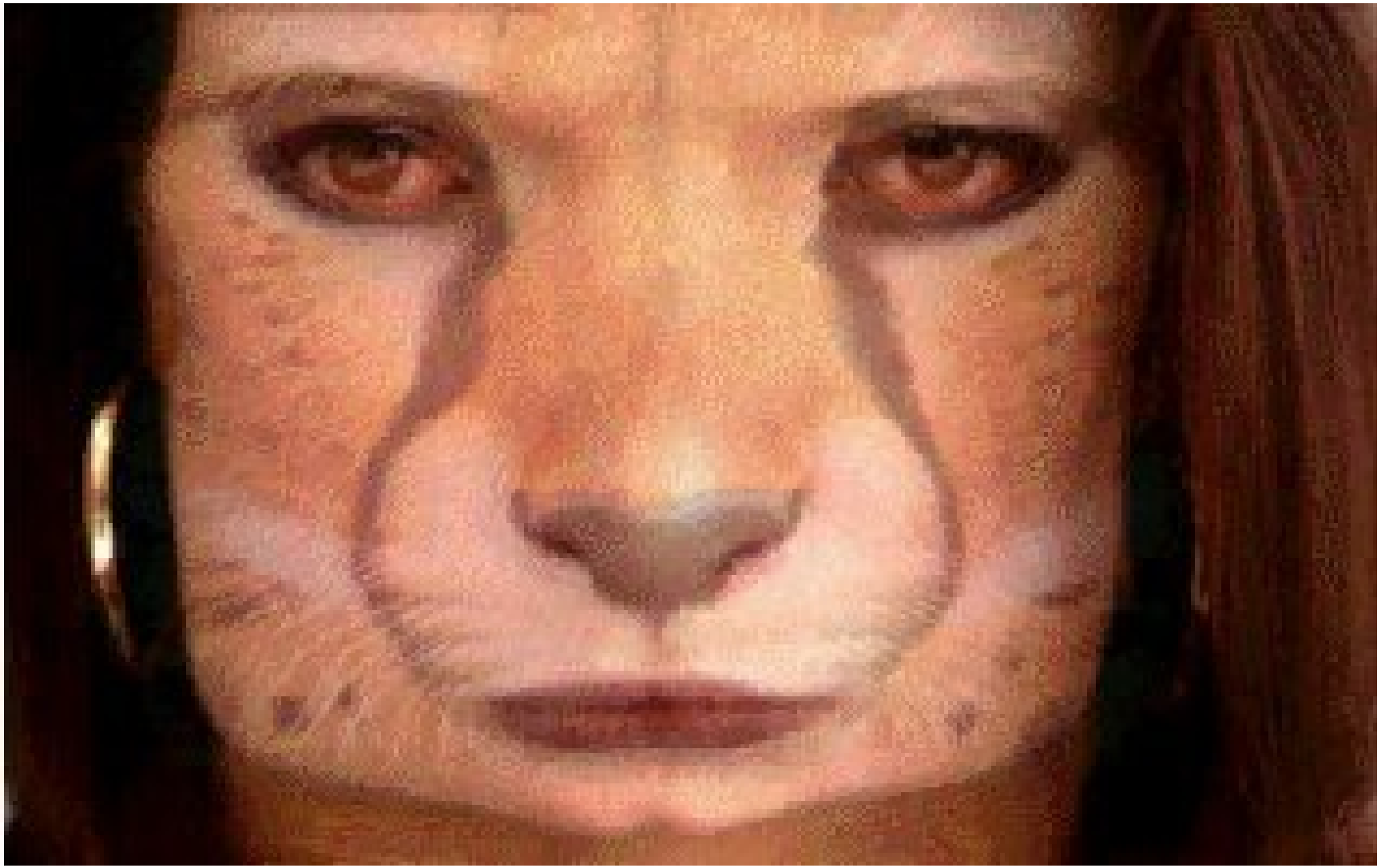
2D preobrazba (morphing)



Shape Transformations

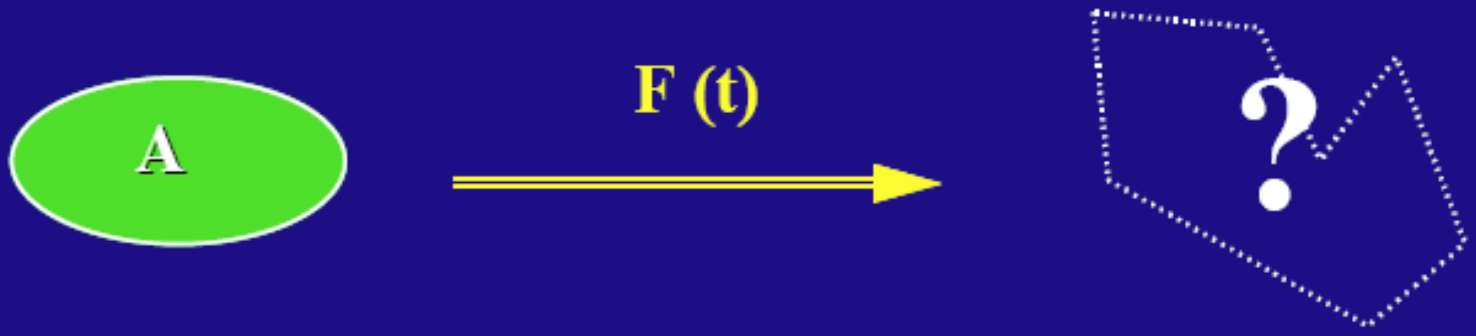
- **Operations**
  - **Warping:** Unary Op
  - **Morphing:** Binary Op

"*Two Related Problems, Same Framework*"

1 - Given Object A and Xform, find Object B

2 - Given Object A and Object B, find Xform

1 - Warping



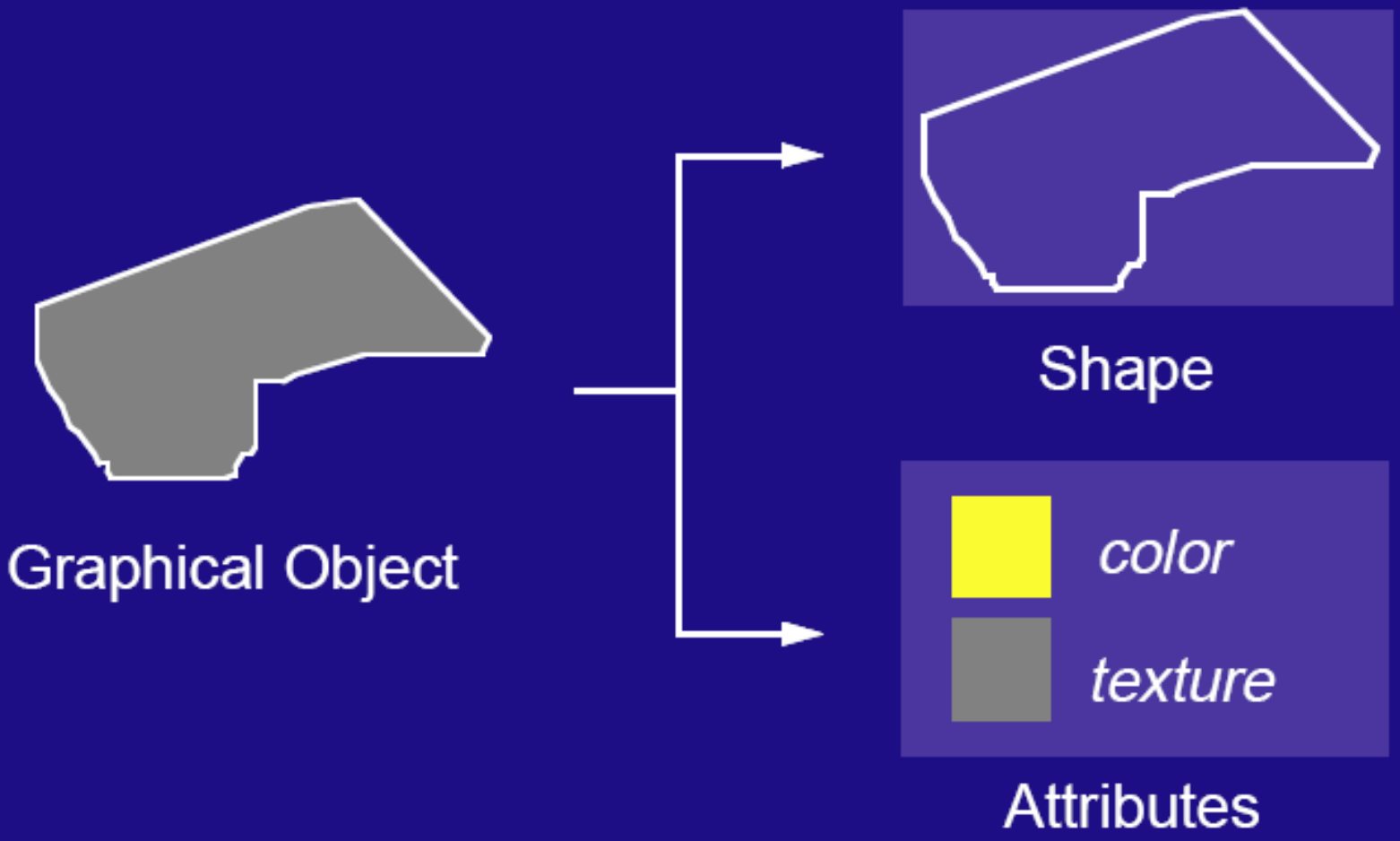
2 - Morphing

Warping and Morphing of Graphical Objects

Processing Pipeline

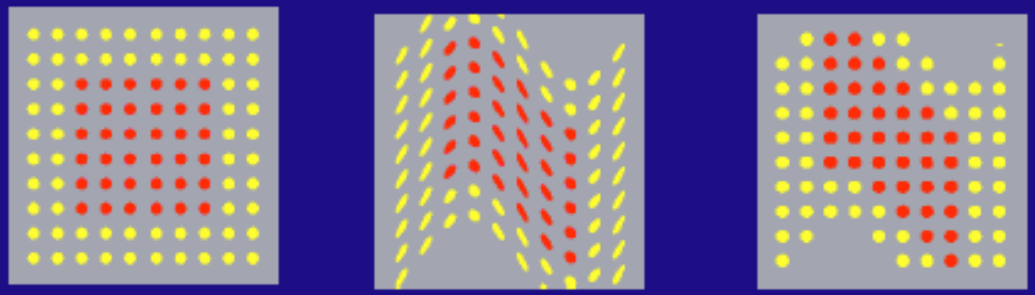
- **Geometry Deformation**
- **Attribute Generation**
- **Object Combination**

Example: 2D Drawing



Attribute Generation

- **Domain Uniformization**
  - **Resampling (adjust to new geometry)**



- **Range Compatibilization** (*morphing*)
  - **Homogenization (attribute equivalence)**

ex: Color Spaces CIE-xyz / RGB

Object Combination

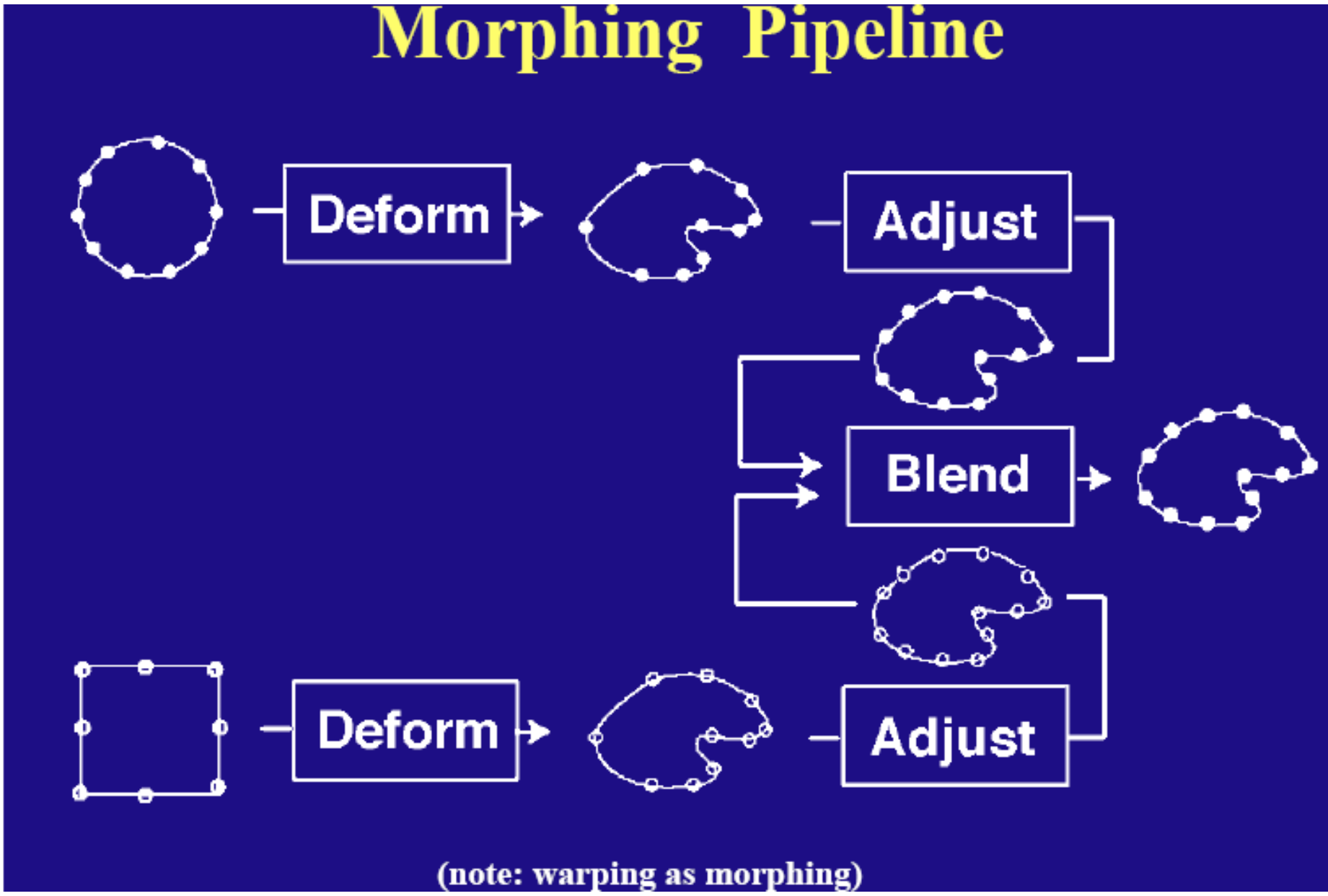
- Correspondence between Objects
  - Domain (shape)
  - Range (attributes)

- Steps
  - Geometry Alignment
  - Attribute Blending

(morphing)

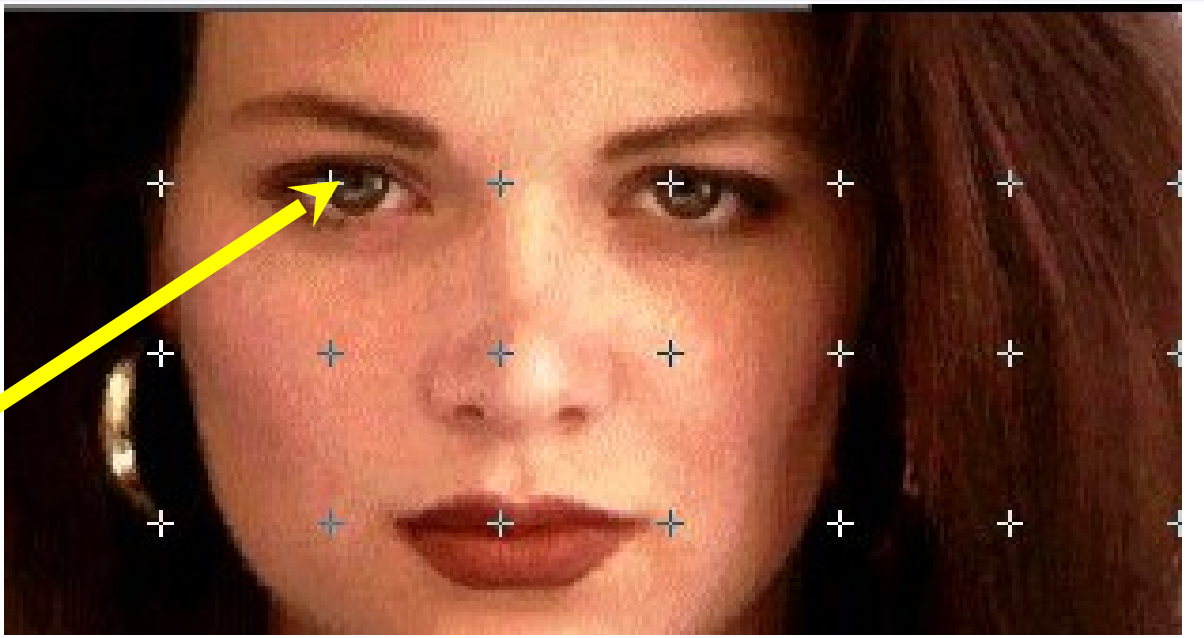
Warping Pipeline

Morphing Pipeline

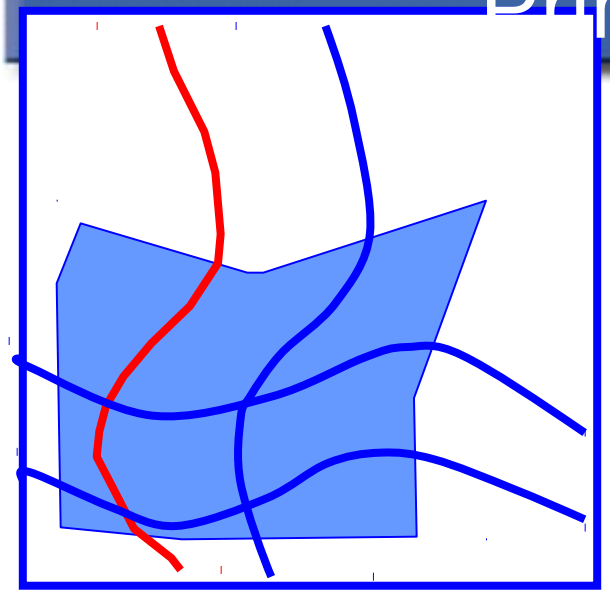


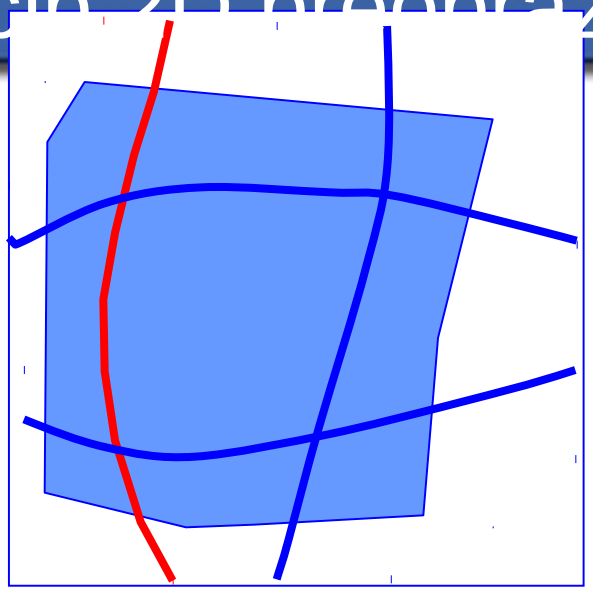
(note: warping as morphing)

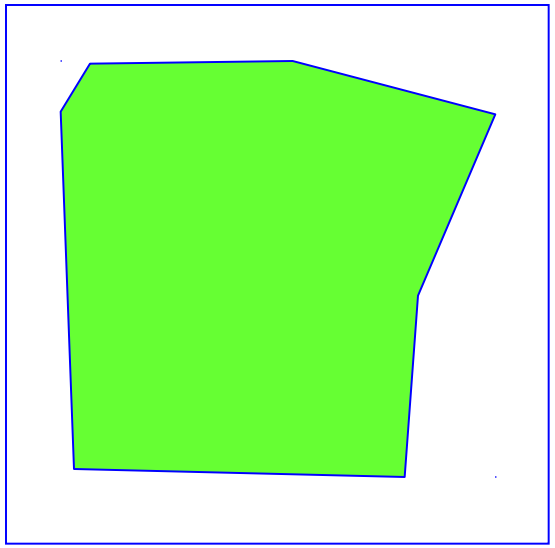
2D preobrazba dekle-tiger

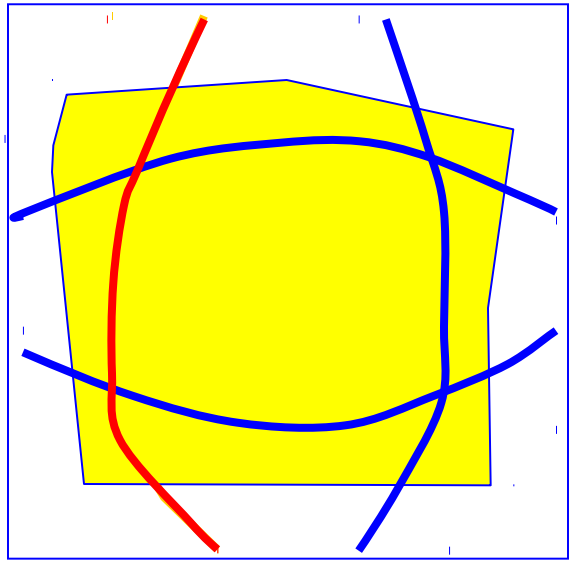


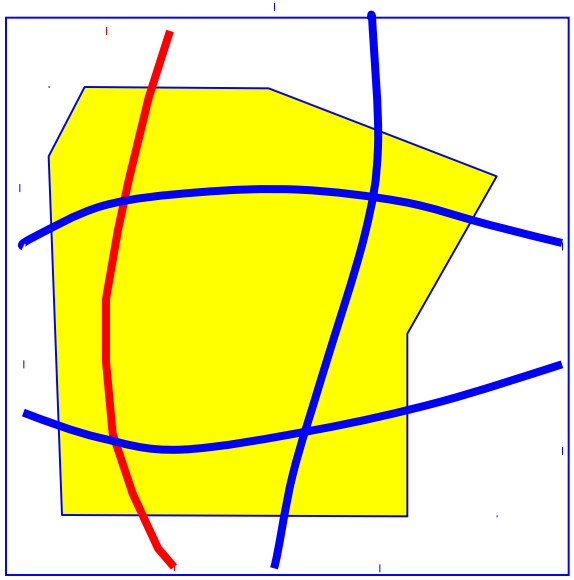
Princip 2D preobrazbe





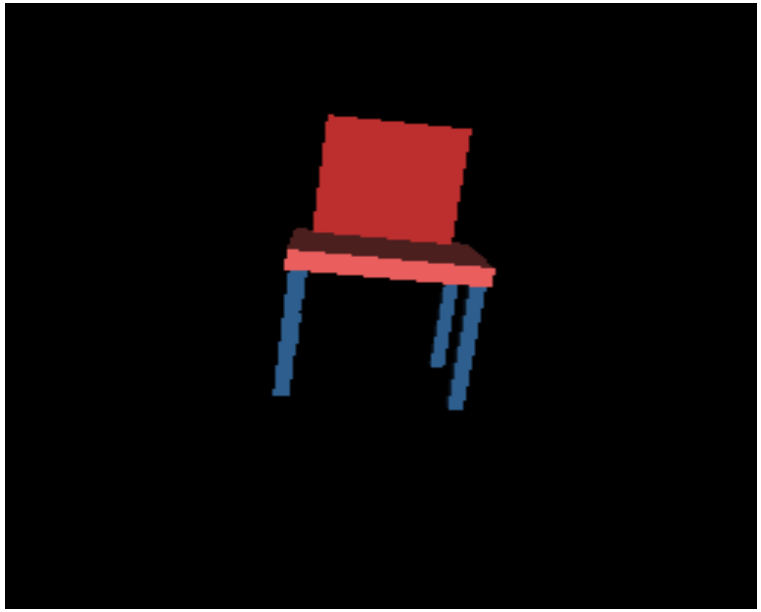






Uvod

- Morphing – derived from the word metamorphosis.
- Metamorphosis means to change shape, appearance or form.Example:



Kaj je preobrazba?

- Morphing can be defined as:
  - Transition from one object to another.
  - Process of transforming one image into another.
- An animation technique that allows you to blend two still images, creating a sequence of in – between pictures that when played in Quick Time, metamorphoses the first image into the second.

Kaj je preobrazba slike?

- Creating a smooth transition between two images
- 3D model based or Image based
- Used for obtaining special effects

Tehnike preobrazbe slike

- Cross-dissolve
- Field morphing
- Mesh morphing
- Radial Basis Functions (RBF)
- Energy minimization
- Multilevel Free-Form Deformation (MFFD)

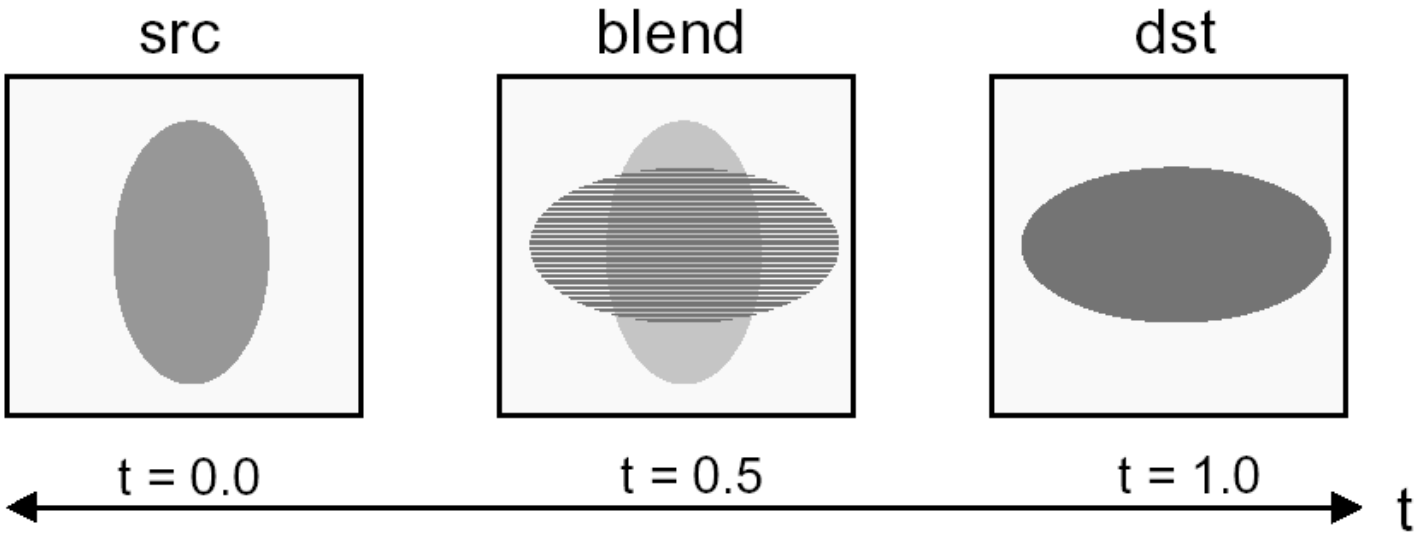
Cross-Dissolve

- Pixel-by-pixel color interpolation
- Very primitive
- Not smooth transitions

Cross-Dissolving

- Blend images with “over” operator
  - alpha of bottom image is 1.0
  - alpha of top image varies from 0.0 to 1.0

$$\text{blend}(i,j) = (1-t)\,\text{src}(i,j) + t\,\text{dst}(i,j) \quad (0 \le t \le 1)$$



Problemi

Problems

- Problem with cross-dissolve is that if features don't line up exactly, we get a double image
- Can try shifting/scaling/etc. one **entire** image to get better alignment, but this doesn't always fix problem
- Can handle more situations by applying different warps to different **pieces** of image
  - Manually chosen
  - Takes care of feature correspondences

Image I_S with mesh M_S defining pieces

Image I_T, mesh M_T

from G. Wolberg, CGI '96

Solution: Mesh Warping

- Source and target images are meshed
- The meshes for both images are interpolated
- The intermediate images are cross-dissolved
- Here, we look at 2D example

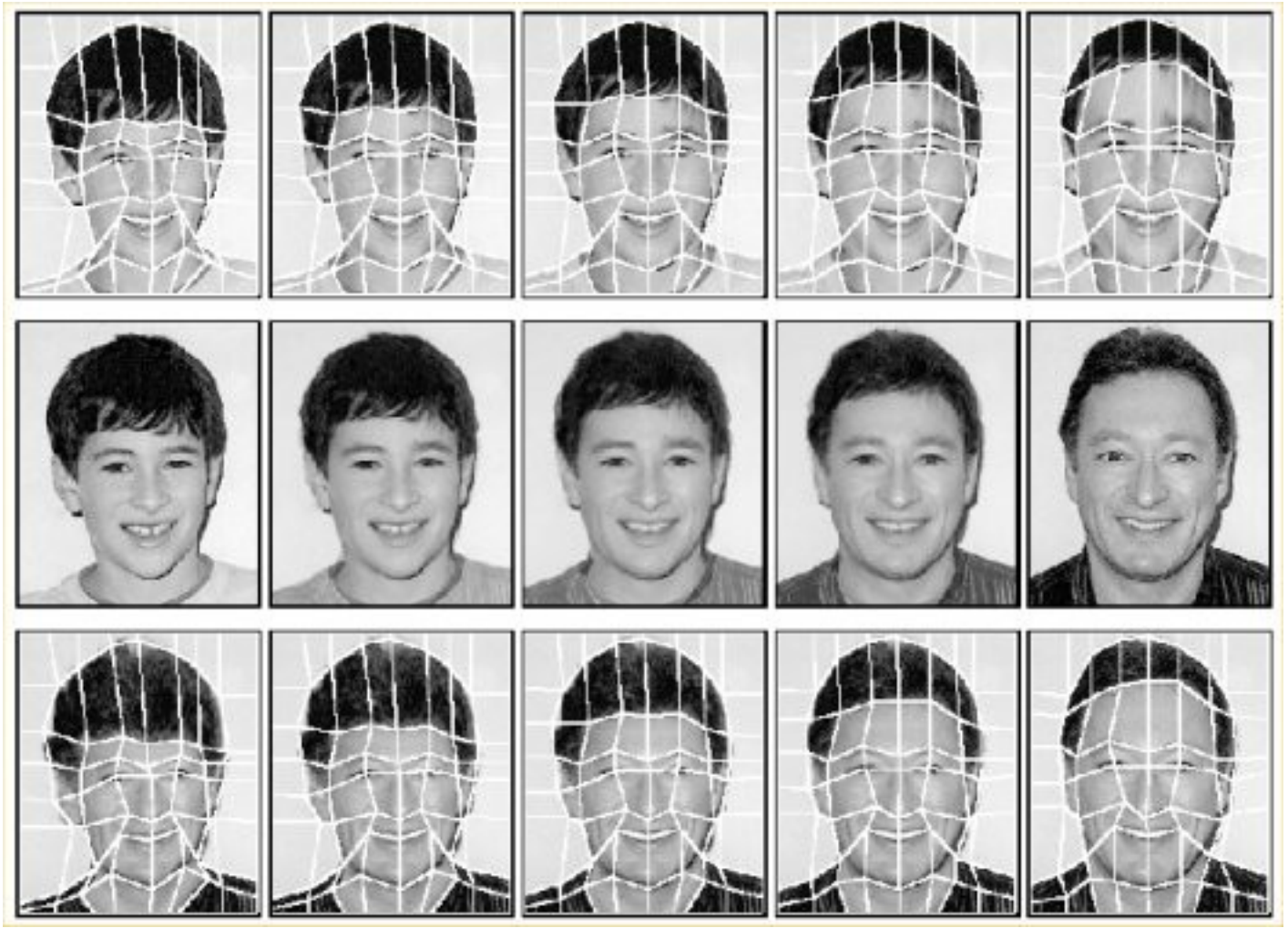
Mesh Warping

- Source and target images are meshed
- The meshes for both images are interpolated
- The intermediate images are cross-dissolved

Mesh Warping

- for each frame f do
  - Linearly interpolate mesh M, between M_s and M_t
  - warp $Image_s$ to I_1, using meshes M_s and M
  - warp $Image_t$ to I_2, using meshes M_t and M
  - Linearly interpolate image I_1 and I_2
- end

Mesh Warping



Mesh Warping

- Hard to fit the mesh in images
- All control points affect the warping equally
- Not enough control in certain areas when needed

Image Morphing

- Combines warping and cross-dissolving

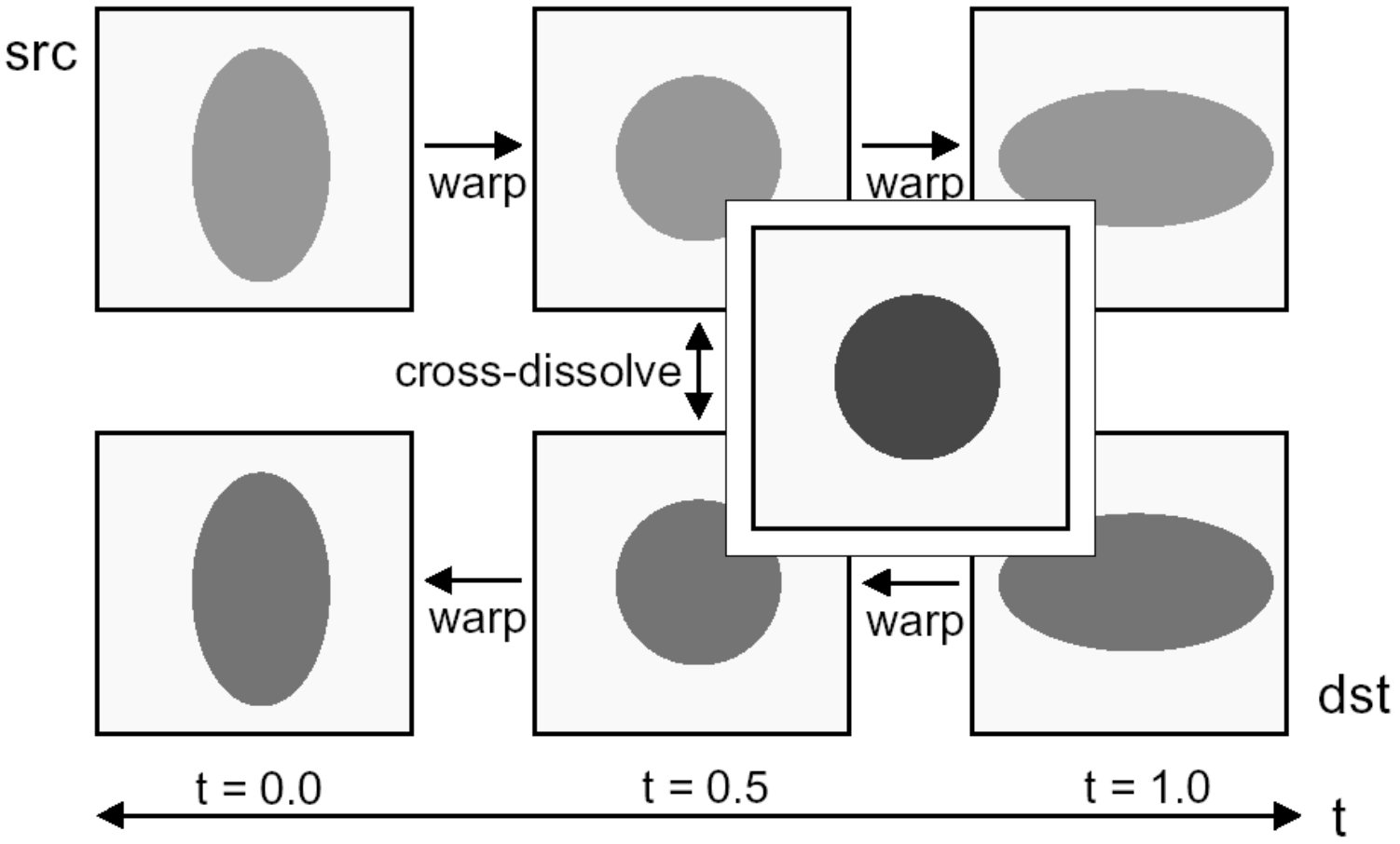
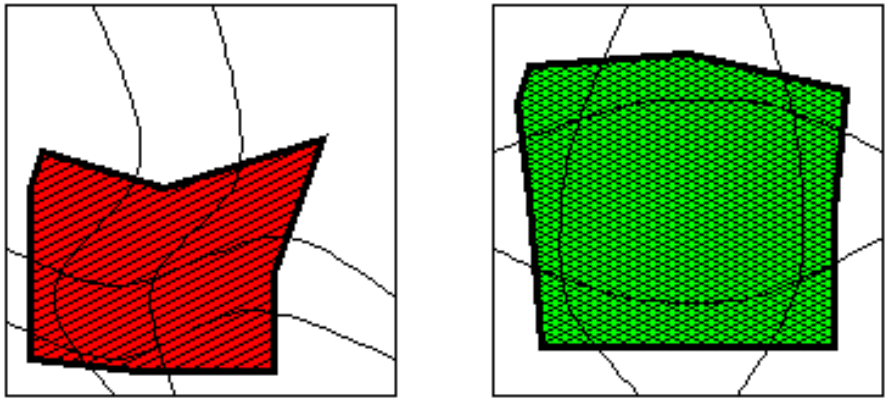


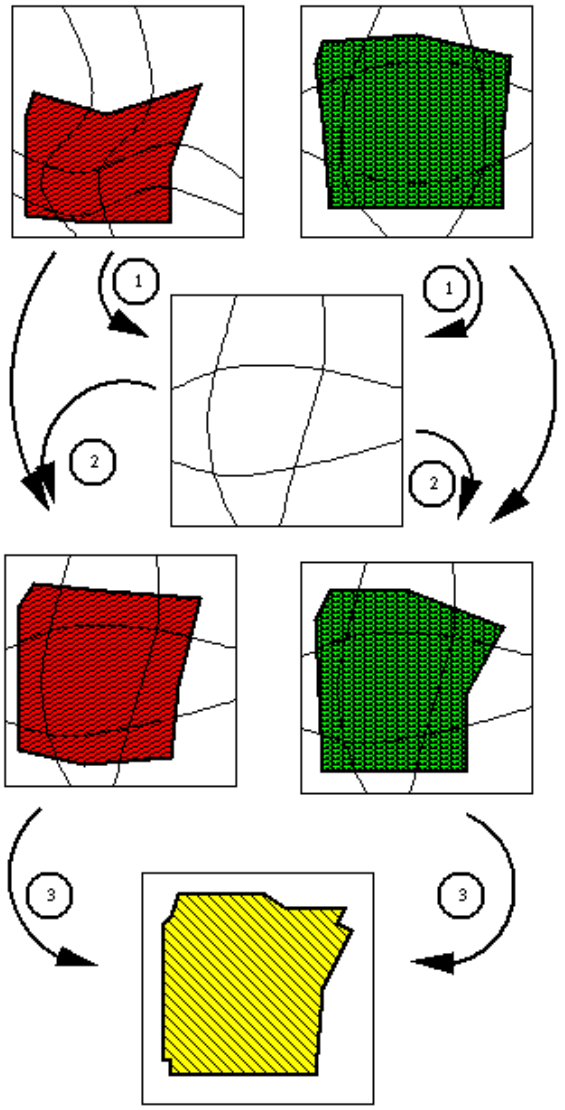
Image morphing

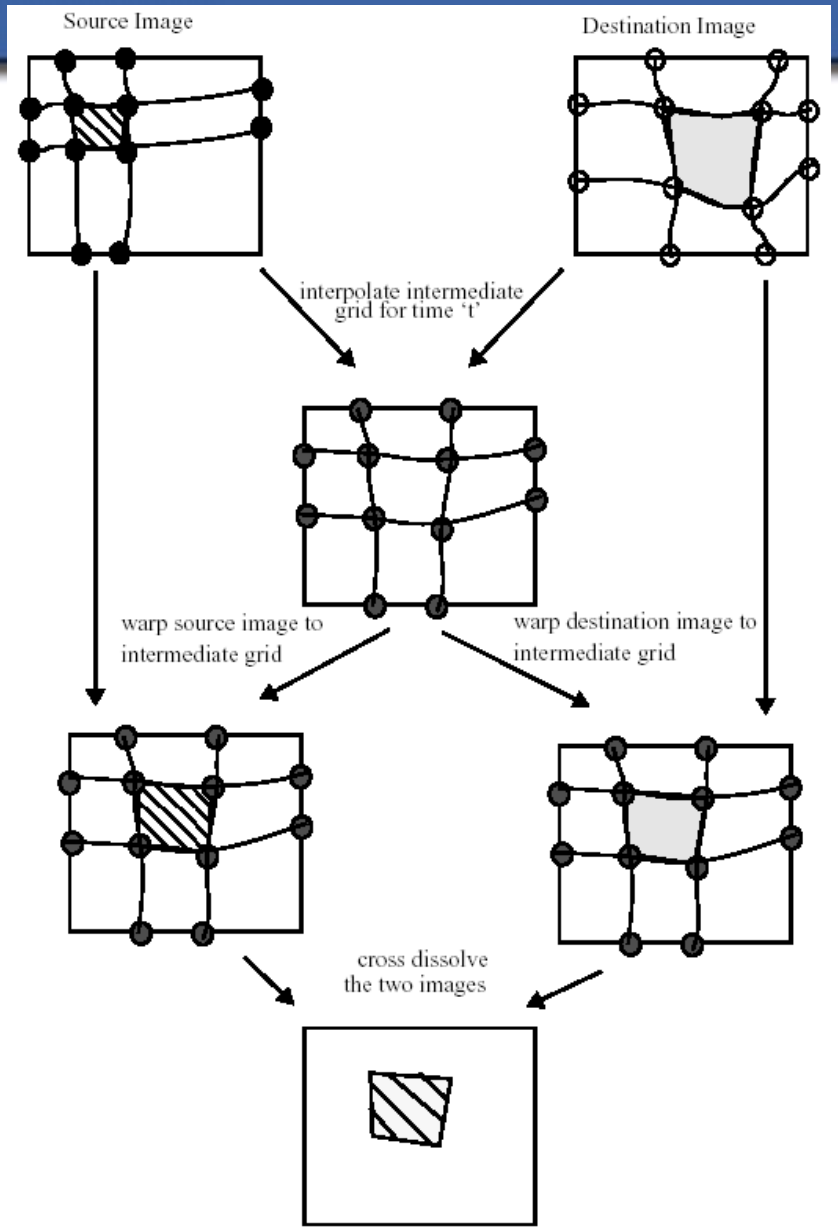
- **Coordinate grid approach**
- **Define curvilinear grid on both images**
  - Take care of grid-to-grid correspondences
  - A curved mesh is then generated using the grid intersection points as control points for an interpolation scheme such as Catmull-Rom splines.



Coordinate Grid Approach

- Interpolate vertices to get intermediate grid
- Create two images by stretching pixels
- Cross dissolve the two




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

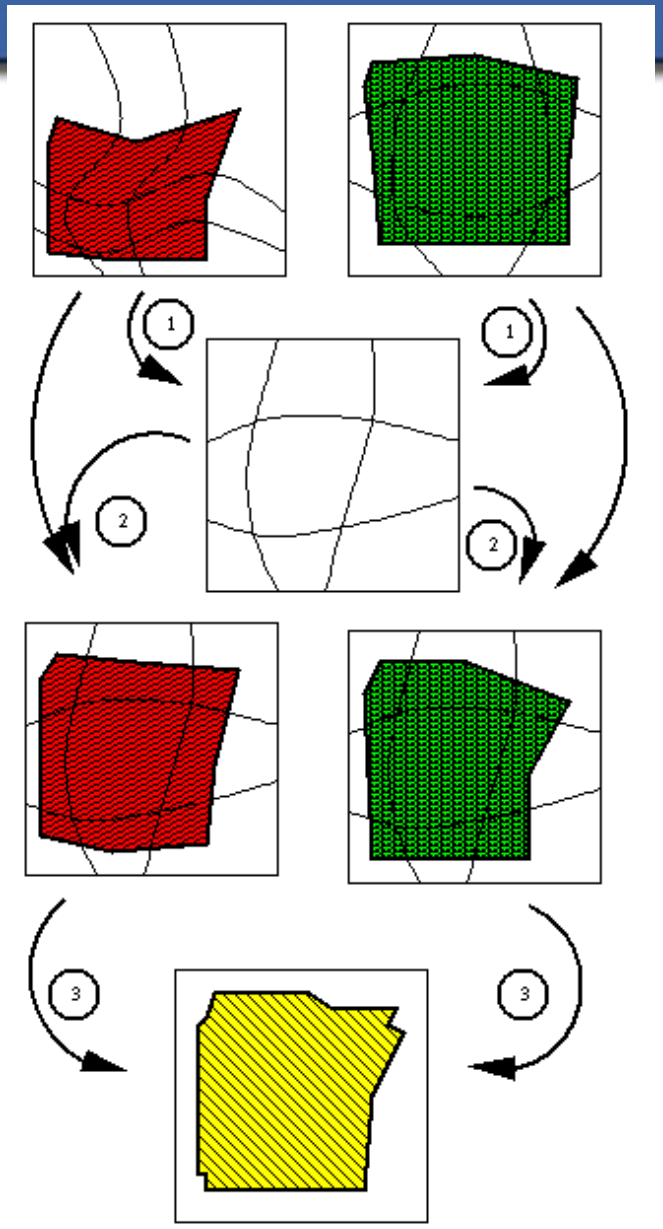
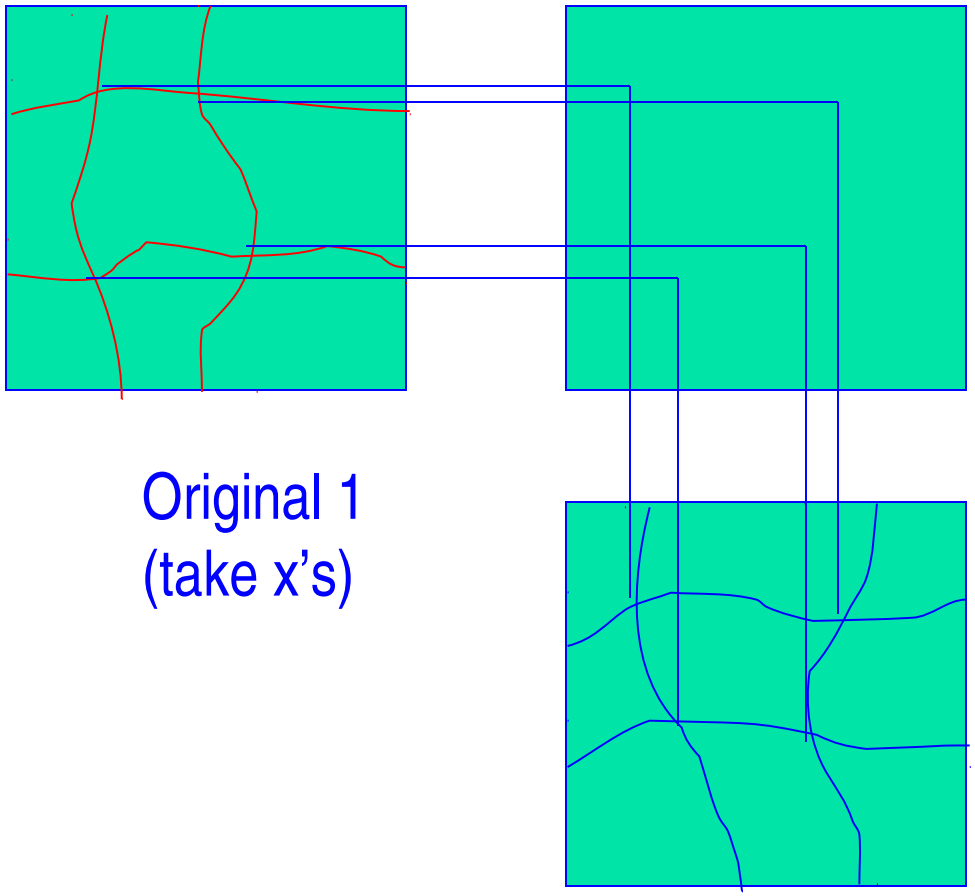

1
1
2
2
3
3

Image morphing details

- Two pass scheme. First pass:
  - Auxiliary grid is created by taking x's from first, y's from intermediate
- For each scanline:
  - Get curve intersections
    - These define separate stretches
  - Get pixel range for each stretch
- Result is passes to the second pass (over columns)

Aux grid

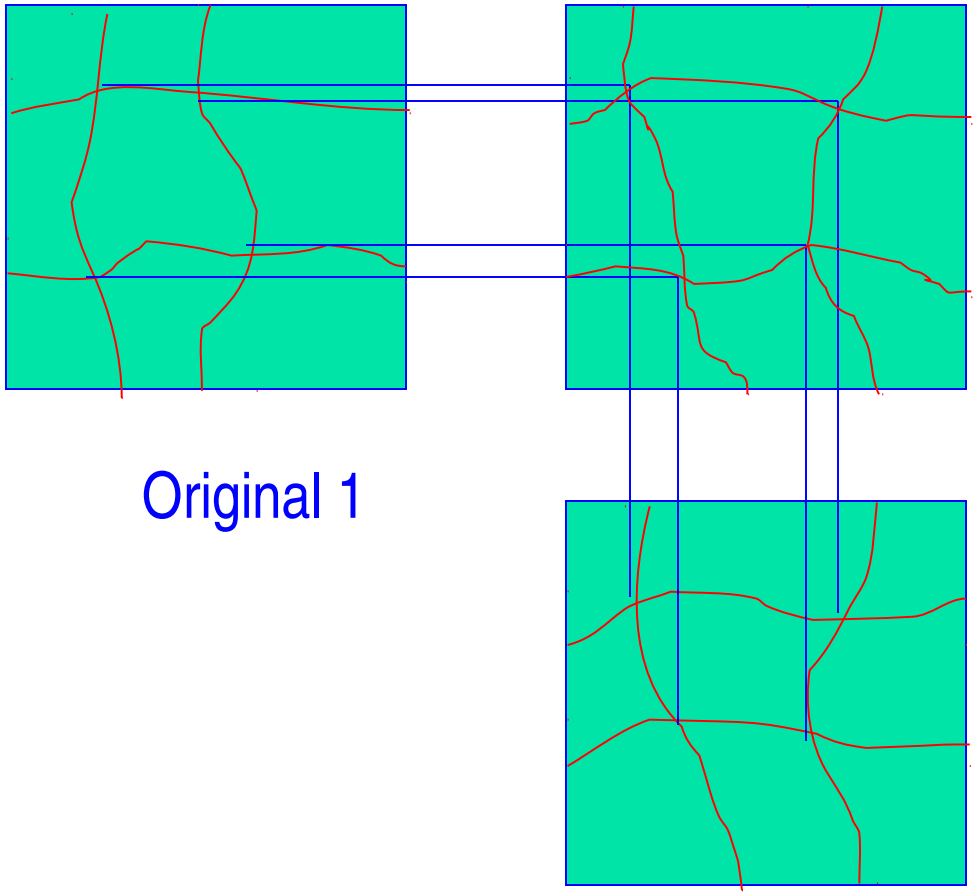


Auxiliary image

Original 1
(take x's)

Intermediate
Image (take y's)

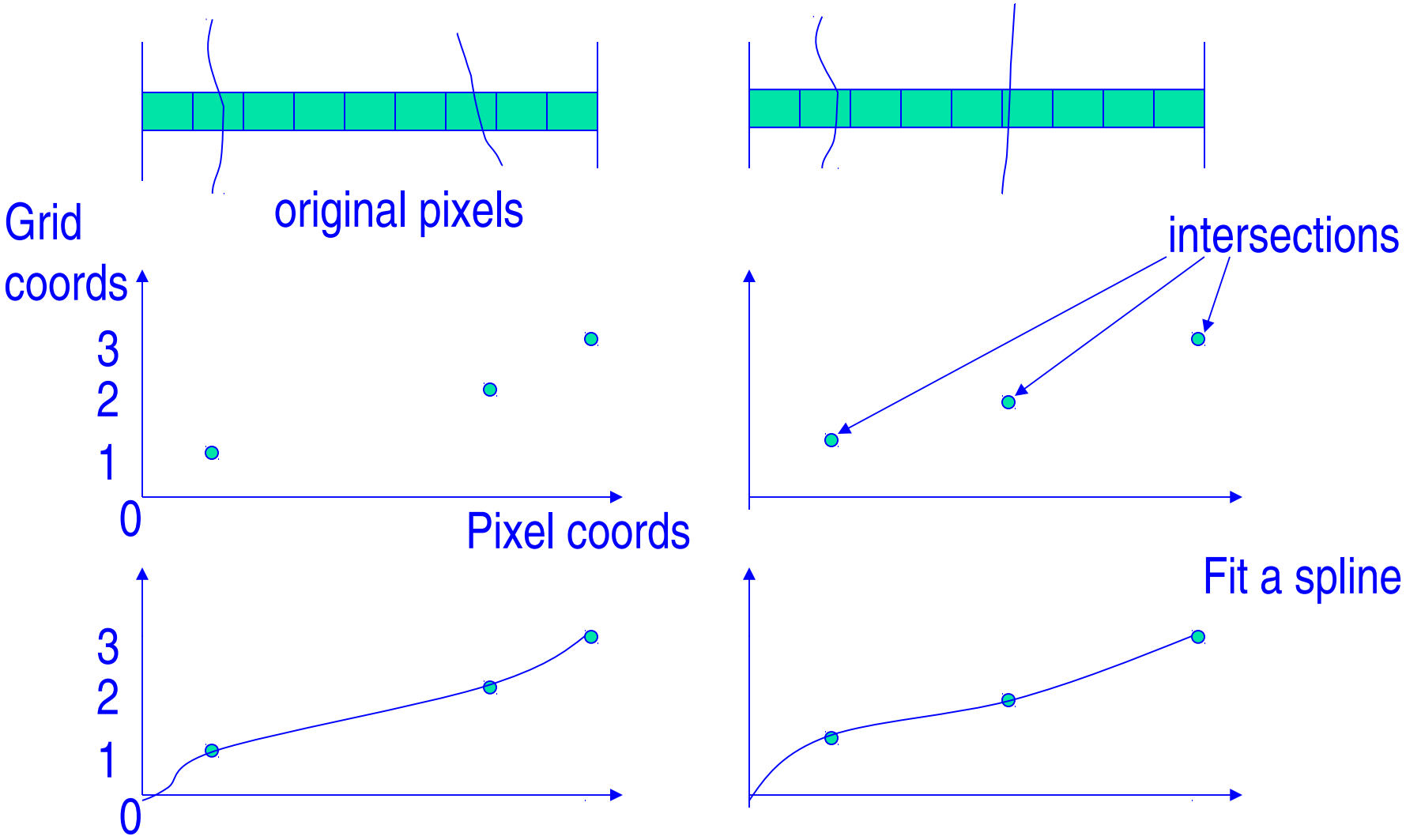
Aux grid



Original 1

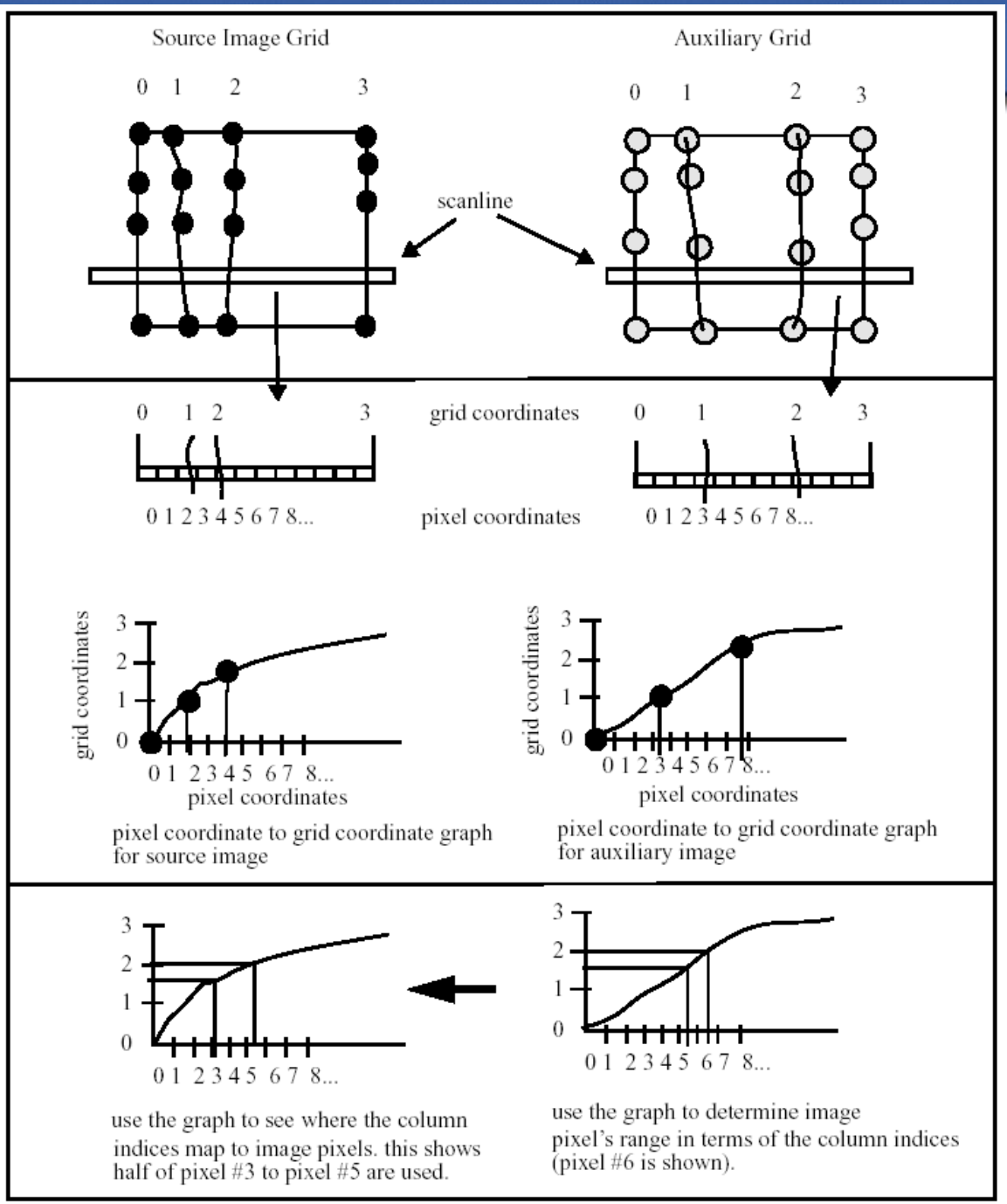
Auxiliary image
Fit a spline
(Catmull-Rom)

Intermediate
Image

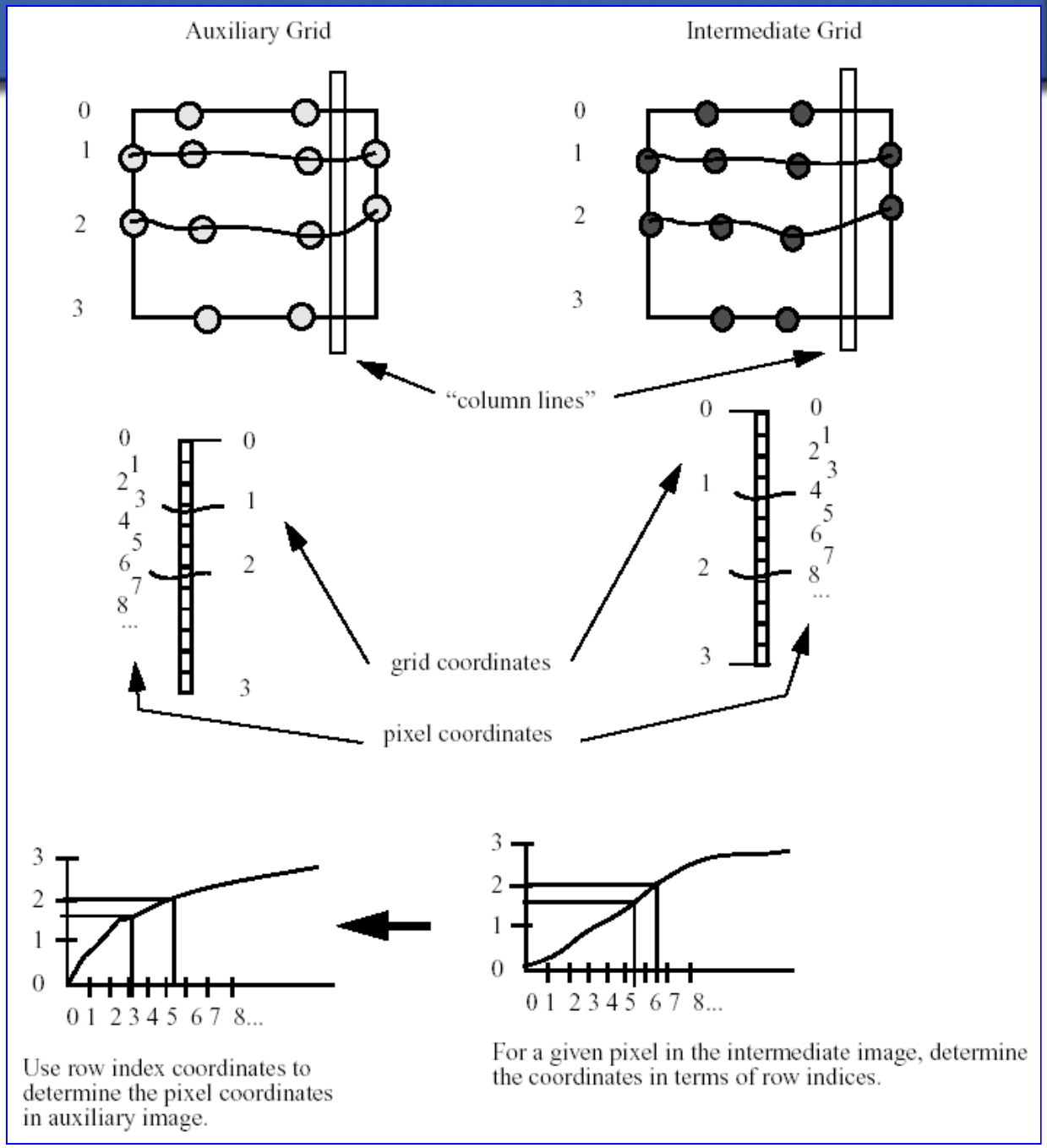

original pixels
Grid coords
3
2
1
0
Pixel coords
intersections
Fit a spline

For given pixel in the auxiliary image, determine the range of pixel coordinates in the source image

For example, pixel 6 of auxiliary grid maps to pixel coordinates 3.1 to 5.5 of image



Establishing the auxiliary pixel range for a pixel of the intermediate image.

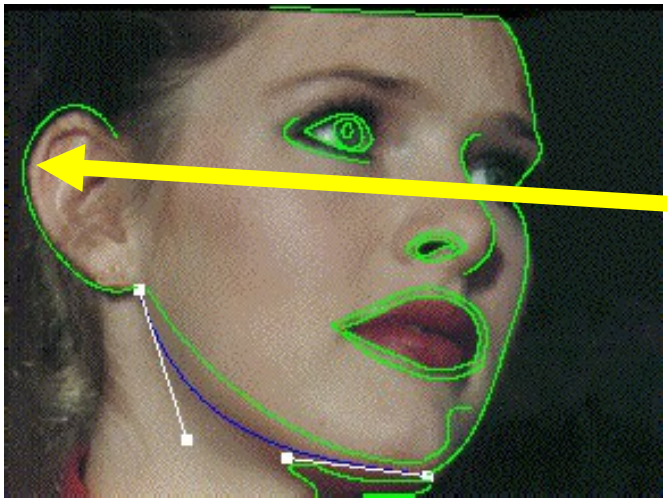


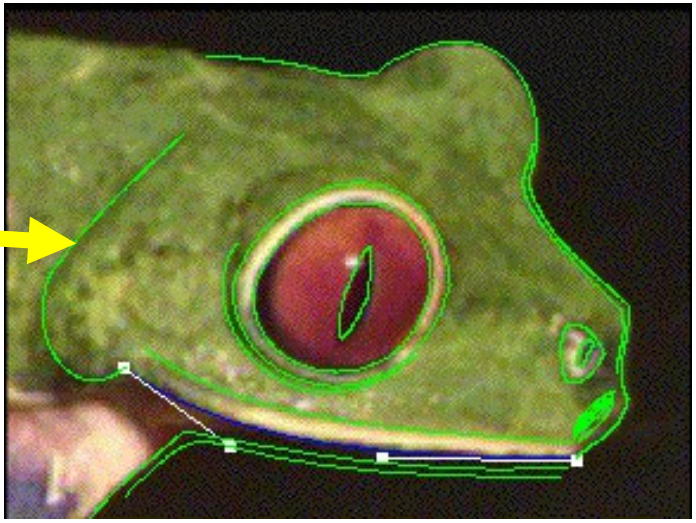
For example, pixel 6 of the intermediate grid maps to pixel coordinates 3.1 to 5.5 of the auxiliary image.

Feature-based morphing

- Want to use just a set of features
  - Rather than complete grid
- Feature = line drawn on the image
- Form intermediate feature image
  - Simple interpolation of features
    - Center/orientation or endpoints
- Map each pixel to each interpolated feature
- Compute associated weight

2D preobrazba dekle-žaba





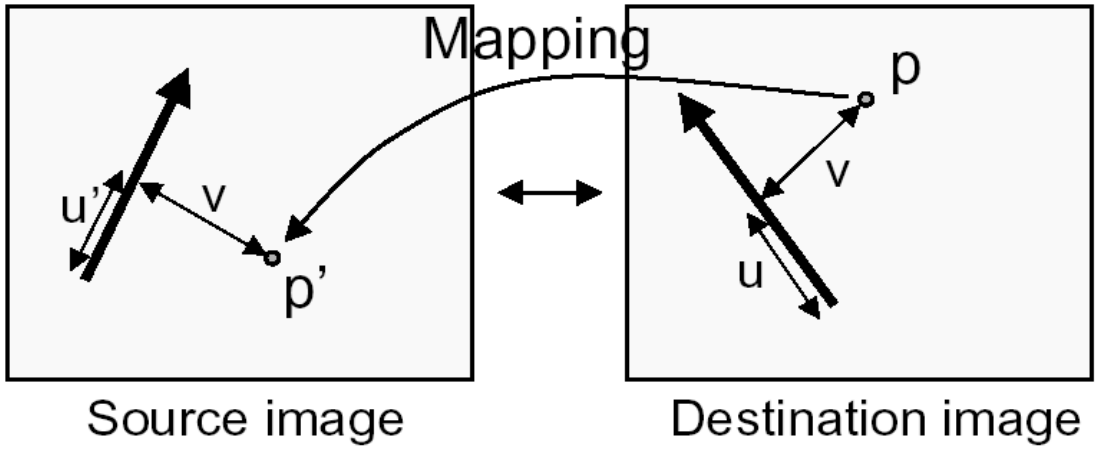
Demo

Feature-Based Warping

- “Which pixel coordinate in the source image do we sample for each pixel in destination image?”
- Correspondence achieved using feature line(s) in source and destination images

Feature-Based Warping

- Beier & Neeley use pairs of lines to specify warp
  - Given p in dst image, where is p' in source image?



u is a fraction

v is a length (in pixels)

Beier & Neeley
SIGGRAPH 92

Feature-Based Warping

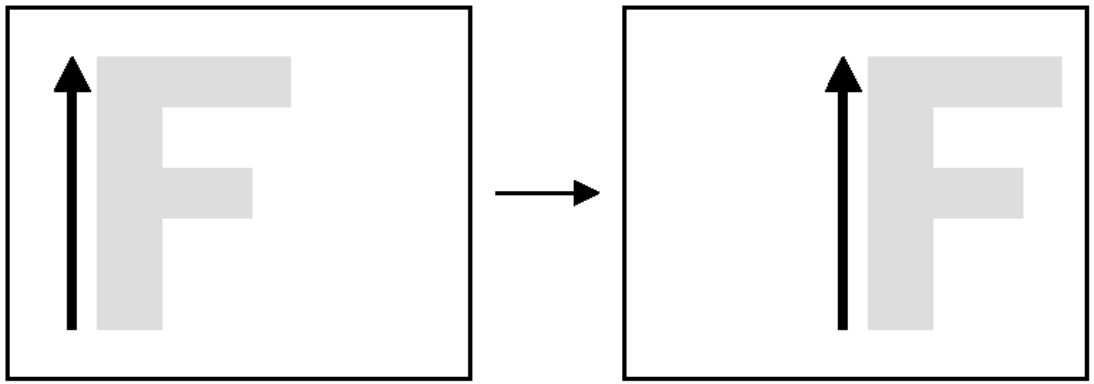
- Transformation with one pair with features(lines)
- Transformation with multiple pairs of features(lines)

Transformation with One Pair of Lines

- A pair of features(lines) – one defined relative to the source image while the other defined relative to destination image
- A pair of lines defines a coordinate mapping from destination pixel (denoted X)to source pixel(denoted X’)

Warping with One Line Pair

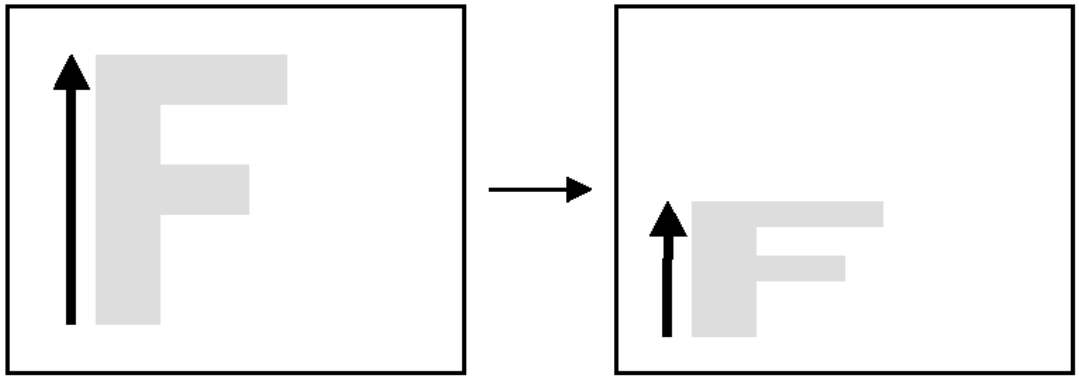
- What happens to the “F”?



Translation!

Warping with One Line Pair

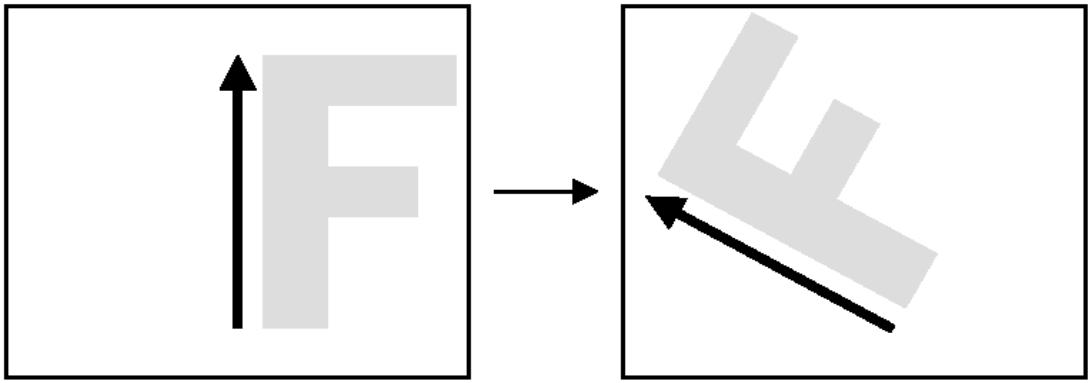
- What happens to the “F”?



Scale!

Warping with One Line Pair

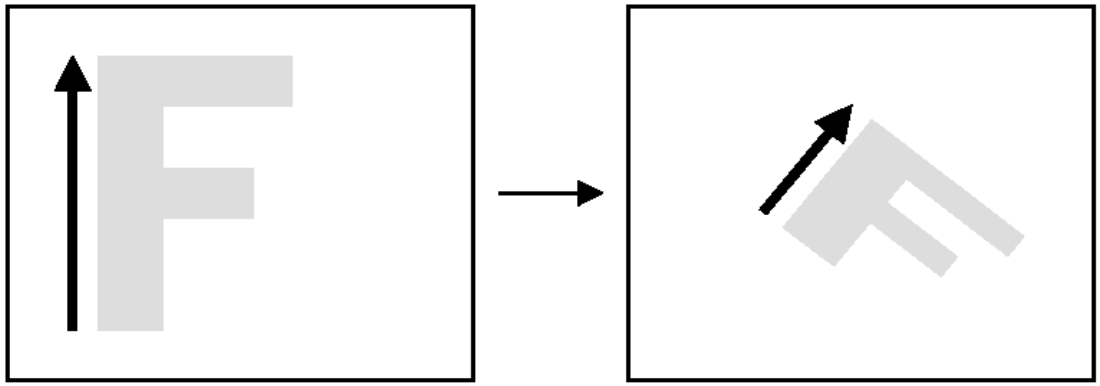
- What happens to the “F”?



Rotation!

Warping with One Line Pair

- What happens to the “F”?

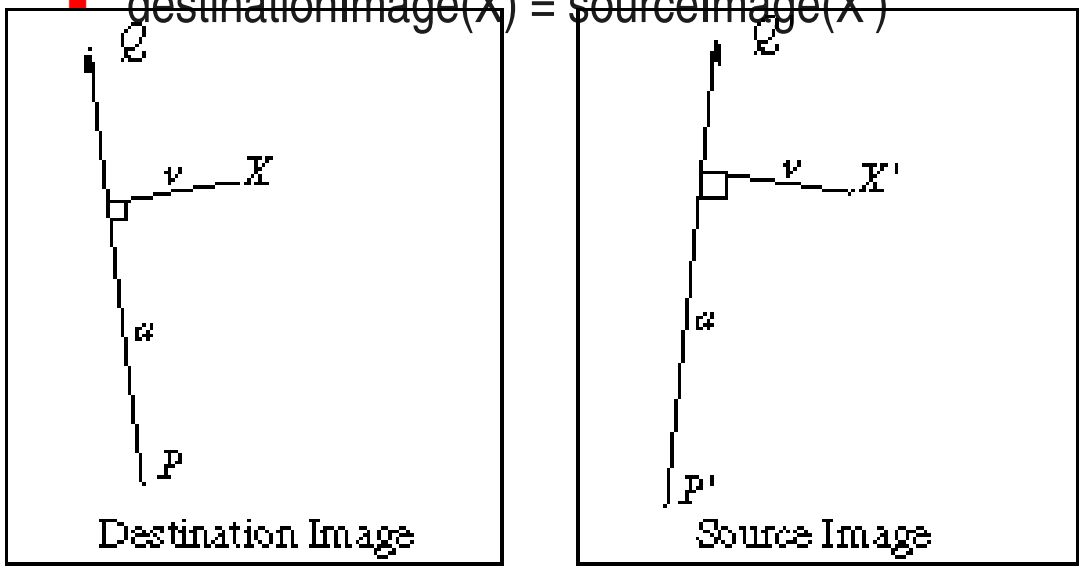


In general, similarity transformations

What types of transformations can’t be specified?

Transformation with One Pair of Lines

- For each pixel X in the destination image
  - Find the corresponding u, v
  - Find the X’ in the source image for that u, v
  - destinationImage(X) = sourceImage(X’)



Transformation with One Pair of Lines

- Equation:

X, X’ represent for pixel of destination and source image. PQ, P’Q’ denote line segments and u, v stand for scalars

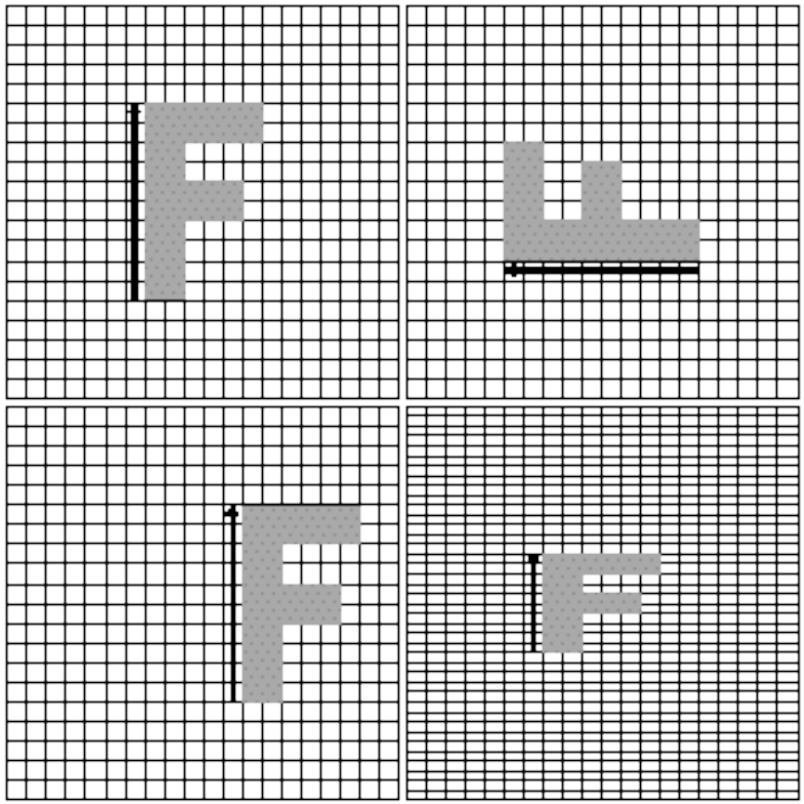
$u = (X-P)\bullet(Q-P) / \|Q-P\|^2$

$v = (X-P)\bullet \text{Perpendicular}(Q-P) / \|Q-P\|$

$X' = P' + u(Q'-P') + v*\text{Perpendicular}(Q'-P') / \|Q'-P'\|$

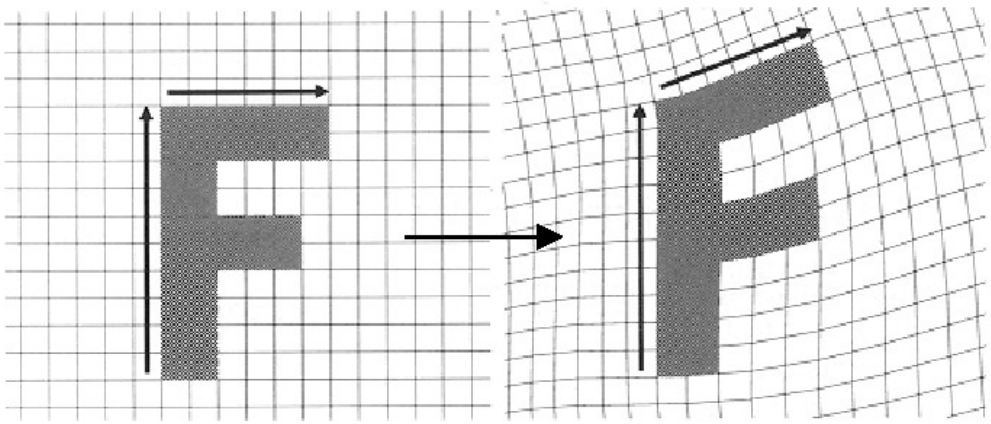
Transformation with One Pair of Lines

- Original image (UL), rotated image (UR), translated image(LL), scaled image (LR)



Warping with Multiple Line Pairs

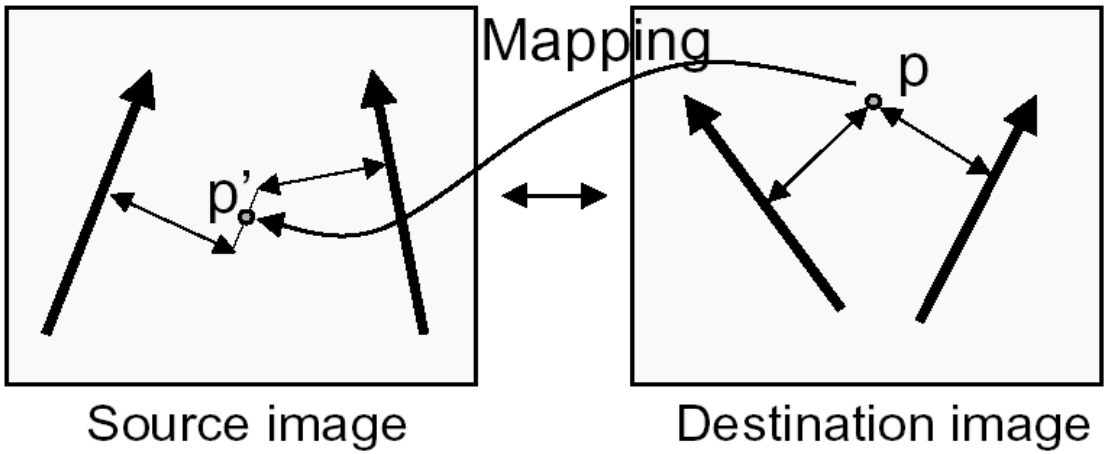
- Use weighted combination of points defined by each pair of corresponding lines



Beier & Neeley, Figure 4

Warping with Multiple Line Pairs

- Use weighted combination of points defined by each pair of corresponding lines



p' is a weighted average

Weighting Effect of Each Line Pair

- To weight the contribution of each line pair, Beier & Neeley use:

$$weight[i] = \left(\frac{length[i]^p}{a + dist[i]} \right)^b$$

Where:

- *length[i]* is the length of L[i]
- *dist[i]* is the distance from X to L[i]
- *a, b, p* are constants that control the warp

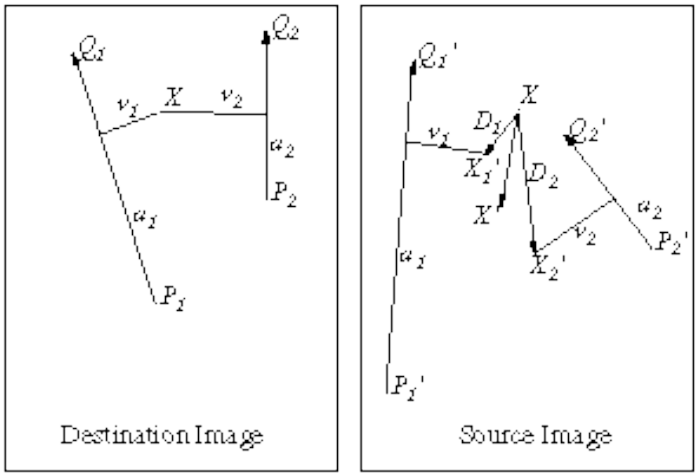
- a – smoothness of warping
- b – falloff of strength with distance
- p – rewarding longer lines

Transformation with Multiple Pairs of Lines

- For each pixel X in destination
  - DSUM=(0,0)
  - weightsum=0
  - For each line Pi Qi
    - Calculate u,v based on Pi Qi
    - Calculate X'I based on u,v and Pi' Qi'
    - Calculate displacement Di=Xi'-Xi
    - Calculate weight
    - DSUM+=Di*weight; weightsum+=weight
  - X' = X+ DSUM/weightsum
  - destinationImage(X) = SourceImage(X')

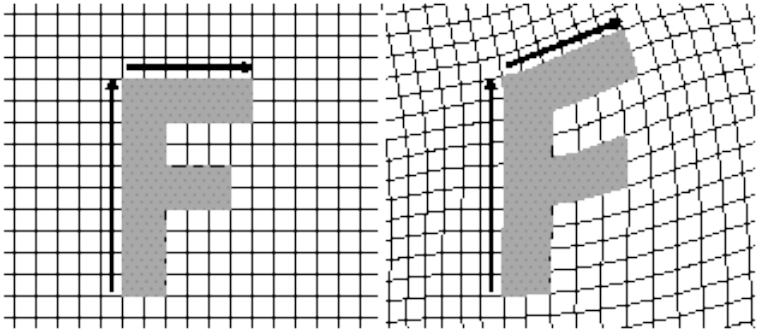
Transformation with Multiple Pairs of Lines

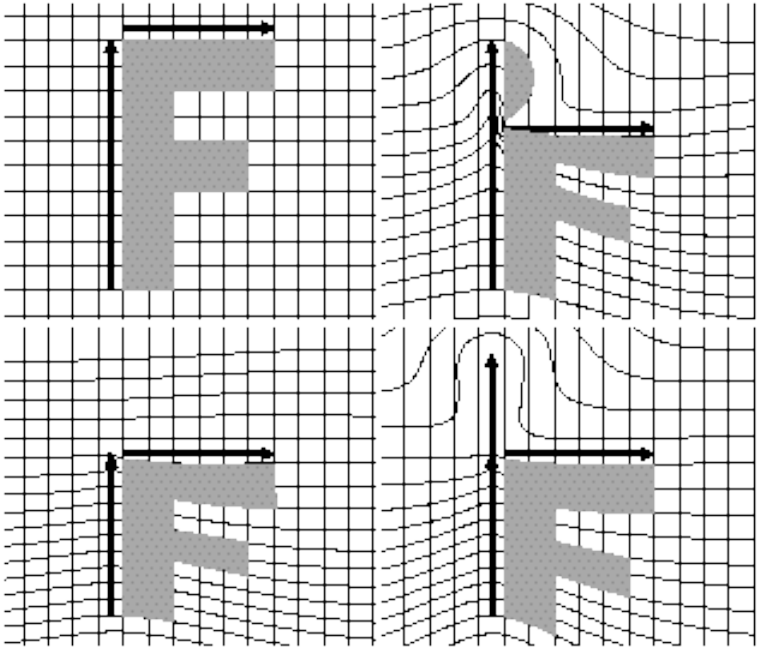
- "Lines" are actually line segments
- Distance from a pixel to a line is
  - abs(v) if 0<u<1
  - Distance from P if u<0
  - Distance from Q if u>0



Transformation with Multiple Pairs of Lines

- Not possible to do uniform scaling or shear





Transformation with Multiple Pairs of Lines

- Advantages
  - Expressive
  - Adding control points is easy
- Disadvantages
  - All line segments need to be referenced for each pixel
  - Line segments have global impact

Warping Pseudocode

```
WarpImage(Image, L'[…], L[…])
begin
    foreach destination pixel p do
        psum = (0,0)
        wsum = 0
        foreach line L[i] in destination do
            p'[i] = p transformed by (L[i],L'[i])
            psum = psum + p'[i] * weight[i]
            wsum += weight[i]
        end
        p' = psum / wsum
        Result(p) = Image(p')
    end
end
```

Warping Pseudocode

```
GenerateAnimation(Image0, L0[…], Image1, L1[…])
begin
    foreach intermediate frame time t do
        for i = 1 to number of line pairs do
            L[i] = line t-th of the way from L0 [i] to L1 [i]
        end
        Warp0 = WarpImage(Image0, L0, L)
        Warp1 = WarpImage(Image1, L1, L)
        foreach pixel p in FinalImage do
            Result(p) = (1-t) Warp0 + t Warp1

    end
end
```

Morphing between two images

- A morph operation blends between two images I0 and I1. And each intermediate frame I is defined by creating a new set of line segments by interpolating lines(features) positions

Morphing between two images

- First way of interpolating is just to interpolate the endpoints – a rotating line would shrink
- Second way of interpolating is to take the center position and orientation of each line – not very obvious to user
- In any case, let user see the interpolation position is a great help for designing

Morphing between two images(cont')

- Images

Figure 7

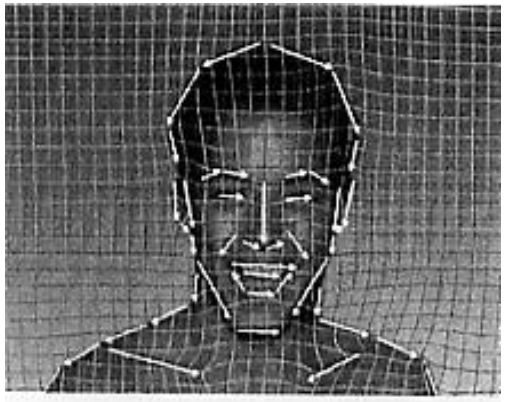
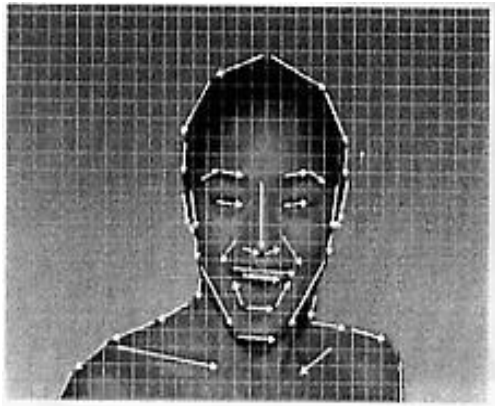


Figure 8



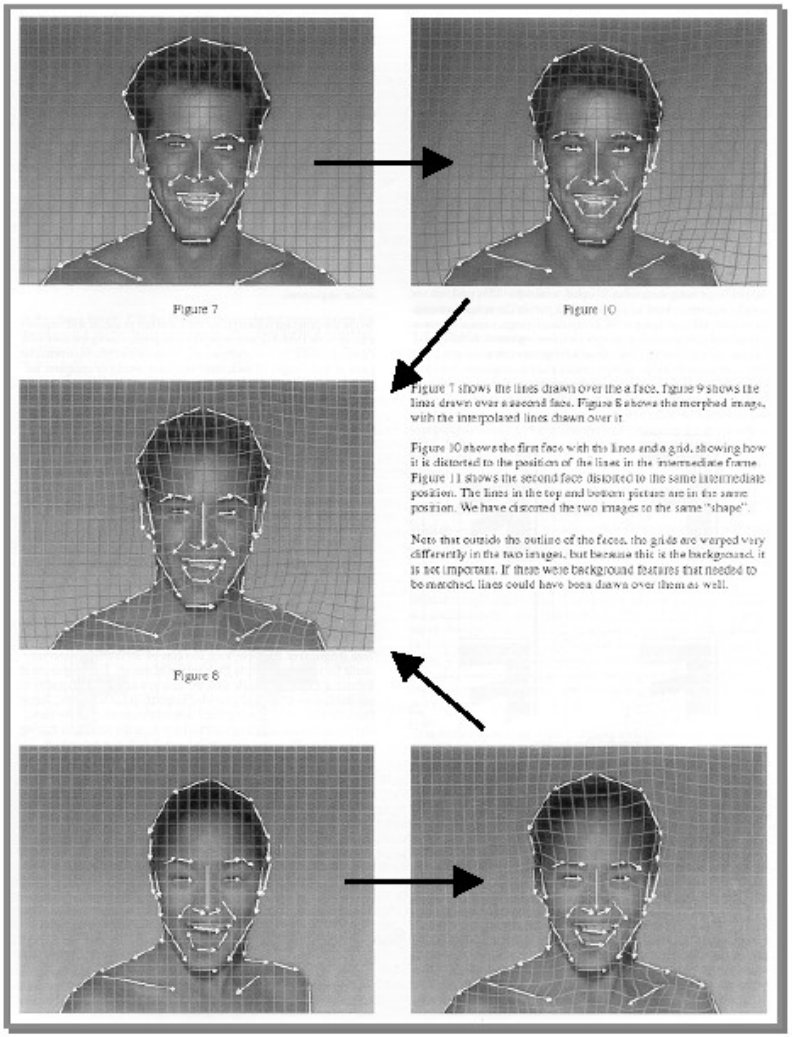
40

Figure 9

Beier & Neeley Example

Image$_0$

Warp$_0$

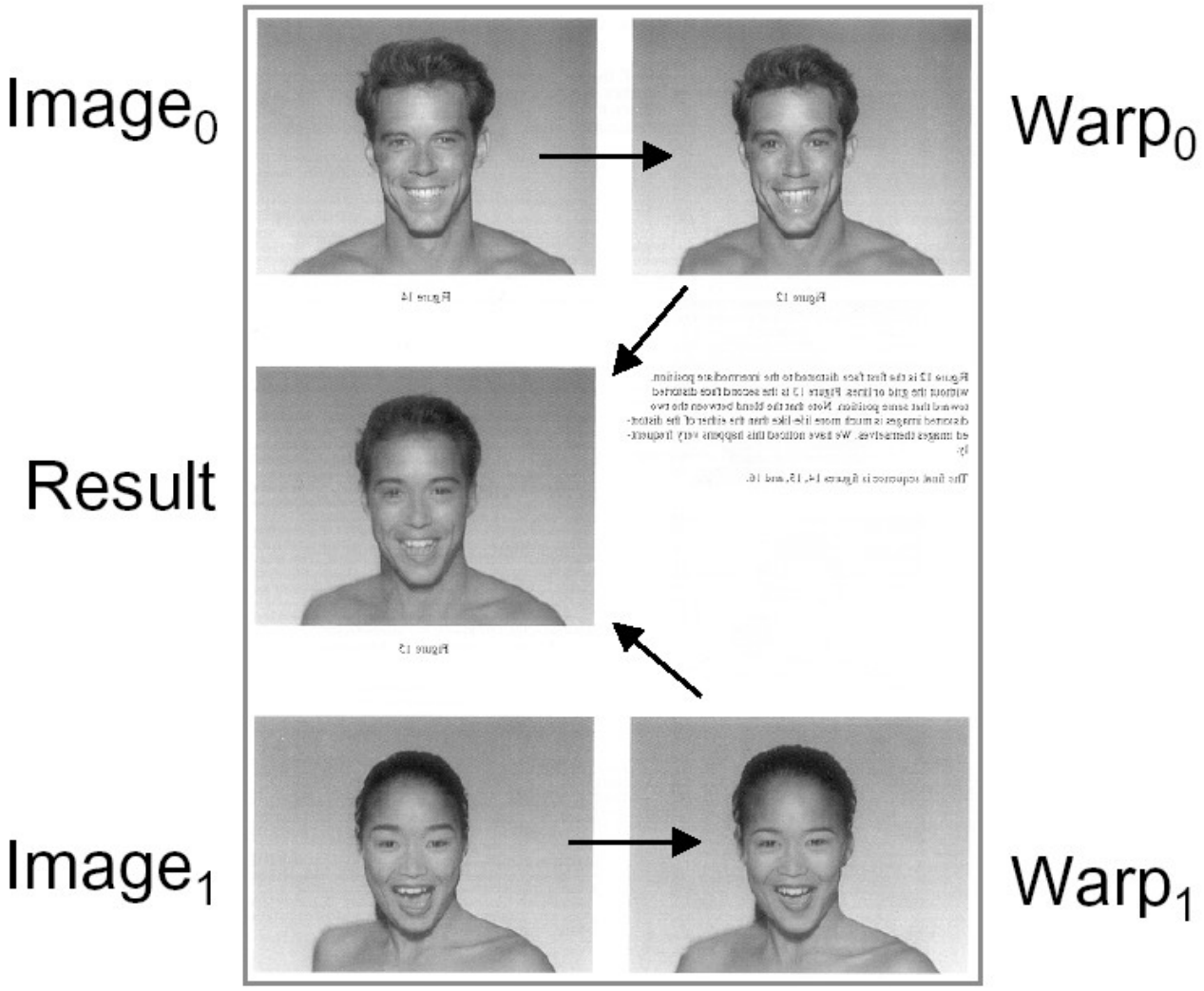


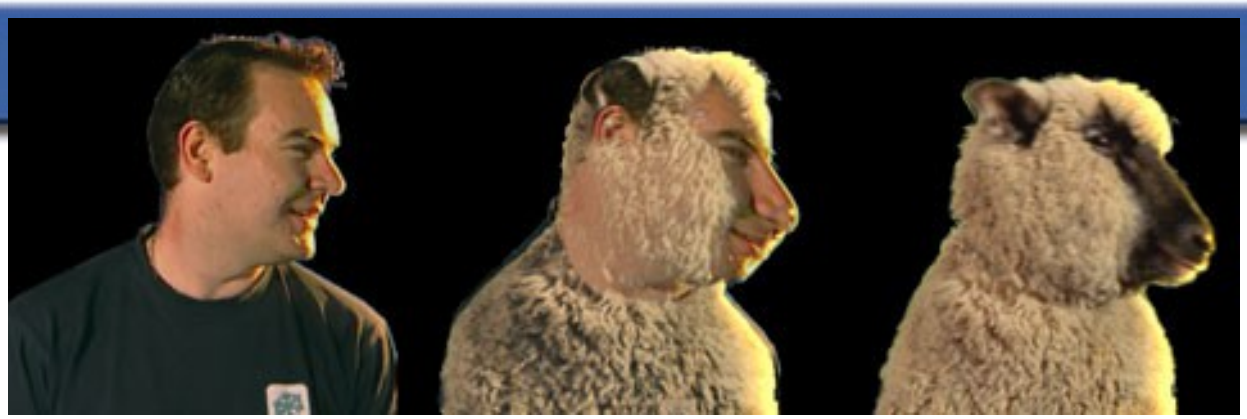
Result

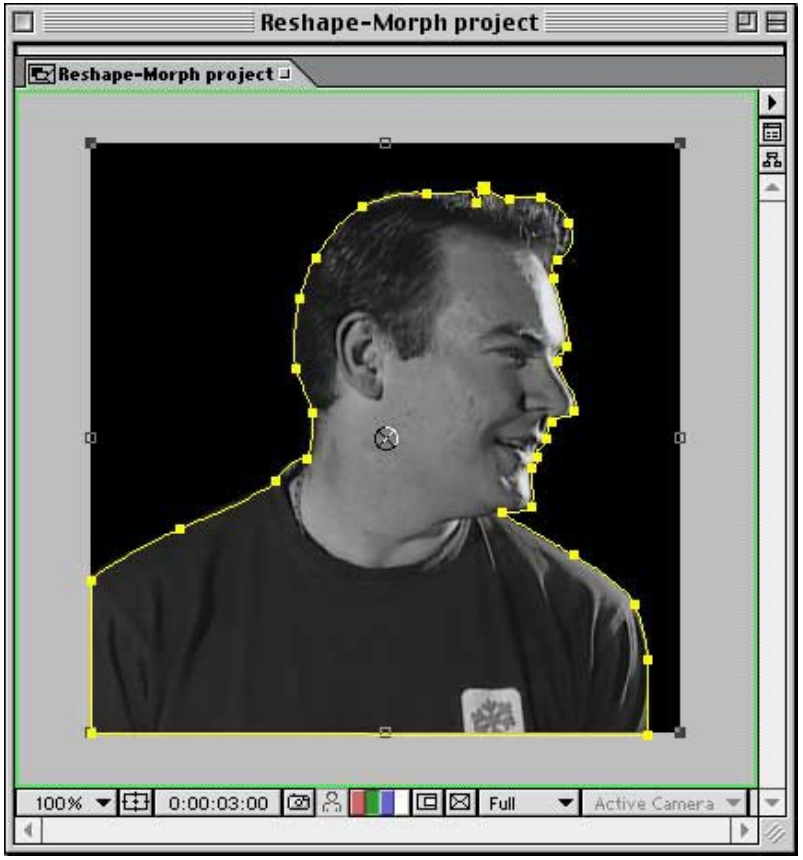
Image$_1$

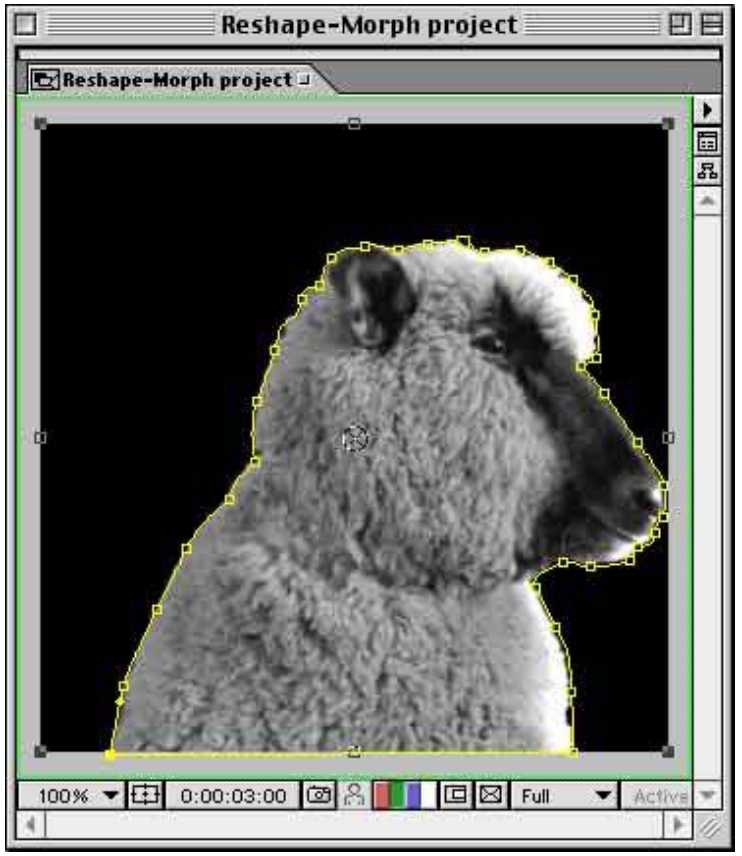
Warp$_1$

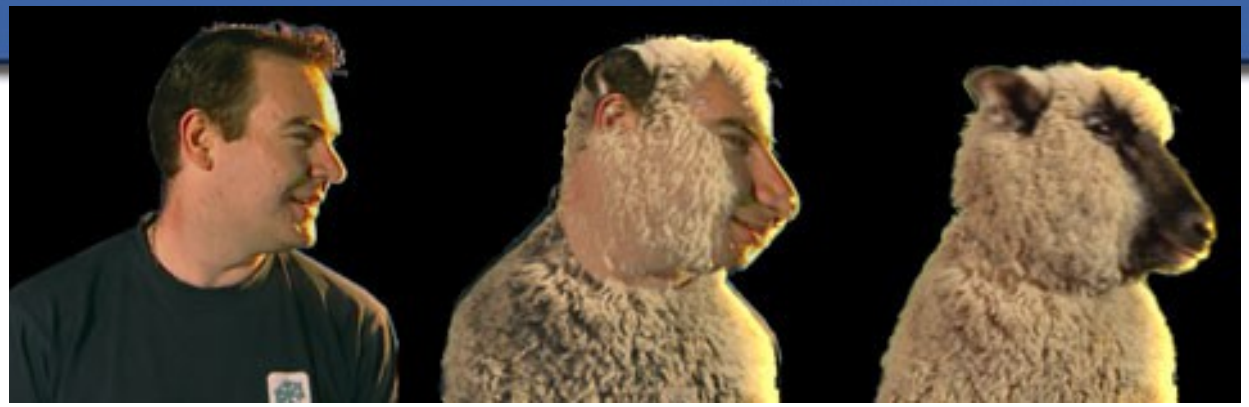
Beier & Neeley Example

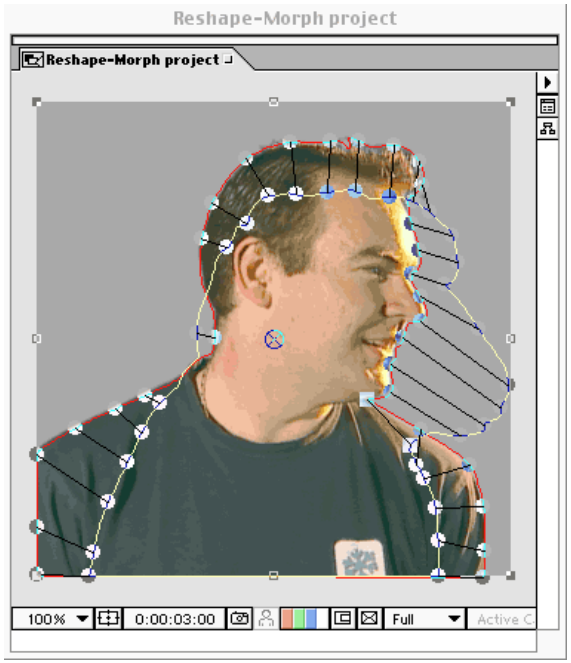


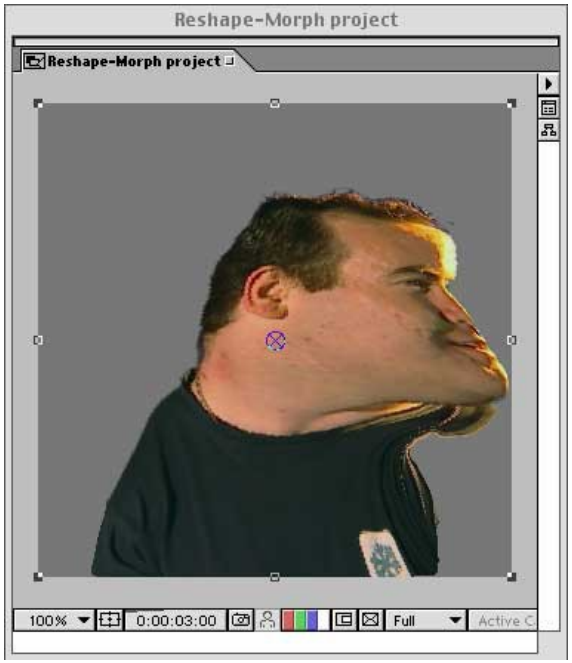


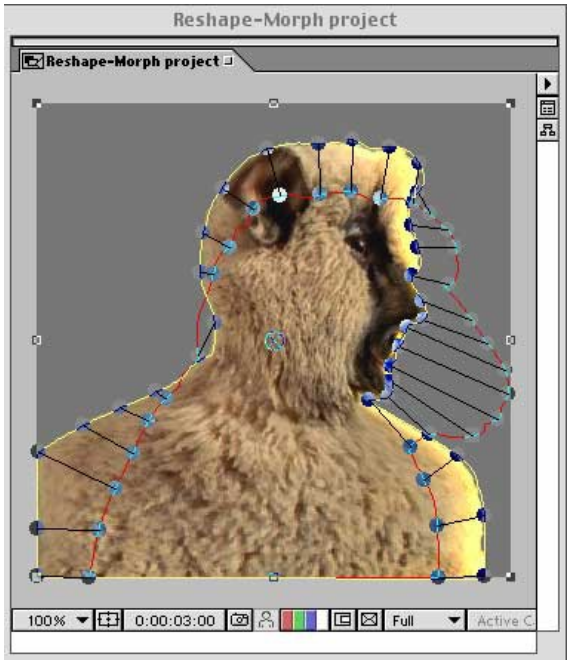

Reshape-Morph project
Reshape-Morph project
100%
0:00:03:00
Full
Active Camera


Reshape-Morph project
Reshape-Morph project
100%
0:00:03:00
Full
Active




Reshape-Morph project
Reshape-Morph project
100%
0:00:03:00
Full


Reshape-Morph project
Reshape-Morph project
100%
0:00:03:00
Full


Reshape-Morph project
Reshape-Morph project
100%
0:00:03:00
Full

Advantages and Disadvantages

- Advantages:

  Much more expressive

  To control features is very natural

- Disadvantages

  Speed

  Control

  “Ghostbusting” – to remove unexpected interpolation generated pixels

Beier-Neely Algorithm

- **Step I** : Interpolating the lines:

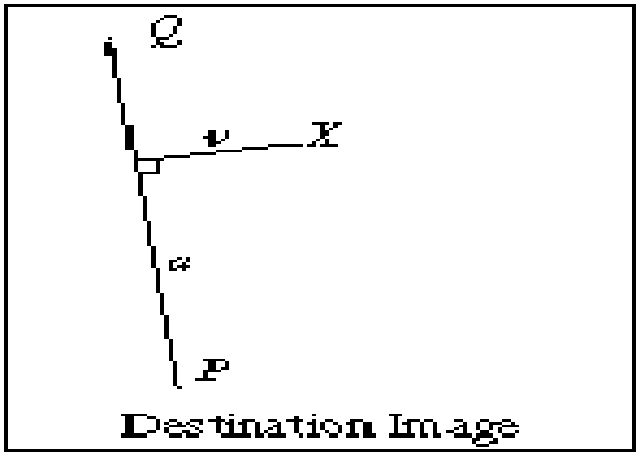
→ Interpolate the coordinates of the end points of every pair of lines.

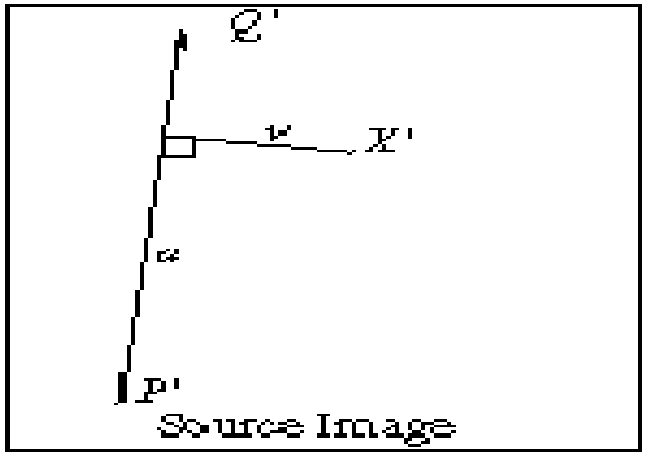
- **Step II** : Warping the Images

→ Each of the source images has to be deformed towards the needed frame.

→ The deformation works pixel by pixel is based on the reverse mapping. This algorithm is called *Beier-Neely* Algorithm.

Beier-Neely Algorithm





Idea is to:

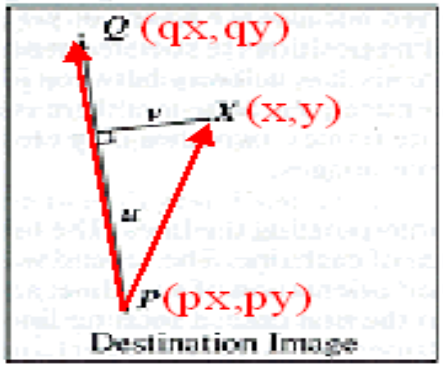
1) Compute position of pixel X in destination image relative to the line drawn in destination image.
   (x,y) → (u,v)

Beier-Neely Algorithm

2) Compute coordinates of pixel in source image whose position relative to the line drawn in source image is (u,v).

(u,v) →(x',y')

$$(x, y) \rightarrow (u, v)$$



$$u = \frac{(X-P).(Q-P)}{\|Q-P\|^2}$$

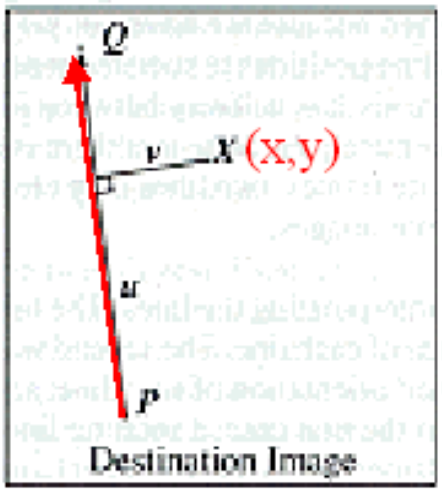
$$= (x-px)(qx-px) + (y-py)(qy-py)$$

$$= (qx-px)^2 + (qy-py)^2$$

$$v = \frac{(X-P).\text{Perpendicular}(Q-P)}{\|Q-P\|}$$

$$= (qy-py, -qx+px)$$

Beier-Neely Algorithm



$$(qx - px)^2 + (qy - py)^2$$

$$= \text{length}$$

$$\text{weight} = \left(\frac{\text{length}^{p}}{a + \text{dist}} \right)^{b}$$

$$= \text{dist}$$

controls influence of line for points near it

abs(v) if $0 < u < 1$

distance of (x, y) from P if $u < 0$

distance of (x, y) from Q if $u > 1$

Beier-Neely Algorithm

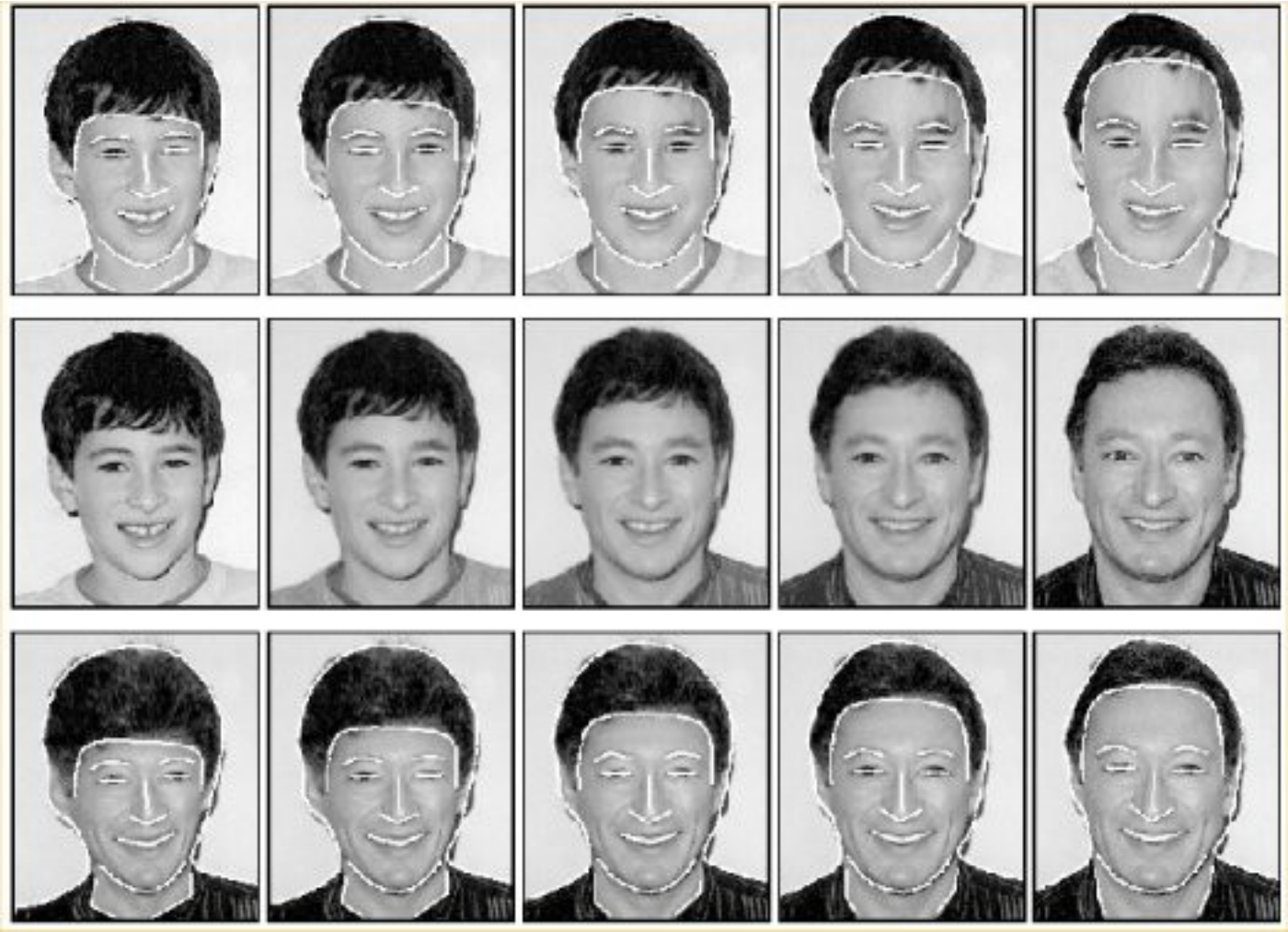
For each pixel X=(x,y) in the destination image
DSUM=(0,0) , weightsum=0
for each line(Pi, Qi)
calculate(ui,vi) based on Pi, Qi
calculate (xi', yi') based on u,v and Pi, Qi
calculate displacement
Di = Xi' – X for this line
compute weight for line(Pi,Qi)

DSUM+=Di*weight

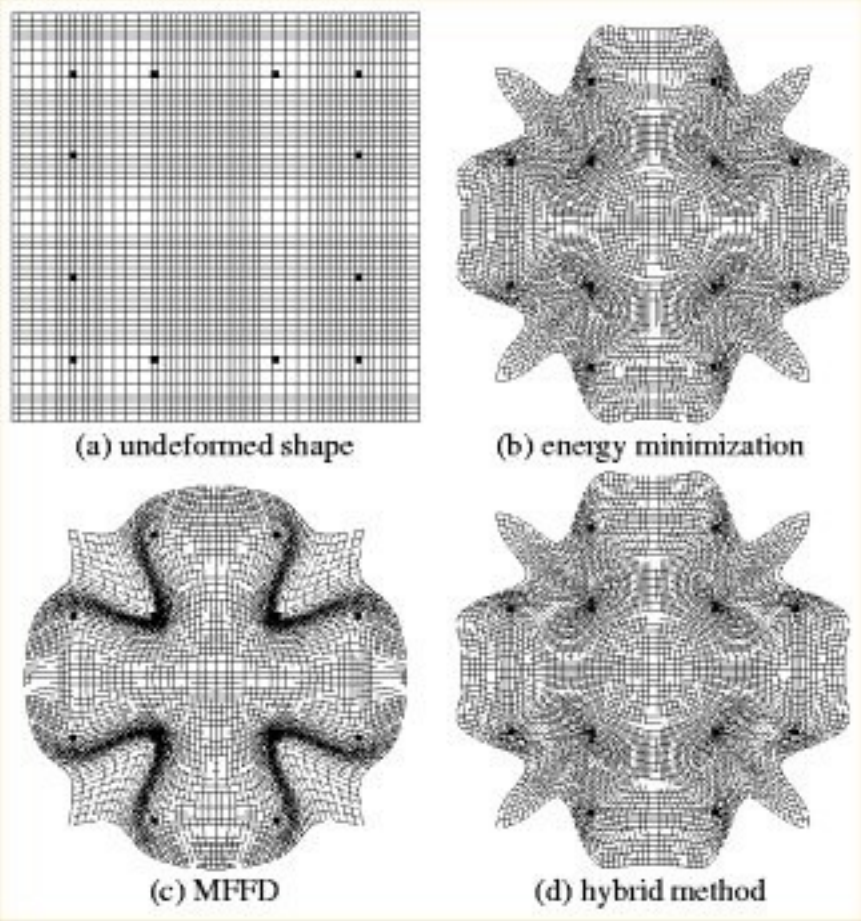
weightsum+=weight

(x'y') = (x,y)+DSUM/weightsum
color at destination pixel(x,y) = color at source pixel(x'y')

Multilevel Free-Form Deformations



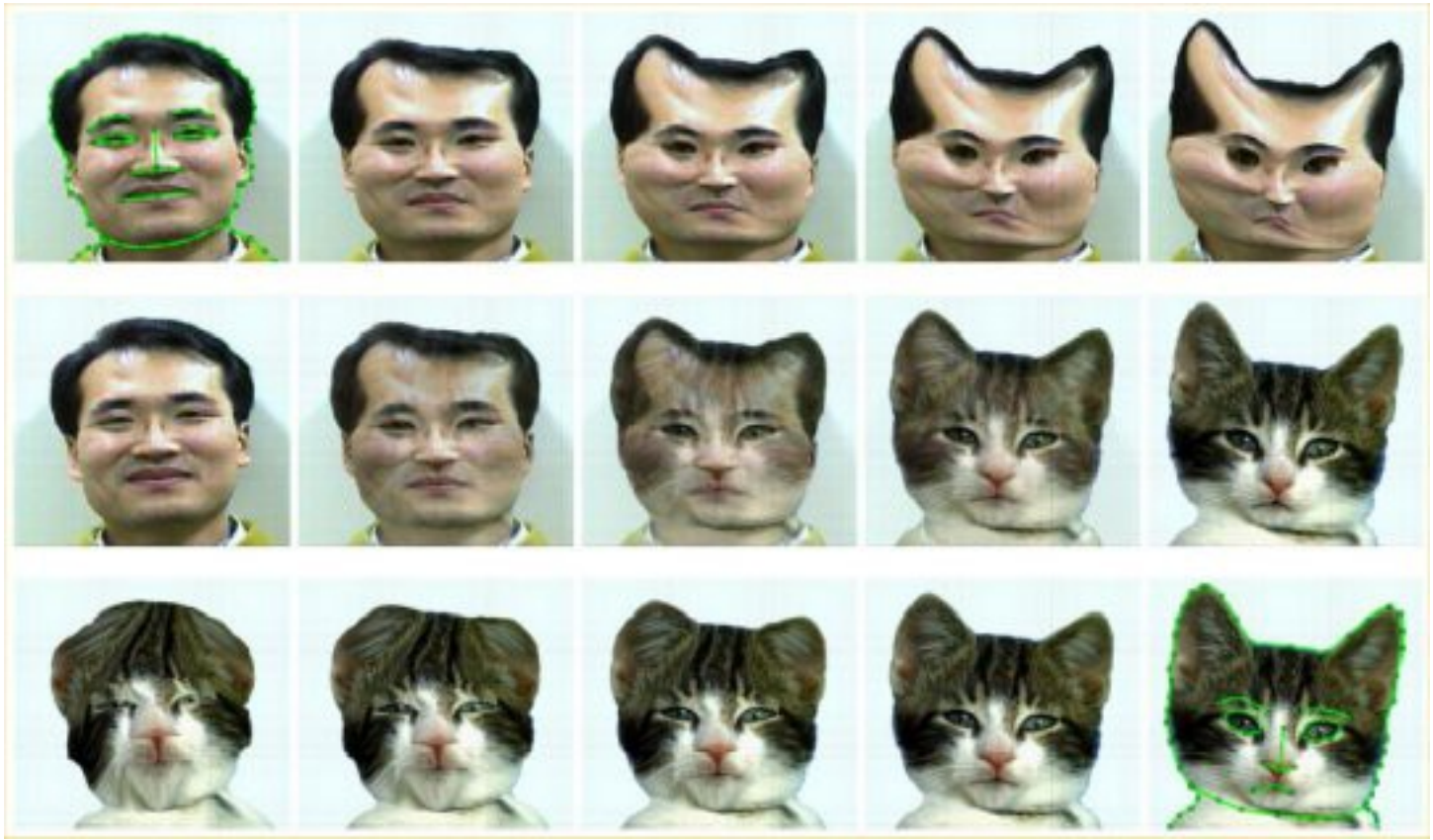
MFFD & energy minimization



Transition Control

- Uniform vs. non-uniform metamorphosis
- Accelerated transition for some features but not all may improve the results dramatically

Transition Control – Uniform Metamorphosis



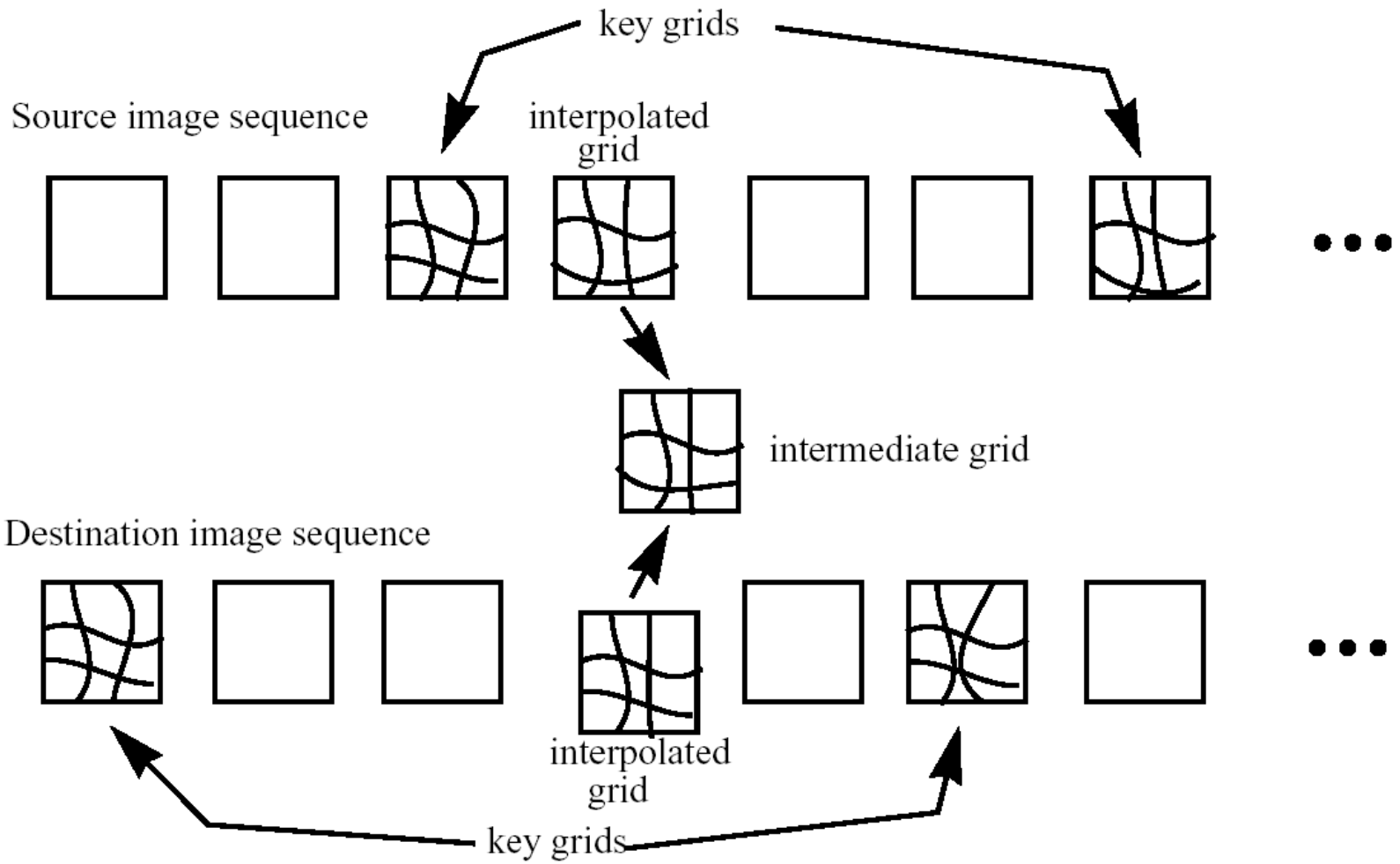
Transition Control – Non-uniform Metamorphosis

Animated Sequences

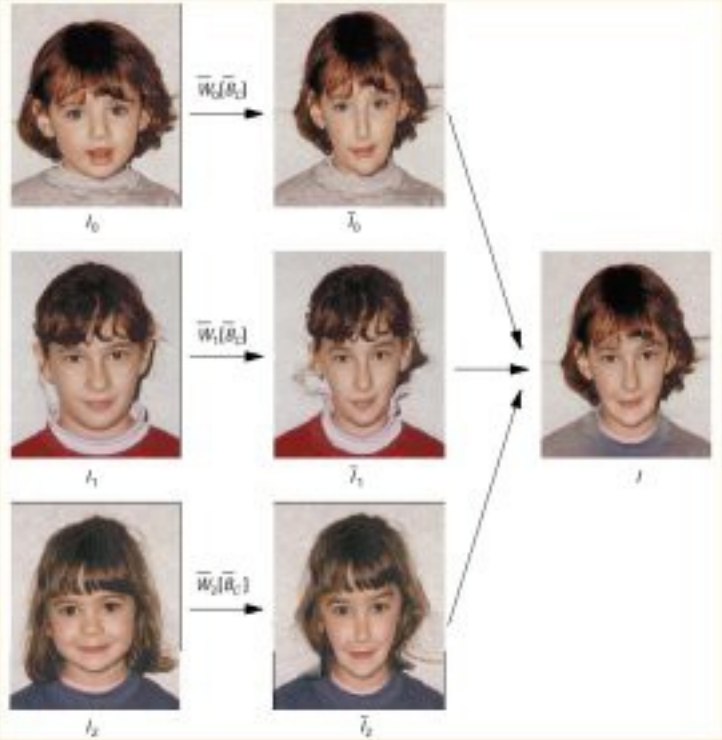
- Morphing two sequences of live action, rather than just two still images
- Mark all features in key frames
- Interpolate the features between key frames
- Do image metamorphosis on set of pair of images… (Black & White video)

Image Morphing

- Morphing for animated sequences:
  - Define grids on key frames
  - Grids for intermediate frames are created
    - Simple x,y interpolation of intersection points
  - Image morphing performed frame-by-frame

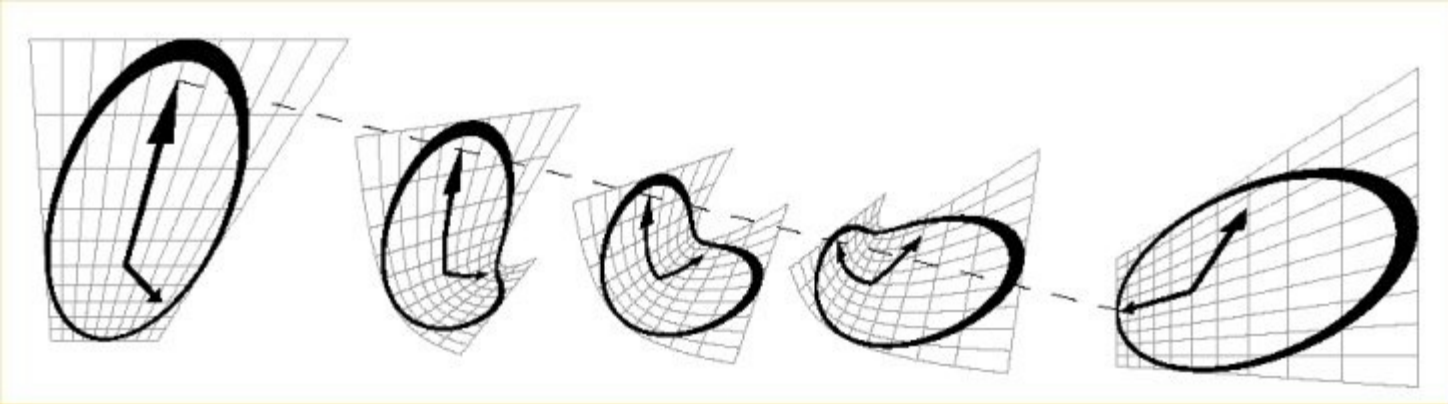

key grids
Source image sequence
interpolated grid
intermediate grid
Destination image sequence
interpolated grid
key grids

Polymorph



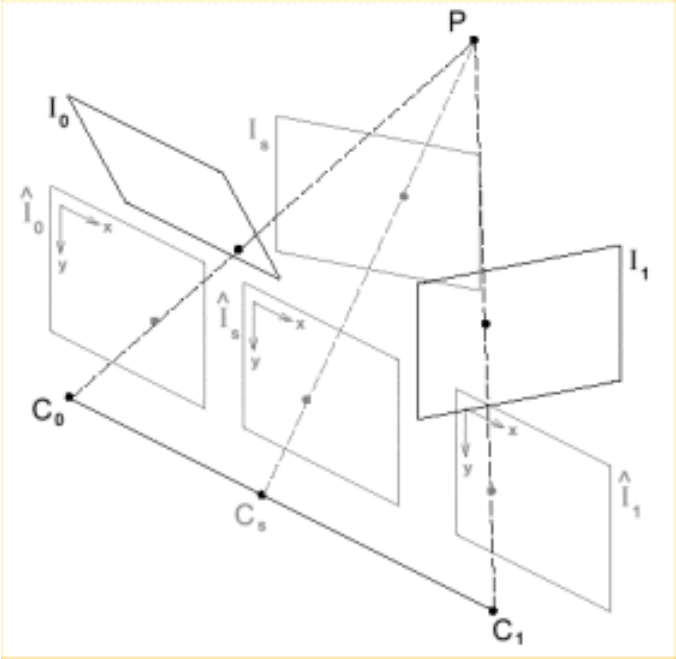
View Morphing

- 2-D image morphing does not necessarily preserve shape of rigid bodies in intermediate images



View Morphing

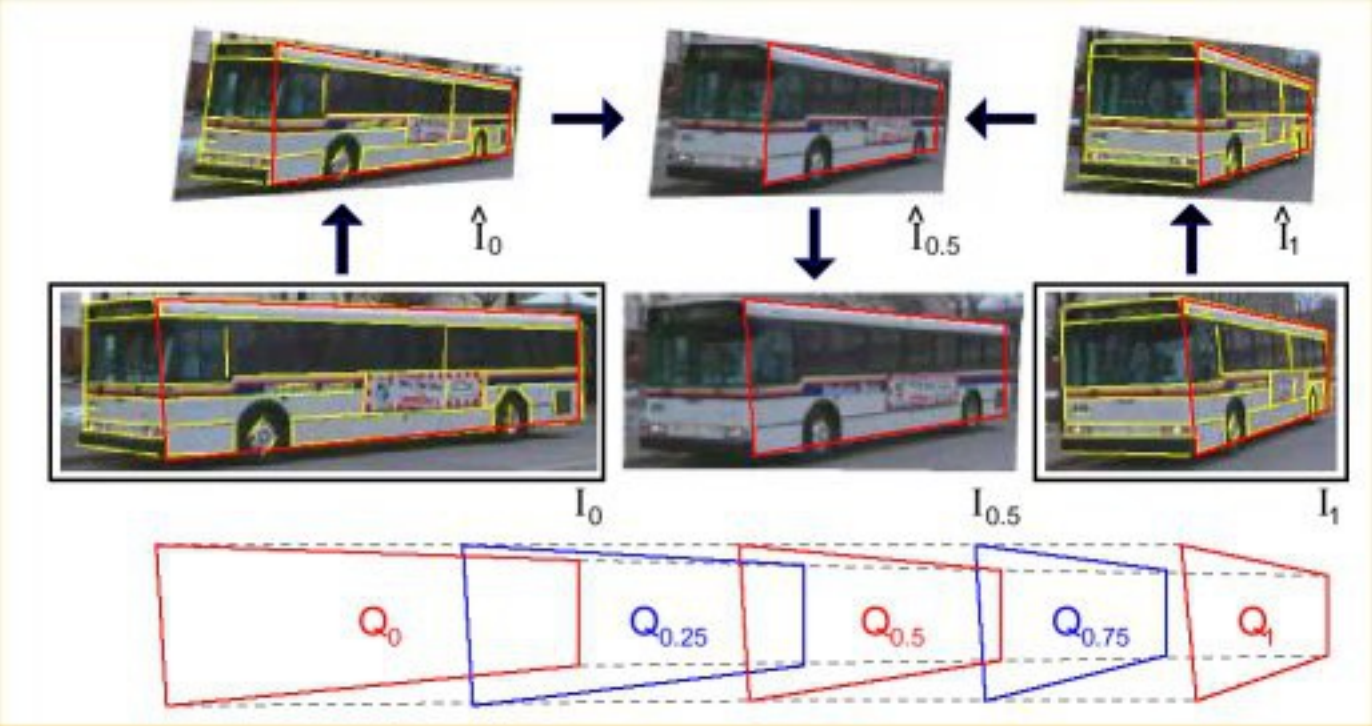
- Makes use of camera location information to project images onto parallel planes



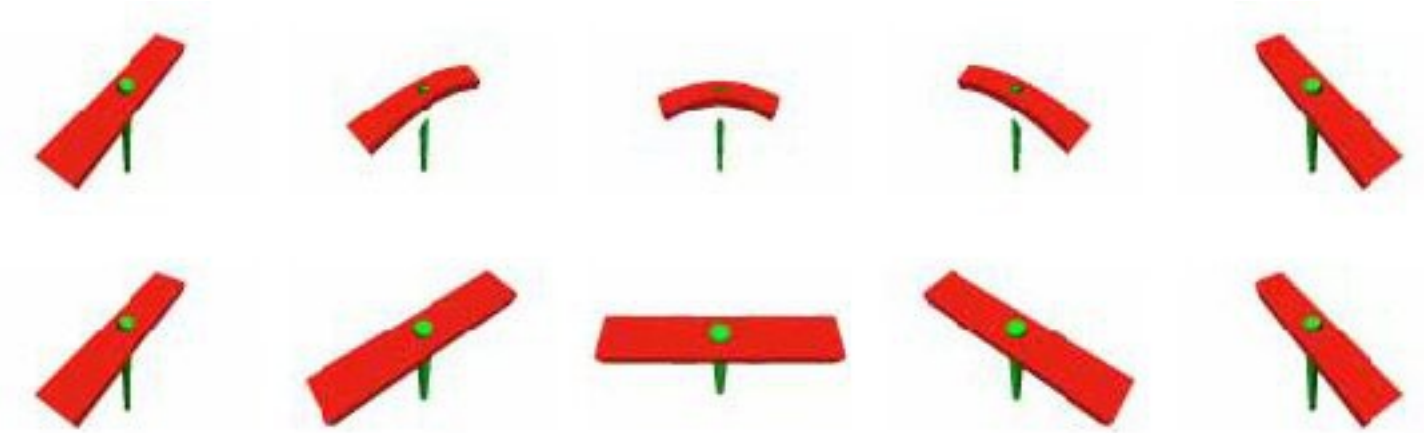
View Morphing Algorithm

- Prewarp
  - apply projective transforms to the source images (H_0^{-1} and H_1^{-1})
- Morph
  - Use any image morphing technique
- Postwarp
  - Apply H_s to obtain final image

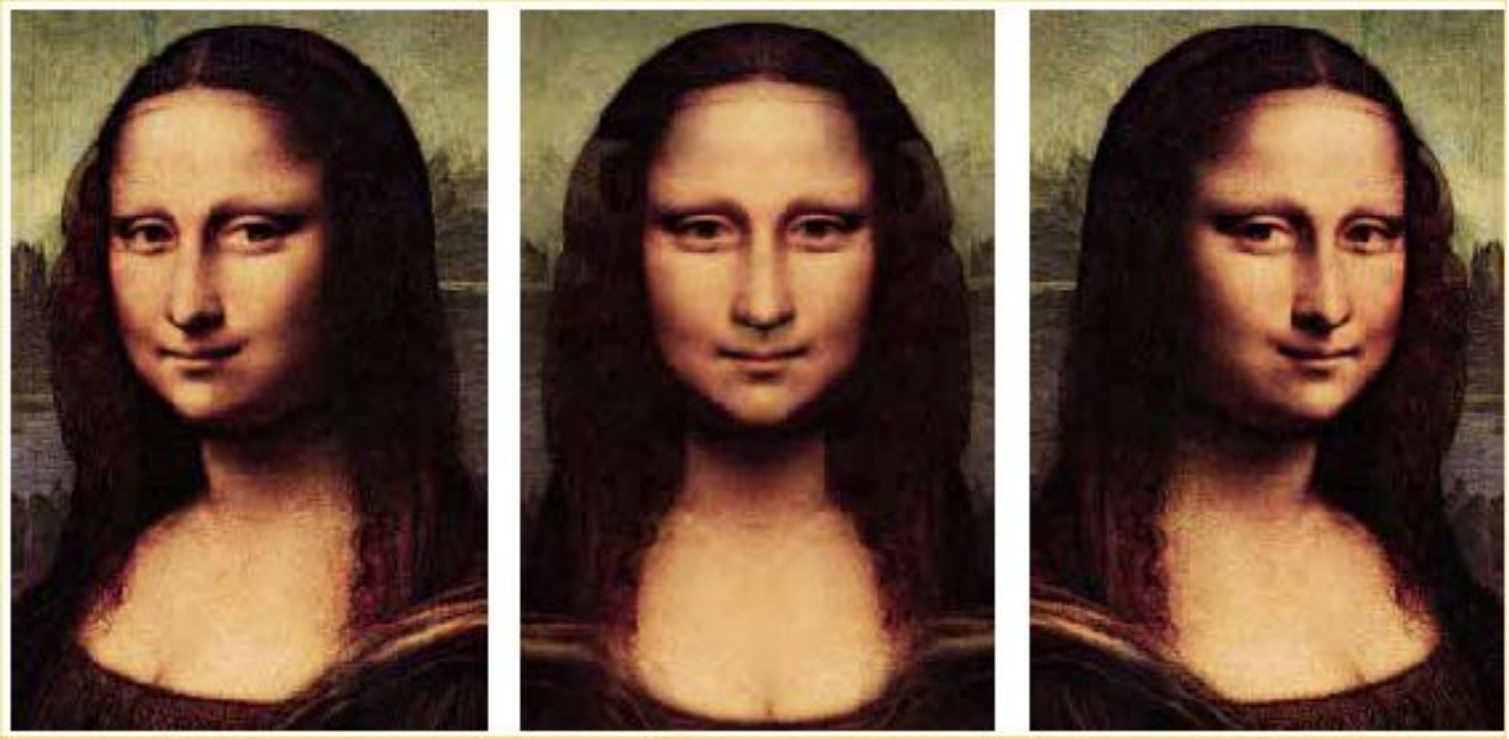
View Morphing



View Morphing



View Morphing



MORPHING EXAMPLES

Examples Contd..