CSCI 1290: Comp Photo

Fall 2018 @ Brown University
James Tompkin

Many slides thanks to James Hays’ old CS 129 course, along with all of its acknowledgements.
Alan hours today

• 16:00 – 18:00
• CIT 203
• HDR project code support
Image Compositing  (Szeliski 9.3.4)

Google Street View

Many slides from Alexei Efros
Compositing Procedure

1. Define mask regions (e.g., using *Intelligent Scissors* in Photoshop)
Compositing Procedure

1. Define mask regions (e.g., using *Intelligent Scissors* in Photoshop)

2. Layer masked image regions
Artifacts in simple masking

- Staircasing
- Fringing or subtle halo

Goes back to one of our motivating questions: *What is a pixel?*

Masks are not enough.
Many composites need blending
Alpha channel

Add one more channel:
  • Image(R,G,B,alpha)

Encodes transparency (or pixel coverage):
  • Alpha = 1: opaque object (complete coverage)
  • Alpha = 0: transparent object (no coverage)
  • 0<Alpha<1: semi-transparent (partial coverage)

Example: alpha = 0.7
Multiple Alpha Blending

So far we assumed that one image (background) is opaque. If blending semi-transparent sprites (the “A over B” operation):

\[ I_{\text{comp}} = \alpha_a I_a + (1-\alpha_a)\alpha_b I_b \]
\[ \alpha_{\text{comp}} = \alpha_a + (1-\alpha_a)\alpha_b \]

Note: sometimes alpha is premultiplied:

\[ \text{im}(\alpha R, \alpha G, \alpha B, \alpha) : \]
\[ I_{\text{comp}} = I_a + (1-\alpha_a) I_b \]
(same for alpha!)
Alpha Blending / Feathering

\[ I_{\text{blend}} = \alpha I_{\text{left}} + (1-\alpha) I_{\text{right}} \]
Setting alpha: simple averaging

Alpha = .5 in overlap region
Setting alpha: center seam

Distance Transform

Alpha = logical(dtrans1>dtrans2)
Setting alpha: blurred seam

Distance transform

Alpha = blurred
Setting alpha: center weighting

\[ \text{Alpha} = \frac{d\text{trans}1}{(d\text{trans}1+d\text{trans}2)} \]
Affect of Window Size
Affect of Window Size
Good Window Size

“Optimal” Window: smooth but not ghosted
Laplacian (derivative of Gaussian) Pyramid
Pyramid Blending

Left pyramid

blend

Right pyramid
Pyramid Blending
Laplacian level 4

Laplacian level 2

Laplacian level 0

Left pyramid  Right pyramid  Blended pyramid
Laplacian Pyramid: Blending

General Approach:

1. Build Laplacian pyramids $LA$ and $LB$ from images $A$ and $B$
2. Build a Gaussian pyramid $GR$ from selected region $R$
3. Form a combined pyramid $LS$ from $LA$ and $LB$ using nodes of $GR$ as weights:
   • $LS(i,j) = GR(i,j) \times LA(i,j) + (1 - GR(i,j)) \times LB(i,j)$
4. Collapse the $LS$ pyramid to get the final blended image
Blending Regions
Horror Photo

david dmartin (Boston College)
Simplification: Two-band Blending

Brown & Lowe, 2003

- Only use two bands: high freq. and low freq.
- Blends low freq. smoothly
- Blend high freq. with no smoothing: use binary alpha
Gradient Domain Image Blending

In Pyramid Blending, we decomposed our image into $2^{nd}$ derivatives (Laplacian) and a low-res image.

Let’s look at a more direct formulation:

- No need for low-res image
  - captures everything (up to a constant)
- Idea:
  - Differentiate
  - Composite
  - Reintegrate
Gradient Domain blending (1D)

Two signals

Regular blending

Blending derivatives

bright

dark
Trickier in 2D:

- Take partial derivatives dx and dy (the gradient field)
- Manipulate them (smooth, blend, feather, etc. as needed)
- Reintegrate
  - But now integral(dx) might not equal integral(dy)

- Find the most agreeable solution
  - Equivalent to solving Poisson equation (partial differential equation)
  - Can use FFT, deconvolution, multigrid solvers, etc.
Perez et al., 2003
Limitations:

- Can’t negate gradients, e.g., contrast reversal (gray on black -> gray on white)
- Colored backgrounds “bleed through”
- Images need to be very well aligned
Figure 7: **Inserting transparent objects.** Mixed seamless cloning facilitates the transfer of partly transparent objects, such as the rainbow in this example. The non-linear mixing of gradient fields picks out whichever of source or destination structure is the more salient at each location.
Gradient Domain Blending

Project details not shown in class
It is impossible to faithfully preserve the gradients.
Simple 2d example

What properties do we want x to have?
Simple 2d example

For unmasked pixels, $x_i = t_i$

For masked pixels, we want the gradients at $x_i$ to match the gradients at $s_i$

But how do we define the gradient? Instead of constraining one or many gradients, in this example we will use the Laplacian.
Simple 2d example

Pixel indexing

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>7</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

target, \( t \)

<table>
<thead>
<tr>
<th>.2</th>
<th>.5</th>
<th>.2</th>
<th>.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>.7</td>
<td>.7</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>.9</td>
<td>.9</td>
<td>.8</td>
<td>.9</td>
</tr>
</tbody>
</table>

source, \( s \)

<table>
<thead>
<tr>
<th>.8</th>
<th>.6</th>
<th>.6</th>
<th>.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>.6</td>
<td>.6</td>
<td>.2</td>
<td>.6</td>
</tr>
<tr>
<td>.6</td>
<td>.8</td>
<td>.6</td>
<td>.6</td>
</tr>
</tbody>
</table>

mask

<table>
<thead>
<tr>
<th>0</th>
<th>-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

Laplacian

\[ x_1 = t_1 \\
\[ x_2 = t_2 \\
\[ x_3 = t_3 \\
\[ x_4 = t_4 \\
\[ 4x_5 - x_4 - x_2 - x_6 - x_8 = 4s_5 - s_4 - s_2 - s_6 - s_8 \\
\[ x_6 = t_6 \\
\[ x_7 = t_7 \\
\[ 4x_8 - x_7 - x_5 - x_9 - x_{11} = 4s_8 - s_7 - s_5 - s_9 - s_{11} \\
\[ x_9 = t_9 \\
\[ x_{10} = t_{10} \\
\[ x_{11} = t_{11} \\
\[ x_{12} = t_{12} \]
### Simple 2d example

**Target, t**

<table>
<thead>
<tr>
<th></th>
<th>0.2</th>
<th>0.5</th>
<th>0.2</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>2</td>
<td>0.9</td>
<td>0.9</td>
<td>0.8</td>
<td>0.9</td>
</tr>
</tbody>
</table>

**Source, s**

<table>
<thead>
<tr>
<th></th>
<th>0.8</th>
<th>0.6</th>
<th>0.6</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6</td>
<td>0.6</td>
<td>0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>0.8</td>
<td>0.6</td>
<td>0.6</td>
</tr>
</tbody>
</table>

**Mask**

* 0
* -1
* 0

**Laplacian**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>4</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Output, x**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Pixel indexing**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>7</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

\[
x_1 = 0.2 \\
x_2 = 0.7 \\
x_3 = 0.9 \\
x_4 = 0.5 \\
4x_5 - x_4 - x_2 - x_6 - x_8 = 0.2 \\
x_6 = 0.9 \\
x_7 = 0.2 \\
4x_8 - x_7 - x_5 - x_9 - x_{11} = -1.6 \\
x_9 = 0.8 \\
x_{10} = 0.2 \\
x_{11} = 0.7 \\
x_{12} = 0.9
\]
Simple 2d example

Pixel indexing

\[
\begin{bmatrix}
1 & 4 & 7 & 10 \\
2 & 5 & 8 & 11 \\
3 & 6 & 9 & 12 \\
\end{bmatrix}
\]

source, s

\[
\begin{bmatrix}
.2 & .5 & .2 & .2 \\
.7 & .7 & .7 & .7 \\
.9 & .9 & .8 & .9 \\
\end{bmatrix}
\]
target, t

mask

\[
\begin{bmatrix}
1 & 1 \\
-1 & -1 & 4 & -1 & -1 \\
1 & -1 & 4 & -1 & -1 \\
1 & -1 & 4 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

Laplacian

\[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 4 & -1 \\
0 & -1 & 0 \\
\end{bmatrix}
\]

output, x

\[
\begin{bmatrix}
\end{bmatrix}
\]

\[
\begin{bmatrix}
.2 \\
.7 \\
.9 \\
.5 \\
.2 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
.2 \\
.7 \\
.9 \\
.5 \\
.2 \\
-1.6 \\
.8 \\
.2 \\
.7 \\
.9 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
? \\
? \\
? \\
? \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
? \\
? \\
? \\
? \\
\end{bmatrix}
\]

A

x = A \backslash b

x

b

= *
Simple 2d example

Source, s:

```
.2  .5  .2  .2
.7  .7  .7  .7
.9  .9  .8  .9
```

target, t:

```
.8  .6  .6  .6
.6  .6  .2  .6
.6  .8  .6  .6
```

mask:

```
1  1
1  1
1  1
1  1
```

Output, x:

```
.2  .5  .2  .2
.7  .62 .18 .7
.9  .9  .8  .9
```

Pixel indexing:

```
1  4  7  10
2  5  8  11
3  6  9  12
```

Output = reshape(x, 3, 4)

Laplacian:

```
0  -1  0
-1  4  -1
0  -1  0
```

Matrix A:

```
1  1
1  1
1  1
1  1
```

```
.2  .7  .9
.9  .9  .9
.5  .5  .5
.2  .2  .2
```

```
.62 .2  .2  .2
.9  .9  .9  .9
.2  .2  .2  .2
```

```
.18 .18 .18 .18
.8  .8  .8  .8
.2  .2  .2  .2
```

```
.9  .9  .9  .9
```

```
.2  .7  .9
.9  .9  .9
.5  .5  .5
.2  .2  .2
```

```
.62 .2  .2  .2
.9  .9  .9  .9
.2  .2  .2  .2
```

```
.18 .18 .18 .18
.8  .8  .8  .8
.2  .2  .2  .2
```

```
.9  .9  .9  .9
```

A * x = b
What’s the difference?

Gradient domain blending   -   No blending
What’s the difference?

- gradient domain blending
- no blending
What’s the difference?

gradient domain blending - no blending =
How does Superman fly?

Super-human powers?
OR
Image Blending
OR
Image Matting?
Physics of Alpha Matting

Semi-transparent objects

Pixels too large to capture surface
Problem Definition:
- The separation of an image $C$ into
  - A foreground object image $C_o$,
  - a background image $C_b$,
  - and an alpha matte $\alpha$.
- $C_o$ and $\alpha$ can then be used to composite the foreground object into a different image.

Hard problem
- Even if alpha is binary, this is hard to do automatically (background subtraction problem).
- For movies/TV, manual segmentation of each frame is difficult.
- Need to make a simplifying assumption…
Average/Median Image

What can we do with this?
Background Subtraction
Background Subtraction

A largely unsolved problem…

One video frame
Estimated background
Difference Image
Thresholded Foreground on blue
Blue Screen
Blue Screen matting

Most common form of matting in TV studios & movies

Petros Vlahos invented blue screen matting in the 50s. His Ultimatte® is still the most popular equipment. He won an Oscar for lifetime achievement.

A form of background subtraction:

- Need a known background
- Compute alpha as SSD(C,Cb) > threshold
  - Or use Vlahos’ formula: $\alpha = 1 - p_1(B - p_2 G)$
- Hope that foreground object doesn’t look like background
  - no blue ties!
- Why blue?
- Why uniform?
The Ultimatte

![Diagram of the Ultimatte system](image)

- **p₁ and p₂**

![Images of Ultimatte in action](images)
What we really want is to obtain a true alpha matte, which involves semi-transparency:

- Alpha between 0 and 1
Matting Problem: Mathematical Definition

For every pixel in the composite image, given
- backing color $C_k = [R_k \ G_k \ B_k]$, and
- composite pixel color $C = [R \ G \ B]$ compute
- foreground pixel color $C_o = [R_o \ G_o \ B_o \ \alpha_o]$ (=$[\alpha_o R_o \ \alpha_o G_o \ \alpha_o B_o]$)
such that

The matting equation

$$C = C_o + (1 - \alpha_o) C_k$$
Why is general matting hard?

Matting Equation:

$$C = C_o + (1 - \alpha_o) C_k$$
Solution #1: No Blue!

Matting Equation:

\[ C = C_o + (1 - \alpha_o) C_k \]

- If we know that the foreground contains no blue, we have \( B_o = 0 \)
- This leaves us with 3 equations and 3 unknowns, which has exactly one solution

\[ R = \alpha_o R_o + (1 - \alpha_o) R_k \quad \leftarrow 3. \text{Solve for } R_o \]
\[ G = \alpha_o G_o + (1 - \alpha_o) G_k \quad \leftarrow 2. \text{Solve for } G_o \]
\[ B = B_k - \alpha_o B_k \quad \leftarrow 1. \text{Solve for } \alpha_o \]

Main difficulty:
Solution #2: Gray or Flesh

Matting Equation:

\[ C = C_o + (1 - \alpha_o) C_k \]

- If we know that the foreground contains gray, that means that \( R_o = B_o = G_o \)
- This leaves us with 3 equations and 2 unknowns
Matting Equation:

\[ C = C_0 + (1 - \alpha_0) C_k \]

- Instead of reducing the number of unknowns, we could attempt to increase the number of equations.
- One way to do this is to photograph an object of interest in front of two known but distinct backgrounds.

How many equations?
How many unknowns?
Does the background need to be constant color?
The Algorithm

For every pixel $p$ in the composite image,
given
- backing color $C_{k1} = [R_{k1}, G_{k1}, B_{k1}]$ at $p$,
- backing color $C_{k2} = [R_{k2}, G_{k2}, B_{k2}]$ at $p$,
- composite pixel color $C_1 = [R_1, G_1, B_1]$ at $p$, and
- composite pixel color $C_2 = [R_2, G_2, B_2]$ at $p$,

solve the system of 6 equations

\[
\begin{align*}
R_1 &= \alpha_0 R_o + (1 - \alpha_0) R_{k1} \\
G_1 &= \alpha_0 G_o + (1 - \alpha_0) G_{k1} \\
B_1 &= \alpha_0 B_o + (1 - \alpha_0) B_{k1} \\
R_2 &= \alpha_0 R_o + (1 - \alpha_0) R_{k2} \\
G_2 &= \alpha_0 G_o + (1 - \alpha_0) G_{k2} \\
B_2 &= \alpha_0 B_o + (1 - \alpha_0) B_{k2}
\end{align*}
\]

for unknowns $R_o, G_o, B_o, \alpha_o$
Triangulation Matting Examples

From Smith & Blinn’s SIGGRAPH’96 paper
More Examples
More examples
Problems with Matting

Images do not look realistic
Lack of Refracted Light
Lack of Reflected Light

Solution:
Modify the Matting Equation
Environment Matting and Compositing

slides by Jay Hetler

Douglas E. Zongker ~ Dawn M. Werner ~ Brian Curless ~ David H. Salsin
SIGGRAPH 99
Environment Matting Equation

\[ C = F + (1 - \alpha)B + \Phi \]

- \( C \sim \) Color
- \( F \sim \) Foreground color
- \( B \sim \) Background color
- \( \alpha \sim \) Amount of light that passes through the foreground
- \( \Phi \sim \) Contribution of light from Environment that travels through the object
Explanation of $\Phi$

\[ \Phi = \sum_{i=1}^{m} \int R_i(x) T_i(x) \, dx \]

R – reflectance image
T – Texture image
Environment Mattes
How much better is Environment Matting?
How much better is Environment Matting?

Alpha Matte  Environment Matte  Photograph