Image Compositing & Morphing

Felix Heide

COS 426, Spring 2022

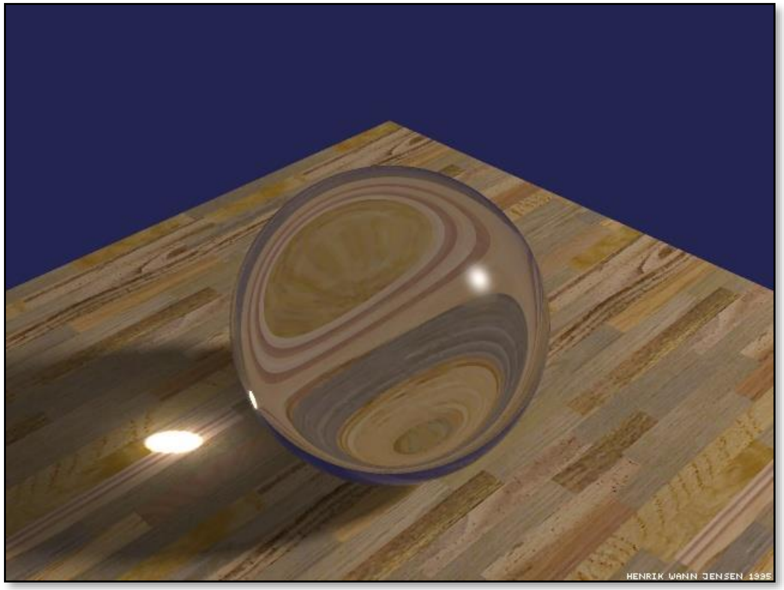
Princeton University

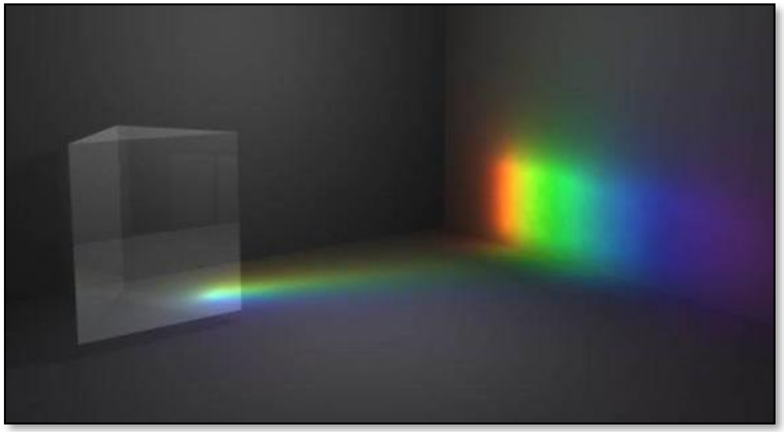
Digital Image Processing

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Types of Transparency

- Refraction
  - Light is bent as it goes through an object
  - Can focus light: caustics
  - Can be color-dependent: dispersion

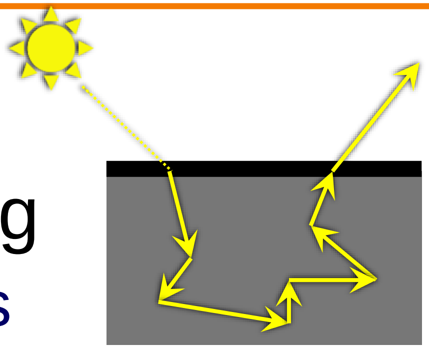




Types of Transparency

- Refraction

- Subsurface scattering
    - Translucent materials
    - Light leaves at different position than it entered



Types of Transparency

- Refraction
- Subsurface scattering
- Today: compositing
  - Separate image into layers with known order
  - ***Pixelwise*** combination: each pixel in each layer can be transparent, opaque, or somewhere in between

Smith & Blinn`84

Example

Jurassic Park (1993)

Image Composition

- Issues:
  - Segmenting image into regions
  - Blending into single image seamlessly

Image Composition

- Issues:
  - Segmenting image into regions
  - Blending into single image seamlessly

Image Matting

- Chroma keying (blue- or green-screen)
  - Photograph object in front of screen with known color

Rosco Spectrum

Image Matting

- Specify segmentation by hand
  - Purely manual: draw matte every frame
  - Semi-automatic: graph-cut (draw a few strokes)
    Separate image regions along minimal cuts (where edges measure differences between adjacent pixels)

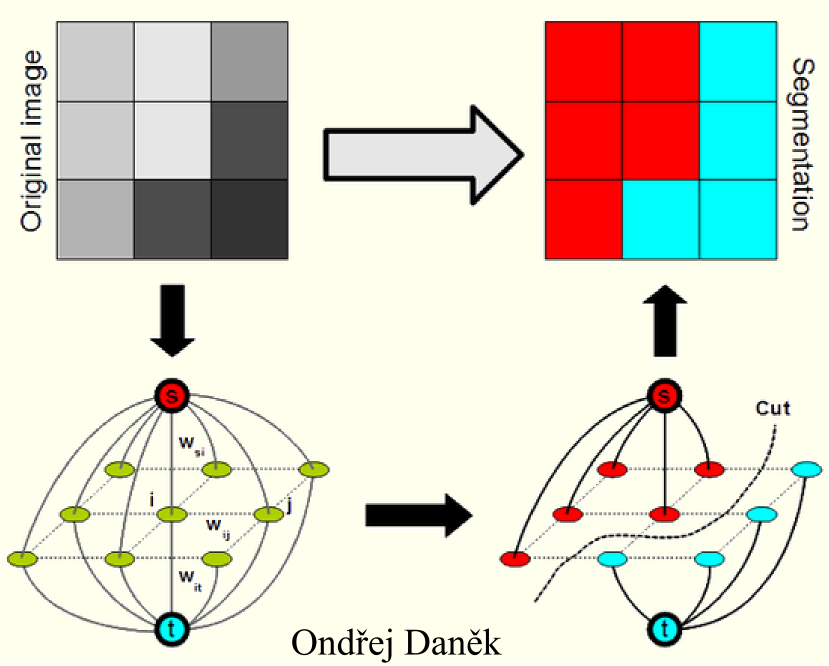


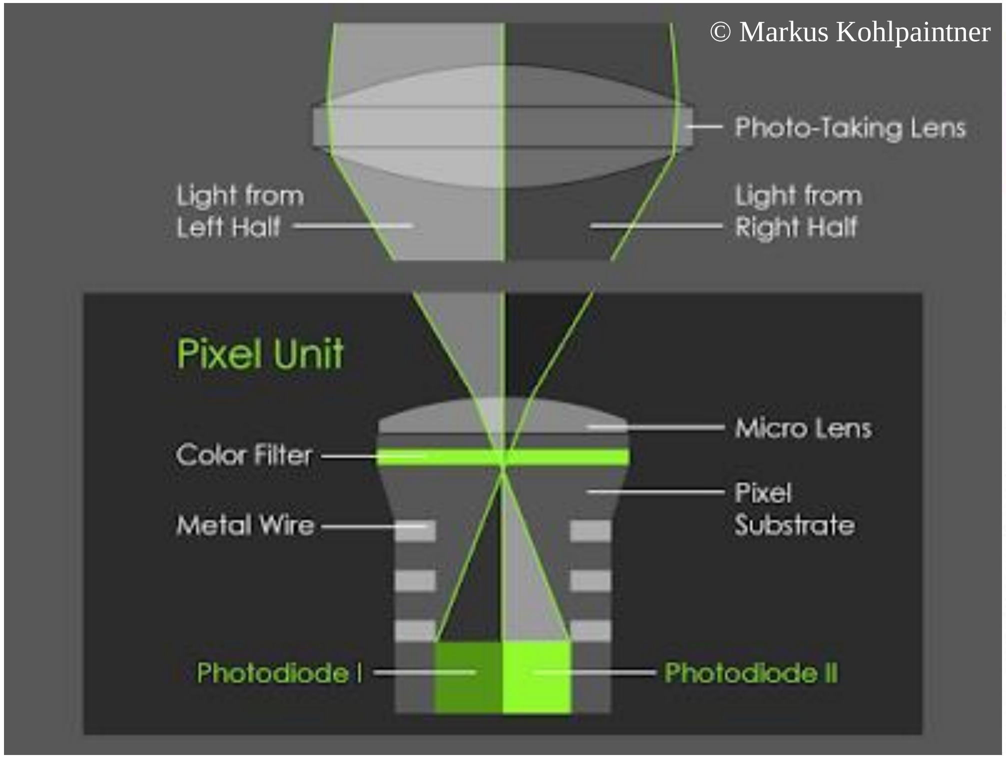
Image Matting

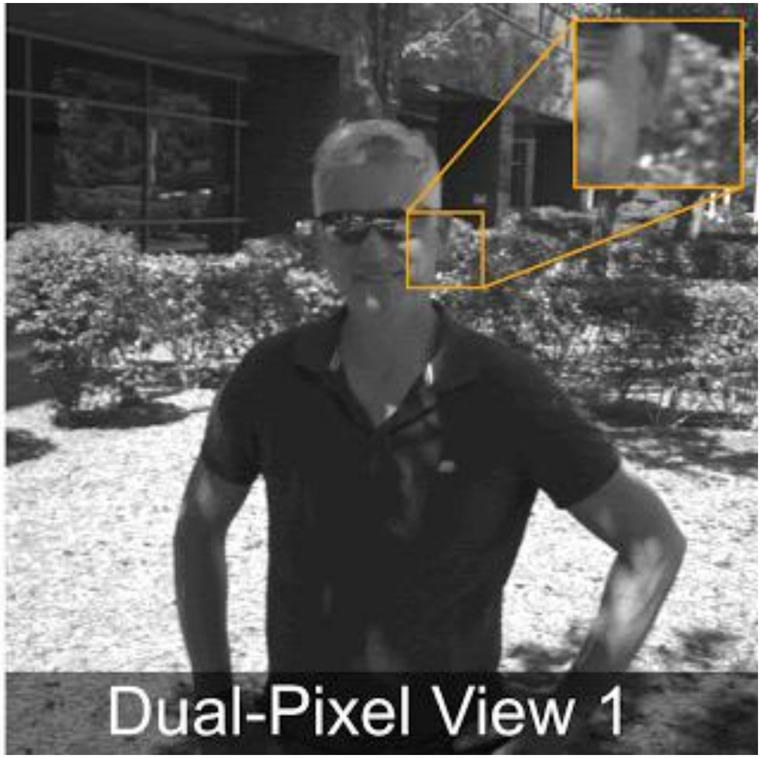
- Novel methods, e.g. flash matting

Sun et al., 2006

Image Matting

- Portrait mode in Google Pixel Phone

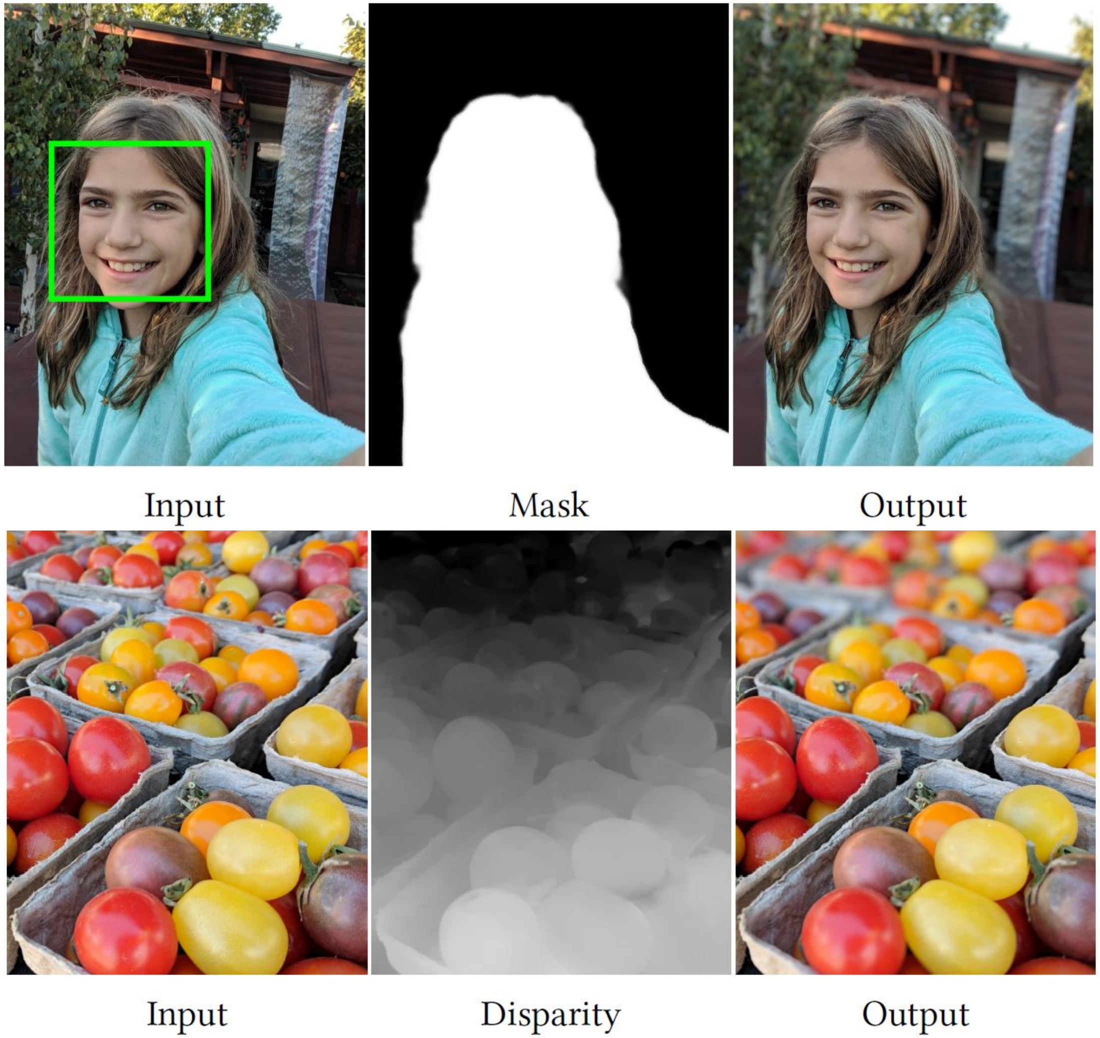




Wadhwa et al., 2018

Image Matting

- Portrait mode blur in Google Pixel Phones



Wadhwa et al., 2018

Image Composition

- Issues:
  - Segmenting image into regions
  - Blending into single image seamlessly

Image Blending

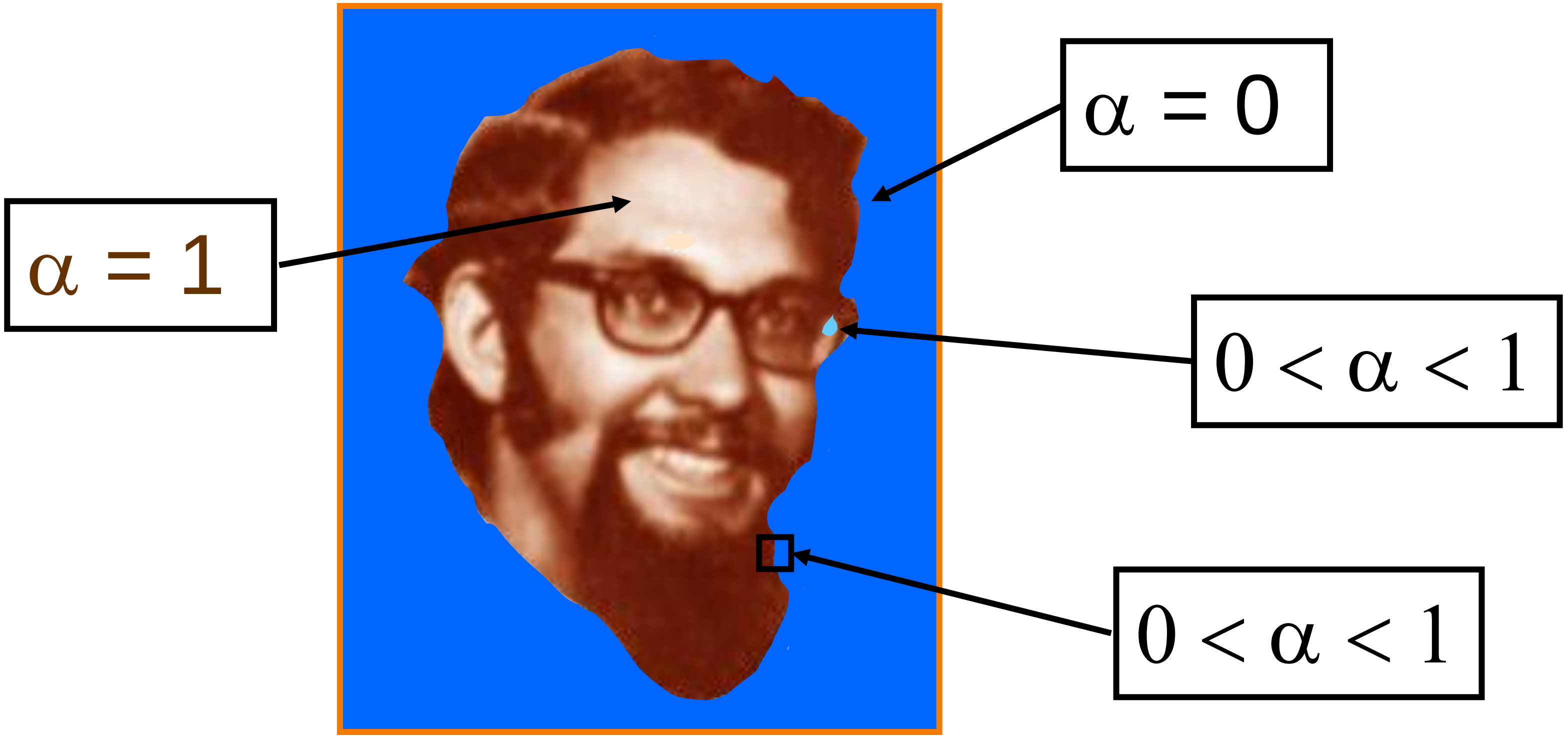
- Ingredients
  - Background image
  - Foreground image with blue background
- Method
  - Non-blue foreground pixels overwrite background

Blending with Alpha Channel

Per-pixel “alpha” channel: controls the linear interpolation between foreground and background pixels when elements are composited.

Blending with Alpha Channel

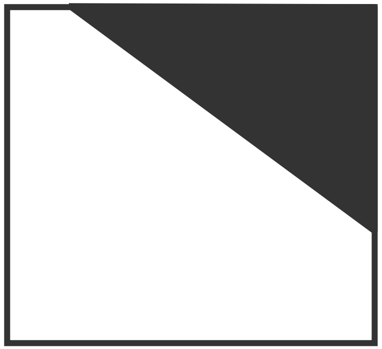
Per-pixel "alpha" channel: controls the linear interpolation between foreground and background pixels when elements are composited.



Alpha Channel

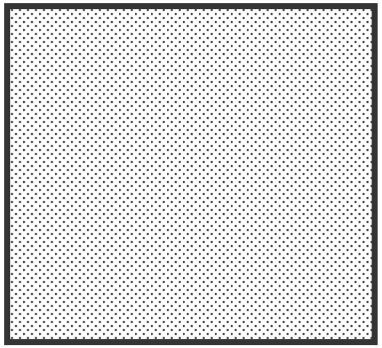
- Encodes pixel coverage information
  - $\alpha = 0$: no coverage (or transparent)
  - $\alpha = 1$: full coverage (or opaque)
  - $0 < \alpha < 1$: partial coverage (or semi-transparent)

- Example: $\alpha = 0.3$



Partial Coverage

or



Semi-Transparent

Alpha Blending: "Over" Operator

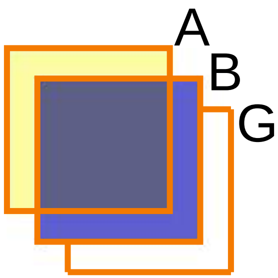
C = A over B

$C = \alpha_A A + (1-\alpha_A) B$

$0 < \alpha < 1$

Compositing Algebra

- Suppose we put A over B over background G

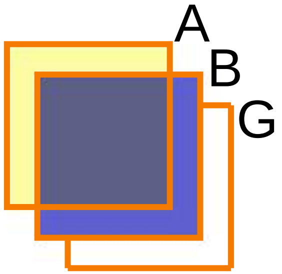


  - How much of B is blocked by A?

$$\alpha_A$$

Compositing Algebra

- Suppose we put A over B over background G



  - How much of B is blocked by A?

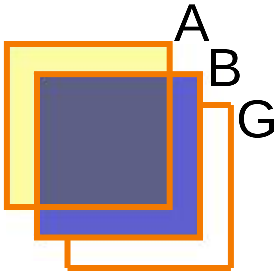
$$\alpha_A$$

  - How much of B shows through A

$$(1-\alpha_A)$$

Compositing Algebra

- Suppose we put A over B over background G



  - How much of B is blocked by A?

$$\alpha_A$$

  - How much of B shows through A

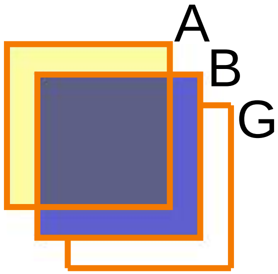
$$(1-\alpha_A)$$

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

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

Compositing Algebra

- Suppose we put A over B over background G



  - Final result?

$$\alpha_A A + (1-\alpha_A)\alpha_B B + (1-\alpha_A)(1-\alpha_B)G$$

$$= \alpha_A A + (1-\alpha_A)\left[\alpha_B B + (1-\alpha_B)G\right]$$

$$= A \text{ over } \left[B \text{ over } G\right]$$

Must perform "over" back-to-front: right associative!

Other Compositing Operations

Composition algebra – 12 combinations

$C' = F_A \alpha_A A + F_B \alpha_B B$

Operation	F_A	F_B
Clear	0	0
A	1	0
B	0	1

Porter & Duff `84

Other Compositing Operations

Composition algebra – 12 combinations

$$C' = F_A \, \alpha_A \, A + F_B \, \alpha_B \, B$$

Operation	F_A	F_B
Clear	0	0
A	1	0
B	0	1
A over B	1	$1-\alpha_A$
B over A	$1-\alpha_B$	1

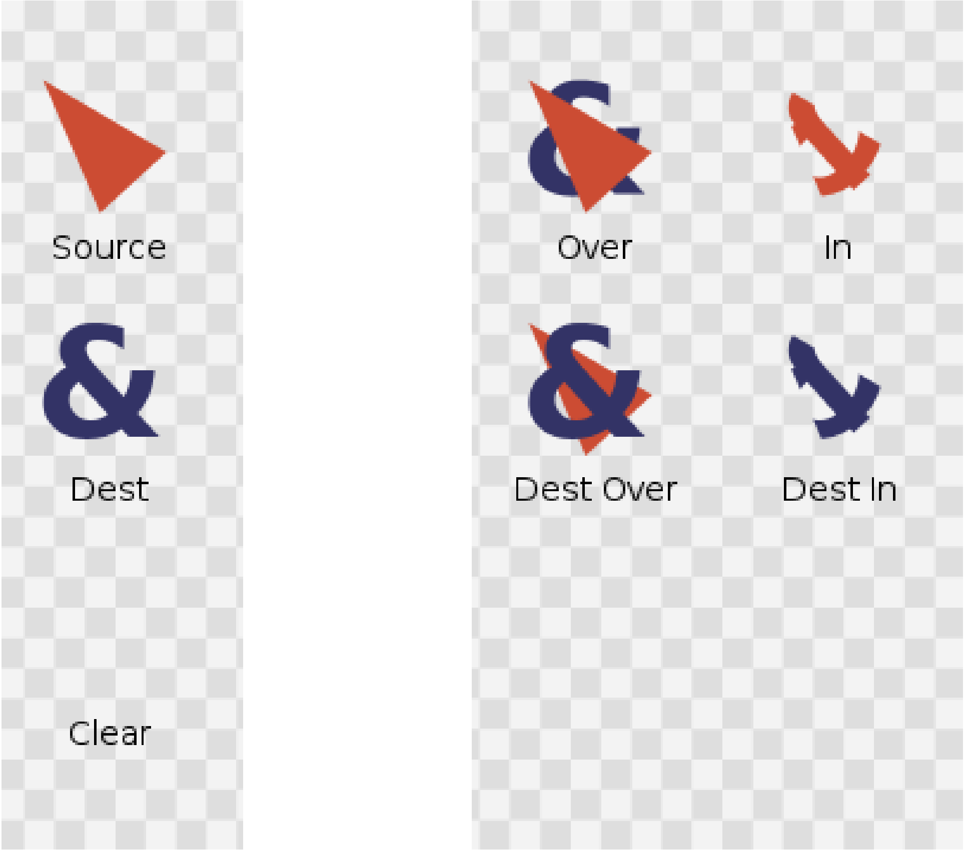
Porter & Duff `84

Other Compositing Operations

Composition algebra – 12 combinations

$$C' = F_A \alpha_A A + F_B \alpha_B B$$

Operation	F_A	F_B
Clear	0	0
A	1	0
B	0	1
A over B	1	$1-\alpha_A$
B over A	$1-\alpha_B$	1
A in B	α_B	0
B in A	0	α_A



Porter & Duff `84

Other Compositing Operations

Composition algebra – 12 combinations

$$C' = F_A\, \alpha_A\, A + F_B\, \alpha_B\, B$$

Operation	F_A	F_B
Clear	0	0
A	1	0
B	0	1
A over B	1	$1-\alpha_A$
B over A	$1-\alpha_B$	1
A in B	α_B	0
B in A	0	α_A
A out B	$1-\alpha_B$	0
B out A	0	$1-\alpha_A$

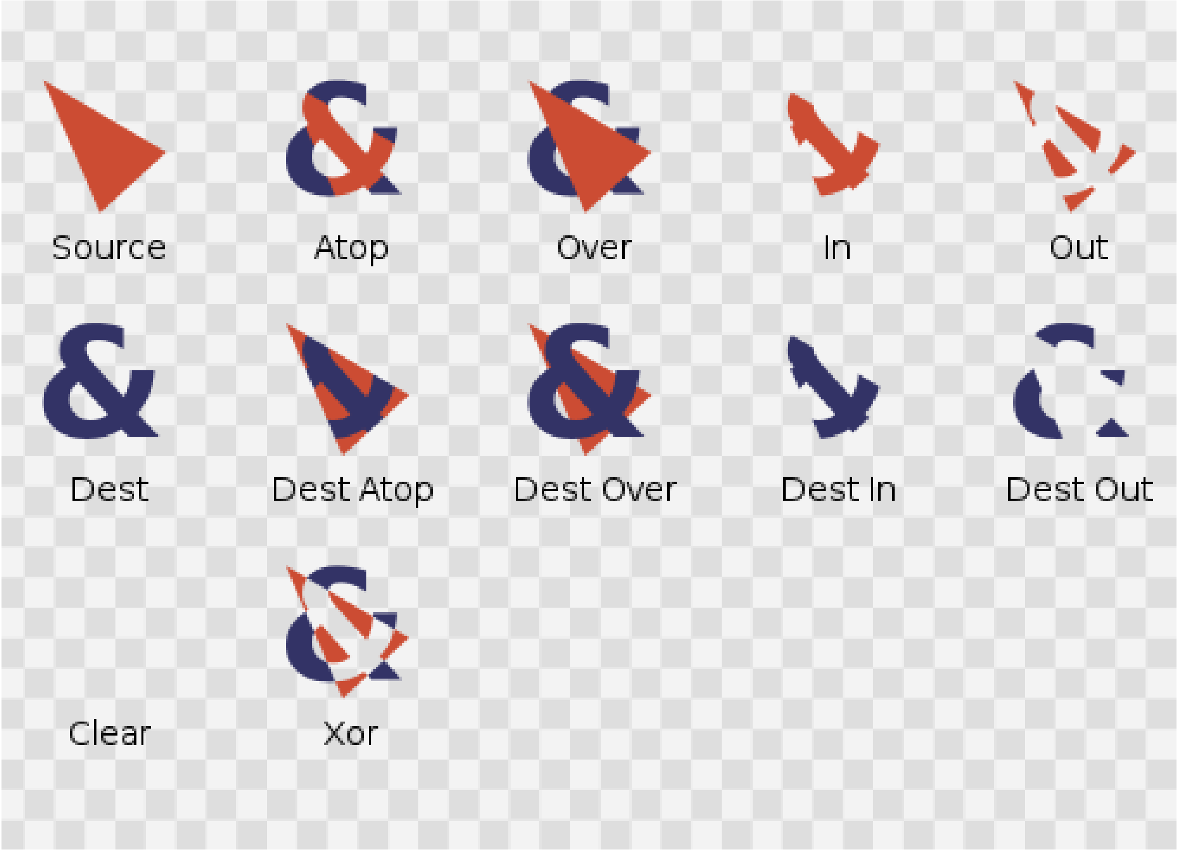
Porter & Duff `84

Other Compositing Operations

Composition algebra – 12 combinations

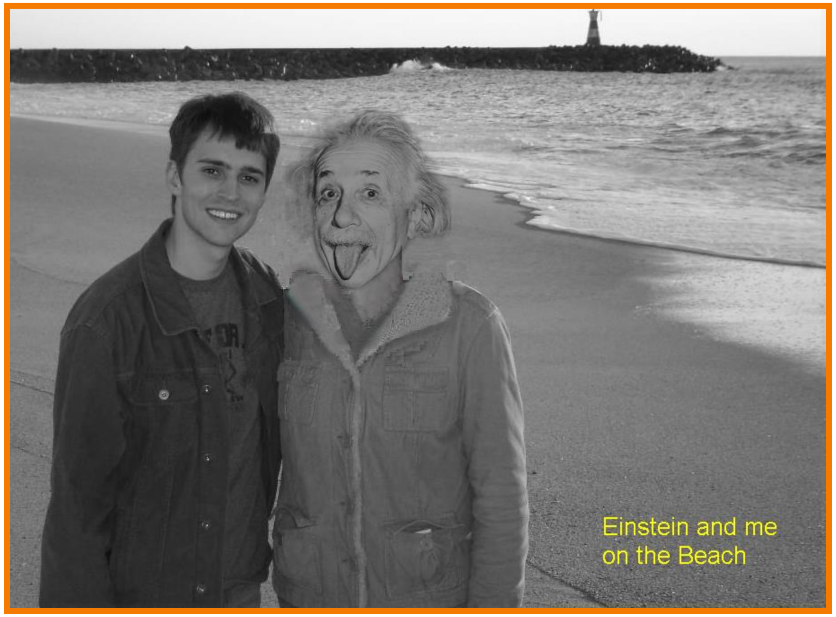
$C' = F_A \alpha_A A + F_B \alpha_B B$

Operation	F_A	F_B
Clear	0	0
A	1	0
B	0	1
A over B	1	$1-\alpha_A$
B over A	$1-\alpha_B$	1
A in B	α_B	0
B in A	0	α_A
A out B	$1-\alpha_B$	0
B out A	0	$1-\alpha_A$
A atop B	α_B	$1-\alpha_A$
B atop A	$1-\alpha_B$	α_A
A xor B	$1-\alpha_B$	$1-\alpha_A$



Porter & Duff `84

COS426 Examples



Darin Sleiter

Kenrick Kin

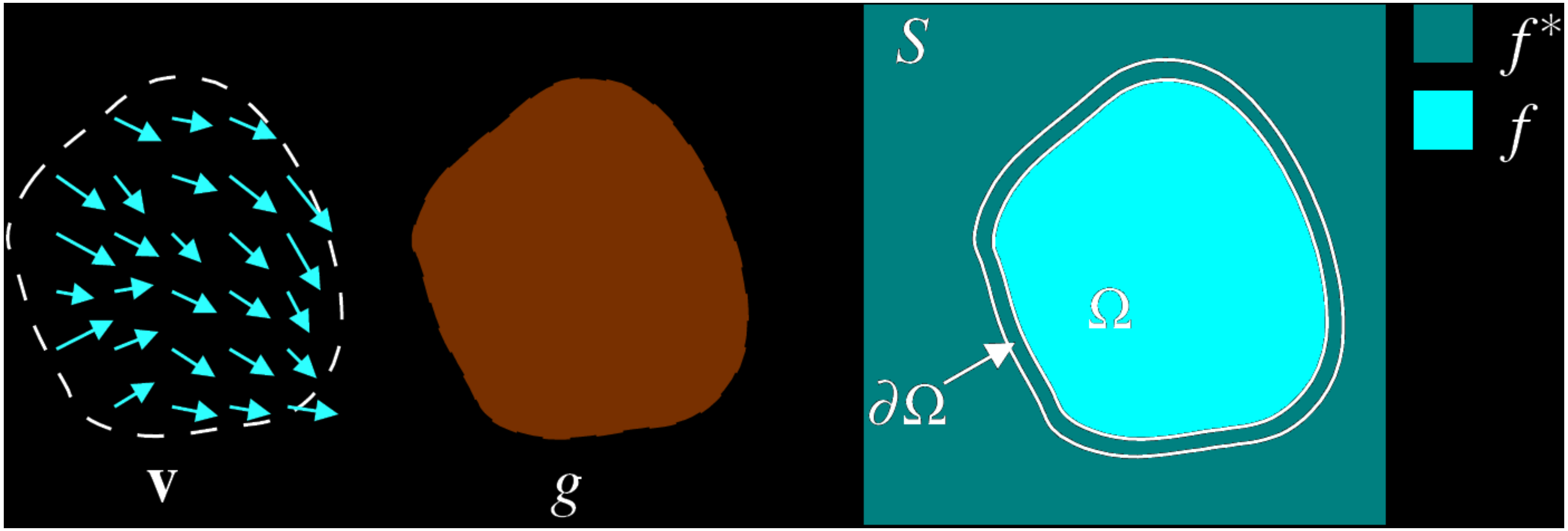
Poisson Image Blending

sources destinations cloning seamless cloning

Poisson Image Blending

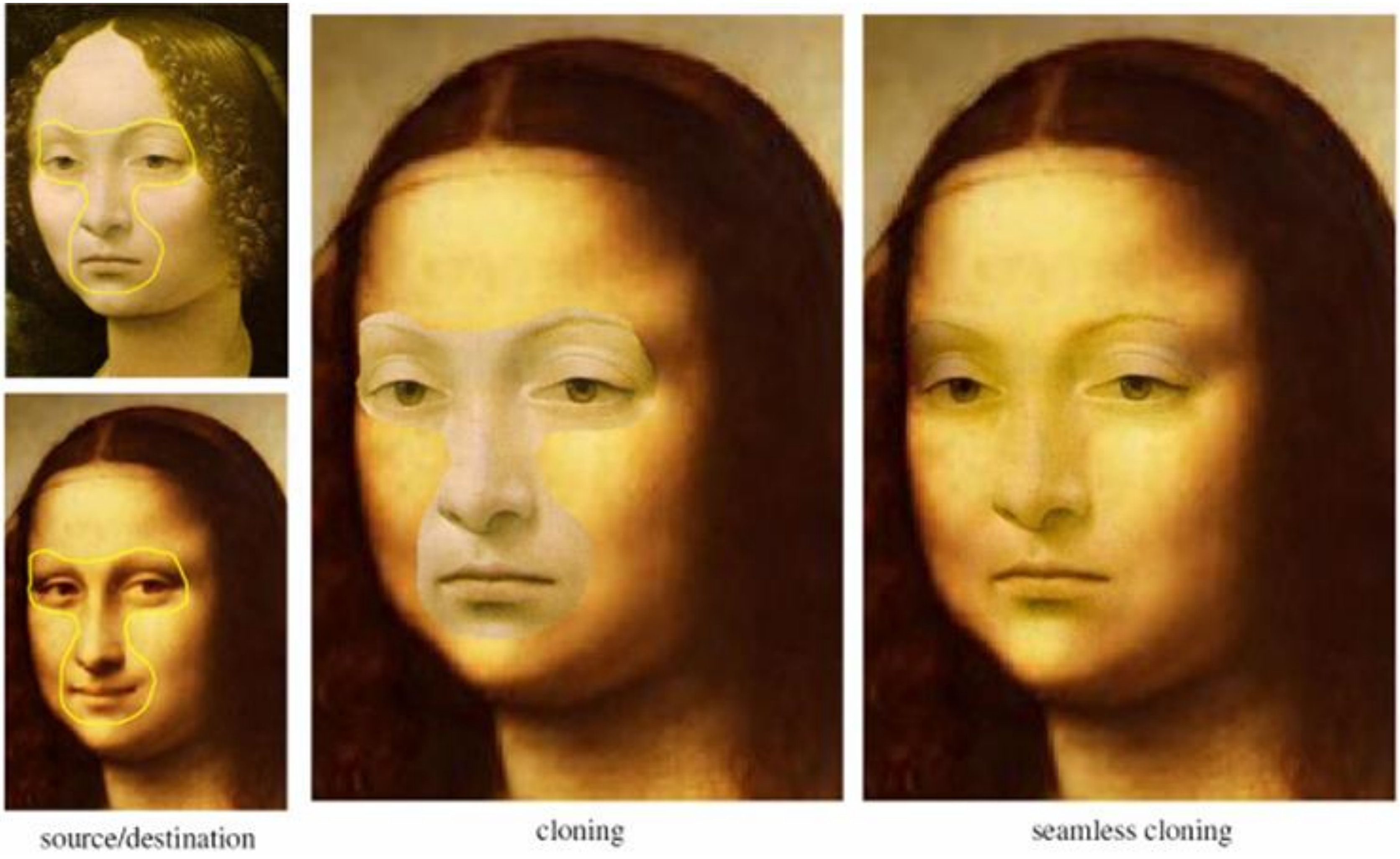
Beyond simple compositing

- Solve for image samples that follow gradients of source subject to boundary conditions imposed by dest



$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

Poisson Image Blending



Digital Image Processing

- Changing pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization
- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter
- Moving image locations
  - Scale
  - Rotate
  - Warp
- Combining images
  - Composite
  - Morph
- Quantization
- Spatial / intensity tradeoff
  - Dithering

Image Morphing

- Animate transition between two images

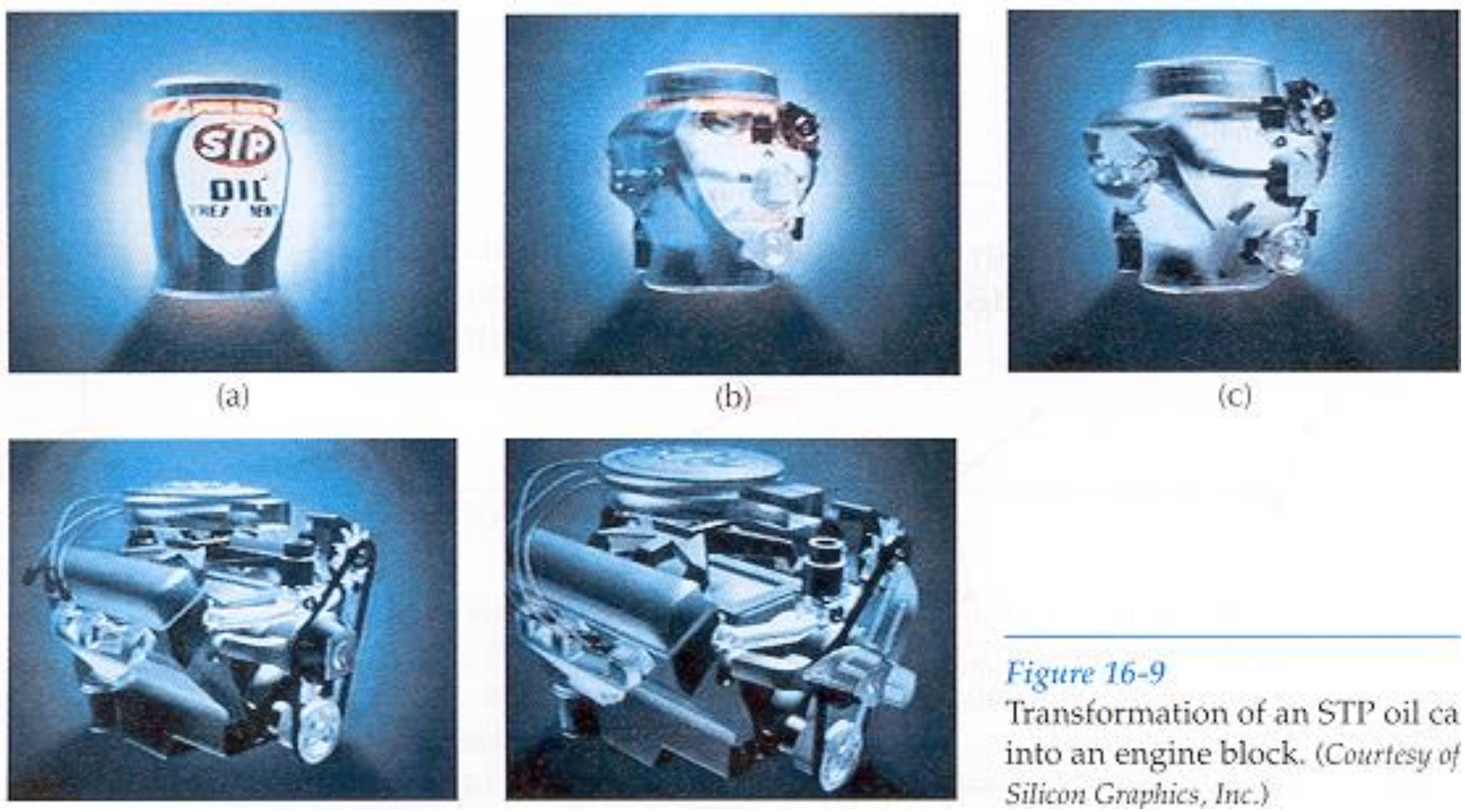


Figure 16-9
Transformation of an STP oil ca
into an engine block. (*Courtesy of
Silicon Graphics, Inc.*)

H&B Figure 16.9

Cross-Dissolving

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

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

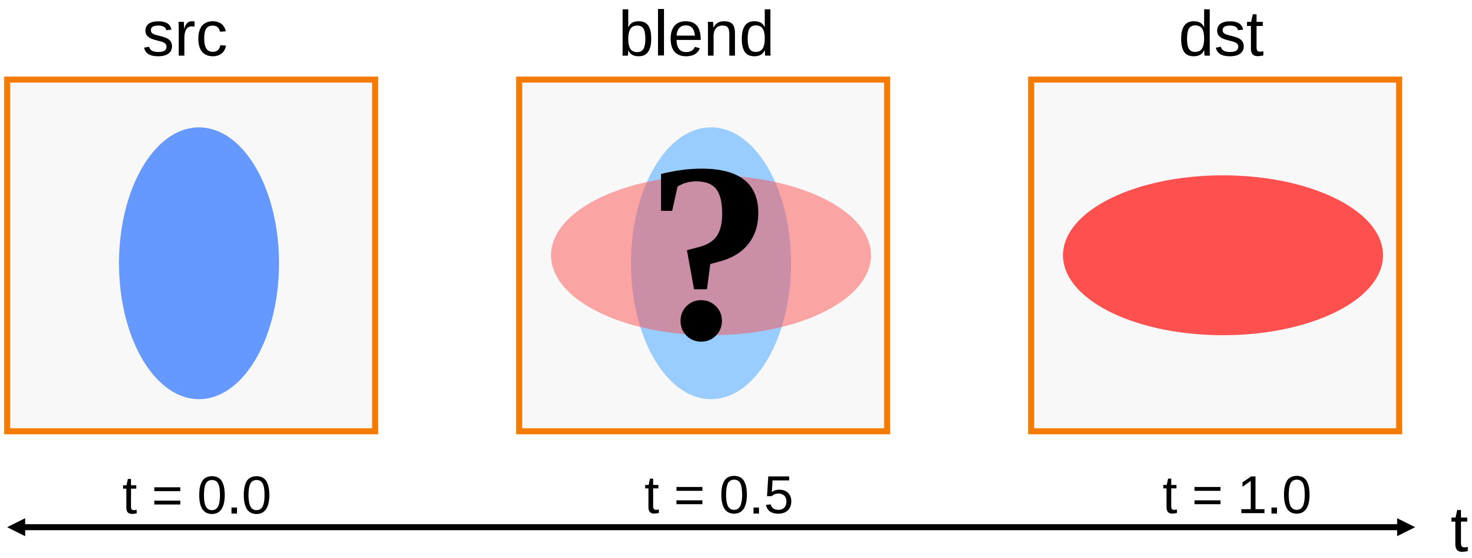
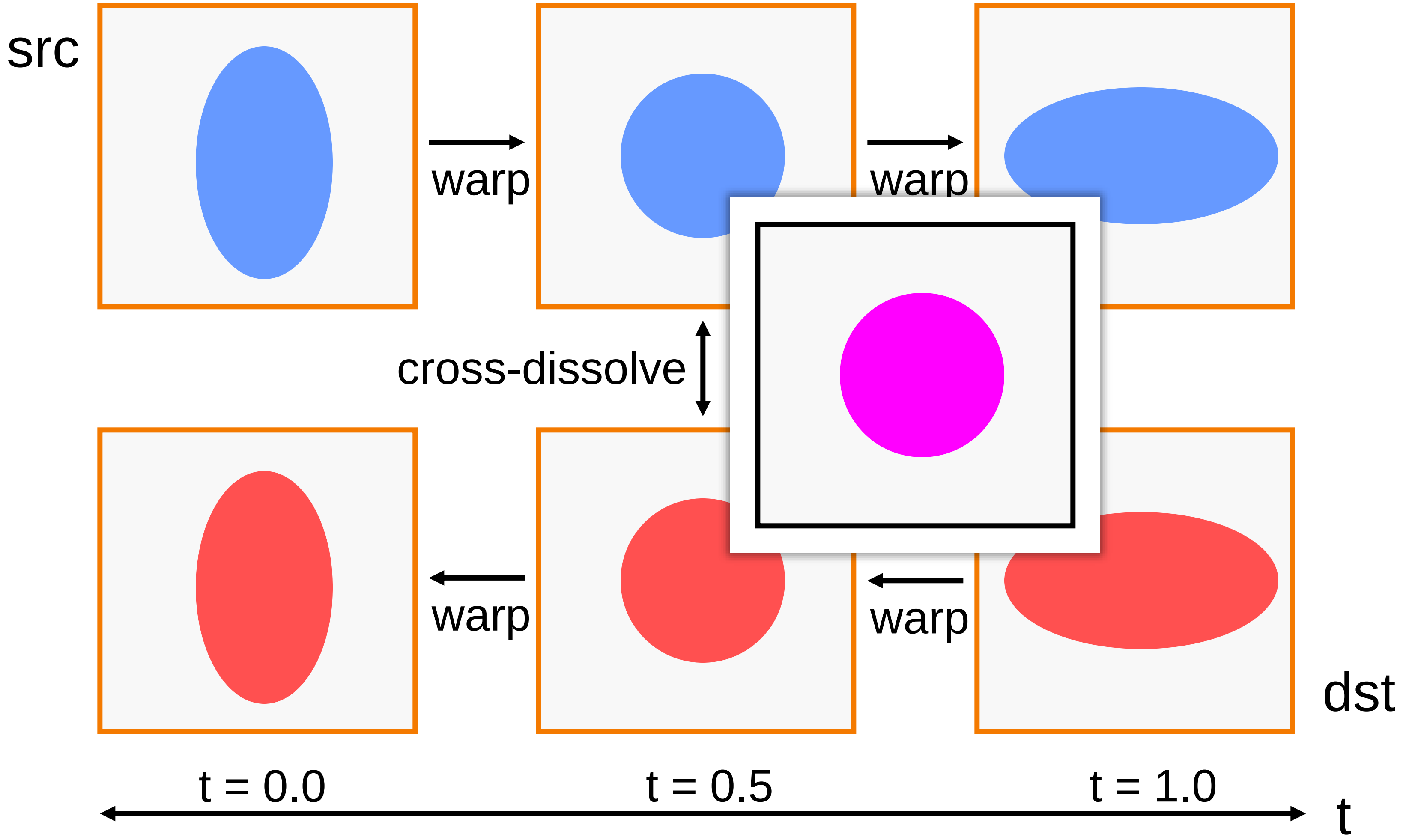


Image Morphing

- Combines warping and cross-dissolving



Beier & Neeley Example

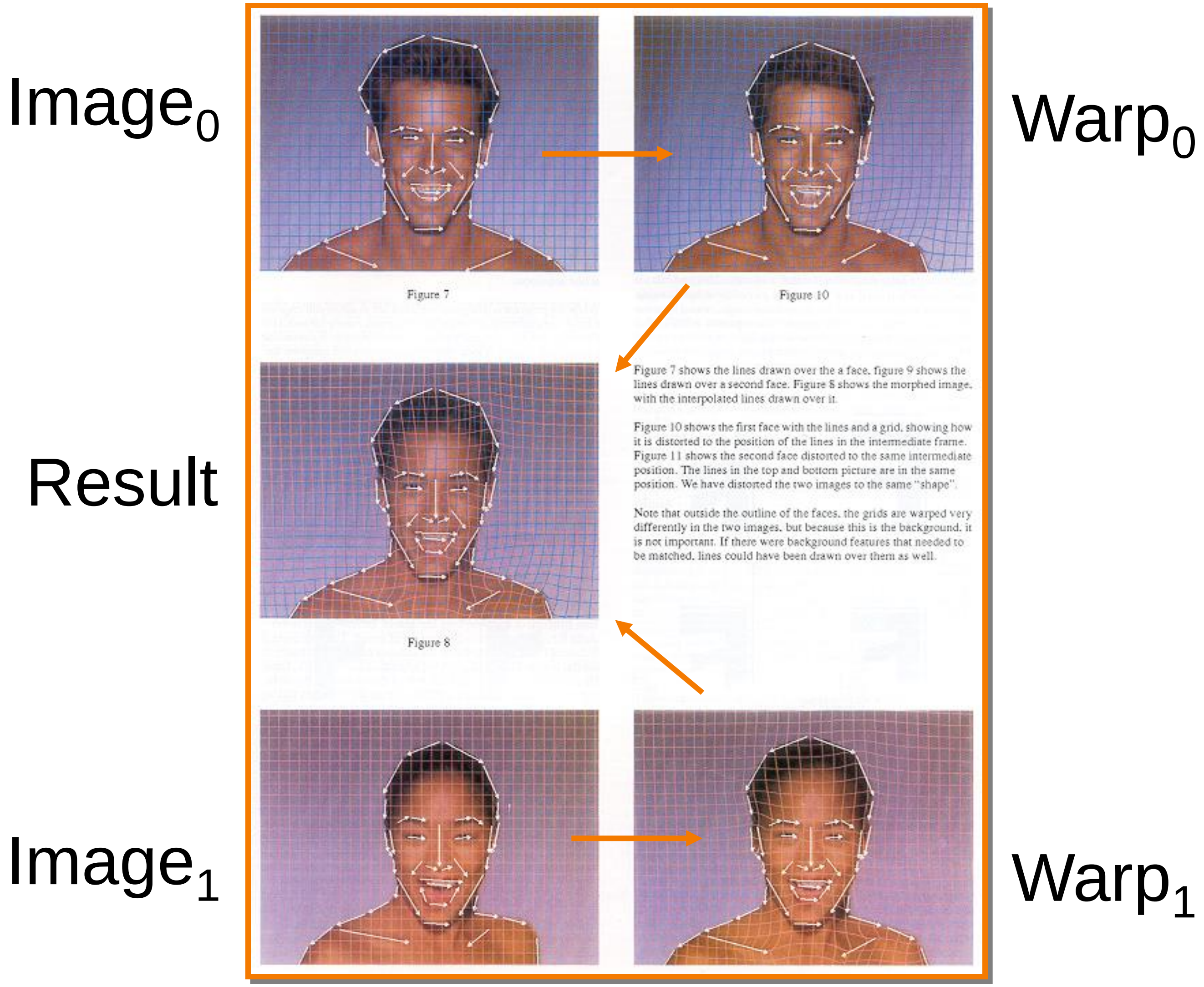


Figure 7 shows the lines drawn over the a face. figure 9 shows the lines drawn over a second face. Figure 8 shows the morphed image, with the interpolated lines drawn over it.

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.

Beier & Neeley Example

Image_0

Warp_0

Result

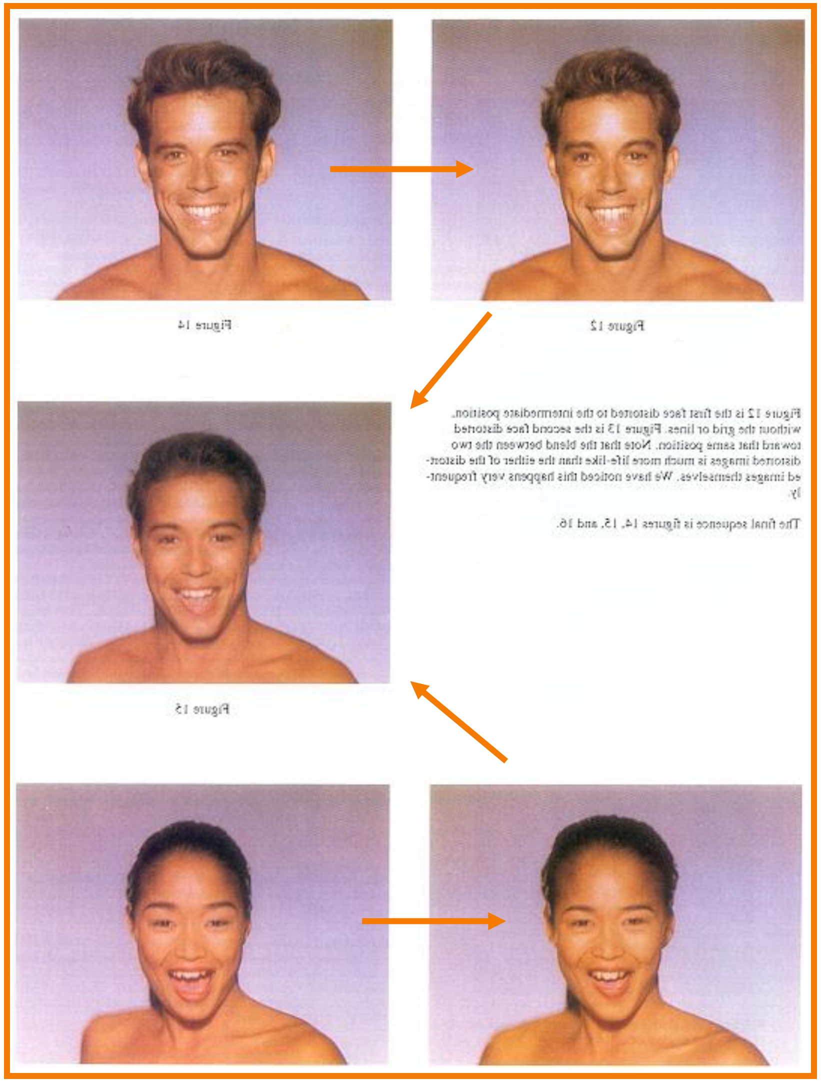


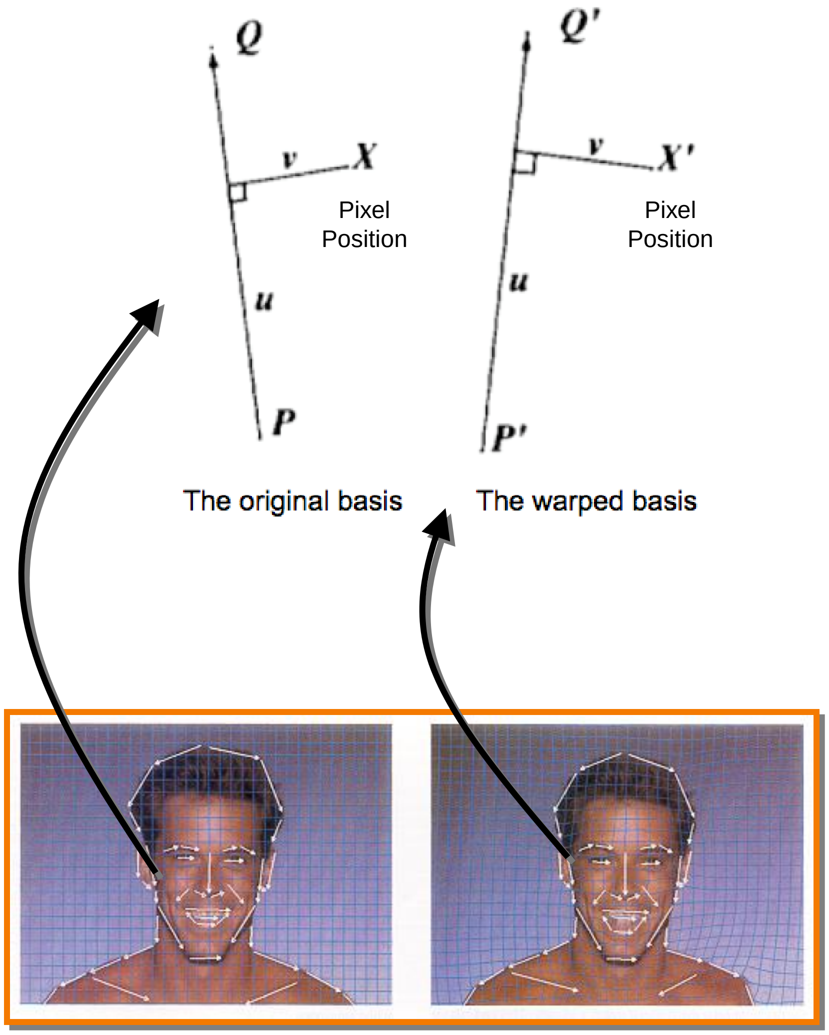
Image_1

Warp_1

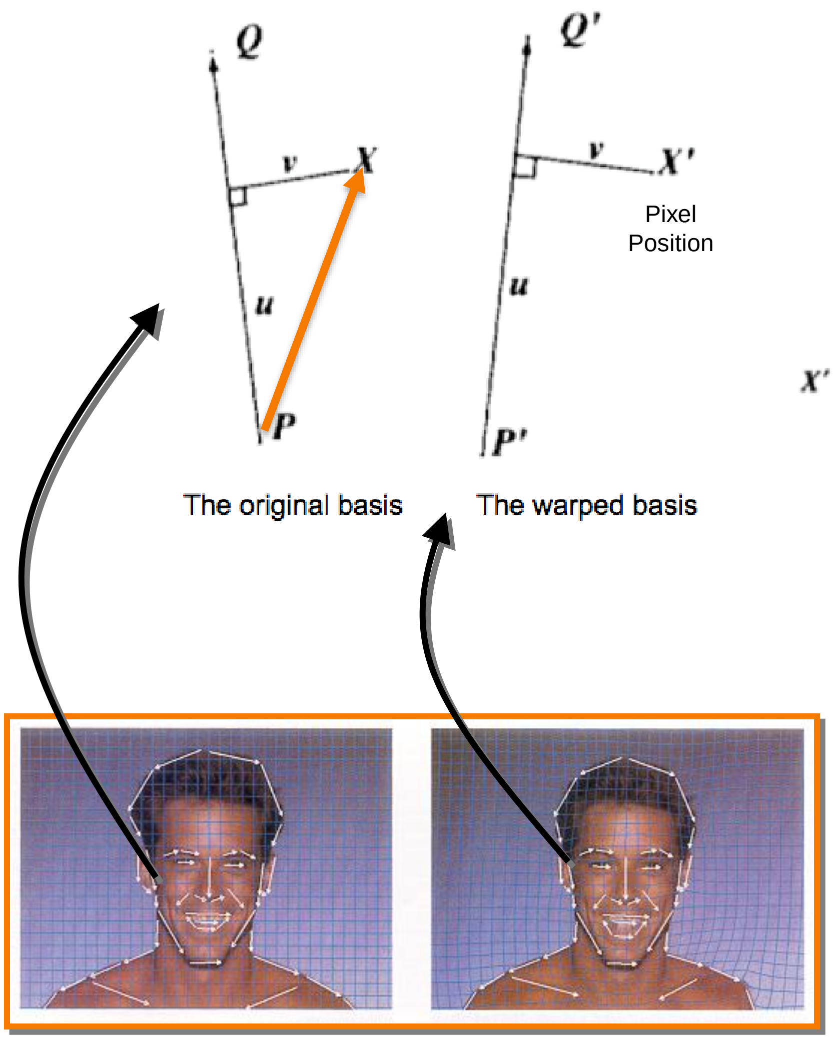
Beier & Neeley Example

Black or White, Michael Jackson (1991)

Warping Pixel Locations



Warping Pixel Locations

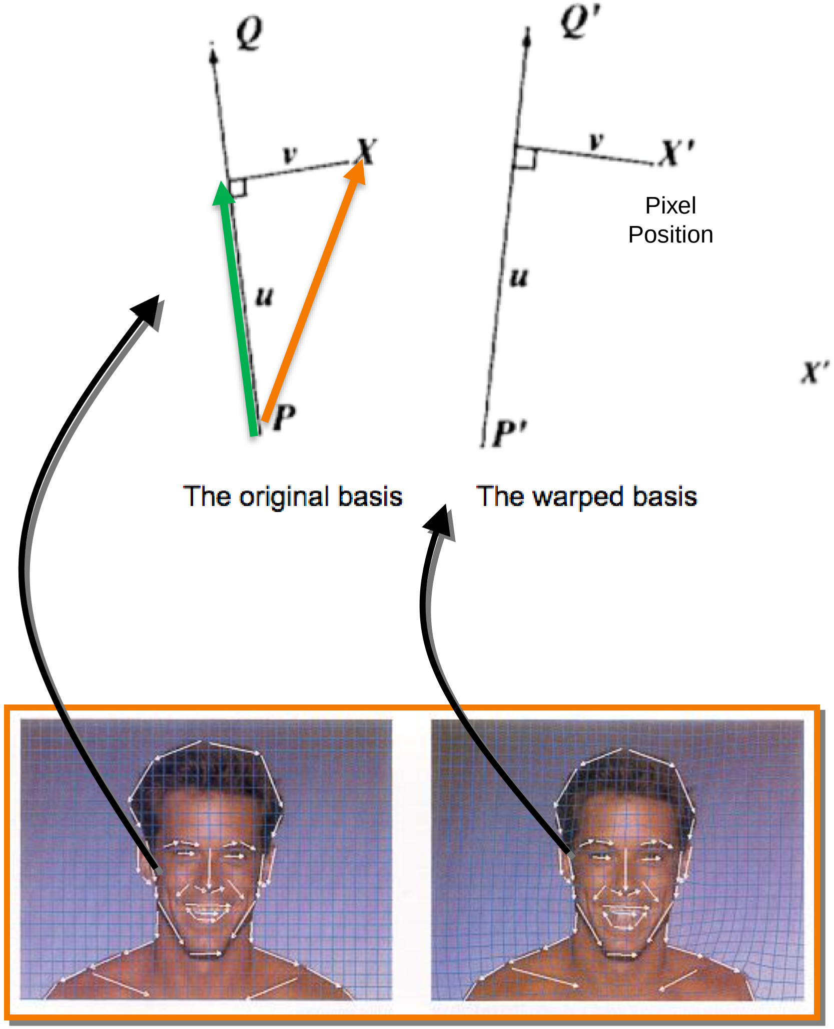


$$u = \frac{(X - P) \cdot (Q - P)}{\| Q - P \|^2}$$

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

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

Warping Pixel Locations

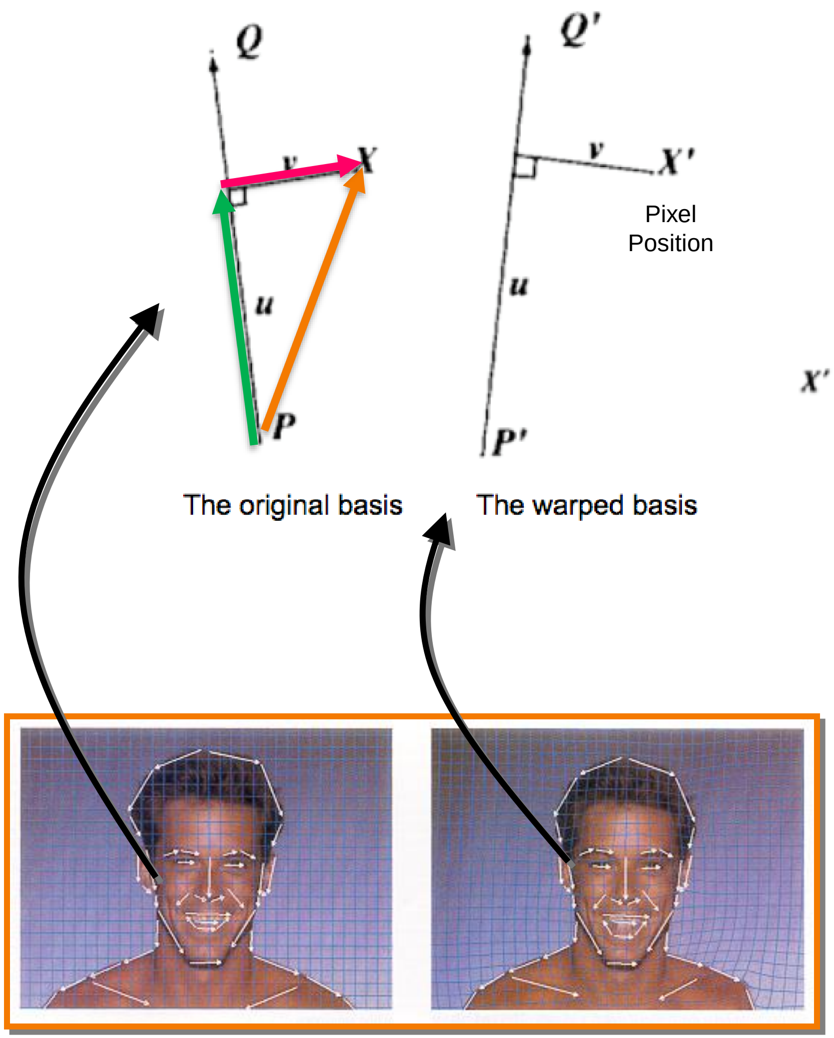


$$u = \frac{(X - P) \cdot (Q - P)}{\| Q - P \|^2}$$

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

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

Warping Pixel Locations

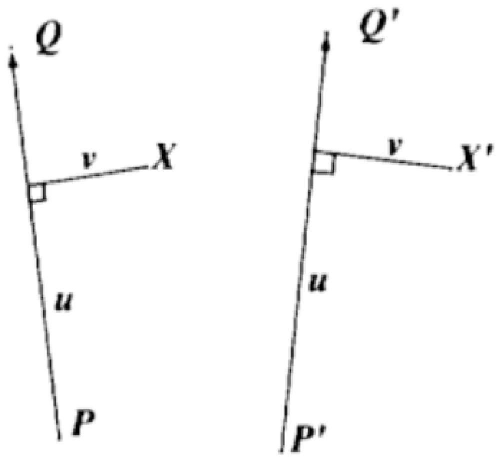


$$u = \frac{(X - P) \cdot (Q - P)}{\| Q - P \|^2}$$

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

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

Warping Pixel Locations



The original basis The warped basis

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

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

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

This generates one warp per line, each of which is a simple rotation and non-uniform scale (scaling is only done along the axis of the line). These warps must then be averaged to get the final warp. In the original paper, the weights for the average are tuned with the formula below. The *dist* variable is the distance of the point from the line segment, and the *length* variable is the length of the line segment.

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

The equations give several parameters to tune, and I got the best results when $a = 0.001$, $b = 2$, and $p = 0$. Ignoring the length of the line segments (by setting p to zero) gave better results than when the length was taken in to account. I used seven contours with 28 line segments to represent the features of each face.

Nice implementation notes from Evan Wallace, Brown University
http://cs.brown.edu/courses/csci1950-g/results/proj5/edwallac/

Warping Pseudocode

```
WarpImage(Image, L_src[…], L_dst[…])
begin
    foreach destination pixel p_dst do
        psum = (0,0)
        wsum = 0
        foreach line L_dst[i] do
            p_src[i] = p_dst transformed by (L_dst[i],L_src[i])
            psum = psum + p_src[i] * weight[i]
            wsum += weight[i]
        end
        p_src = psum / wsum
        Result(p_dst) = Resample(p_src)
    end
end
```

Morphing Pseudocode

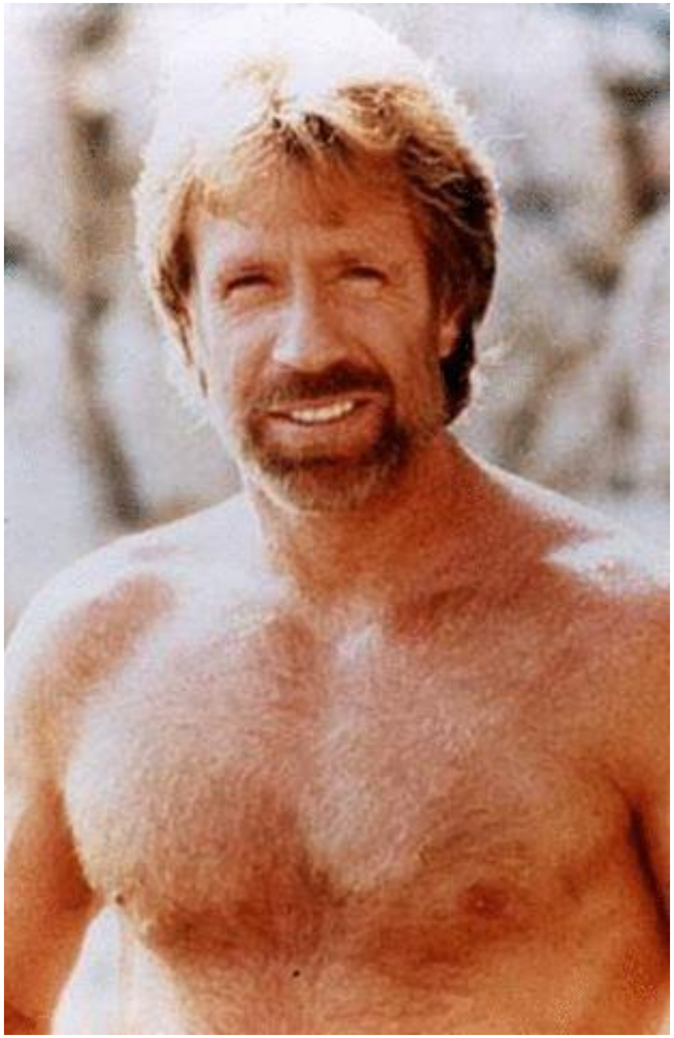
```
GenerateAnimation(Image_0, L_0[…], Image_1, L_1[…])
begin
    foreach intermediate frame time t do
        for i = 1 to number of line pairs do
            L[i] = line t^th of the way from L_0[i] to L_1[i]
        end
        Warp_0 = WarpImage(Image_0, L_0, L)
        Warp_1 = WarpImage(Image_1,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

COS426 Example

Amy Ousterhout

COS426 Examples

Sam Payne



Matt Matl

Image Composition Applications

- "Computational photography":
  new photographic effects that inherently use multiple images + computation
- Example: stitching images into a ***panorama***

[Michael Cohen]

Image Composition Applications

- Stoboscopic images

[Michael Cohen]

Image Composition Applications

- Extended depth-of-field

[Michael Cohen]

Scene Completion Using Millions of Photographs

James Hays and Alexei A. Efros

SIGGRAPH 2007

Slides by J. Hays and A. Efros

Hays et al. SIGGRAPH 07

Hays et al. SIGGRAPH 07

Hays et al. SIGGRAPH 07

Image Completion

?

Hays et al. SIGGRAPH 07

Image Completion

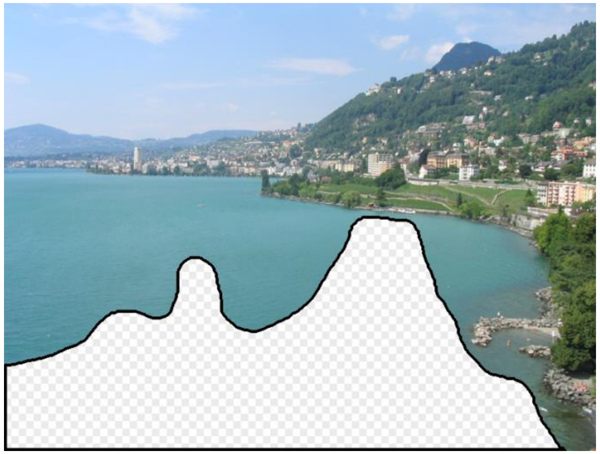
2.3 Million unique images from Flickr

Hays et al. SIGGRAPH 07

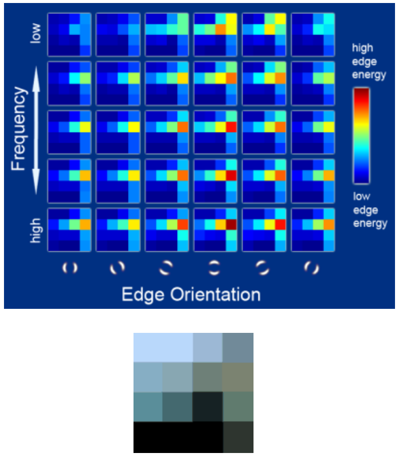
Scene Completion Result

Hays et al. SIGGRAPH 07

Image Completion Algorithm



Input image



Scene Descriptor

...

Image Collection

...

200 matches

Mosaicing

20 completions

Hays et al. SIGGRAPH 07

Hays et al. SIGGRAPH 07

Hays et al. SIGGRAPH 07

Hays et al. SIGGRAPH 07

Hays et al. SIGGRAPH 07

Summary

- Image compositing
  - Alpha channel
  - Porter-Duff compositing algebra
- Image morphing
  - Warping
  - Compositing
- Compositing in Computational Photography