COMP322: Computer Vision & Image Processing

Edge Detection

Image Credit: Pixar Animation Studios

Announcements

- HW 1 Questions due this Thursday, 10:30 AM
- Broader Impact Discussion
  - Immediately after class today
- Workshop
  - Location: CIT 101
- Lecture notes
  - Scribe volunteers

COMP322: Computer Vision & Image Processing

Last lecture

- Fourier Series and Transform
- Fourier Transform Properties
- Convolution Theorem

Image Credit: Pixar Animation Studios

COMP322: Computer Vision & Image Processing

This lecture

- Edge Detection

Image Credit: Pixar Animation Studios

Edge Detection

https://towardsdatascience.com/edge-detection-in-python-a3c263a13e03

Edge Detection

Definition

The process of identifying parts of a digital image with sharp changes (discontinuities) in image intensity.

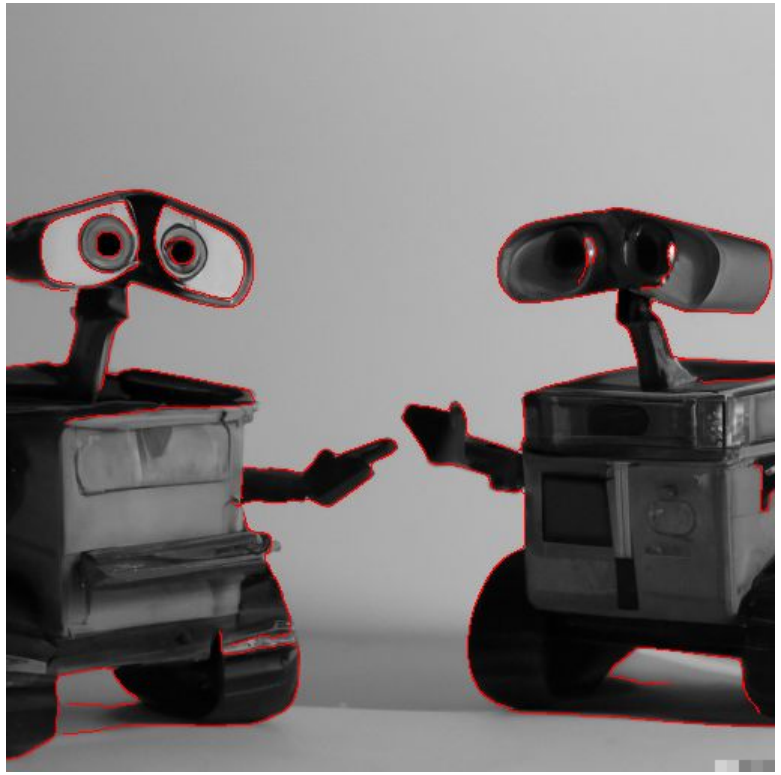
Beginning to extract information.

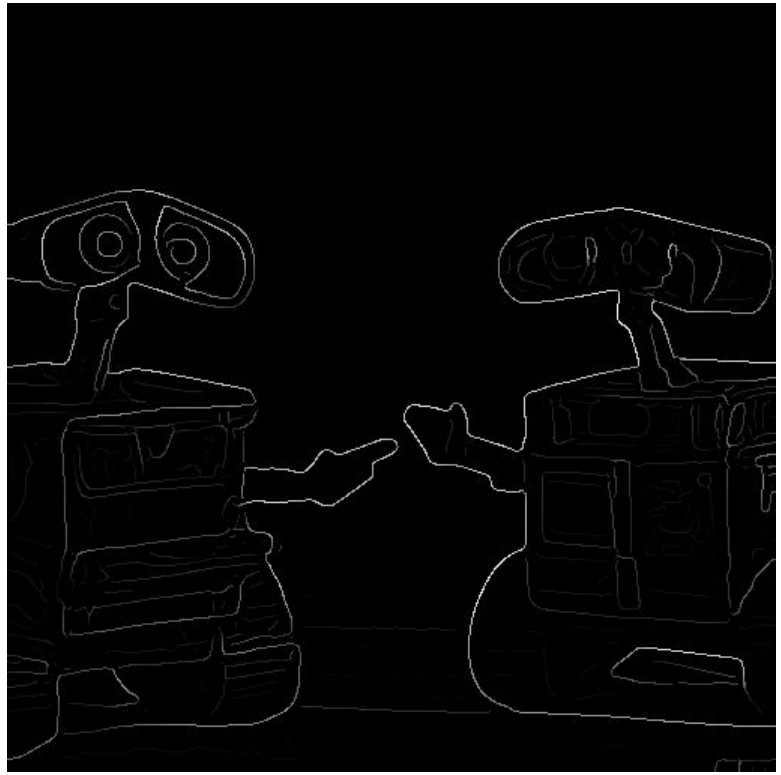
Helpful to

- Recognize objects
- Reconstruct scenes
- Edit images (artistically)

Question:

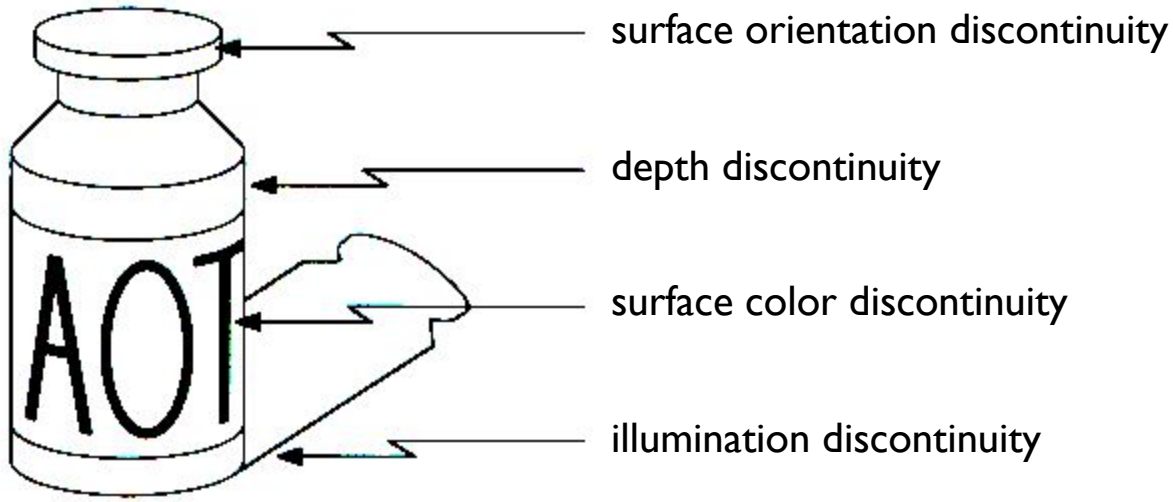
What natural phenomena produce image edges?





Dall-E: "wall-e having a discussion with another wall-e". http://bigwww.epfl.ch/demo/ip/demos/edgeDetector

Causes of Edges

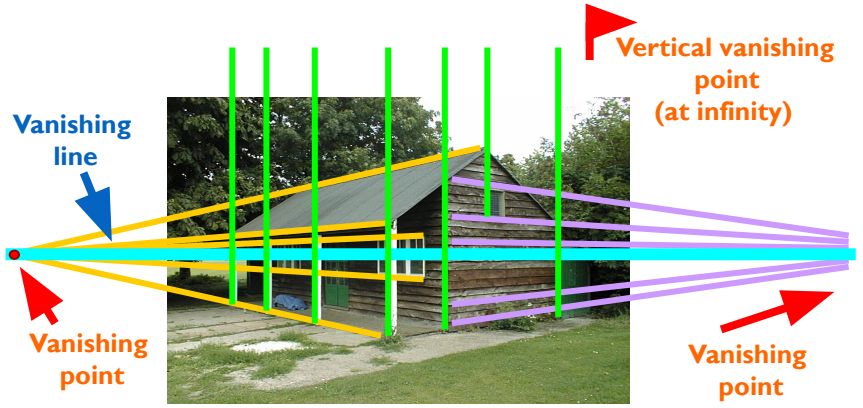


Edges are caused by a variety of factors

Use of Edges

- Extract information
  - Recognize objects
  - Reconstruct scenes

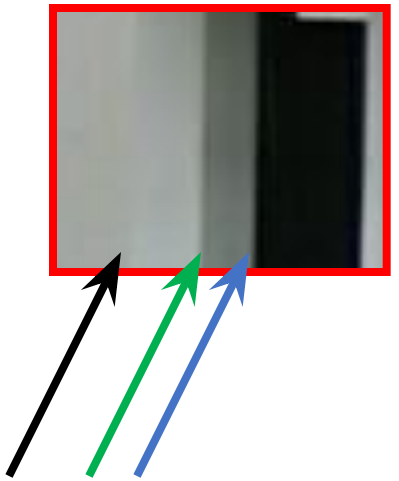
- Help recover geometry and viewpoint



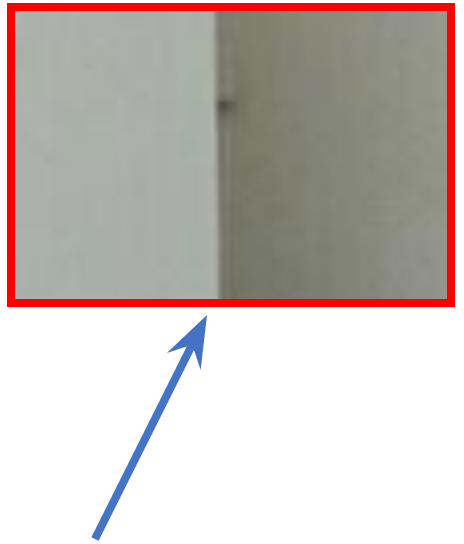
- Edit images directly
  - Artistic effects

Closer Look at Edges

Closer Look at Edges



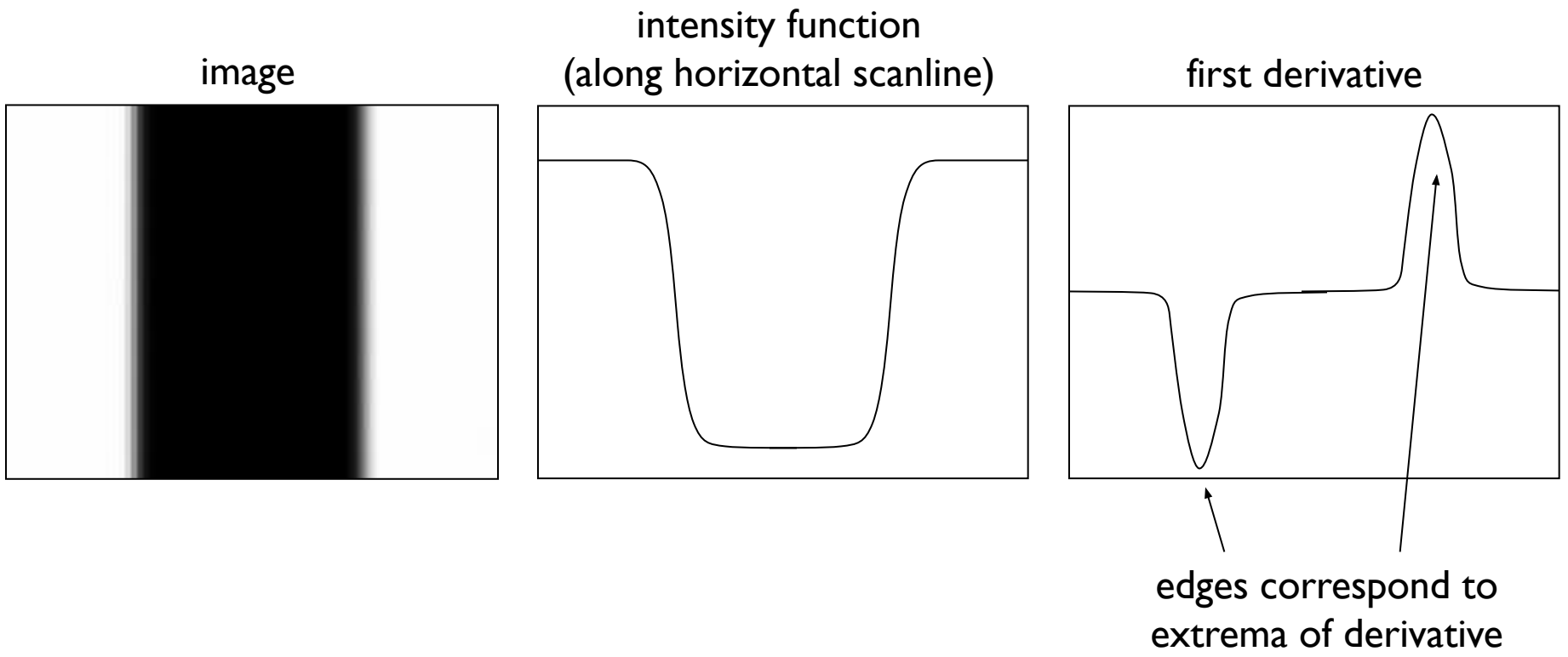
Closer Look at Edges



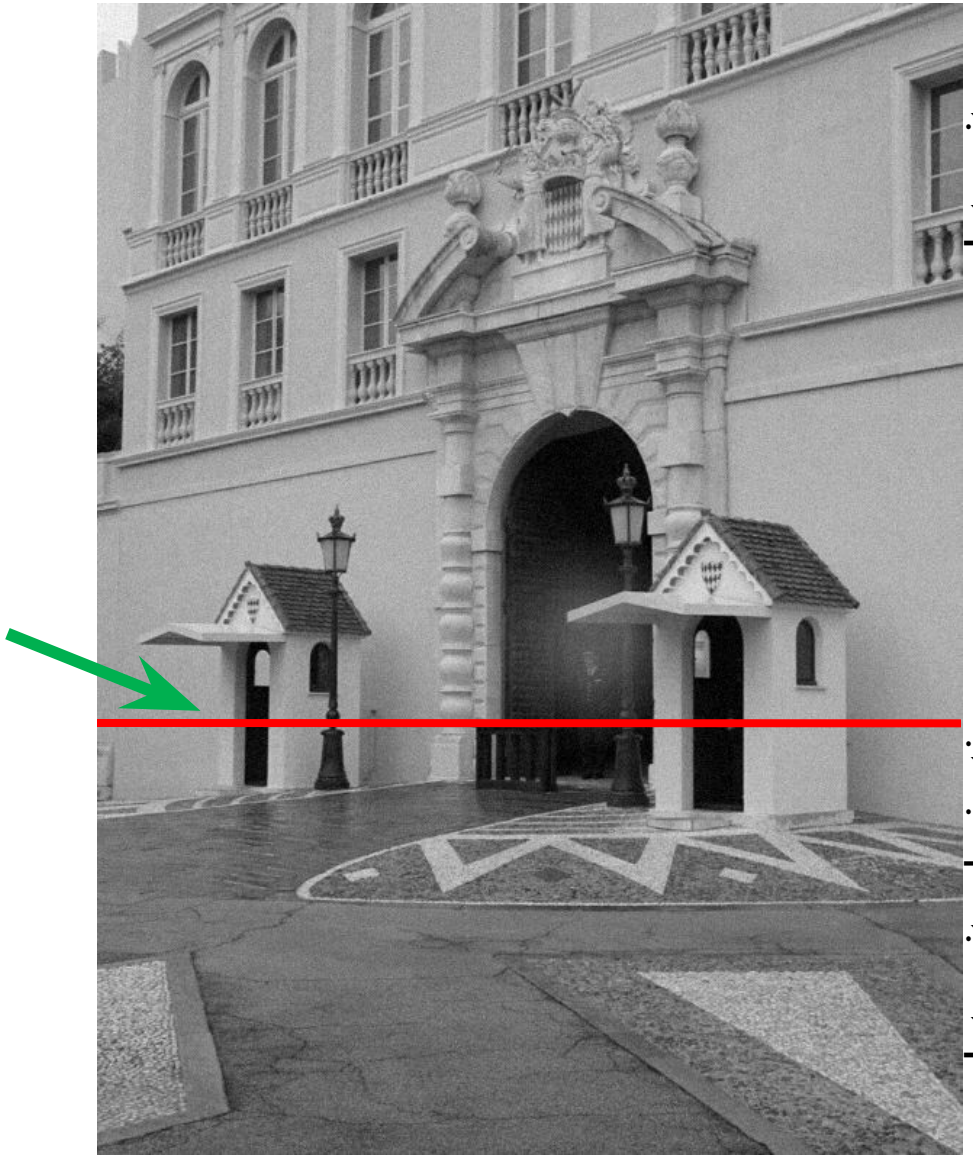
Closer Look at Edges

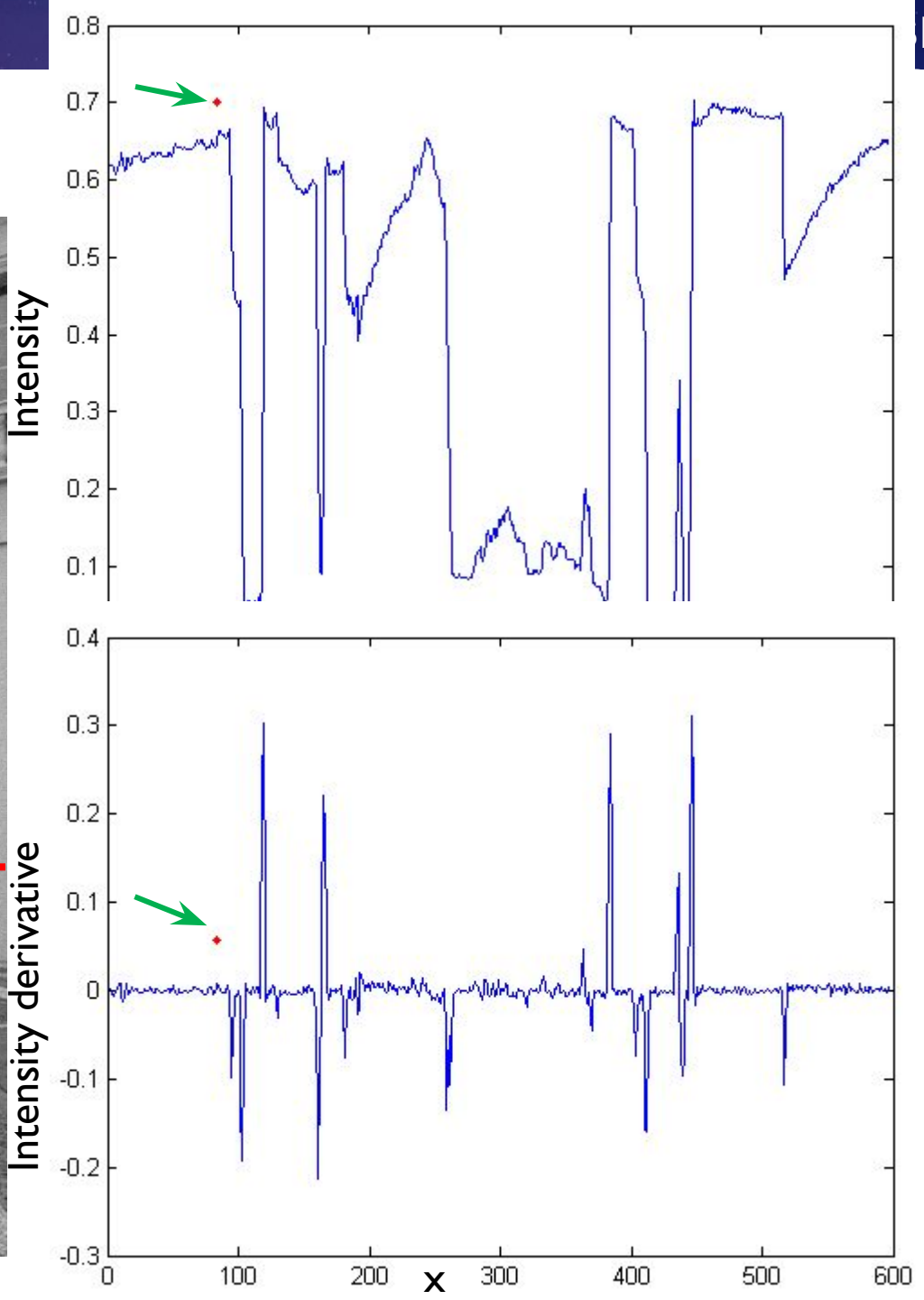
Characterizing Edges

An edge is a place of sharp change (discontinuity) in the image intensity function. Simple algorithm for edge detection: find gradient.

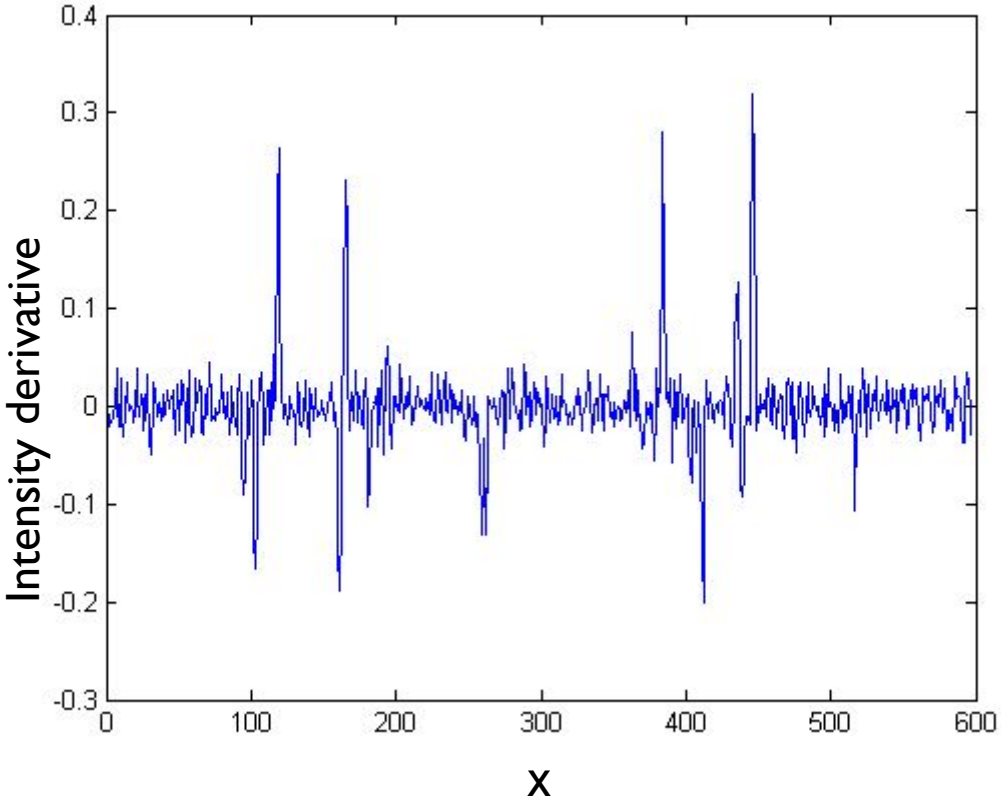


Intensity Profile





Adding Gaussian noise

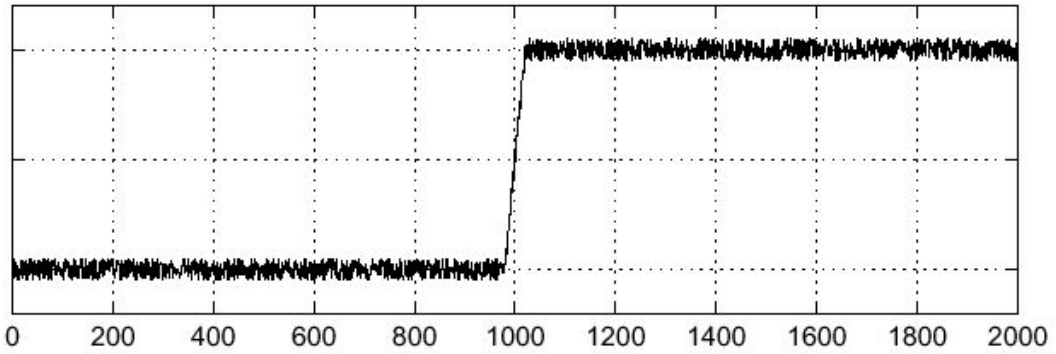


Effects of Noise

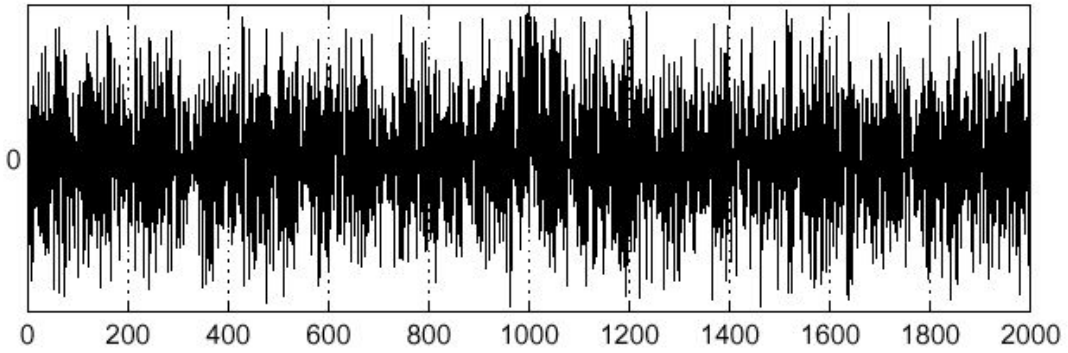
Consider a noisy signal

- Here's how the derivative looks like

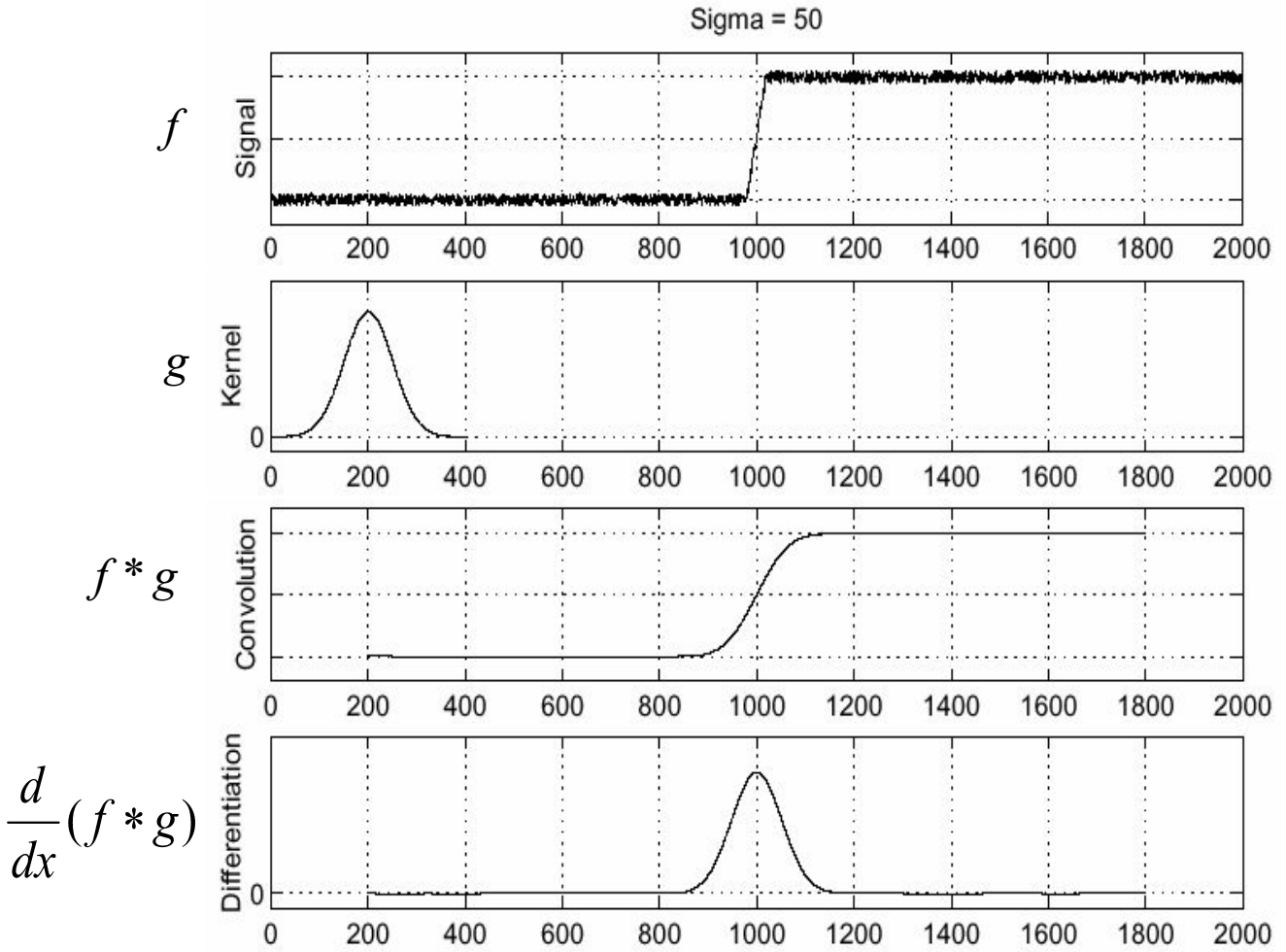
$f(x)$



$\frac{d}{dx}f(x)$



Solution: Smoothing Filter



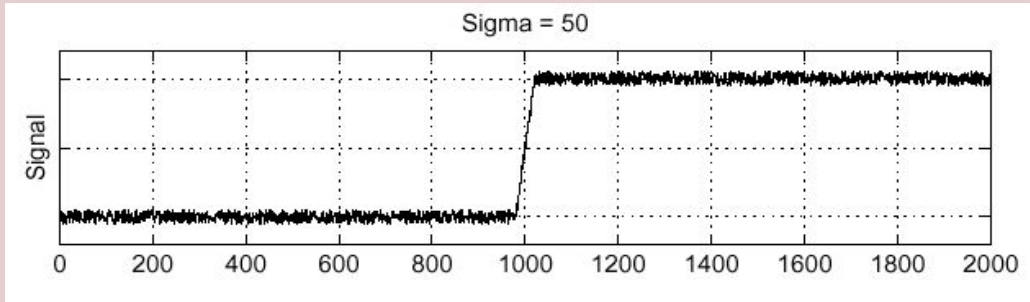
To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Derivative Theorem of Convolution

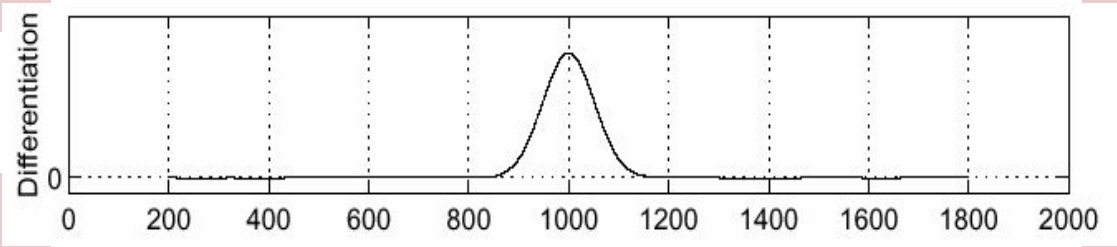
Theorem

Convolution is differentiable: $\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$

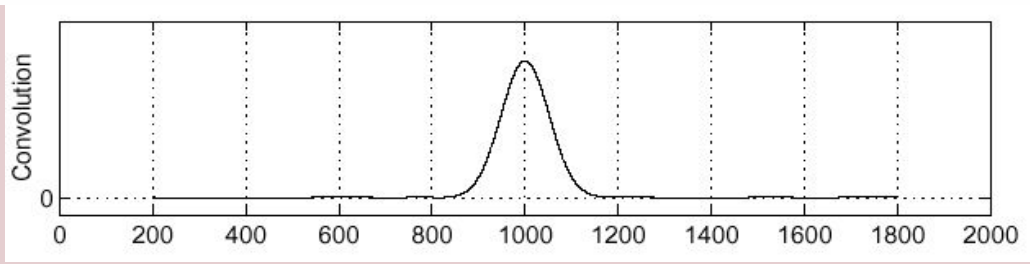
f



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



$f * \frac{d}{dx} g$



Derivatives in Practice

Digital images are discrete signals

Derivative $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$.

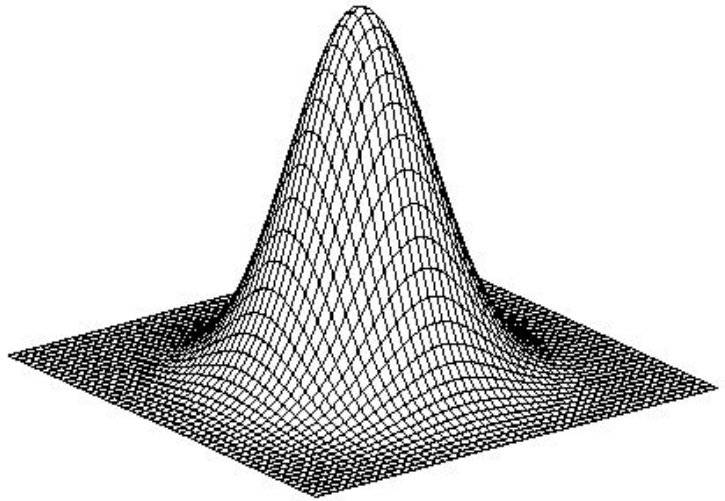
We approximate the true derivative with finite differences.
The smallest sampled 'h' in images is 1 (one pixel).

$$f'(x) \approx f(x+1) - f(x).$$

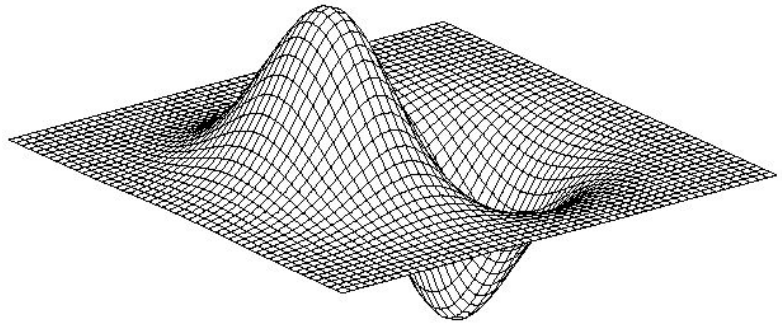
$$f'(x) \approx f(x-1) - f(x+1).$$

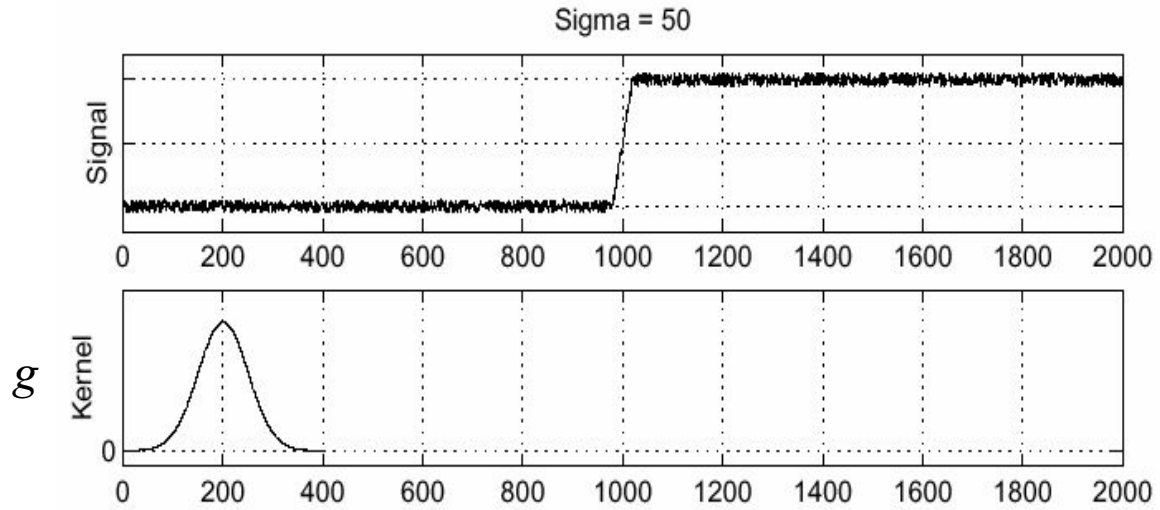
Symmetric form produces derivative estimate at x exactly, rather than at the border between (x+1) and x as above.

Derivative of 2D Gaussian Filter



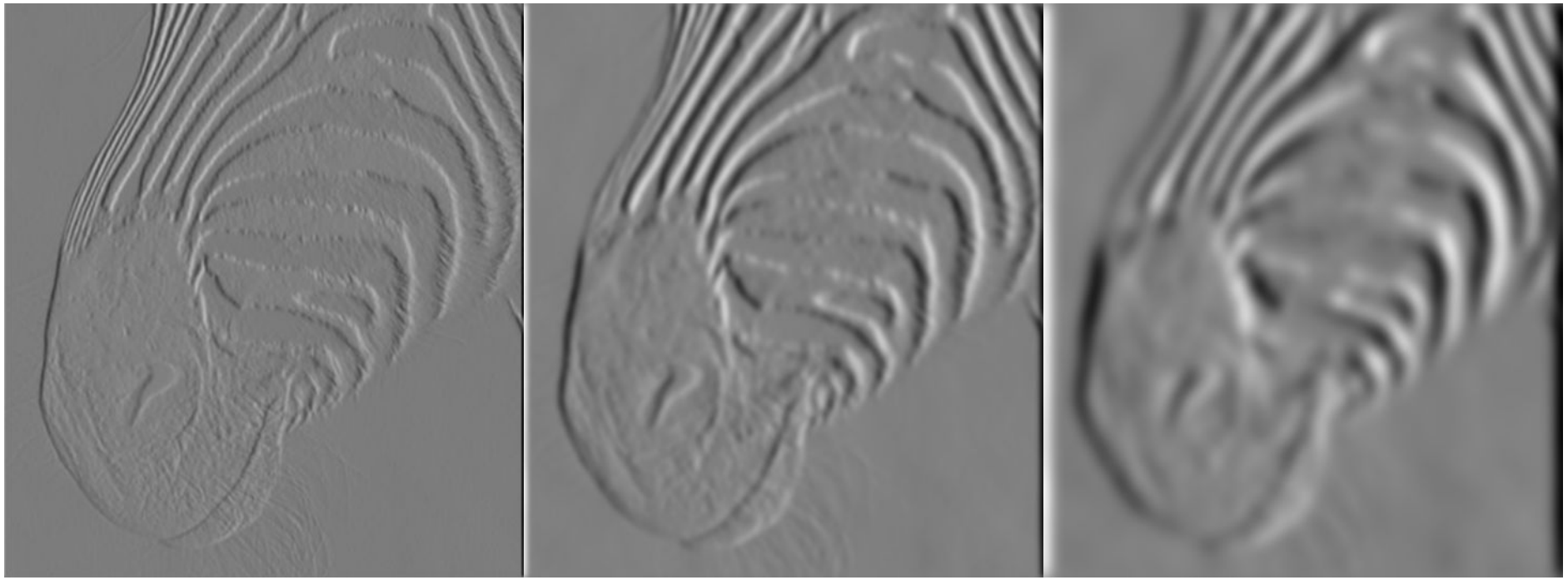
* [1 -1] =





Smoothing-Localization Tradeoff

Smoothed derivative removes noise, but blurs edge.
Also finds edges at different frequencies.



1 pixel

3 pixels

7 pixels

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

Designing an Edge Detector

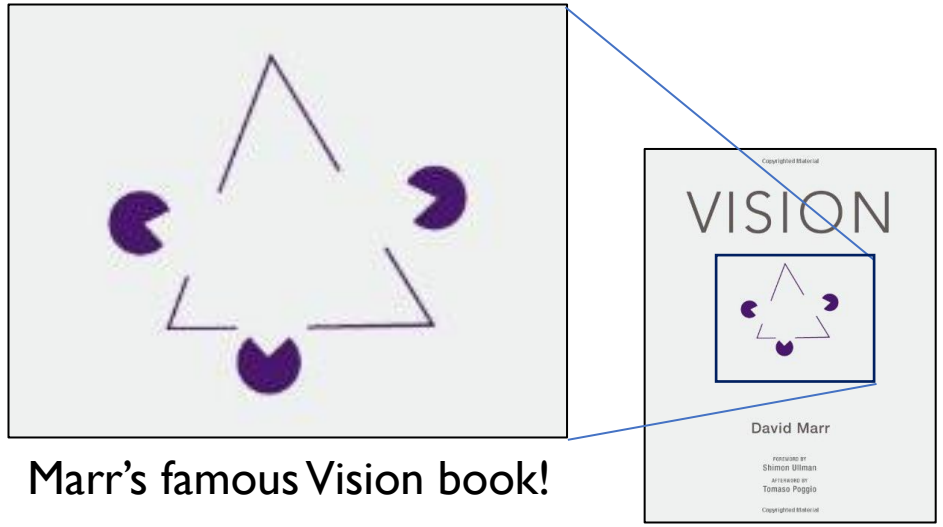
"All real edges"

- We can aim to differentiate later which edges are 'useful' for our applications.

- Our perceptual system can complete shapes even when no edge exists.

- *No correct answer*

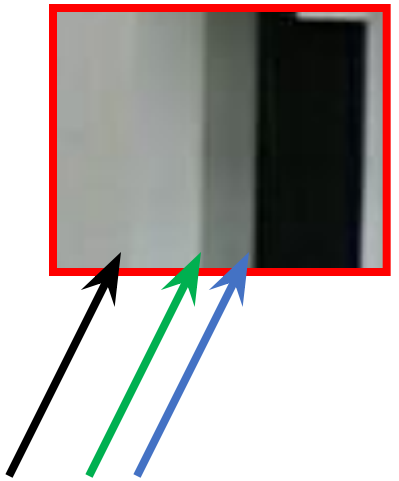
"Edges are imposed, not detected."
- Koenderink



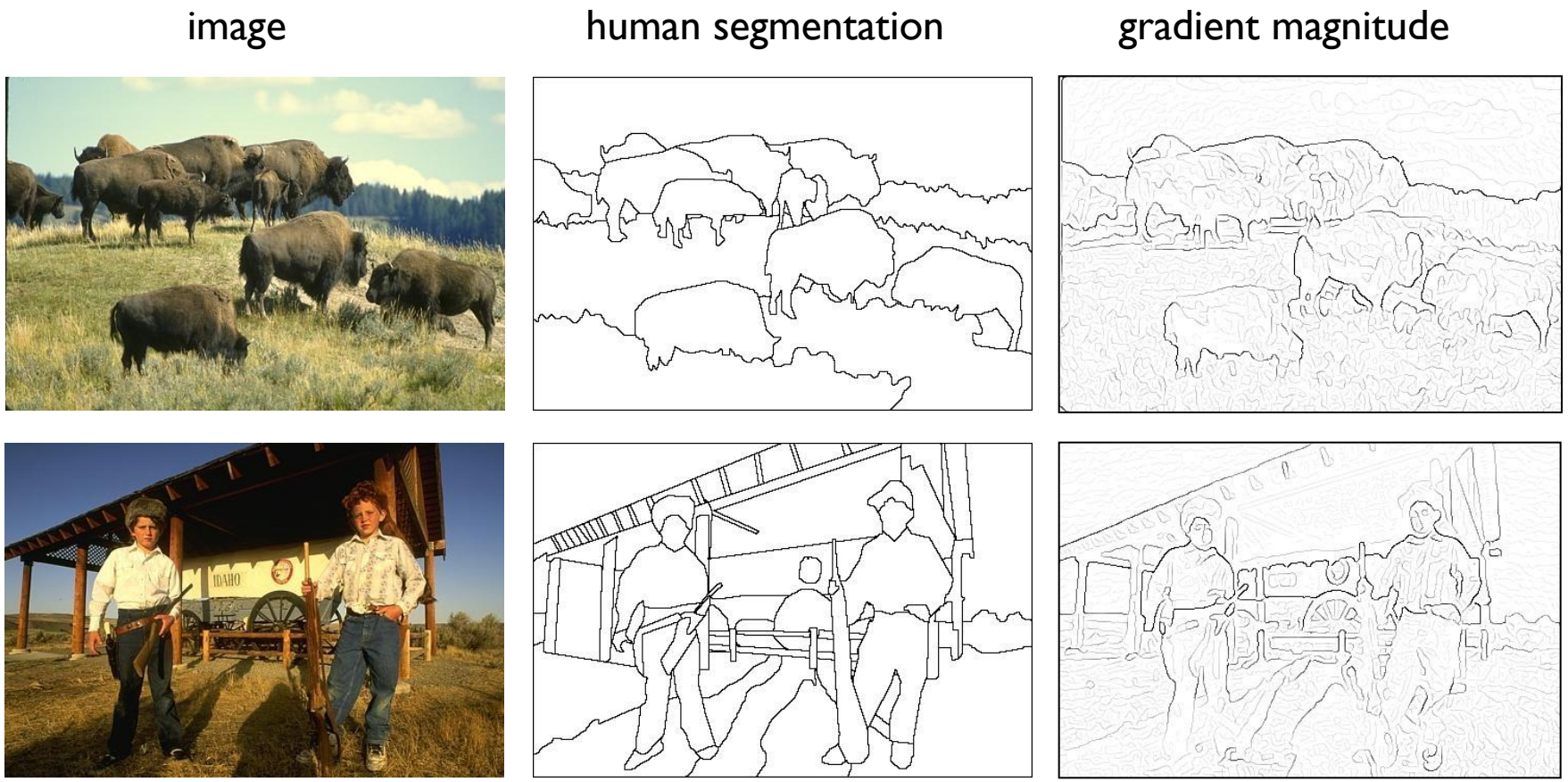
Marr's famous Vision book!

Closer Look at Edges

Derek Hoiem



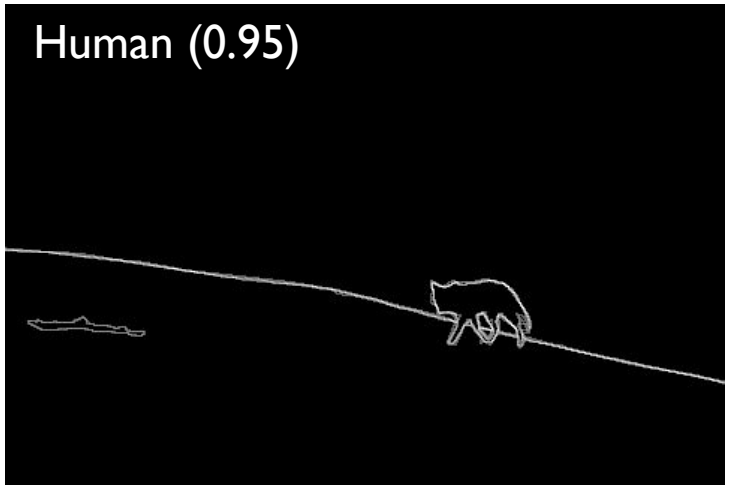
Where do humans see boundaries?

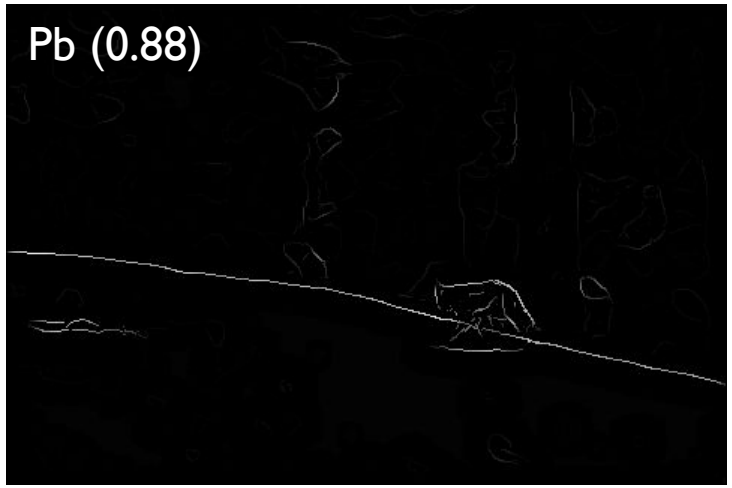


James Hays. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

pB Boundary Detector vs. Humans

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

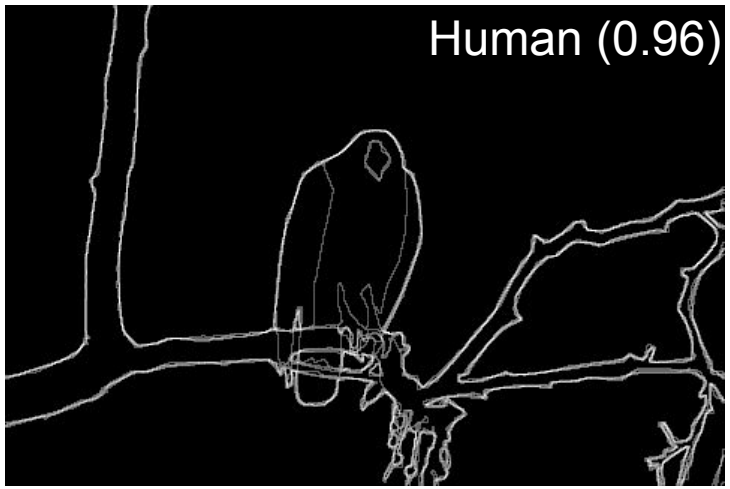




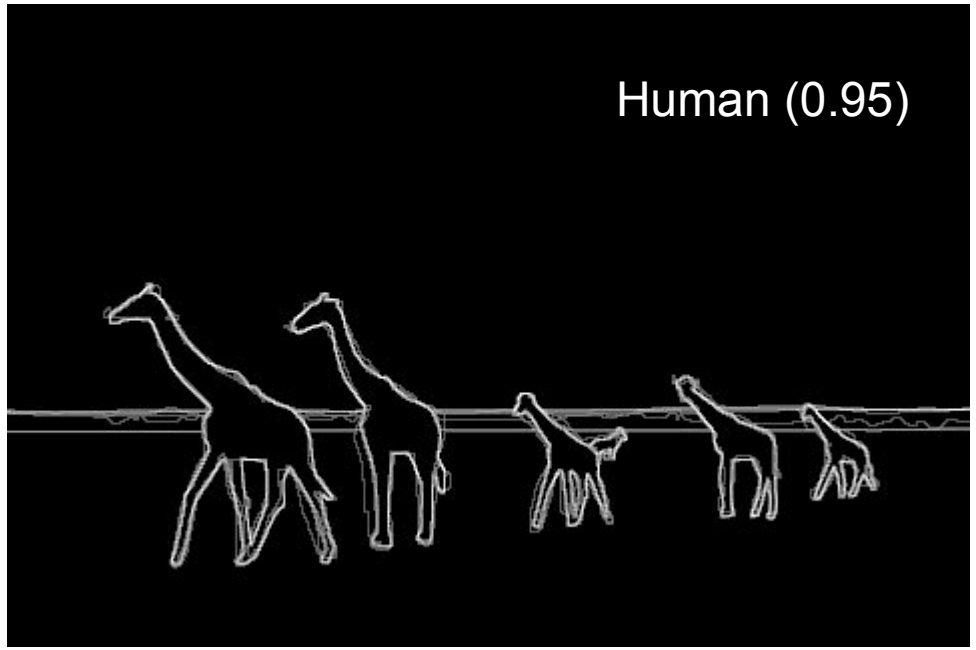
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/mfm-pami-boundary.pdf

pB Boundary Detector vs. Humans

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



http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/mfm-pami-boundary.pdf

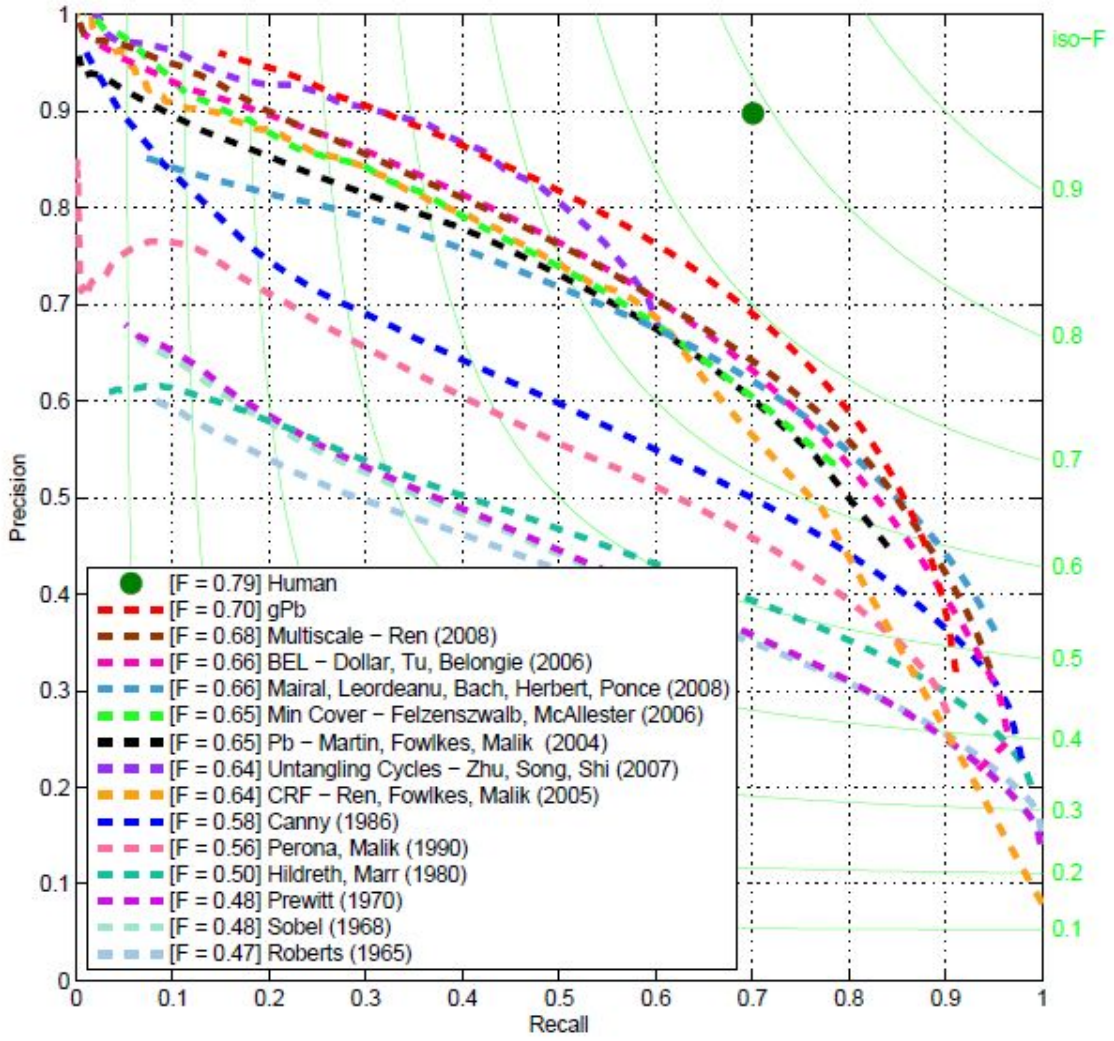


Score = confidence of edge.
For humans, this is averaged across multiple participants.

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

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

45 Years of Boundary Detection*



Arbelaez, Maire, Fowlkes, and Malik, TPAMI 2011.

* Pre-deep learning

Image Editing

(a)

(b)

(c)

Image Editing in the Contour Domain, TPAMI 2001.

Image Editing

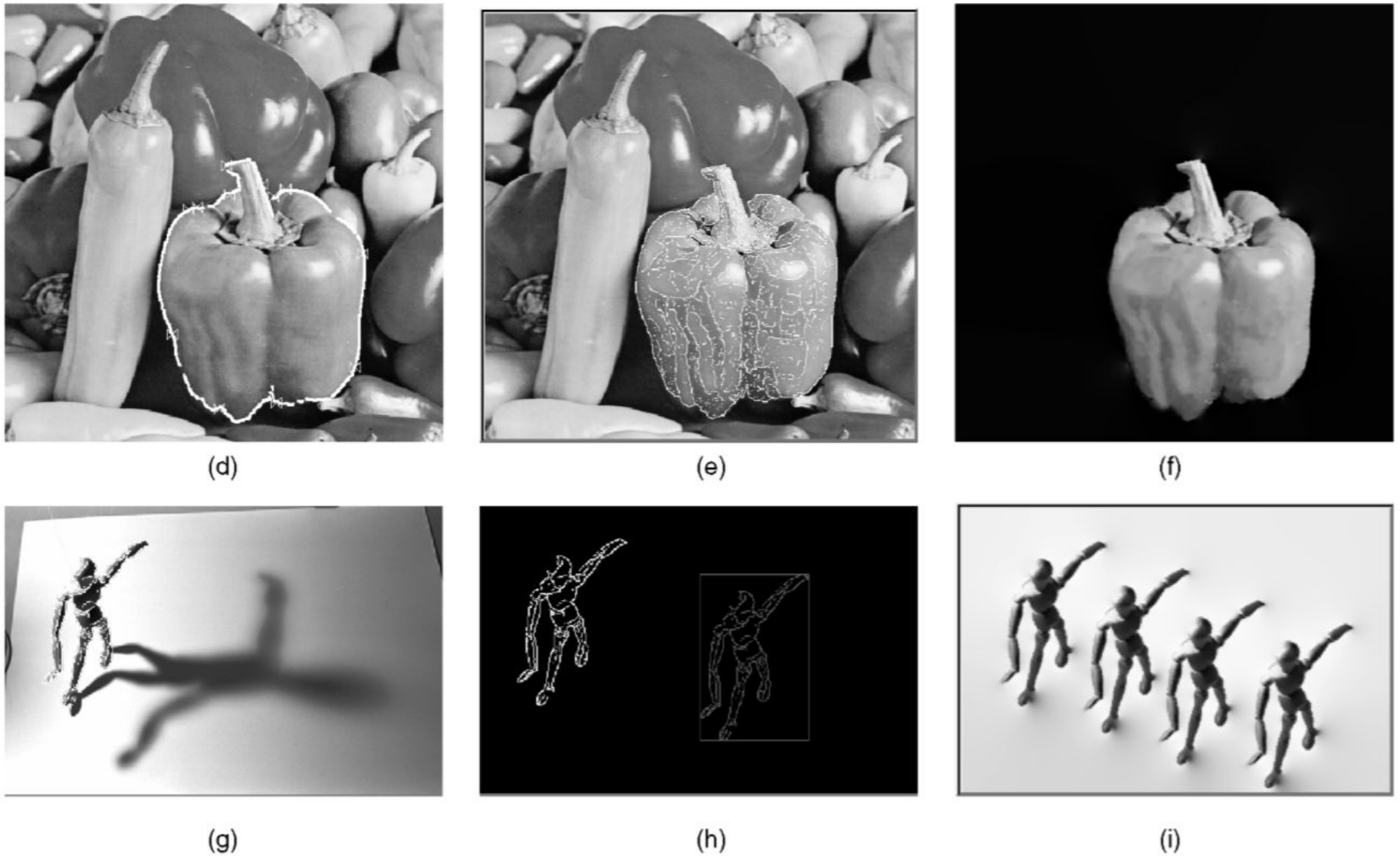


Image Editing in the Contour Domain, TPAMI 2001.

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

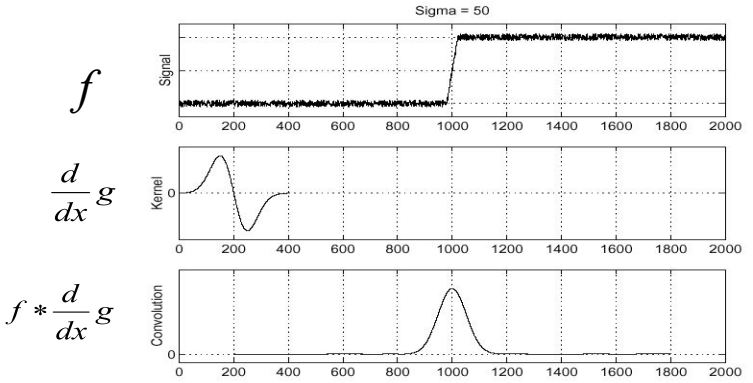
Modern methods that "learn" from data.

Poor use of object and high-level information.

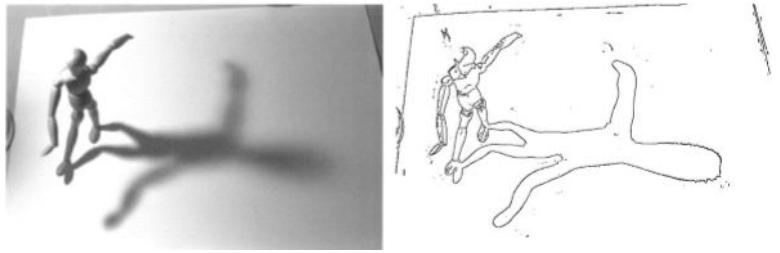
Summary

- Edge detection to identify sharp changes in image intensity

- Edges help with three Rs



- Derivative of Gaussian and linear combination of convolutions



- Properties of good edges

A Better Edge Detector

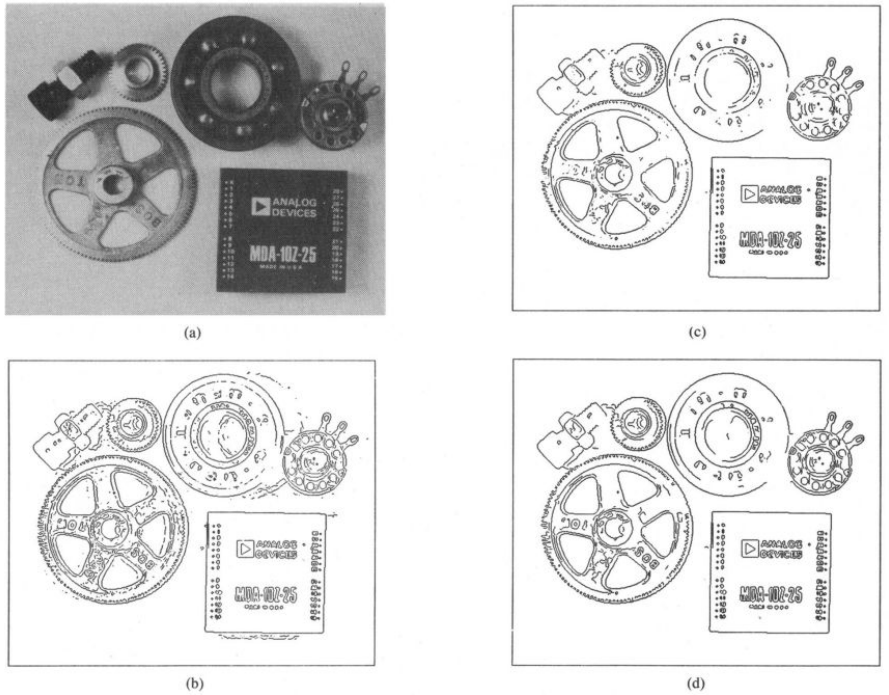
- The gradient magnitude is large along a thick "trail" or "ridge," so how do we identify the actual edge points?

- How do we link the edge points to form curves?

David Forsyth

Canny Edge Detector

Probably the most widely used edge detector in computer vision.



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

40,530 citations!

Demonstrator Image

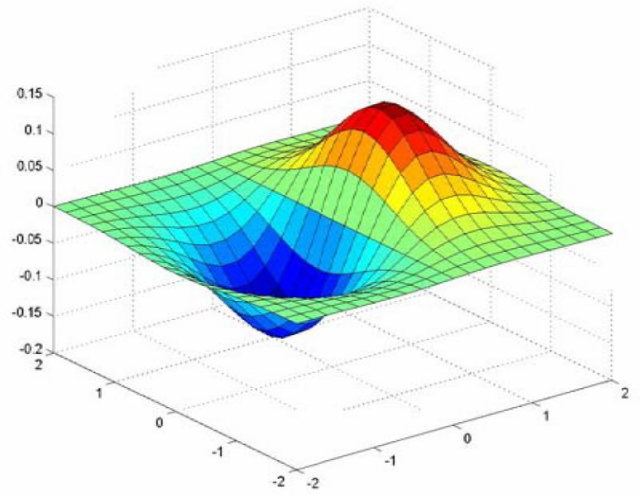
`rgb2gray('img.png')`

Canny Edge Detector

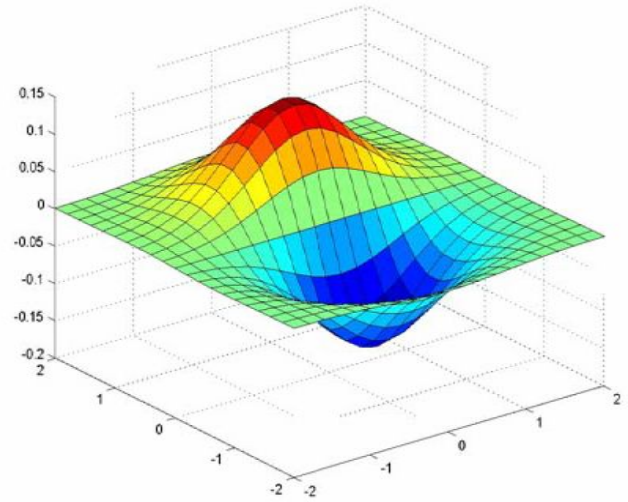
Algorithm

1. Filter image with x, y derivatives of Gaussian

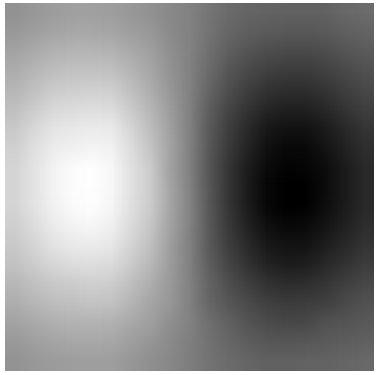
Derivative of Gaussian filter

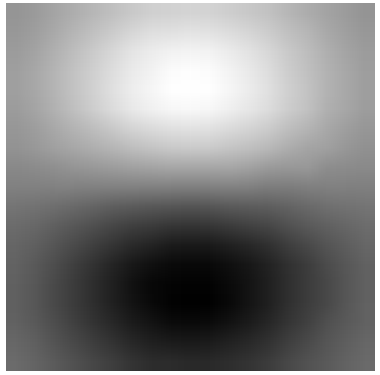


x-direction



y-direction

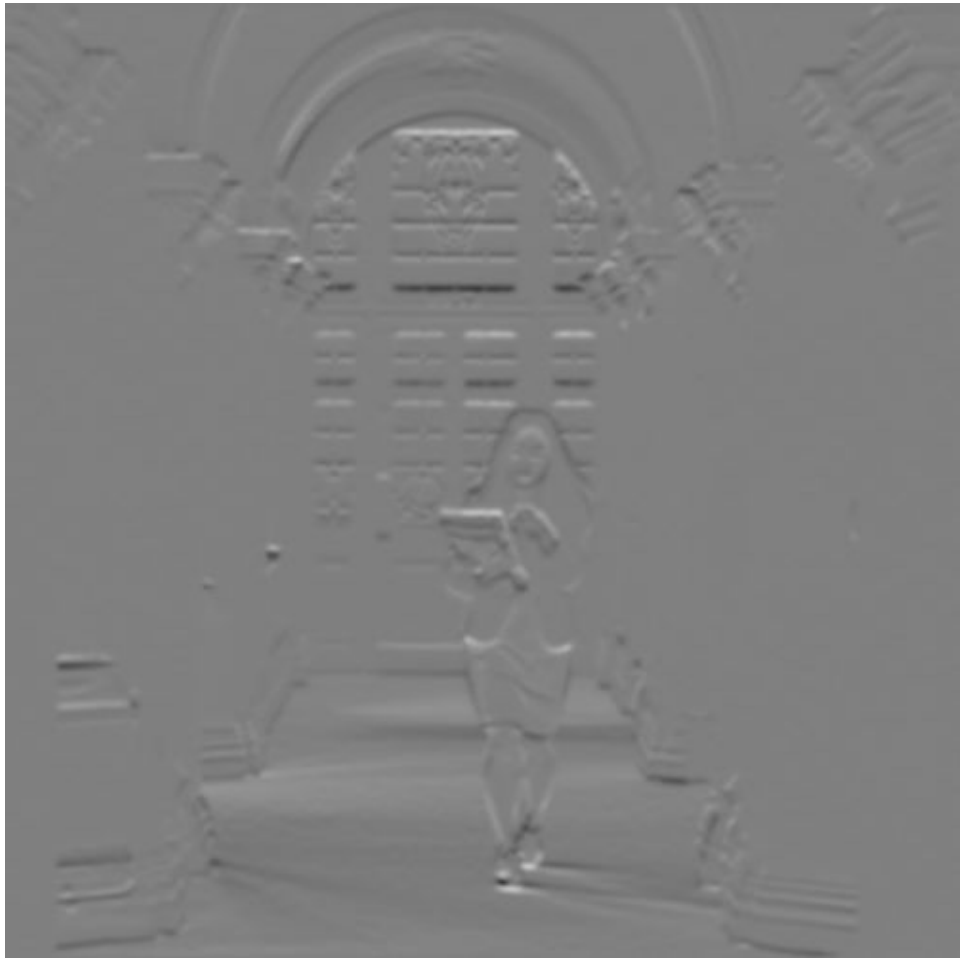




Compute Gradients

X' Derivative of Gaussian

Y Derivative of Gaussian



(x2 + 0.5 for visualization)

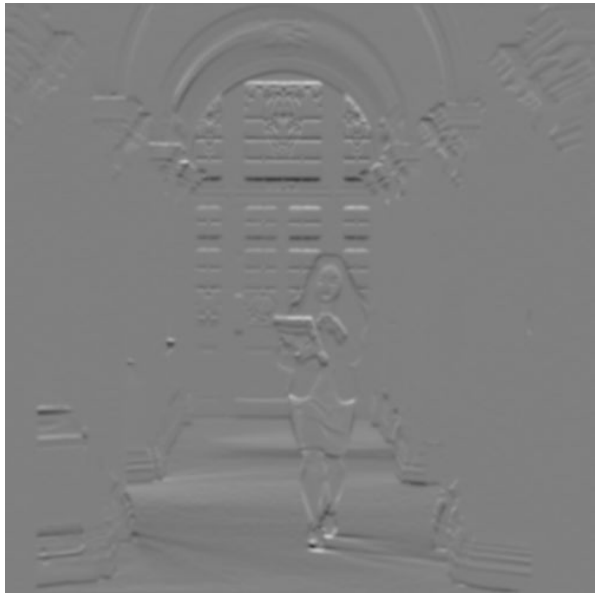
Canny Edge Detector

Algorithm

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

Compute Gradient Magnitude

sqrt(XDerivOfGaussian .^2 + YDerivOfGaussian .^2) = gradient magnitude

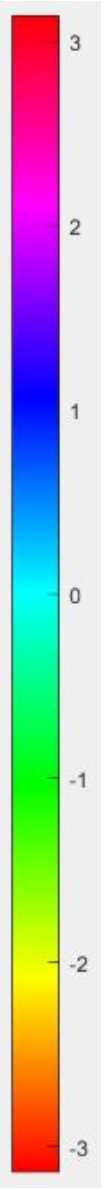


(x4 for visualization)

Compute Gradient Orientation

- Threshold magnitude at minimum level
- Get orientation via

```
theta = atan2(yDeriv, xDeriv)
```

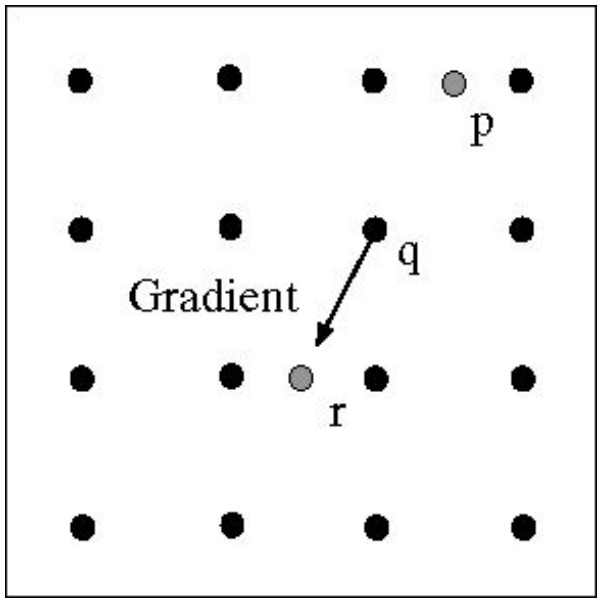


Canny Edge Detector

Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient (or can just use sobel)
3. Non-maximum suppression:
   a. Thin multi-pixel wide "ridges" to single pixel width

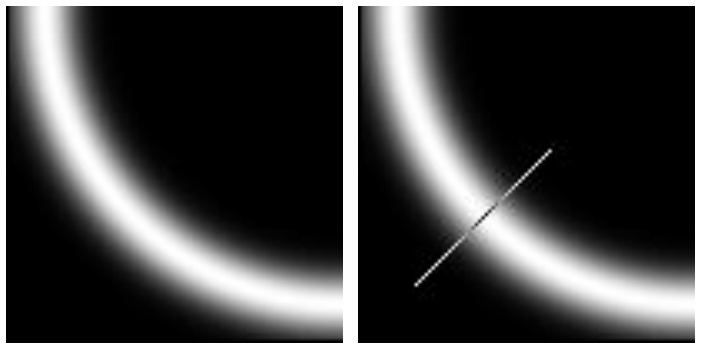
Non-Maximum Suppression for Each Orientation



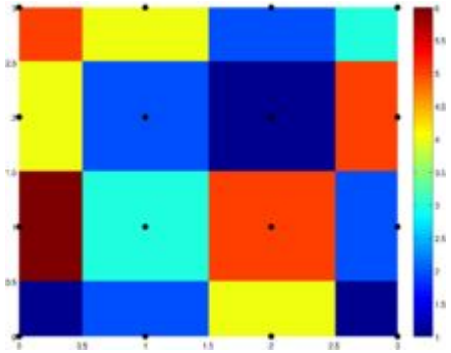
At pixel q:

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

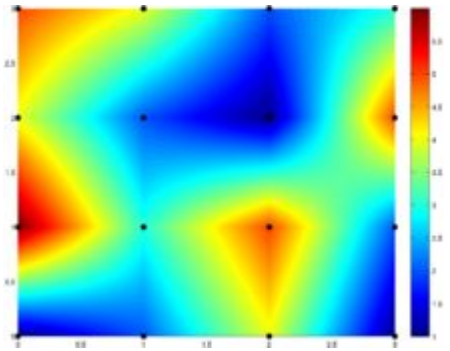
Interpolate along gradient direction to get these values.



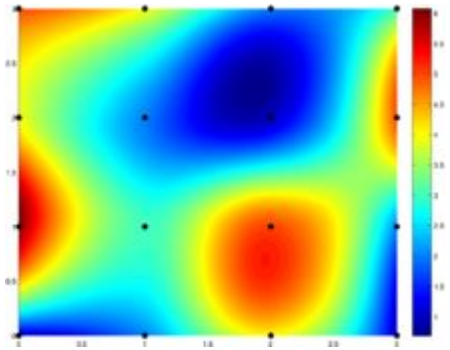
Additional Slide: Interpolation Options



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

Examples from http://en.wikipedia.org/wiki/Bicubic_interpolation

Before Non-max Suppression

Gradient magnitude (x4 for visualization)

After Non-max Suppression

Gradient magnitude (x4 for visualization)

Canny Edge Detector

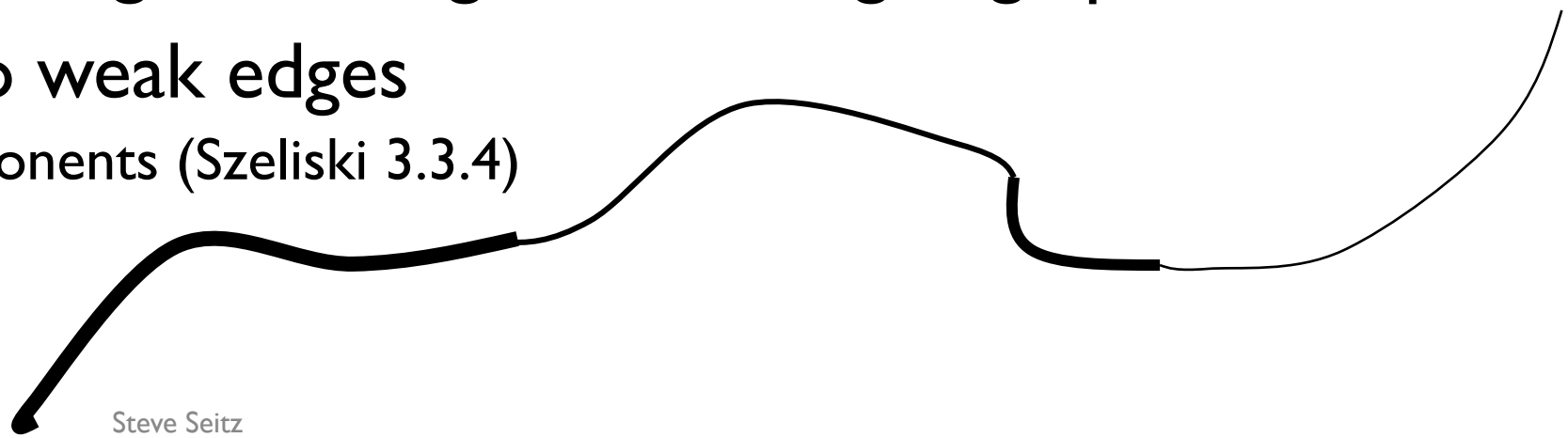
Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide "ridges" to single pixel width
4. 'Hysteresis' Thresholding

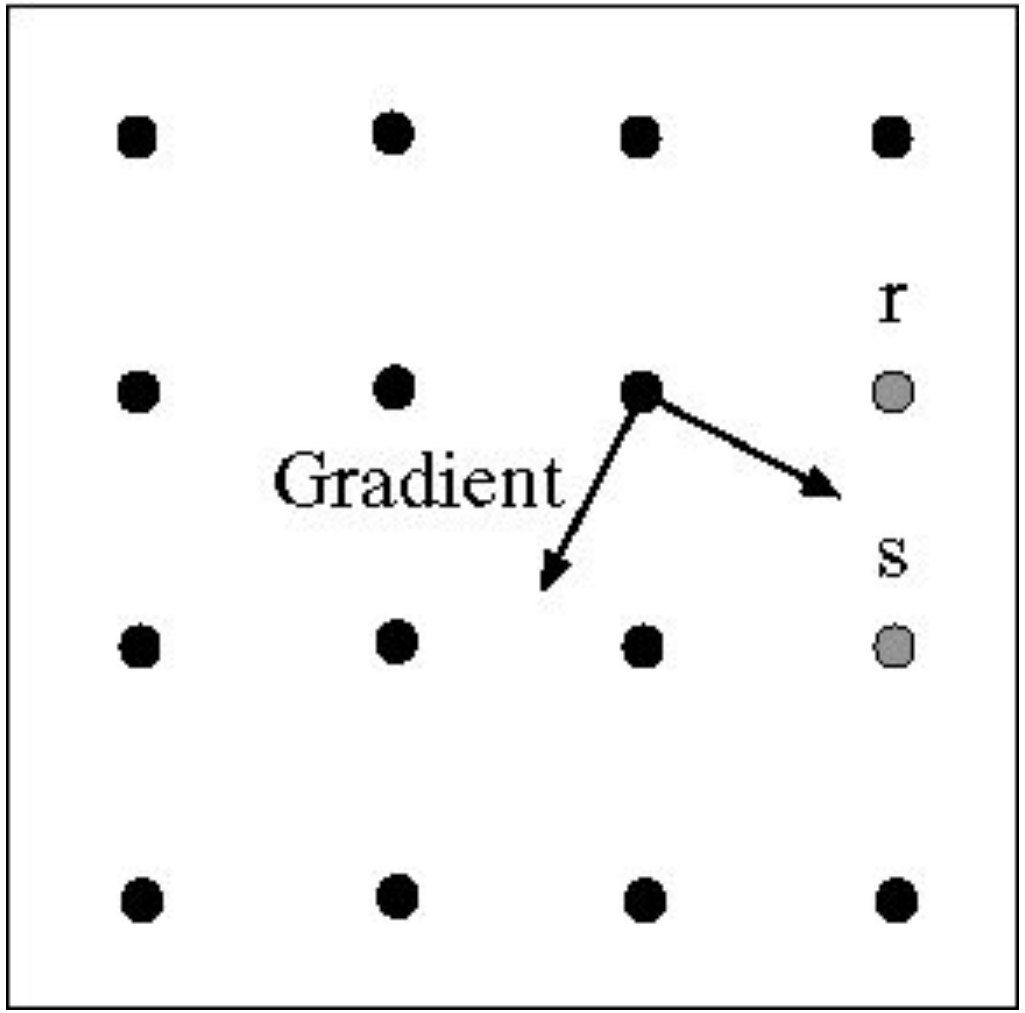
‘Hysteresis’ Thresholding

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

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

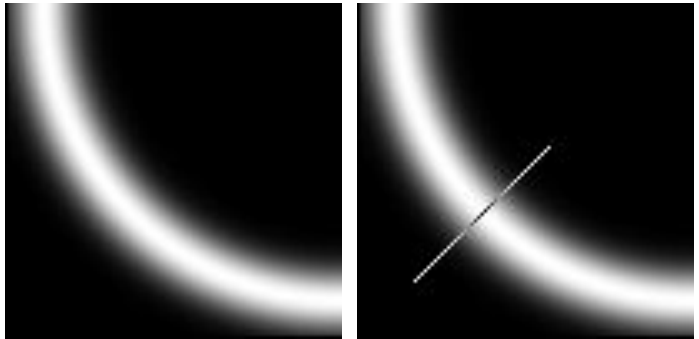


Additional Slide: Edge Linking



Source: D. Forsyth

Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



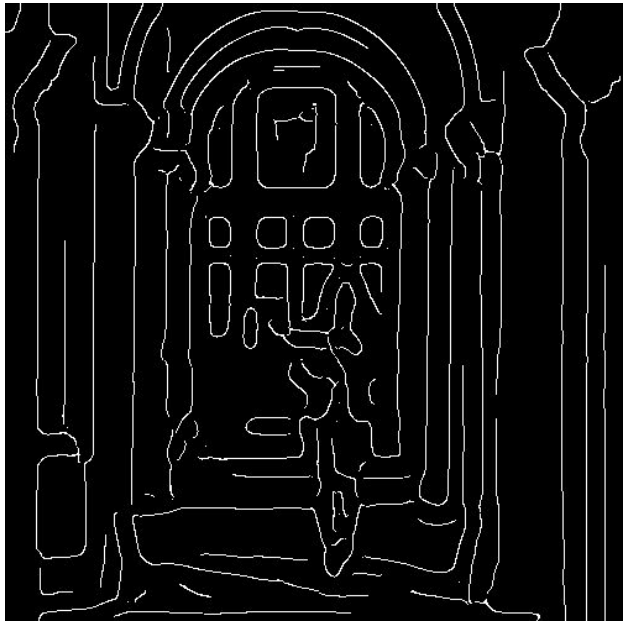
Final Canny Edges

$$\sigma = \sqrt{2}, t_{low} = 0.05, t_{high} = 0.1$$

Effect of σ (Gaussian kernel spread/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

Canny Edge Detector

Algorithm

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   a. Thin multi-pixel wide "ridges" to single pixel width
4. 'Hysteresis' Thresholding:
   a. Define two thresholds: low and high
   b. Use the high threshold to start edge curves and the low threshold to continue them
   c. 'Follow' edges starting from strong edge pixels
      i. Connected components (Szeliski 3.3.4)