Image Composition

COS 526
Princeton University

Modeled after lecture by Alexei Efros.
Slides by Efros, Durand, Freeman, Hays, Fergus, Lazebnik, Agarwala, Shamir, and Perez.
Image Blending

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

2. Blend them into the composite (in the right order)

Composite by David Dewey

Slide credit: A. Efros
Image Composition

Laplacian pyramid blending
Poisson composition
Graphcut seams
Texture synthesis
Image Composition

Laplacian pyramid blending
Poisson composition
Graphcut seams
Texture synthesis
Image Blending

Without Blending
Alpha Blending / Feathering

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

Slide credit: A. Efros
Affect of Window Size

Slide credit: A. Efros
Affect of Window Size

Slide credit: A. Efros
Good Window Size

“Optimal” Window: smooth but not ghosted

Slide credit: A. Efros
What is the Optimal Window?

To avoid seams
• window = size of largest prominent feature

To avoid ghosting
• window <= 2*size of smallest prominent feature

Natural to cast this in the *Fourier domain*
• largest frequency <= 2*size of smallest frequency
• image frequency content should occupy one “octave” (power of two)

Slide credit: A. Efros
What if the Frequency Spread is Wide

Idea (Burt and Adelson)

• Different window sizes for different frequencies

Method

• Decompose image into octaves (frequency bands)
• Feather each octave with appropriate window size
• Sum feathered octave images to reconstruct blended image

Slide credit: A. Efros
Laplacian Pyramid

Lowpass Images

Bandpass Images

Slide credit: A. Efros
Laplacian Pyramid Blending

Left pyramid

Blend

Right pyramid

Slide credit: A. Efros
Laplacian Pyramid Blending

Slide credit: A. Efros
laplacian level 4

laplacian level 2

laplacian level 0

left pyramid       right pyramid       blended pyramid
Laplacian Pyramid Blending

Slide credit: A. Efros
Laplacian Pyramid Blending

© david dmartin (Boston College)

Slide credit: A. Efros
Problem with Blending

What if colors/intensities are different?
Image Composition

Laplacian pyramid blending

Poisson composition

Graphcut seams

Texture synthesis
Gradient domain image editing

Motivation:
- Human visual system is very sensitive to gradient
- Gradient encode edges and local contrast quite well

Approach:
- Edit in the gradient domain
- Reconstruct image from gradient
Gradient domain image editing

sources/destinations

cloning

seamless cloning

Slide credit: F. Durand
Gradient domain
Seamless Poisson cloning

Given vector field $\mathbf{v}$ (pasted gradient), find the value of $f$ in unknown region that optimizes:

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$
Discrete Poisson solver

Minimize variational problem

\[
\min_{f|\Omega} \sum_{\langle p, q \rangle \cap \Omega \neq \emptyset} (f_p - f_q - v_{pq})^2, \text{ with } f_p = f_p^*, \text{ for all } p \in \partial \Omega
\]

Discretized gradient

Discretized

v: g(p) - g(q)

Boundary condition

Rearrange and call \( N_p \) the neighbors of \( p \)

\[
|N_p| f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial \Omega} f_q^* + \sum_{q \in N_p} v_{pq}
\]

Big yet sparse linear system

Only for boundary pixels

Slide credit: F. Durand
Image Composition Results

sources

destinations

cloning

seamless cloning

sources/destinations

cloning

seamless cloning
source/destination  cloning  seamless cloning

Perez et al. SIGGRAPH 03
Figure 2: **Concealment.** By importing seamlessly a piece of the background, complete objects, parts of objects, and undesirable artifacts can easily be hidden. In both examples, multiple strokes (not shown) were used.
Problem with composition

Misaligned (moving) objects become ghosts

Slide credit: A. Efros
Image Composition

Laplacian pyramid blending
Poisson cloning

Graph cut seams

Texture synthesis
Graph Cuts

General idea

- Single source image per segment (avoids blurring)
- Careful cut placement, plus optional blending (avoids seams)
Graph Cuts in Image Composition

overlapping blocks

vertical boundary

\[ \text{overlap error} = 2 \]

min. error boundary

Slide credit: A. Efros
Graph Cut Algorithm

Minimum cost cut can be computed in polynomial time
(max-flow/min-cut algorithms)

Boykov&Jolly, ICCV’01
Graph Cuts in Image Segmentation

Lazy Snapping [Li 2004]
Interactive segmentation using graphcuts
Graph cuts in Image Retargeting

Seam Carving
Graph cuts in Image Retargeting

Seam Carving
Problems with Graph Cuts
Image Composition

Laplacian pyramid blending
Poisson composition
Graphcut seams
Texture synthesis
Nonparametric Texture Synthesis

• Assume patches in output image should locally match patches in input image

• *Markov property:*  
  \[ p(\text{pixel} \mid \text{rest of image}) = p(\text{pixel} \mid \text{neighborhood}) \]

• Use patches from set of input images to model  
  \[ p(\text{pixel} \mid \text{neighborhood}) \]
Motivation from Language

Shannon (1948) proposed a way to generate English-looking text using \textit{N-grams}:

Assume a Markov model

Large corpus gives probability distribution for each letter, given \( N-1 \) previous letters

Starting from a seed, repeatedly sample conditional probabilities to generate new letters

Can also use whole words instead of letters
Results (using alt.singles corpus):

• “As I've commented before, really relating to someone involves standing next to impossible.”
• “One morning I shot an elephant in my arms and kissed him.”
• “I spent an interesting evening recently with a grain of salt.”

Notice how well local structure is preserved!

• Now let’s try this in 2D...
Efros & Leung Algorithm

Initially proposed by Garber (1981), but dismissed as too computationally expensive!
Efros & Leung Algorithm

Assume Markov property, sample from $P(p|N(p))$

Building explicit probability tables infeasible

Instead, we search the input image for all sufficiently similar neighborhoods and pick one match at random.

Synthesizing a pixel

non-parametric sampling

Input image
Finding matches

E.g., sum of squared differences (SSD)

- *Gaussian-weighted* to make sure closer neighbors are in better agreement

\[ \| \star \left( \begin{bmatrix} \text{image} \end{bmatrix} - \begin{bmatrix} \text{image} \end{bmatrix} \right) \| ^2 \]
Hole Filling
Extrapolation
Practical texture synthesis

Fast similarity search
Coherence
Multiresolution
Patches
Quilting
Similarity Search

Perform fast approximate nearest neighbor search using spatial data structure

- tree-structured vector quantization (TSVQ)
- kd-tree (optionally with PCA)

Perform fast approximate nearest neighbor search using randomized algorithm

- Patch-Match [Barnes09]
Coherence

input image

completed portion (grey)

output image
Coherence

![Image 1]

![Image 2]

![Image 3]

![Image 4]
Multiresolution

For textures with large-scale structures, use a *Gaussian pyramid* to reduce required neighborhood size.

1. Synthesize at low-resolution.
2. Repeat for higher-res levels: “neighborhood” consists of generated pixels at this level and all neighboring pixels at lower level.
Multiresolution

Example:

input

<Diagram>

output

input

<Copy>

search

neighborhood
Multiresolution

Results

1 level 5×5

1 level 11×11

3 levels 5×5
Patch-Based Synthesis

Copy patches of pixels rather than pixels

Observation: neighbor pixels are highly correlated

- Exactly the same as Efros & Leung but $P(B|N(B))$
- Much faster: synthesize all pixels in a block at once
Image Quilting [Efros & Freeman]

Regularly arranged patches
Efros & Freeman Algorithm

- Decompose image into tiles (patches)
- Synthesize tiles in raster order
- Search input texture for tile that satisfies overlap constraints (above and left)
- Paste new tile into resulting texture
  - Adjust overlap areas with graph cut, or other image composition method
Efros & Freeman Example

Random placement of blocks

Neighboring blocks constrained by overlap

Minimal error boundary cut

Example
Wexler Algorithm

- Initialize output image
- Decompose image into tiles (patches)

- Iteratively pick tile
  - Search for highest scoring tile from source images (e.g., best matches colors)
  - Adjust overlap areas with graph cut, or other image composition method
Image Melding [Darabi12]

- Initialize output image
- Decompose image into tiles (patches)

- Iteratively
  - Replace all tiles based on matches of color and gradient after gain normalization
  - Reconstruct image colors and gradients by voting with overlapping tiles
  - Solve for pixel colors with Poisson method
Image Melding: Combining Inconsistent Images using Patch-based Synthesis

Soheil Darabi\textsuperscript{1}, Connelly Barnes\textsuperscript{2}, Eli Shechtman\textsuperscript{2}, Dan B Goldman\textsuperscript{2}, Pradeep Sen\textsuperscript{1}

\textsuperscript{1}UNM Advanced Graphics Lab, \textsuperscript{2}Adobe Systems
Our Next Assignment ....