Today

- Image Stitching
- Image Matting

Compositing with moving objects

- Moving objects become ghosts
  - So far we only tried to blend between two images. What about finding an optimal seam?
Cutting & Stitching

- Segment the mosaic
  - Single source image per segment
  - Avoid artifacts along boundaries
    - Dijkstra's algorithm

Cutting: Finding Seams and Boundaries

- Fundamental Concept: The Image as a Graph
  - Intelligent Scissors: Good boundary = short path
  - Graph cuts: Good region has low cutting cost

Segmentation
Semi-automated segmentation

User provides imprecise and incomplete specification of region – your algorithm has to read his/her mind.

Key problems
1. What groups of pixels form cohesive regions?
2. What pixels are likely to be on the boundary of regions?
3. Which region is the user trying to select?

What makes a good region?
• Contains small range of color/texture
• Looks different than background
• Compact

What makes a good boundary?
• High gradient along boundary
• Gradient in right direction
• Smooth

The Image as a Graph

Node: pixel
Edge: cost of path or cut between two pixels
Intelligent Scissors
Mortenson and Barrett (SIGGRAPH 1995)

A good image boundary has a short path through the graph.

Intelligent Scissors
Mortenson and Barrett (SIGGRAPH 1995)

Intelligent Scissors
• Formulation: find good boundary between seed points
• Challenges
  – Minimize interaction time
  – Define what makes a good boundary
  – Efficiently find it

Intelligent Scissors: method
1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
4. Get path from seed to cursor, choose new seed, repeat
Intelligent Scissors: method

1. Define boundary cost between neighboring pixels
   a) Lower if edge is present (e.g., with edge(im, 'canny'))
   b) Lower if gradient is strong
   c) Lower if gradient is in direction of boundary

2. User specifies a starting point (seed)
   – Snapping

3. Compute lowest cost from seed to each other pixel
   – Dijkstra’s shortest path algorithm
**Dijkstra’s shortest path algorithm**

*Initialize*, given seed $s$:
- Compute $\text{cost}_s(q, r) \%$ cost for boundary from pixel $q$ to neighboring pixel $r$
- $\text{cost}(s) = 0 \%$ total cost from seed to this point
- $A = \{s\} \%$ set to be expanded
- $E = \{\} \%$ set of expanded pixels
- $P(q) \%$ pointer to pixel that leads to $q$

*Loop* while $A$ is not empty
1. $q = \text{pixel in } A \text{ with lowest cost}$
2. Add $q$ to $E$
3. for each pixel $r$ in neighborhood of $q$ that is not in $E$
   a) $\text{cost}_\text{tmp} = \text{cost}(q) + \text{cost}_s(q, r)$
   b) if ($r$ is not in $A$) OR ($\text{cost}_\text{tmp} < \text{cost}(r)$)
      i. $\text{cost}(r) = \text{cost}_\text{tmp}$
      ii. $P(r) = q$
      iii. Add $r$ to $A$

**Intelligent Scissors: method**

1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
3. Compute lowest cost from seed to each other pixel
4. Get new seed, get path between seeds, repeat

**Intelligent Scissors: improving interaction**

1. Snap when placing first seed
2. Automatically adjust to boundary as user drags
3. Freeze stable boundary points to make new seeds

**Where will intelligent scissors work well, or have problems?**
Grab cuts and graph cuts

Segmentation with graph cuts

\[ \text{Energy}(y; \theta, \text{data}) = \sum_{i} \psi_i(y_i; \theta, \text{data}) \sum_{i,j \text{edges}} \psi_{ij}(y_i, y_j; \theta, \text{data}) \]

Colour Model

Gaussian Mixture Model (typically 5-8 components)
Graph cuts segmentation

1. Define graph
   - usually 4-connected or 8-connected
2. Set weights to foreground/background
   - Color histogram or mixture of Gaussians for background and foreground
     \[ \text{ unary potential}(x) = -\log \left( \frac{P(c(x); \theta_{\text{foreground}})}{P(c(x); \theta_{\text{background}})} \right) \]
3. Set weights for edges between pixels
   \[ \text{ edge potential}(x, y) = k_1 + k_2 \exp \left( -\frac{\|f(x) - f(y)\|^2}{2\sigma^2} \right) \]
4. Apply min-cut/max-flow algorithm
5. Return to 2, using current labels to compute foreground, background models

What is easy or hard about these cases for graphcut-based segmentation?

Easier examples
More difficult Examples

Limitations of Graph Cuts

• Requires associative graphs
  – Connected nodes should prefer to have the same label

• Is optimal only for binary problems

Lazy Snapping [Li et al., 2004]

Other applications: stitching

Graphcut Textures – Kwatra et al. SIGGRAPH 2003

Ideal boundary:
1. Similar color in both images
2. High gradient in both images
Other applications: stitching

Graphcut Textures – Kwatra et al. SIGGRAPH 2003

Putting it all together

• Compositing images
  – Have a clever blending function
    • Feathering
    • Center-weighted
    • blend different frequencies differently
  – Choose the right pixels from each image
    • Graph-cuts

Interactive Digital Photomontage

[Agarwala et al., 2004]
Challenges

- Find good seams between parts of images so they can be joined with few visible artifacts

- Blend along seams to reduce or remove any artifacts remaining after joining

Graph Cut

\[
C(L) = \sum_p C_d(p, L(p)) + \sum_{p,q} C_i(p, q, L(p), L(q))
\]

Labeling \( L(p) \) gives the source image for each pixel \( p \).

Cost of a labeling (L) is data penalty (\( C_d \)) and interaction penalty (\( C_i \)).

Data penalty - distance to image objective

Interaction penalty - distance to seam objective.

**Want to minimize the cost function so that seams are minimized and image objectives are met as well as possible.**

Selective Composites
Summary

- Treat image as a graph
  - Pixels are nodes
  - Between-pixel edge weights based on gradients
  - Sometimes per-pixel weights for affinity to foreground/background

- Good boundaries are a short path through the graph (Intelligent Scissors, Seam Carving)

- Good regions are produced by a low-cost cut (GrabCuts, Graph Cut Stitching)
Today

• Image Stitching
• Image Matting

How does Superman fly?

Super-human powers?
OR
Image Matting and Compositing?

Motivation: compositing
Combining multiple images. Typically, paste a foreground object onto a new background

• Movie special effect
• Multi-pass CG
• Combining CG & film
• Photo retouching
  – Change background
  – Fake depth of field
  – Page layout: extract objects, magazine covers
Page layout, magazine covers

Photo editing
- Edit the background independently from foreground
Technical Issues

- **Compositing**
  - How exactly do we handle transparency?

- **Smart selection**
  - Facilitate the selection of an object

- **Matte extraction**
  - Resolve sub-pixel accuracy, estimate transparency

- **Smart pasting**
  - Don’t be smart with copy, be smart with paste
  - See homework (pyramid splining)
  - See also in a couple weeks (gradient manipulation)

- **Extension to video**
  - Where life is always harder

---

Alpha

- $\alpha$: 1 means opaque, 0 means transparent
- 32-bit images: R, G, B, $\alpha$

---

Why fractional alpha?

- Motion blur, small features (hair) cause partial occlusion

---

With binary alpha

From Digital Domain
With fractional alpha

Photoshop layer masks

Composing Two Elements

Optical Printing

From: “Industrial Light and Magic,” Thomas Smith (p. 181)

From: “Special Optical Effects,” Zoran Perisic
Limitations of alpha

- Hard to represent stainglasses
  - It focuses on subpixel occlusion (0 or 1)
- Does not model more complex optical effects
  - e.g. magnifying glass

Compositing

- Given the foreground color $F=(R_F, G_F, B_F)$, the background color $(R_B, G_B, B_B)$ and $\alpha$ for each pixel
- The over operation is: $C = \alpha F + (1-\alpha)B$
  - (in the premultiplied case, omit the first $\alpha$)

Matting problem

- Inverse problem:
  Assume an image is the over composite of a foreground and a background
- Given an image color $C$, find $F$, $B$ and $\alpha$ so that $C = \alpha F + (1-\alpha)B$
Matting ambiguity

- \( C = \alpha F + (1-\alpha)B \)
- How many unknowns, how many equations?

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

7 unknowns: \(\alpha\) and triplets for \(F\) and \(B\)
3 equations, one per color channel

With known background (e.g. blue/green screen):
4 unknowns, 3 equations

Traditional blue screen matting

- Invented by Petro Vlahos
  (Technical Academy Award 1995)
- Recently formalized by Smith & Blinn
- Initially for film, then video, then digital
- Assume that the foreground has no blue
- Note that computation of \(\alpha\) has to be analog, needs to be simple enough

From Cinefex
### Traditional blue screen matting

- Assume that blue $b$ and green $g$ channels of the foreground respect $b \cdot a_2 g$ for $a_2$ typically between 0.5 and 1.5
- $\alpha = 1 - a_1 (b - a_2 g)$
  - clamped to 0 and 1
  - $a_1$ and $a_2$ are user parameters
  - Note that $\alpha = 1$ where assumption holds

### Blue/Green screen matting issues

- **Color limitation**
  - Annoying for blue-eyed people
  - Adapt screen color (in particular green)
- **Blue/Green spilling**
  - The background illuminates the foreground, blue/green at silhouettes
  - Modify blue/green channel, e.g. set to min $(b, a_2 g)$
- **Shadows**
  - How to extract shadows cast on background

### Blue/Green screen matting issues

- **Color limitation**
  - Annoying for blue-eyed people
  - Adapt screen color (in particular green)
- **Blue/Green spilling**
  - The background illuminates the foreground, blue/green at silhouettes
  - Modify blue/green channel, e.g. set to min $(b, a_2 g)$
- **Shadows**
  - How to extract shadows cast on background

---

**From the Art & Science of Digital Compositing**

---

![Plate 52](image-url)
Extension: Chroma key

- Blue/Green screen matting exploits color channels
- Chroma key can use an arbitrary background color
- See e.g.
  - [URL](http://www.cs.utah.edu/~michael/chroma/)
  - Keith Jack, "Video Demystified", Independent Pub Group (Computer), 1996

Recall: Matting ambiguity

- \( C = \alpha F + (1-\alpha)B \)
- 7 unknowns: \( \alpha \) and triplets for \( F \) and \( B \)
- 3 equations, one per color channel

Natural matting

[Ruzon & Tomasi 2000, Chuang et al. 2001]

- Given an input image with arbitrary background
- The user specifies a coarse Trimap (known Foreground, known background and unknown region)
- Goal: Estimate \( F, B, \) alpha in the unknown region
  - We don’t care about \( B \), but it’s a byproduct/unknown

Bayes theorem

\[
P(x|y) = \frac{P(y|x) \; P(x)}{P(y)}
\]

The parameters you want to estimate
What you observe
Likelihood function
Prior probability
Constant w.r.t. parameters \( x \).
Matting and Bayes

- What do we observe?

\[ P(x|y) = P(y|x) \frac{P(x)}{P(y)} \]

The parameters you want to estimate
What you observe
Prior probability
Constant w.r.t. parameters \( x \).

Matting and Bayes

- What do we observe: Color \( C \)
- What are we looking for?

\[ P(x|C) = \frac{P(C|x) P(x)}{P(C)} \]

The parameters you want to estimate
Color you observe
Prior probability
Constant w.r.t. parameters \( x \).

Matting and Bayes

- What do we observe: Color \( C \)
- What are we looking for: \( F, B, \alpha \)

\[ P(F,B,\alpha|C) = \frac{P(C|F,B,\alpha) P(F,B,\alpha)}{P(C)} \]

Foreground, background, transparency you want to estimate
Color you observe
Prior probability
Constant w.r.t. parameters \( x \).
Matting and Bayes

- What do we observe: Color C
- What are we looking for: F, B, α
- Likelihood probability?
  - Given F, B and Alpha, probability that we observe C

\[
P(F,B,\alpha|C) = \frac{P(C|F,B,\alpha) P(F,B,\alpha)}{P(C)}
\]

Foreground, background, transparency you want to estimate

Likelihood function

Prior probability

Constant w.r.t. parameters x.

Prior probability:

- How likely is the foreground to have color F? the background to have color B? transparency to be α?

\[
P(F,B,\alpha|C) = \frac{P(C|F,B,\alpha) P(F,B,\alpha)}{P(C)}
\]
**Let's derive**

- Assume $F$, $B$ and $\alpha$ are independent

$$P(F,B,\alpha | C) = \frac{P(C|F,B,\alpha) P(F,B,\alpha)}{P(C)} = \frac{P(C|F,B,\alpha) P(F) P(B) P(\alpha)}{P(C)}$$

- But multiplications are hard!
- Make life easy, work with log probabilities

$L$ means log $P$ here:

$$L(F,B,\alpha | C) = L(C|F,B,\alpha) + L(F) + L(B) + L(\alpha) - L(C)$$

- And ignore $L(C)$ because it is constant

**Log Likelihood: $L(C|F,B,\alpha)$**

- Gaussian noise model: $\frac{\text{color difference}^2}{\sigma_C^2}$

- Take the log:

$$L(C|F,B,\alpha) = - ||C - \alpha F - (1-\alpha) B||^2 / \sigma_C^2$$

- Unfortunately not quadratic in all coefficients (product $\alpha B$)

**Prior probabilities $L(F)$ & $L(B)$**

- Gaussians based on pixel color from known regions
**Prior probabilities** \( L(F) & L(B) \)

- Gaussians based on pixel color from known regions
  - Can be anisotropic Gaussians
  - Compute the means \( \bar{F} \) and \( \bar{B} \) and covariance \( \Sigma_F, \Sigma_B \)

\[
\bar{F} = \frac{1}{N_F} \sum F_i \\
\Sigma_F = \frac{1}{N_F} \sum (F_i - \bar{F})(F_i - \bar{F})^T \\
L(F) = -(F - \bar{F})^T \Sigma_F^{-1} (F - \bar{F}) / 2
\]

- Same for \( B \)

**Prior probabilities** \( L(\alpha) \)

- What about alpha?
- Well, we don’t really know anything
- Keep \( L(\alpha) \) constant and ignore it
  - But see coherence matting for a prior on \( \alpha \)

**Questions?**
Recap: Bayesian matting equation

- Maximize \( L(C|F,B,\alpha) + L(F) + L(B) + L(\alpha) \)

\[
L(C|F,B,\alpha) = -\frac{1}{\sigma_C^2} \| C - \alpha F - (1-\alpha) B \|^2
\]

\[
L(F) = -(F - \overline{F})^T \Sigma_F^{-1} (F - \overline{F}) / 2
\]

\[
L(B) = -(B - \overline{B})^T \Sigma_B^{-1} (B - \overline{B}) / 2
\]

- Unfortunately, not a quadratic equation because of the product \((1-\alpha) B\)

\[\Rightarrow\] iteratively solve for \(F, B\) and for \(\alpha\)

For \(\alpha\) constant

- Derive \(L(C|F,B,\alpha) + L(F) + L(B) + L(\alpha)\) wrt \(F\) & \(B\), and set to zero gives

\[
\begin{bmatrix}
\Sigma_F^{-1} + I \alpha^2 / \sigma_C^2 & I \alpha(1-\alpha) / \sigma_C^2 \\
I \alpha(1-\alpha) / \sigma_C^2 & \Sigma_B^{-1} + I (1-\alpha)^2 / \sigma_C^2
\end{bmatrix}
\begin{bmatrix}
F \\
B
\end{bmatrix}
= \begin{bmatrix}
\Sigma_F^{-1} \overline{F} + C \alpha / \sigma_C^2 \\
\Sigma_B^{-1} \overline{B} + C(1-\alpha) / \sigma_C^2
\end{bmatrix},
\]

For \(F\) & \(B\) constant

- Derive \(L(C|F,B,\alpha) + L(F) + L(B) + L(\alpha)\) wrt \(\alpha\), and set to zero gives

\[
\alpha = \frac{(C - B) \cdot (F - B)}{\| F - B \|^2}
\]

Recap: Bayesian matting

- The user specifies a trimap
- Compute Gaussian distributions \(\overline{F}, \Sigma_F\) and \(\overline{B}, \Sigma_B\) for foreground and background regions
- Iterate
  - Keep \(\alpha\) constant, solve for \(F\) & \(B\) (for each pixel)
  - Keep \(F\) & \(B\) constant, solve for \(\alpha\) (for each pixel)

Note that pixels are treated independently
Results

• From Chuang et al. 2001
Extensions: Video

- Interpolate trimap between frames
- Exploit the fact that background might become visible

Video Matting of Complex Scenes

Tony T.Y. Chou, Anima Anandkumar, Ryan Cotter, Frank Salisbury, Richard Szeliski

1University of Washington  2Microsoft Research