Templates, Image Pyramids, and Filter Banks

Computer Vision
James Hays, Brown

Slides: Hoiem and others
Reminder

• Project 1 due Friday
Fourier Bases

Teases away fast vs. slow changes in the image.

This change of basis is the Fourier Transform
Fourier Bases

in Matlab, check out: `imagesc(log(abs(fftshift(fft2(im)))));`
Man-made Scene
Can change spectrum, then reconstruct
Low and High Pass filtering
What is the spatial representation of the hard cutoff in the frequency domain?
1. Match the spatial domain image to the Fourier magnitude image
Today’s class

• Template matching

• Image Pyramids

• Filter banks and texture
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 🕵️‍♂️ in image
- **Method 0:** filter the image with eye patch

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

What went wrong?

Input

Filtered Image

f = image
g = filter
Matching with filters

- **Goal:** find \(\text{\includegraphics{eye.png}}\) in image

- **Method 1:** filter the image with zero-mean eye

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

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

- True detections
- False detections
Matching with filters

• Goal: find \( \square \) in image

• Method 2: SSD

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

- Goal: find in image
- Method 3: Normalized cross-correlation
Q: What is the best method to use?

A: Depends

• SSD: faster, sensitive to overall intensity
• Normalized cross-correlation: slower, invariant to local average intensity and contrast
• But really, neither of these baselines are representative of modern recognition.
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 \[\xrightarrow{\text{Sample}}\] Low-Res Image
Gaussian pyramid

512  256  128  64  32  16  8

Source: Forsyth
Template Matching with Image Pyramids

Input: Image, Template
1. Match template at current scale
2. Downsample image
3. Repeat 1-2 until image is very small
4. Take responses above some threshold, perhaps with non-maxima suppression
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?
2D edge detection filters

\[ h_\sigma(u, v) = \frac{1}{2\pi \sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \]

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]

\[ \nabla^2 h_\sigma(u, v) \]

\[ \nabla^2 \text{ is the Laplacian operator:} \]

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]
Laplacian filter

unit impulse

Gaussian

Laplacian of Gaussian

Source: Lazebnik
Computing Gaussian/Laplacian Pyramid

Can we reconstruct the original from the laplacian pyramid?

Laplacian pyramid

Source: Forsyth
Hybrid Image
Hybrid Image in Laplacian Pyramid

High frequency $\rightarrow$ Low frequency
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
Major uses of image pyramids

- Compression

- Object detection
  - Scale search
  - Features

- Detecting stable interest points

- Registration
  - Course-to-fine
Application: Representing Texture
Texture and Material

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
Texture and Orientation

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
What is texture?

Regular or stochastic patterns caused by bumps, grooves, and/or markings
How can we represent texture?

- Compute responses of blobs and edges at various orientations and scales
Overcomplete representation: filter banks

LM Filter Bank

Code for filter banks: www.robots.ox.ac.uk/~vgg/research/texclass/filters.html
Filter banks

- Process image with each filter and keep responses (or squared/abs responses)
How can we represent texture?

• Measure responses of blobs and edges at various orientations and scales

• Idea 1: Record simple statistics (e.g., mean, std.) of absolute filter responses
Can you match the texture to the response?

Filters

A

B

C

Mean abs responses
Representing texture by mean abs response
Representing texture

• Idea 2: take vectors of filter responses at each pixel and cluster them, then take histograms (more on in later weeks)
Review of last three days
Review: Image filtering

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

\[
f[.,.] \\
\]

\[
h[.,.] \\
\]

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

Credit: S. Seitz
Image filtering

\[ f[\cdot, \cdot] \]

\[ h[\cdot, \cdot] \]

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

\[
\begin{bmatrix}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{bmatrix}
\]

intensity image

\[
\ast
\]

=
Filtering in frequency domain

1. **Intensity Image**
2. **FFT** of the intensity image.
3. **Inverse FFT** of the modified frequency domain.
4. **Resulting Image** after applying the filter.
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

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

\[
\begin{bmatrix}
-1 & 1
\end{bmatrix}
\]

FFT of Gradient Filter

FFT of Gaussian
Review of Last 3 Days

• Derivative of Gaussian
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
    • Teases apart different frequency bands while keeping spatial information
    • Can be used for compositing in graphics
  – Downsampling
    • Need to sufficiently low-pass before downsampling
Next Lectures

• Image representation (e.g. SIFT) and matching across multiple views (e.g. Stereo, Structure from Motion).