

# Computer Animation

## 5-Deformations

### SS 13



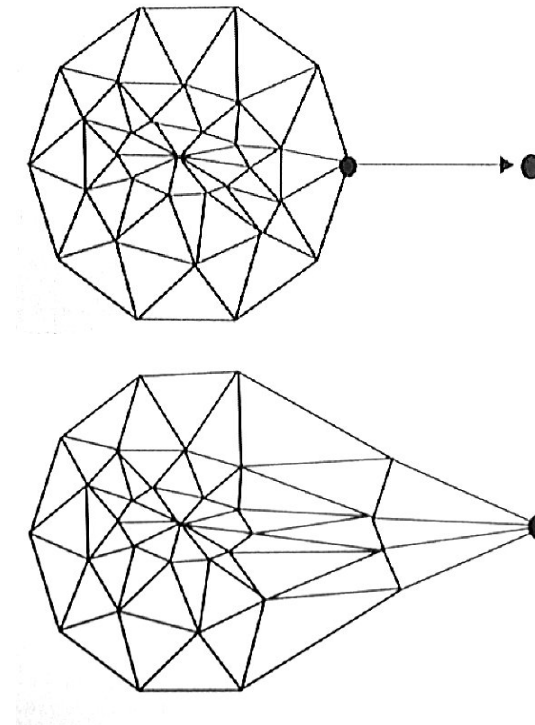
Prof. Dr. Charles A. Wüthrich,  
Fakultät Medien, Medieninformatik  
Bauhaus-Universität Weimar  
caw AT medien.uni-weimar.de

# Deforming objects


- Deforming objects is a powerful animation technique
- The simplest transformation that can be applied to an object is an affine transformation
  - Affine transformations preserve straight lines
  - In general, an affine transformation is a composition of rotations, translations, dilations and shears.
  - They are represented by 3x3 matrices plus a translation:  
 $\underline{x}' = \underline{x}A + \underline{k}$  with A non singular (i.e.  $\text{Det } A \neq 0$ )
- They can be used for the squashing and stretching of an object and the shearing of a sudden stop of an object

# Warping objects

- A simple way to modify the shape of an object is to
  - pull one or more vertices and
  - propagate displacement to neighbouring points
- Such displacement can be attenuated by attenuating relative displacement of first node and transfer it to neighbours



# Warping objects

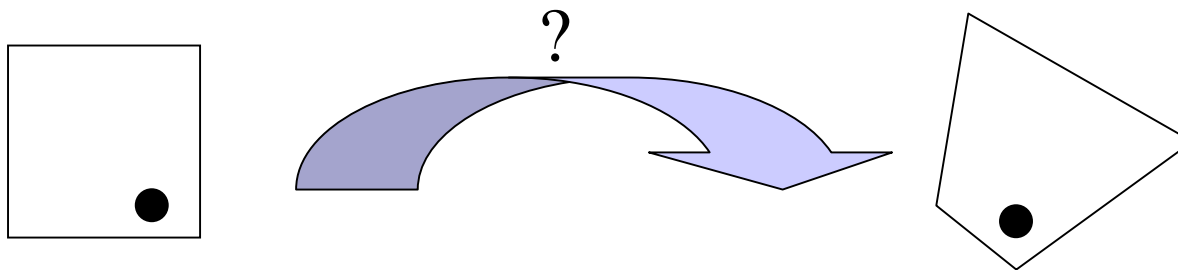
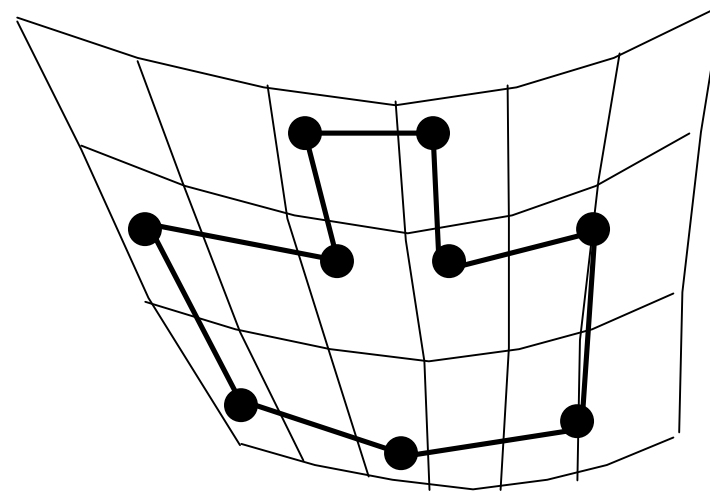
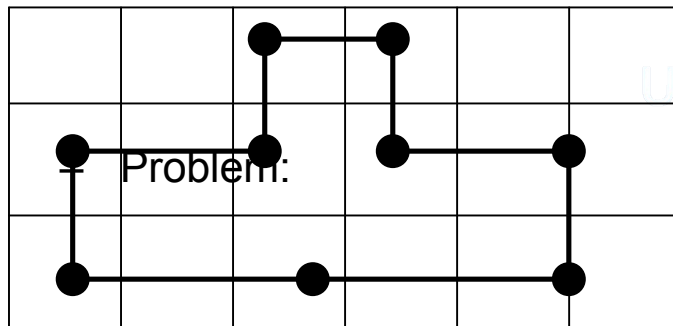
- For this, a distance function may be used, e.g.
    - Min # of edges to be displaced
    - Min distance travelled on object surface between seed vertex and vertex to be displaced
  - Attenuation is chosen as function of distance metric
- 
- One way to do it is to allow the user to select a function from a set of power functions
  - For a vertex which is at distance  $i$  from seeded vertex, a scale factor is applied to the displacement
$$S(i) = \begin{cases} 1 - (i/n + 1)^{k+1} & k \geq 0 \\ (1 - (i/n + 1)^{-k+1}) & k < 0 \end{cases}$$
  - When  $k=0$  we have here linear attenuation
  - $k < 0$  have the effect of quite rigid displacement

# Coordinate grid deformation

- Sederberg introduced *free form deformation* (FFD)
- The idea is to use a local coord system to encase the area of the object to be distorted
  - Typically, this would be initially an orthogonal grid
  - Object vertices are then expressed in this local coord system
  - The user is then allowed to distort local coordinates
- Notice that normally one would choose distortion so that the transformation is continuous
  - i.e. the space is not torn apart
- Transformations can be however non-linear
  - Thus, deformations done this way are more flexible
  - AND they are more natural and intuitive for the user

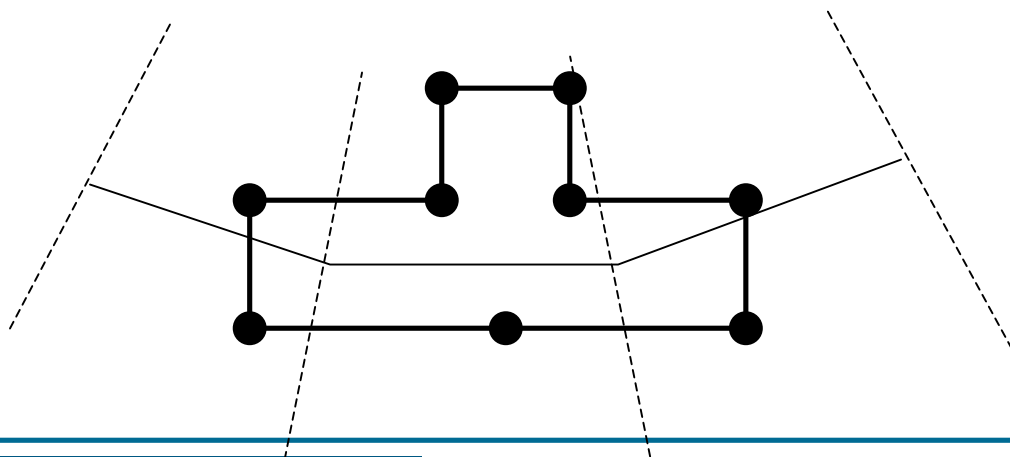
## 2D grid deformation

- The idea here is pretty simple:
  - Take your grid
  - Map to warped space by using bilinear interpolation for the unknown points



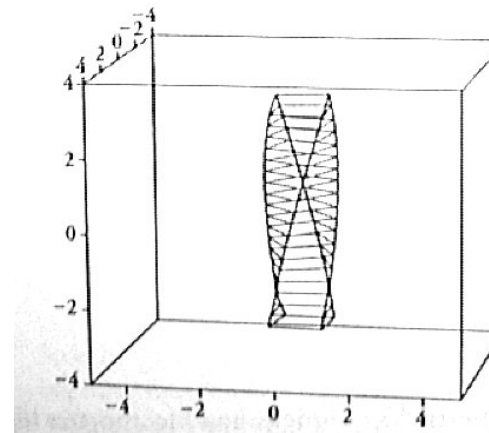
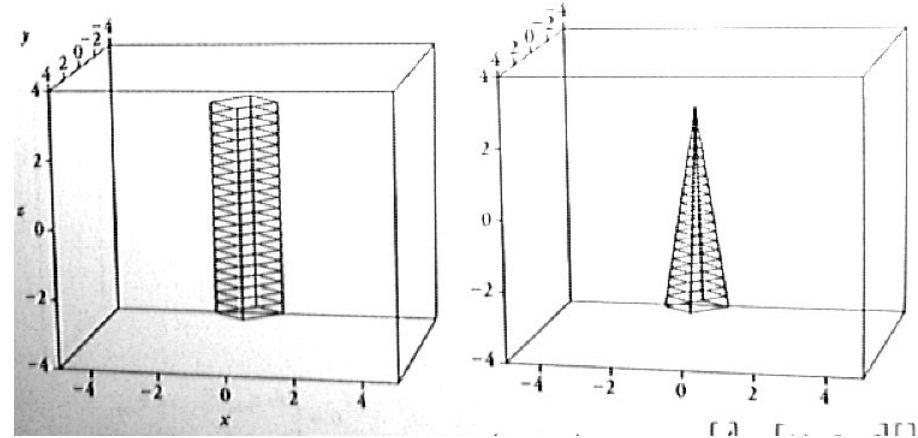
# Polyline deformation

- A similar method employs a polyline to do the deformation:
  - The user specifies a polyline to map the object
  - Each polyline segment boundary point partitions the space in two (boundary lines)
- Each vertex is mapped on the corresponding polyline segment:
  - Construct line between boundary lines // to the polyline segment
  - Store per vertex the following:
    1. Closest polyline segment
    2. Distance from polyline segment
    3. Ratio of length of polyline segment and distance
  - The polyline is then displaced for deformation and new point coordinates are retrieved



# Global deformation

- Alan Barr proposed a method for globally deforming the space into which an object is embedded
- He applies a 3x3 transform matrix which is a function of the point to be transformed:  
 $P' = M(P) \cdot P$
- The transformation function can in principle be any function





# Free form deformation (FFD)

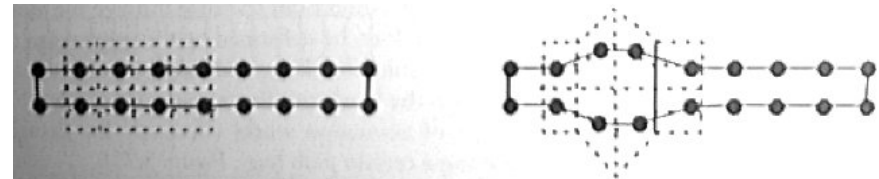
- Basically it is an extension of the 2D technique
- A local coordinate grid is laid around the object
- Transformations are applied to the local grid
- Positions of the new object points are computed through trilinear interpolation
  - Trilinear interpolation is the generalization of bilinear interpolation to 3D
- Alternatively Bezier, or any kind of interpolation can be used

# Compositing FFDs

- FFDs can be composed sequentially or hierarchically
- This way detail can be added bit by bit to the model to achieve the desired level
- For example, if a bent tube is transformed, one can embed at the bend another finer grained FFD which is attached to the larger scale FFD
  - The finer FFD will control better the FFD of the bend

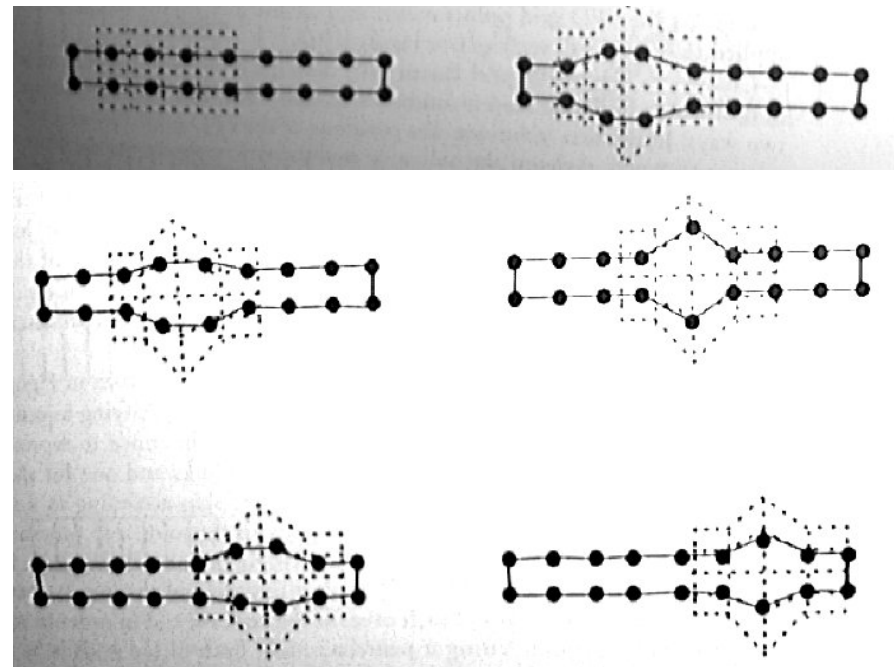
# Deformation tools

- A deformation tool is the composition of an initial lattice and a final lattice (Cocquillart)
  - The initial lattice is user defined and is embedded in the region of the model to be modified (=animated)
  - The final lattice is the copy of the initial lattice that has been deformed by the user
- To the object a deformation tool is therefore defined which controls the deformation of my object
- One animates the object by specifying the movement of the deformation tool relative to the object



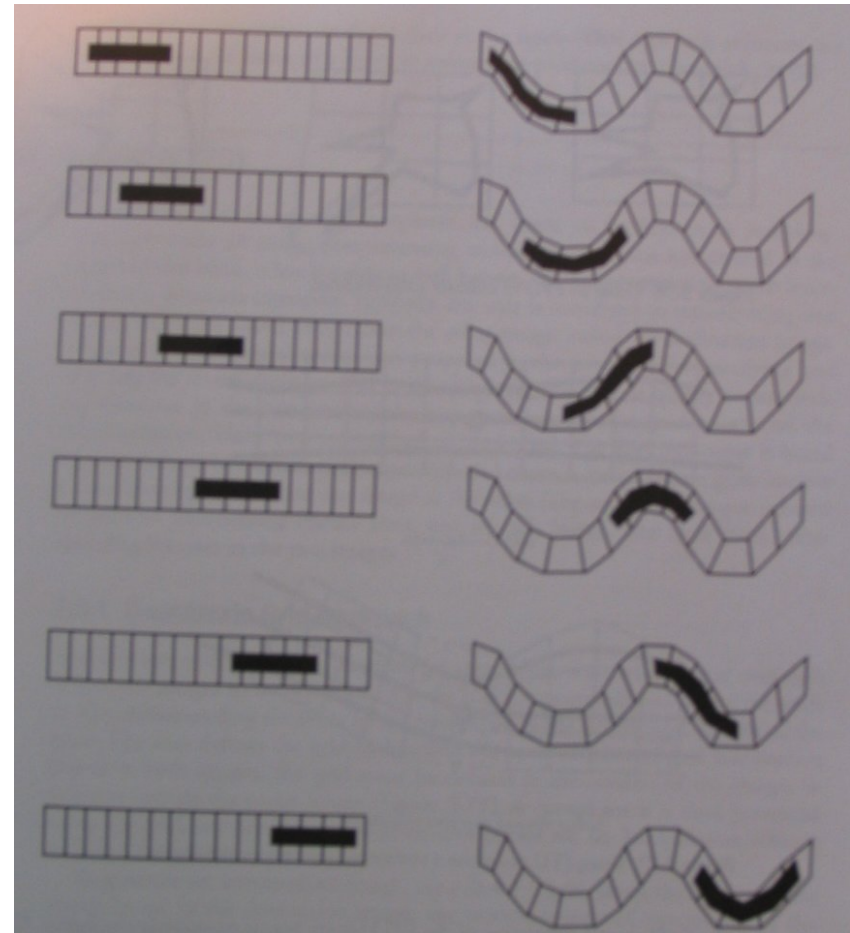
Deformation tool on an object

Moving the deformation tool



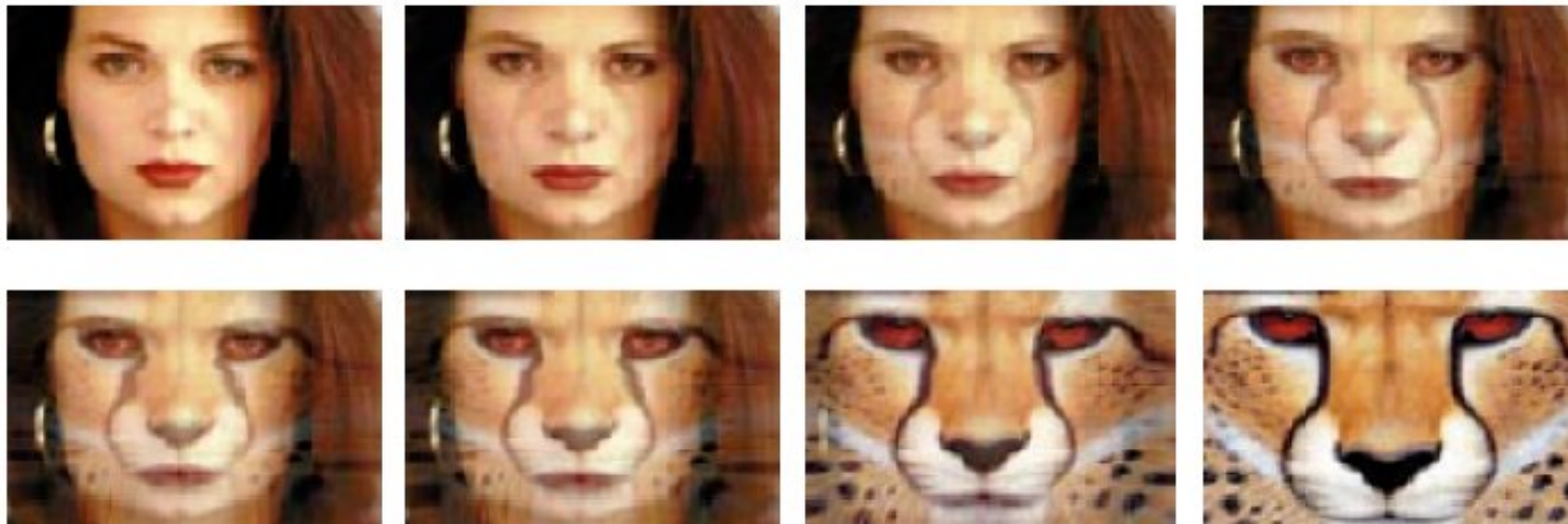
# Animating the FFDs

- It is obvious that what we are doing is separating the logical space of the animation from the actual 3D space
- Look at the example: the logical animation is pretty simple
- Not so the real animation



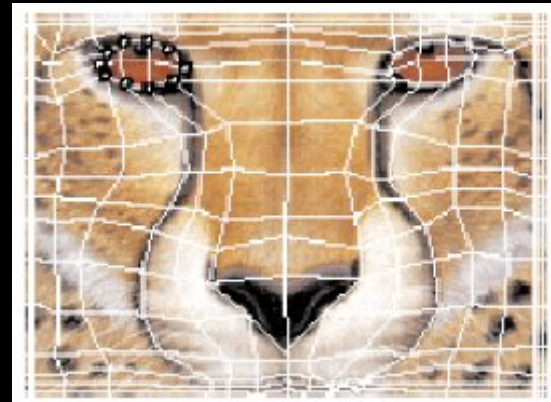
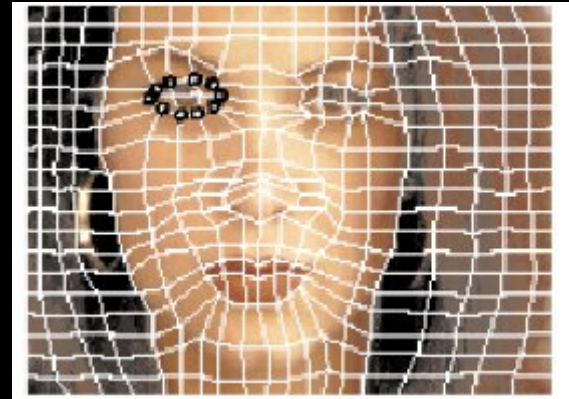
# Morphing (2D)

- Morphing stands for two-dimensional image metamorphosis
- Typically, the user wants to transform one image (source image) into another image (destination image)



# Morphing

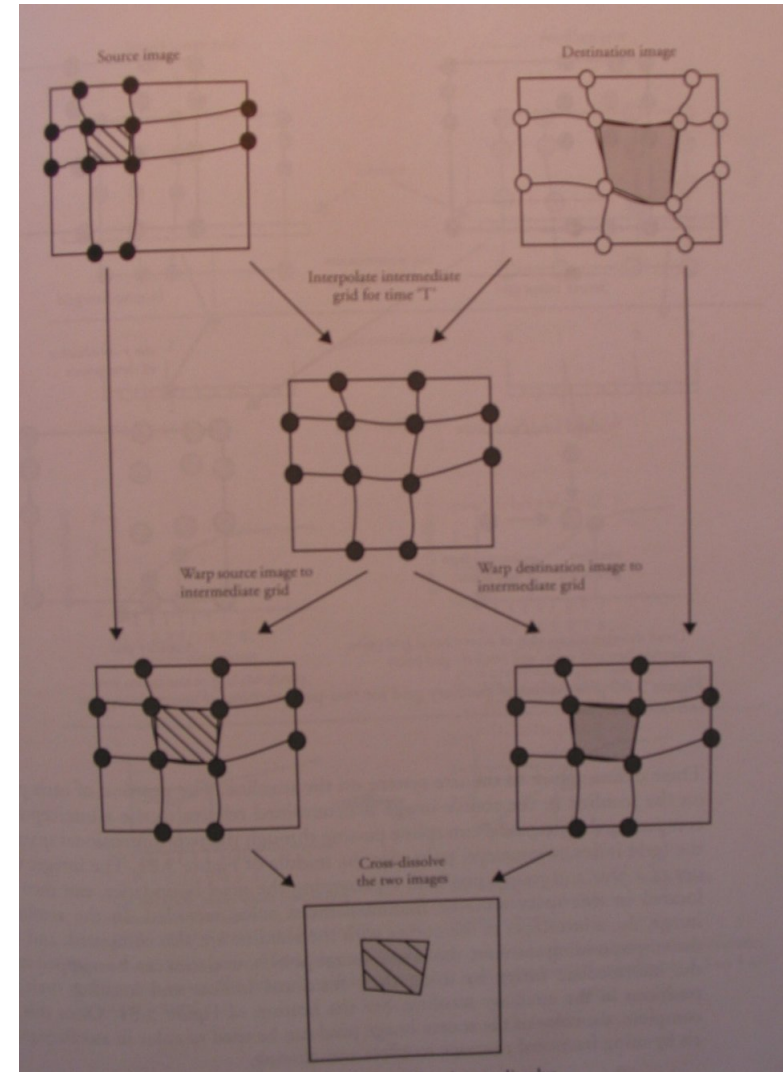
- The user defines a curvilinear grid over each of the two images
- Corresponding elements in the pictures must lie in corresponding cells on the grids
- The number of the cells must also be the same
- To calculate  $k$  intermediate images, the vertices (= points of intersection of the curves) are either linearly interpolated or higher order interpolation is used





# Morphing

- Pixels from source and destination are stretched so that the warped versions of both the source and the destination coincide
- On a pixel by pixel basis a cross dissolve is then performed to obtain the desired effect
- This is done pixel wise on a scanline basis



# Feature based morphing

- Instead of using a grid, the user can establish the correspondence between images by using feature lines
- These feature lines are interpolated to form an intermediate feature line set
- The interpolation can be based
  - either on interpolating endpoints
  - Or on interpolating center points and orientation
- In any case, a map is made for each pixel to each interpolated feature line, and a weight is computed saying how much influence that feature line should have on that pixel
- Such mapping is used in the source image to locate the source image pixel corresponding to that intermediate image pixel
- The weight is used to average the source image locations generated by multiple feature lines into the final source image location
- This location is used to compute the color of the intermediate image pixel
- Same procedure is applied to destination image
- The resulting intermediate images are then cross dissolved



# 3D shape interpolation

- Morphing 3D object is another pair of shoes
- Here, there is no general purpose solution available yet
- The real problem is the fact that this is a topologically complex problem, with deep roots in math
- Topology says that a continuous transformation from an object to another is only possible if the two objects are isomorphic
- Basically, there are two type of techniques:
  - Surface based
  - Volume based
- Surface based techniques modify the polygonal structure of the objects until there is a 1 to 1 map between the polygons
- The number of holes (genus) of the surfaces determine if they are transformable into each other or not

# 3D shape interpolation

- For most cases, the shape transformation problem can be split in two subproblems:
  - The correspondence problem, i.e. Establishing a map between elements
  - The interpolation problem, i.e creating the inbetween interpolations for the objects
- Moreover, a user must be given control about this, and might want to decide for himself which parts of an object are transformed into which parts of the other one

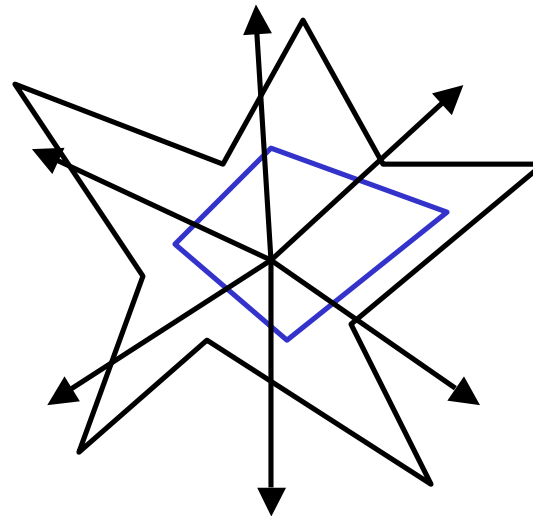
# 3D shape interpolation

- Simplest case: if one object has been derived from another one, then the polygonal structure is the same
- Thus they are easy to transform from one into another
- A simple interpolation between corresponding elements should be good enough



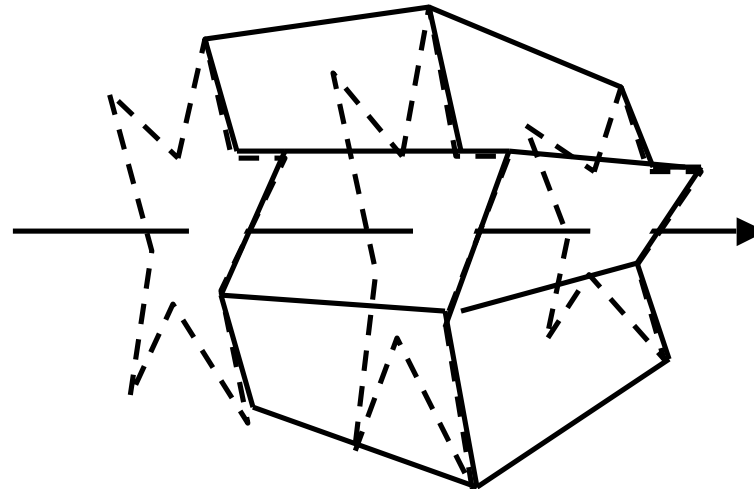
# Other shapes

- Star shaped polyhedra can be transformed into each other by using polar coordinates (those are „2D“)
- Remember: a star shaped object is an object such that there is a point in the interior such that all points of the contour of the object are visible from this point
- Here one can simply sample along rays from the centre and interpolate corresponding points



# Other shapes

- If the objects have an axis, and along this axis the intersection of the object with the slice is star shaped,
- then one can interpolate the slices as said before to transform one object into the other
- This approach can be extended to include the case when there is no axis, but there is a polyline which can function as an axis



# Other shapes

- If the shapes are not such that the two former algorithms can be applied, there are also other possibilities
- If the objects are both of genus 0, one can try to map the objects to an intermediate common object such as a sphere
- Once both objects are mapped to a sphere, then the union of the corresponding mappings is constructed and used to map it inversely to the original objects
- All genus 0 objects can of course be mapped to a sphere, because they are homeomorphic to a sphere
- The problem is to find the homeomorphism, and there is no unique way to do so
- Even if one does so, this method is quite expensive, since the polygons of each object are subdivided in the second step.

# End

---



Copyright (c) 1988 ILM

+++ Ende - The end - Finis - Fin - Fine +++ Ende - The end - Finis - Fin - Fine +++