Image Blending and Compositing
Compositing Procedure

1. Extract Sprites (e.g. using *Intelligent Scissors* in Photoshop)

2. Blend them into the composite (in the right order)
Alpha Blending / Feathering

\[ I_{\text{blend}} = \alpha I_{\text{left}} + (1-\alpha)I_{\text{right}} \]
Pyramid Blending
Gradient Domain vs. Frequency Domain

In Pyramid Blending, we decomposed our images into several frequency bands, and transferred them separately

• But boundaries appear across multiple bands

But what about representation based on derivatives (gradients) of the image?:

• Represents local change (across all frequencies)
• No need for low-res image
  – captures everything (up to a constant)
• Blending/Editing in Gradient Domain:
  – Differentiate
  – Copy / Blend / edit / whatever
  – Reintegrate
Gradients vs. Pixels

Craik-O’Brien Cornsweet Effect

Actual Luminance Profile  Perceived Luminance Profile
Gilchrist Illusion
(c.f. Exploratorium)
White?
Drawing in Gradient Domain

Real-Time Gradient-Domain Painting

James McCann
Carnegie Mellon University

Nancy S. Pollard
Carnegie Mellon University

James McCann & Nancy Pollard
Real-Time Gradient-Domain Painting,
SIGGRAPH 2009
(paper came out of this class!)

http://www.youtube.com/watch?v=RvhkAfrA0-w&feature=youtu.be
Gradient Domain blending (1D)

Two signals

Regular blending

Blending derivatives

bright
dark
Gradient Domain Blending (2D)

Trickier in 2D:

• Take partial derivatives $dx$ and $dy$ (the gradient field)
• Fiddle around with them (copy, blend, smooth, feather, etc)
• Reintegrate
  – But now $\int dx$ might not equal $\int dy$
• Find the most agreeable solution
  – Equivalent to solving Poisson equation
  – Can be done using least-squares ($\backslash$ in Matlab)
Gradient hole-filling (1D)

target

source
It is impossible to faithfully preserve the gradients.
Example

Gradient Visualization

Source: Evan Wallace
Poisson Blending Algorithm

A good blend should preserve gradients of source region without changing the background

Treat pixels as variables to be solved

- Minimize squared difference between gradients of foreground region and gradients of target region
- Keep background pixels constant

\[
v = \arg\min_v \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - (s_i - s_j))^2 + \sum_{i \in S, j \in N_i \cap -S} ((v_i - t_j) - (s_i - s_j))^2
\]

Perez et al. 2003
Examples

Gradient domain processing

\[ v = \arg\min_v \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - (s_i - s_j))^2 + \sum_{i \in S, j \in N_i \cap S} ((v_i - t_j) - (s_i - s_j))^2 \]

<table>
<thead>
<tr>
<th>source image</th>
<th>background image</th>
<th>target image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 20 5 20 9 20 13 20</td>
<td>1 10 5 10 9 10 13 10</td>
<td>1 10 5 10 9 10 13 10</td>
</tr>
<tr>
<td>2 20 6 80 10 20 14 20</td>
<td>2 10 6 10 10 10 14 10</td>
<td>2 10 6 (v_1) 10 14 10</td>
</tr>
<tr>
<td>3 20 7 20 11 80 15 20</td>
<td>3 10 7 10 10 10 15 10</td>
<td>3 10 (v_2) 10 15 10</td>
</tr>
<tr>
<td>4 20 8 20 12 20 16 20</td>
<td>4 10 8 10 10 16 10</td>
<td>4 10 8 10 10 16 10</td>
</tr>
</tbody>
</table>
Gradient-domain editing

Creation of image = least squares problem in terms of: 1) pixel intensities; 2) differences of pixel intensities

\[
\hat{v} = \arg \min_v \sum_i \left( a_i^T v - b_i \right)^2
\]

\[
\hat{v} = \arg \min_v (Av - b)^2
\]

Use Matlab least-squares solvers for numerically stable solution with sparse A
Perez et al., 2003
What’s the difference?

gradient domain blending - no blending

Slide by Mr. Hays
Limitations:

- Can’t do contrast reversal (gray on black -> gray on white)
- Colored backgrounds “bleed through”
- Images need to be very well aligned
Gradient Domain as Image Representation

See GradientShop paper as good example:

GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering

Pravin Bhat\textsuperscript{1} \quad C. Lawrence Zitnick\textsuperscript{2} \quad Michael Cohen\textsuperscript{1,2} \quad Brian Curless\textsuperscript{1}

\textsuperscript{1}University of Washington \quad \textsuperscript{2}Microsoft Research

http://www.gradientshop.com/
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation

![Diagram of pixel gradient with arrows pointing to each other, indicating manipulation of gradient values.](image-url)
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation

![Image of pixel gradients](image.png)
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
  - gradients – low level image-features
  - gradients – give rise to high level image-features
  - manipulate local gradients to manipulate global image interpretation
Motivation for gradient-domain filtering?

- Can be used to exert high-level control over images
Optimization framework
GradientShop

- Optimization framework
  - Input unfiltered image – $u$
GradientShop

- Optimization framework
  - Input unfiltered image – $u$
  - Output filtered image – $f$
Optimization framework

- Input unfiltered image – \( u \)
- Output filtered image – \( f \)
- Specify desired pixel-differences – \((g^x, g^y)\)

Energy function

\[
\min_f \ (f_x - g^x)^2 + (f_y - g^y)^2
\]
Optimization framework

- Input unfiltered image – \( u \)
- Output filtered image – \( f \)
- Specify desired pixel-differences – \( (g^x, g^y) \)
- Specify desired pixel-values – \( d \)

Energy function

\[
\min_f (f_x - g^x)^2 + (f_y - g^y)^2 + (f - d)^2
\]
Optimization framework

- Input unfiltered image – \( u \)
- Output filtered image – \( f \)
- Specify desired pixel-differences – \( (g^x, g^y) \)
- Specify desired pixel-values – \( d \)
- Specify constraints weights – \( (w^x, w^y, w^d) \)

Energy function

\[
\min_{f} \ w^x (f_x - g^x)^2 + w^y (f_y - g^y)^2 + w^d (f - d)^2
\]
GradientShop

**Inputs**

- $u$
- $u_x$
- $u_y$

**Application specific filtering**

**Constraints**

- $d$
- $g_x$
- $g_y$
GradientShop

**Inputs**

- $u$
- $u_x$
- $u_y$

**Application specific filtering**

**Constraints**

- $d$
- $g_x$
- $g_y$

**Least squares solver**

**Solution** - $f$
Pseudo image relighting

- change scene illumination in post-production
- example
Pseudo image relighting

- change scene illumination in post-production
- example

manual relight
Pseudo image relighting

- change scene illumination in post-production
- example

input
Pseudo image relighting

- change scene illumination in post-production
- example

GradientShop relight
Pseudo image relighting

- change scene illumination in post-production
- example

GradientShop relight
Pseudo image relighting

- change scene illumination in post-production
- example

GradientShop relight
Pseudo image relighting

- change scene illumination in post-production
- example

GradientShop relight
Pseudo image relighting

$u \quad o \quad f$
Pseudo image relighting

Energy function

$$\min_f w_x (f_x - g_x)^2 + w_y (f_y - g_y)^2 + w_d (f - d)^2$$
Pseudo image relighting

Energy function

\[
\min_{f} w^x (f_x - g_x)^2 + w^y (f_y - g_y)^2 + w^d (f - d)^2
\]

• Definition:
  • \( d = u \)
Pseudo image relighting

Energy function

$$\min_f \ w^x(f_x - g^x)^2 + w^y(f_y - g^y)^2 + w^d(f - d)^2$$

- Definition:
  - $$d = u$$
  - $$g^x(p) = u_x(p) * (1 + a(p))$$
  - $$a(p) = \max(0, -\nabla u(p) \cdot o(p))$$
Pseudo image relighting

Energy function

\[
\min_{f} w_x (f_x - g_x)^2 + w_y (f_y - g_y)^2 + w_d (f - d)^2
\]

- **Definition:**
  - \( d = u \)
  - \( g^x(p) = u_x(p) \cdot (1 + a(p)) \)
  - \( a(p) = \max(0, -\nabla u(p) \cdot o(p)) \)
Sparse data interpolation

- Interpolate scattered data over images/video
Sparse data interpolation

- Interpolate scattered data over images/video
- Example app: Colorization*

*Levin et al. – SIGGRAPH 2004
Sparse data interpolation
Sparse data interpolation

Energy function

\[
\min_f \quad w_x (f_x - g_x)^2 + w_y (f_y - g_y)^2 + w_d (f - d)^2
\]
Sparse data interpolation

Energy function

\[
\min_f \ w^x (f_x - g^x)^2 + w^y (f_y - g^y)^2 + w^d (f - d)^2
\]

- Definition:
  - \( d = \text{user\_data} \)
Sparse data interpolation

Energy function:

$$\min \min_{f} \quad w^x(f_x - g^x)^2 + \quad w^y(f_y - g^y)^2 + \quad w^d(f - d)^2$$

• Definition:
  • $$d = \text{user\_data}$$
  • if user\_data($p$) defined
    $$w^d(p) = 1$$
  else
    $$w^d(p) = 0$$

user data
Sparse data interpolation

Energy function

\[
\min_f \ w^x(f_x - g^x)^2 + \ w^y(f_y - g^y)^2 + \ w^d(f - d)^2
\]

- **Definition:**
  - \( d = \text{user\_data} \)
  - if \( \text{user\_data}(p) \) defined
    - \( w^d(p) = 1 \)
  - else
    - \( w^d(p) = 0 \)
  - \( g^x(p) = 0; \ g^y(p) = 0 \)
Sparse data interpolation

Energy function

\[
\min_f \quad w^x(f_x - g^x)^2 + w^y(f_y - g^y)^2 + w^d(f - d)^2
\]

- Definition:
  - \( d = \text{user\_data} \)
  - if user\_data\((p)\) defined
    \( w^d(p) = 1 \)
  - else
    \( w^d(p) = 0 \)
  - \( g^x(p) = 0; \ g^y(p) = 0 \)
  - \( w^x(p) = 1/(1 + c^*|u_x(p)|) \)
  - \( w^y(p) = 1/(1 + c^*|u_y(p)|) \)
Don’t blend, CUT!

So far we only tried to blend between two images. What about finding an optimal seam? 

Moving objects become ghosts
Segment the mosaic

- Single source image per segment
- Avoid artifacts along boundaries
  - Dijkstra’s algorithm
Minimal error boundary

overlapping blocks

vertical boundary

overlap error

min. error boundary
Seam Carving

Seam Carving for Content-Aware Image Resizing

Shai Avidan
Mitsubishi Electric Research Labs

Ariel Shamir
The Interdisciplinary Center & MERL

http://www.youtube.com/watch?v=6NcIJJXr7ugc
Seam Carving

• **Basic Idea:** remove unimportant pixels from the image
  – Unimportant = pixels with less “energy”

\[ E_1(I) = \left| \frac{\partial}{\partial x} I \right| + \left| \frac{\partial}{\partial y} I \right|. \]

• **Intuition for gradient-based energy:**
  – Preserve strong contours
  – Human vision more sensitive to edges – so try remove content from smoother areas
  – Simple, enough for producing some nice results
  – See their paper for more measures they have used
Finding the Seam?
The Optimal Seam

\[ E(I) = |\frac{\partial}{\partial x} I| + |\frac{\partial}{\partial y} I| \quad \Rightarrow \quad s^* = \arg \min_s E(s) \]
Dynamic Programming

• **Invariant property:**
  - $M(i,j) =$ minimal cost of a seam going through $(i,j)$ (satisfying the seam properties)
**Dynamic Programming**

\[
M(i, j) = E(i, j) + \min(M(i - 1, j - 1), M(i - 1, j), M(i - 1, j + 1))
\]

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>8</th>
<th>12</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2+5</td>
<td>3</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Programming

\[ M(i, j) = E(i, j) + \min(M(i - 1, j - 1), M(i - 1, j), M(i - 1, j + 1)) \]
Dynamic Programming

\[ M(i, j) = E(i, j) + \min(M(i - 1, j - 1), M(i - 1, j), M(i - 1, j + 1)) \]
Searching for Minimum

• Backtrack (can store choices along the path, but do not have to)
Backtracking the Seam

Michael Rubinstein — MIT CSAIL — mrub@mit.edu
Backtracking the Seam
Backtracking the Seam
Graphcuts

What if we want similar “cut-where-things-agree” idea, but for closed regions?

- Dynamic programming can’t handle loops
Graph cuts – a more general solution

Minimum cost cut can be computed in polynomial time

(max-flow/min-cut algorithms)
e.g. Lazy Snapping

Interactive segmentation using graphcuts

Also see the original Boykov&Jolly, ICCV’01, “GrabCut”, etc, etc ,etc.
Putting it all together

Compositing images

- Have a clever blending function
  - Feathering
  - blend different frequencies differently
  - Gradient based blending
- Choose the right pixels from each image
  - Dynamic programming – optimal seams
  - Graph-cuts

Now, let’s put it all together:

- Interactive Digital Photomontage, 2004 (video)
Interactive Digital Photomontage

Aseem Agarwala, Mira Dontcheva
Maneesh Agrawala, Steven Drucker, Alex Colburn
Brian Curless, David Salesin, Michael Cohen

http://www.youtube.com/watch?v=kzV-5135bGA