Feature Based Image Metamorphosis,
Beier and Neely 1992

Image Compositing and Morphing

Connelly Barnes

CS 4810: Graphics

Acknowledgement: slides by Jason Lawrence,
Misha Kazhdan, Allison Klein, Tom Funkhouser, Adam Finkelstein and David Dobkin

Outline

- Image Compositing
  - Blue-screen mattes
  - Alpha channel
  - Porter-Duff compositing algebra
- Image Morphing

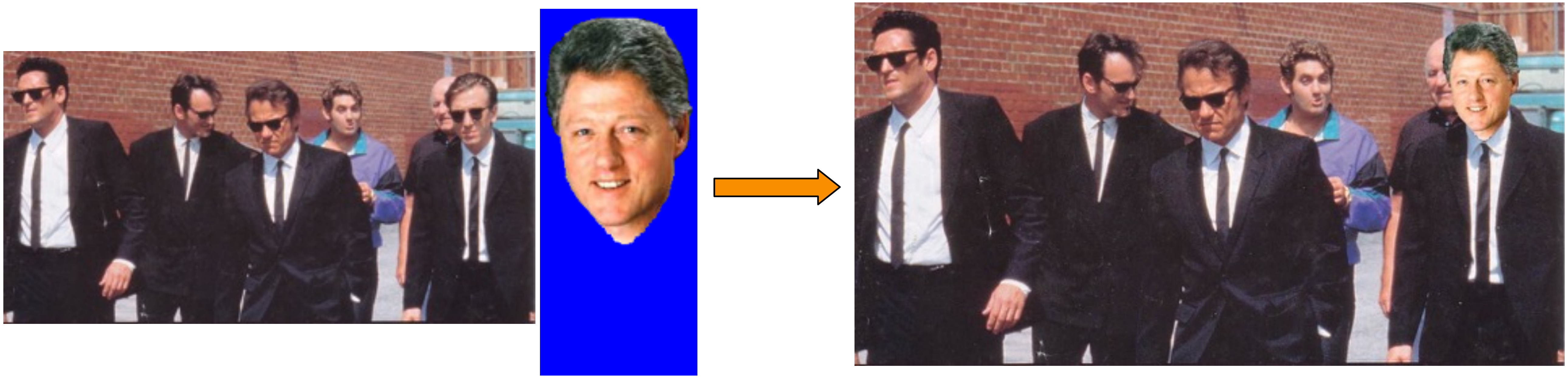
Image Compositing

- Separate an image into "elements"
  - Render independently
  - Composite together
- Applications
  - Cel animation
  - Chroma-keying
  - Blue-screen matting

Bill makes ends meet by going into film

Blue-Screen Matting

- Composite foreground and background images
  - Create background image
  - Create foreground image with blue background
  - Insert non-blue foreground pixels into background



Blue-Screen Matting

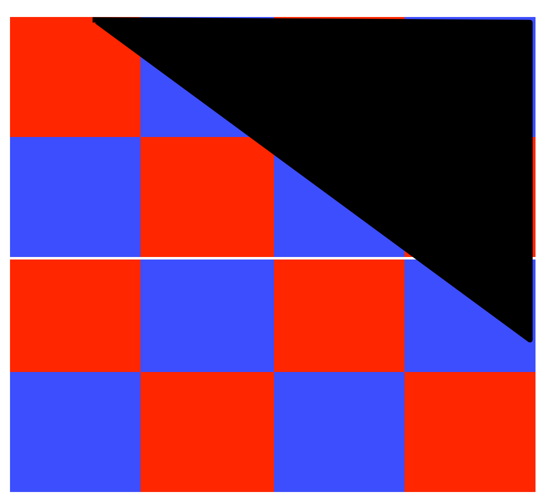
- Composite foreground and background images
  - Create background image
  - Create foreground image with blue background
  - Insert non-blue foregrou ound

Problem: lack of partial coverage results in a haloing effect along the boundary!

Alpha Channel

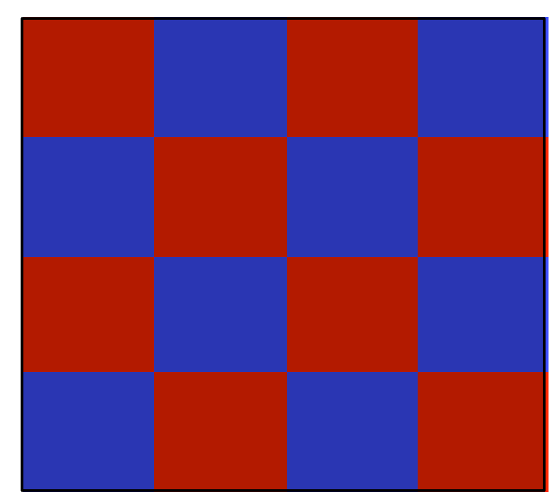
- Encodes pixel coverage information
  - o $\alpha = 0$: no coverage (or transparent)
  - o $\alpha = 1$: full coverage (or opaque)
  - o $0 < \alpha < 1$: partial coverage (or semi-transparent)

- Single Pixel Example: $\alpha = 0.3$



Partial Coverage

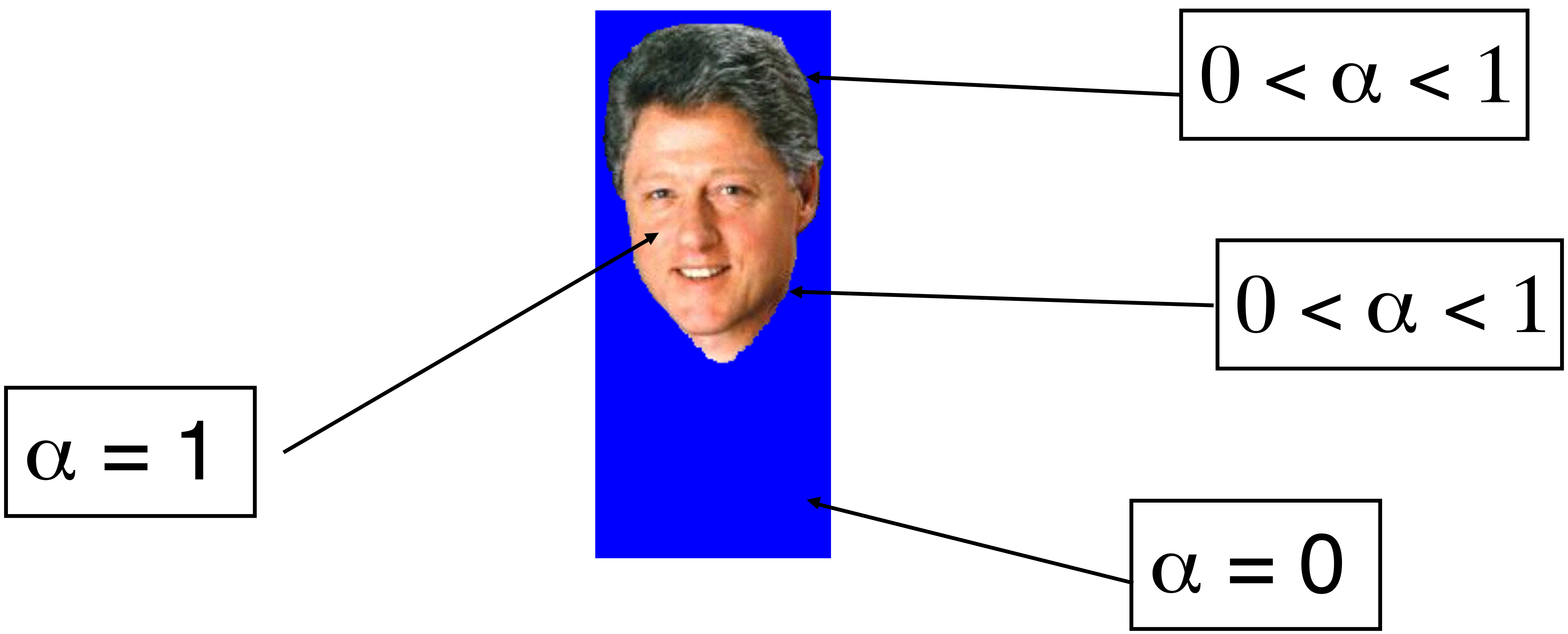
or



Semi-Transparent

Compositing with Alpha

Controls the linear interpolation of foreground and background pixels when elements are composited.



Pixels with Alpha

- Alpha channel convention, alpha premultiplied:
  o(r, g, b, α) represents a pixel that has opacity α and color $C = (r/\alpha, g/\alpha, b/\alpha)$
  - »Color components are pre-multiplied by α
  - »Simplifies compositing math
- Alpha channel convention, alpha not premultiplied:
  o(r, g, b, α) represents a pixel that has opacity α and color $C = (r, g, b)$
  - »Color C can be displayed directly

Pixels with Alpha

- What is the meaning of the following? (pre-multiplied)

  o(0, 1, 0, 1) = Full green, full coverage

  o(0, 1/2, 0, 1) = ?

  o(0, 1/2, 0, 1/2) = ?

  o(0, 1/2, 0, 0) = ?

Pixels with Alpha

- What is the meaning of the following? (pre-multiplied)

  o(0, 1, 0, 1) = Full green, full coverage

  o(0, 1/2, 0, 1) = Half green, full coverage

  o(0, 1/2, 0, 1/2) = ?

  o(0, 1/2, 0, 0) = ?

Pixels with Alpha

- What is the meaning of the following? (pre-multiplied)
  o(0, 1, 0, 1) = Full green, full coverage

  o(0, 1/2, 0, 1) = Half green, full coverage

  o(0, 1/2, 0, 1/2) = Full green, half coverage

  o(0, 1/2, 0, 0) = ?

Pixels with Alpha

- What is the meaning of the following? (pre-multiplied)

  o(0, 1, 0, 1) = Full green, full coverage

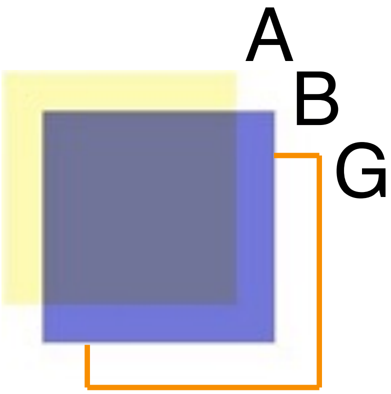
  o(0, 1/2, 0, 1) = Half green, full coverage

  o(0, 1/2, 0, 1/2) = Full green, half coverage

  o(0, 1/2, 0, 0) = Undefined

Semi-Transparent Objects

- Suppose we put A over B over background G



  - **o** How much of B is blocked by A?

$$\alpha_A$$

  - **o** How much of B shows through A

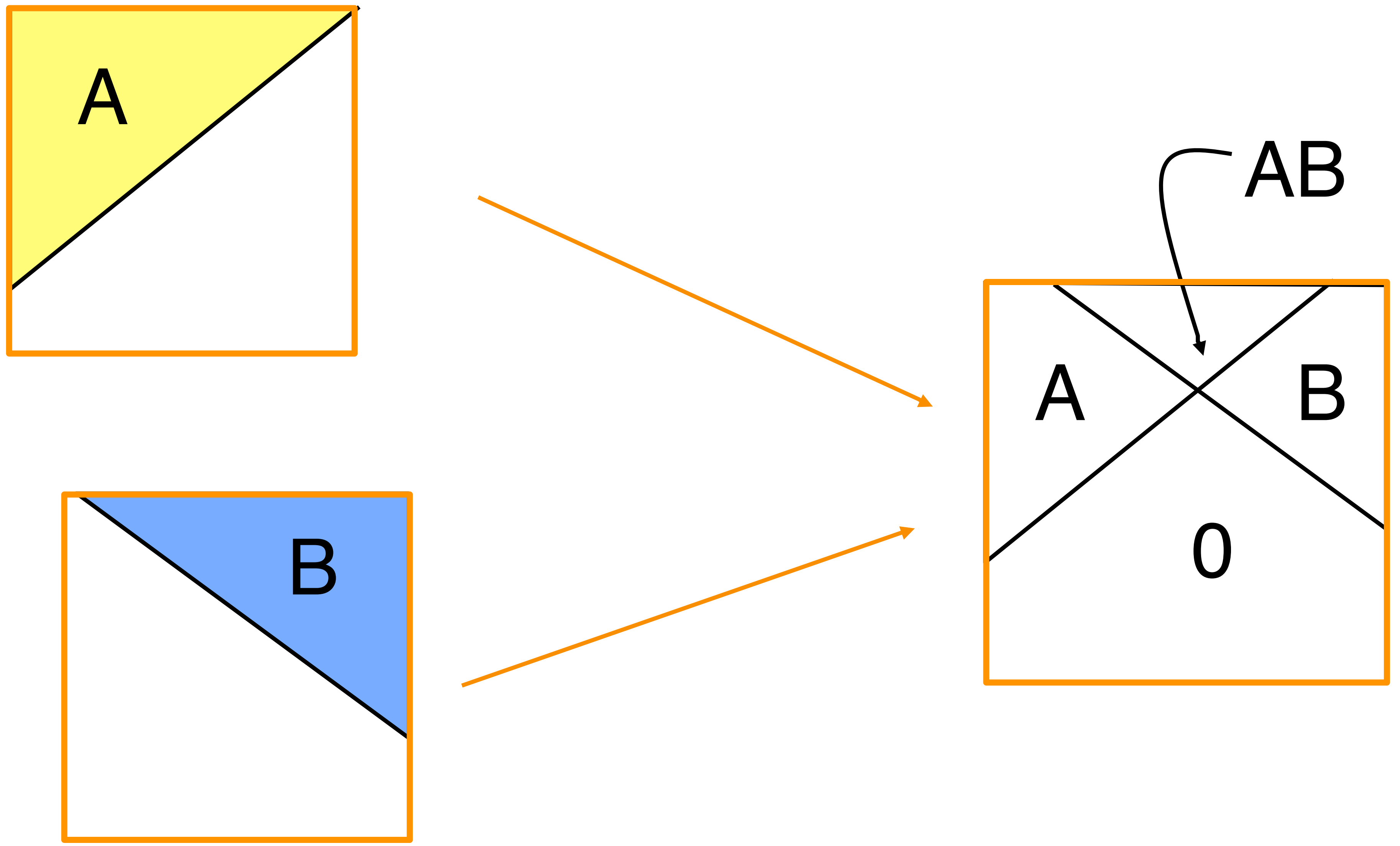
$$(1-\alpha_A)$$

  - **o** How much of G shows through both A and B?

$$(1-\alpha_A)(1-\alpha_B)$$

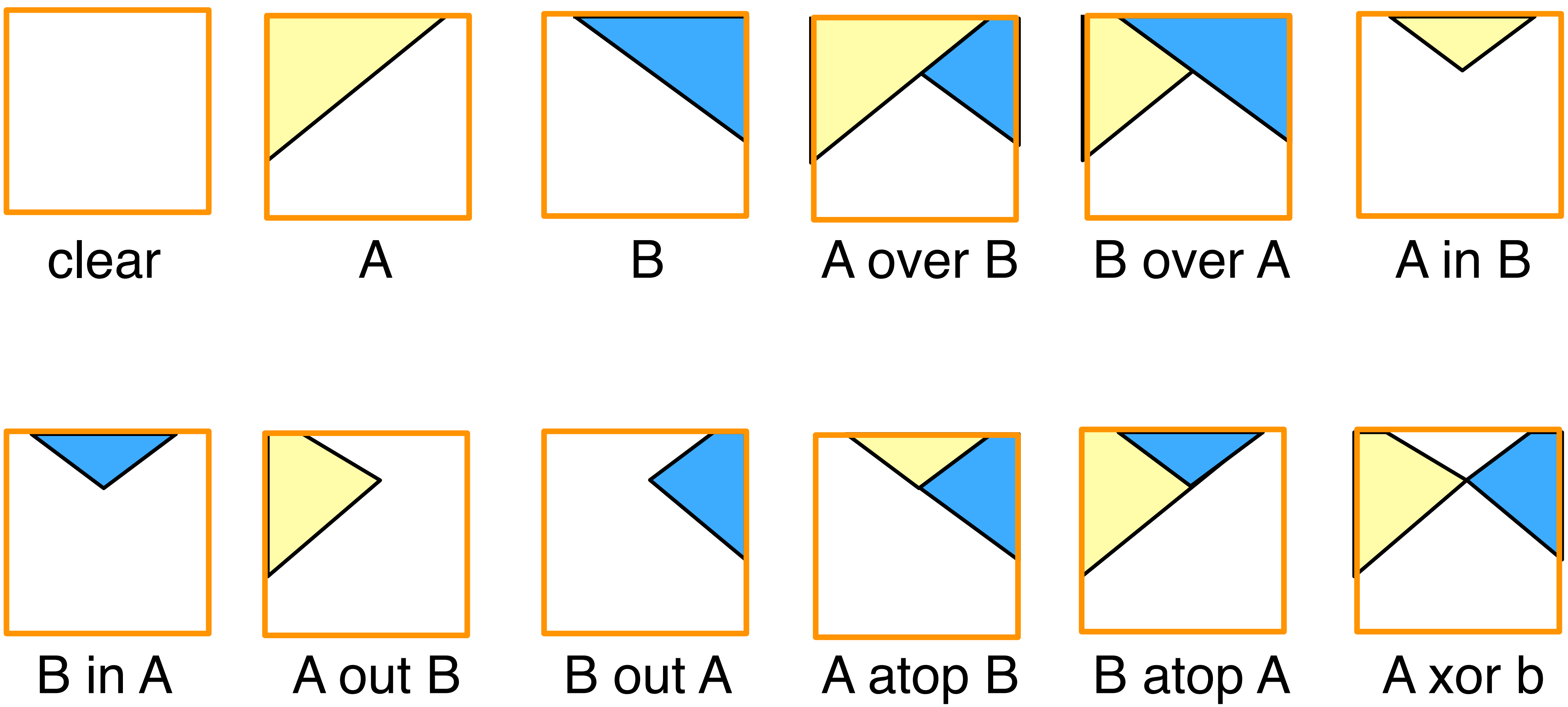
Opaque Objects

- How do we combine 2 partially covered pixels?
  - 4 regions (0, A, B, AB)
  - 3 possible colors (0, A, B)



Composition Algebra

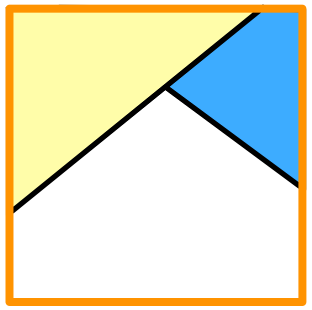
- 12 possible combinations



Porter & Duff `84

Example: C = A Over B

- For colors that are not premultiplied:
  - $C = [\alpha_A A + (1-\alpha_A)\, \alpha_B B] \,/\, [\alpha_A + (1-\alpha_A)\, \alpha_B]$
  - $\alpha = \alpha_A + (1-\alpha_A)\, \alpha_B$
- For colors that are premultiplied:
  - $C' = A' + (1-\alpha_A)\, B'$
  - $\alpha = \alpha_A + (1-\alpha_A)\, \alpha_B$



A over B

Image Composition "Goofs"

- Visible hard edges
- Incompatible lighting/shadows
- Incompatible camera focal lengths

Overview

- Image Compositing
- Image morphing
  - Specifying correspondences
  - Warping
  - Blending

Image Morphing

- Animate transition between two images

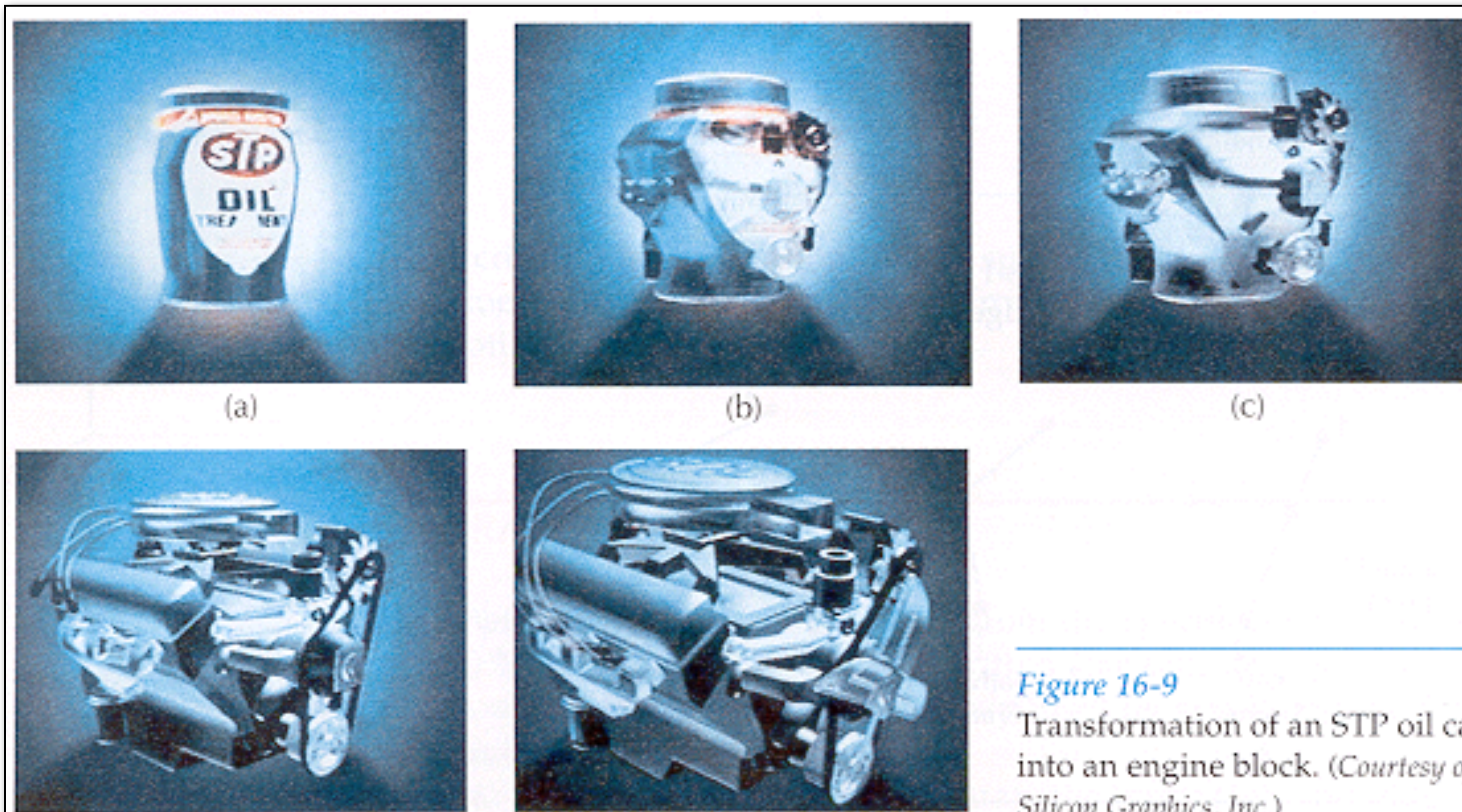
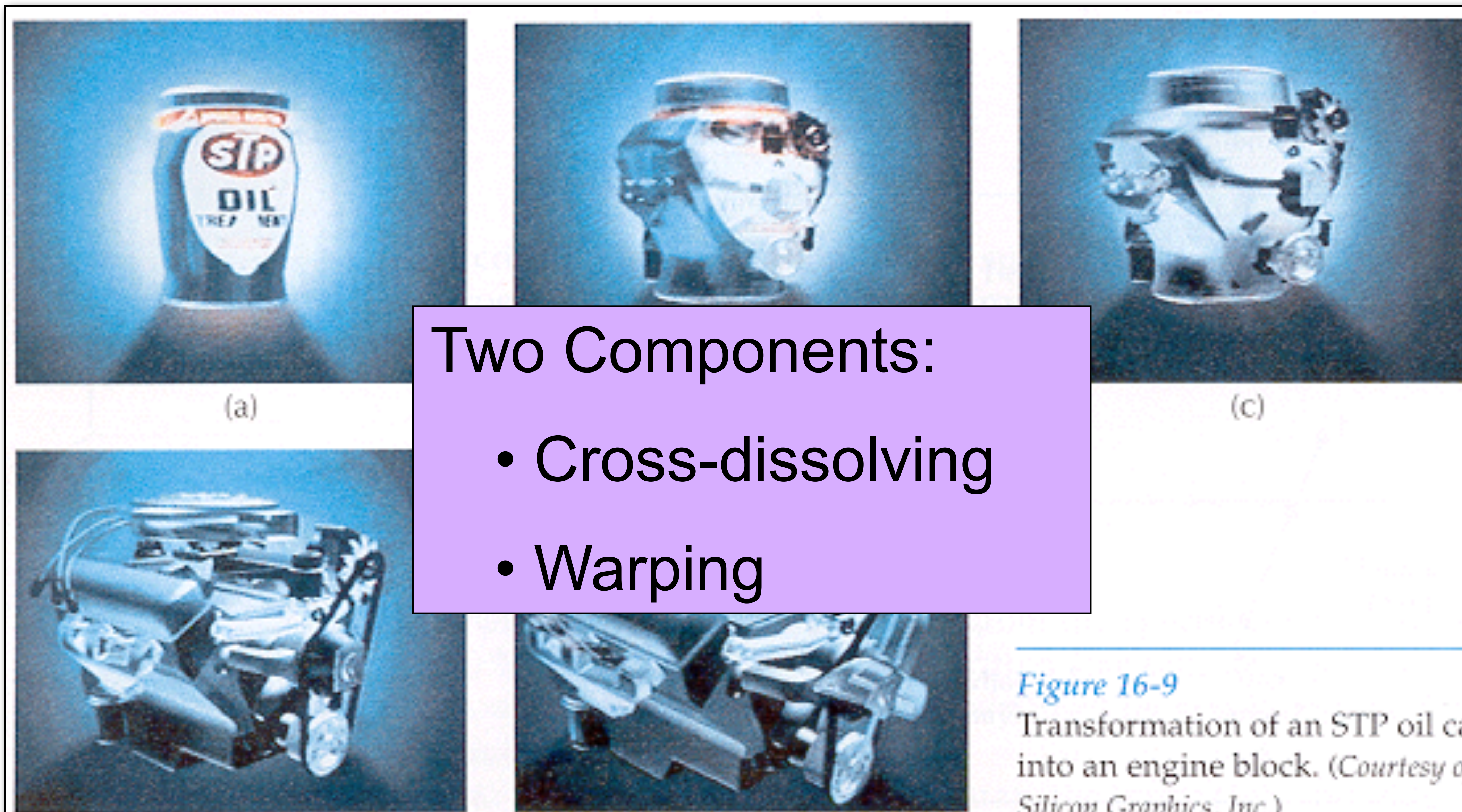


Figure 16-9
Transformation of an STP oil ca
into an engine block. (*Courtesy of*
Silicon Graphics, Inc.)

H&B Figure 16.9

Image Morphing

- Animate transition between two images



Two Components:

- Cross-dissolving
- Warping

H&B Figure 16.9

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

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

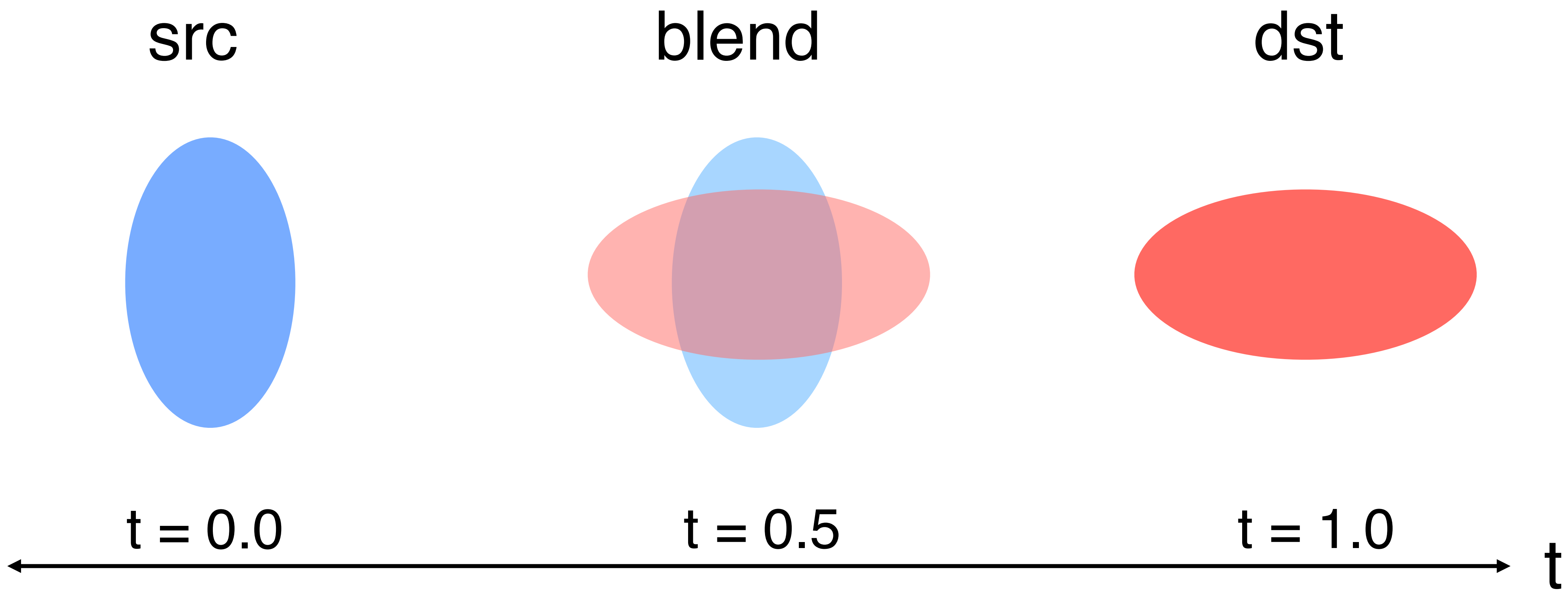


Image Warping

Deform the source so that it looks like the target

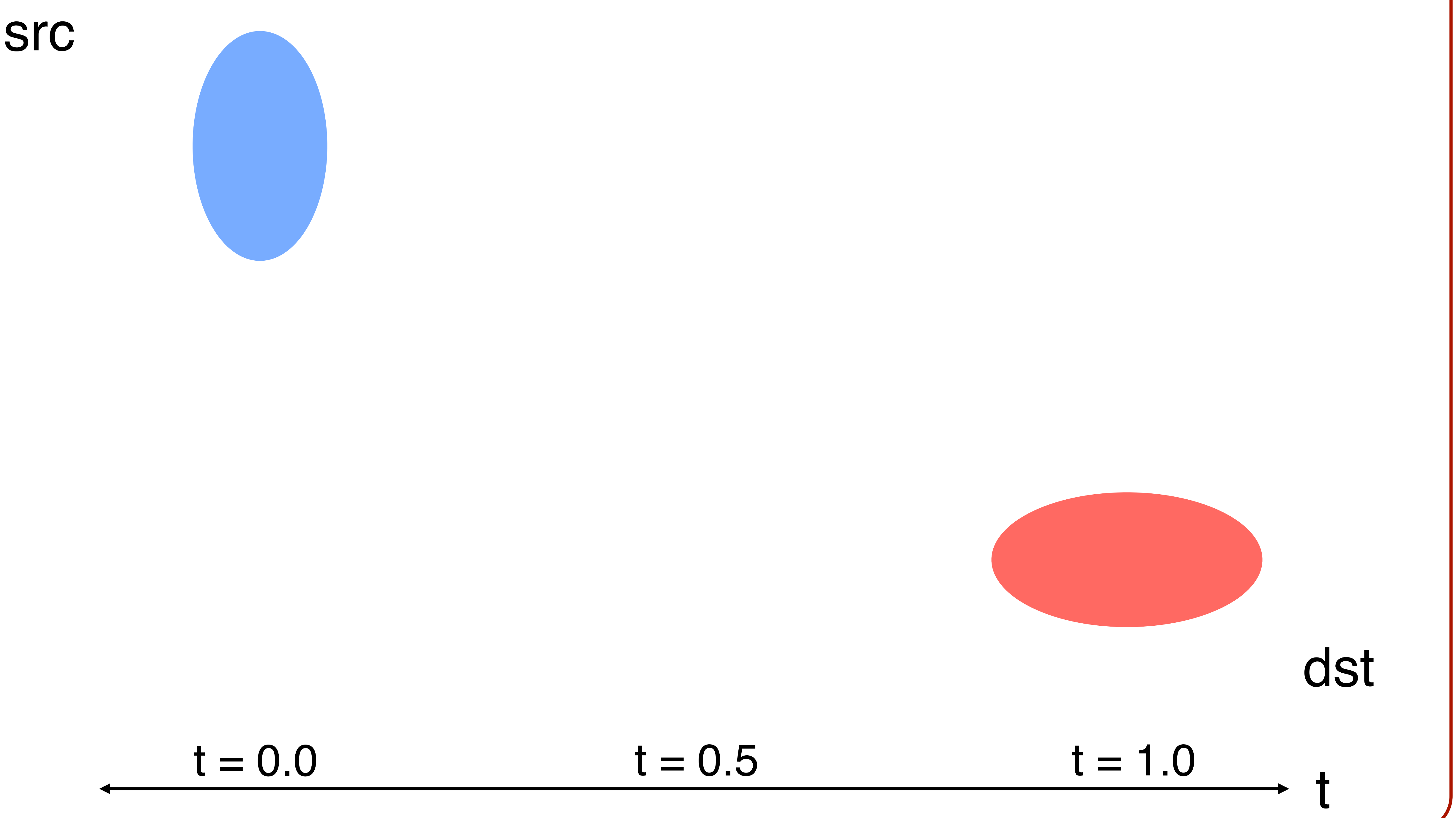


Image Warping

Deform the source so that it looks like the target

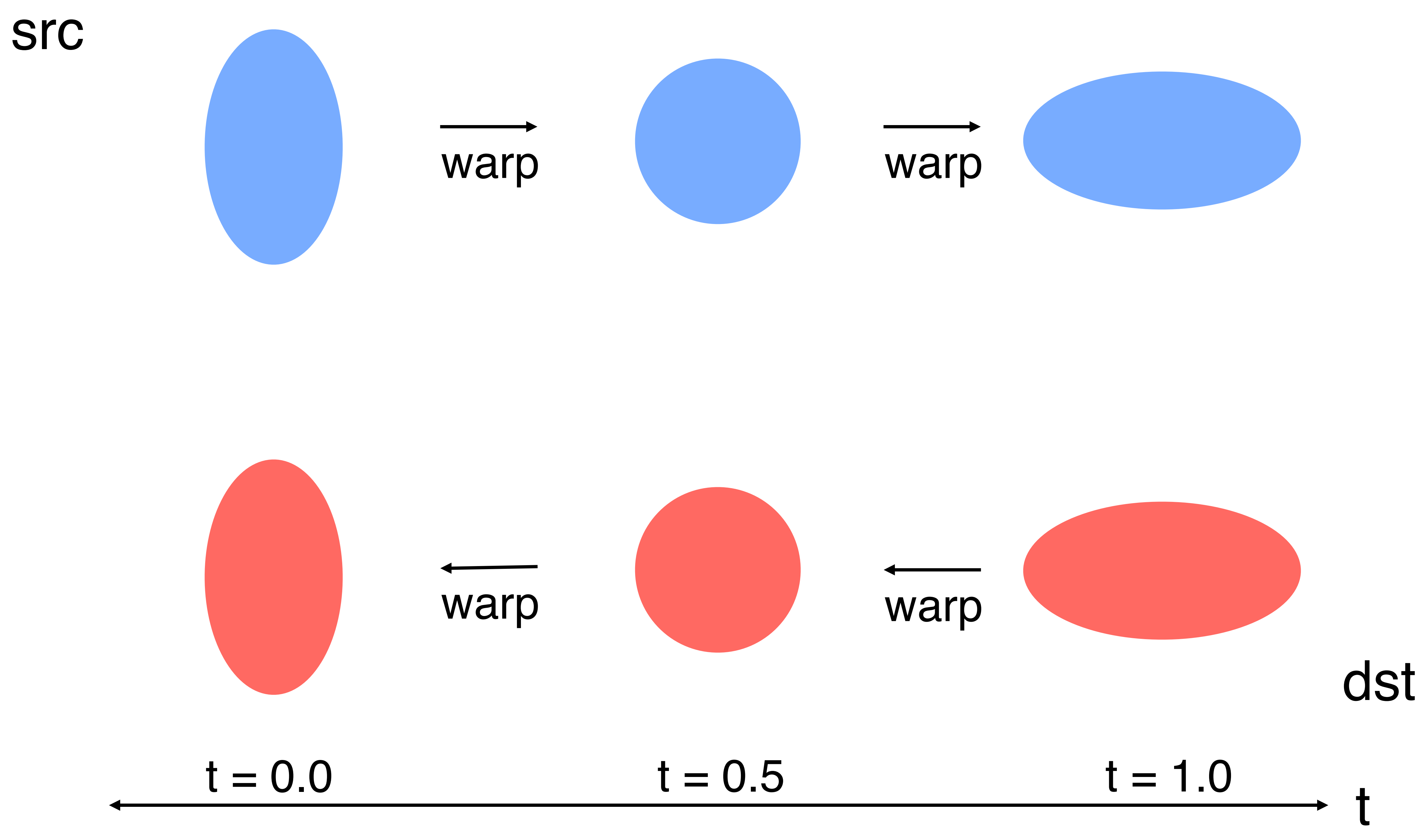


Image Morphing

Combines cross-dissolving and warping

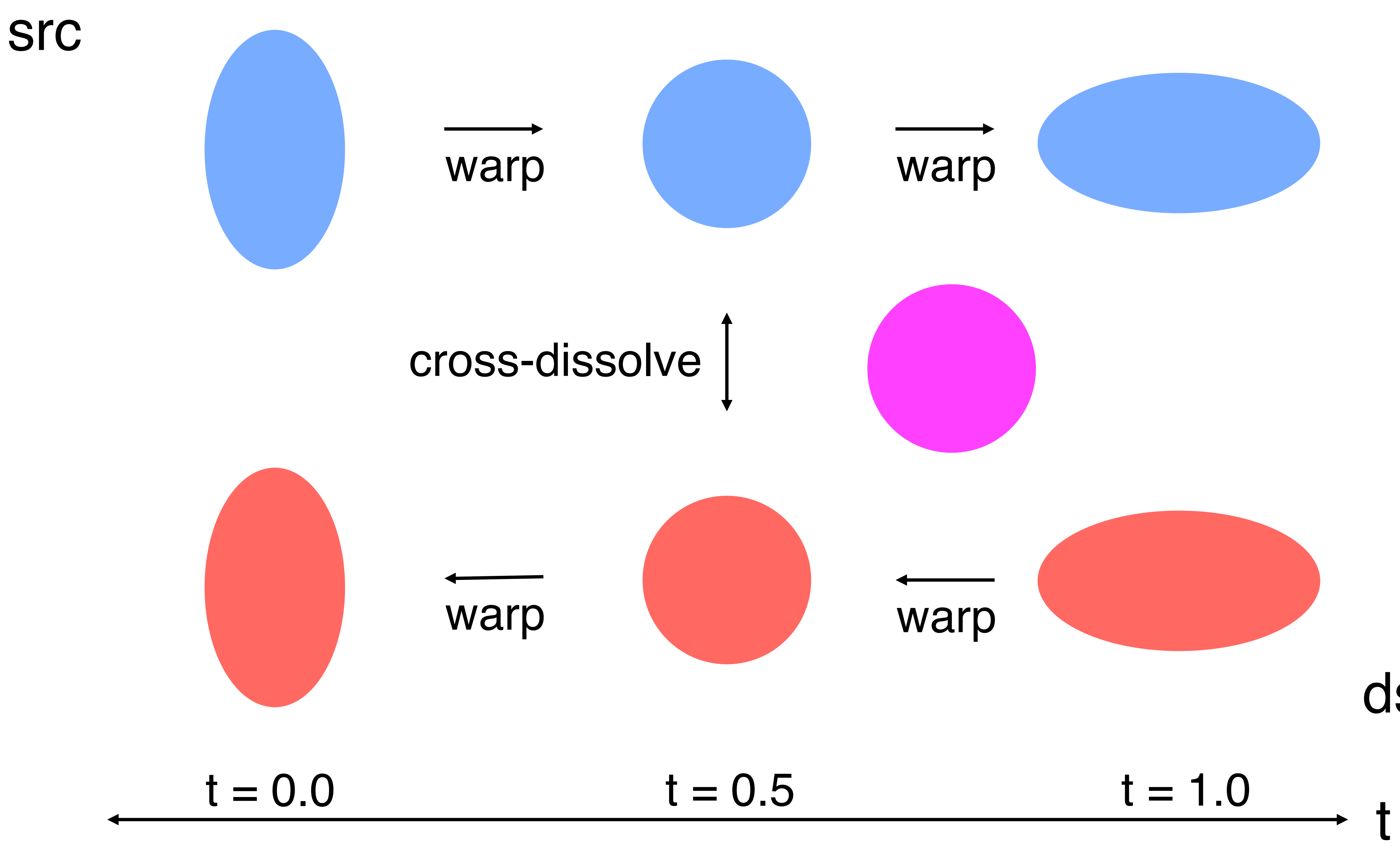
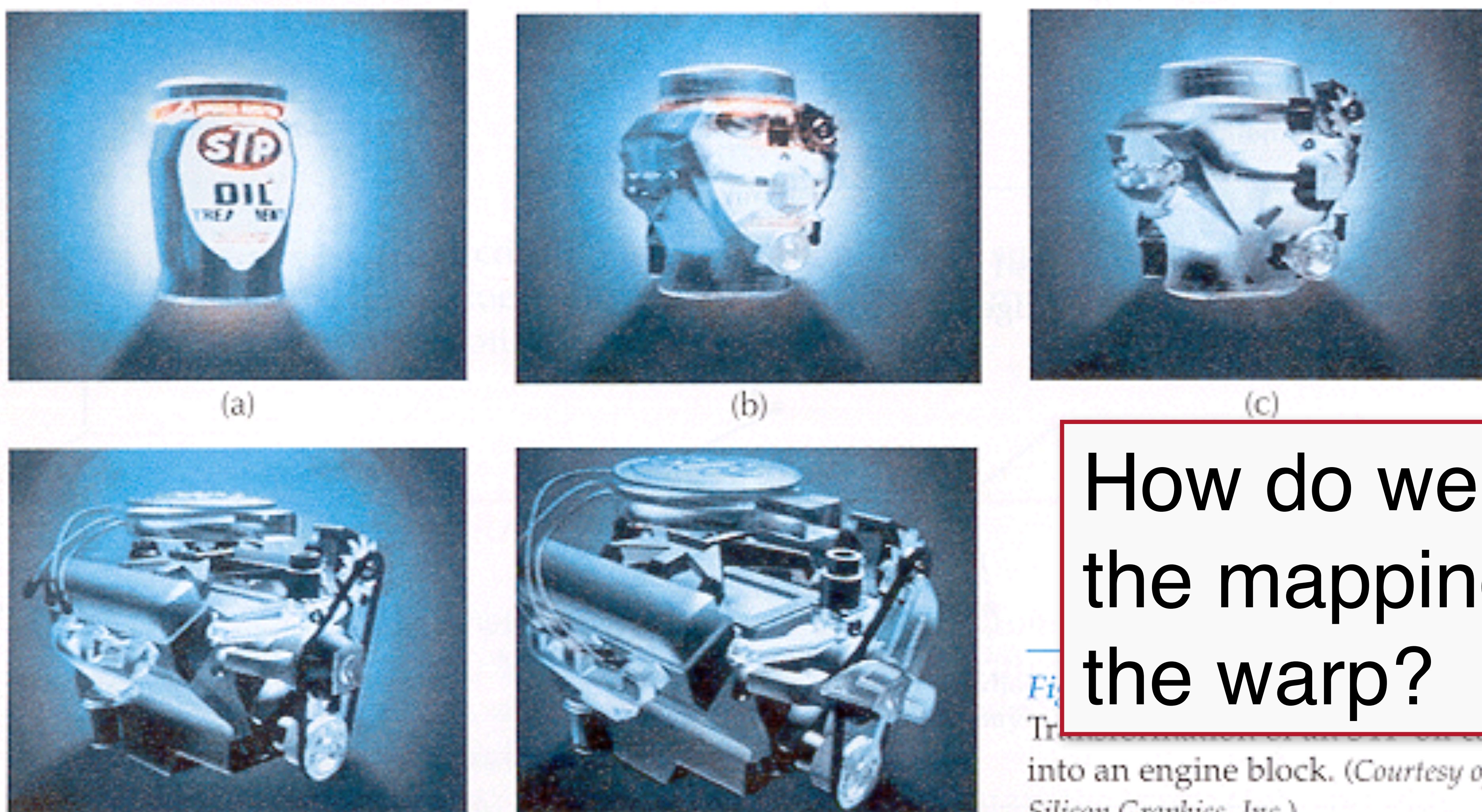


Image Morphing

- The warping step is the hard one
  - o Aim to align features in images

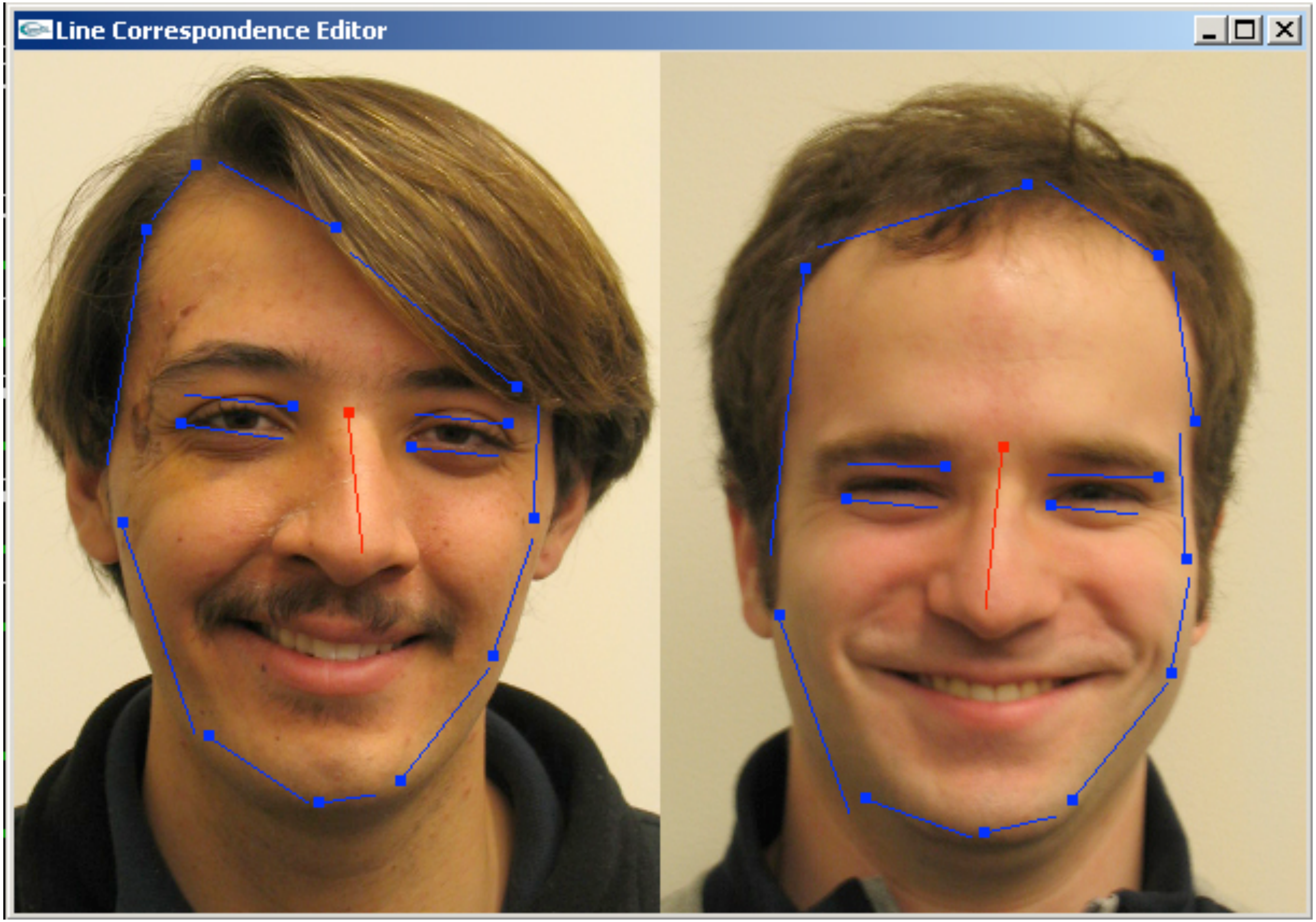


Fi
Tr
into an engine block. (*Courtesy of Silicon Graphics, Inc.*)

How do we specify the mapping for the warp?

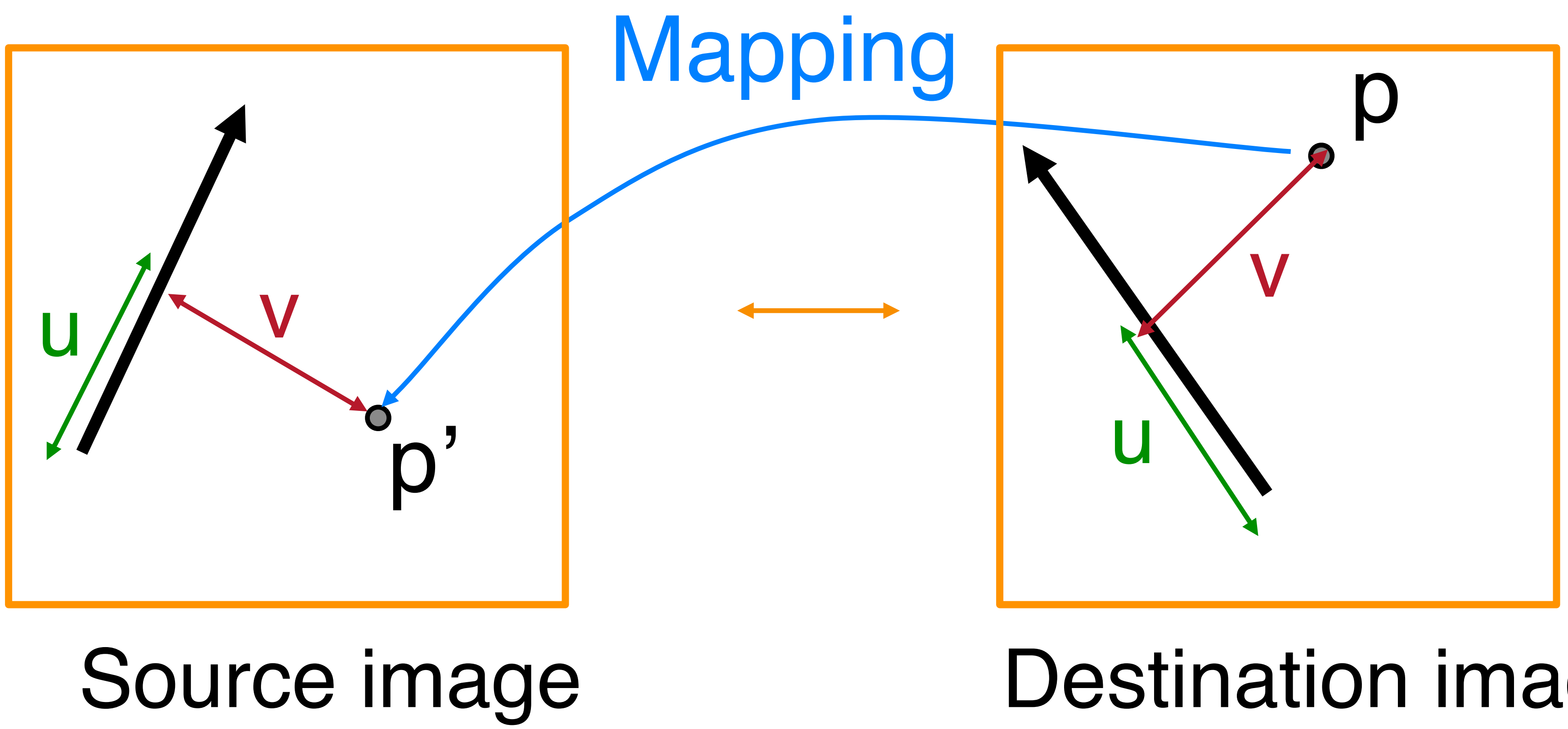
H&B Figure 16.9

Image Correspondence



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

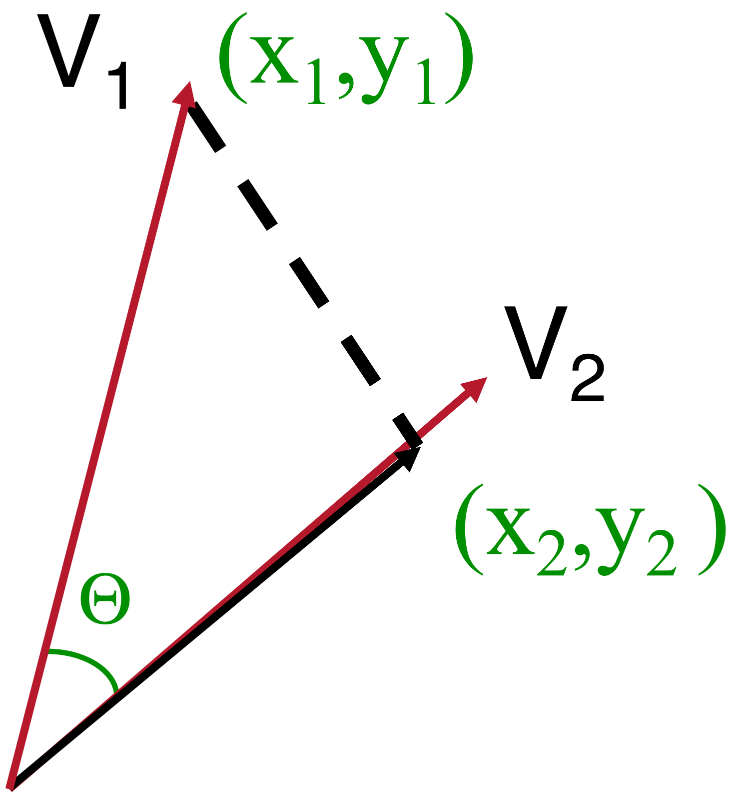
v is a length (in pixels)

Beier & Neeley
SIGGRAPH 92

Feature-Based Warping

How do I calculate u and v?

1. Recall the dot product
2. $V_1 \cdot V_2 = x_1 x_2 + y_1 y_2$
3. $V_1 \cdot V_2 = \|V_1\| \, \|V_2\| \cos(\Theta)$



Feature-Based Warping

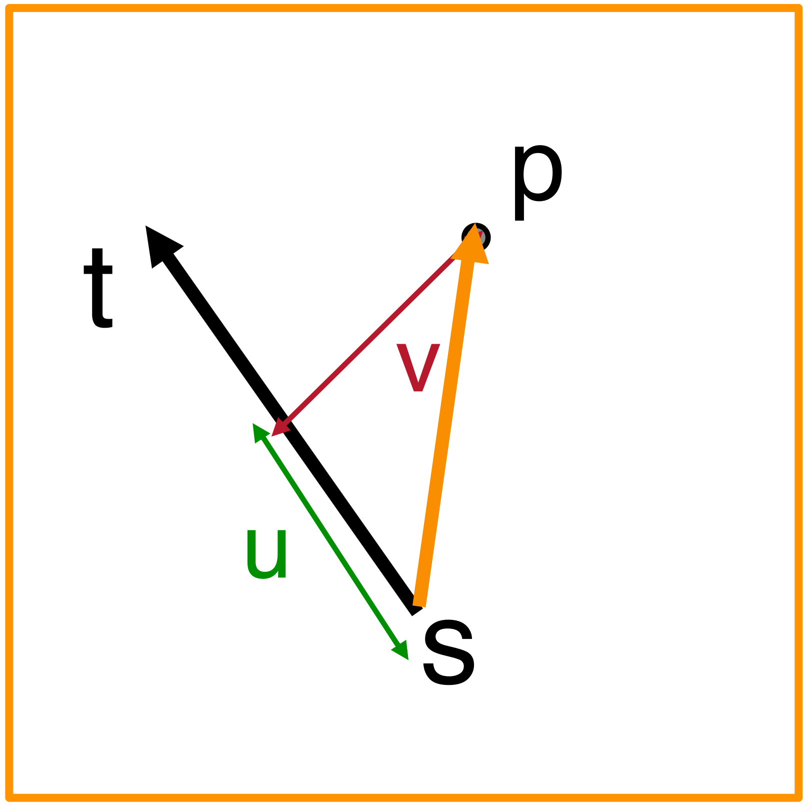
How do I calculate u and v?

$$u = \frac{(p - s) \bullet (t - s)}{\| t - s \|^2}$$

Equation 1 from B&N paper

Why?

Remember: u is a fraction



Feature-Based Warping

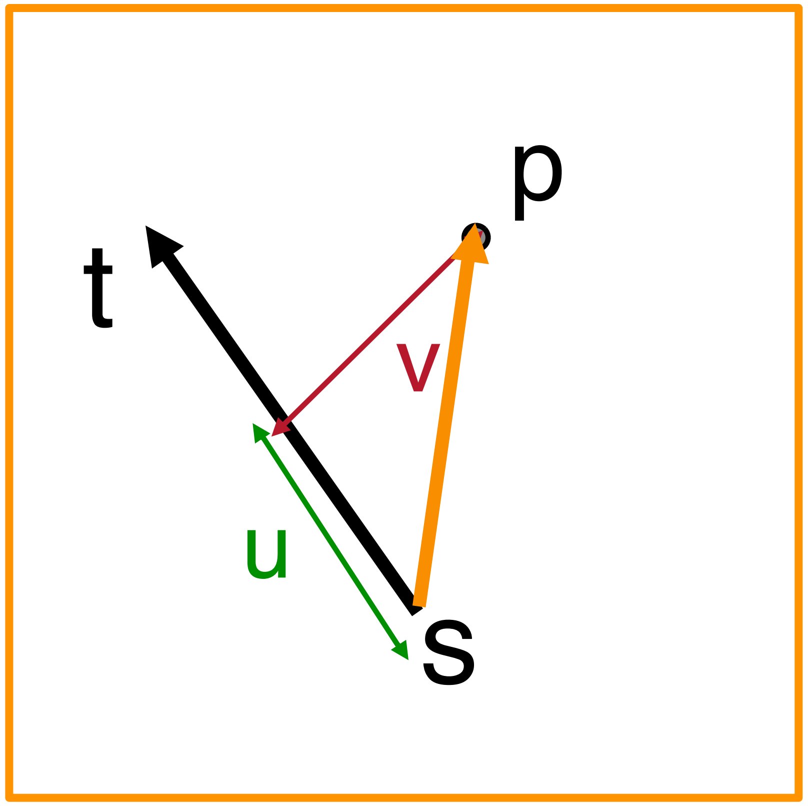
How do I calculate u and v?

$$v = \frac{(p - s) \bullet Perp(t - s)}{\| t - s \|}$$

Equation 2 from B&N paper

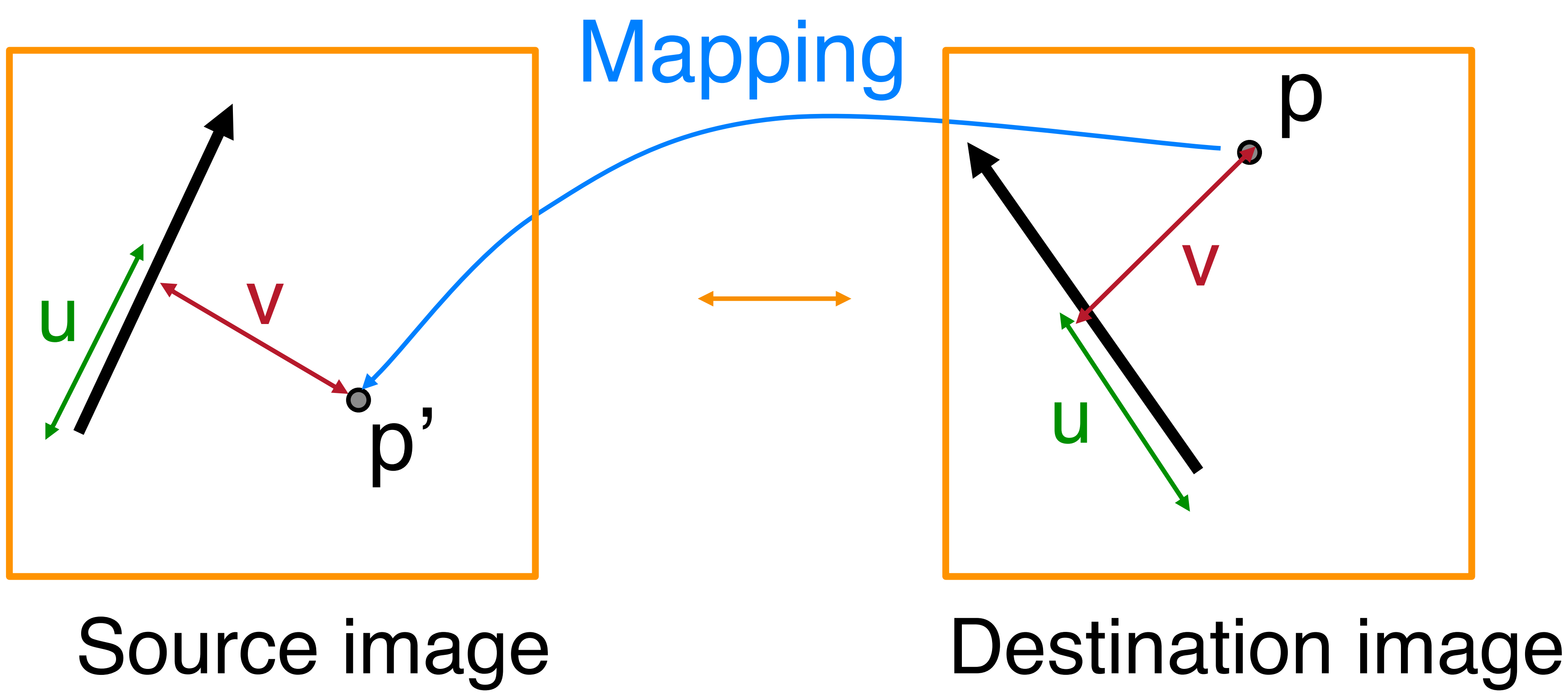
Why?

v is a length (in pixels)



Feature-Based Warping

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



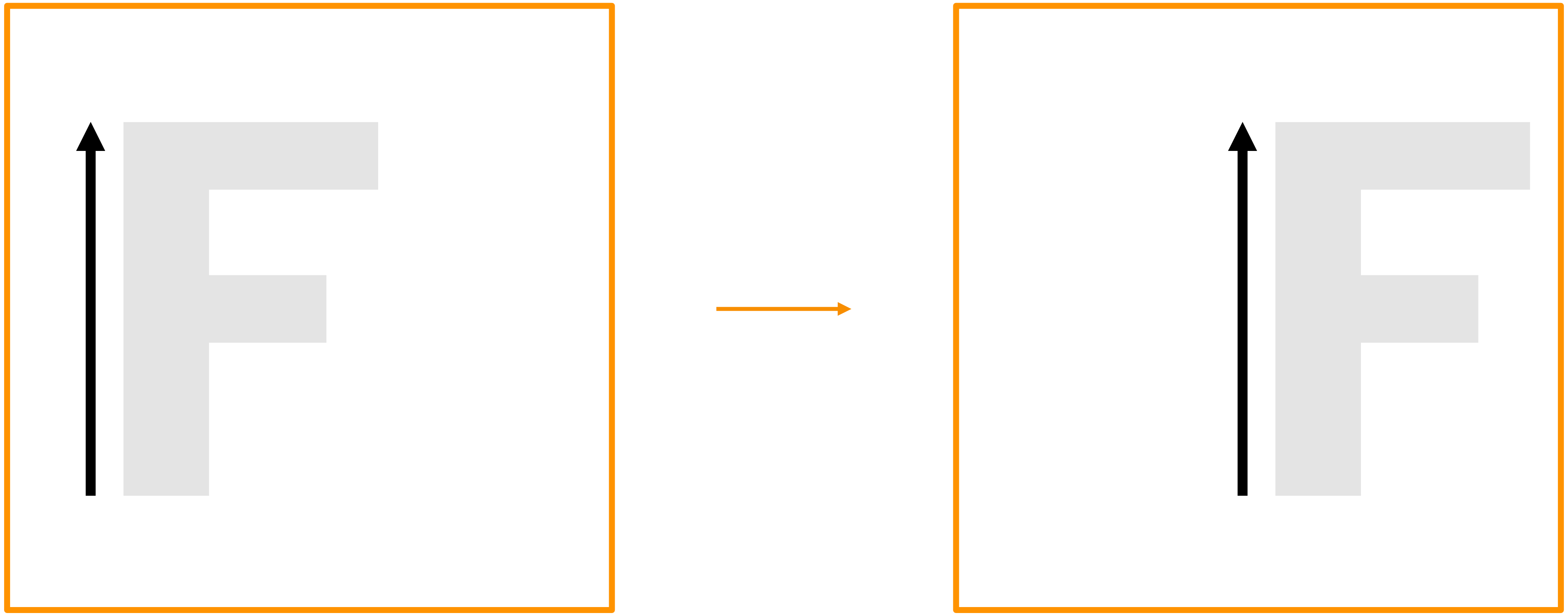
u is a fraction

v is a length (in pixels)

Beier & Neeley
SIGGRAPH 92

Warping with One Line Pair

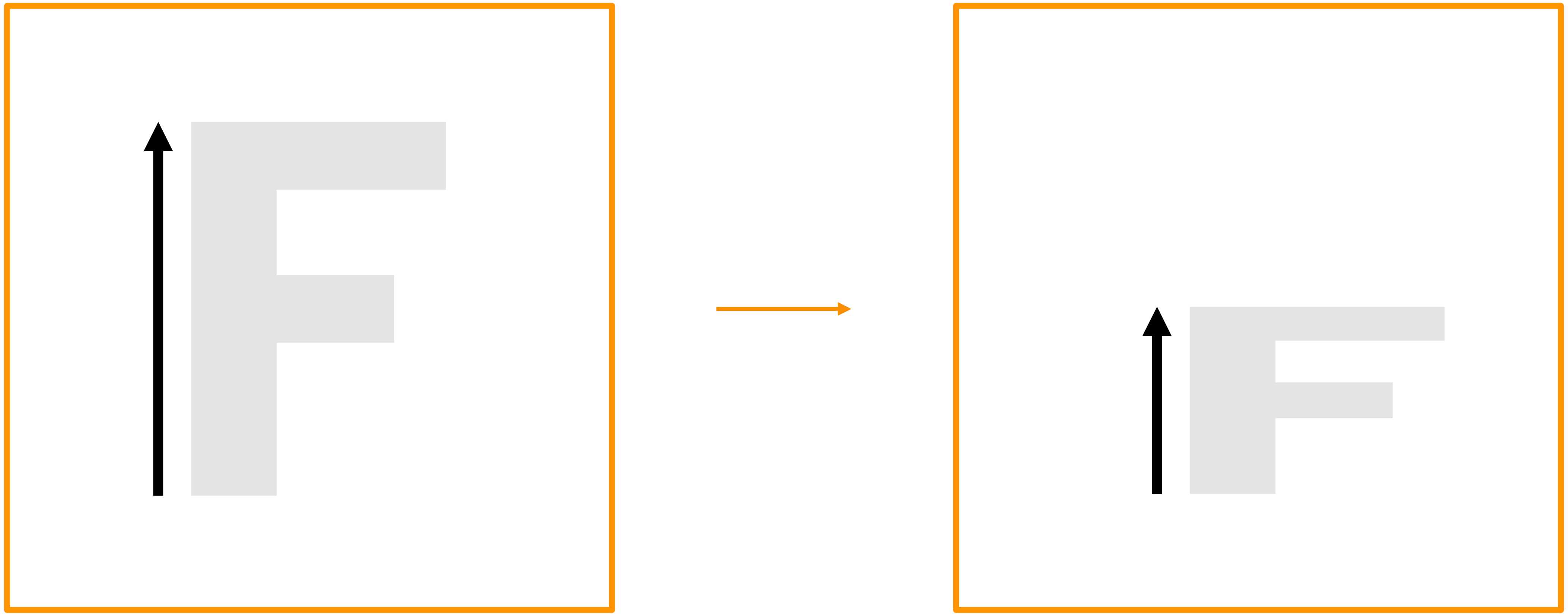
- What happens to the “F”?



Translation!

Warping with One Line Pair

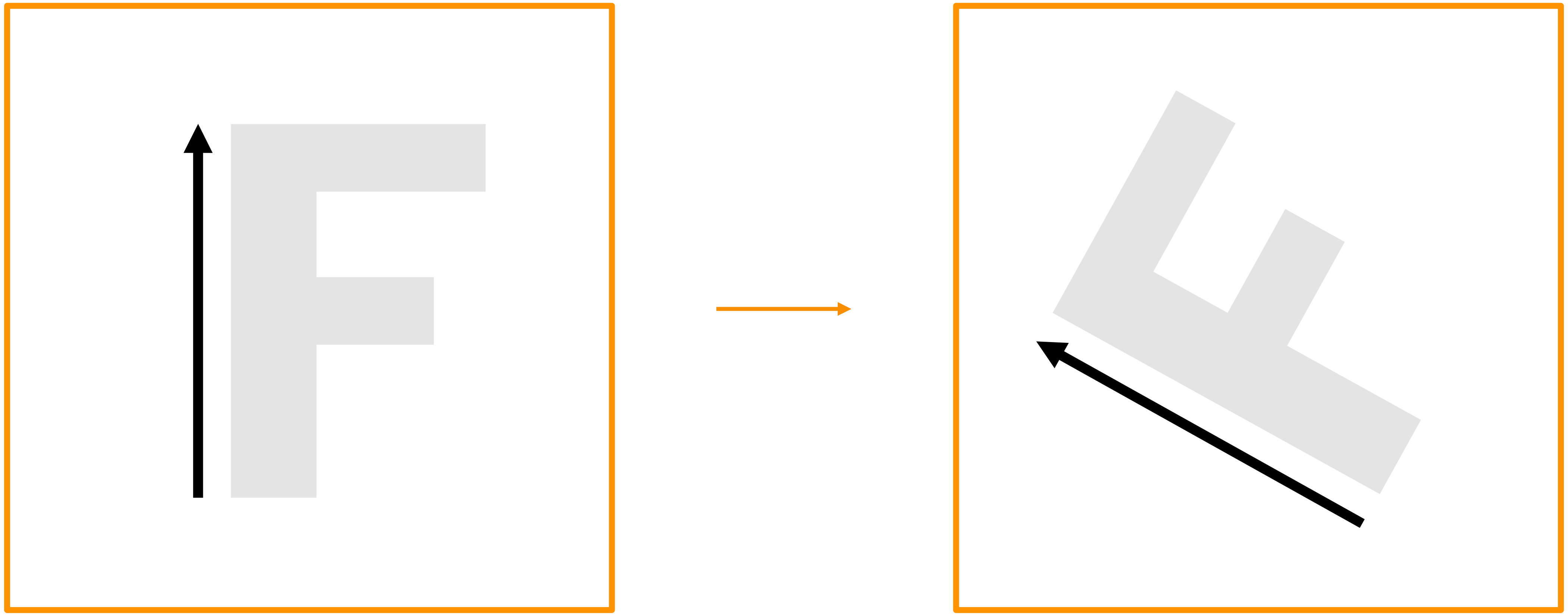
- What happens to the “F”?



Scale!

Warping with One Line Pair

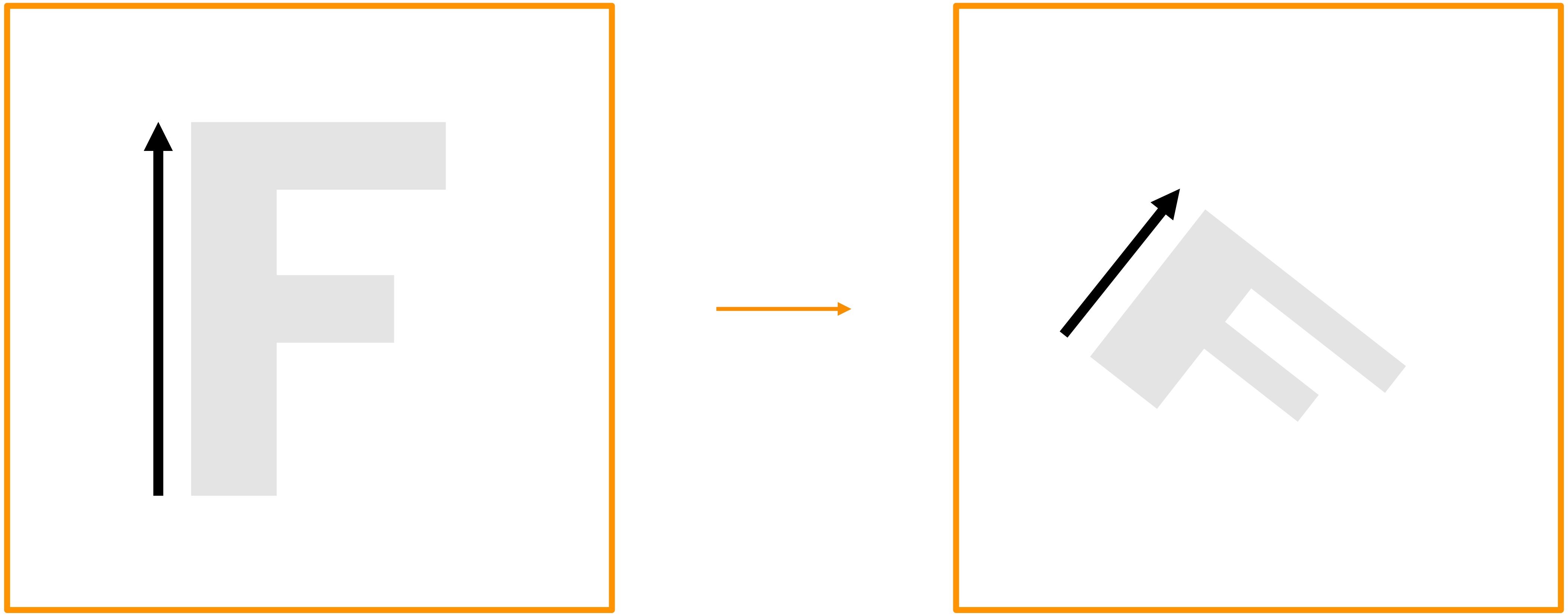
- What happens to the "F"?



Rotation!

Warping with One Line Pair

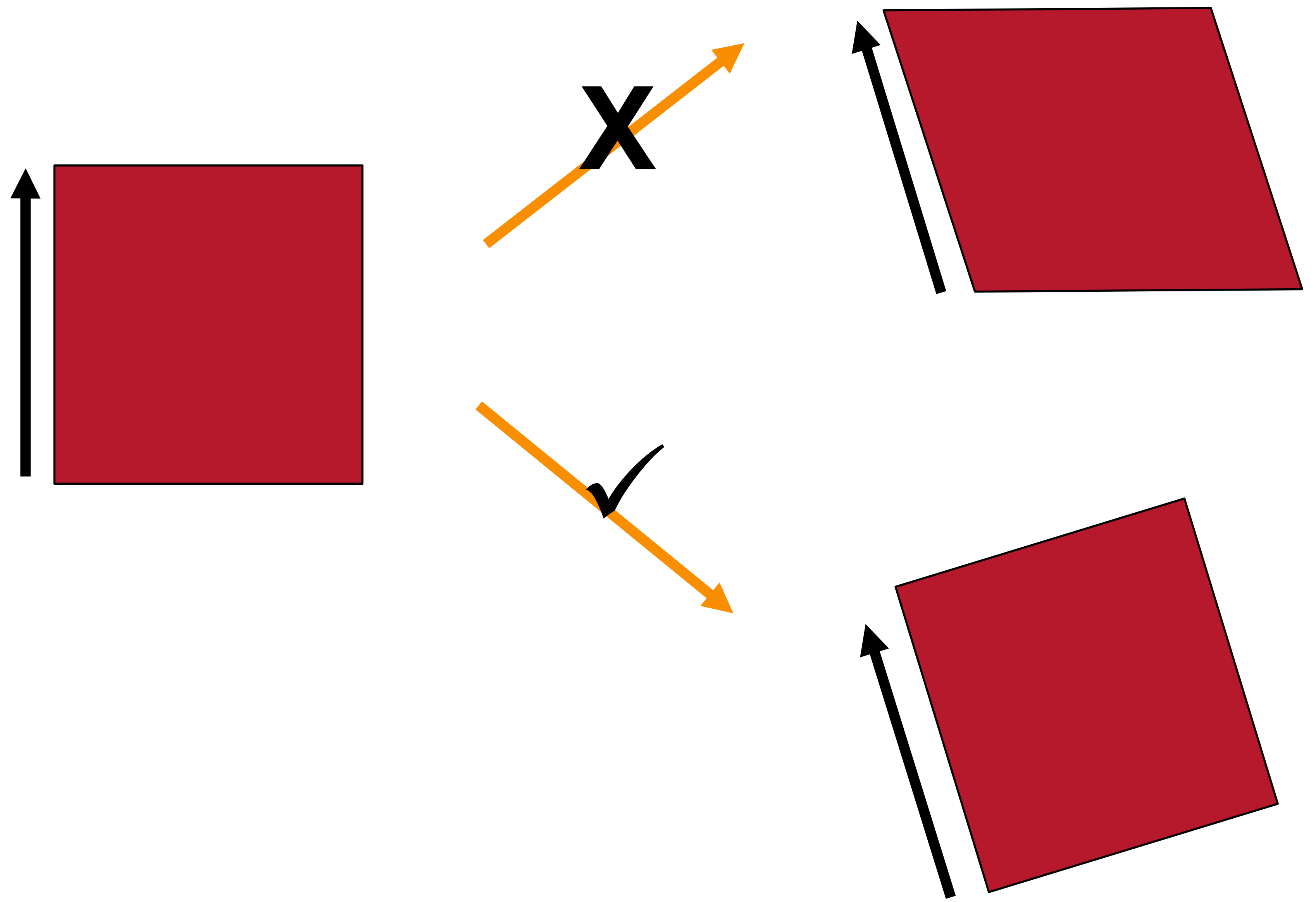
- What happens to the “F”?



What types of transformations can’t be specified?

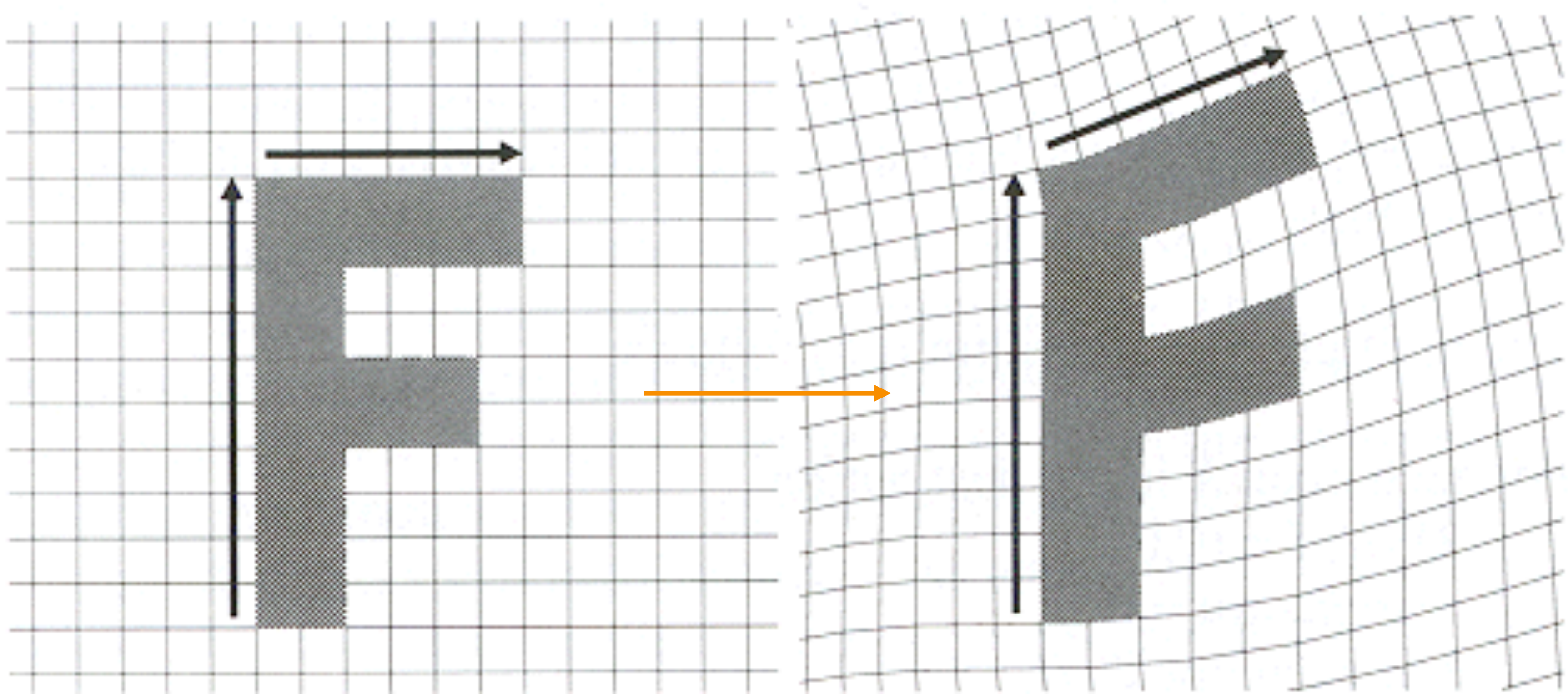
Warping with One Line Pair

- Can’t specify skews, mirrors, angular changes…



Warping with Multiple Line Pairs

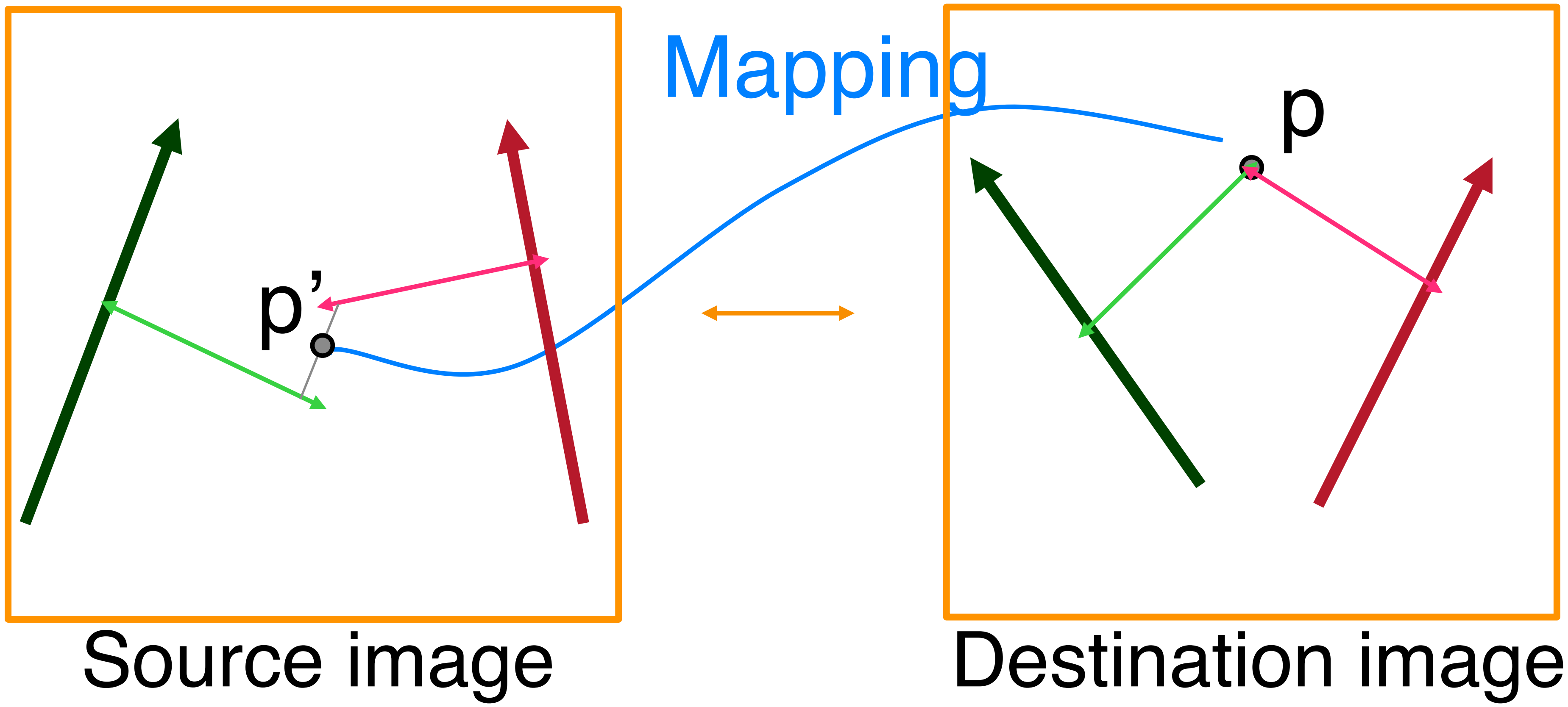
- Use weighted combination of points defined by each pair of corresponding lines



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 p to L[i]
- *a, b, p* are constants that control the warp

Feature-Based Warping

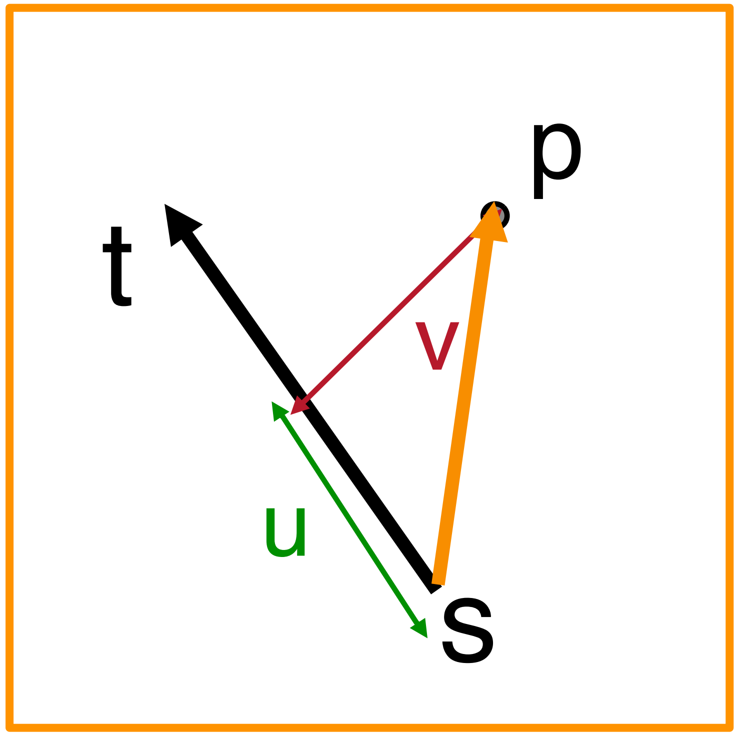
How do I calculate dist? Dist is either…

- abs(v) if u is >= 0 and <= 1

  OR

- distance to the closest endpoint i.e.

$$Min(\| p - s \|, \| p - t \|)$$



Warping Pseudocode

```
WarpImage(Image, L’[…], L[…])
begin
    for each destination pixel p do
        psum = (0,0)
        wsum = 0
        for each 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

```
WarpImage(Image, L'[…], L[…])
begin
    for each destination pixel p do
        psum = (0,0)
        wsum = 0
                                        [i])
                                        ]
            wsum += weight[i]
        end
        p' = psum / wsum
        Result(p) = Image(p')
    end
end
```

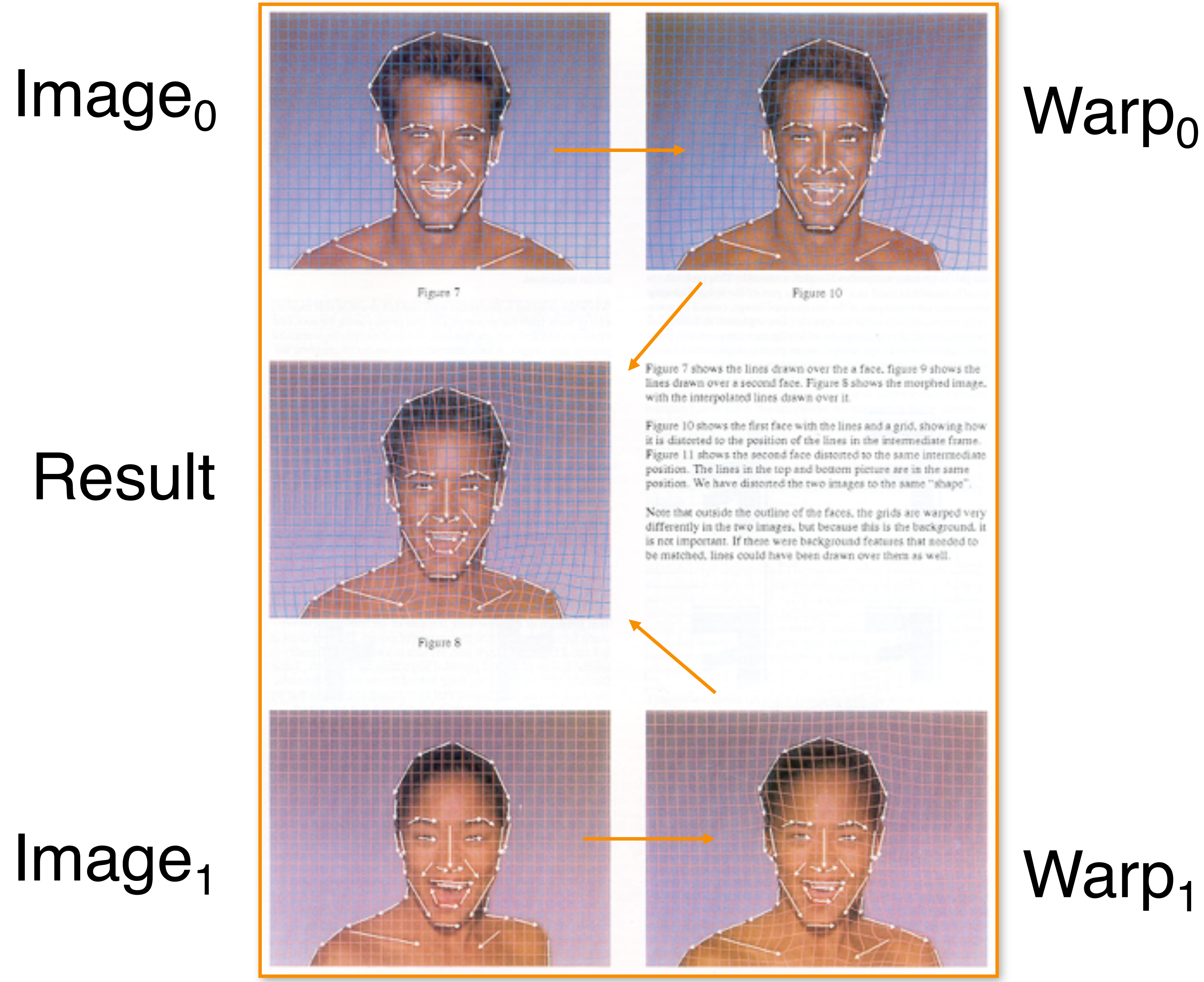
This warps the image so that the lines L' go to L

Morphing Pseudocode

```
GenerateAnimation(Image_0, L_0[…], Image_1, L_1[…])
begin
    for each intermediate frame time t do
        for i = 1 to number of line pairs do
            L[i] = line t-th of the way from L_0 [i] to L_1 [i]
        end
        Warp_0 = WarpImage(Image_0, L_0, L)
        Warp_1 = WarpImage(Image_1, L_1, L)
        for each pixel p in FinalImage do
            Result(p) = (1-t) Warp_0 + t Warp_1

    end
end
```

Beier & Neeley Example



Beier & Neeley Example

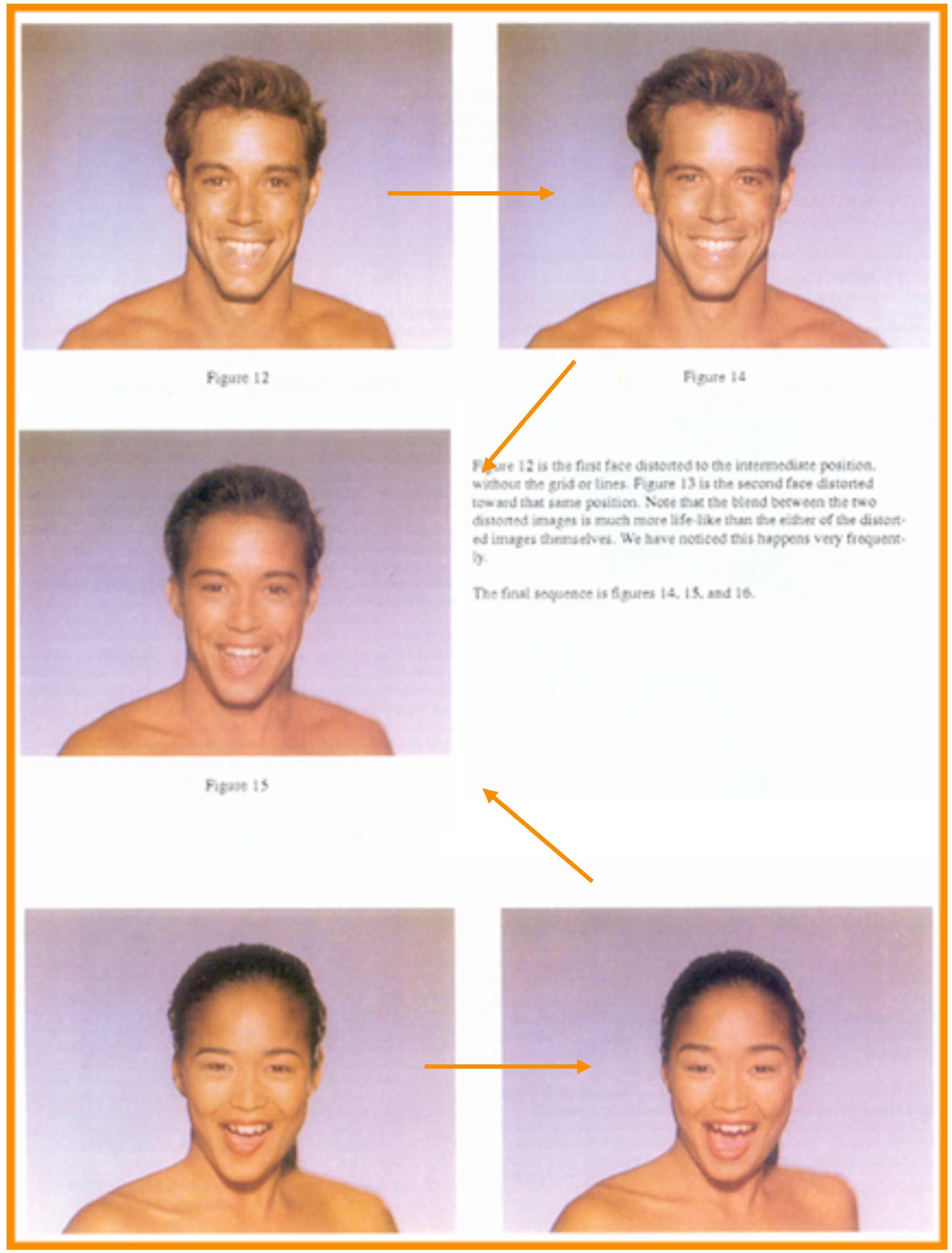


Image Processing

- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Composite
  - Morph

Summary: Image Processing

- Image representation
  - A pixel is a sample, not a little square
  - Images have limited resolution
  - Image processing is a resampling problem
- Halftoning and dithering
  - Reduce visual artifacts due to quantization
  - Distribute errors among pixels
  - Exploit spatial integration in our eye

Summary: Image Processing

- Sampling and reconstruction
  - Reduce visual artifacts due to aliasing
  - Filter to avoid undersampling
  - Blurring is better than aliasing