I, Robot
Isaac Asimov
1950
Future Vision

Eye Robot
CSCI
1430
2017 MWF 1pm
Computer Vision
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 fft (fft2 in matlab)
    2. Pointwise-multiply ffts
    3. Convert result to spatial domain with ifft2

Did anyone play with the code?
Review of Filtering

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

\[
[-1 \ 1]
\]

<table>
<thead>
<tr>
<th>FFT of Gradient Filter</th>
<th>FFT of Gaussian</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="FFT of Gradient Filter" /></td>
<td><img src="image2.png" alt="FFT of Gaussian" /></td>
</tr>
</tbody>
</table>

Gaussian

Hays
More Useful Filters

Single Gaussian

1st Derivative

2nd Derivative (Laplacian of Gaussian)

1st Derivative of Gaussian

Earl F. Glynn
Things to Remember

• Sometimes it makes sense 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 auto-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.

• What are some ‘causes’ of visual edges?

Source: D. Lowe
Origin of Edges

- Edges are caused by a variety of factors

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

Source: D. Hoiem
Closeup of edges

Source: D. Hoiem
Closeup of edges
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
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 $\frac{d}{dx} (f * g)$
Derivative theorem of convolution

- Convolution is differentiable:

\[
\frac{d}{dx} (f * g) = f * \frac{d}{dx} g
\]

- This saves us one operation:

\[
\frac{d}{dx} f * g
\]
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 on 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.

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 boundary detector

Figure from Fowlkes

Martin, Fowlkes, Malik 2004: Learning to Detect Natural Boundaries…

Figure from Fowlkes
pB Boundary Detector

Figure from Fowlkes
Results

Human (0.95)
Pb (0.88)
Results

Human Pb (0.96)

Global Pb (0.88)
Human (0.90)

Pb (0.35)

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

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 take into account 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

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

\[ \text{x-direction} \quad \text{y-direction} \]
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

\[
\sqrt{ (X\text{DerivOfGaussian} \cdot \Delta x)^2 + (Y\text{DerivOfGaussian} \cdot \Delta y)^2 } = \text{gradient magnitude}
\]

(x4 for visualization)
Compute Gradient Orientation

- Threshold magnitude at minimum level
- Get orientation via $\theta = \text{atan2}(gy, gx)$
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

Source: D. Lowe, L. Fei-Fei
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)
Final Canny Edges

\[ \sigma = \sqrt{2}, t_{\text{low}} = 0.05, t_{\text{high}} = 0.1 \]
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)

• MATLAB: edge(image, ‘canny’)

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

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

Sidebar: Interpolation options

• imx2 = imresize(im, 2, interpolation_type)

• ‘nearest’
  – Copy value from nearest known
  – Very fast but creates blocky edges

• ‘bilinear’
  – Weighted average from four nearest known pixels
  – Fast and reasonable results

• ‘bicubic’ (default)
  – Non-linear smoothing over larger area (4x4)
  – Slower, visually appealing, may create negative pixel values