

## **Lecture Notes #2 - Image Morphing**

Reading:

- Beier & Neely, Feature-Based Image Metamorphosis, SIGGRAPH '92.

Optional:

- A. Lerios, C. Garfinkle, M. Levoy. Feature-based Volume Metamorphosis. SIGGRAPH '95.

## **Outline of lecture**

- Cross-dissolving
- Morphing
  - Warping with one line pair
  - Warping with multiple line pairs
  - Interpolating lines

## Image Morphing

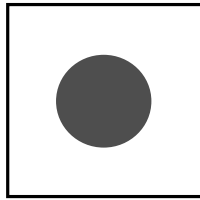
Simplest type of metamorphosis is "cross-dissolve".

Each pixel  $p$  at time  $t$  in  $[0,1]$  is computed through linear interpolation.

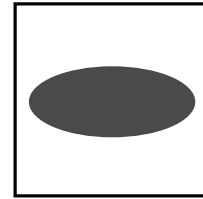
$$p = (1-t) p(\text{start}) + t p(\text{end})$$

where  $p(\text{start})$  and  $p(\text{end})$  are the starting and ending pixel values.

Not very effective



Starting image



Ending image

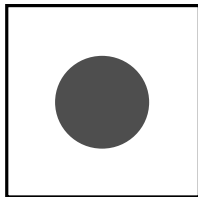
## Morphing

"Morphing" involves a combination of

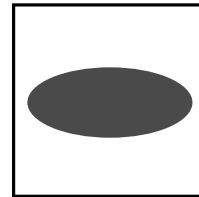
**1. Image warping**

**2. Cross - dissolving**

The middle image is key: if it looks good, the whole sequence will.



Starting image



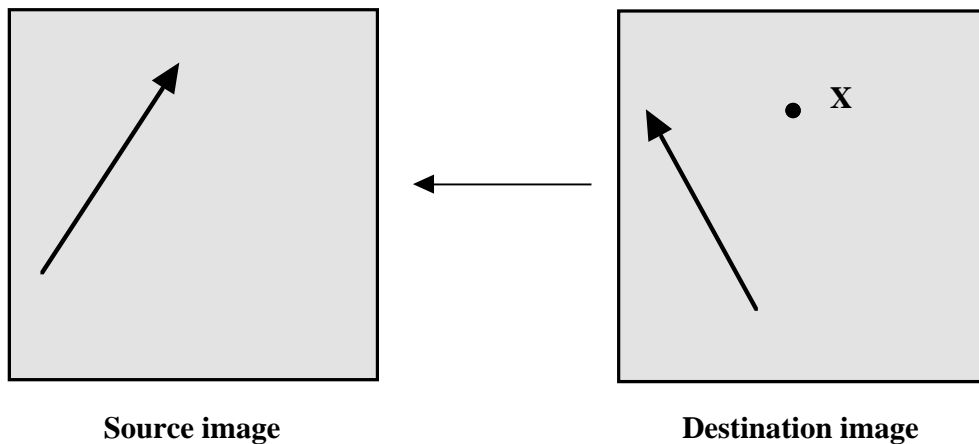
Ending image

## Image Warping

- The image warping step is the difficult one:
- There are two basic styles:
  - Forward warping: pixels from the source image are warped into the destination image.
  - Reverse mapping: the opposite.
- Q: which style should we choose?

## Warping by one Line-Pair

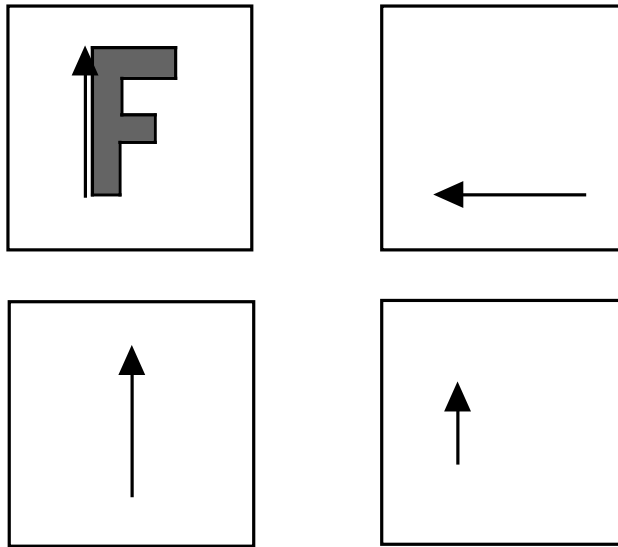
Beier and Neely use a pair of lines to specify the warp:



- $u$  is a fraction
- $v$  is measured in pixels

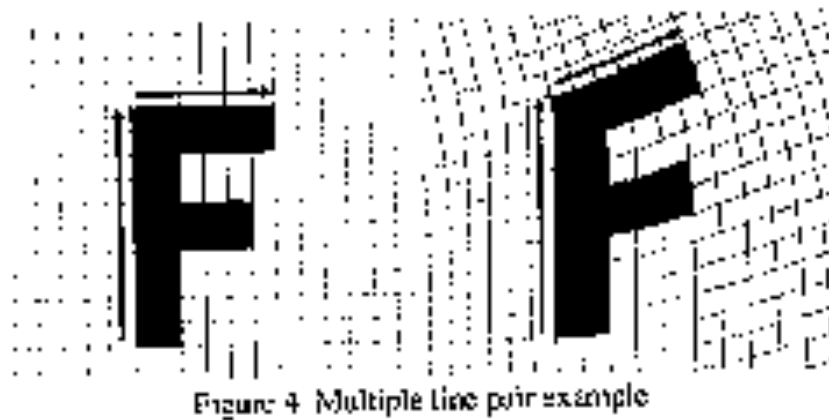
## Warping with one line pair, cont.

- What happens when we warp with one line pair?

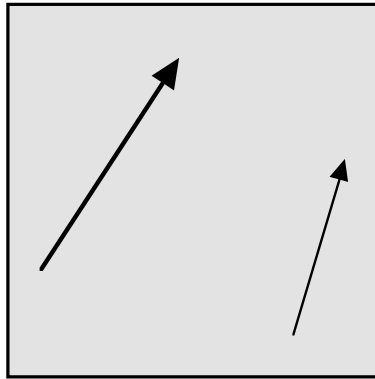


## Non-linear Warping

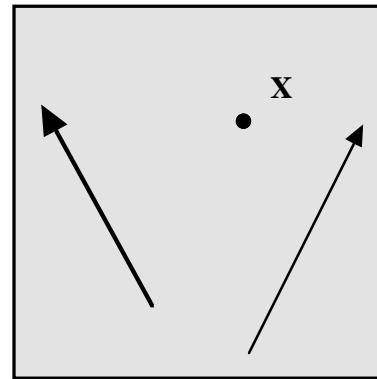
For multiple line pairs, Beier and Neely use a weighted combination of points defined by some mapping.



## Warping by Multiple Line-Pairs



Source image



Destination image

## Warping Pseudo-code

```
WarpImage(SourceImage, L[...], L[...])  
begin  
  foreach destination pixel X do  
    XSum = (0,0)  
    WeightSum = 0  
    foreach line L[i] in destination do  
      X'[i] = X transformed by (L[i], L'[i])  
      weight[i] = weight assigned to X'[i]  
      XSum = XSum + X'[i] * weight[i]  
      WeightSum += weight[i]  
    end  
    X' = XSum/WeightSum  
    DestinationImage(X) = SourceImage(X')  
  end  
return Destination  
end
```

## Morphing Pseudo-code

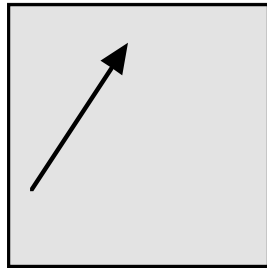
Here's how you create an animated morph:

```
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
            FinalImage(p) = (1-t) Warp0(p) + t Warp1(p)
        end
    end
end
```

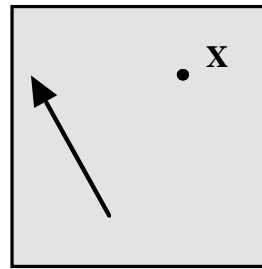
## Remaining Issues

- There are still a few unresolved details:
  - Exactly how to compute X' from X and a single line pair L, L'.
  - Exactly how to weight each X'.
  - Exactly how to interpolate between lines.

## Computing $X'$ from $X$ and a line pair



Source  
image



Destination  
image

We need formulas for  $u$  and  $v$ :

## Weighting each $X'$

To weight the contribution of each line, Beier and Neeley use

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

where:

- $length$  is the length of  $L[i]$
- $dist$  is the distance from  $X$  to  $L[i]$
- $a, b, p$  are constants that control the warp (see paper)

## Interpolating Lines

- Method 1: interpolating endpoints
- Method 2: interpolating midpoints, length and orientation.