Templates and Image Pyramids

Computational Photography
Derek Hoiem, University of Illinois
Project 1

• Due Monday at 11:59pm
  – Options for displaying results
    • Web interface or redirect
      (http://www.pa.msu.edu/services/computing/faq/auto-redirect.html)
    • Backup (e.g., project server not working): send me a link
      – E-mail me: expected points, code, link to webpage; no need to e-mail images/results

• Questions?

• Remember to sign up for bulletin board (if not done already)
Review

1. Match the spatial domain image to the Fourier magnitude image

A

B

C

D

E
Today’s class: applications of filtering

• Template matching

• Coarse-to-fine alignment
  – Project 2

• Denoising, Compression (as time allows)
Template matching

• Goal: find in image

• Main challenge: What is a good similarity or distance measure between two patches?
  – Correlation
  – Zero-mean correlation
  – Sum Square Difference
  – Normalized Cross Correlation
Matching with filters

• Goal: find \( \text{ } \) in image

• Method 0: filter the image with eye patch

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

What went wrong?

Input

Filtered Image
Matching with filters

- Goal: find ☠ in image
- Method 1: filter the image with zero-mean eye

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

Input Filtered Image (scaled) Thresholded Image

True detections
False detections
Matching with filters

• Goal: find \( \begin{array}{c} \text{eye} \end{array} \) in image

• Method 2: SSD

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

Input 1- \( \sqrt{\text{SSD}} \) Thresholded Image

True detections

Thresholded Image
Matching with filters

Can SSD be implemented with linear filters?

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

• Goal: find 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 an 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: \texttt{normxcorr2(template, im)}
Matching with filters

- **Goal:** find 🕵️‍♀️ in image
- **Method 3:** Normalized cross-correlation

![Input](image1.png) ![Normalized X-Correlation](image2.png) ![Thresholded Image](image3.png)

**True detections**
Matching with filters

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

Input Normalized X-Correlation Thresholded Image

True detections
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

1. Image
2. Gaussian Filter
3. Low-Pass Filtered Image
4. Sample
5. Low-Res Image
Gaussian pyramid

Source: Forsyth
Laplacian filter

unit impulse → Gaussian ≈ Laplacian of Gaussian

Source: Lazebnik
Laplacian pyramid

Source: Forsyth
Computing Gaussian/Laplacian Pyramid

Can we reconstruct the original from the laplacian pyramid?

Hybrid Image in Laplacian Pyramid

High frequency $\rightarrow$ Low frequency

Extra points for project 1
Project 2: Image Alignment

- Try SSD alignment
- Try normxcorr2 alignment
- Simple implementation will work for small images
- But larger images will take forever (well, many hours)
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?
Question

Can you align the images using the FFT?

Implementation is extra points for project 2
How is it that a 4MP image can be compressed to a few hundred KB without a noticeable change?
Lossy Image Compression (JPEG)

Block-based Discrete Cosine Transform (DCT)

Slides: Efros
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

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

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
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)
Denoising

Additive Gaussian Noise

Gaussian Filter
Reducing Gaussian noise

Smoothing with larger standard deviations suppresses noise, but also blurs the image

Source: S. Lazebnik
Reducing salt-and-pepper noise by Gaussian smoothing

3x3  5x5  7x7
Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window.

![Median Filtering Diagram]

- Is median filtering linear?

Source: K. Grauman
Median filter

• What advantage does median filtering have over Gaussian filtering?
  – Robustness to outliers

Source: K. Grauman
Median filter

Salt-and-pepper noise

Median filtered

- MATLAB: medfilt2(image, [h w])

Source: M. Hebert
Median vs. Gaussian filtering

Gaussian

3x3 | 5x5 | 7x7
---|---|---

Median
Other filter choices

• Weighted median (pixels further from center count less)

• Clipped mean (average, ignoring few brightest and darkest pixels)

• Bilateral filtering (weight by spatial distance and intensity difference)

Image: [http://vision.ai.uiuc.edu/?p=1455](http://vision.ai.uiuc.edu/?p=1455)
Review of Last 3 Days

• Filtering in spatial domain
  – Slide filter over image and take dot product at each position
  – Remember linearity (for linear filters)
  – Examples
    • 1D: [-1 0 1], [0 0 0 0.5 1 1 1 0.5 0 0 0]
    • 1D: [0.25 0.5 0.25], [0 0 0 0.5 1 1 1 0.5 0 0 0]
    • 2D: [1 0 0 ; 0 2 0 ; 0 0 1]/4
Review of Last 3 Days

- Linear filters for basic processing
  - Edge filter (high-pass)
  - Gaussian filter (low-pass)

\[-1 \ 1\]
Review of Last 3 Days

• Derivative of Gaussian
Review of Last 3 Days

• Filtering in frequency domain
  – Can be faster than filtering in spatial domain (for large filters)
  – Can help understand effect of filter
  – Algorithm:
    1. Convert image and filter to fft (fft2 in matlab)
    2. Pointwise-multiply ffts
    3. Convert result to spatial domain with ifft2
Review of Last 3 Days

• Applications of filters
  – 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
    • Can be used for blending (later)
    • More compact image representation
Review of Last 3 Days

• Applications of filters
  – Downsampling
    • Need to sufficiently low-pass before downsampling
  – Compression
    • In JPEG, coarsely quantize high frequencies
  – Reducing noise (important for aesthetics and for later processing such as edge detection)
    • Gaussian filter, median filter, bilateral filter
Next class

• Light and color