CS5670: Computer Vision
Lecture 2: Edge detection

From Sandlot Science
**Announcements**

- Project 1 (Hybrid Images) is now on the course webpage (see *Projects* link)
  - Due Friday, Feb 10, by 8pm on Github Classroom
  - Artifact due Monday, Feb 13, by 8pm on CMSX
  - Project to be done individually
  - Skeleton code available soon on Github Classroom – instructions for setting up Python environment on the project webpage
- Course webpage: [https://www.cs.cornell.edu/courses/cs5670/2023sp/](https://www.cs.cornell.edu/courses/cs5670/2023sp/)
  - Has lectures, projects, office hours, etc
- In-class Quiz first 10 minutes of class this Thursday
Project 1: Hybrid Images
Project 1 Demo
Edge detection

- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels
Origin of edges

- Edges are caused by a variety of factors

  - surface normal discontinuity
  - depth discontinuity
  - surface color discontinuity
  - illumination discontinuity
Images as functions...

- Edges look like steep cliffs
Characterizing edges

• An edge is a place of *rapid change* in the image intensity function.

Source: L. Lazebnik
Image derivatives

• How can we differentiate a digital image $F[x,y]$?
  – Option 1: reconstruct a continuous image, $f$, then compute the derivative
  – Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a linear filter?

$$\frac{\partial f}{\partial x} \quad H_x$$

$$\frac{\partial f}{\partial y} \quad H_y$$

Source: S. Seitz
The gradient points in the direction of most rapid increase in intensity

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \]

\[ \nabla f = [0, \frac{\partial f}{\partial y}] \]

The edge strength is given by the gradient magnitude:

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]

The gradient direction is given by:

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

- how does this relate to the direction of the edge?

Source: Steve Seitz
Image gradient

Source: L. Lazebnik
Effects of noise

Noisy input image

Where is the edge?

Source: S. Seitz
Solution: smooth first

To find edges, look for peaks in $\frac{d}{dx}(f \ast h)$.

Source: S. Seitz
Differentiation is convolution, and convolution is associative: $\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$

This saves us one operation: $f$
The 1D Gaussian and its derivatives

\[ G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \]

\[ G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma} \left( \frac{x}{\sigma} \right) G_\sigma(x) \]
2D edge detection filters

Gaussian

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

derivative of Gaussian ($x$)

$$\frac{\partial}{\partial x} h_\sigma(u, v)$$
Derivative of Gaussian filter

\[ x \text{-direction} \quad \text{and} \quad y \text{-direction} \]
The Sobel operator

• Common approximation of derivative of Gaussian

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{bmatrix}
\quad \frac{1}{8}
\]

\[
\begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{bmatrix}
\quad \frac{1}{8}
\]

\(s_x\) \hspace{2cm} \(s_y\)

• The standard definition of the Sobel operator omits the 1/8 term
  – doesn’t make a difference for edge detection
  – the 1/8 term \textbf{is} needed to get the right gradient magnitude
Sobel operator: example

Example

original image

Demo: [http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/](http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/)

Image credit: Joseph Redmon
Finding edges

smoothed gradient magnitude
Finding edges

smoothed gradient magnitude
Finding edges

thresholding

where is the edge?
Get Orientation at Each Pixel

- Get orientation (below, threshold at minimum gradient magnitude)

\[ \theta = \text{atan2}(g_y, g_x) \]
Non-maximum suppression

- Check if pixel is local maximum along gradient direction
  - requires *interpolating* pixels p and r
Before Non-max Suppression
After Non-max Suppression
Thresholding edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - \( R > T \): strong edge
  - \( R < T \) but \( R > t \): weak edge
  - \( R < t \): no edge
- Why two thresholds?
Connecting edges

- Strong edges are edges!
- Weak edges are edges iff they connect to strong
- Look in some neighborhood (usually 8 closest)
Canny edge detector

MATLAB: `edge(image,'canny')`

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei, J. Redmon
Canny edge detector

• Our first computer vision pipeline!
• Still a widely used edge detector in computer vision


• Depends on several parameters:
  
  high threshold
  low threshold
  \( \sigma \) : width of the Gaussian blur
Canny edge detector

- The choice of $\sigma$ depends on desired behavior
  - large $\sigma$ detects “large-scale” edges
  - small $\sigma$ detects fine edges

Source: S. Seitz
**Scale space** [Witkin 83]

- Properties of scale space (w/ Gaussian smoothing)
  - edge position may shift with increasing scale (σ)
  - two edges may merge with increasing scale
  - an edge may **not** split into two with increasing scale
Questions?