Piazza etiquette with 300 students

• Read the project description
• Check top-level notes post on Piazza
• Search for similar questions on Piazza

• Question number in title
• Description of problem
• Description of how you have tried to debug it
• Code: divide and conquer; minimal non-working example really helps
Gradescope – Late submissions

• Yes, there is a grace period
Gradescope – question PDFs

• Change of plan to help with grading 300 scripts

• No longer asking you to assign pages

• Now: please stick to pages
  – we give you plenty of space.
Grok

To understand intuitively; completely; [to the point of sharing an existence.]

1961  Robert A. Heinlein book; coined term
1980s  Took on meaning in computing circles

"There isn't any software! Only different internal states of hardware. It's all hardware! It's a shame programmers don't grok that better."

https://en.wikipedia.org/wiki/Grok
Review of Filtering

- 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 Fourier domain (e.g., `numpy.fft.fft2()`)
  2. Element-wise multiply their decompositions
  3. Convert result to spatial domain with inverse Fourier transform (e.g., `numpy.fft.ifft2()`)

You will play with code in Proj2 questions
Review of Filtering

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

\[-1 \ 1\]

FFT of Gradient Filter

FFT of Gaussian

Gaussian
More Useful Filters

Single Gaussian

$y$

$y'$

$y''$

1st Derivative of Gaussian

2nd Derivative (Laplacian of Gaussian)

Earl F. Glynn
Things to Remember

Sometimes it helps to think of images and filtering in the frequency domain
  – Fourier analysis

Can be faster to filter using FFT for large images
  – \( N \log N \) vs. \( N^2 \) for convolution/correlation

Images are mostly smooth
  – Basis for compression

Remember to low-pass before sampling
  • Otherwise you create aliasing
EDGE / BOUNDARY DETECTION

Szeliski 4.2
Edge detection

**Goal:** Identify visual changes (discontinuities) in an image.

Intuitively, semantic information is encoded in edges.

Think-pair-share:
What are some ‘causes’ of visual edges?

Source: D. Lowe
Origin of Edges

- Edges are caused by a variety of factors:
  - surface normal discontinuity
  - depth discontinuity
  - surface color discontinuity
  - illumination discontinuity

Source: Steve Seitz
Why do we care about edges?

Extract information
   – Recognize objects

Help recover geometry and viewpoint
Closeup of edges

Source: D. Hoiem
Closeup of edges
Closeup of edges

Source: D. Hoiem
Closeup of edges

Source: D. Hoiem
Characterizing edges

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

image

intensity function
(along horizontal scanline)

first derivative

edges correspond to extrema of derivative
Intensity profile

Source: D. Hoiem
With a little Gaussian noise

Intensity derivative

Source: D. Hoiem
Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

\[ f(x) \]

\[ \frac{d}{dx} f(x) \]

Where is the edge?

Source: S. Seitz
Effects of noise

• Difference filters respond strongly to noise
  – Image noise results in pixels that look very different from their neighbors
  – Generally, the larger the noise the stronger the response

• What can we do about it?

Source: D. Forsyth
Solution: smooth first

- To find edges, look for peaks in \( \frac{d}{dx}(f \ast g) \)

Source: S. Seitz
Derivative theorem of convolution

- Convolution is differentiable:
  \[
  \frac{d}{dx} (f \ast g) = f \ast \frac{d}{dx} g
  \]

- This saves us one operation:

Source: S. Seitz
Derivative of 2D Gaussian filter

\[ * [1\ -1] = \]
Tradeoff between smoothing and localization

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth
Think-Pair-Share

What is a good edge detector?

Do we lose information when we look at edges?

Are edges ‘complete’ as a representation of images?
Designing an edge detector

• Criteria for a good edge detector:
  – **Good detection**: the optimal detector should find all real edges, ignoring noise or other artifacts
  – **Good localization**
    • the edges detected must be as close as possible to the true edges
    • the detector must return one point only for each true edge point

• Cues of edge detection
  – Differences in color, intensity, or texture across the boundary
  – Continuity and closure
  – High-level knowledge

Source: L. Fei-Fei
Designing an edge detector

• “All real edges”
  • We can aim to differentiate later which edges are ‘useful’ for our applications.
  • If we can’t find all things which could be called an edge, we don’t have that choice.

• Is this possible?
Closeup of edges
What information would we need to ‘invert’ the edge detection process?
Elder – Are Edges Incomplete? 1999

Edge ‘code’:
- position,
- gradient magnitude,
- gradient direction,
- blur size.

Figure 8. Top left: Original image. Top right: Detected edge locations. Middle left: Intermediate solution to the heat equation. Middle right: Reconstructed luminance function. Bottom left: Reblurred result. Bottom right: Error map (reblurred result—original). Bright indicates overestimation of intensity, dark indicates underestimation. Edge density is 1.7%. RMS error is 10.1 grey levels, with a 3.9 grey level DC component, and an estimated 1.6 grey levels due to noise removal.
Where do humans see boundaries?

- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

pB slides: Hays
Results

Score = confidence of edge. For humans, this is averaged across multiple participants.
Results

Score = confidence of edge. For humans, this is averaged across multiple participants.
Score = confidence of edge. For humans, this is averaged across multiple participants.
Score = confidence of edge. For humans, this is averaged across multiple participants.

For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsdsm/108082-color.html
45 years of boundary detection

[Pre deep learning]

Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)
State of edge detection

Local edge detection works well
   – ‘False positives’ from illumination and texture edges (depends on our application).

Some methods to consider longer contours

Modern methods that actually “learn” from data.

Poor use of object and high-level information.
Summary: Edges primer

• Edge detection to identify visual change in image

• Derivative of Gaussian and linear combination of convolutions

• What is an edge? What is a good edge?
Canny edge detector

- Probably the most widely used edge detector in computer vision.

- Theoretical model: step-edges corrupted by additive Gaussian noise.

- Canny showed that first derivative of Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization.


22,000 citations!
Demonstrator Image

\texttt{rgb2gray('img.png')}
Canny edge detector

1. Filter image with $x$, $y$ derivatives of Gaussian

Source: D. Lowe, L. Fei-Fei
Derivative of Gaussian filter

$x$-direction

$y$-direction
Compute Gradients

X Derivative of Gaussian

Y Derivative of Gaussian

(x^2 + 0.5 for visualization)
Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient

Source: D. Lowe, L. Fei-Fei
Compute Gradient Magnitude

\[
\sqrt{(X\text{DerivOfGaussian} \, \cdot \, 2 + Y\text{DerivOfGaussian} \, \cdot \, 2)} = \text{gradient magnitude}
\]
Compute Gradient Orientation

- Threshold magnitude at minimum level
- Get orientation via $\theta = \text{atan2}(y\text{Deriv}, x\text{Deriv})$
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” to single pixel width
Non-maximum suppression for each orientation

At pixel q:
We have a maximum if the value is larger than those at both p and at r.

Interpolate along gradient direction to get these values.

Source: D. Forsyth
Before Non-max Suppression

Gradient magnitude (x4 for visualization)
After non-max suppression

Gradient magnitude (x4 for visualization)
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” to single pixel width
4. ‘Hysteresis’ Thresholding

Source: D. Lowe, L. Fei-Fei
‘Hysteresis’ thresholding

- Two thresholds – high and low
- Grad. mag. > high threshold? = strong edge
- Grad. mag. < low threshold? noise
- In between = weak edge

- ‘Follow’ edges starting from strong edge pixels
  - Continue them into weak edges
    - Connected components (Szeliski 3.3.4)

Source: S. Seitz
Final Canny Edges

\[ \sigma = \sqrt{2}, t_{low} = 0.05, t_{high} = 0.1 \]
Effect of \( \sigma \) (Gaussian kernel spread/size)

The choice of \( \sigma \) depends on desired behavior

- large \( \sigma \) detects large scale edges
- small \( \sigma \) detects fine features

Source: S. Seitz
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” to single pixel width
4. ‘Hysteresis’ Thresholding:
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
   - ‘Follow’ edges starting from strong edge pixels
     • Connected components (Szeliski 3.3.4)

• Python: e.g., skimage.feature.canny()

Source: D. Lowe, L. Fei-Fei
Sidebar: Bilinear Interpolation

\[ f(x, y) \approx \begin{bmatrix} 1 - x & x \end{bmatrix} \begin{bmatrix} f(0, 0) & f(0, 1) \\ f(1, 0) & f(1, 1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix}. \]

http://en.wikipedia.org/wiki/Bilinear_interpolation
Sidebar: Interpolation options

e.g., skimage.transform.rescale(1, 2, order=x)

x == 0 -> ‘nearest neighbor’
  – Copy value from nearest known
  – Very fast but creates blocky edges

x == 1 -> ‘bilinear’ (default)
  – Weighted average from four nearest known pixels
  – Fast and reasonable results

x == 3 -> ‘bicubic’
  – Fit cubic spline to pixel intensities
  – Non-linear interpolation over larger area (4x4)
  – Slower, visually appealing, may create negative pixel values in cubic function fitting

Canny edge demo!!!
From Luke Murray (Fall 2017 TA)


• Written in [https://idyll-lang.org/](https://idyll-lang.org/)