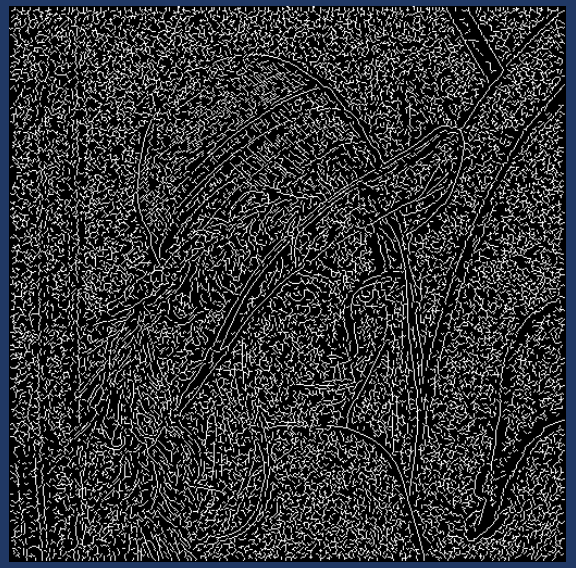
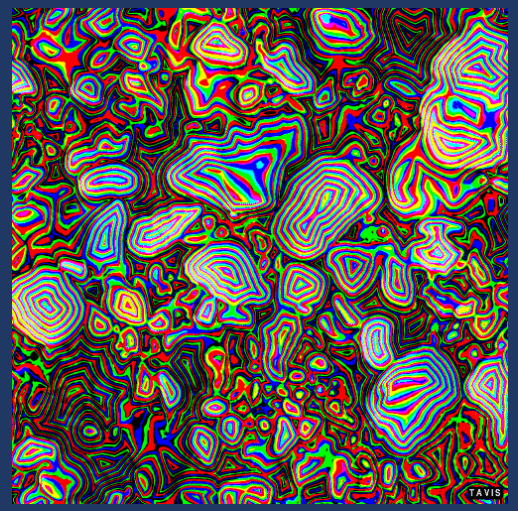
Edges

February 20th, 2019

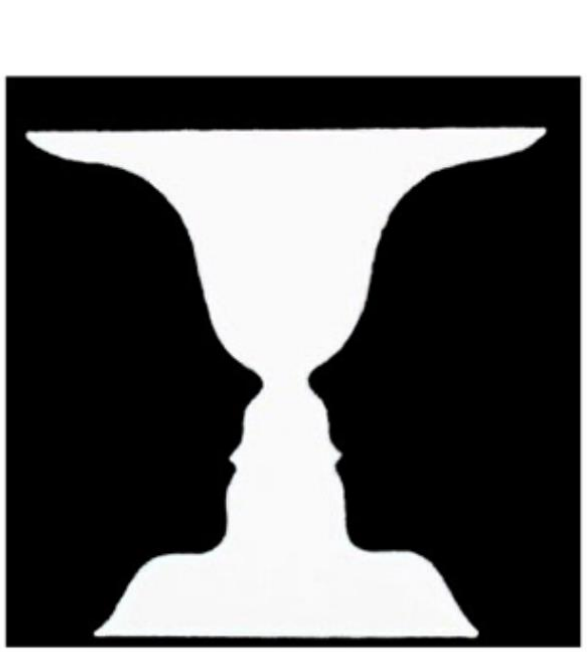
Presented by Dr. Sander Ali Khowaja





Edges

- **Edges** are significant local changes in the image and are **important** features for analyzing images.
- **Edges** typically occur on the boundary between two different regions in an image.

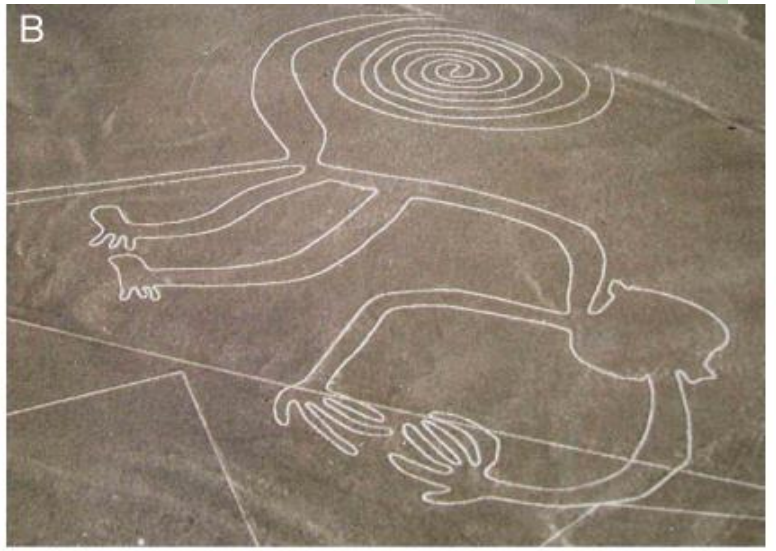


(A)Cave painting at Chauvet, France, about 30,000 B.C.;

(B)Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;

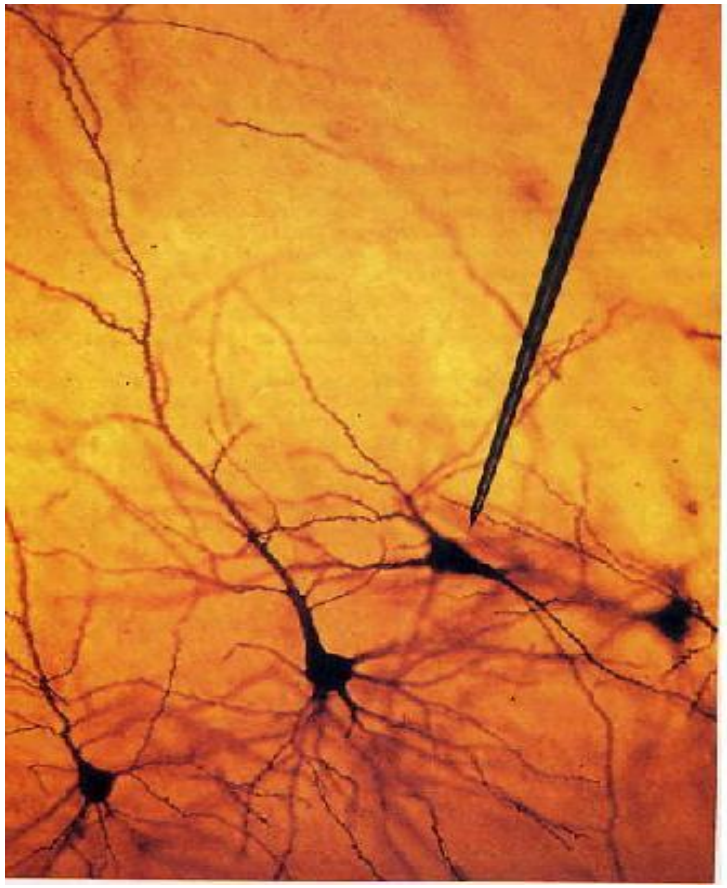
(C)Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;

(D)Line drawing by 7-year old I. Lleras (2010 A.D.).



Edges are Important from human perspective

We know edges are special from human (mammalian) vision studies



Hubel & Wiesel, 1960s

Hubel and Wiesel started out by trying to get neurons in the visual cortex to respond to traditional visual stimuli such as dots. They had little success. No matter the size or position of the circles, their neurons just couldn't care less. Then one day while they were recording from yet another stubbornly silent neuron, it suddenly started firing like crazy as they changed the projection slide that they were using to present the stimuli. Turns out, the cell was responding to the edge of the slide. That's when Hubel and Wiesel discovered that there are cells specialized for detecting lines—a basic feature of the visual world.

Edges are important

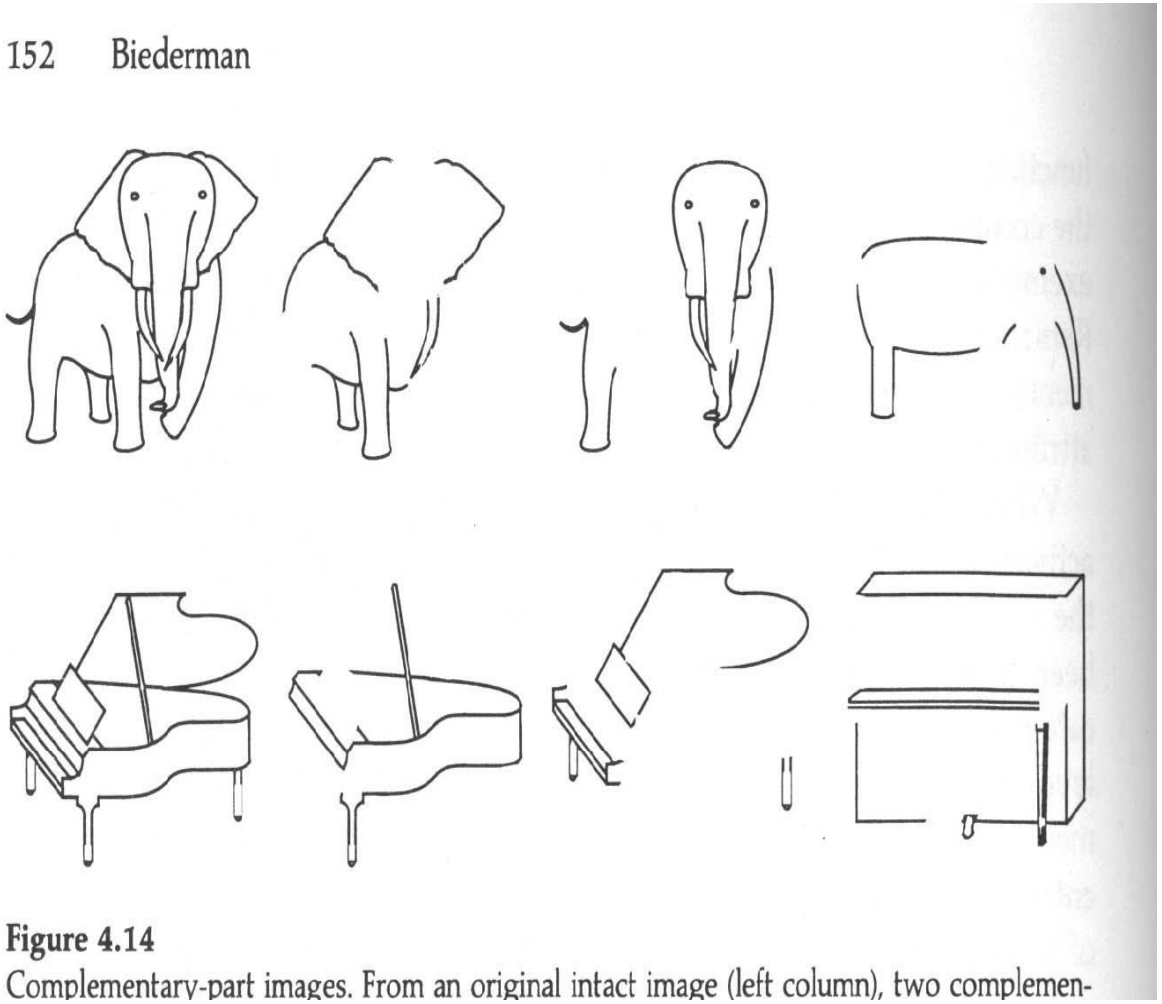
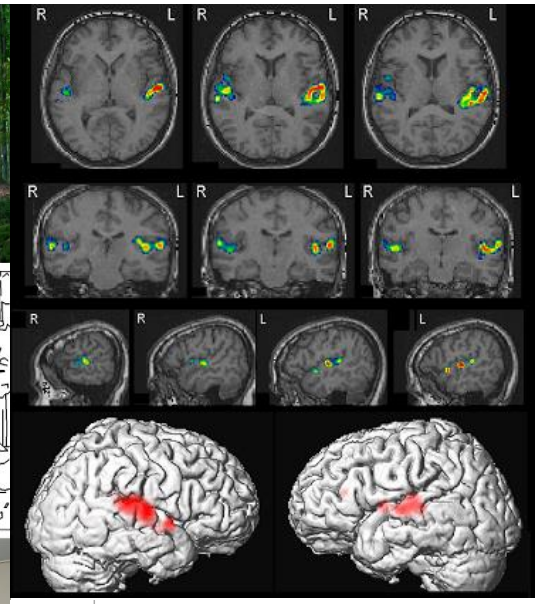


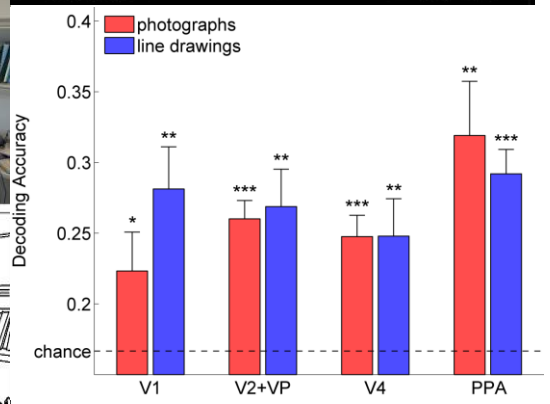
Figure 4.14
Complementary-part images. From an original intact image (left column), two complemen-

Part-deleted complementary images were created by deleting approximately half the components from each image. In this way each complementary images contained about 50% of the original contour. Put together the two complementary images would reproduce the intact line drawing. A same name, different exemplar image was also created (right column) to assess non-visual priming in a naming experiment (Biederman and Cooper, 1991).

Performance of human subjects on identical and complement images was strikingly different indicating that complementary part-deleted image pairs were **not equivalent to each other** for human observers. In fact, the complement image was perceived as if it were a different exemplar of the same category.

Edges are Important





They collected fMRI data while participants viewed photographs and line drawings of beaches, city streets, forests, highways, mountains, and offices. Despite the marked difference in scene statistics, we were able to decode scene category from fMRI data for line drawings just as well as from activity for color photographs,

global scene structure, which is preserved in line drawings, plays an integral part in representing scene categories

Walther, Chai, Caddigan, Beck & Fei-Fei, *PNAS, 2011*

Edge Detection

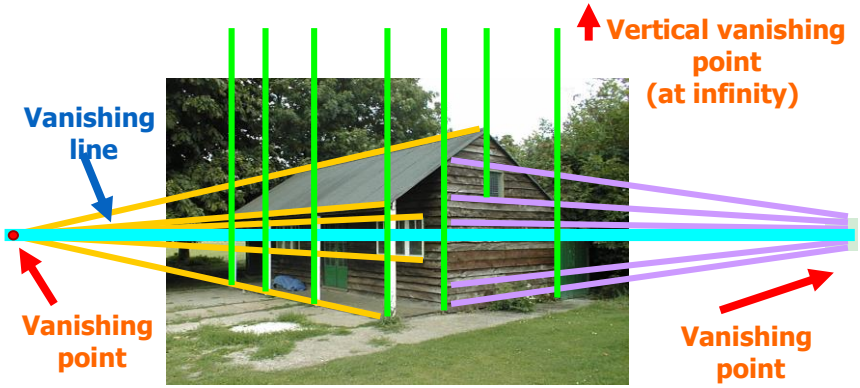
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



Source: J. Hayes

Origins of Edges

surface normal discontinuity

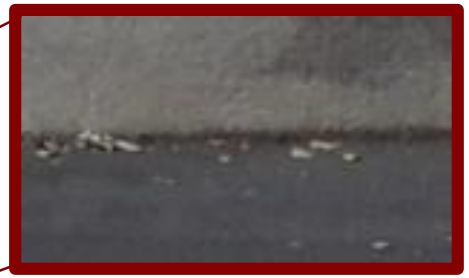
depth discontinuity

surface color discontinuity

illumination discontinuity

Close up of Edges

Surface normal discontinuity



Close up of Edges

Depth discontinuity

Close up of Edges

Surface color discontinuity

Image Gradients (Derivatives in 1D)

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

Example

$$y = x^2 + x^4 \qquad y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = 2x + 4x^3 \qquad \frac{dy}{dx} = \cos x + (-1)e^{-x}$$

Discrete Derivative in 1D

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Types of discrete derivative in 1D

Backward $$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Forward $$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Central $$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

1D discrete derivative filters

- Backward filter: [0 1 -1]

$$f(x) - f(x-1) = f'(x)$$

- Forward: [-1 1 0]

$$f(x) - f(x+1) = f'(x)$$

- Central: [1 0 -1]

$$f(x+1) - f(x-1) = f'(x)$$

1D derivative example

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

$$f'(x) = \quad 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

Discrete Derivative in 2D

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_x}{f_y}$$

2D discrete filters

$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

What happens when we apply this filter?

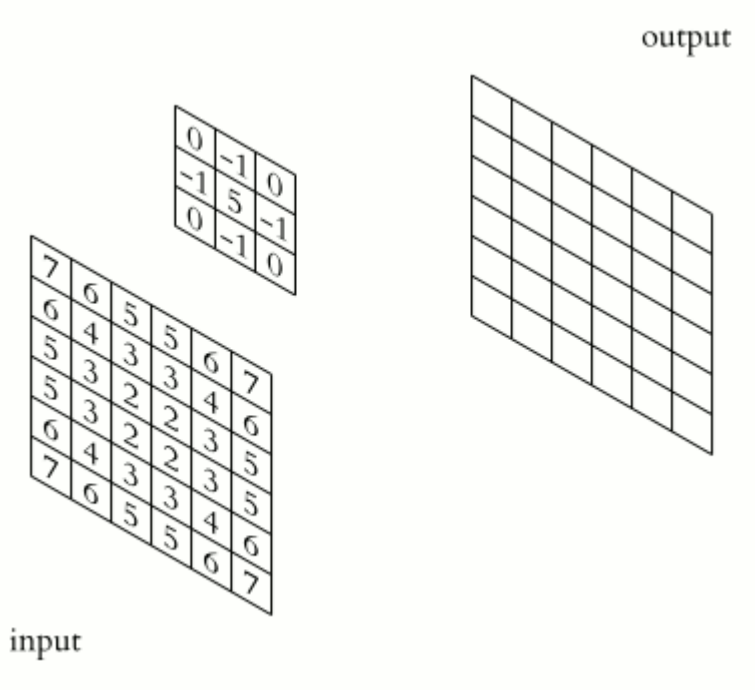
$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Example

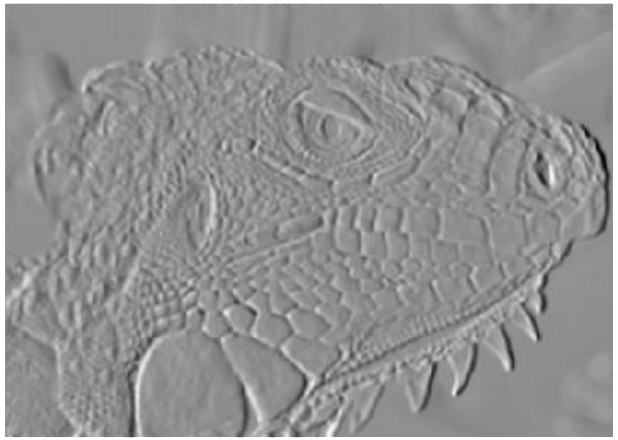
$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

3x3 image gradient filters

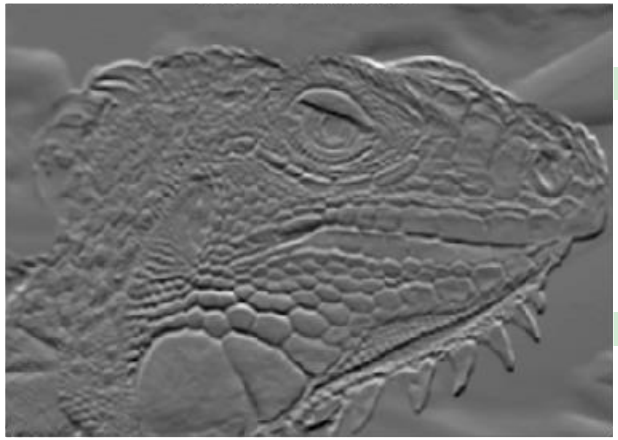
$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



Derivative in x direction



Derivative in y direction



Characterizing edges

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

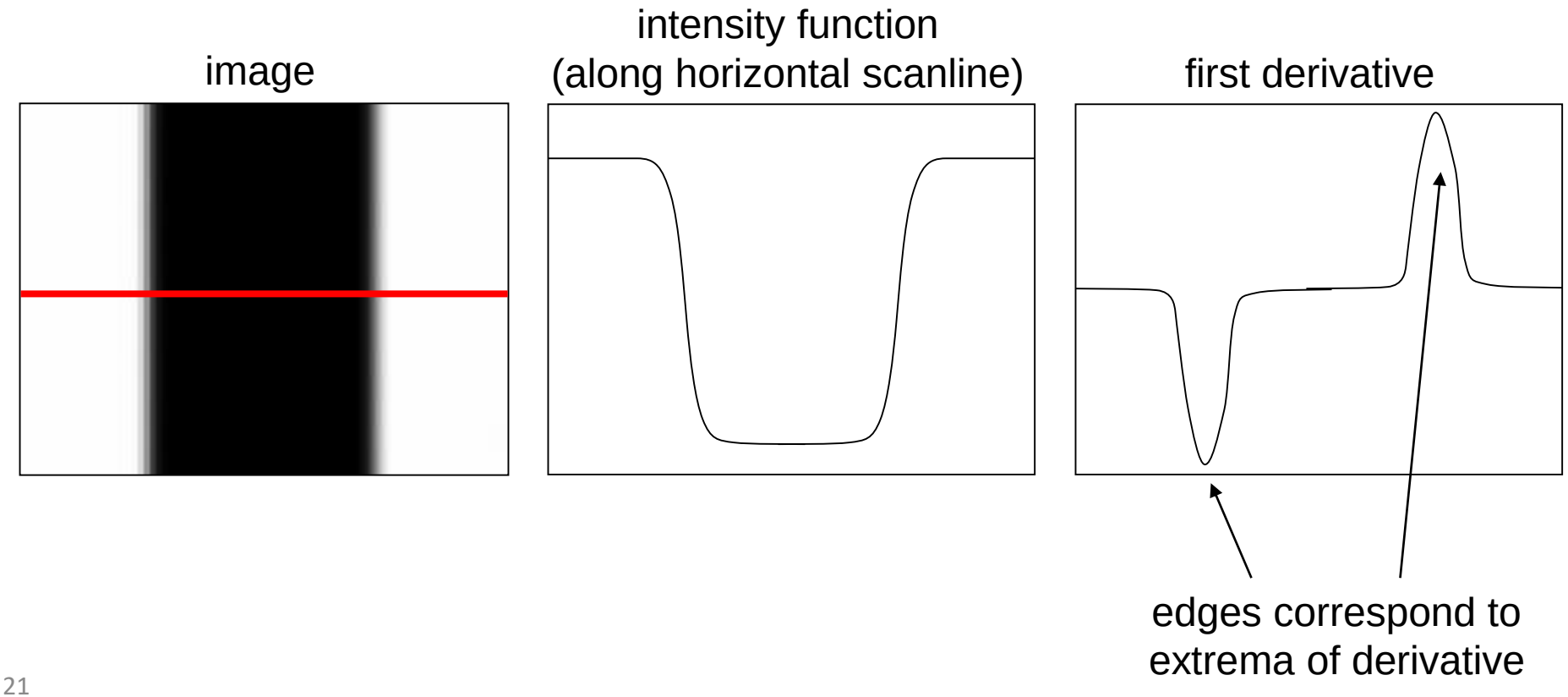
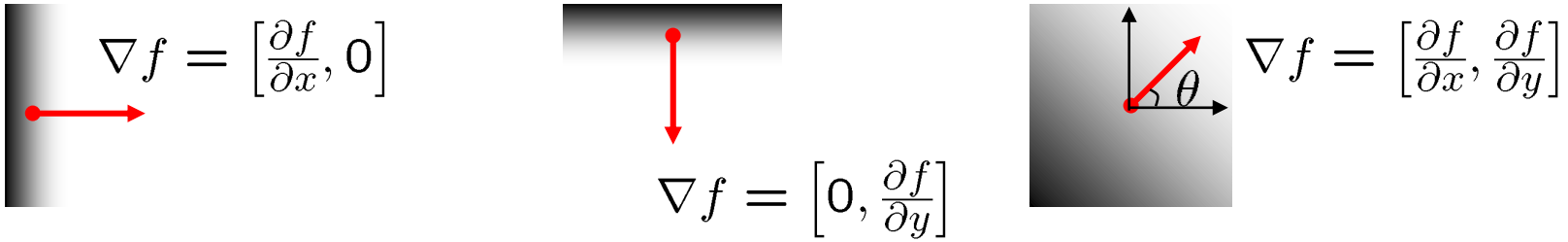


Image Gradient

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



The gradient vector points in the direction of most rapid increase in intensity

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

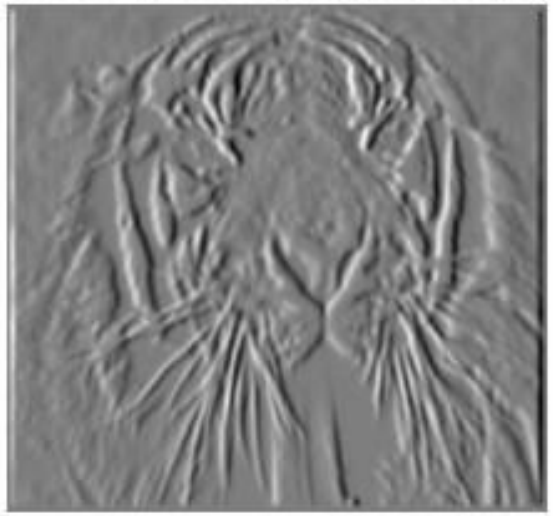
Source: Steve Seitz

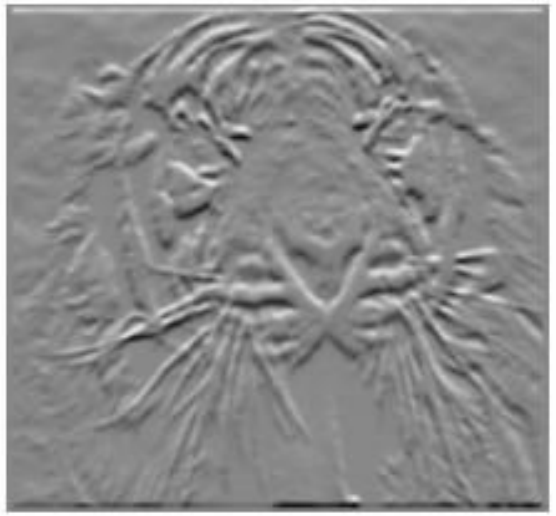
Finite Differences: Example

Original Image

Gradient magnitude

x-direction

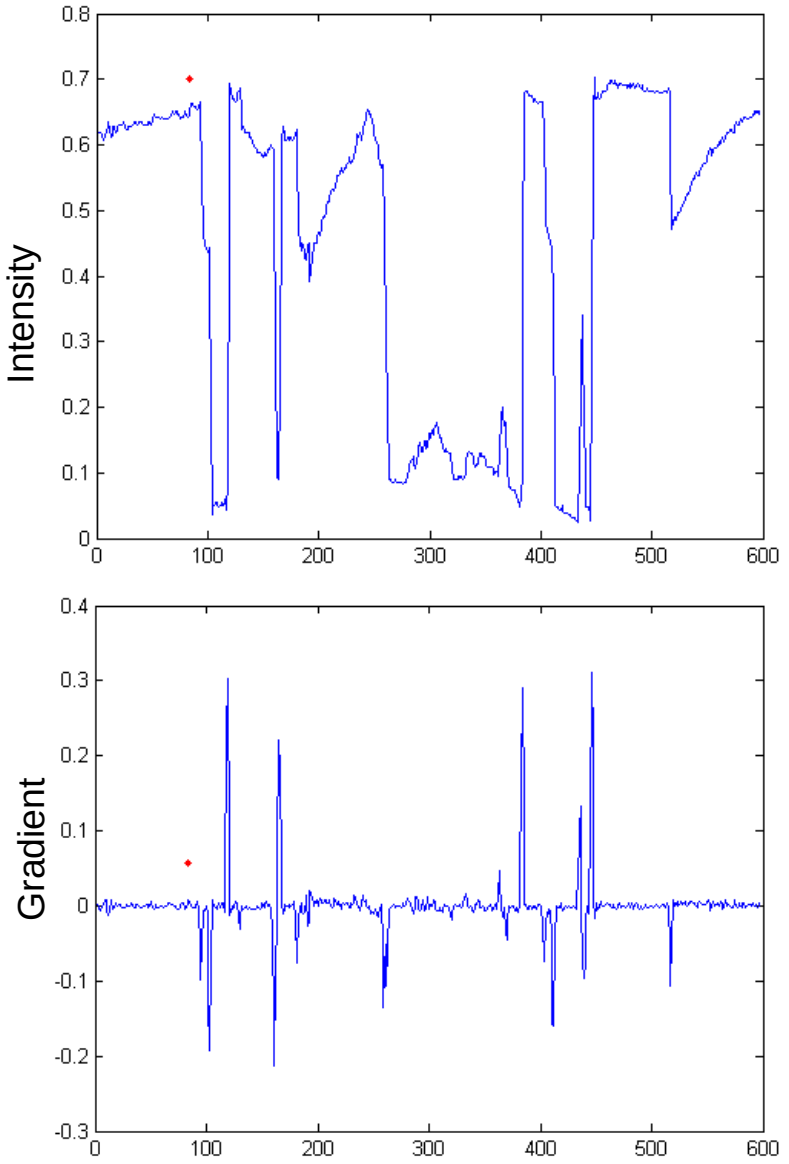




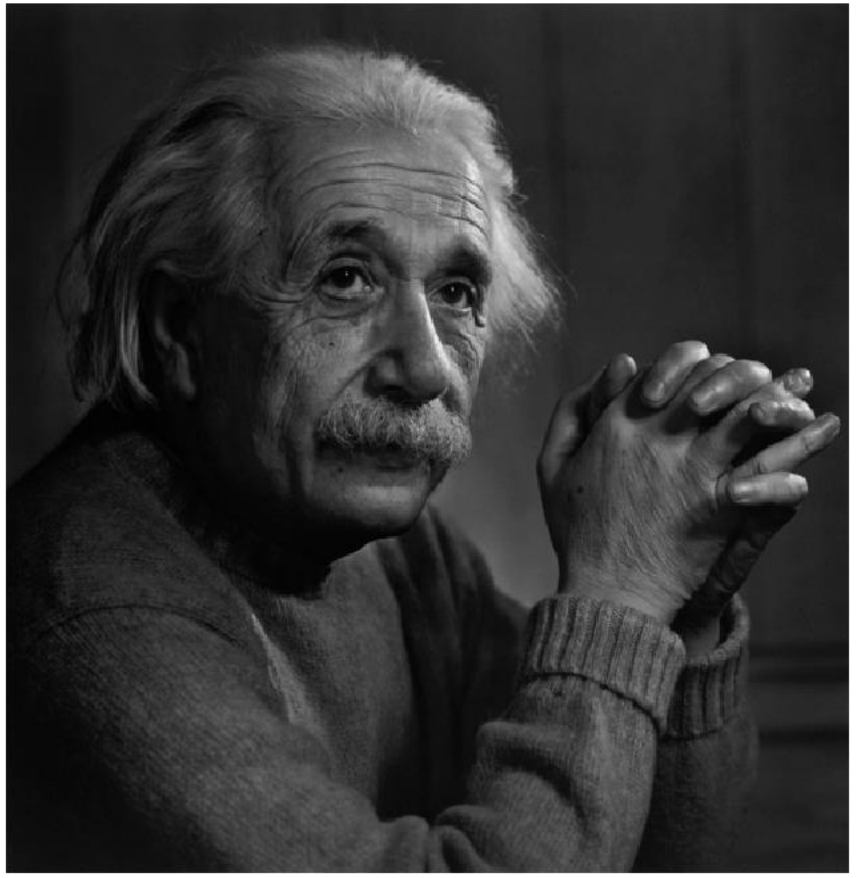
y-direction

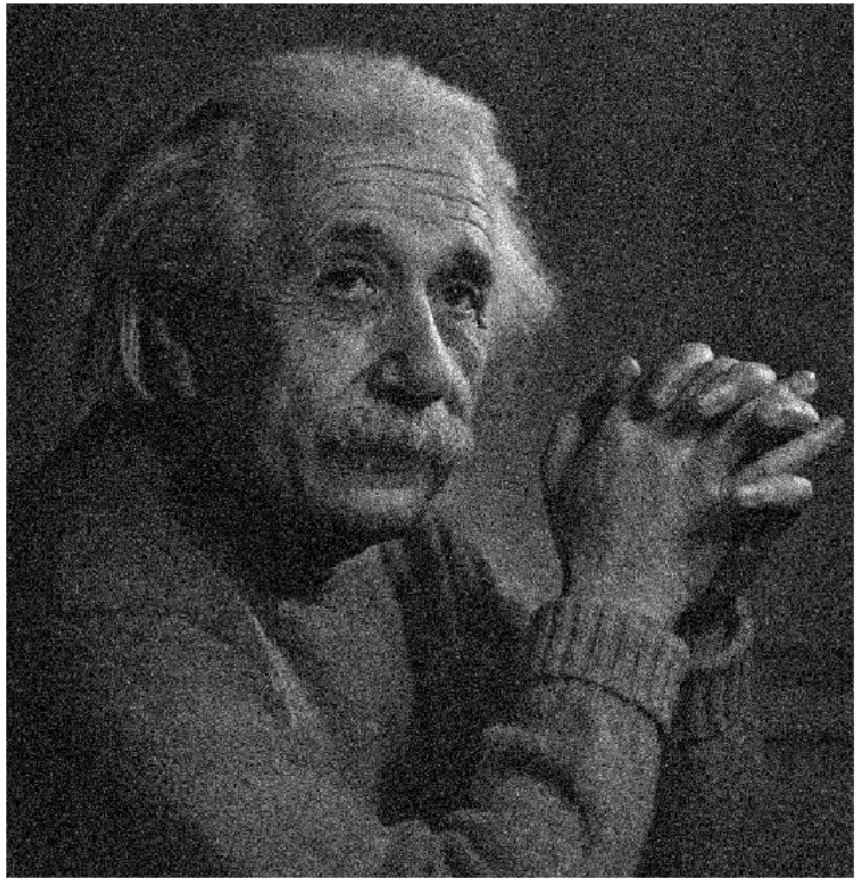
- Which one is the gradient in the x-direction? How about y-direction?

Intensity Profile



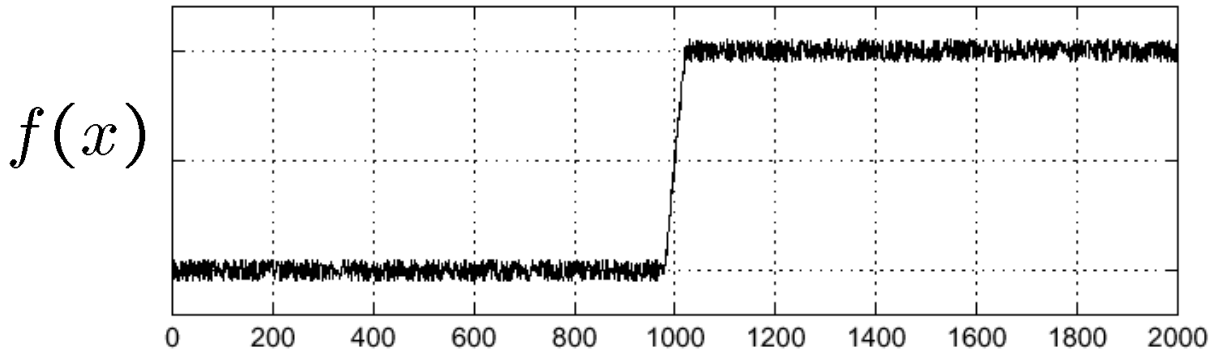
Effects of Noise

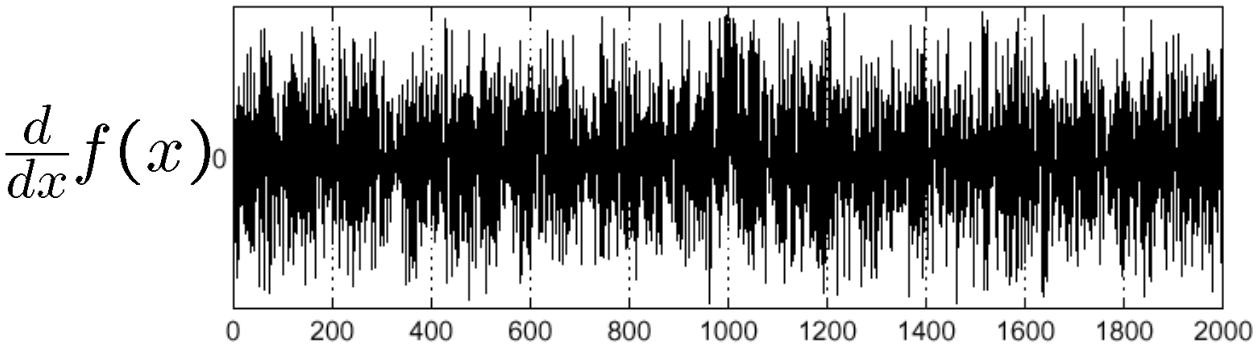




Effects of Noise

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





Where is the edge?

Source: S. Seitz

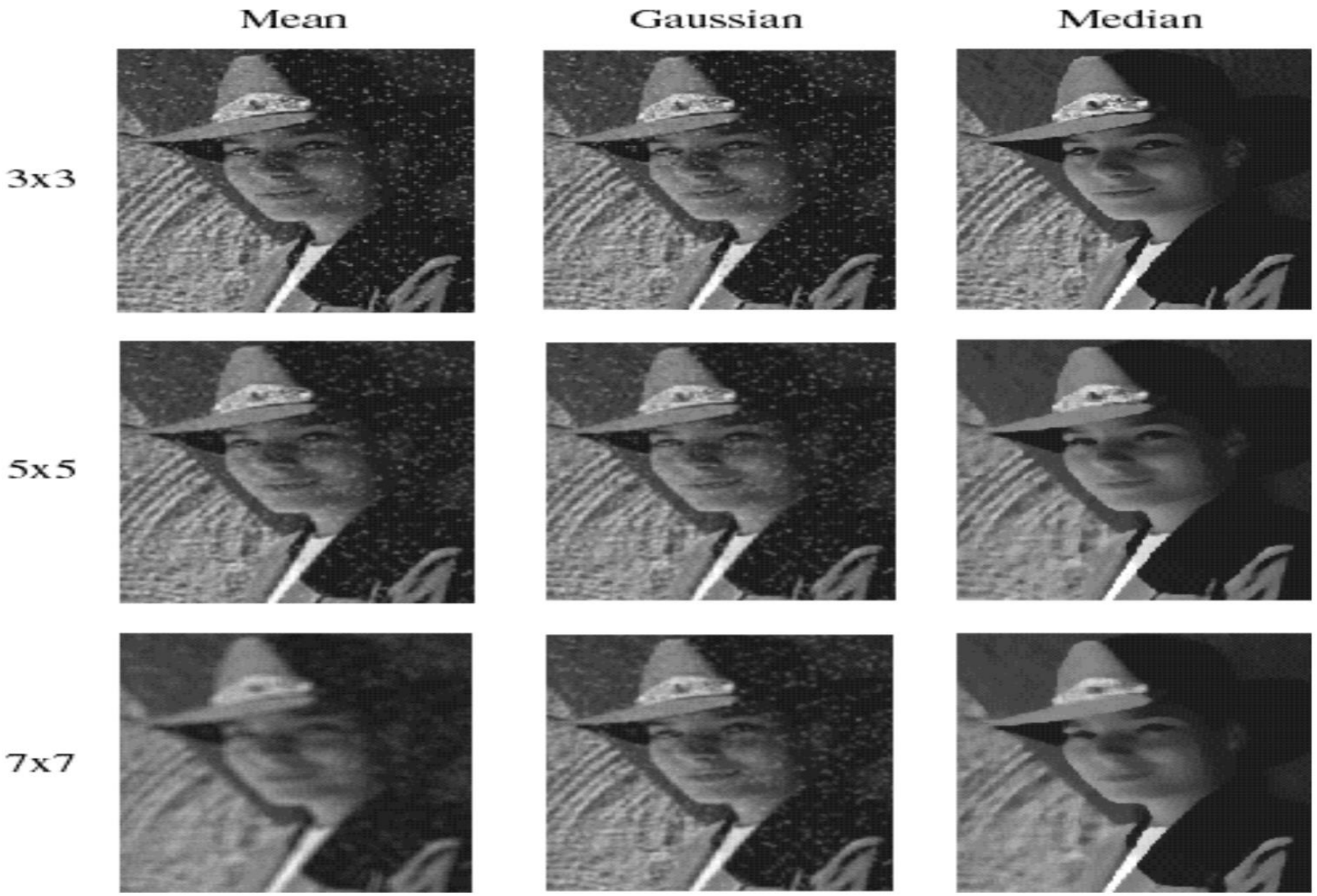
Effects of Noise

- Finite 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

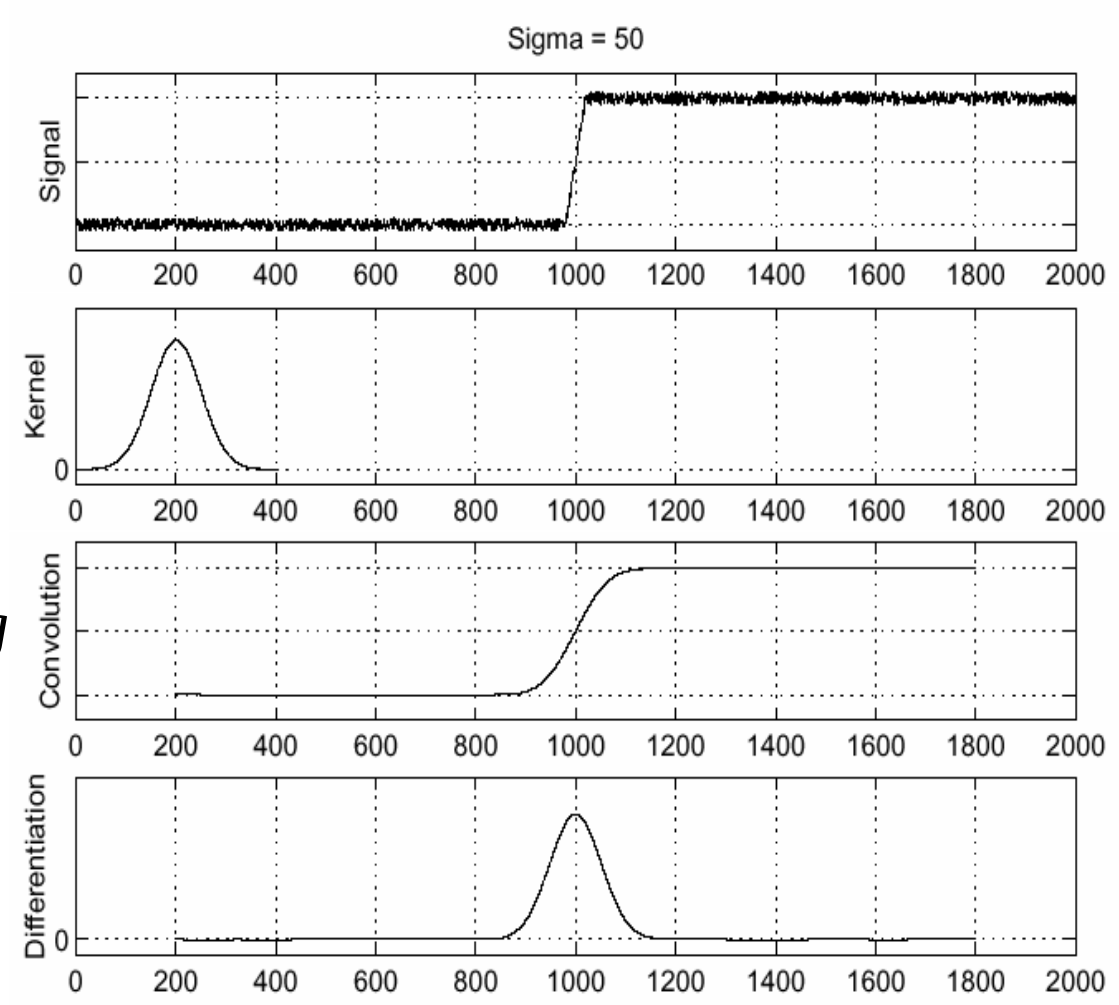
- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

Smoothing with different filters



Solution: Smooth First



f

g

$f * g$

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

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

Source: S. Seitz

Derivative Theorem of Convolution

- This theorem gives us a very useful property:

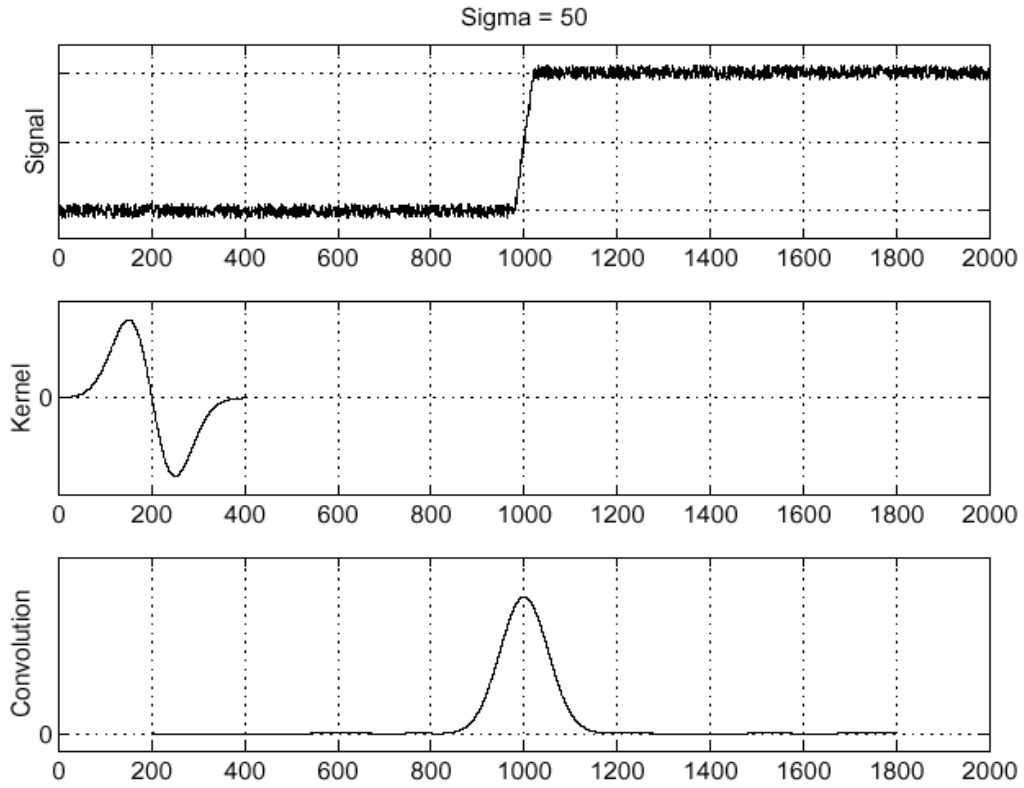
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

$\frac{d}{dx} g$

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



Source: S. Seitz

First Order derivative edge detectors

- Practical filters perform image smoothing first prior to the edge detection
- Example: Sobel Filter

-1	0	1
-2	0	2
-1	0	1

- Effectively does a smoothing followed by a differencing

1
2
1

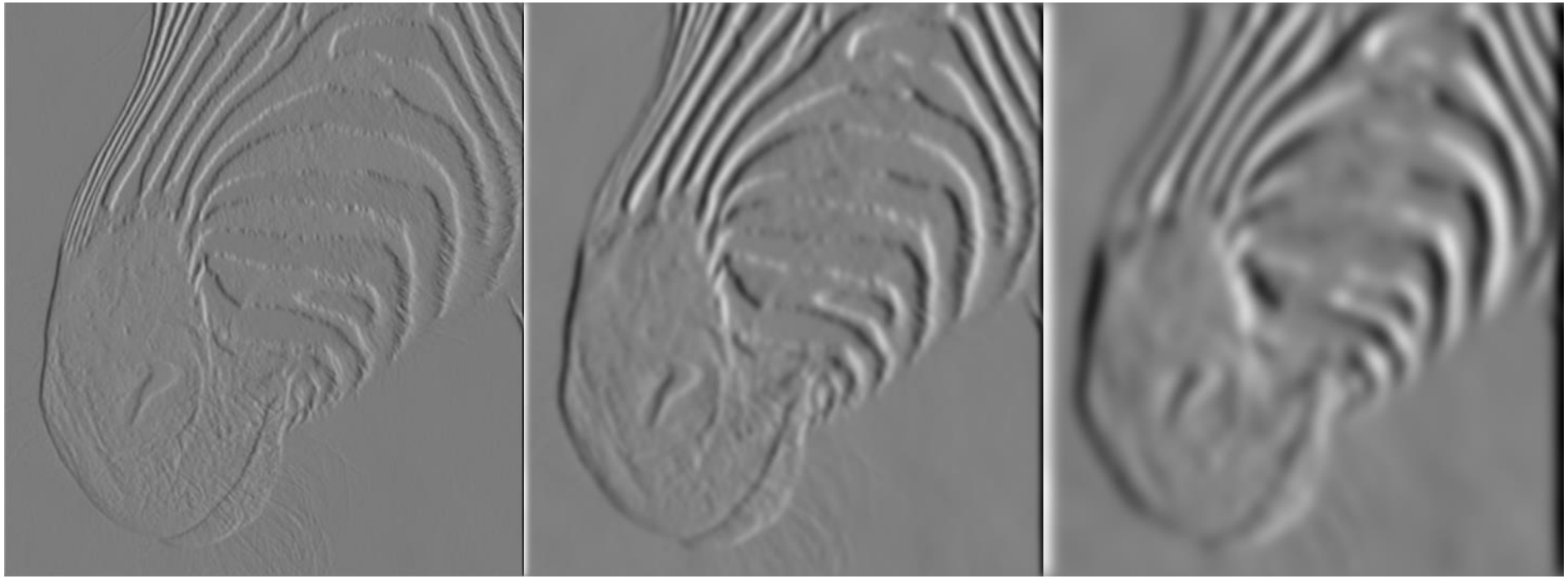
→ Smoothing

-1	0	+1

→ Differencing

Derivative of Gaussian Filter

Tradeoff between smoothing at different scales



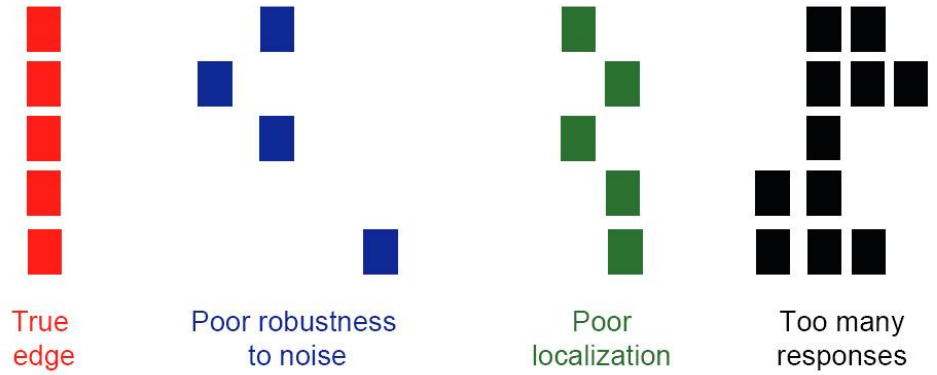
1 pixel 3 pixels 7 pixels

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

Source: D. Forsyth

Designing an edge detector

- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge

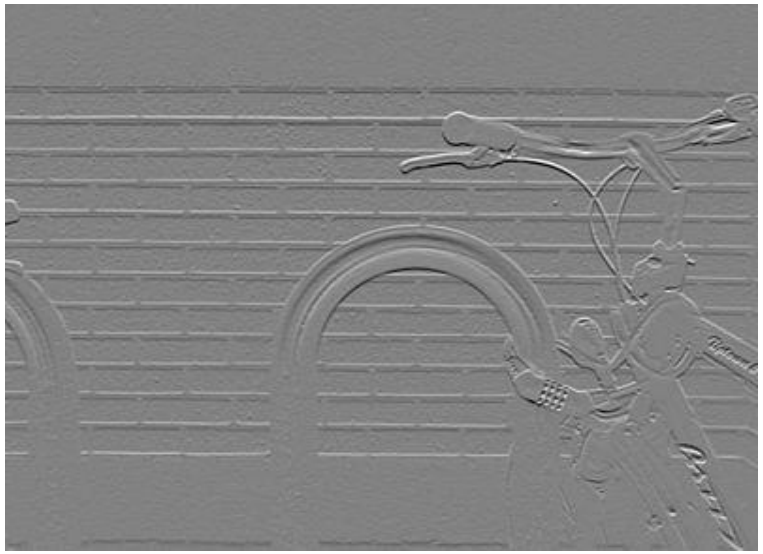


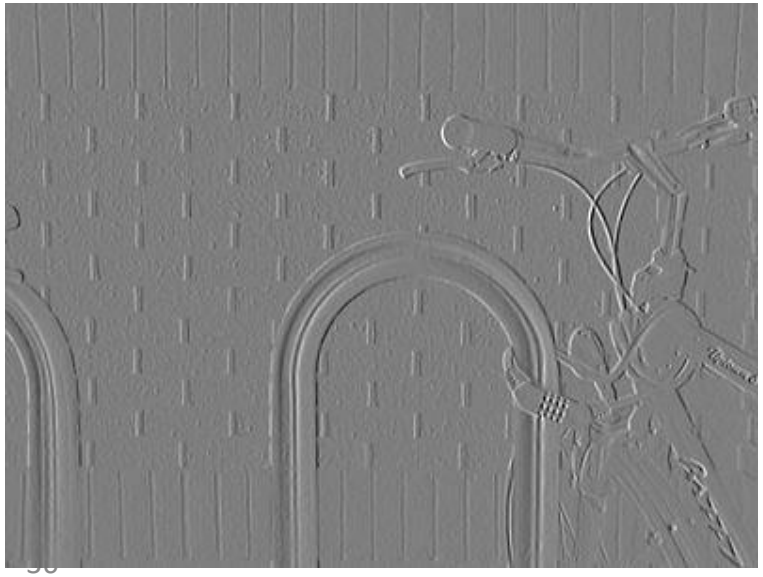
Sobel Operator

- uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives
- one for horizontal changes, and one for vertical

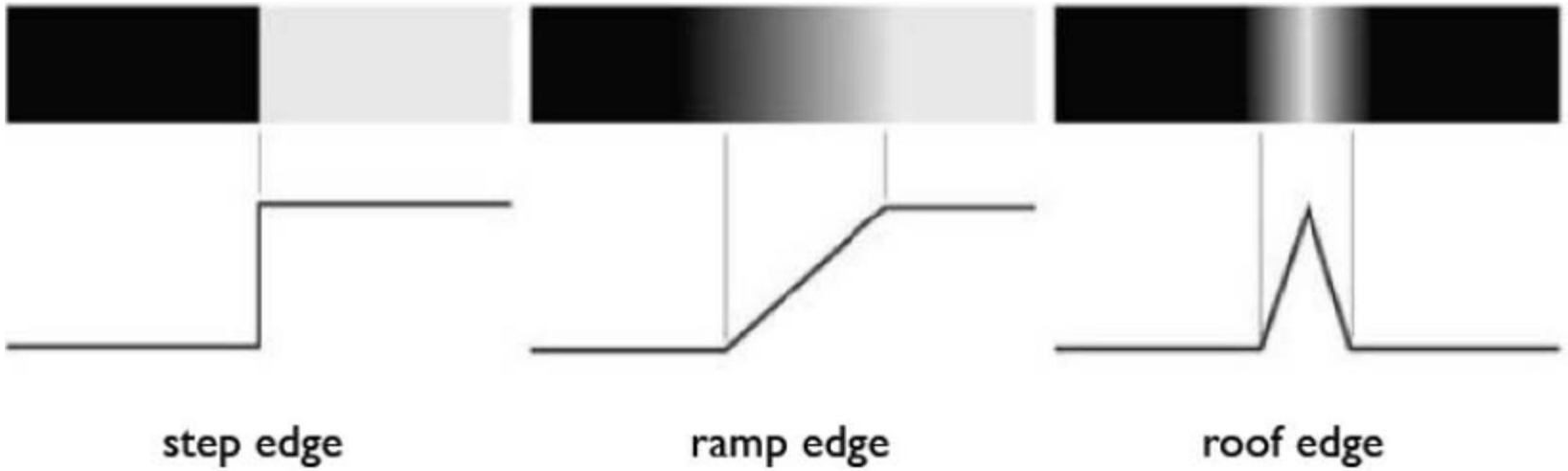
$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel Filter Example





Sobel Operator Limitations



- Poor Localization (Trigger response in multiple adjacent pixels)
- Thresholding value favors certain directions over others
  - Can miss oblique edges more than horizontal or vertical edges
  - False negatives

Canny Edge Detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown 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.

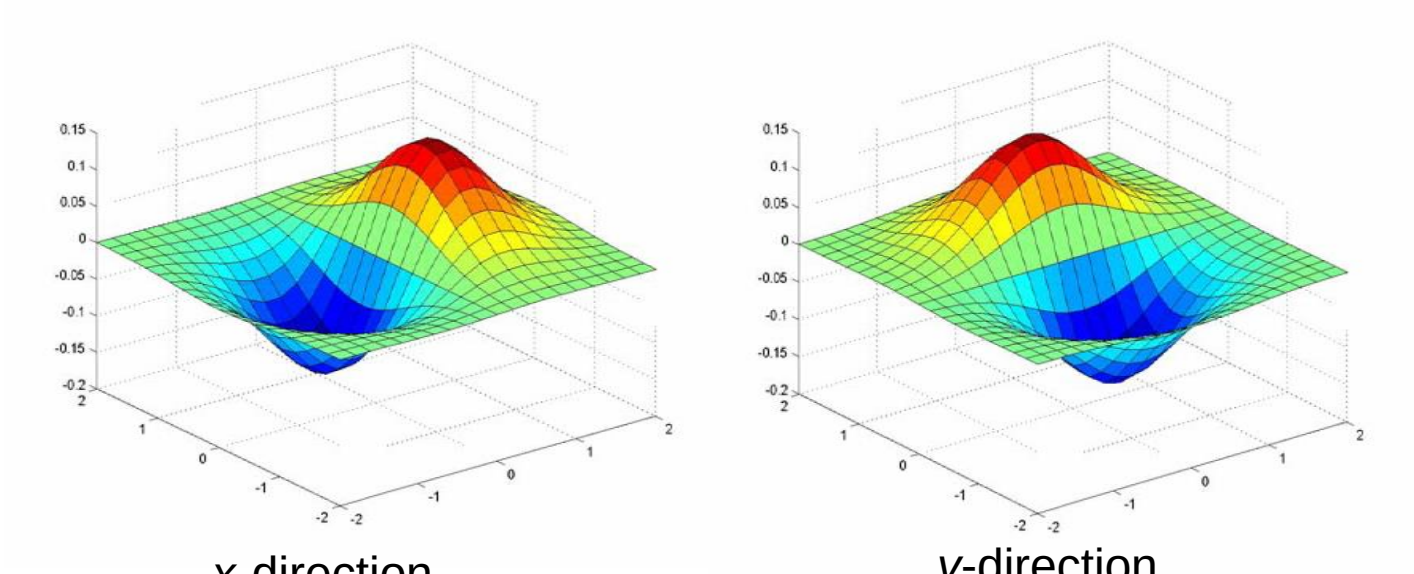
Canny Edge Detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response
- Use hysteresis and connectivity analysis to detect edges

Example

- original image

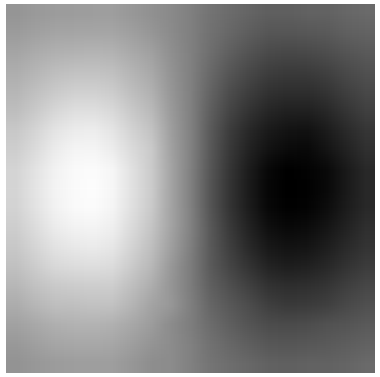
Derivative of Gaussian Filter

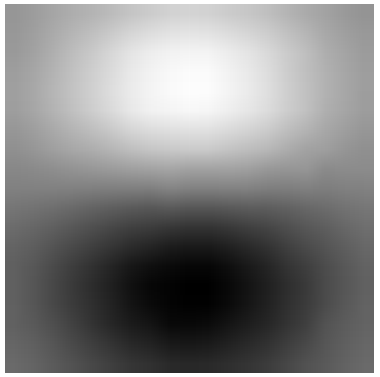


x-direction

y-direction

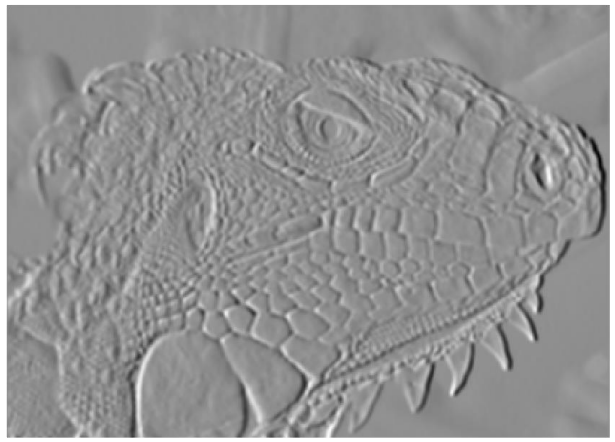
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



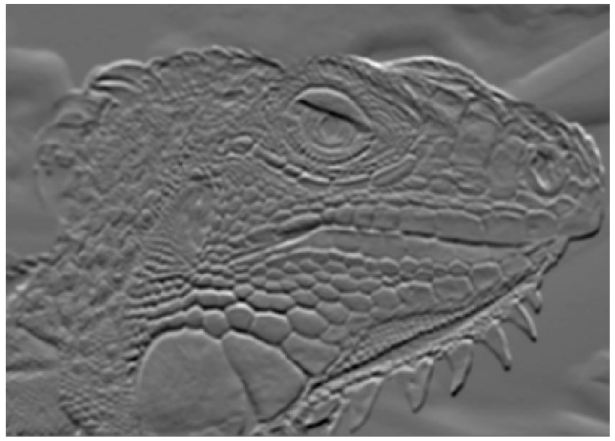


$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Compute Gradients-Difference of Gaussian



X-Derivative of Gaussian

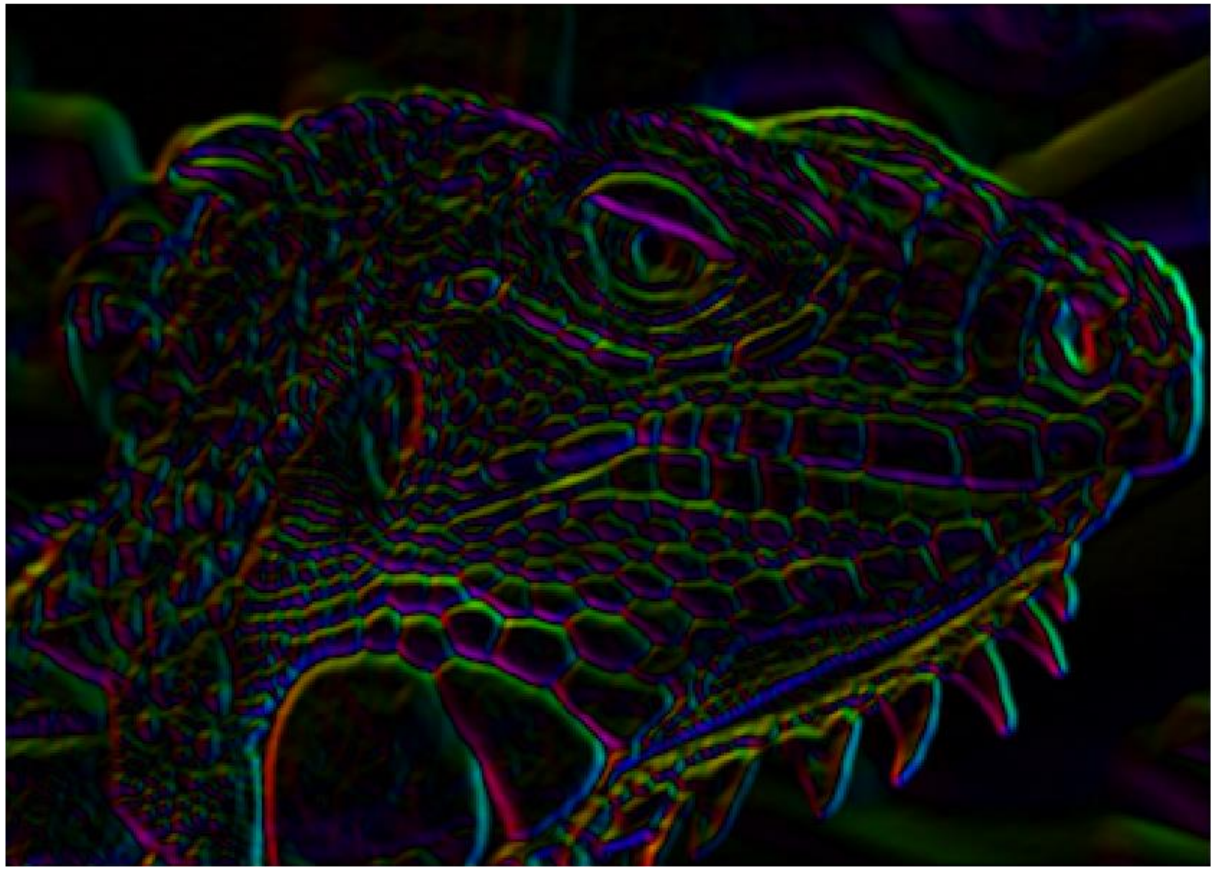


Y-Derivative of Gaussian

Gradient Magnitude

Source: J. Hayes

Get Orientation at each pixel



$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

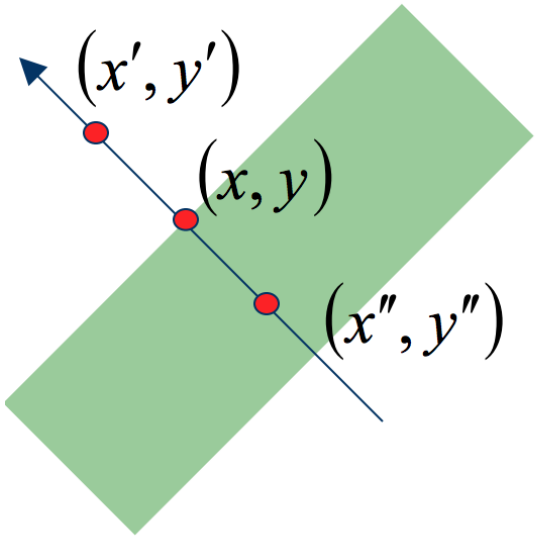
Source: J. Hayes

Canny Edge Detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response

Remove Spurious Gradients

$|\nabla G|(x, y)$ is the gradient at pixel (x, y)

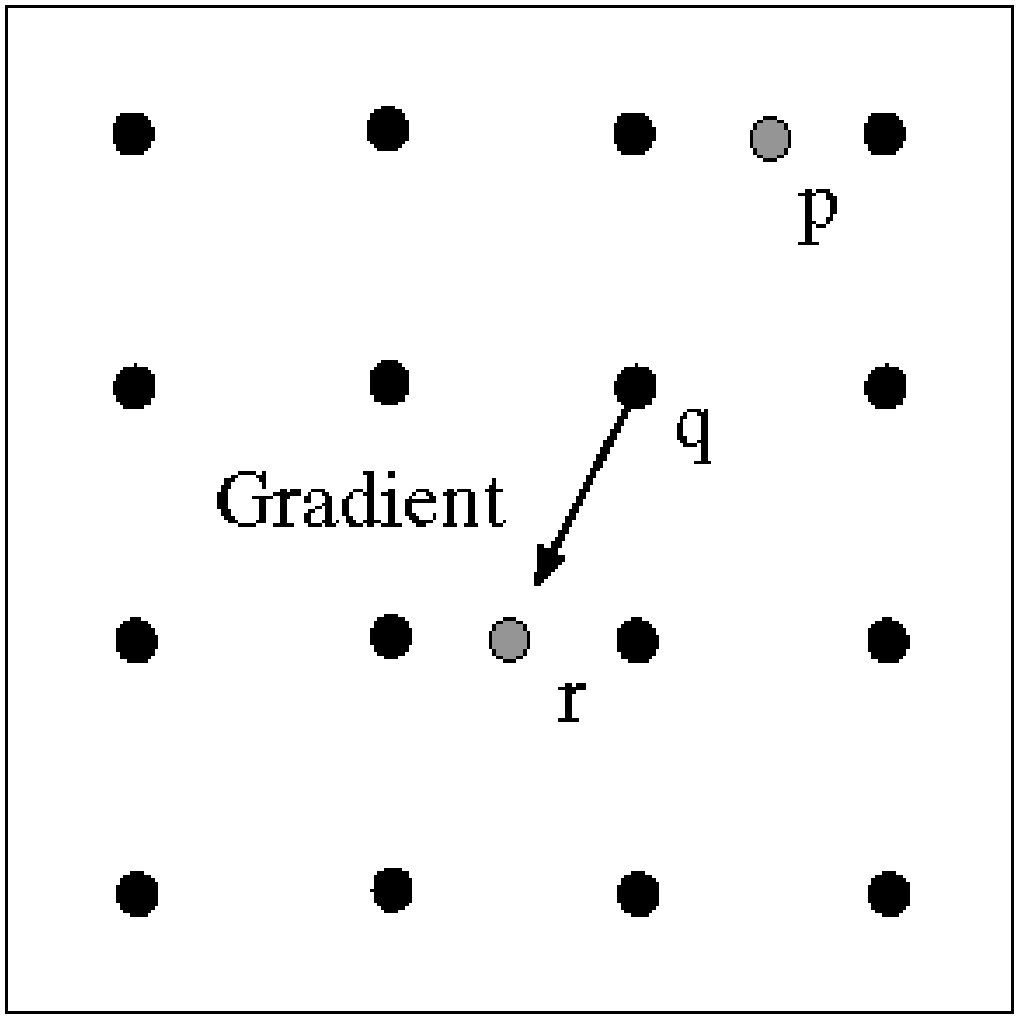


$$M(x,y) = \begin{cases} |\nabla G|(x,y) \text{ if } |\nabla G|(x,y) > |\nabla G|(x',y') \\ \qquad\qquad \& |\nabla G|(x,y) > |\nabla G|(x'',y'') \\ 0 \qquad\qquad \text{otherwise} \end{cases}$$

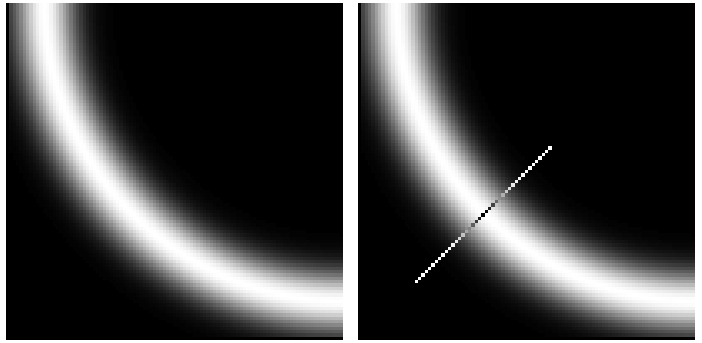
x' and **x**" are the neighbors of **x** along normal direction to an edge

Alper Yilmaz, Mubarak Shah Fall 2012, UCF

Non-Maximal Suppression



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



Source: D. Forsyth

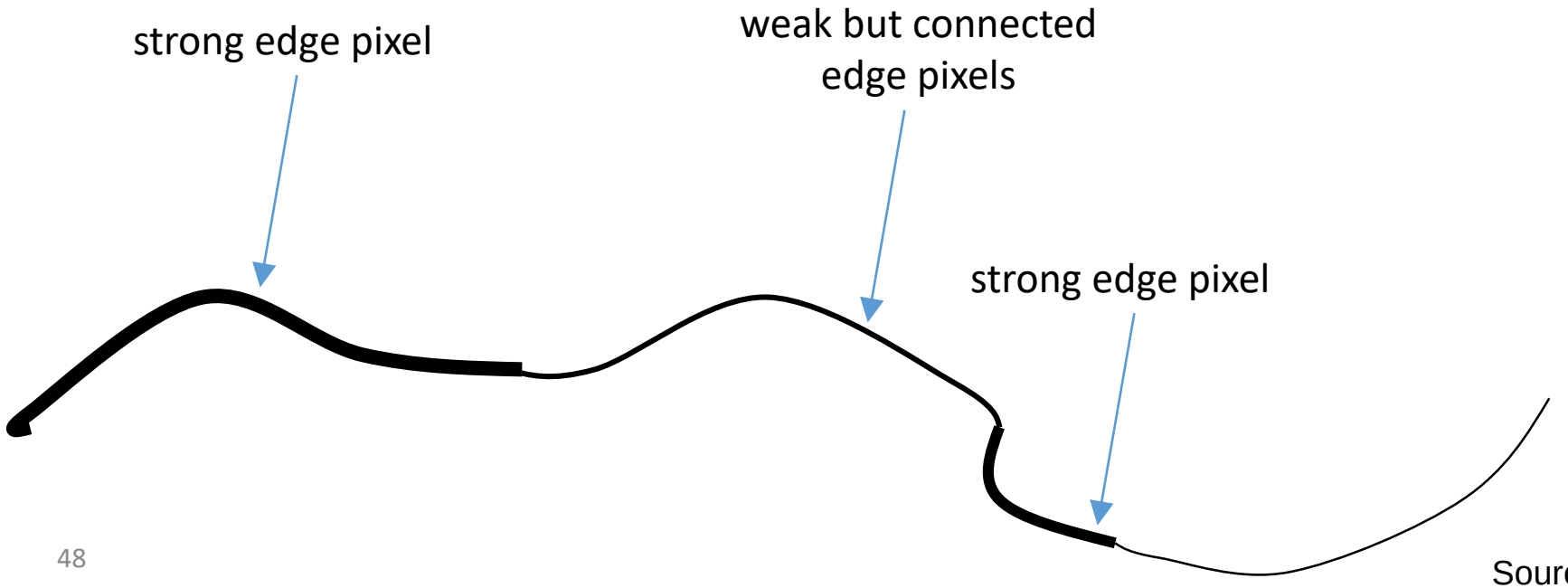
Non-Max Suppression

Before

After

Hysteresis Thresholding

- Avoid streaking near threshold value
- Define two thresholds: Low and High
  - If less than Low, not an edge
  - If greater than High, strong edge
  - If between Low and High, weak edge
    - Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to an ‘strong edge pixel’ directly or via pixels between Low and High



Source: S. Seitz

Final Canny Edges

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” down to single pixel width
4. Thresholding and linking (hysteresis):
   - 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 spread/size)

original

Canny 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

Source: S. Seitz

Gradients (e.g. Canny)

Color

Texture

Combined

Human

Edge Detection Python

- import numpy as np
- import cv2
- image = cv2.imread('image path')
- image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
- cv2.imshow("Original",image)
- lap =cv2.Laplacian(image, cv2.CV_64F)
- lap = np.uint8(np.absolute(lap))
- cv2.imshow("Laplacian",lap)
- cv2.waitKey(0)

Edge Detection Python Sobel

- import numpy as np
- import cv2
- image = cv2.imread('image path')
- image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
- cv2.imshow("Original",image)
- sobelX = cv2.Sobel(image, cv2.CV_64F, 1, 0)
- sobelY = cv2.Sobel(image, cv2.CV_64F, 0, 1)
- sobelX = np.uint8(np.absolute(sobelX))
- sobelY = np.uint8(np.absolute(sobelY))
- sobelCombined = cv2.bitwise_or(SobelX, SobelY)

- cv2.imshow("Sobel X",sobelX)
- cv2.imshow("Sobel Y", sobelY)
- cv2.imshow("Sobel Combined", sobelCombined)
- cv2.waitKey(0)

Use cv2.canny for canny edge detector

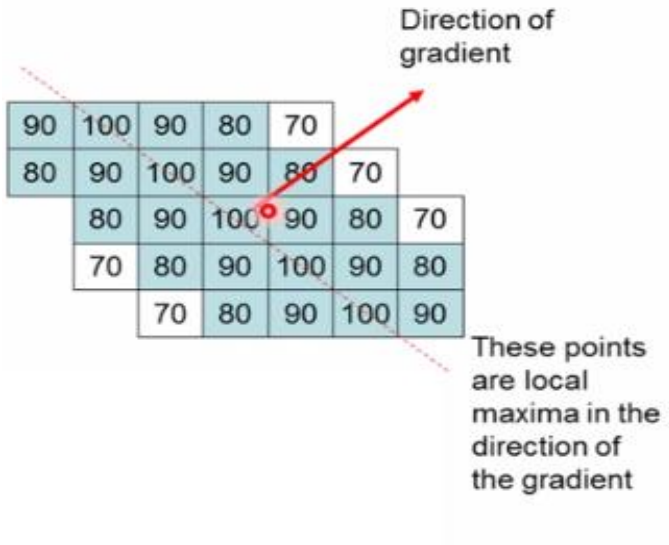
Edge Detection MATLAB

- Gradient
- I=double(rgb2gray(imread('p1.jpg')));
- [fx,fy]=gradient(I);
- fmag = (fx.^2 + fy.^2).^0.5;
- Imshow(fmag, [])

Edge Detection MATLAB

- E = edge(uint8(I),’sobel’);
- E = edge(uint8(I),’canny’);
- E = edge(uint8(I),’Laplacian’);

Edge Detection MATLAB (Edge Thining)



$$\theta = \tan^{-1}\left[\frac{\partial f/\partial y}{\partial f/\partial x}\right]$$

```
ang = atan2(fy,fx);
angThresh = ang.*double(E);
ang2 = floor(8*(angThresh+pi)/(2*pi));
figure, imshow(ang2*32,[]);
colormap jet
```

Edge Detection MATLAB

- Visualizing Scale Difference
- for sigma=0.2:0.4:20
- E = edge(I,'log', 0, sigma); % zero means use thresh = 0
- imshow(E, []), title(sprintf('sigma = %f', sigma));
- pause
- end

Edge Detection MATLAB

```
Hysteresis Thresholding
I=imread('house.jpg');
for sigma=0.5:0.5:5
   E=edge(I,'canny',[],sigma);
   imshow(E);
   title(sprintf('sigma=%f',sigma));
   pause
end

for tHigh=0.05:0.05:0.4
   E1=edge(I,'canny',[0.4*tHigh tHigh], 1.5);
   imshow(E1);
   title(sprintf('tHigh=%f',tHigh));
   pause;
end

[E2, thresh]=edge(I,'canny',[],1.5);
figure, imshow(E2), title('dual thresh');
E3=edge(I,'canny',[thresh(1) thresh(1)+eps],1.5);
figure, imshow(E3), title('low thresh only');
E4=edge(I,'canny',[thresh(2) thresh(2)+eps],1.5);
figure, imshow(E4), title('high thresh only');
```

Gaussian Mask Construction

- For example, G(-l,-l) = $e^{-[(-1)^2 + (-1)^2] / 2(1)^2} =$ 0.367879
- Similarly G (0, 1) = 0.60

0	0	2	3	2	0	0
0	5	21	35	21	5	0
2	21	94	153	94	21	2
3	35	153	255	153	35	3
2	21	94	153	94	21	2
0	5	21	35	21	5	0
0	0	2	3	2	0	0

Gaussian Mask contd

The mask of the Gaussian filter for $\sigma = 2$ multiplied by 255 and truncated to the nearest integer is shown in Figure:

0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	3	11	26	50	73	82	73	50	26	11	3	0
1	6	20	50	93	136	154	136	93	50	20	6	1
2	9	30	73	136	198	225	198	136	73	30	9	2
2	11	34	82	154	225	255	225	154	82	34	11	2
2	9	30	37	136	198	225	198	136	73	30	9	2
1	6	20	50	93	136	154	136	93	50	20	6	1
0	3	11	26	50	73	82	37	50	26	11	3	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0

- *Questions*
  - *Feel Free to ask*