Artificial Intelligence II

Part 2: Lecture 2

Yalda Mohsenzadeh

Slides are adapted from Olga Vesker (UW), Steve Seitz (UW), David Jacobs (UMD), D. Lowe (UBC), Hong Man

René Magritte, “Decalcomania”

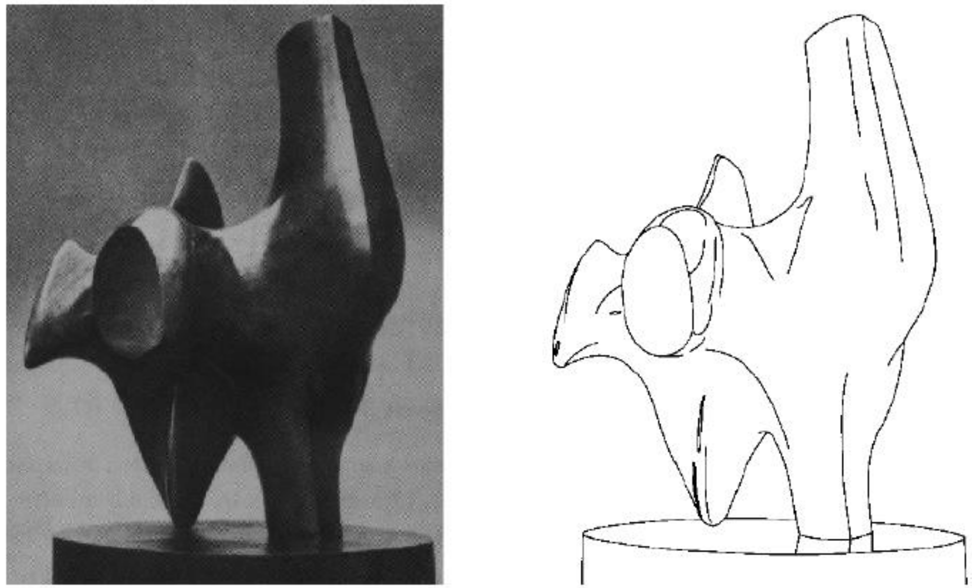
Part 2: Lecture 2
Computer Vision

Edge Detection

Outline

- Edge Detection
- Edge types
- Image Gradient
- Canny Edge Detector

Edge detection



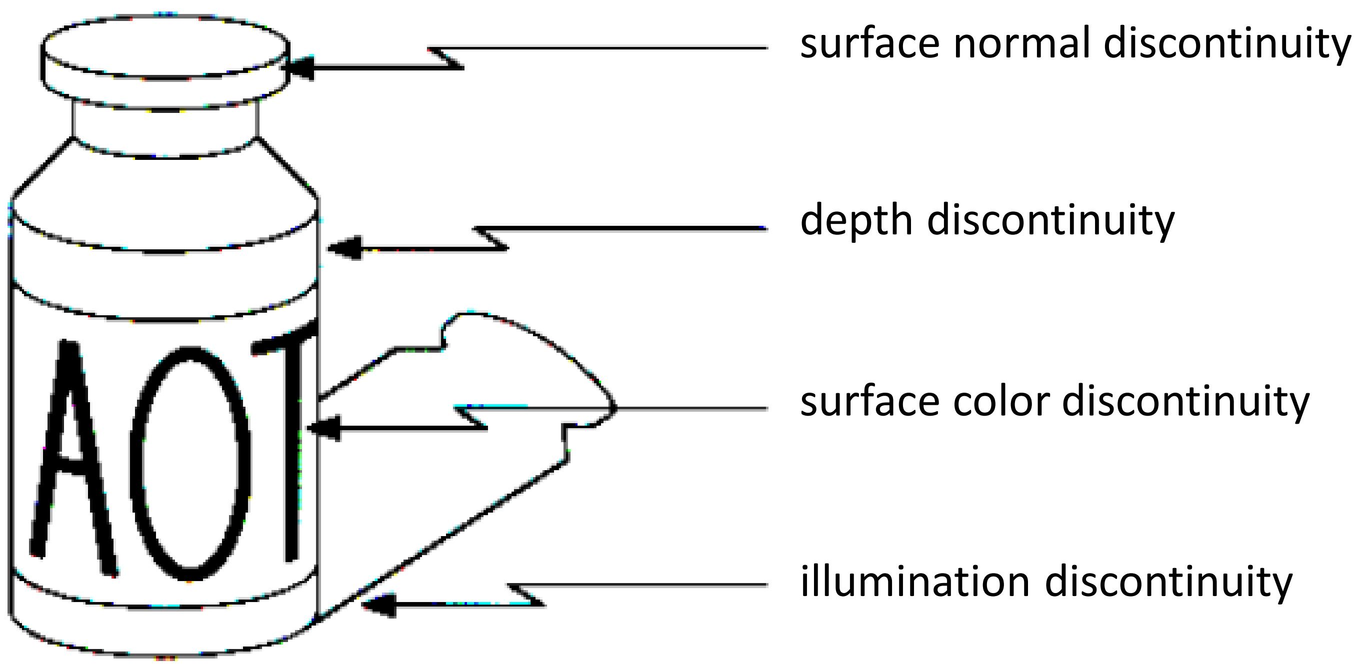
- It is the process of converting a 2D image into a set of "prominent" curves
  - What is a "prominent" curve or edge? Intuitively, it's a place where abrupt changes occur
- Why?
  - Extracts salient features of the scene
  - More compact representation than pixels

Edge detection

- Artists also do it
- They do it much better, they have high level knowledge which edges are more **perceptually** important

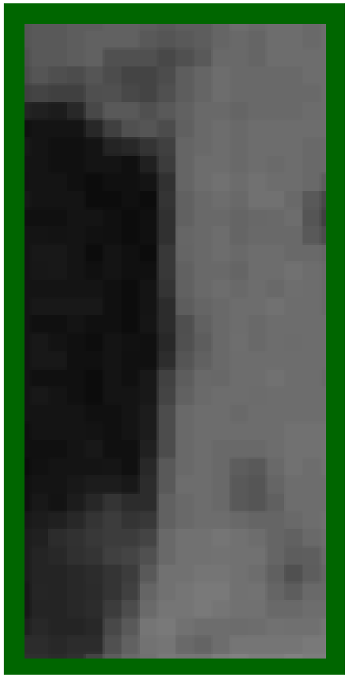
Origin of Edges

- Edges are caused by a variety of factors

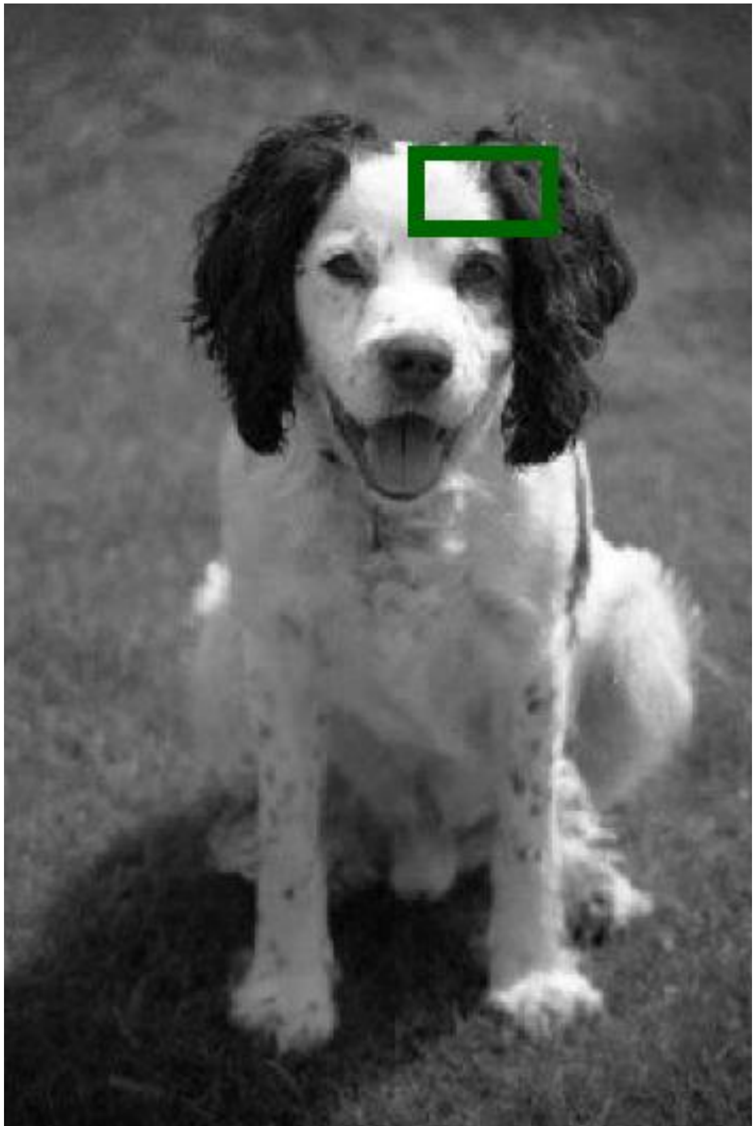


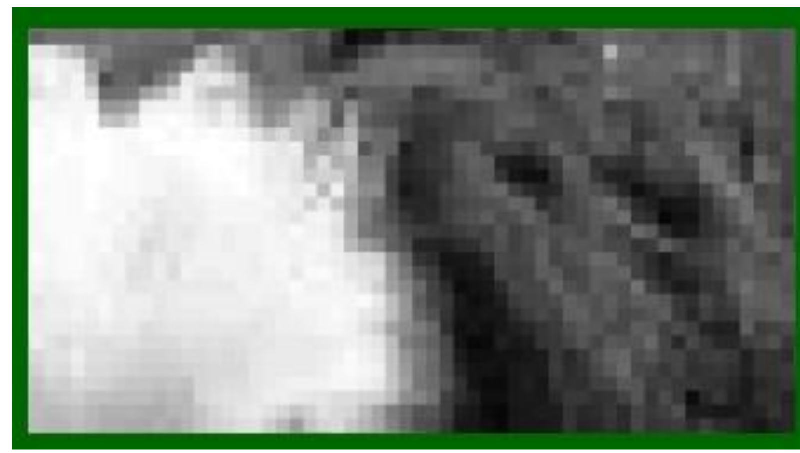
Surface Normal Discontinuity

Depth Discontinuity

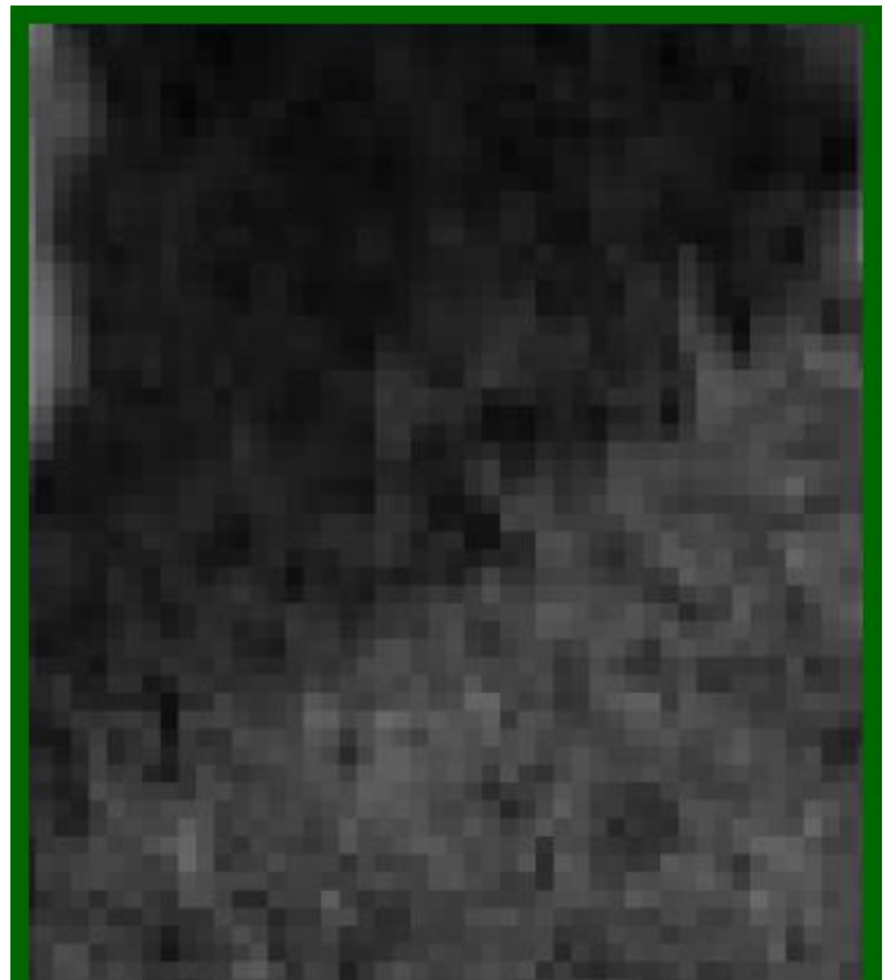


Surface Color Discontinuity



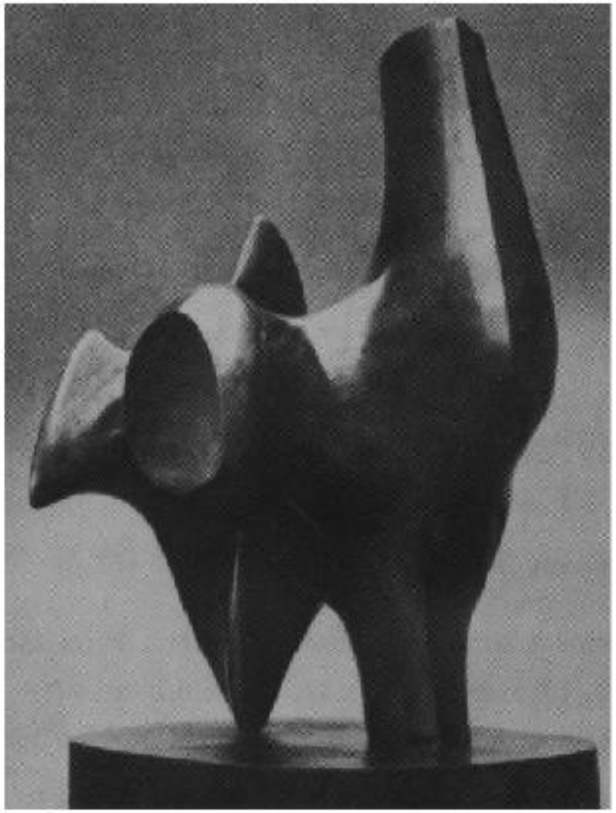


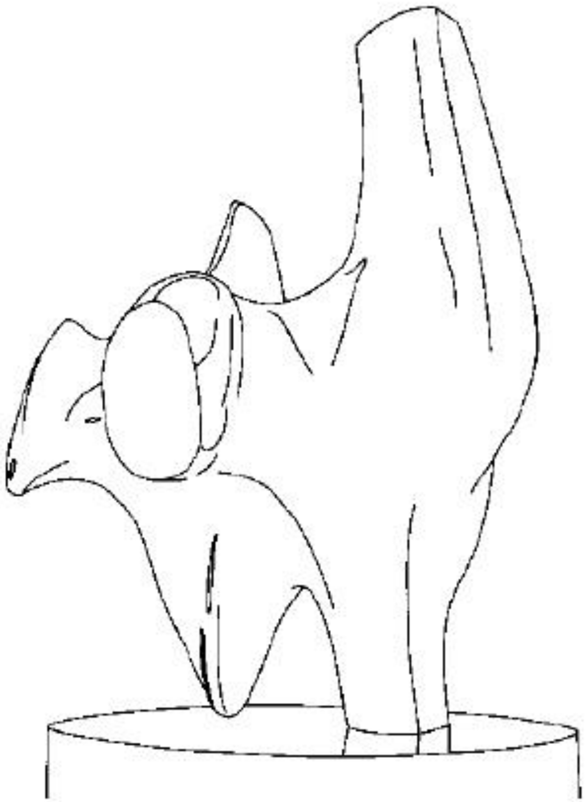
Illumination Discontinuity



Edge detection

- How can you tell that a pixel is on an edge?





Characterizing edges

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

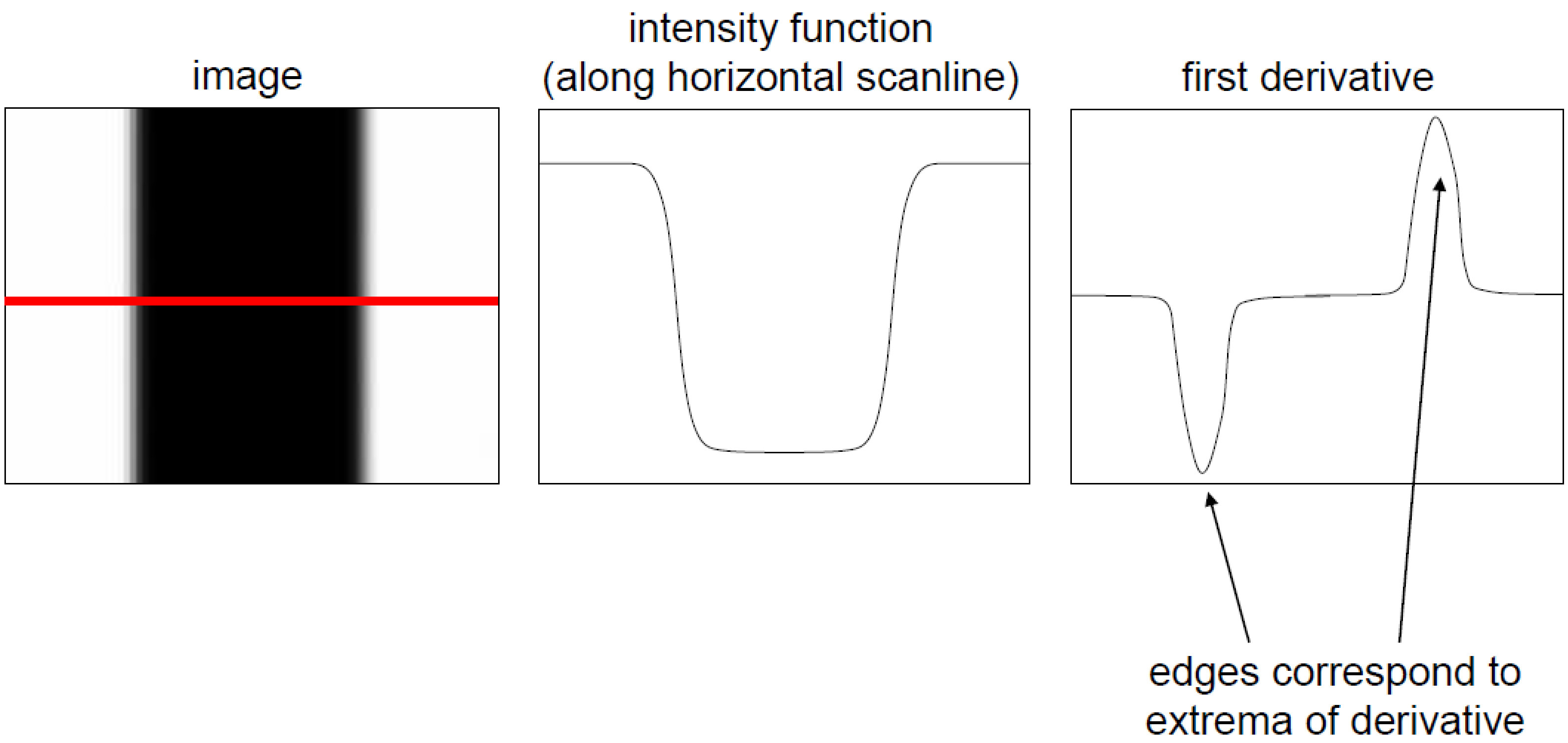
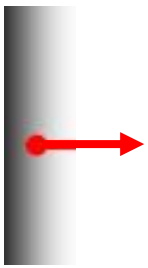
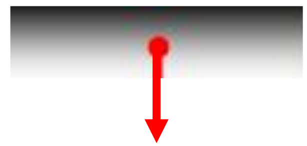


Image gradient

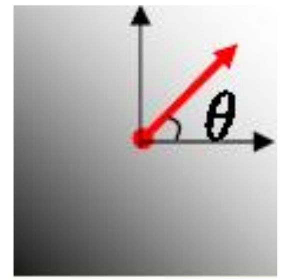
- The gradient of an image: $\Delta f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



$$\Delta f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\Delta f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- The gradient points in the direction of most rapid increase in intensity
- The gradient direction is given by:
  - $\theta = \tan^{-1}\left(\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial y}\right)$
  - Gradient direction is perpendicular to edge
- The edge strength is given by the gradient magnitude

The discrete gradient

- How can we differentiate a digital image?
- Take discrete derivative (finite difference)

$$\frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y)$$

- How would you implement this as a convolution?

	-1	1

h

- Similarly, $\frac{\partial f(x,y)}{\partial x} = f(x,y+1) - f(x,y)$

	-1	
	1	

h

The discrete gradient

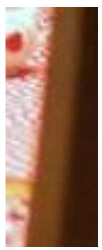
- The discrete gradient simply detects changes between neighboring pixels

$$\frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y)$$

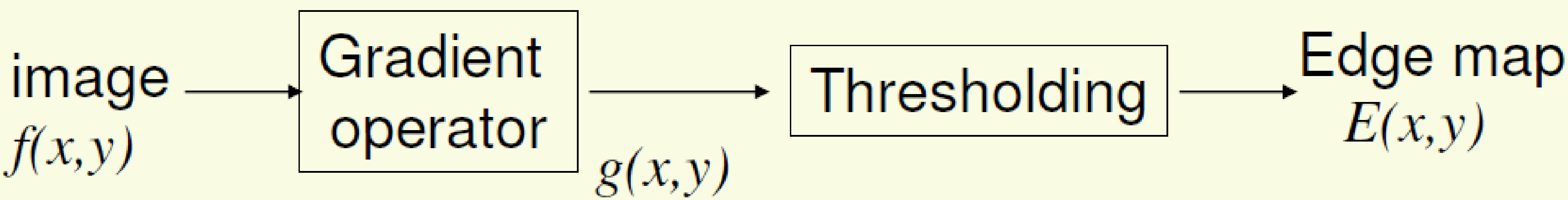
Change in vertical direction

$$\frac{\partial f(x,y)}{\partial x} = f(x,y+1) - f(x,y)$$

change in horizontal direction



- Basic edge detection algorithm:



$$E(x,y) = \begin{cases} 1 & |g(x,y)| > threshold \\ 0 & otherwise \end{cases}$$

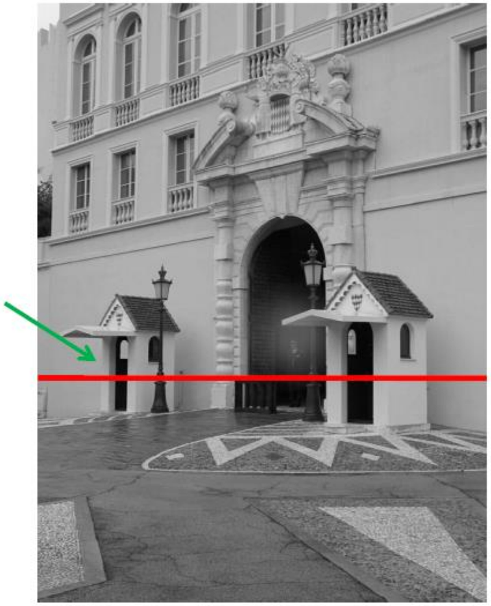
The Sobel operator

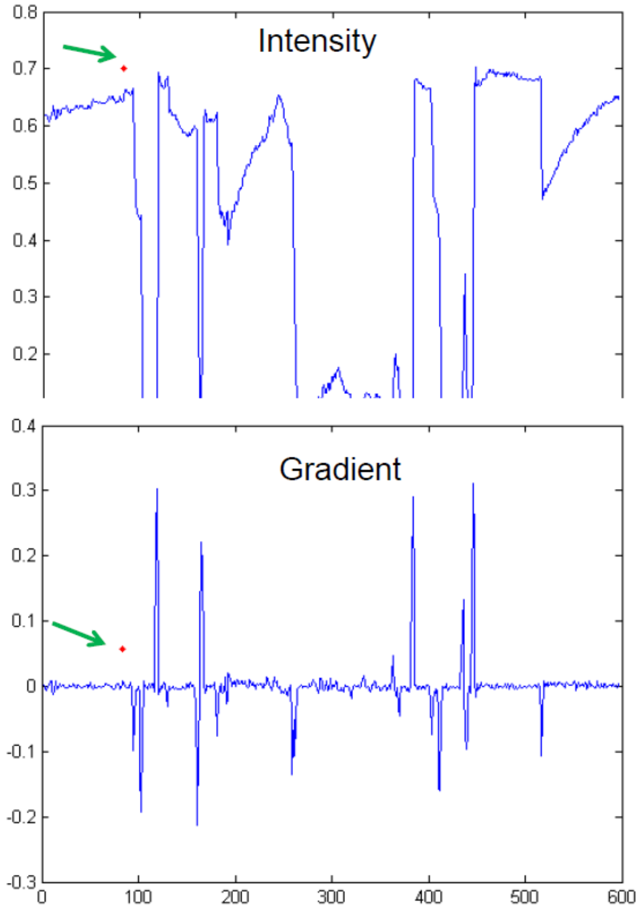
- Better approximations of the derivatives
- The Sobel operators are very commonly used

$$s_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad s_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- The standard definition of the Sobel operator omits the 1/8 term
- It does not make a difference for edge detection
- However, the 1/8 term is needed to get the right gradient value

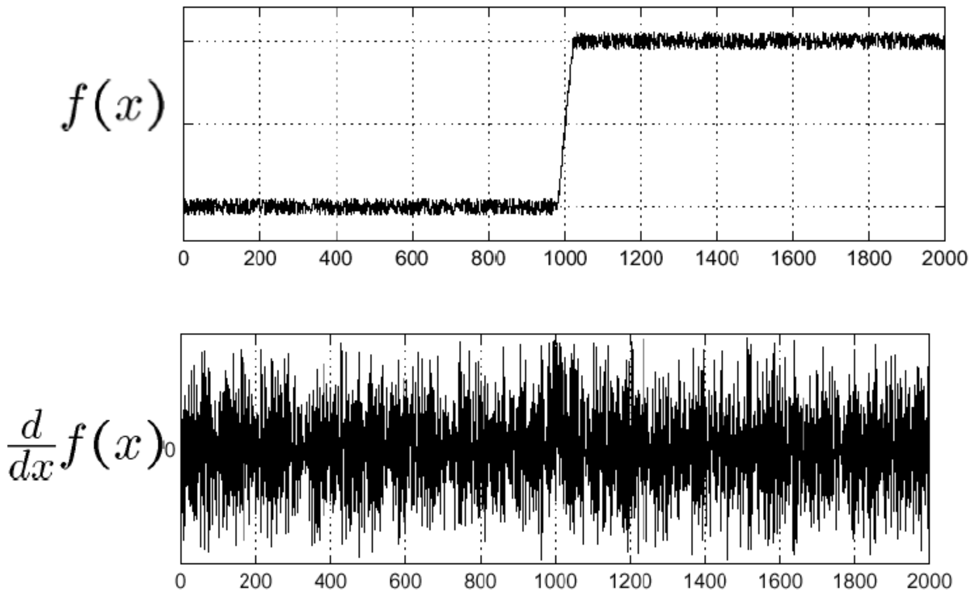
Intensity profile





Effect of Noise

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

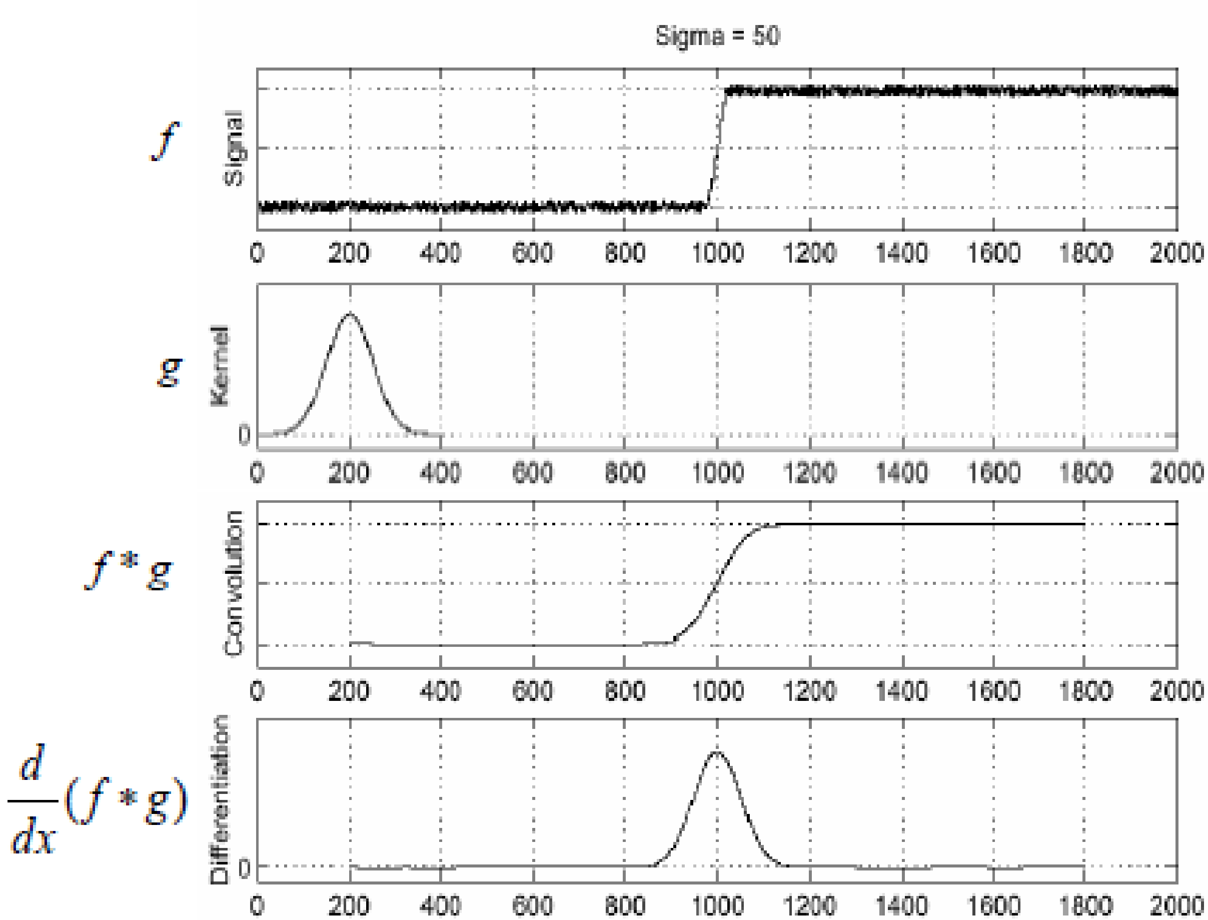


- Where is the edge?

Effects of Noise

- Difference filters respond strongly to noise!
  - Image noise results in pixels which look very different from their neighbors.
  - Generally, the larger the noise the stronger the response!
- How do we deal with noise?
- We already know, filter the noise out
  - Using Gaussian kernel (for example)
- First convolve image with a Gaussian filter
- Then convolve image with an edge detection filter like Sobel

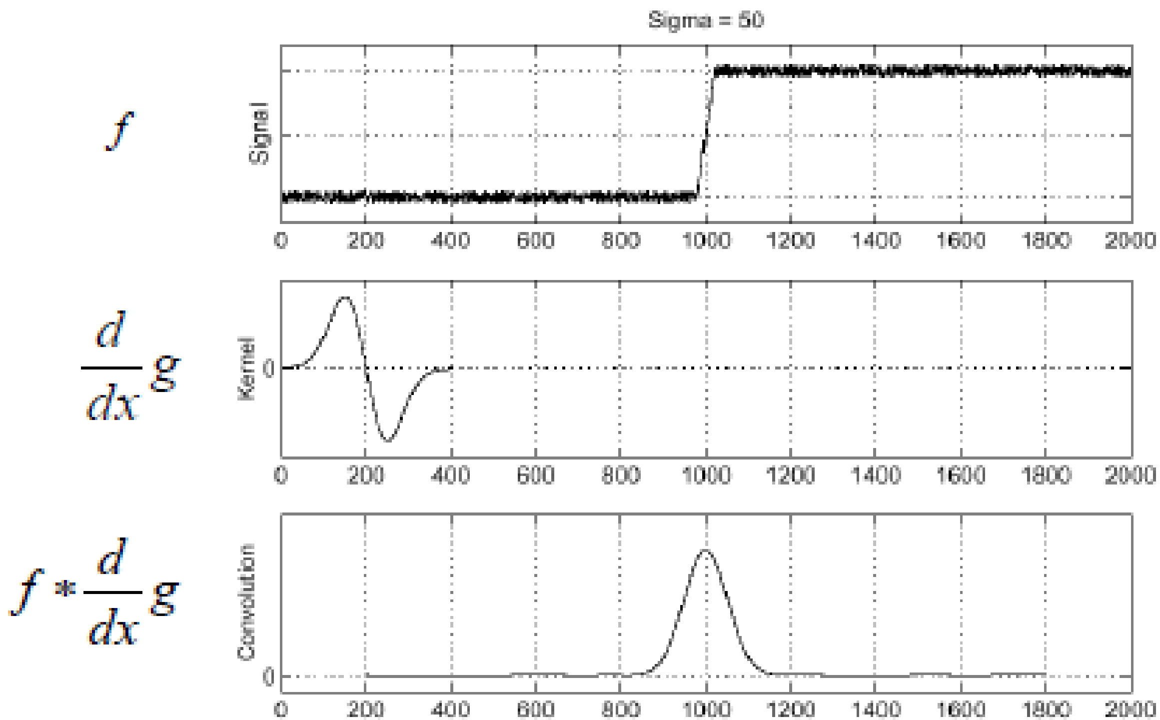
Solution: smooth first



Derivative theorem of convolution

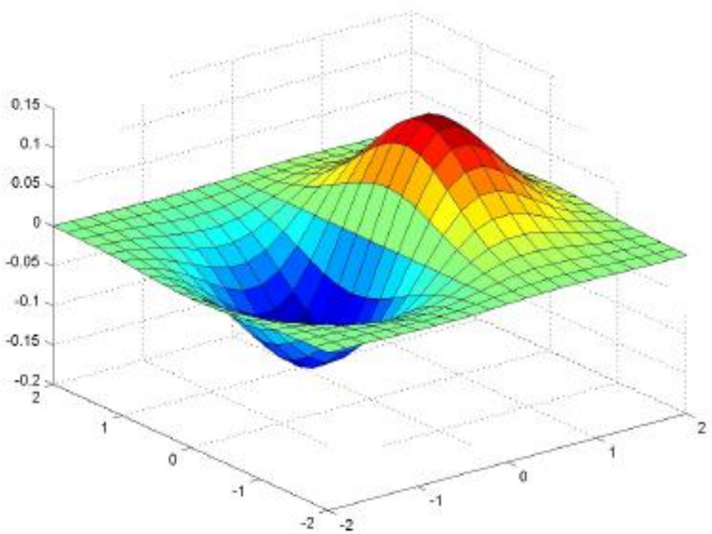
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

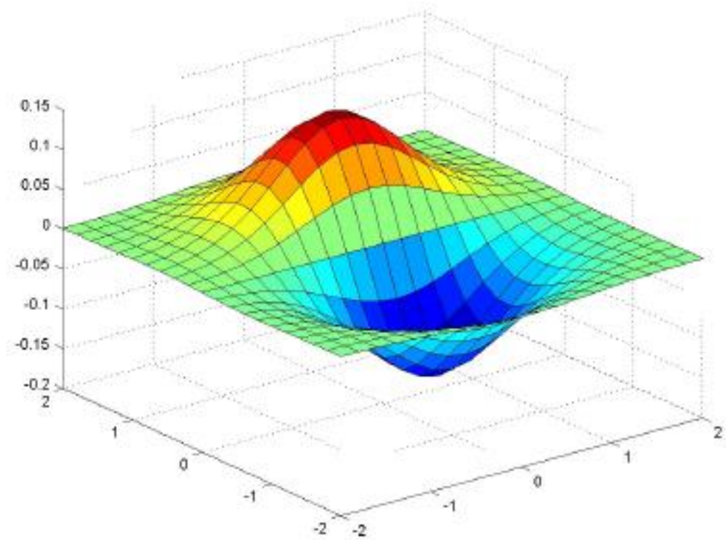
- This saves us one operation:



Source: S. Seitz

Derivative of Gaussian

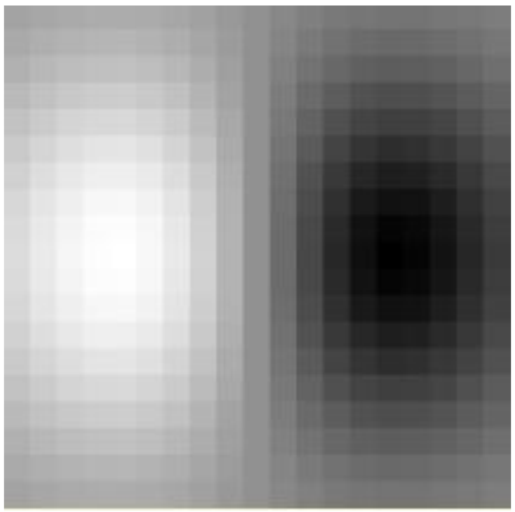




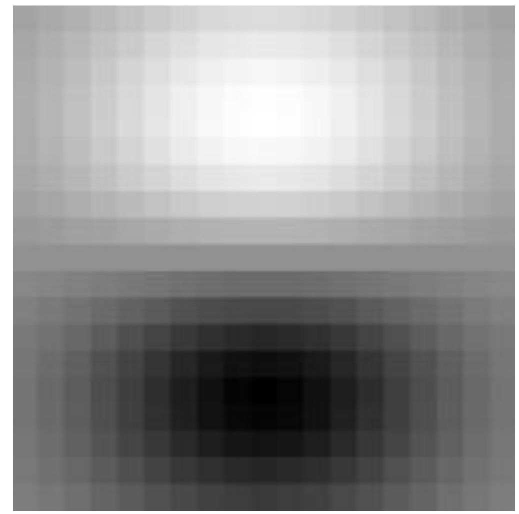
$$\frac{\partial G_\sigma}{\partial x}$$

$$\frac{\partial G_\sigma}{\partial y}$$

Derivative of Gaussian



$$\frac{\partial G_\sigma}{\partial x}$$



$$\frac{\partial G_\sigma}{\partial y}$$

- Bright corresponds to positive values, dark to negative values

Derivative of Gaussian: Example

- If we ignore normalizing constant: $G_\sigma(x, y) = \; e^{-\frac{(x^2+y^2)}{2\sigma^2}}$
- differentiate with respect to x and y
- $\frac{\partial G_\sigma(x,y)}{\partial x} = -\frac{x}{\sigma^2} . e^{-\frac{(x^2+y^2)}{2\sigma^2}}$ and $\frac{\partial G_\sigma(x,y)}{\partial y} = -\frac{y}{\sigma^2} . e^{-\frac{(x^2+y^2)}{2\sigma^2}}$
- Plug some values to get gradient detection masks $H_x \; and \; H_y$
- For example, let $\sigma = 5$, and let (x, y) be in $[-2 \;\times 2][-2 \times 2]$ window

(-2,2)	(-1,2)	(0,2)	(1,2)	(2,2)
(-2,1)	(-1,1)	(0,1)	(1,1)	(2,1)
(-2,0)	(-1,0)	(0,0)	(1,0)	(2,0)
(-2,-1)	(-1,-1)	(0,-1)	(1,-1)	(2,-1)
(-2,-2)	(-1,-2)	(0,-2)	(1,-2)	(2,-2)

H_x

0.04	0.08	0	-0.08	-0.04
0.16	0.37	0	-0.37	-0.16
0.27	0.61	0	-0.61	-0.27
0.16	0.37	0	-0.37	-0.16
0.04	0.08	0	-0.08	-0.04

H_y

-0.04	-0.04	-0.04	-0.04	-0.04
-0.08	-0.08	-0.08	-0.08	-0.08
0	0	0	0	0
0.08	0.08	0.08	0.08	0.08
0.04	0.04	0.04	0.04	0.04

Derivative of Gaussian: Example

H_x

0.04	0.08	0	-0.08	-0.04
0.16	0.37	0	-0.37	-0.16
0.27	0.61	0	-0.61	-0.27
0.16	0.37	0	-0.37	-0.16
0.04	0.08	0	-0.08	-0.04

H_y

-0.04	-0.04	-0.04	-0.04	-0.04
-0.08	-0.08	-0.08	-0.08	-0.08
0	0	0	0	0
0.08	0.08	0.08	0.08	0.08
0.04	0.04	0.04	0.04	0.04

121	121	122	123	122	123
121	121	122	123	122	123
122	123	124	123	124	123
120	122	122	123	122	123
121	121	124	123	124	123
125	120	124	123	124	123

Apply H_x to the red image pixel: -0.78

Apply H_y to the red image pixel: 0.46

121	121	122	123	20	20
121	121	122	123	22	22
122	123	124	123	24	21
120	122	122	123	22	22
121	121	124	123	24	23
125	120	124	123	24	24

Apply H_x to the red image pixel: **217**

Apply H_y to the red image pixel: 0.69

Derivative of Gaussian: Example

A mask looks like a pattern it is trying to detect!

121	121	122	123	122	123
121	121	122	123	122	123
122	123	124	123	124	123
120	122	122	123	122	123
20	22	24	22	24	23
20	22	21	22	23	24

Apply H_x to the red image pixel: -0.69

Apply H_y to the red image pixel: **-217**

121	121	122	123	20	20
121	121	122	123	22	22
122	123	124	123	24	21
120	122	122	123	22	22
121	121	124	123	24	23
125	120	124	123	24	24

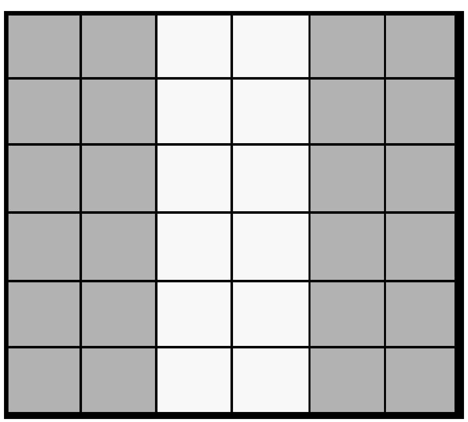
Apply H_x to the red image pixel: **217**

Apply H_y to the red image pixel: 0.69

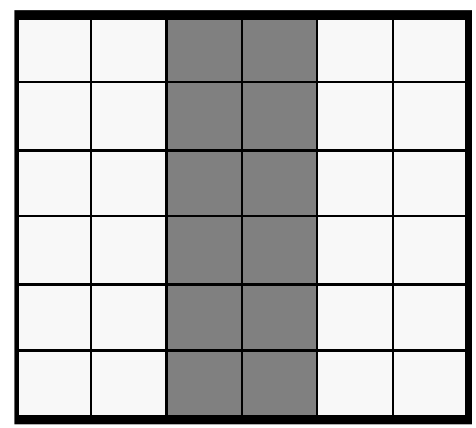
What does this mask detects?

H

2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2



Strong negative response

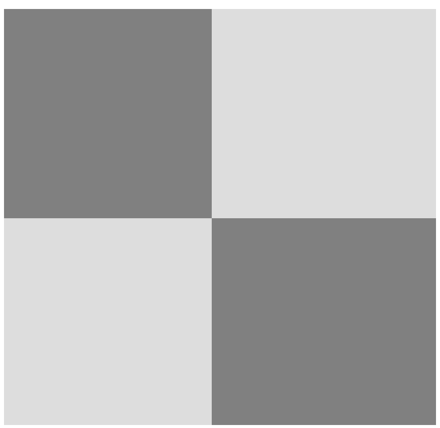


Strong positive response

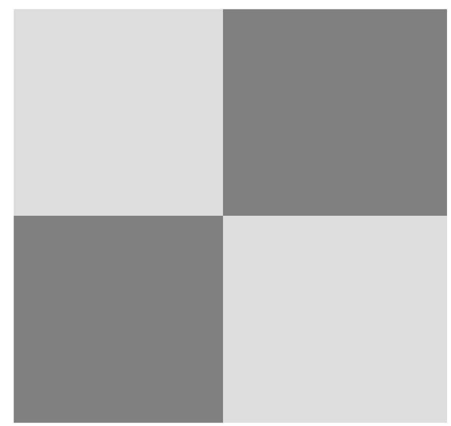
What does this mask detects?

H

2	2	-2	-2
2	2	-2	-2
-2	-2	2	2
-2	-2	2	2



Strong negative response



Strong positive response

There is Always a trade-off between smoothing and good edge localization!

Image with Edge

Edge Location

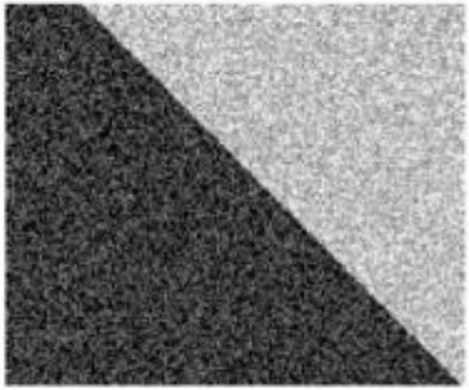
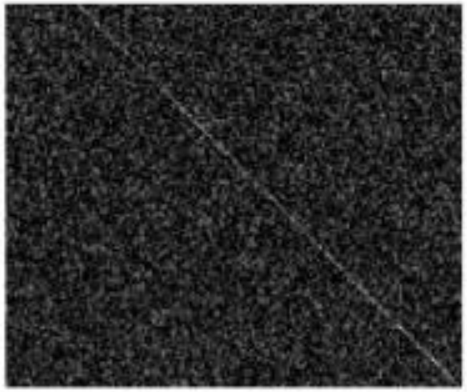


Image + Noise



Derivatives detect
edge *and* noise

Smoothed derivative removes
noise, but blurs edge

The Canny edge detector

- Original image (Lena)

The Canny edge detector

- Norm of the gradient

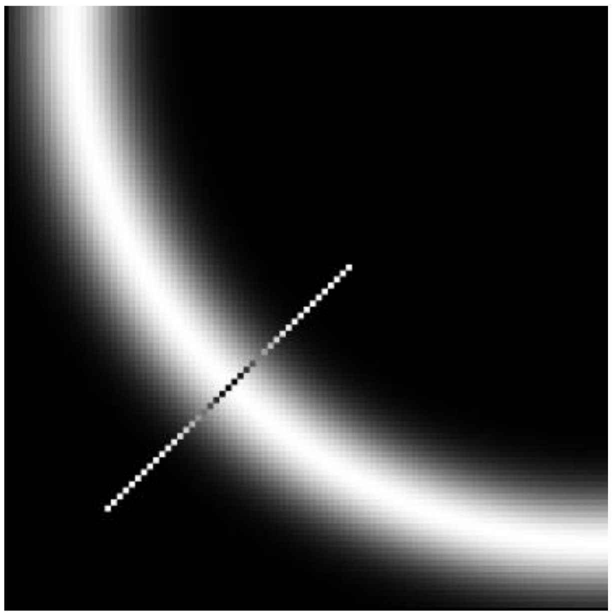
The Canny edge detector

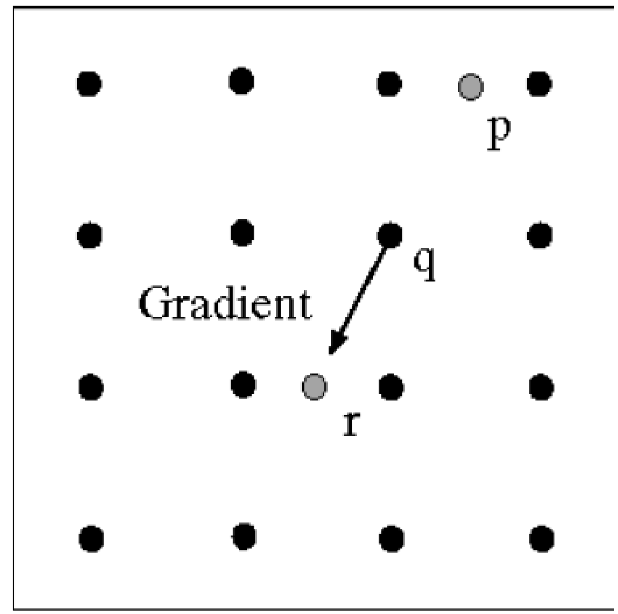
- thresholding

The Canny edge detector

- Thinning
- Non-maximum suppression

Non-maximum suppression





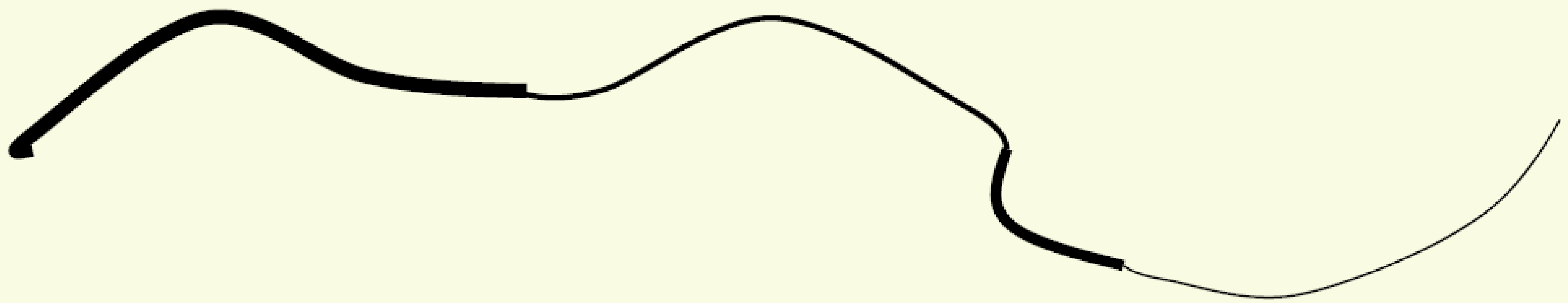
- Check if pixel is local maximum along gradient direction
- Requires checking interpolated pixels p and r

Canny Hysteresis thresholding

- Keep both a high threshold H and a low threshold L
- Any edges with strength <L are discarded
- Any edges with strength > H are kept
- An edge p with strength between L and H is kept only if there is a path of edges with strength >L connecting p to an edge of strength >H

Hysteresis

- Strong Edges reinforce adjacent weak edges
- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use hysteresis
  - use a high threshold to start edge curves and a low threshold to continue them.



Effect of Gaussian kernel width

original　　Original with $\sigma = 1$　　Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
- large σ detects large scale edges
- small σ detects fine features

Canny edge detector

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute magnitude of gradient at every pixel

$$M(x, y) = |\nabla I| = \sqrt{I_x^2 + I_y^2}$$

3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

4. Hysteresis Thresholding

   - Select the pixels such that $M > T_h$ (high threshold)

   - Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points

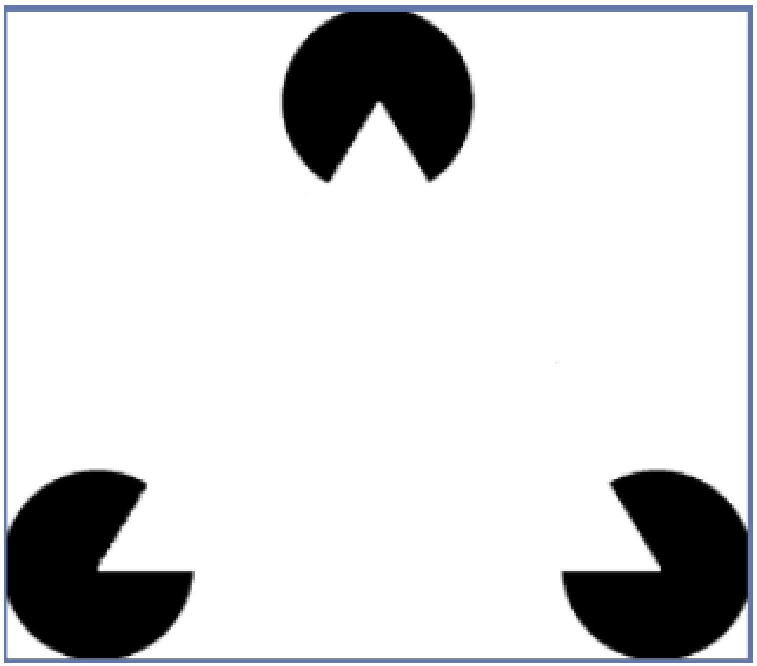
Source: D. Lowe, L. Fei-Fei

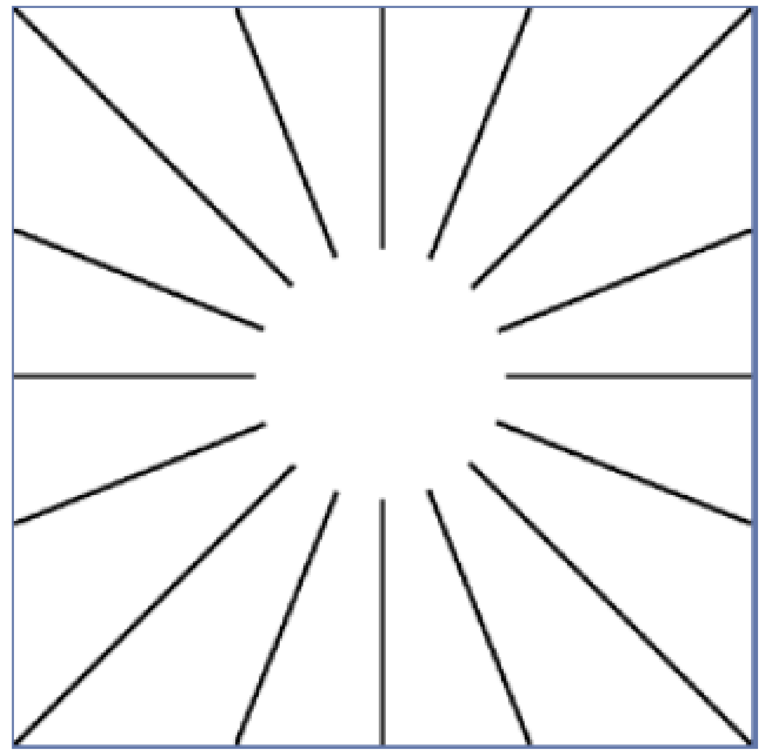
Why is Canny so Dominant?

- Still widely used.

1. Theory is nice (but end result same).
2. Details good (magnitude of gradient).
3. Code was distributed.
4. Perhaps this is about all you can do with linear filtering.

Illusory Contours





- Triangle and circle floating in front of background
- Not possible to detect the “illusory” contours using local edge detection