Edge Detection

Lu Sheng (盛律) Spring 2024

Previous lectures

- We are now touched on
  - Image formation
  - Image processing
- Now we are moving on to
  - Feature detection and matching

The big picture ...

Feature Detection

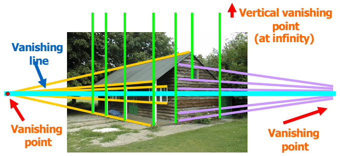
Feature Description

Matching
Indexing
Detection
...

Edge detection

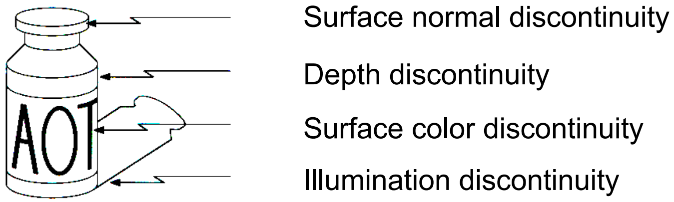
- **Goal:** Identify visual changes (discontinuities) in an image
- **Ideal output:** Artist's line drawing (but artist is also using object-level knowledge)

Why do we care about edges?



- Extract information
  - Recognize objects
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- Help recover geometry and viewpoint

Origin of edges



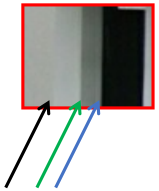
- Edges are caused by a variety of factors

Closeup of edges

There are a lot of edges with various causes

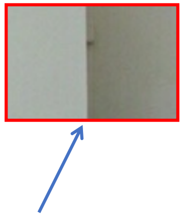
Closeup of edges

There are a lot of edges with various causes



Closeup of edges

There are a lot of edges with various causes

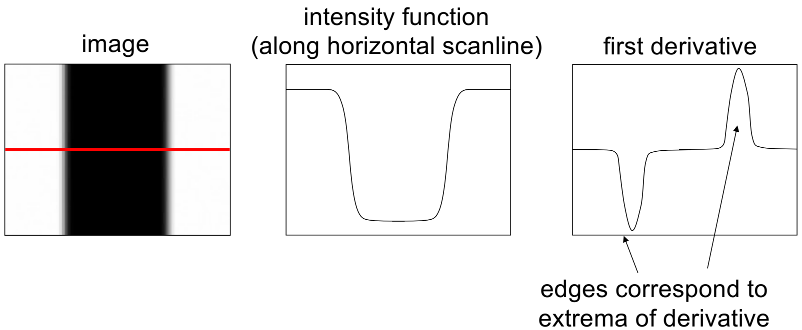


Closeup of edges

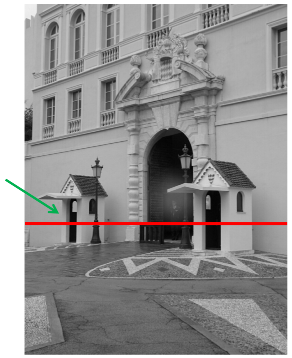
There are a lot of edges with various causes

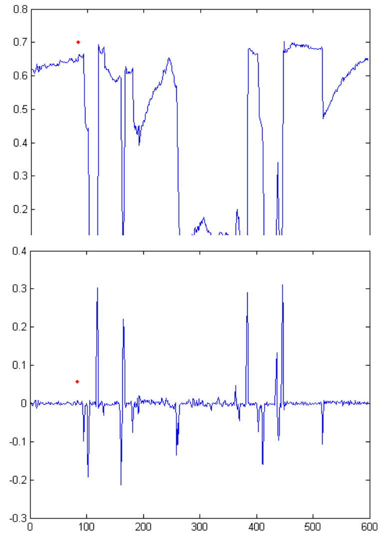
Characterizing edges

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

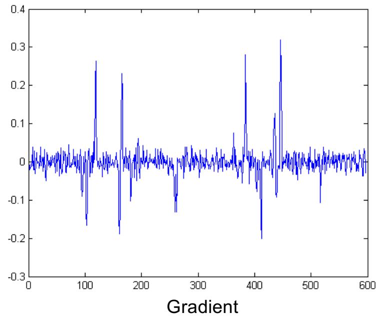


Intensity profile



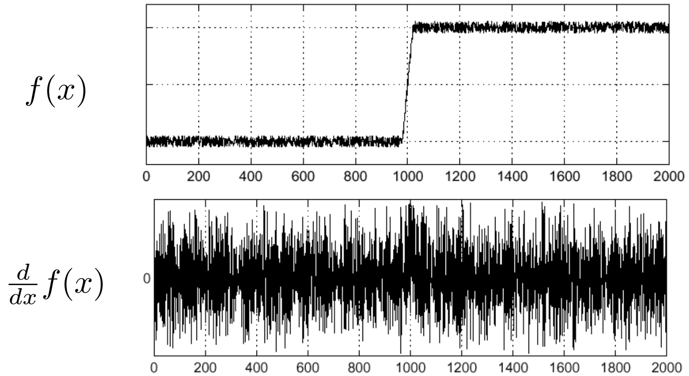


But with a little Gaussian noise



Effects of noise

- Consider a single row or column of the image

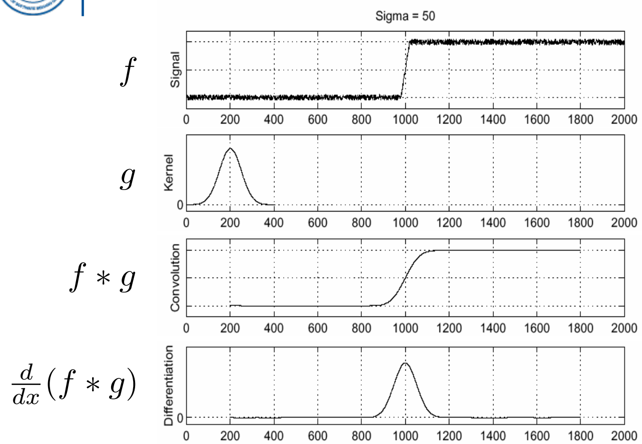


Effects of noise

- 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?

Solution: smooth first



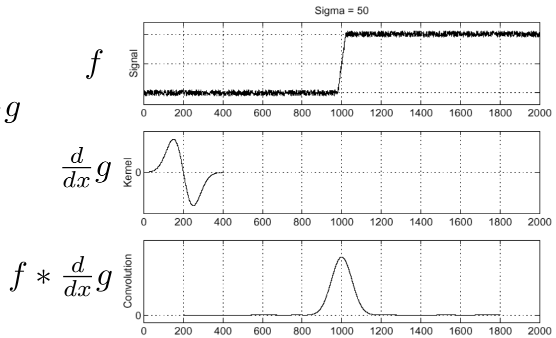
To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Derivative theorem of convolution

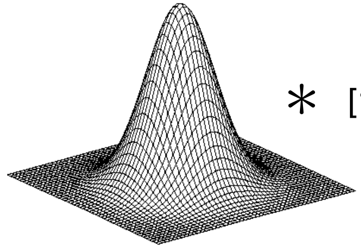
- Convolution is differentiable:

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

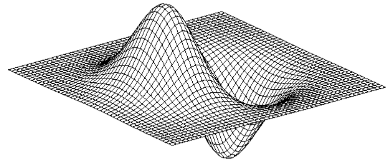
- This saves us one operation



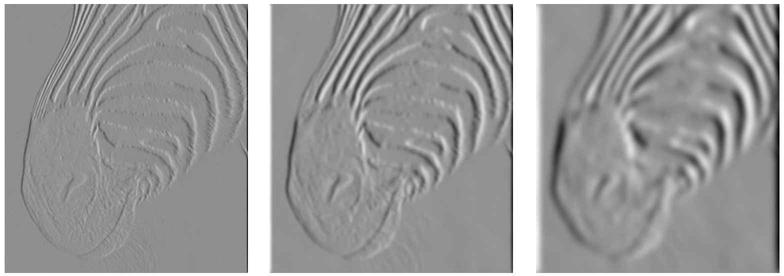
Derivative of 2D Gaussian filter



$*$ [1 -1] =



Trade-off between smoothing and localization



1 pixel 3 pixels 7 pixels

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

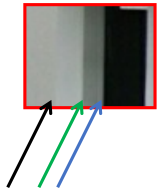
More problems...

- The gradient magnitude is large along a thick "trail" or "ridge", so how do we identify the actual edge points?
- How do we link the edge points to form curves?

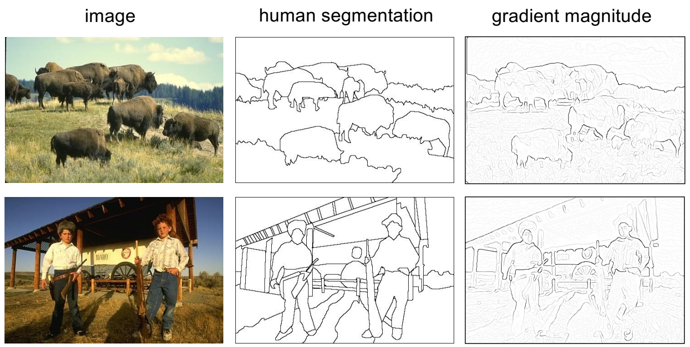
Designing an edge detector

- 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

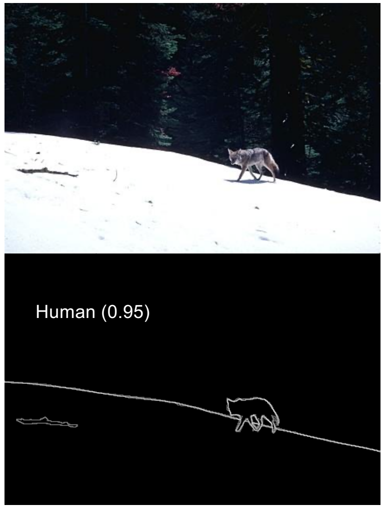
What are real edges?



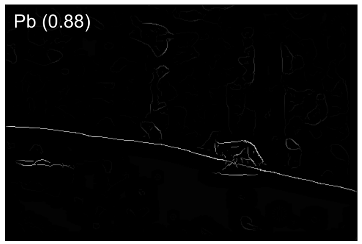
Where do humans see boundaries?

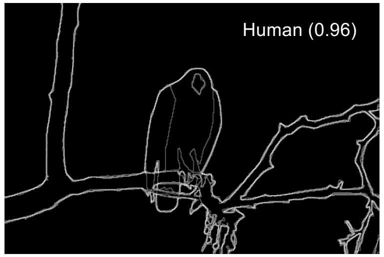


- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

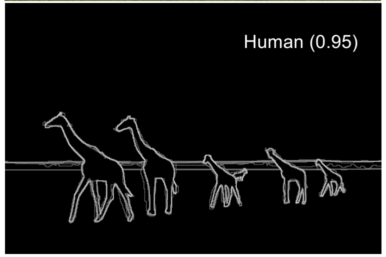


- Score = confidence of edge
- For humans, this is averaged across multiple participants

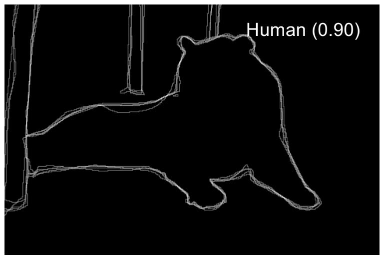




- Score = confidence of edge
- For humans, this is averaged across multiple participants

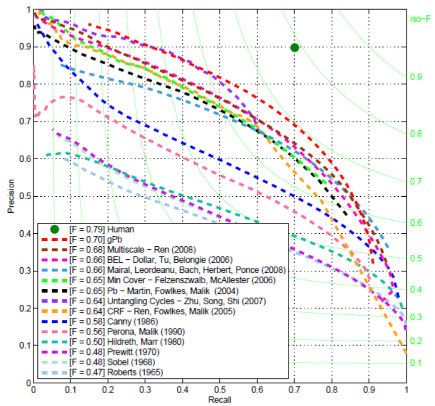


- Score = confidence of edge
- For humans, this is averaged across multiple participants



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

45 years of boundary detection



Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)

State of edge detection

- Local edge detection works well
  - But ‘False positives’ from illumination and texture edges
- Some methods consider longer contours
  - But could probably do better
- Modern methods that learn from data
- Poor use of object and high-level information

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

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

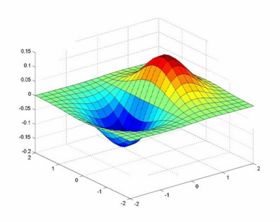
Citation: 35420!

Example

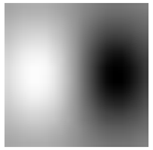
Canny edge detector

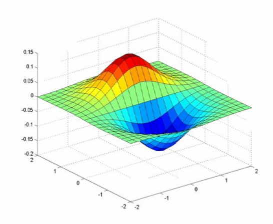
1. Filter image with x, y derivatives of Gaussian

Derivative of Gaussian filter

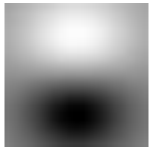


x-direction

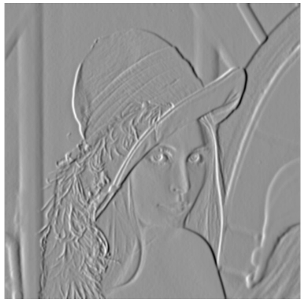




y-direction



Compute Gradients (DoG)



X-Derivative of Gaussian

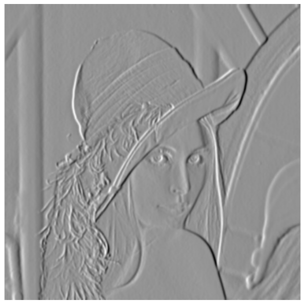
Y-Derivative of Gaussian

Canny edge detector

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

Compute gradient magnitude

$$\sqrt{\mathrm{DoG}_x(I)^2 + \mathrm{DoG}_y(I)^2} = \text{gradient magnitude}$$

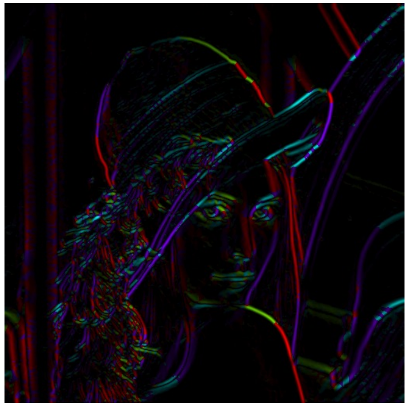


X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

Compute gradient orientation



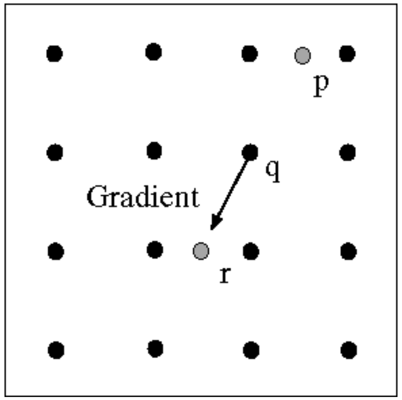
- Threshold at minimum level
- Get orientation via

$$\theta = \tan^{-1} \frac{\mathrm{DoG}_y I}{\mathrm{DoG}_x I}$$

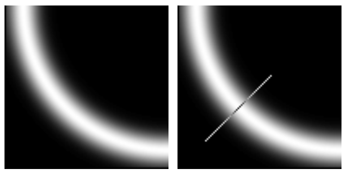
Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradients
3. Non-maximum suppression:
   - Thin multi-pixel wide ‘ridges’ to single pixel width

Non-maximum suppression for each orientation

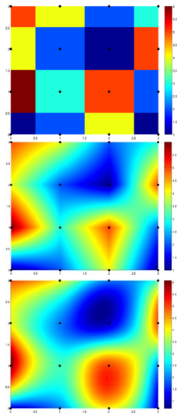


At *q*, we have a maximum if the value is larger than those at both *p* and at *r*. Interpolate to get these values.

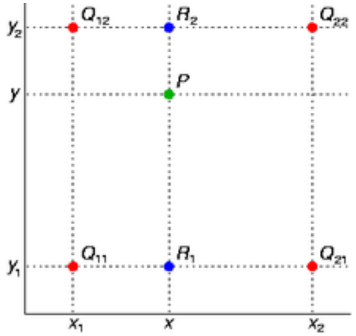


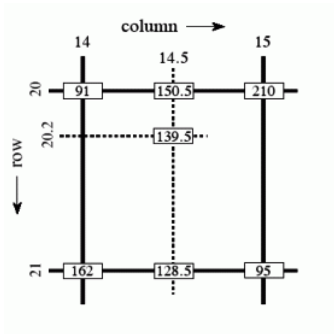
Sidebar: interpolation options

- `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`
  - Non-linear smoothing over larger area (4x4)
  - Slower, visually appealing, may create negative pixel values



Sidebar: bilinear interpolation





$$f(x,y) \simeq \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}$$

http://en.wikipedia.org/wiki/Bilinear_interpolation

Before non-maximum suppression

After non-maximum suppression

Canny edge detector

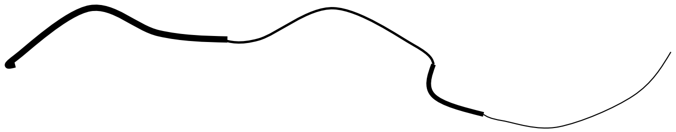
1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradients
3. Non-maximum suppression:
   - Thin multi-pixel wide ‘ridges’ to single pixel width
4. Thresholding and linking

Thresholding and linking

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Thresholding and linking

- 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)



Find canny edges

Canny edge detector

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

Effect of σ (Gaussian kernel size)

Original | $\sigma = \sqrt{2}$ | $\sigma = 4\sqrt{2}$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Questions?

- Edge detection: goal and ideal output
- Characteristics of edge
- What is a good edge detector
- Canny edge detector