Low-pass, Band-pass, High-pass filters

low-pass:

High-pass / band-pass:
Edges in images
What does blurring take away?
What does blurring take away?

smoothed (5x5 Gaussian)
High-Pass filter

smoothed – original
Image “Sharpening”

What does blurring take away?

original \[ \quad - \quad \text{smoothed (5x5)} \quad = \quad \text{detail} \]

Let’s add it back:

original \[ \quad + \quad \alpha \quad = \quad \text{sharpened} \]
Unsharp mask filter

\[ f + \alpha(f - f * g) = (1 + \alpha)f - \alpha f * g = f * ((1 + \alpha)e - \alpha g) \]
Hybrid Images

Salvador Dali

“Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln”, 1976
Band-pass filtering

Gaussian Pyramid (low-pass images)
Laplacian Pyramid

How can we reconstruct (collapse) this pyramid into the original image?
Blending
Alpha Blending / Feathering

\[ I_{\text{blend}} = \alpha I_{\text{left}} + (1-\alpha)I_{\text{right}} \]
Affect of Window Size
Affect of Window Size
“Optimal” Window: smooth but not ghosted
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)
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

Better implemented in \textit{spatial domain}
Octaves in the Spatial Domain

Lowpass Images

Bandpass Images
Pyramid Blending

Left pyramid  blend  Right pyramid

level k-2

level k-1

level k (= 1 pixel)
Pyramid Blending
Laplacian levels:

- Level 4
- Level 2
- Level 0

Left pyramid, right pyramid, blended pyramid.
Blending Regions
Laplacian Pyramid/Stack Blending

General Approach:

1. Build Laplacian pyramid/stack $L_X$ and $L_Y$ from images $X$ and $Y$
2. Build a Gaussian pyramid/stack $G_a$ from the binary alpha mask $a$
3. Form a combined pyramid/stack $L_{Blend}$ from $L_X$ and $L_Y$ using the corresponding levels of $G_a$ as weights:
   - $L_{Blend}(i,j) = G_a(l,j) \times L_X(l,j) + (1-G_a(l,j)) \times L_Y(l,j)$
4. Collapse the $L_{Blend}$ pyramid/stack to get the final blended image
Horror Photo

© david dmartin (Boston College)
Results from this class (fall 2005)
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 “Laplacian Stack” Blending

Low frequency ($\lambda > 2$ pixels)

High frequency ($\lambda < 2$ pixels)
Linear Blending
2-band Blending
Da Vinci and Peripheral Vision

https://en.wikipedia.org/wiki/Speculations_about_Mona_Lisa#Smile
Leonardo playing with peripheral vision

Livingstone, Vision and Art: The Biology of Seeing
Clues from Human Perception

Early processing in humans filters for various orientations and scales of frequency.

Perceptual cues in the mid frequencies dominate perception.

When we see an image from far away, we are effectively subsampling it.

Early Visual Processing: Multi-scale edge and blob filters.
Frequency Domain and Perception

Campbell-Robson contrast sensitivity curve
Lossy Image Compression (JPEG)

Block-based Discrete Cosine Transform (DCT)
Using DCT in JPEG

The first coefficient $B(0,0)$ is the DC component, the average intensity.

The top-left coeffs represent low frequencies, the bottom right – high frequencies.
Image compression using DCT

Quantize

- More coarsely for high frequencies (which also tend to have smaller values)
- Many quantized high frequency values will be zero

Encode

- Can decode with inverse dct

Filter responses

\[ G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \]

Quantization table

\[ Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \]

Quantized values

\[ B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
JPEG Compression Summary

Subsample color by factor of 2
  • People have bad resolution for color

Split into blocks (8x8, typically), subtract 128

For each block
  a. Compute DCT coefficients
  b. Coarsely quantize
     – Many high frequency components will become zero
  c. Encode (e.g., with Huffman coding)

http://en.wikipedia.org/wiki/YCbCr
http://en.wikipedia.org/wiki/JPEG
Block size in JPEG

Block size

- small block
  - faster
  - correlation exists between neighboring pixels

- large block
  - better compression in smooth regions

- It’s 8x8 in standard JPEG
JPEG compression comparison

89k  12k