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
  o Blue-screen mattes
  o Alpha channel
  o 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
  o Create background image
  o Create foreground image with blue background
  o 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 foreground pixels into blue background

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

• Single Pixel Example: $\alpha = 0.3$

Partial Coverage

Semi-Transparent

or
Compositing with Alpha

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

\[ 0 < \alpha < 1 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]
Pixels with Alpha

• Alpha channel convention:
  \((r, g, b, \alpha)\) represents a pixel that is \(\alpha\) covered by the color \(C = (r\alpha, g\alpha, b\alpha)\)
  » Color components are pre-multiplied by \(\alpha\)
  » Can display \((r,g,b)\) values directly

• What is the meaning of the following?
  \(\circ(0, 1, 0, 1) = ?\)
  \(\circ(0, 1/2, 0, 1) = ?\)
  \(\circ(0, 1/2, 0, 1/2) = ?\)
  \(\circ(0, 1/2, 0, 0) = ?\)
Pixels with Alpha

• Alpha channel convention:
  \( (r, g, b, \alpha) \) represents a pixel that is \( \alpha \) covered by the color \( C = (r^\alpha, g^\alpha, b^\alpha) \)
  » Color components are pre-multiplied by \( \alpha \)
  » Can display \( (r,g,b) \) values directly

• What is the meaning of the following?
  \( (0, 1, 0, 1) = \) Full green, full coverage
  \( (0, 1/2, 0, 1) = ? \)
  \( (0, 1/2, 0, 1/2) = ? \)
  \( (0, 1/2, 0, 0) = ? \)
Pixels with Alpha

• Alpha channel convention:
  \( (r, g, b, \alpha) \) represents a pixel that is \( \alpha \) covered by the color \( C = (r*\alpha, g*\alpha, b*\alpha) \)
  »Color components are pre-multiplied by \( \alpha \)
  »Can display \( (r,g,b) \) values directly

• What is the meaning of the following?
  \( (0, 1, 0, 1) = \) Full green, full coverage
  \( (0, 1/2, 0, 1) = \) Half green, full coverage
  \( (0, 1/2, 0, 1/2) = ? \)
  \( (0, 1/2, 0, 0) = ? \)
Pixels with Alpha

- Alpha channel convention:
  \((r, g, b, \alpha)\) represents a pixel that is \(\alpha\) covered by the color \(C = (r\alpha, g\alpha, b\alpha)\)
  
  » Color components are pre-multiplied by \(\alpha\)
  » Can display \((r, g, b)\) values directly

- What is the meaning of the following?
  \(\bullet (0, 1, 0, 1) = \text{Full green, full coverage}\)
  \(\bullet (0, 1/2, 0, 1) = \text{Half green, full coverage}\)
  \(\bullet (0, 1/2, 0, 1/2) = \text{Full green, half coverage}\)
  \(\bullet (0, 1/2, 0, 0) = ?\)
Pixels with Alpha

• Alpha channel convention:
  \((r, g, b, \alpha)\) represents a pixel that is \(\alpha\) covered by the color \(C = (r\alpha, g\alpha, b\alpha)\)
  » Color components are pre-multiplied by \(\alpha\)
  » Can display \((r, g, b)\) values directly

• What is the meaning of the following?
  \((0, 1, 0, 1)\) = Full green, full coverage
  \((0, 1/2, 0, 1)\) = Half green, full coverage
  \((0, 1/2, 0, 1/2)\) = Full green, half coverage
  \((0, 1/2, 0, 0)\) = Undefined
Semi-Transparent Objects

- 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?
  \[ (1-\alpha_A)(1-\alpha_B) \]
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

- clear
- A
- B
- A over B
- B over A
- A in B
- B in A
- A out B
- B out A
- A atop B
- B atop A
- A xor b

Porter & Duff '84
Example: C = A Over B

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

- For colors that are premultiplied:
  \[ C' = A' + (1-\alpha_A) B' \]
  \[ \alpha = \alpha_A + (1-\alpha_A) \alpha_B \]
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 can into an engine block. (Courtesy of Silicon Graphics, Inc.)
Image Morphing

- Animate transition between two images

Two Components:
- Cross-dissolving
- Warping

H&B Figure 16.9

Transformation of an STP oil can into an engine block. (Courtesy of Silicon Graphics, Inc.)
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

\[
\text{blend}(i,j) = (1-t) \text{src}(i,j) + t \text{dst}(i,j) \quad (0 \leq t \leq 1)
\]

\[
\begin{align*}
\text{src} & \quad \text{blend} & \quad \text{dst} \\
\text{t = 0.0} & \quad \text{t = 0.5} & \quad \text{t = 1.0}
\end{align*}
\]
Image Warping

Deform the source so that it looks like the target

src

t = 0.0

t = 0.5

t = 1.0

dst
Image Warping

Deform the source so that it looks like the target

\( \text{src} \)

\( \text{dst} \)

\( t = 0.0 \)

\( t = 0.5 \)

\( t = 1.0 \)
Image Morphing

Combines cross-dissolving and warping

src

warp

cross-dissolve

warp

dst

$\text{t} = 0.0$ $\text{t} = 0.5$ $\text{t} = 1.0$
Image Morphing

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

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?

![Mapping diagram]

Source image \quad \text{Mapping} \quad \text{Destination image}

\( u \) is a fraction
\( v \) is a length (in pixels)
Feature-Based Warping

How do I calculate $u$ and $v$?

1. Recall the dot product
2. $V_1 \cdot V_2 = x_1x_2 + y_1y_2$
3. $V_1 \cdot V_2 = \|V_1\| \|V_2\| \cos(\Theta)$
Feature-Based Warping

How do I calculate \( u \) and \( v \)?

\[
 u = \frac{(p - s) \cdot (t - s)}{|| t - s ||^2}
\]

Equation 1 from B&N paper

Why?

Remember: \( u \) is a fraction
Feature-Based Warping

How do I calculate u and v?

\[ v = \frac{(p - s) \cdot Perp(t - s)}{\| t - s \|} \]

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?

\[ u \text{ is a fraction} \\
\text{v is a length (in pixels)} \]
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

Source image

Destination image

Mapping

\( p' \) is a weighted average
Weighting Effect of Each Line Pair

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

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

Where:
• \( \text{length}[i] \) is the length of \( L[i] \)
• \( \text{dist}[i] \) is the distance from \( p \) to \( L[i] \)
• \( a, b, p \) are constants that control the warp
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

WarplImage(Image, L’[…], L[…])
begin
  for each 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
  end
  Result(p) = Image(p')
end
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

Image_0

Warp_0

Result

Image_1

Warp_1
Beier & Neeley Example

**Image**$_0$

**Warp**$_0$

**Result**

**Image**$_1$

**Warp**$_1$

Figure 12

Figure 14

Figure 15

Figure 13 is the second face distorted toward the same position. Note that the blend between the two distorted images is much more lifelike than either of the distorted images themselves. We have noticed this happens very frequently.

The final sequence is figures 14, 15, and 16.
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
  o A pixel is a sample, not a little square
  o Images have limited resolution
  o Image processing is a resampling problem

• Halftoning and dithering
  o Reduce visual artifacts due to quantization
  o Distribute errors among pixels
  o 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