Review

Blending

- Pyramid representation
  - Over complete representation
  - Smooth before down sampling
  - Laplacian – bandpass filtering
- Linear interpolation

SIFT

- Feature matching
- Invariant to translation, rotation, scale, brightness, etc..

Today: deformations

- Correspondences based on features
- Sampling

Topic 12:

Image Morphing

1. Intro to basic image morphing
2. The Baier-Neely morphing algorithm

"Feature Based Image Metamorphosis", 1992

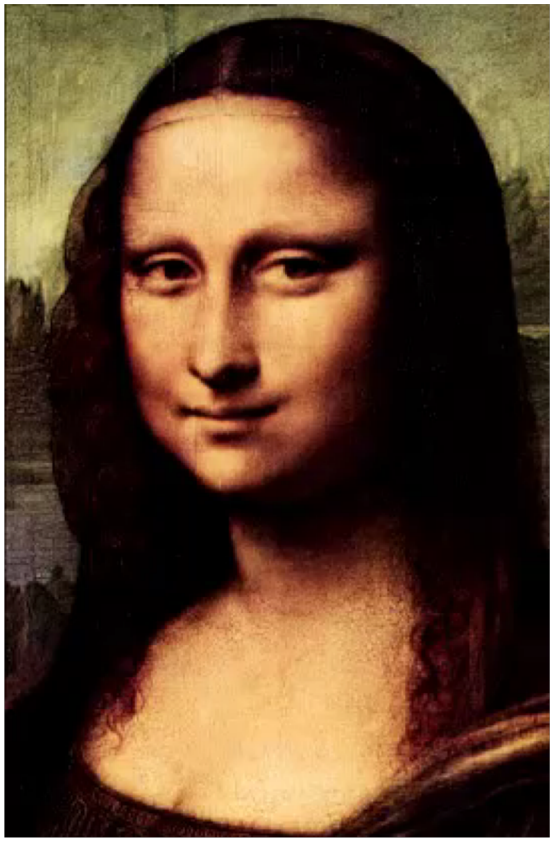
Image Morphing

Introduction to image morphing

- Basic idea
- Beier-Neely morphing

Extensions: View Morphing

A combination of view synthesis & image morphing



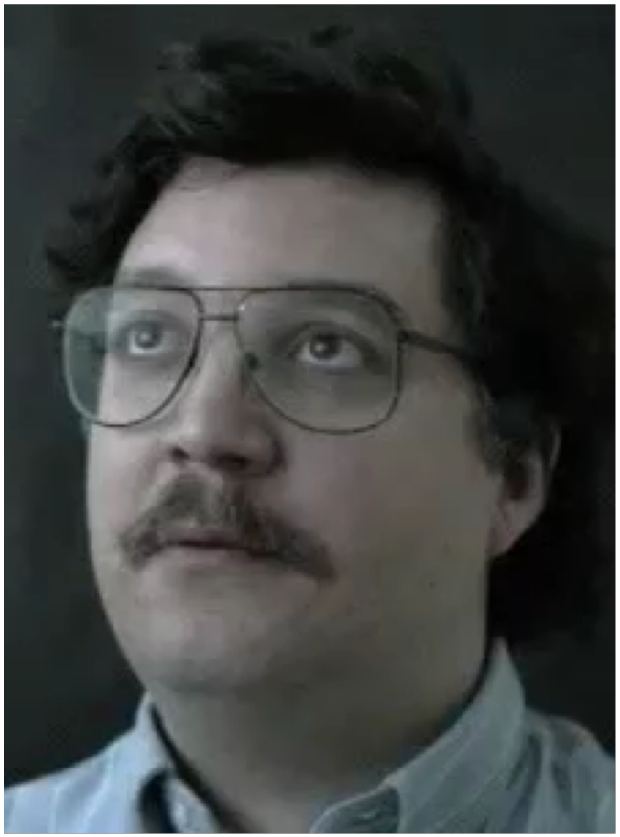
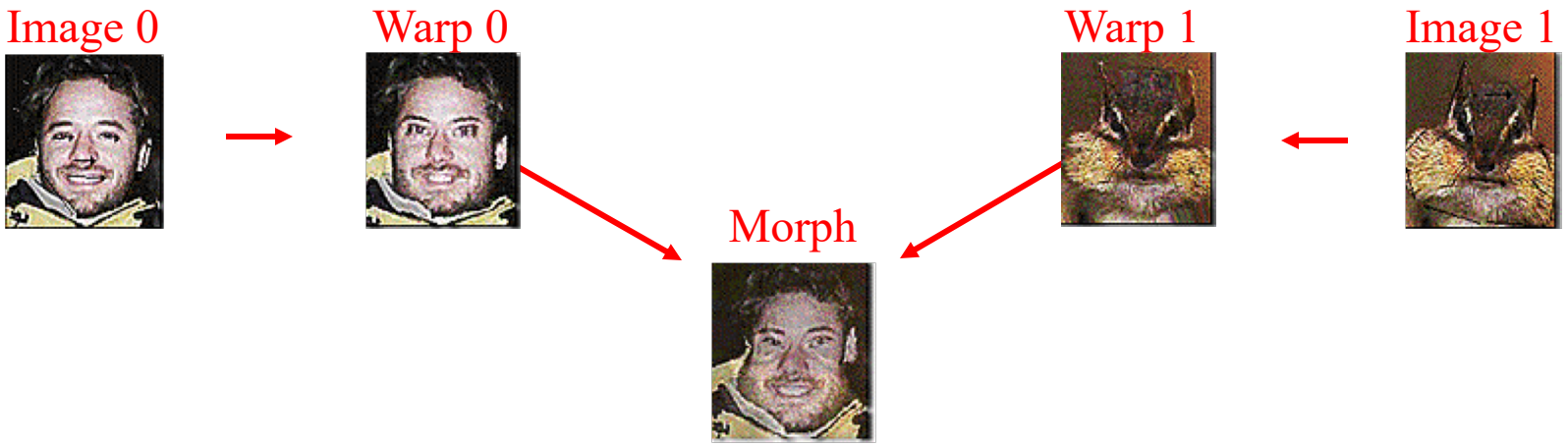


Image Morphing

A combination of generalized image warping with a cross-dissolve between pixels

Morphing involves two steps:

- Pre-warp the two images
- Cross-dissolve their colors



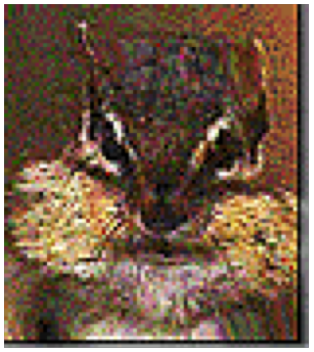
Cross-Dissolving Two Images

Cross-dissolve

A weighted combination of two images, pixel-by-pixel

Image 0

Image 1



Combination controlled by a single interpolation parameter t:

$$Dest = t \cdot (Image1) + (1-t)(Image2)$$

Image Pre-Warping

Why warp first?

In order to align features that appear in both images (e.g., eyes, mouth, hair, etc). Without such an alignment, we would get a “double-image” effect!!

Image pre-warping

Re-position all pixels in the source images to avoid the “double-image” effect as much as possible

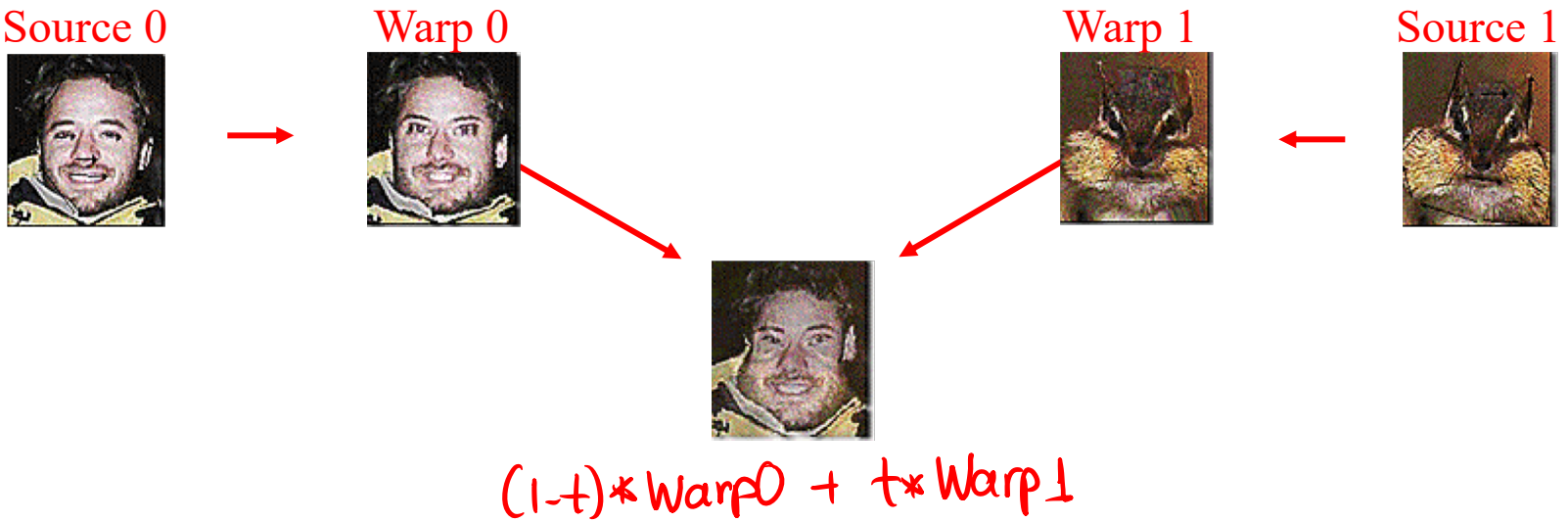
Pre-warping implemented using the Field Morphing Algorithm

Image Morphing

Both morphing steps specified by same parameter t

- Warp the two images according to t
- Cross-dissolve their colors according to t

Morphing videos generated by creating a sequence of images, defined by a sequence of t-values (e.g., 0,0.1,0.2,…,0.9,1)



Morphing Example

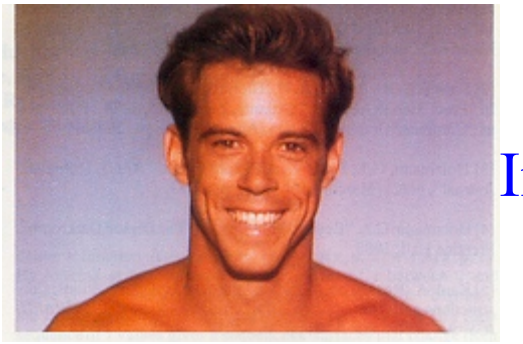


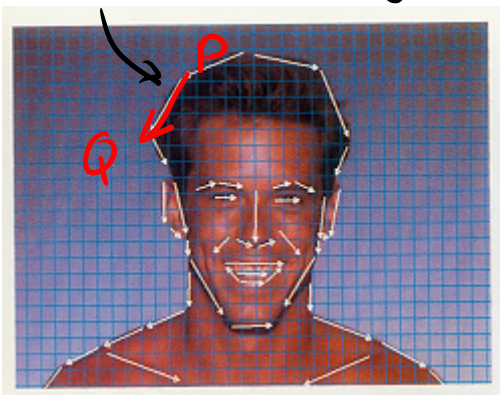
Image 0

Intermediate Images

Image 1

Beier-Neely Field Morphing Algorithm (1992)

Warped image computed using Field Morphing Algorithm



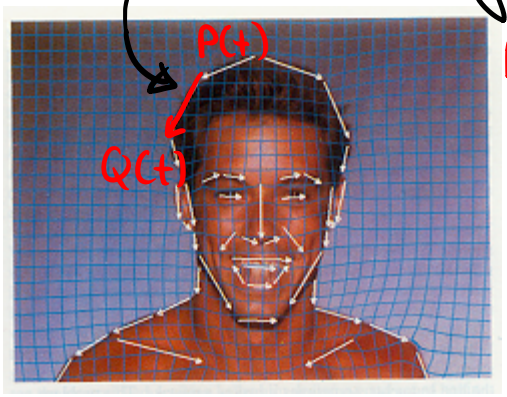
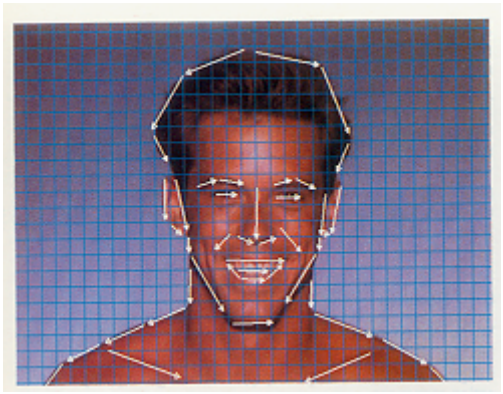
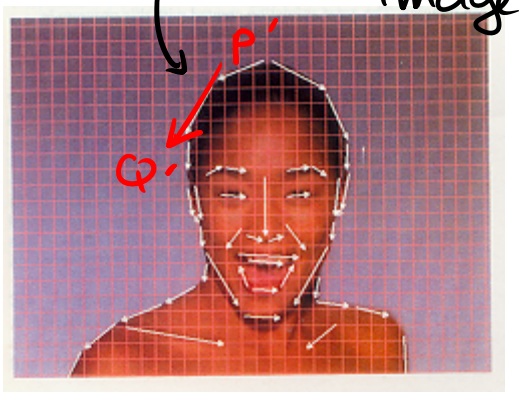


Image warp specified by interactively drawing lines in the two source images

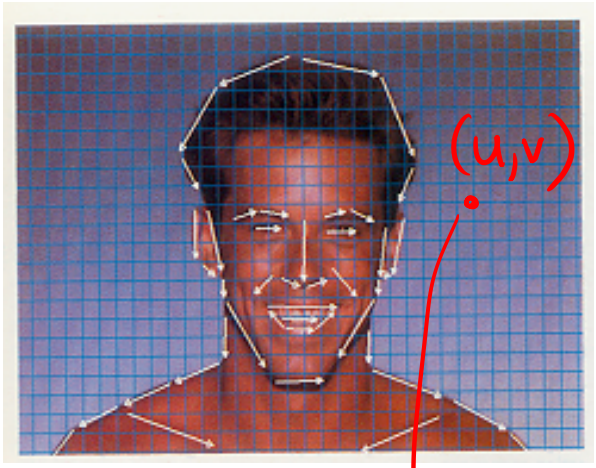




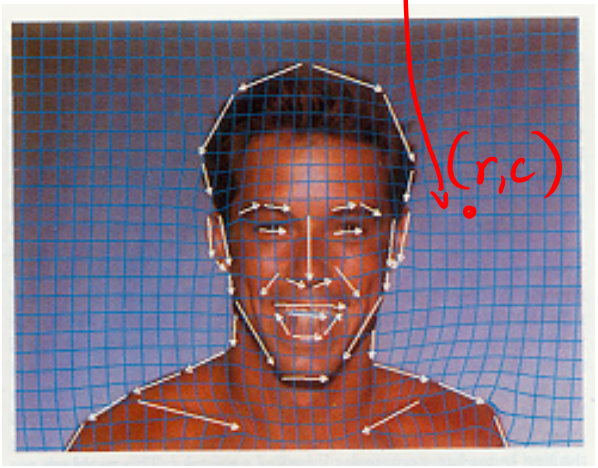
Morphing by Backward Mapping

To completely determine the morph we need to define the functions U(r,c), V(r,c)

Image 0



Intermediate warp t



Called "backward" because the loop is over the destination pixels

Backward mapping:

```
for r = rmin to rmax
  for c = cmin to cmax
    r' = U(r,c)
    c' = V(r,c)
    copy pixel at source (r',c')
      to destination (r,c)
```

Image 0

intermediate image

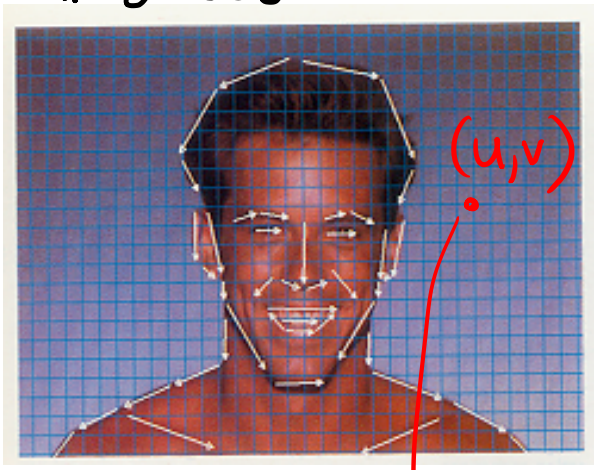
(U, V) are the mapping functions

Anti-Aliasing for Backward Mapping

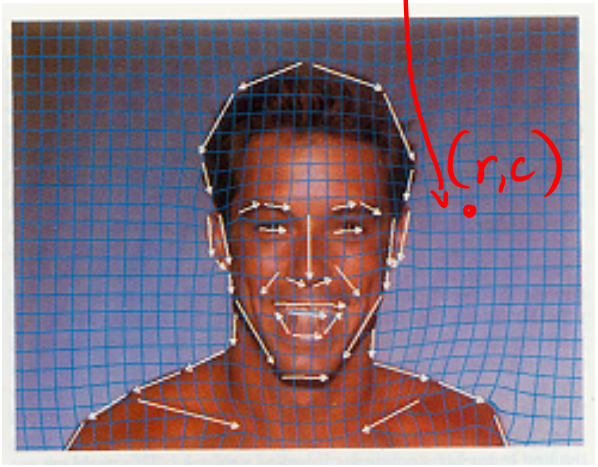
To avoid aliasing we <u>super-sample</u> the source & dest images using interpolation (linear, Gaussian, etc)

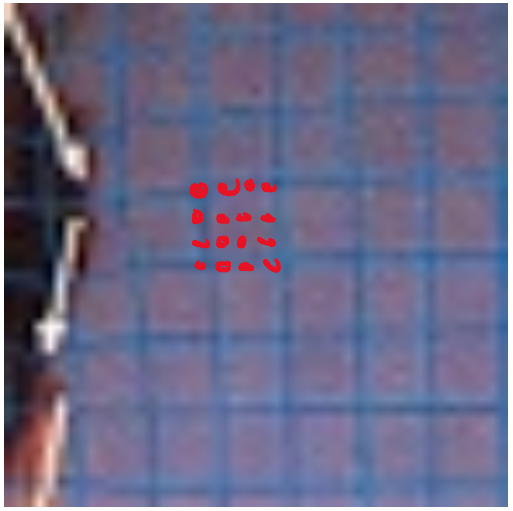
$$D(r,c) = \sum_{\Delta r=0}^{1} \sum_{\Delta c=0}^{1} D(r+\Delta r, c+\Delta c) = \sum_{\Delta r} \sum_{\Delta c} S(U(r+\Delta r, c+\Delta c), V(r+\Delta r, c+\Delta c))$$

Image 0



Intermediate warp t

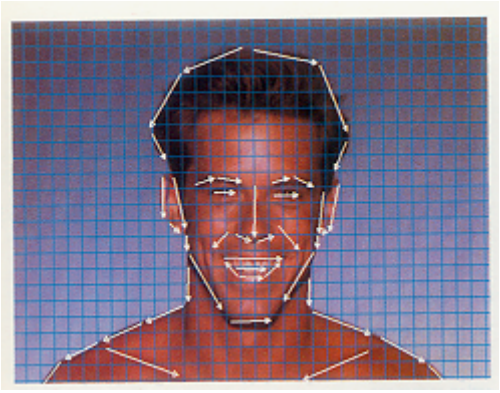




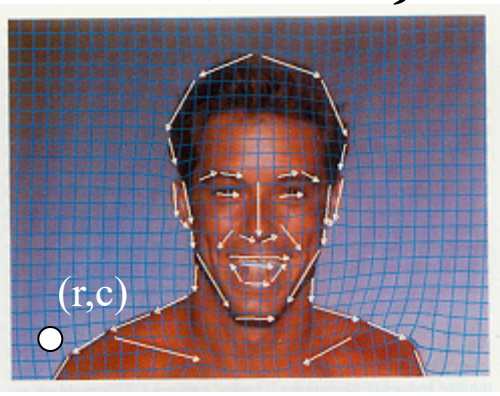
Intermediate Morphs

A single parameter t defines two warps, one applied to image 0 and one to image 1

Image 0



Warp 0 (for t)



Warp 1 (for t)

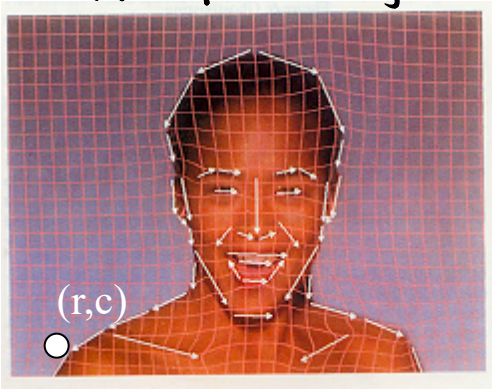
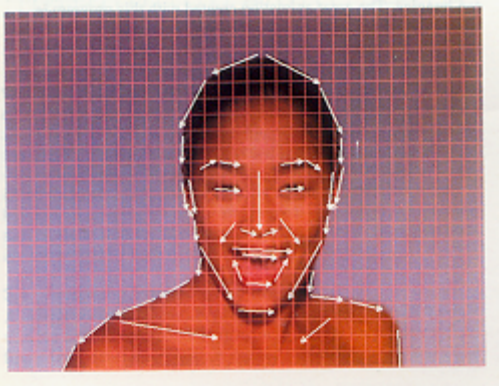


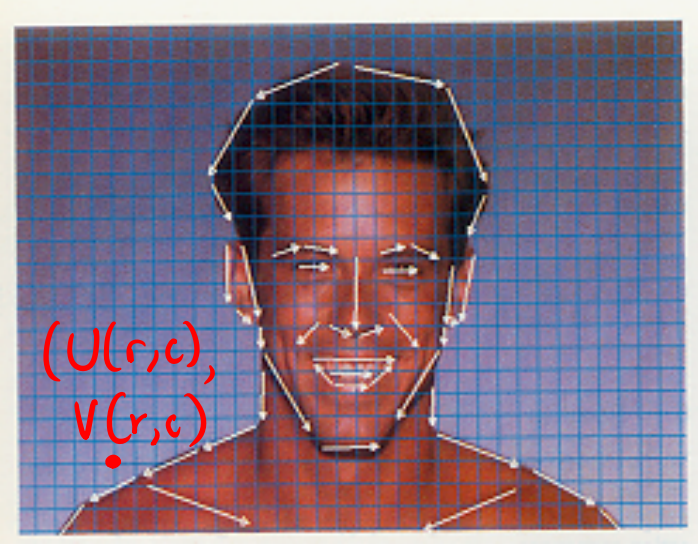
Image 1

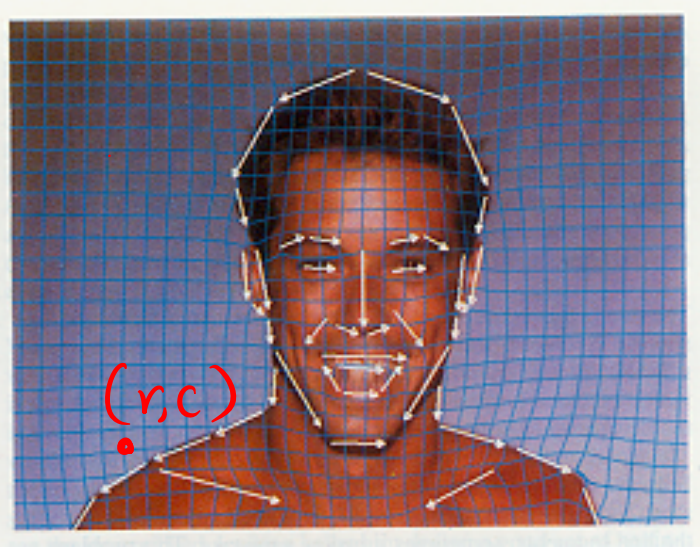


* pixel (r,c) in Warp 0 and pixel (r,c) in Warp 1 should be in exact correspondence

Coordinate Maps in Field Morphing

Coordinate map: A pair of functions $U(r,c), V(r,c)$ that map each destination pixel to its source location





Two cases:

1. Coordinate map defined by a single line pair
2. Coordinate map defined by multiple line pairs

Coordinate Maps from One Line Pair

Steps:

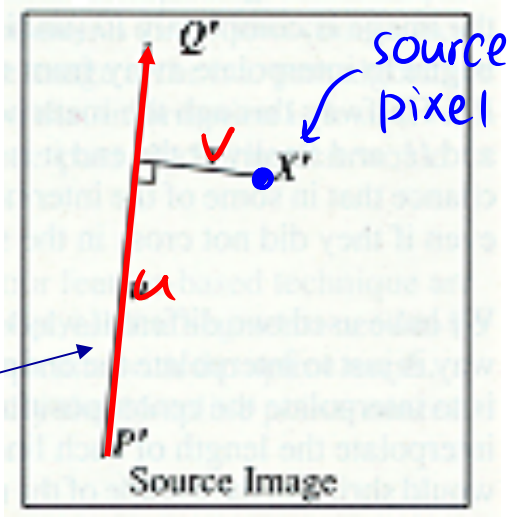
1. Compute position of pixel X in Image 1 relative to destination line

   $(r,c) \rightarrow (u,v)$

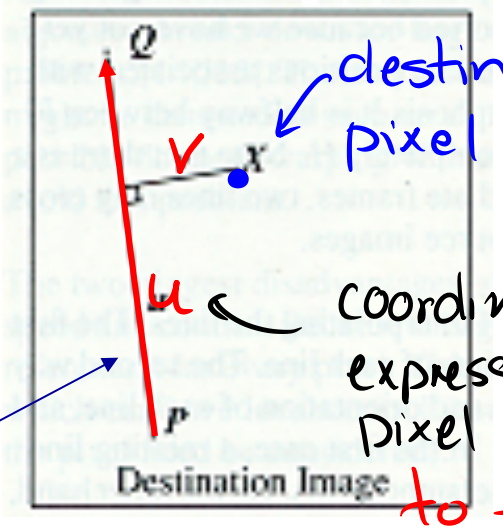
   (u, v) are the coordinates

2. Compute (r',c') coordinates of pixel X' in Image 0 whose position relative to source line is (u,v)

   $(u,v) \rightarrow (r',c')$



source line



destination line

Computing Pixel Positions Relative to a Line

Position of pixel X in Image 1 relative to destination line

$(r,c) \rightarrow (u,v)$

given by

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

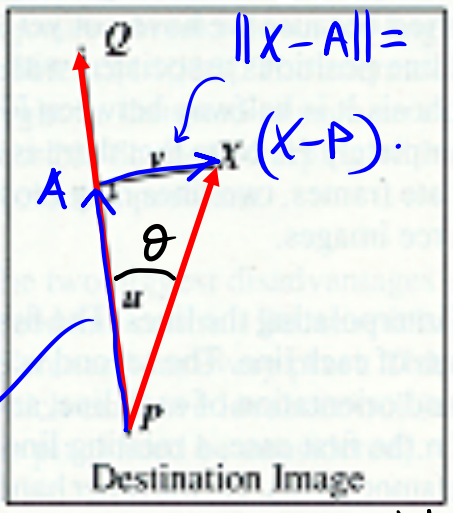
compare to:

$$\cos\theta = \frac{(X-P)(Q-P)}{\|X-P\| \cdot \|Q-P\|}$$

so $u = \cos\theta \cdot \frac{\|X-P\|}{\|Q-P\|}$

$$v = \frac{(X-P) \cdot Perp(Q-P)}{\|Q-P\|}$$

$\|A-P\| = \|X-P\| \cdot \cos\theta$



$\|X-A\| = (X-P) \cdot$

Exercise: Prove this!

Position of X expressed in terms of line endpoints:

$$X = P + u \cdot (Q-P) + v \cdot Perp(Q-P) \cdot \frac{1}{\|Q-P\|}$$

Computing Pixel Positions Relative to a Line

Position of pixel X in Image 0 relative to source line

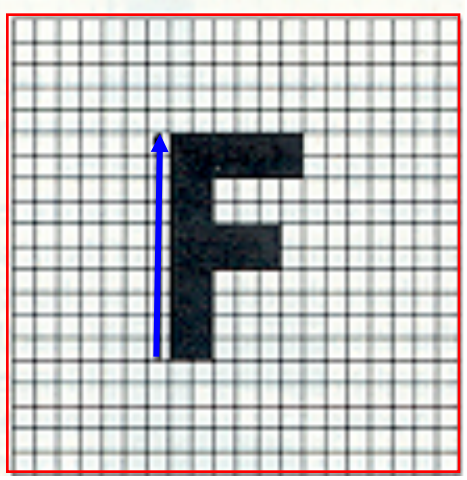
$(r,c) \rightarrow (u,v)$

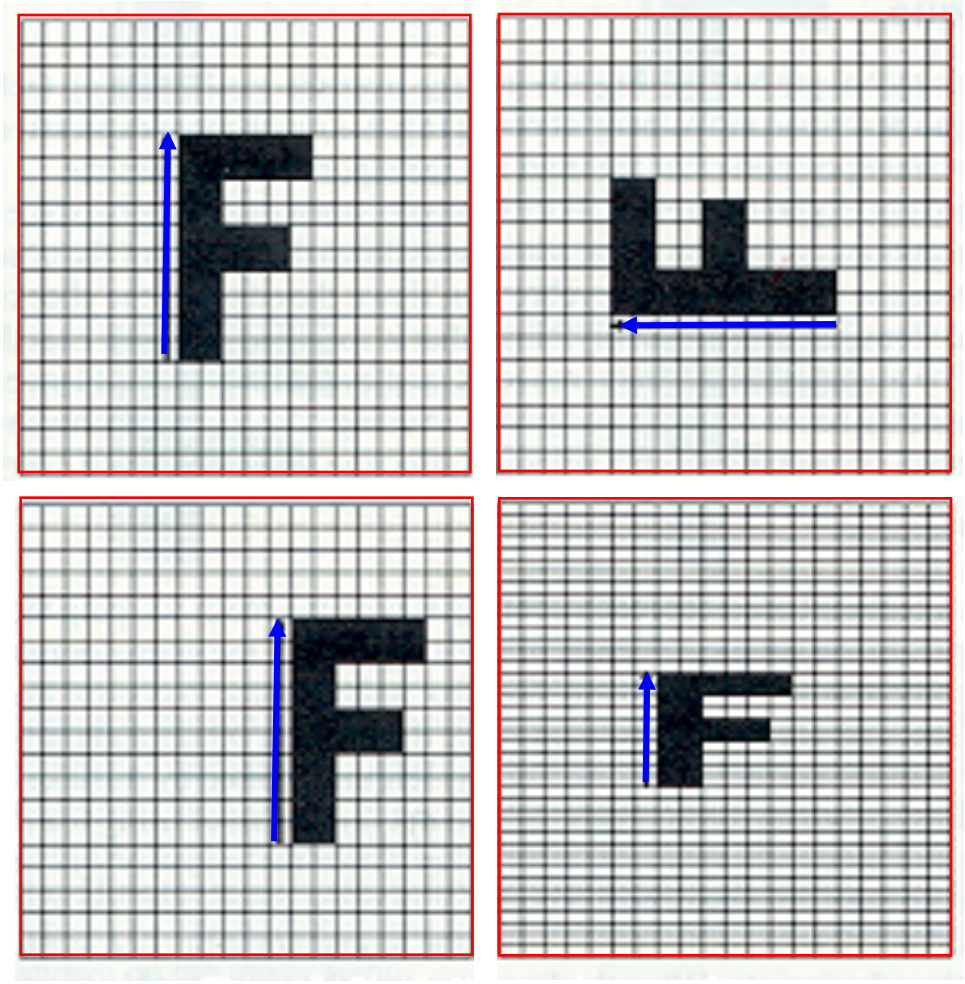
Position of X' expressed in terms of new line endpoints:

$$X' = P' + u \cdot (Q' - P') + v \cdot Perp(Q' - P') \cdot \frac{1}{\|Q' - P'\|}$$

Pixel Coordinates Relative to a Line

Examples





Coordinate Maps from One Line Pair

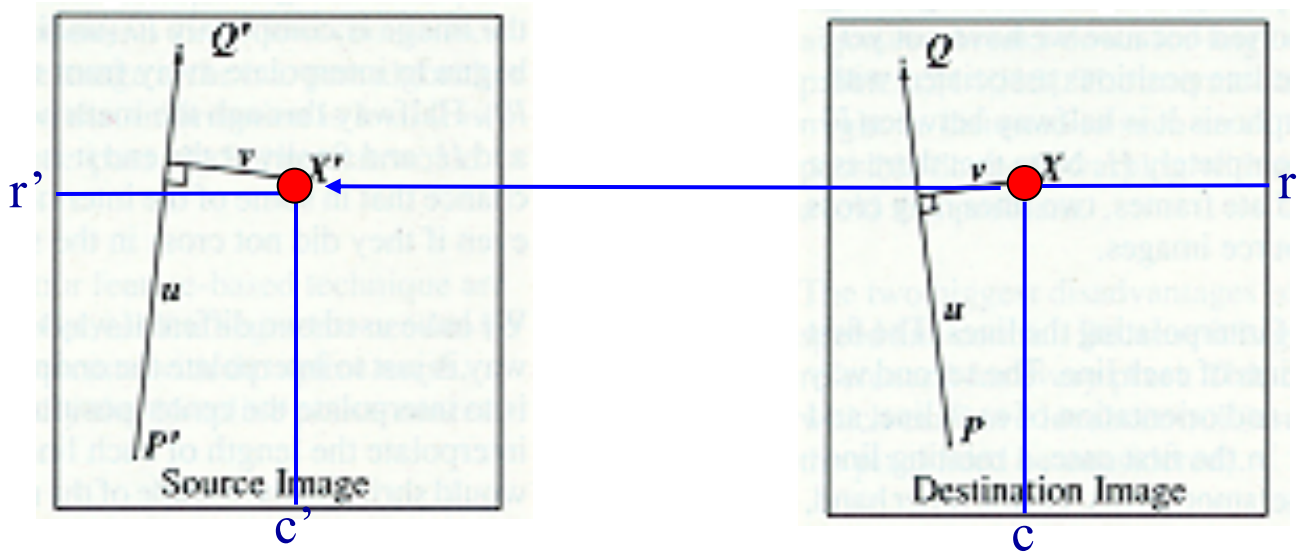
Field warping algorithm (single-line case)

For each pixel (r,c) in the destination image

find the corresponding (u,v) coordinates of the pixel

find the (r',c') in source image for that (u,v)

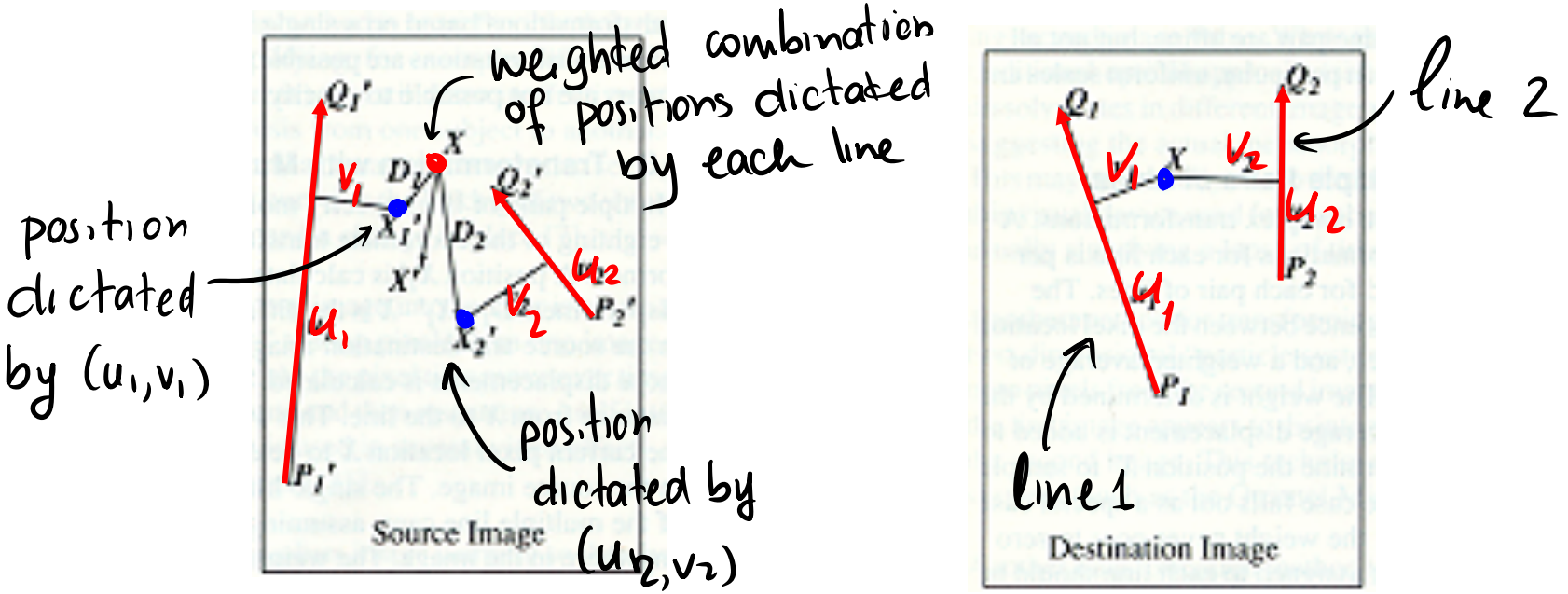
color at destination pixel (r,c) = color at source pixel (r',c')



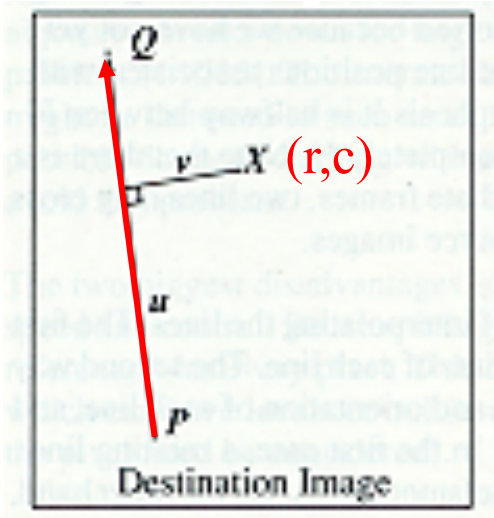
Coordinate Maps from Multiple Line Pairs

Field warping algorithm (multiple-line case)

1. Apply single-line field warping algorithm to each line separately, to get N source pixel positions (r_i', c_i') for every destination pixel (N= # of line pairs)
2. Compute source position (r',c') as weighted average of positions (r_i', c_i')



Computing the Averaging Weights



length of line

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

distance of pixel from line.

controls influence of line for points near it

$\text{abs}(v)$ if $0 < u < 1$

distance of (r,c) from P if $u < 0$

distance of (r,c) from Q if $u > 1$

Beier-Neely Field Warping Algorithm

For each pixel (r,c) in destination image

DSUM=(0,0)

weightsum = 0

for each line (P_i,Q_i)

calculate (u_i,v_i)

based on P_i,Q_i

calculate (r_i',c_i')

based on u,v & P_i',Q_i'

calculate displacement

$D_i=X_i'-X_i$ for this line

calculate weight

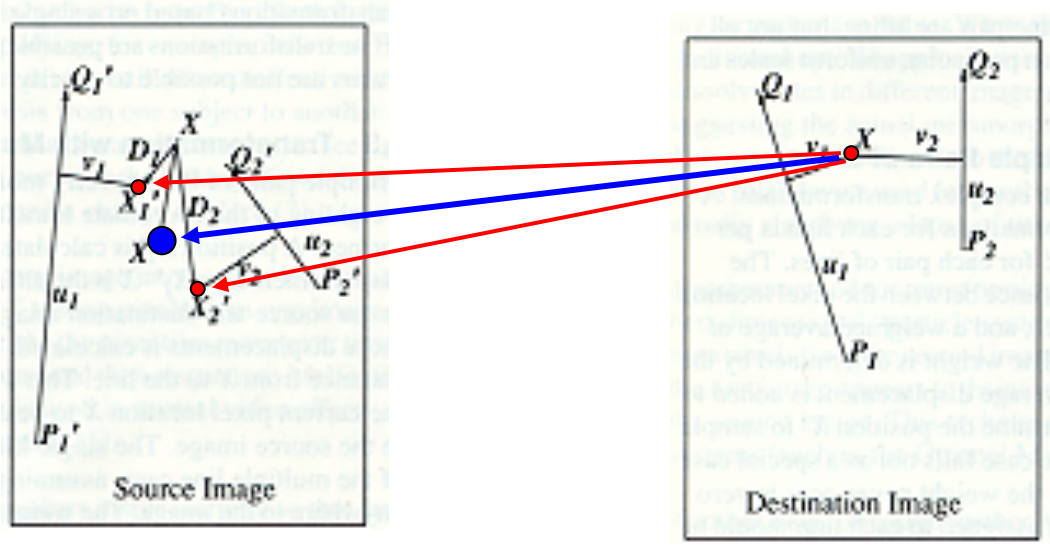
for line (P_i,Q_i)

DSUM += Di*weight

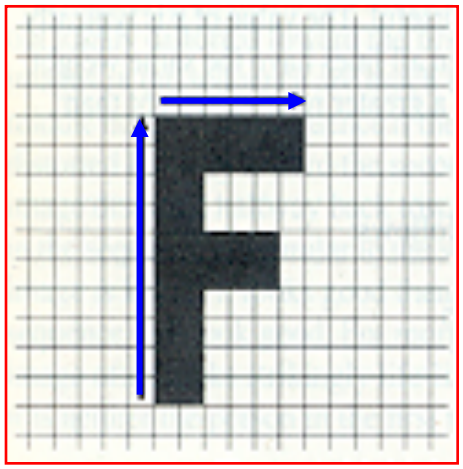
weightsum += weight

(r',c') = (r,c) + DSUM/weightsum

color at destination pixel (r,c) = color at source pixel (r',c')



Warping Example



Morphing Example

Figure 7

Figure 8

Figure 10

Figure 7 shows the lines drawn over the a face, figure 9 shows the lines drawn over a second face. Figure 8 shows the morphed image, with the interpolated lines drawn over it.

Figure 10 shows the first face with the lines and a grid, showing how it is distorted to the position of the lines in the intermediate frame. Figure 11 shows the second face distorted to the same intermediate position. The lines in the top and bottom picture are in the same position. We have distorted the two images to the same "shape".

Note that outside the outline of the faces, the grids are warped very differently in the two images, but because this is the background, it is not important. If there were background features that needed to be matched, lines could have been drawn over them as well.

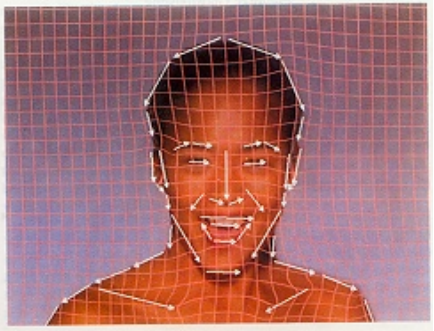


Figure 12

Figure 12 is the first face distorted to the intermediate position, without the grid or lines. Figure 13 is the second face distorted toward that same position. Note that the blend between the two distorted images is much more life-like than the either of the distorted images themselves. We have noticed this happens very frequently.

The final sequence is figures 14, 15, and 16.

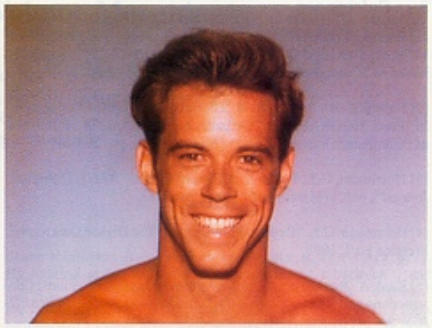


Figure 14

Figure 15