Image Pyramids and Applications

Computer Vision
Jia-Bin Huang, Virginia Tech

Golconda, René Magritte, 1953
Administrative stuffs

- HW 1 will be posted tonight, due 11:59 PM Sept 25
- Anonymous feedback
Previous class: Image Filtering

• Sometimes it makes sense to think of images and filtering in the frequency domain
  • Fourier analysis

• Can be faster to filter using FFT for large images (N logN vs. N^2 for auto-correlation)

• Images are mostly smooth
  • Basis for compression

• Remember to low-pass before sampling
Spatial domain

Frequency domain

FFT

Inverse FFT
Today’s class

• Template matching

• Image Pyramids

• Compression

• Introduction to HW1
Template matching

• Goal: find 🙆 in image

• Main challenge: What is a good similarity or distance measure between two patches? $D(\square, \square)$
  • Correlation
  • Zero-mean correlation
  • Sum Square Difference
  • Normalized Cross Correlation
Matching with filters

• Goal: find [ ] in image

• Method 0: filter the image with eye patch

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

\( g = \text{filter} \quad f = \text{image} \)

What went wrong?
Matching with filters

- **Goal:** find \( \text{eye} \) in image
- **Method 1:** filter the image with zero-mean eye

\[
h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) \left( f[m+k, n+l] \right)
\]

- **Input**
- **Filtered Image (scaled)**
- **Thresholded Image**

**True detections**

**False detections**
Matching with filters

• Goal: find in image

• Method 2: Sum of squared differences (SSD)

\[ h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2 \]
Matching with filters

Can SSD be implemented with linear filters?

\[ h[m, n] = \sum_{k, l} (g[k, l] - f[m + k, n + l])^2 \]

\[ h[m, n] = \sum_{k, l} (g[k, l])^2 - 2 \sum_{k, l} g[k, l] f[m + k, n + l] + \sum_{k, l} (f[m + k, n + l])^2 \]

Constant  Filtering with g  Filtering with box filter
Matching with filters

• Goal: find an eye in image
• Method 2: SSD

\[ h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2 \]

What’s the potential downside of SSD?
Matching with filters

- Goal: find \( \text{\begin{picture}(20,20)
\put(10,10){\includegraphics[width=10px]{eye.png}}
\end{picture}} \) in image
- Method 3: Normalized cross-correlation

\[
h[m,n] = \frac{\sum_{k,l}(g[k,l] - \bar{g})(f[m + k, n + l] - \bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l] - \bar{g})^2\sum_{k,l}(f[m + k, n + l] - \bar{f}_{m,n})^2\right)^{0.5}}
\]

**Matlab:** `normxcorr2(template, im)`
Matching with filters

- Goal: find 🙆 in image
- Method 3: Normalized cross-correlation

![Input](image1)

![Normalized X-Correlation](image2)

![Thresholded Image](image3)
Matching with filters

• Goal: find 🕵️ in image

• Method 3: Normalized cross-correlation
Q: What is the best method to use?

A: Depends

• Zero-mean filter: fastest but not a great matcher
• SSD: next fastest, sensitive to overall intensity
• Normalized cross-correlation: slowest, invariant to local average intensity and contrast
Q: What if we want to find larger or smaller eyes?

A: Image Pyramid
Review of Sampling

Image $\xrightarrow{\text{Gaussian Filter}}$ Low-Pass Filtered Image $\xrightarrow{\text{Sub-sample}}$ Low-Res Image
Gaussian pyramid

Source: Forsyth
Template Matching with Image Pyramids

Input: Image, Template
1. Match template at current scale

2. Downsample image
   • In practice, scale step of 1.1 to 1.2

3. Repeat 1-2 until image is very small

4. Take responses above some threshold, perhaps with non-maxima suppression
Laplacian filter

unit impulse  ➖  Gaussian  ≈  Laplacian of Gaussian

Source: Lazebnik
Laplacian pyramid
Creating the Gaussian/Laplacian Pyramid

- Use same filter for smoothing in each step (e.g., Gaussian with $\sigma = 2$)
- Downsample/upsample with “nearest” interpolation
Hybrid Image in Laplacian Pyramid

High frequency $\rightarrow$ Low frequency
Reconstructing image from Laplacian pyramid

Image = \( L_1 + \text{Smooth}(\text{Upsample}(G_2)) \)

- Use same filter for smoothing as in deconstruction
- Upsample with “nearest” interpolation
- Reconstruction will be lossless
Major uses of image pyramids

• Object detection
  • Scale search
  • Features

• Detecting stable interest points

• Course-to-fine registration

• Compression
Coarse-to-fine Image Registration

1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   • Search smaller range

Why is this faster?

Are we guaranteed to get the same result?
Applications: Pyramid Blending
Applications: Pyramid Blending
Pyramid Blending

- At low frequencies, blend slowly
- At high frequencies, blend quickly
Image representation

• Pixels:
  • great for spatial resolution, poor access to frequency

• Fourier transform:
  • great for frequency, not for spatial info

• Pyramids/filter banks:
  • balance between spatial and frequency information
How is it that a 4MP image (12000KB) can be compressed to 400KB without a noticeable change?
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

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

\[\begin{array}{c}
u \\ \downarrow \\ v
\end{array}\]

### 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

1. Convert image to YCrCb
2. Subsample color by factor of 2
   • People have bad resolution for color
3. Split into blocks (8x8, typically), subtract 128
4. 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
Lossless compression (PNG)

1. Predict that a pixel’s value based on its upper-left neighborhood
2. Store difference of predicted and actual value
3. Pkzip it (DEFLATE algorithm)
Three views of image filtering

• Image filters in spatial domain
  • Filter is a mathematical operation on values of each patch
  • Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  • Filtering is a way to modify the frequencies of images
  • Denoising, sampling, image compression

• Templates and Image Pyramids
  • Filtering is a way to match a template to the image
  • Detection, coarse-to-fine registration
HW 1 – Hybrid Image

• Hybrid image = Low-Freq( Image A ) + Hi-Freq( Image B )
HW 1 – Image Pyramid
HW 1 – Edge Detection

Derivative of Gaussian filters

x-direction  y-direction
Things to remember

• Template matching (SSD or Normxcorr2)
  • SSD can be done with linear filters, is sensitive to overall intensity

• Gaussian pyramid
  • Coarse-to-fine search, multi-scale detection

• Laplacian pyramid
  • More compact image representation
  • Can be used for compositing in graphics

• Compression
  • In JPEG, coarsely quantize high frequencies
Thank you

• See you this Thursday

• Next class:
  • Edge detection