Cutting & Growing

Tamara Berg
CSE 590 Computational Photography

Many slides from James Hays, Alyosha Efros, Derek Hoiem
Announcements

• HW2 due Oct 2

• Project Proposals
  – Send email to sbu590@gmail, with 1-2 paragraph summary by Oct 5
  – Presentations in class Oct 10 (5 min)
    • Idea
    • Data
    • Challenges
    • Timeline
Compositing with moving objects

Moving objects become ghosts

Don’t blend, cut!
Cutting & Stitching

- Segment the mosaic
  - Single source image per segment
  - Pick a good seam (cut) so that when you merge the segments it looks ok.
Cutting: Finding Seams and Boundaries

Segmentation
Cutting: Finding Seams and Boundaries
compositing from different scenes
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
Semi-automated segmentation

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

Key problems
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)
Intelligent Scissors
Mortenson and Barrett (SIGGRAPH 1995)

A good image boundary has a short path through the graph.
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\((\text{im}, 'canny')\))

b) Lower if gradient is strong

c) Lower if gradient is in direction of boundary
Gradients, Edges, and Path Cost

Gradient Magnitude

Edge Image

Path Cost
Intelligent Scissors: method

1. Define boundary cost between neighboring pixels
2. User specifies a starting point (seed)
   – Snapping
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
   - Djikstra’s shortest path algorithm
Dijkstra’s shortest path algorithm
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

Magic Wand

Intelligent Scissors

GrabCut

Regions

Boundary

Regions & Boundary

Source: Rother
Segmentation with graph cuts

\[
\text{Energy}(y; \theta, \text{data}) = \sum_i \psi_1(y_i; \theta, \text{data}) \sum_{i,j \in \text{edges}} \psi_2(y_i, y_j; \theta, \text{data})
\]
Segmentation with graph cuts

\[ \text{Energy}(y; \theta, \text{data}) = \sum_i \psi_1(y_i; \theta, \text{data}) \sum_{i,j \in \text{edges}} \psi_2(y_i, y_j; \theta, \text{data}) \]
Colour Model

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

Boykov and Jolly (2001)

Cut: separating source and sink; Energy: collection of edges

Min Cut: Global minimal energy in polynomial time
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
   \[
   \text{edge\_potential}(x, y) = k_1 + k_2 \exp \left\{ -\frac{\|c(x) - c(y)\|^2}{2\sigma^2} \right\}
   \]

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

Camouflage & Low Contrast

Initial Rectangle

Initial Result

Fine structure

Harder Case
Lazy Snapping (Li et al. SG 2004)
Limitations of Graph Cuts

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

• Is optimal only for binary problems
Other applications: stitching

Graphcut Textures – Kwarra et al. SIGGRAPH 2003
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

Interactive Digital Photomontage
Agarwala et al Siggraph 2004
Image Objectives

* User specified global image objectives for composite:
  Designated color
  Luminance
  Contrast
  Eraser (color most different from current composite)
  Source Image,
  ...

* Or Brush interface
Seam Objectives

Measures suitability of a seam between two image regions

- Match colors across seams
- Match colors and color gradients across seams
- Match colors across seams, but prefer seams that lie along edges
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.**
Summary of big ideas

• 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)
Growing
Cutting: Cut someone out to make more of them
But what if we want less of somebody?
Texture synthesis & hole-filling
Texture

- Texture depicts spatially repeating patterns
- Textures appear naturally and frequently
Texture Synthesis

• Goal of Texture Synthesis: create new samples of a given texture
• Many applications: virtual environments, hole-filling, texturing surfaces
Trivial solutions

1. Tiling

2. Mirrored Tiling

3. Randomly Generate Pixels of Same Color
The Challenge

Need to model the whole spectrum: from repeated to stochastic texture
One idea: Build Probability Distributions

Basic idea

1. Compute statistics of input texture (e.g., histogram of edge filter responses)
2. Generate a new texture that keeps those same statistics

One idea: Build Probability Distributions

But it (usually) doesn’t work

- Probability distributions are hard to model well

Input

Synthesized
Another idea: Sample from the image

- Assuming Markov property, compute $P(p \mid N(p))$
  - Building explicit probability tables infeasible
  - Instead, we search the input image for all similar neighborhoods — that’s our pdf for $p$
  - To sample from this pdf, just pick one match at random

Efros and Leung 1999 SIGGRAPH
Another idea: Sample from the image

The input image itself provides its truest probability distribution

1. For each pixel to be synthesized, find matching regions in source image
2. Fill pixel with center pixel of randomly selected matching region

Efros and Leung 1999 SIGGRAPH
Details

• How to match patches?
  – Gaussian-weighted SSD (more emphasis on nearby pixels)

• What order to fill in new pixels?
  – “Onion skin” order: pixels with most neighbors are synthesized first
  – To synthesize from scratch, start with a randomly selected small patch from the source texture

• How big should the patches be?
Idea from Shannon (Information Theory)

• Generate English-sounding sentences by modeling the probability of each word given the previous words (n-grams)

• Large “n” will give more structured sentences

“I spent an interesting evening recently with a grain of salt.”
Size of Neighborhood Window
Varying Window Size

Increasing window size
Texture synthesis algorithm

While image not filled

1. Get unfilled pixels with filled neighbors, sorted by number of filled neighbors

2. For each pixel, get top N matches based on visible neighbors
   - Patch Distance: Gaussian-weighted SSD

3. Randomly select one of the matches and copy pixel from it
Synthesis Results

french canvas

rafia weave
More Results

white bread

brick wall
Homage to Shannon
Hole Filling
Extrapolation
Summary

• The Efros & Leung texture synthesis algorithm
  – Very simple
  – Surprisingly good results
  – Synthesis is easier than analysis!
  – ...but very slow
Exercise – Texture Synthesis

Form groups of 2

Implement basic texture synthesis

Try it on some example textures from the web

Pseudo-Code here:
http://graphics.cs.cmu.edu/people/efros/research/NPS/alg.html
Texture synthesis algorithm

While image not filled

1. Get unfilled pixels with filled neighbors, sorted by number of filled neighbors

2. For each pixel, get top N matches based on visible neighbors
   - Patch Distance: Gaussian-weighted SSD

3. Randomly select one of the matches and copy pixel from it
Image Quilting [Efros & Freeman 2001]

Observation: neighbor pixels are highly correlated

Idea: unit of synthesis = block

- Exactly the same but now we want $P(B|N(B))$
- Much faster: synthesize all pixels in a block at once
Input texture

Random placement of blocks

Neighboring blocks constrained by overlap

Minimal error boundary cut
Minimal error boundary

overlapping blocks

vertical boundary

overlap error

min. error boundary

\[ \text{overlap error} = 2 \]
Failures
Political Texture Synthesis!

Bush campaign digitally altered TV ad

President Bush’s campaign acknowledged Thursday that it had digitally altered a photo that appeared in a national cable television commercial. In the photo, a handful of soldiers were multiplied many times.

This section shows a sampling of the duplication of soldiers.

Original photograph
In-painting natural scenes

Key idea: Filling order matters

**In-painting Result**

- Image with Hole
- Raster-Scan Order
- Onion-Peel (Concentric Layers)
- Gradient-Sensitive Order

Filling order

Fill a pixel that:

1. Is surrounded by other known pixels
2. Is a continuation of a strong gradient or edge

Comparison

Original  With Hole  Onion-Ring Fill  Criminisi

Comparison

Concentric Layers

Gradient Sensitive
Texture Transfer

• Try to explain one object with bits and pieces of another object:
Texture Transfer

Constraint

Texture sample
Texture Transfer

Take the texture from one image and “paint” it onto another object

Same as texture synthesis, except an additional constraint:
1. Consistency of texture
2. Patches from texture should correspond to patches from constraint in some way
Make your own sacred toast
Related idea: Image Analogies

Image Analogies, Hertzmann et al. SG 2001
Image analogies

• Define a similarity between A and B
• For each patch in B:
  – Find a matching patch in A, whose corresponding A’ also fits in well with existing patches in B’
  – Copy the patch in A’ to B’
• Algorithm is done iteratively, coarse-to-fine
Artistic Filters

A

A'

B

B'
Colorization
Super-resolution
Super-resolution (result!)
Things to remember

• Texture synthesis and hole-filling can be thought of as a form of probabilistic hallucination

• Simple, similarity-based matching is a powerful tool
  – Synthesis
  – Hole-filling
  – Transfer
  – Artistic filtering
  – Super-resolution
  – Recognition, etc.

• Key is usually how to define similarity
Cutting & Growing with Images
Creating and Exploring a Large Photorealistic Virtual Space

Josef Sivic, Biliana Kaneva, Antonio Torralba, Shai Avidan, William T Freeman

Discussion moved to Oct 10