Blending and Compositing

René Magritte, “The Red Model”

Many slides from Alexei Efros, Allan Jepson, Robert Collins

CSC320: Introduction to Visual Computing
Michael Guerzhoy
Image Compositing

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

2. Blend them into the composite (in the right order)
Need Blending for Compositing

- The transition between the object and the background in real images is not sudden
- Thin features (e.g., hair) cause “mixed“ pixels
- Motion while the picture is taken causes blur
- Semi-transparent objects
Combining Two Images

- The transition is not smooth
Alpha Blending / Feathering

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

- An array the same size as the image
- $\alpha$ can be 1 (object 1), 0 (background/object 2), or between 0 and 1 (somewhere in between)
Effect of Window Size

- “Ghosting” happens if the transition is too slow
Effect of Window Size

- "Seams" are visible if the transition is too fast
Good Window Size

“Optimal” Window: smooth but not ghosted
What is the Optimal Window?

To avoid seams
  • window \( \geq \) size of largest prominent feature (and all the features)

To avoid ghosting
  • window \( \leq 2 \times \) size of smallest prominent feature

(explanation on the blackboard)
What is the Optimal Window?

For feathering to work:

• largest frequency $\leq 2$*size of smallest frequency
• So image frequency content should occupy one “octave” (power of two)
  • I.e., $|F(\omega)|$ is large only for $2^k \leq |\omega| \leq 2^{k+1}$

• Key idea: Coarse structure should blend very slowly between images (lots of feathering), while fine details should transition more quickly
Reminder: 2D Discrete Fourier Transform

\[ \hat{h}(k, l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} e^{-i(\omega_k n + \omega_l m)} h(n, m) \]

\[ h(n, m) = \frac{1}{NM} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} e^{i(\omega_k n + \omega_l m)} \hat{h}(k, l) \]

Often it is convenient to express frequency in vector notation with \( \vec{k} = (k, l)^t \), \( \vec{n} = (n, m)^t \), \( \vec{\omega}_{kl} = (\omega_k, \omega_l)^t \) and \( \vec{\omega}^t \vec{n} = \omega_k n + \omega_l m \).

2D Fourier Basis Functions: Sinusoidal waveforms of different wavelengths (scales) and orientations. Sinusoids on \( N \times M \) images with 2D frequency \( \vec{\omega}_{kl} = (\omega_k, \omega_l) = 2\pi(k/N, l/M) \) are given by:

\[ e^{i (\vec{\omega}^t \vec{n})} = e^{i \omega_k n} e^{i \omega_l m} = \cos(\vec{\omega}^t \vec{n}) + i \sin(\vec{\omega}^t \vec{n}) \]
What if the Frequency Spread is Wide

**Idea (Burt and Adelson)**

- Compute \( F_{\text{left}} = \text{FFT}(I_{\text{left}}) \), \( F_{\text{right}} = \text{FFT}(I_{\text{right}}) \)
- Decompose Fourier image into octaves (bands)
  - \( F_{\text{left}} = F_{\text{left}}^1 + F_{\text{left}}^2 + \ldots \)
- Feather corresponding octaves \( F_{\text{left}}^i \) with \( F_{\text{right}}^i \)
  - Can compute inverse FFT and feather in spatial domain
- Sum feathered octave images in frequency domain
- (In practice, we implement this in spatial domain)
Laplacian Pyramid: Overview

Lowpass Images

Bandpass Images
Laplacian Pyramid: Overview

We can recover the original image if we just have the coarsest Gaussian and all the laplacian “correction” images.
Reminder: Gaussian Pyramid

- Multi-level representation of an image
- The next level is smoothed and then downsampled every time

First three levels scaled to be the same size:
Laplacian Pyramid

• Each band of the Laplacian pyramid is the difference between two adjacent levels of the Gaussian pyramid, $[\vec{I}_0, \vec{I}_1, ..., \vec{I}_N]$

  • $\vec{b}_k = \vec{I}_k - E \vec{I}_{k+1}$

• $EI_{k+1}$ is the up-sampled smoothed version of $I_{k+1}$
Laplacian Pyramid

A Laplacian pyramid with four levels:
The Laplacian Pyramid in Frequency Domain

- Reminder:
  - Each level of the Laplacian pyramid is the result of filtering an image with a band-pass filter

High-pass / band-pass:
Band-passed Hybrid Image

High frequency → Low frequency
Reconstructing the Image from the Laplacian

**Construction:** of \([\vec{b}_0, \vec{b}_1, \ldots, \vec{b}_{L-1}, \vec{I}_L]\).

\[
\begin{align*}
\vec{I}_0 &= \vec{I} \\
\vec{I}_{k+1} &= R \vec{I}_k \\
\vec{b}_k &= \vec{I}_k - E \vec{I}_{k+1}
\end{align*}
\]

**Reconstruction:** of \(\vec{I}\) is exact (for any filters) and straightforward:

\[
\begin{align*}
\vec{I}_k &= \vec{b}_k + E \vec{I}_{k+1} \\
\vec{I} &= \vec{I}_0
\end{align*}
\]
Pyramid Blending

Left pyramid       blend       Right pyramid
Pyramid Blending
Laplacian levels:

- **Level 4**: Images show a smooth transition with minimal detail.
- **Level 2**: Images begin to exhibit more detail, particularly in the right pyramid.
- **Level 0**: Images reveal the finest details, with the left pyramid showing the most pronounced features.

The blended pyramid combines features from all levels, presenting a comprehensive view of the image with varying degrees of detail.
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:
   - $LS(i,j) = GR(l,j) \times LA(l,j) + (1 - GR(l,j)) \times LB(l,j)$
4. Collapse the $LS$ pyramid to get the final blended image
Blending Regions
Horror Photo

© david dmartin (Boston College)
Stitching Photos for Panoramas
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
2-band Blending

Low frequency ($\lambda > 2$ pixels)

High frequency ($\lambda < 2$ pixels)
Linear Blending
2-band Blending
Don’t blend, CUT!

Moving objects become ghosts

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

overlapping blocks

vertical boundary

overlap error

min. error boundary