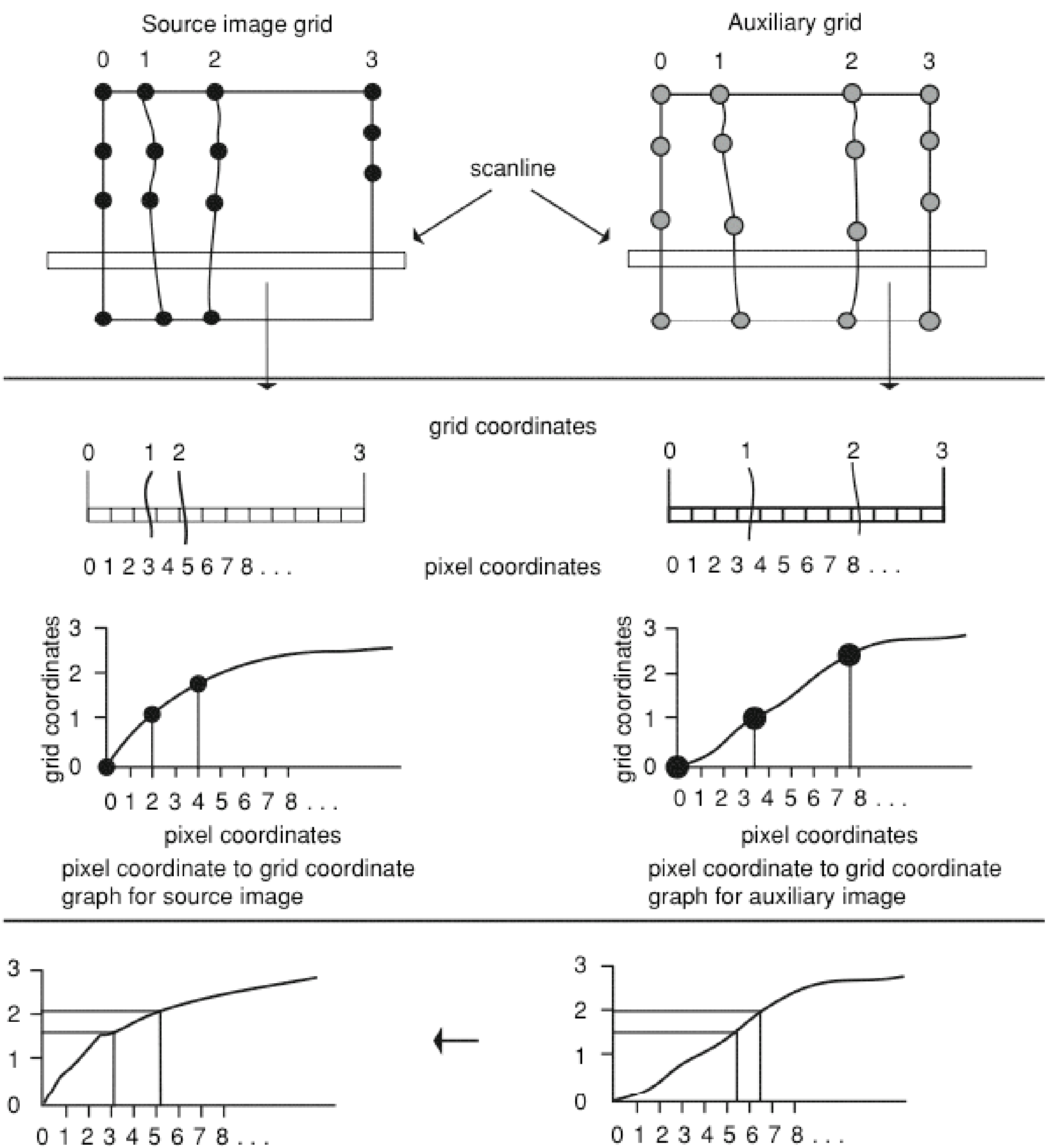

Source image
Destination image
Interpolate intermediate grid for time 'T'
Warp source image to intermediate grid
Warp destination image to intermediate grid
Cross-dissolve the two images

Morph



Details showing relationship of source image grid point, intermediate grid point, and auxiliary grid point

Morphing



pixel coordinate to grid coordinate graph for source image

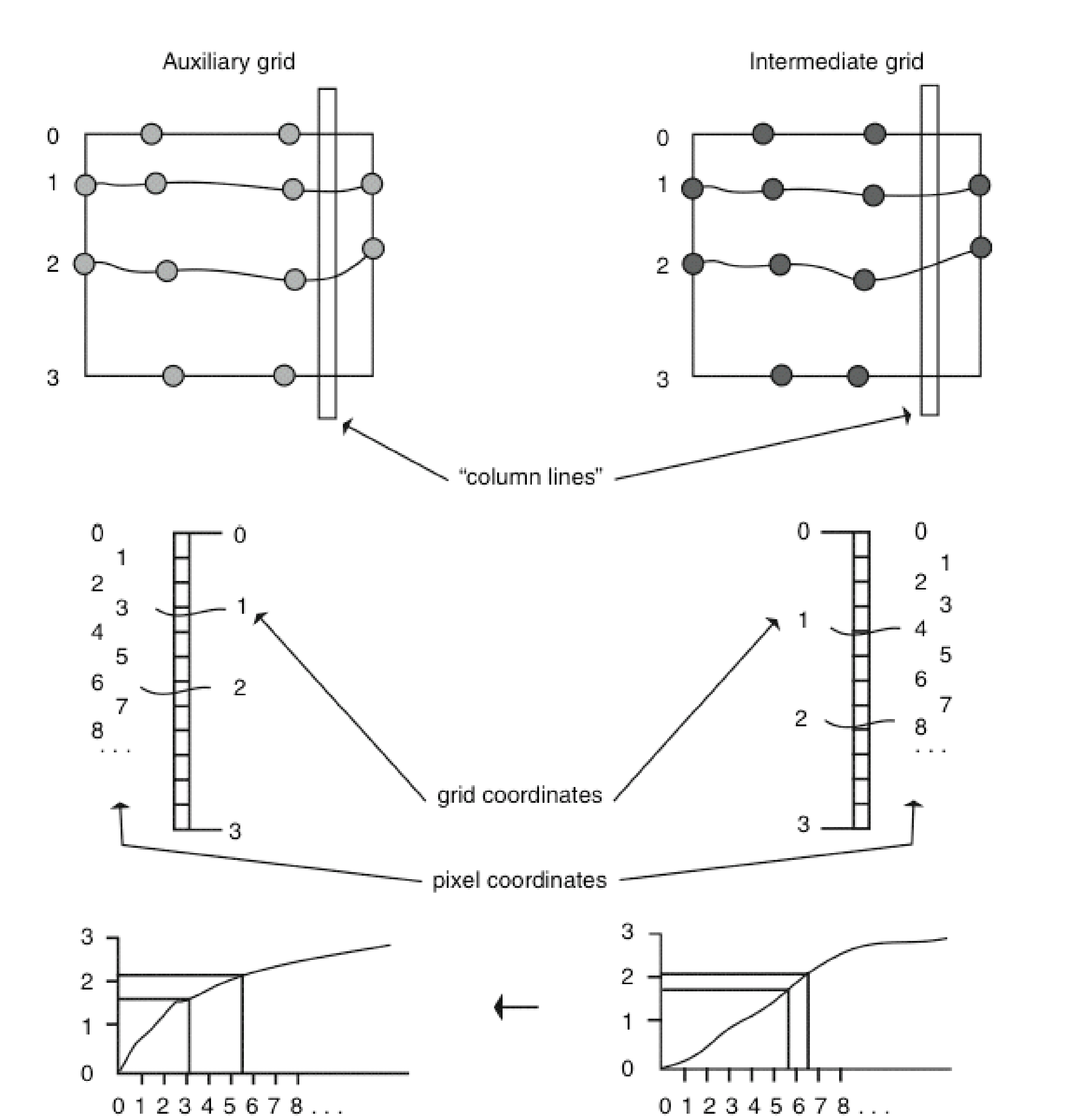
pixel coordinate to grid coordinate graph for auxiliary image

Use the graph to see where the column indices map to image pixels. (Here, half of pixel 3 and all of pixels 4 and 5 are useful)

Use the graph to determine the image pixel's range in terms of the column indices (pixel 6 is shown)

Rick Parent

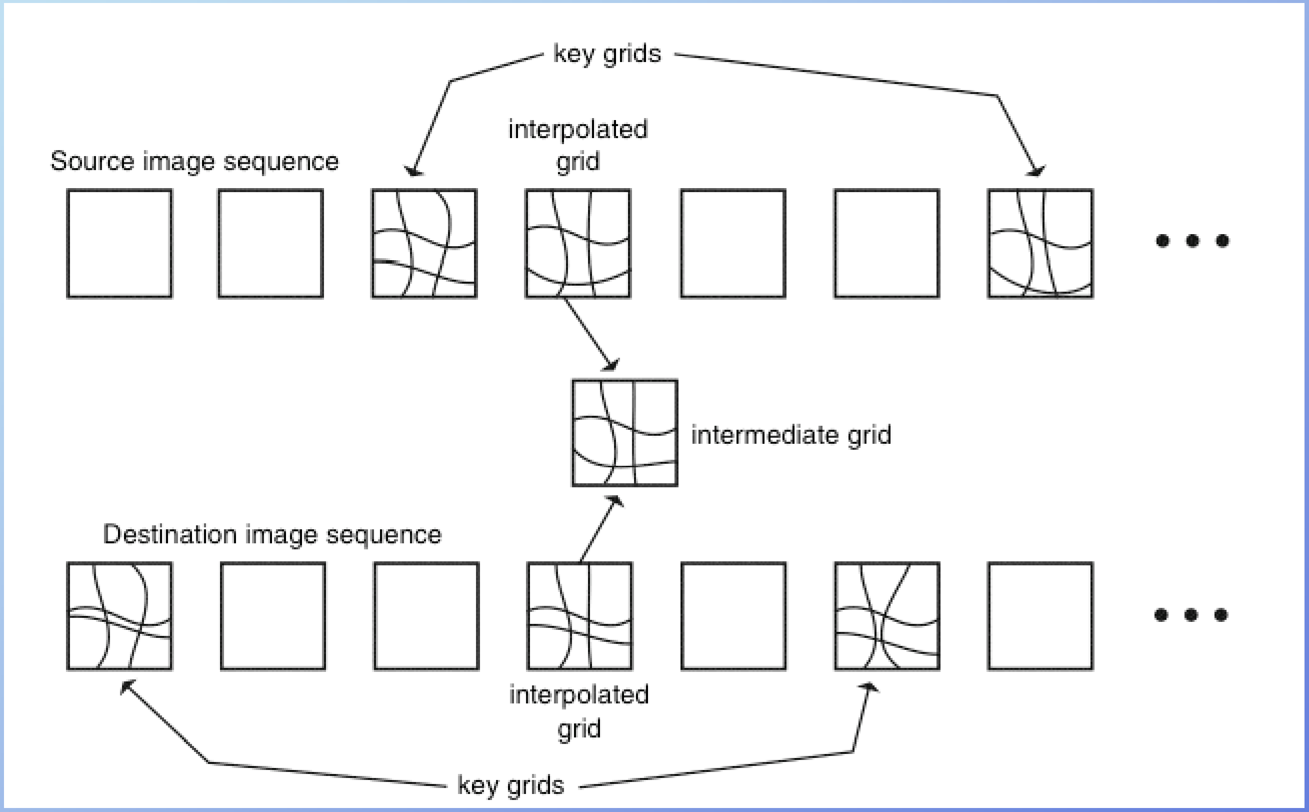
Morphing



Use row index coordinates to determine the pixel coordinates in auxiliary image

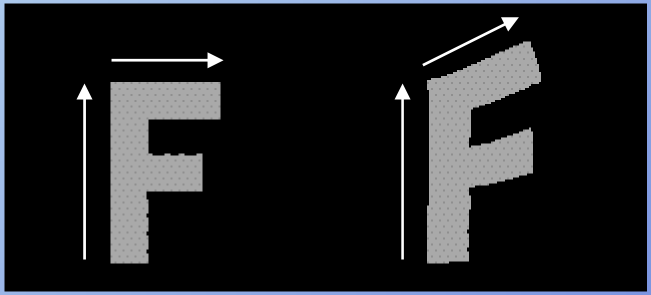
For a given pixel in the intermediate image, determine the coordinates in terms of row indices

Morphing


key grids
Source image sequence
interpolated grid
intermediate grid
Destination image sequence
interpolated grid
key grids

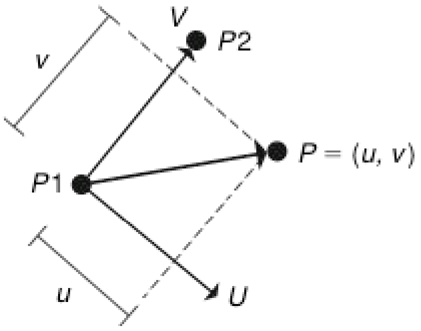
Morphing: feature based

Given: corresponding user-defined feature lines in source and destination images



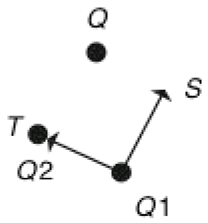
Morphing: feature based

Locate each pixel relative to each feature line in source and destination images



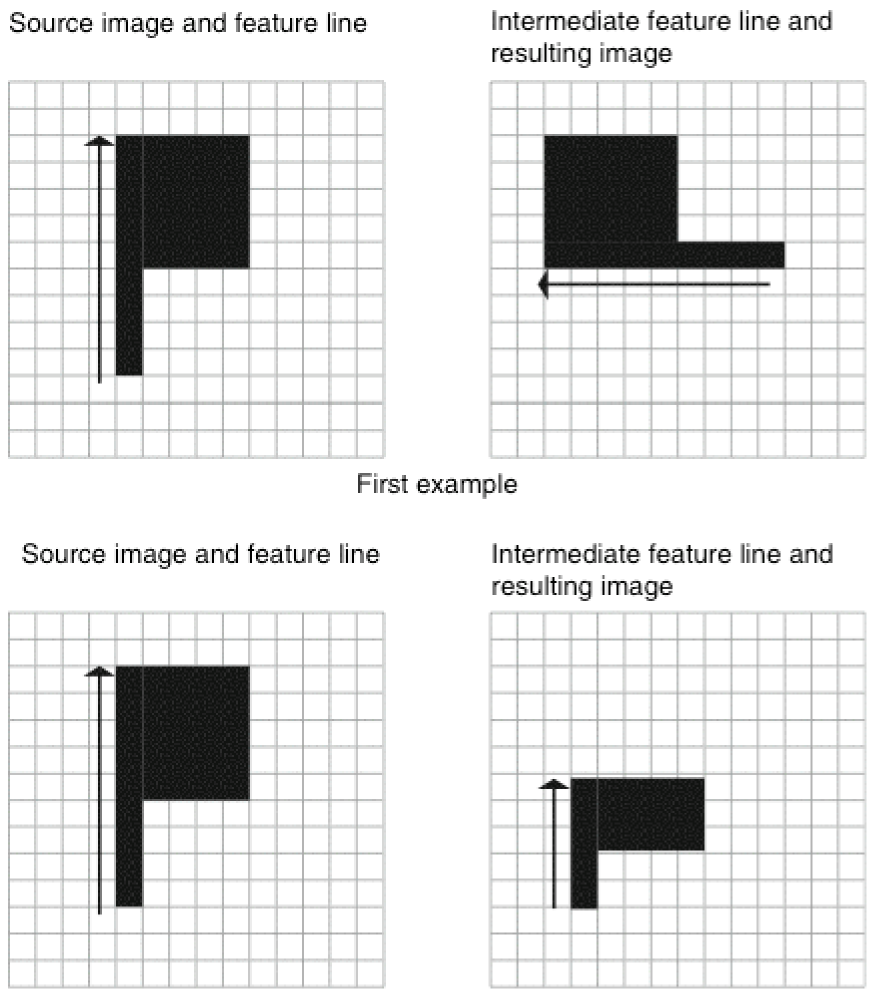
$$v = (P - P_1) \cdot \frac{(P_2 - P_1)}{|P_2 - P_1|^2}$$

$$u = \left|(P - P_1) \times \frac{(P_2 - P_1)}{|P_2 - P_1|^2}\right|$$



$$T = Q_2 - Q_1$$
$$S = (T_y, -T_x)$$
$$Q = Q_1 + uS + vT$$

Morphing: feature based


Source image and feature line
Intermediate feature line and resulting image

First example

Source image and feature line
Intermediate feature line and resulting image

Second example