CS4501: Introduction to Computer Vision
Filtering, Frequency, and Edges

Various slides from previous courses by:
D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL), S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook), J. M. Frahm (UNC), V. Ordonez (UVA).
Last Class

- Convolution Operation
- Image Blurring / Gaussian Blur
- Image Gradients: The Sobel Operator
Today’s Class

• Recap on Sobel Operator
• Filtering in Frequency
• Canny Edge Detector (also next class)
Image filtering: Convolution operator
Important Filter: Sobel operator

\[
k(x, y) = \begin{bmatrix}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{bmatrix}
\]

Other filters

Vertical Edge (absolute value)

Sobel

1 0 -1
2 0 -2
1 0 -1

Slide by James Hays
Other filters

Sobel

Horizontal Edge
(absolute value)

Slide by James Hays
Sobel operators are equivalent to 2D partial derivatives of the image

- Vertical sobel operator – Partial derivative in X (width)
- Horizontal sobel operator – Partial derivative in Y (height)

- Can compute magnitude and phase at each location.

- Useful for detecting edges
Sobel filters are (approximate) partial derivatives of the image

Let \( f(x, y) \) be your input image, then the partial derivative is:

\[
\frac{\partial f(x, y)}{\partial x} = \lim_{h \to 0} \frac{f(x + h, y) - f(x, y)}{h}
\]

Also:

\[
\frac{\partial f(x, y)}{\partial x} = \lim_{h \to 0} \frac{f(x + h, y) - f(x - h, y)}{2h}
\]
But digital images are not continuous, they are discrete

Let $f[x, y]$ be your input image, then the partial derivative is:

$$\Delta_x f[x, y] = f[x + 1, y] - f[x, y]$$

Also:

$$\Delta_x f[x, y] = f[x + 1, y] - f[x - 1, y]$$
But digital images are not continuous, they are discrete

Let $f[x, y]$ be your input image, then the partial derivative is:

$$\Delta_x f[x, y] = f[x + 1, y] - f[x, y]$$

Also:

$$\Delta_x f[x, y] = f[x + 1, y] - f[x - 1, y]$$
Frequency

Figure by National Instruments
Any function can be approximated by a polynomial function

Taylor Series expansion

\[ f(x) = f(a) + \frac{f'(a)}{1!} (x - a) + \frac{f''(a)}{2!} (x - a)^2 + \frac{f'''(a)}{3!} (x - a)^3 + \cdots. \]

...if you let your polynomial have a high degree

...AND you can compute the derivatives of the original function easily.
\[ e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots \]

\[ \sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \]

\[ \cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots \]

\[ \ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots \quad \text{for } |x| < 1 \]

\[ \tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots \quad \text{for } |x| < 1 \]
Difficult in practice

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.

• Don’t believe it?
  – Neither did Lagrange, Laplace, Poisson and other big wigs
  – Not translated into English until 1878!

• But it’s (mostly) true!
  – called Fourier Series
  – there are some subtle restrictions

...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.
Example

Square wave

Approximation Using sines

\[
f(x) = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x + \frac{1}{7} \sin 7x
\]
Discrete Fourier Transform

\[ F(u) = \sum_{x=0}^{N-1} f(x) \left[ \cos \left(-2\pi \left(\frac{ux}{N}\right)\right) + i \sin \left(-2\pi \left(\frac{ux}{N}\right)\right) \right] \]
Keep in mind Euler’s Equation

\[ e^{ix} = \cos x + i \sin x \]

We can compute the real and the imaginary part of the complex number.
Discrete Fourier Transform

\[ F(u) = \sum_{x=0}^{N-1} f(x) \left( \cos \left( -2\pi \frac{ux}{N} \right) + i \sin \left( -2\pi \frac{ux}{N} \right) \right) \]

\[ F(u) = \sum_{x=0}^{N-1} f(x) \exp \left( -2\pi i \frac{ux}{N} \right) \]
Discrete Fourier Transform

$$F(u) = \sum_{x=0}^{N-1} f(x) \exp \left[ -2\pi i \left( \frac{ux}{N} \right) \right]$$

Inverse Discrete Fourier Transform

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) \exp \left[ 2\pi i \left( \frac{ux}{N} \right) \right]$$
More generally for images (2D DFT and iDFT)

\[ F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp \left[ -2\pi i \left( \frac{xu}{M} + \frac{yv}{N} \right) \right] \]

\[ f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \exp \left[ 2\pi i \left( \frac{xu}{M} + \frac{yv}{N} \right) \right] \]
Discrete Fourier Transform - Visualization

- $|F(u,v)|$ generally decreases with higher spatial frequencies
- phase appears less informative

Slide by A. Zisserman
Fourier Transform

• Fourier transform stores the magnitude and phase at each frequency
  • Magnitude encodes how much signal there is at a particular frequency
  • Phase encodes spatial information (indirectly)
  • For mathematical convenience, this is often notated in terms of real and complex numbers

\[ A = \pm \sqrt{R(\omega)^2 + I(\omega)^2} \quad \text{Amplitude:} \]
\[ \phi = \tan^{-1} \frac{I(\omega)}{R(\omega)} \quad \text{Phase:} \]
Image Filtering in the Frequency Domain

$f(x, y)$

$|F(u, v)|$
Image Filtering in the Frequency Domain

$f(x,y)$

$|F(u,v)|$
Image Filtering in the Frequency Domain

\( f(x,y) \)

\( |F(u,v)| \)

original

low pass

high pass

Slide by A. Zisserman
The Convolution Theorem

• The Fourier transform of the convolution of two functions is the product of their Fourier transforms

\[ F[g * h] = F[g]F[h] \]

• Convolution in spatial domain is equivalent to multiplication in frequency domain!

\[ g * h = F^{-1}[F[g]F[h]] \]

How can this be useful?
Blurring in the Time vs Frequency Domain

Example by A. Zisserman
Blurring in the Time vs Frequency Domain

\[ f(x,y) \rightarrow \text{Gaussian scale=3 pixels} \rightarrow g(x,y) \]

Fourier transform

\[ |F(u,v)| \rightarrow \text{x} \]
Blurring in the Time vs Frequency Domain

Example by A. Zisserman
Why Frequency domain?

• Because the Discrete Fourier Transform can be computed fast using the Fast Fourier Transform FFT algorithm.

• Because the running time does not depend on the size of the kernel matrix.

• However rarely used these days because most filters used in Computer vision are 3x3, 5x5, e.g. relatively small.
Final Thoughts – JPEG Image Compression

- Small amount of information can recover almost the original image with some loss in resolution.

- Images are dominated by low frequency information. e.g. no need to store repeated pixels.

- In practice JPEG uses a simpler transformation called Discrete Cosine Transform DCT.
Questions?
Edge Detection
**Edge Detection**

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

- **Ideal:** artist’s line drawing (but artist is also using object-level knowledge)

Source: D. Lowe
Why do we care about edges?

• Extract information, recognize objects

• Recover geometry and viewpoint

Source: J. Hays
Origin of Edges

• Edges are caused by a variety of factors

Source: Steve Seitz
Edge Detection

• Back to Sobel

But Ideally we want an output where:
\[ g(x,y) = 1 \text{ if edge} \]
\[ g(x,y) = 0 \text{ if background} \]
Edge Detection

• Sobel + Thresholding

\[
g(x, y) = \begin{cases} 
1, & f(x, y) \geq \tau \\
0, & f(x, y) < \tau 
\end{cases}
\]
• Sobel + Thresholding

\[ g(x, y) = \begin{cases} 
1, & f(x, y) \geq \tau \\
0, & f(x, y) < \tau 
\end{cases} \]

Problems:

• Edges are too wide: We want 1-pixel wide edges if possible.
• Lots of disconnected edges: We want to respect continuity or connectivity.
Solution: Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Non-Maximum Suppression
- Hysteresis and connectivity analysis

Similar to Sobel: Blurring + Gradients
Example

- original image

Source: Juan C. Niebles and Ranjay Krishna.
Derivative of Gaussian filter

Source: Juan C. Niebles and Ranjay Krishna.
Compute gradients (DoG)

X-Derivative of Gaussian  Y-Derivative of Gaussian  Gradient Magnitude

Source: J. Hays
Get orientation at each pixel

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]

Source: J. Hays
Compute gradients (DoG)

Source: Juan C. Niebles and Ranjay Krishna.
Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response

Source: Juan C. Niebles and Ranjay Krishna.
Non-maximum suppression

• Edge occurs where gradient reaches a maxima
• Suppress non-maxima gradient even if it passes threshold
• Only eight angle directions possible
  • Suppress all pixels in each direction which are not maxima
  • Do this in each marked pixel neighborhood

Source: Juan C. Niebles and Ranjay Krishna.
Remove spurious gradients

$|\nabla G|(x, y)$ is the gradient at pixel $(x, y)$

$$M(x, y) = \begin{cases} |\nabla G|(x, y) & \text{if } |\nabla G|(x, y) > |\nabla G|(x', y') \\ & \text{and } |\nabla G|(x, y) > |\nabla G|(x'', y'') \\ 0 & \text{otherwise} \end{cases}$$

$x'$ and $x''$ are the neighbors of $x$ along normal direction to an edge

Source: Juan C. Niebles and Ranjay Krishna.
Non-maximum suppression

- Edge occurs where gradient reaches a maxima
- Suppress non-maxima gradient even if it passes threshold
- Only eight angle directions possible
  - Suppress all pixels in each direction which are not maxima
  - Do this in each marked pixel neighborhood

Source: Juan C. Niebles and Ranjay Krishna.
Non-maximum suppression

At q, we have a maximum if the value is larger than those at both p and at r.
Interpolate to get these values.

Source: D. Forsyth
Non-max Suppression

Before                         After

Source: Juan C. Niebles and Ranjay Krishna.
Canny edge detector

• Suppress Noise
• Compute gradient magnitude and direction
• Apply Non-Maximum Suppression
  • Assures minimal response
• Use hysteresis and connectivity analysis to detect edges

Source: Juan C. Niebles and Ranjay Krishna.
Hysteresis thresholding

- Avoid streaking near threshold value
- Define two thresholds: Low and High
  - If less than Low, not an edge
  - If greater than High, strong edge
  - If between Low and High, weak edge

Source: Juan C. Niebles and Ranjay Krishna.
Hysteresis thresholding

If the gradient at a pixel is

• above High, declare it as an ‘strong edge pixel’
• below Low, declare it as a “non-edge-pixel”
• between Low and High
  • Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to an ‘strong edge pixel’ directly or via pixels between Low and High

Source: Juan C. Niebles and Ranjay Krishna.
Hysteresis thresholding

Source: S. Seitz
Final Canny Edges

Source: Juan C. Niebles and Ranjay Krishna.
Canny edge detector

1. Filter image with $x$, $y$ derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   • Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   • Define two thresholds: low and high
   • Use the high threshold to start edge curves and the low threshold to continue them

Source: Juan C. Niebles and Ranjay Krishna.
Canny Edge Detector

• Classic algorithm in Computer Vision / Image Analysis
• Commonly implemented in most libraries

• e.g. in Python you can find it in the `skimage` package. OpenCV also has an implementation with python bindings.
Corners (and Interest Points)

• How to find corners? What is a corner?