Image Compositing and Morphing

Jason Lawrence

CS 4810: Graphics

Acknowledgement: slides by 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”
  o Render independently
  o Composite together

• Applications
  o Cel animation
  o Chroma-keying
  o 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 foreground pixels into 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](image1)

or

![Semi-Transparent](image2)
Compositing with Alpha

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

$\alpha = 1$

$0 < \alpha < 1$

$\alpha = 0$
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?
  - \((0, 1, 0, 1)\) = ?
  - \((0, 1/2, 0, 1)\) = ?
  - \((0, 1/2, 0, 1/2)\) = ?
  - \((0, 1/2, 0, 0)\) = ?
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
  - \((0, 1/2, 0, 1)\) = ?
  - \((0, 1/2, 0, 1/2)\) = ?
  - \((0, 1/2, 0, 0)\) = ?
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:
  o \((r, g, b, \alpha)\) 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?
  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

- 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
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: $C = A \text{ Over } B$

• 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
Image Morphing

- Animate transition between two images

Two Components:
- Cross-dissolving
- Warping

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

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

\[
\begin{array}{ccc}
\text{src} & \text{blend} & \text{dst} \\
\text{t = 0.0} & \text{t = 0.5} & \text{t = 1.0}
\end{array}
\]
Image Warping

Deform the source so that it looks like the target

$ src $ $ dst $

$t = 0.0$ $ t = 0.5 $ $ t = 1.0 $
Image Warping

Deform the source so that it looks like the target

\[ \text{src} \rightarrow \text{warp} \rightarrow \text{dst} \]

\[ t = 0.0 \quad \rightarrow \quad t = 0.5 \quad \rightarrow \quad t = 1.0 \]
Image Morphing

Combines cross-dissolving and warping

src

\( t = 0.0 \)

warp

cross-dissolve

\( t = 0.5 \)

dst

\( t = 1.0 \)
Image Morphing

• The warping step is the hard one
  o Aim to align features in images
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)
Feature-Based Warping

How do I calculate u and v?

1. Recall the dot product
2. $V_1 \cdot V_2 = dx_1 dx_2 + dy_1 dy_2$
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 $v$?

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

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
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:

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

Where:
• \text{length}[i] is the length of L[i]
• \text{dist}[i] is the distance from X to L[i]
• \text{a, b, p} are constants that control the warp
How do I calculate dist? Dist is either…

• abs(v) if u is $\geq 0$ and $\leq 1$

OR

• distance to the closest endpoint i.e.

$$\min (\|p - s\|, \|p - t\|)$$
Warping Pseudocode

\[\text{WarpImage(Image, L'[...], L[...])}\]
begin
    foreach 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
        \( p' = psum / wsum \)
        Result(p) = Image(p')
    end
end
Warping Pseudocode

\[ \text{WarpImage(Image, L'[...], L[...])} \]

begin
  \text{foreach destination pixel } p \text{ do}
    \text{psum} = (0,0)
    \text{wsum} = 0
    \text{foreach line } L[i] \text{ in destination do}
      \text{p'}[i] = \text{p transformed by } (L[i], L'[i])
      \text{psum} = \text{psum} + \text{p'}[i] \times \text{weight}[i]
      \text{wsum} += \text{weight}[i]
    end
    \text{p'} = \text{psum} / \text{wsum}
  end
  \text{Result(p)} = \text{Image(p')}
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
  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
Beier & Neeley Example

**Image** \( I_0 \)  

**Warp** \( W_0 \)  

**Result**  

**Image** \( I_1 \)  

**Warp** \( W_1 \)  

---

**Figure 7** shows the lines drawn over the face. Figure 8 shows the lines drawn over a second face. Figure 9 shows the warped image with the interpolated lines drawn over it.

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

Note that outside the outline of the face, 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

$\text{Image}_0$  $\text{Warp}_0$

Result

$\text{Image}_1$  $\text{Warp}_1$

Figure 12:
The first face distorted to the intermediate position, without the grid or lines. 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.
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