Image Gradients and Edge Detection

Dr. Mohamed Waleed Fakhr

2023

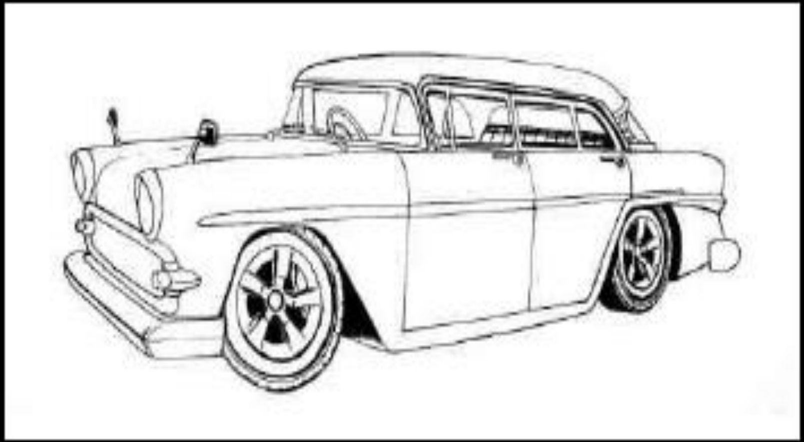
Edges

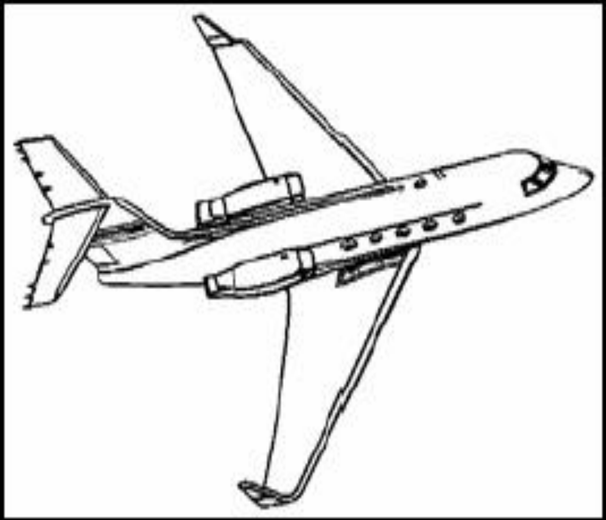
Edge detection

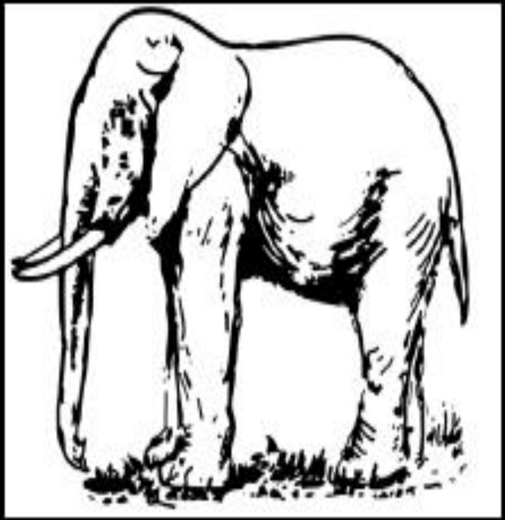
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

Source: D. Lowe

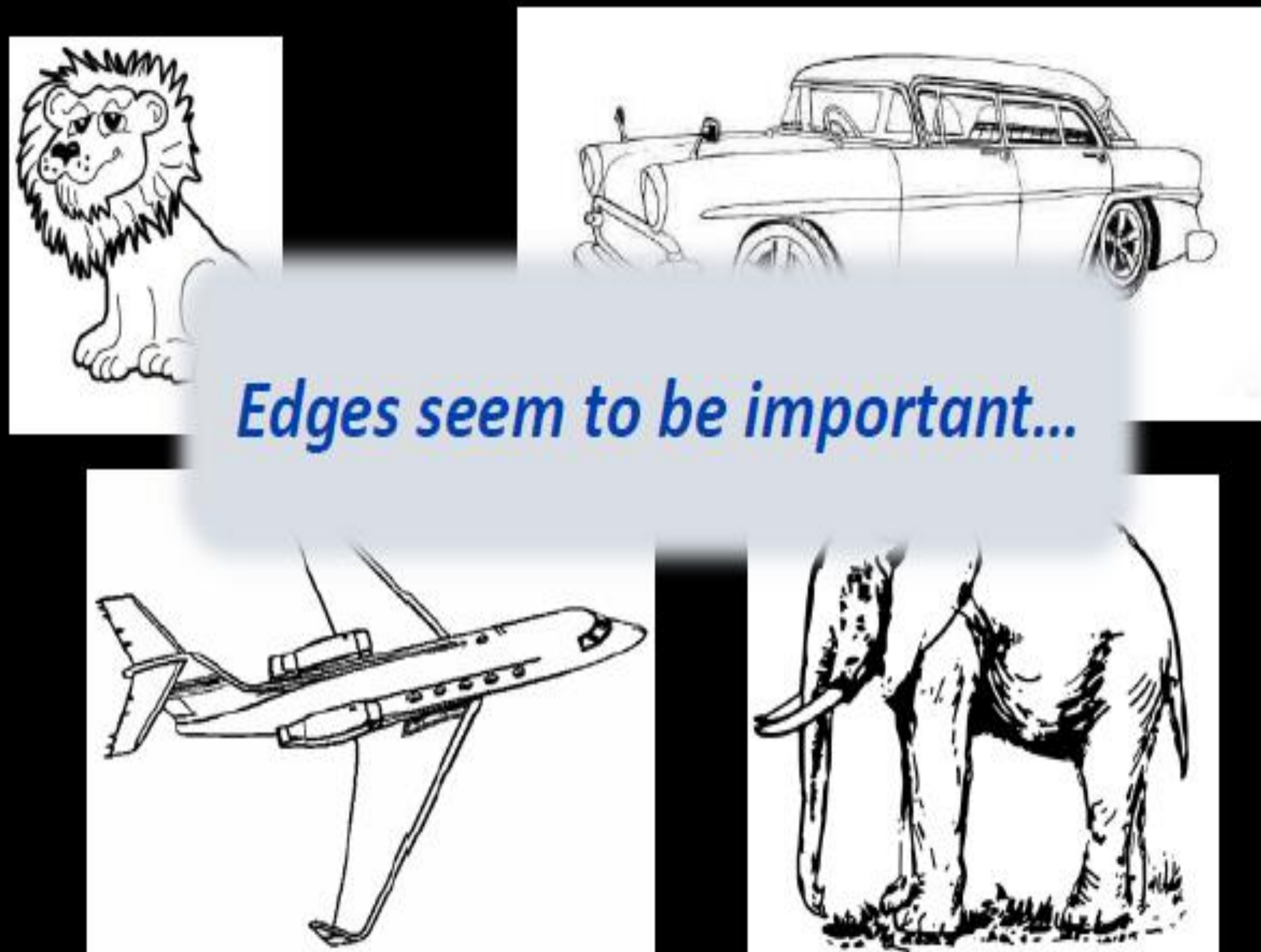
Reduced images







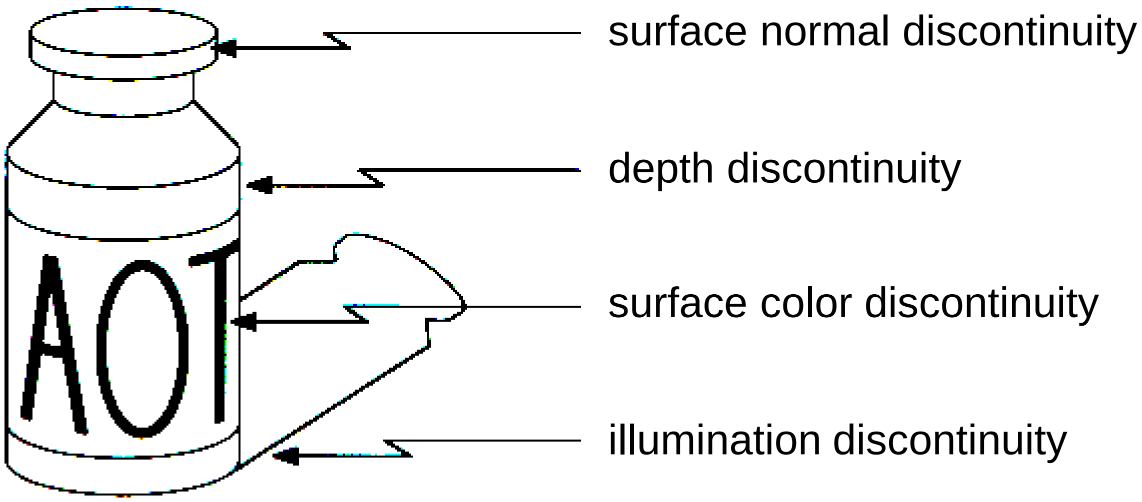
Reduced images



Edges seem to be important…

Origin of edges

Edges are caused by a variety of factors:



Source: Steve Seitz

Edge Detection

Basic idea: look for a neighborhood with strong signs of change.

Problems:

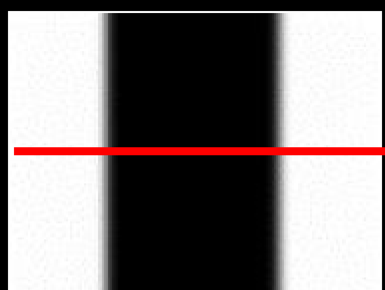
- neighborhood size
- how to detect change

81	82	26	24
82	33	25	25
81	82	26	24

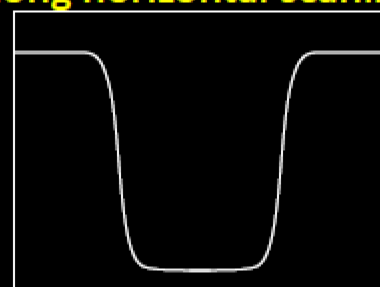
Derivatives and edges

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

image



intensity function
(along horizontal scanline)



Source: S. Lazebnik

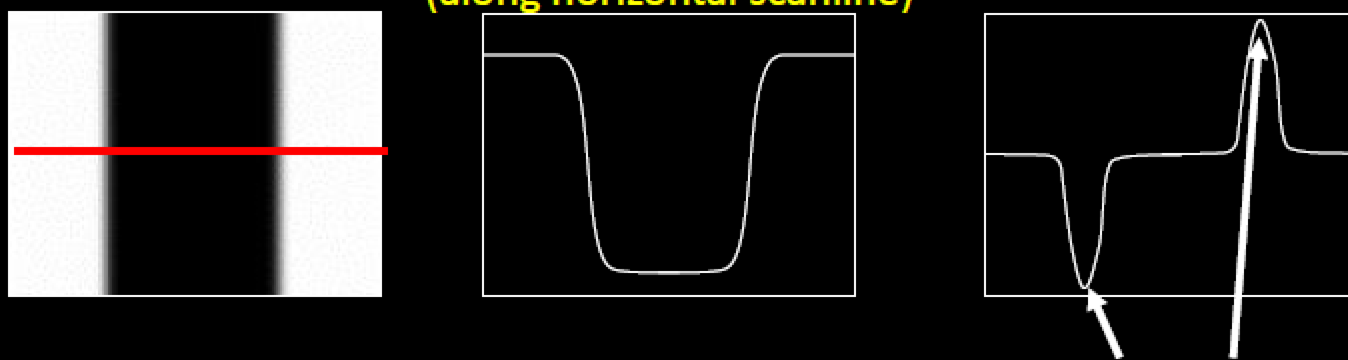
Derivatives and edges

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

Source: S. Lazebnik

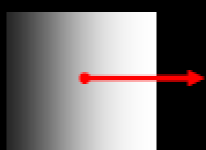
Differential Operators

- Differential operators –when applied to the image returns some derivatives.
- Model these "operators" as masks/kernels that compute the image gradient function.
- Threshold the this gradient function to select the edge pixels.
- Which brings us to the question:

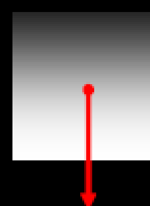
What’s a gradient?

Image gradient

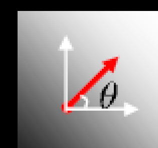
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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

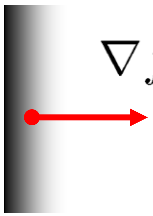


$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

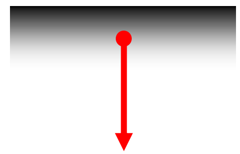
The gradient points in the direction of most rapid increase in intensity

Image gradient

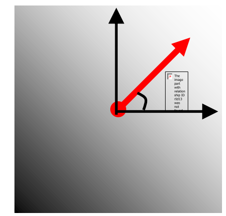
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

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

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

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

Differentiation and convolution

Recall, for 2D function, f(x,y):

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon} \right)$$

This is linear and shift invariant, so must be the result of a convolution.

We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

(which is obviously a convolution)

-1	**1**

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$

-1	1

(correlation filters)

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$

-1	1

(correlation filters)

$$\frac{\partial f(x,y)}{\partial y}$$

$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$? or $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

Finite difference filters

Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$

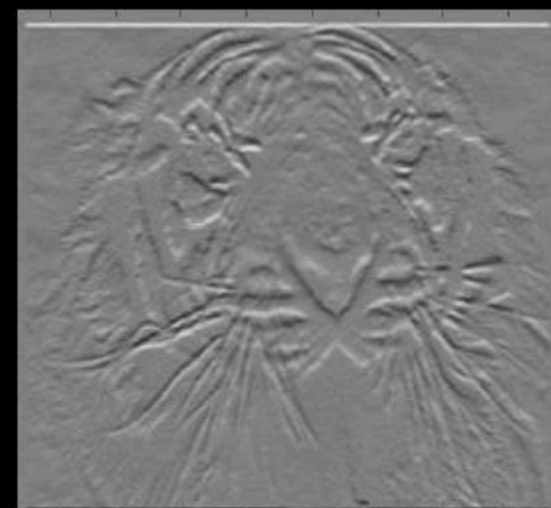
Sobel: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$

Roberts: $M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$

Source: K. Grauman

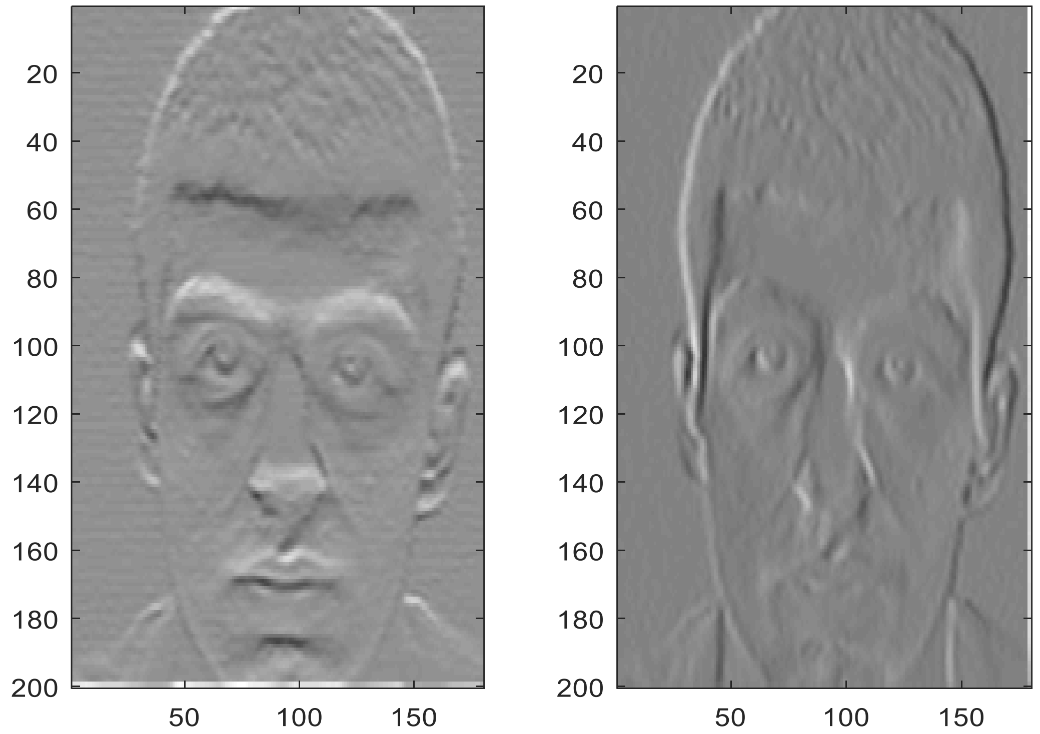
Matlab does gradients

```
filt = fspecial('sobel')

filt =

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

outim = imfilter(double(im),filt);
imagesc(outim);
colormap gray;
```

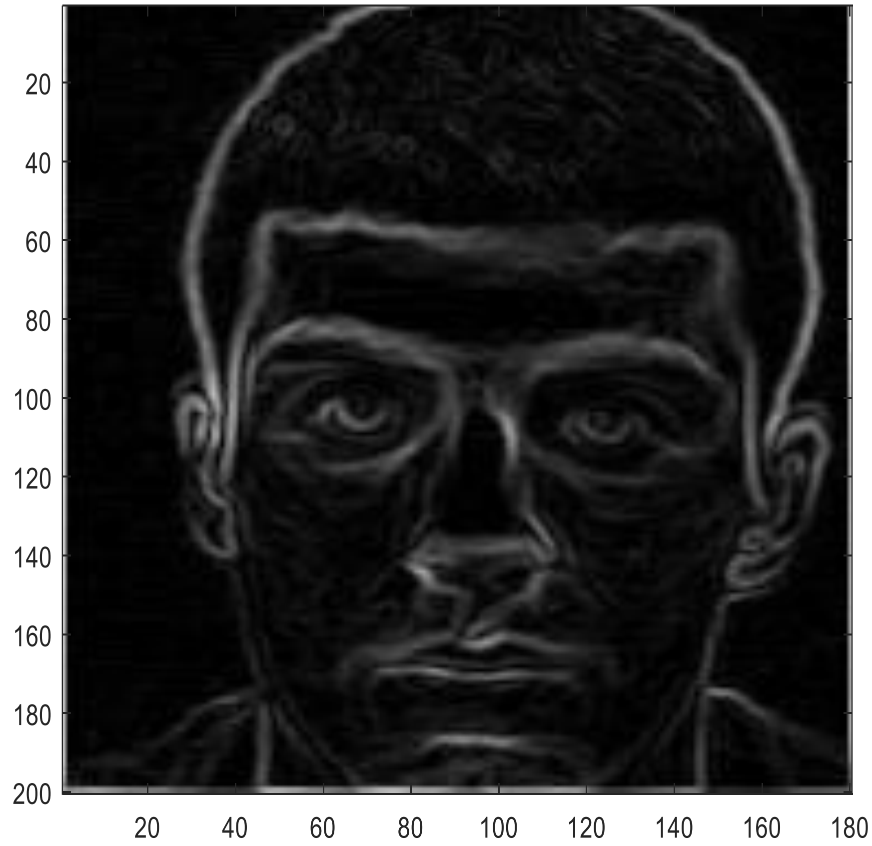


Matlab Example

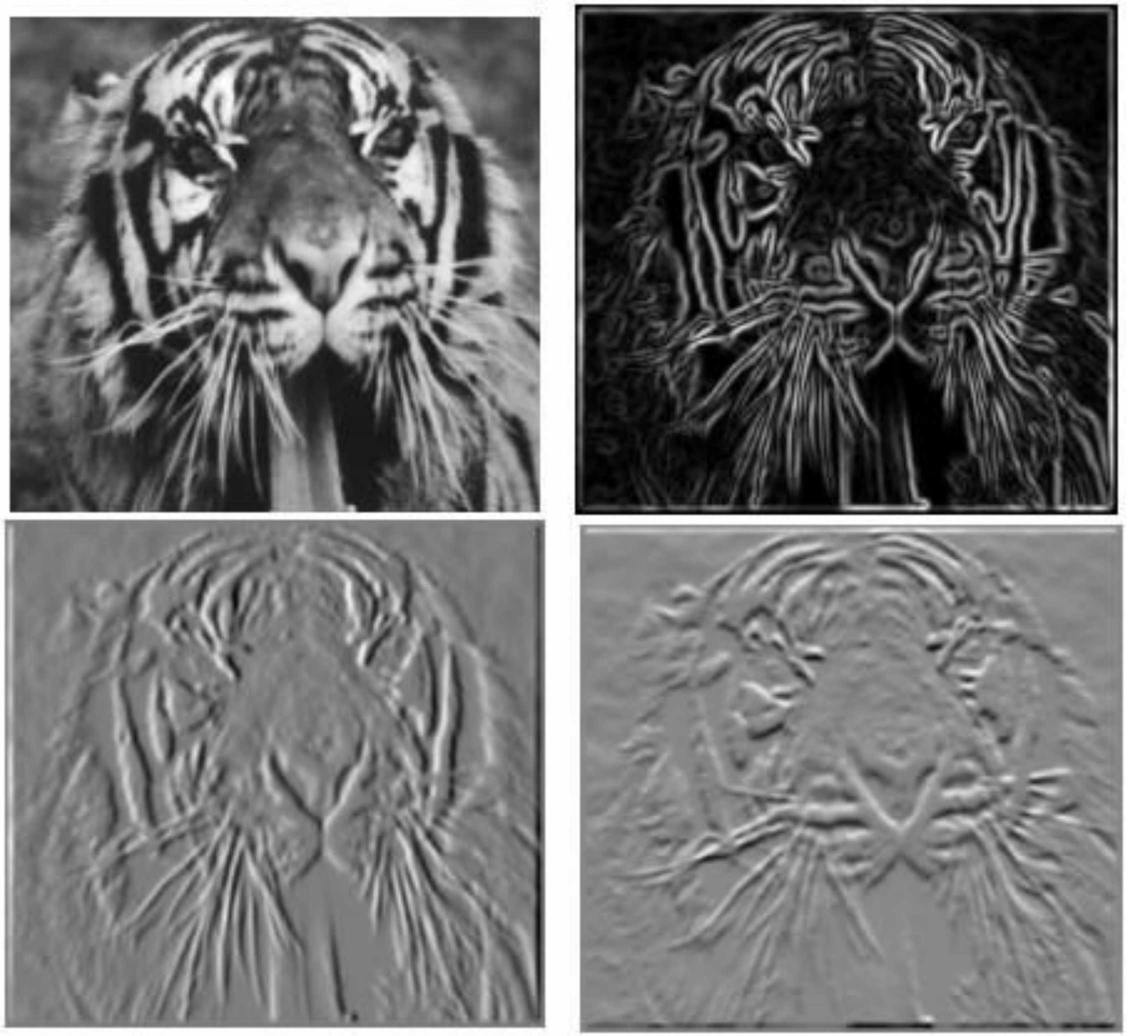
```
clear all,

x = imread('person.bmp');
xg = rgb2gray(x);

filt1 = fspecial('sobel');    %the vertical-gradient filter
filt2 = filt1';               %the horizontal-gradient filter

XV = imfilter(double(xg), filt1);
XH = imfilter(double(xg), filt2);

figure(),
subplot(1,2,1), imagesc(XV);
subplot(1,2,2), imagesc(XH);
colormap gray;
```



```
%Now we get the gradient strength at each point:
LL = size(xg);
L1=LL(1);
L2=LL(2);

for i=1:L1
    for j=1:L2
        XX(i,j) = sqrt(XH(i,j)^2 + XV(i,j)^2);
    end
end

figure, colormap gray,
imagesc(XX);
```



Finite differences: example

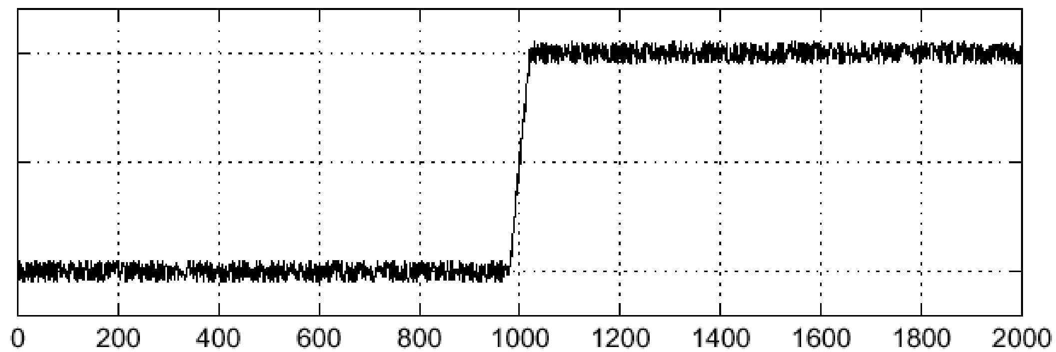


Which one is the gradient in the x-direction (resp. y-direction)?

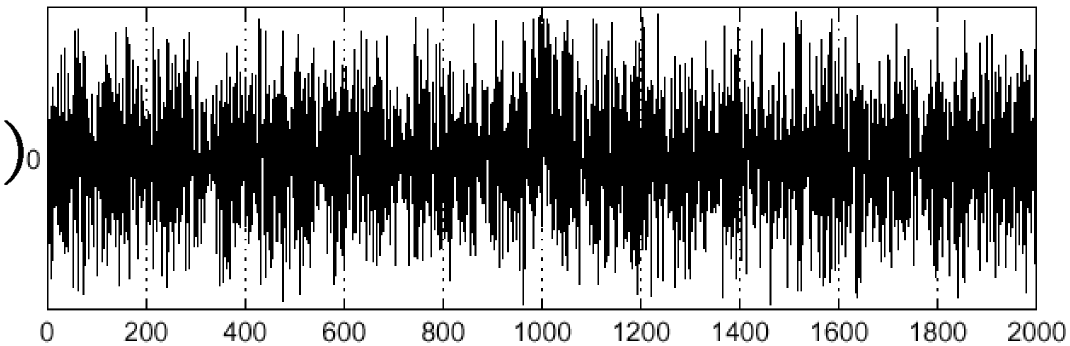
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



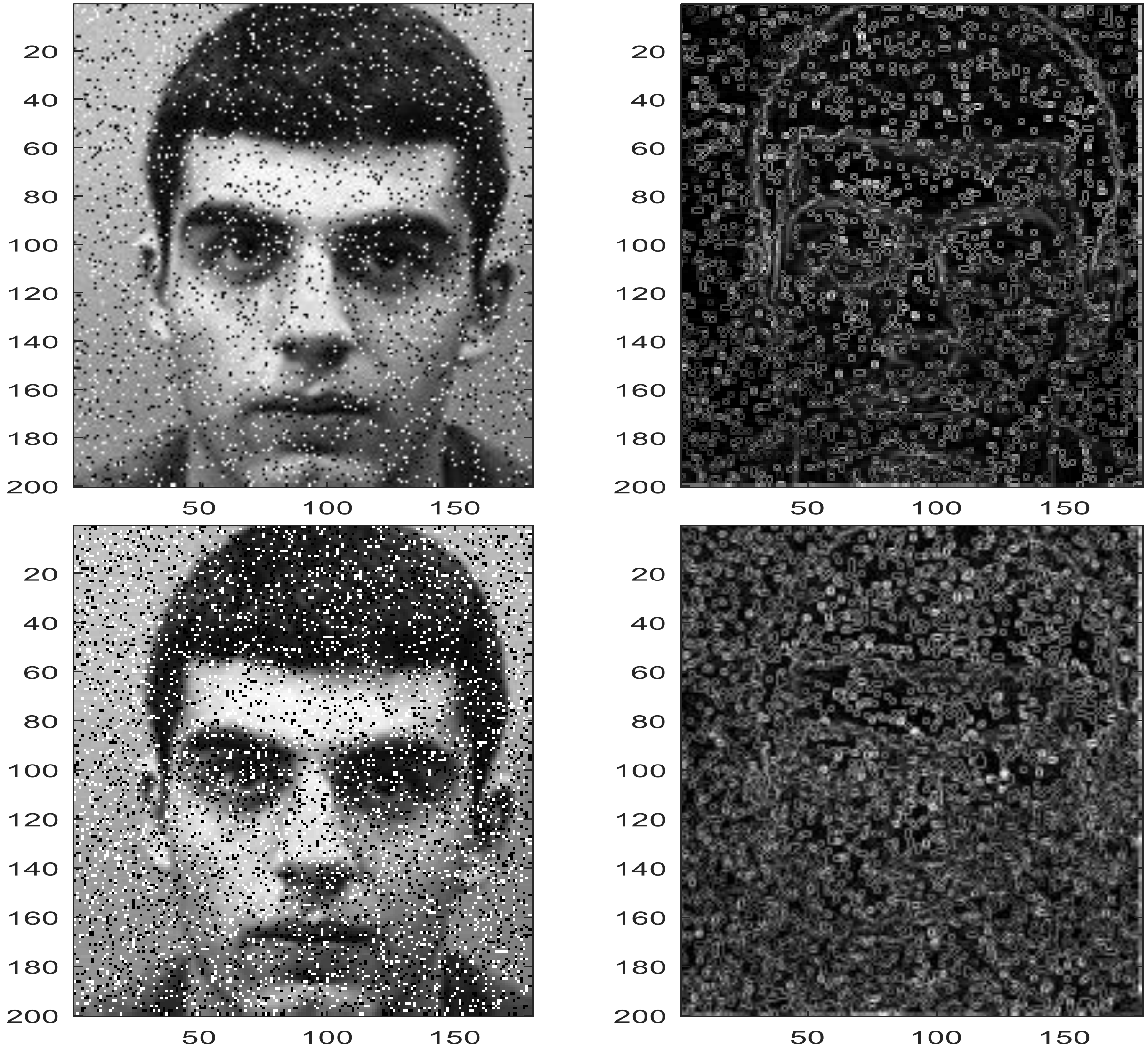
$$\frac{d}{dx}f(x)$$



Where is the edge?

Source: S. Seitz

When noise is added : top 0.1, bottom 0.2

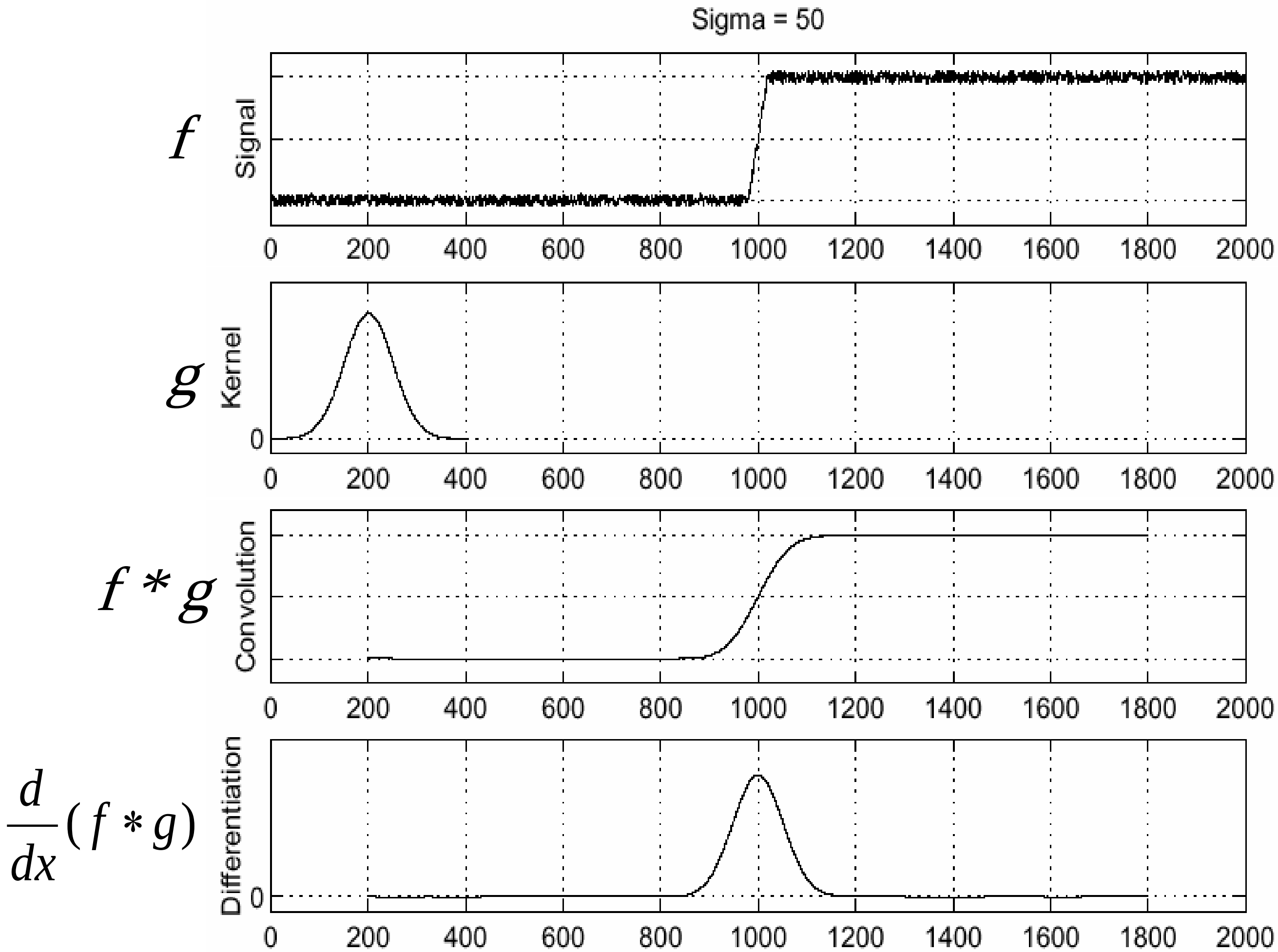


Effects of noise

- 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 from their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

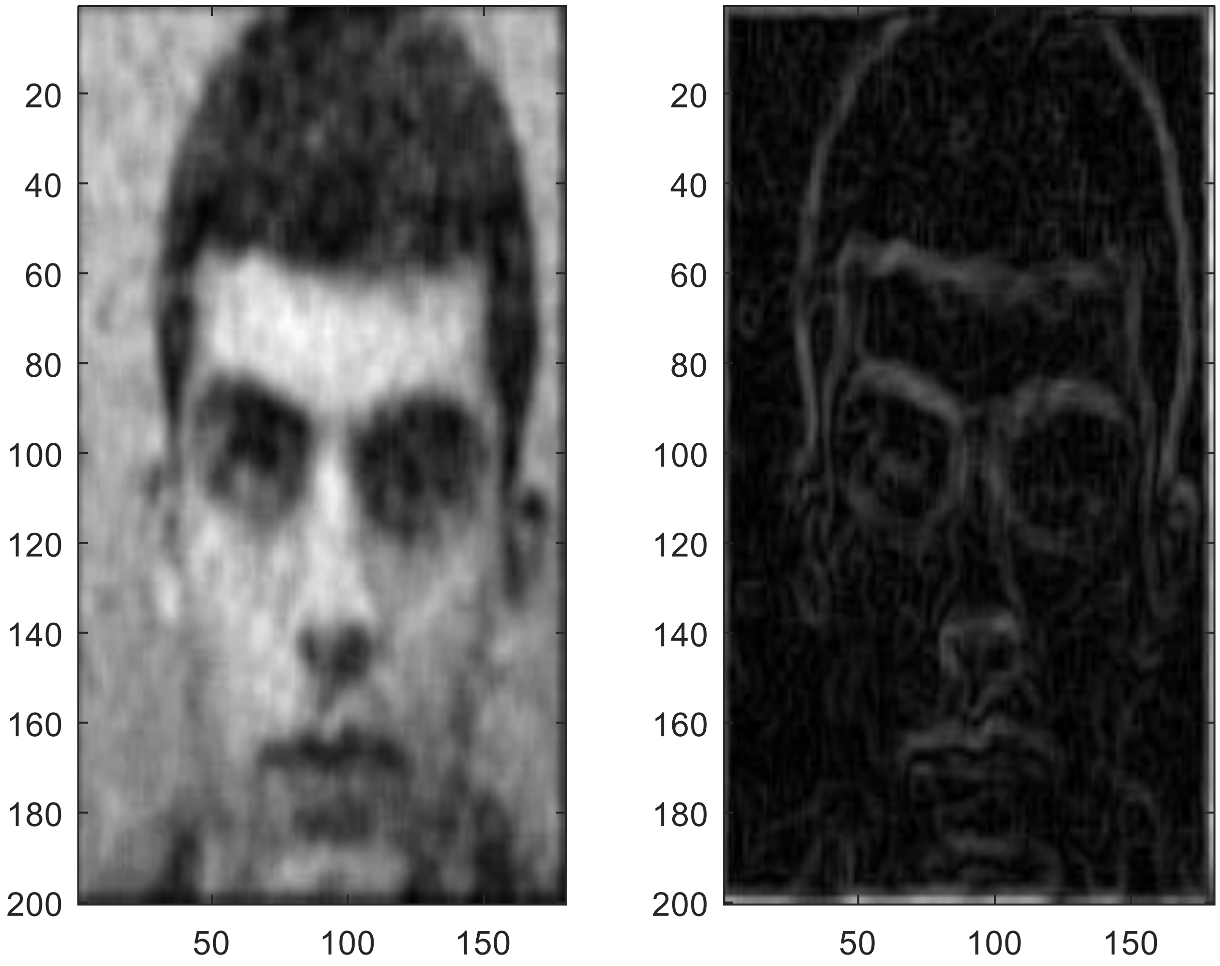
Solution: smooth first



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

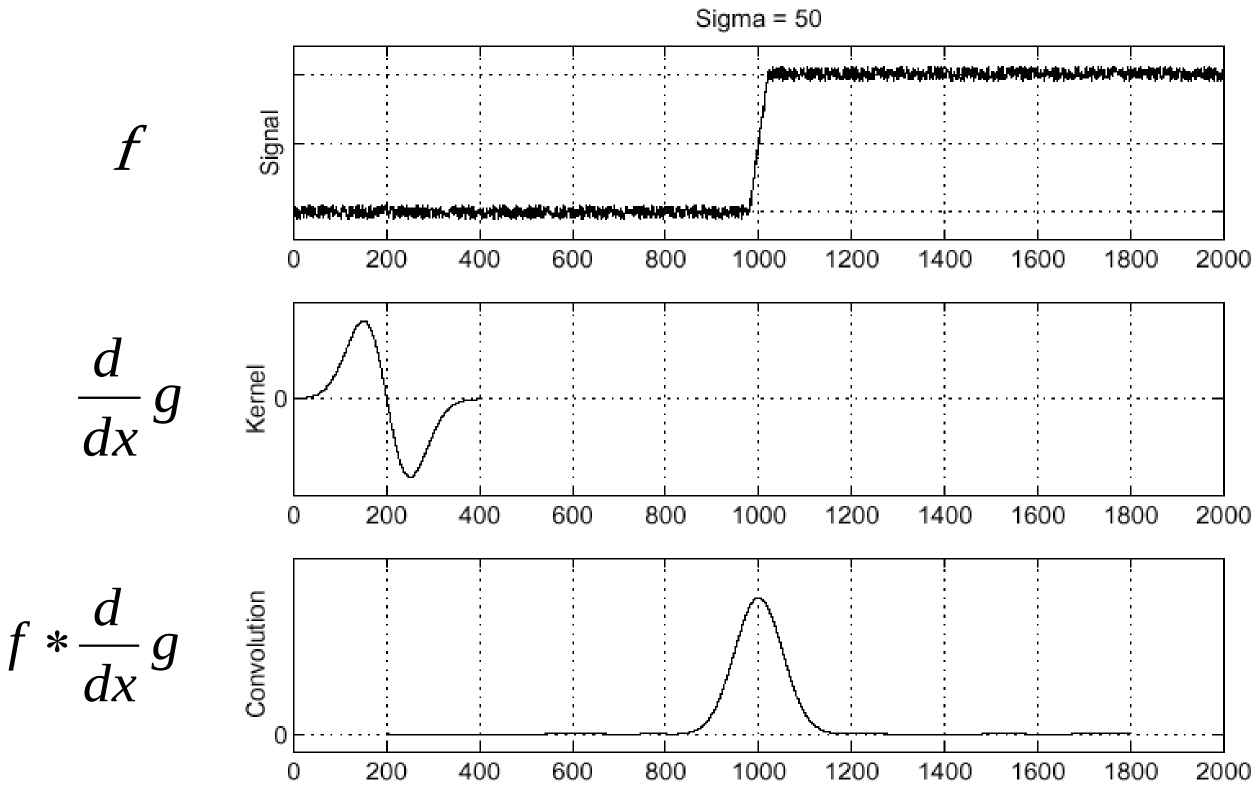
Source: S. Seitz

When noisy image is smoothed first, we get the good edges back ☺



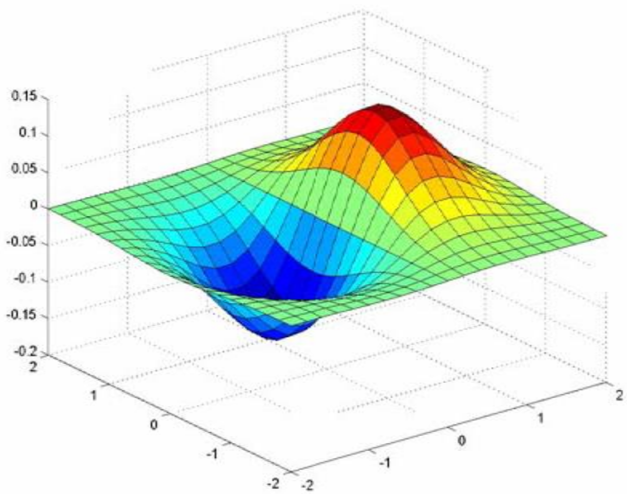
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$
- This saves us one operation:

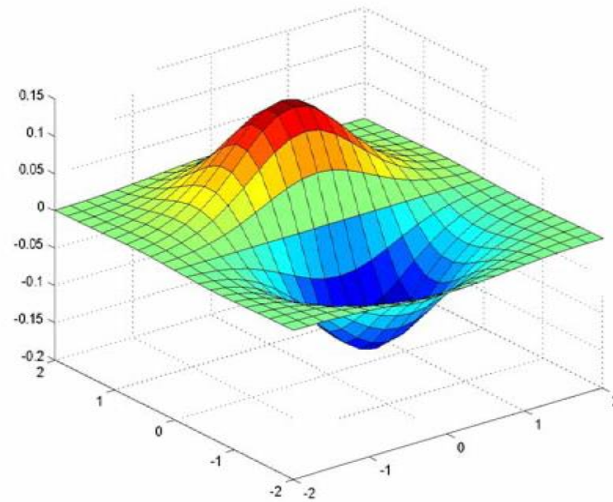


Source: S. Seitz

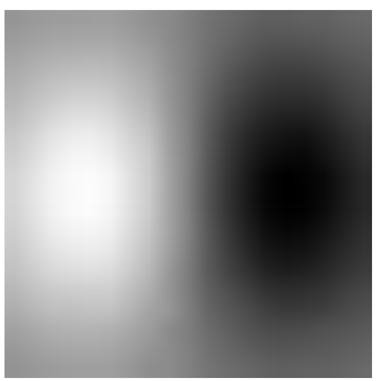
Derivative of Gaussian filter in 2D

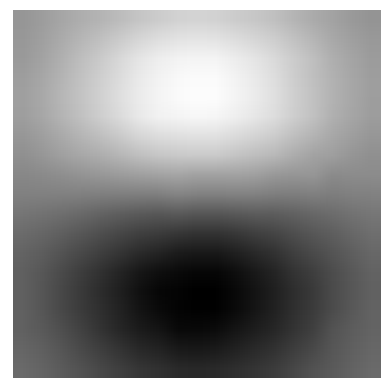


x-direction



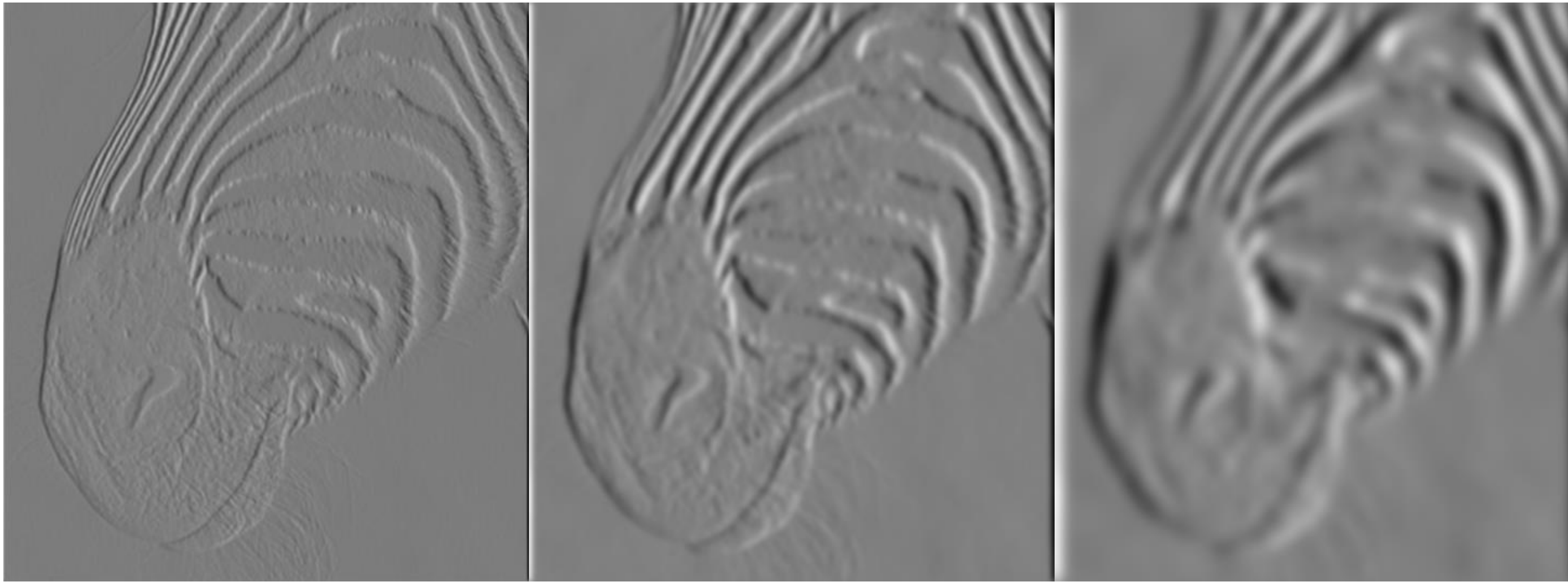
y-direction





Which one finds horizontal/vertical edges?

Effect of σ



1 pixel   3 pixels   7 pixels

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

Source: D. Forsyth

Implementation issues

- 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?

Source: D. Forsyth

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
- MATLAB: edge(image, ‘canny’)

Source: L. Fei-Fei

Canny Edge Detector

Steps:

1. Apply **directional derivatives** of Gaussian

2. Compute **gradient magnitude** and **gradient direction**

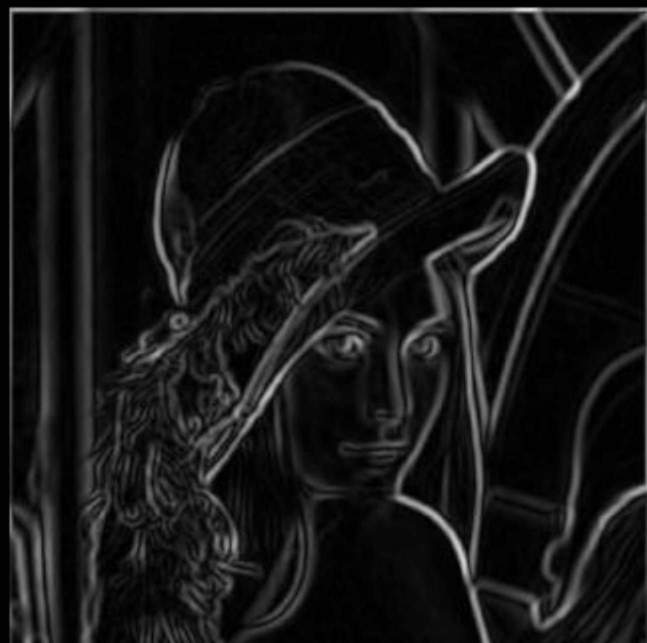
3. **Non-maximum** suppression
   – thin multi-pixel wide "ridges" down to single pixel width

4. **Linking** and thresholding
   – Low, high edge-strength thresholds
   – Accept all edges over low threshold that are connected to edge over high threshold

The Canny edge detector

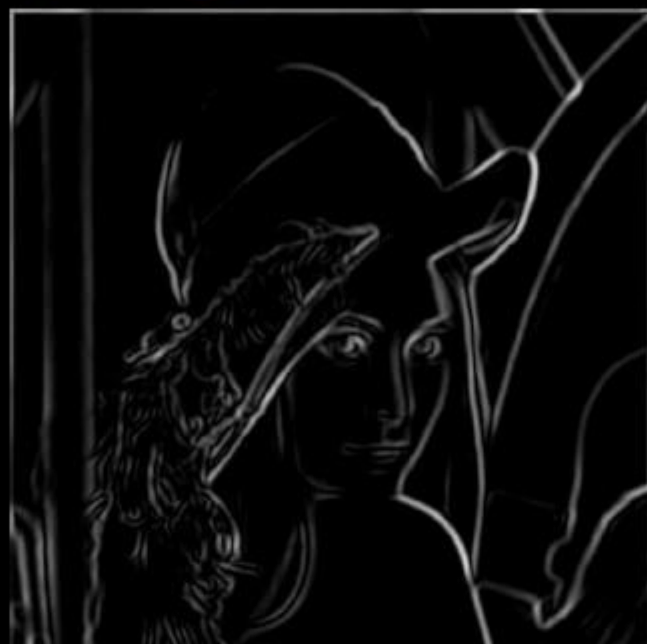
original image (Lena)

The Canny edge detector



magnitude of the gradient

The Canny edge detector

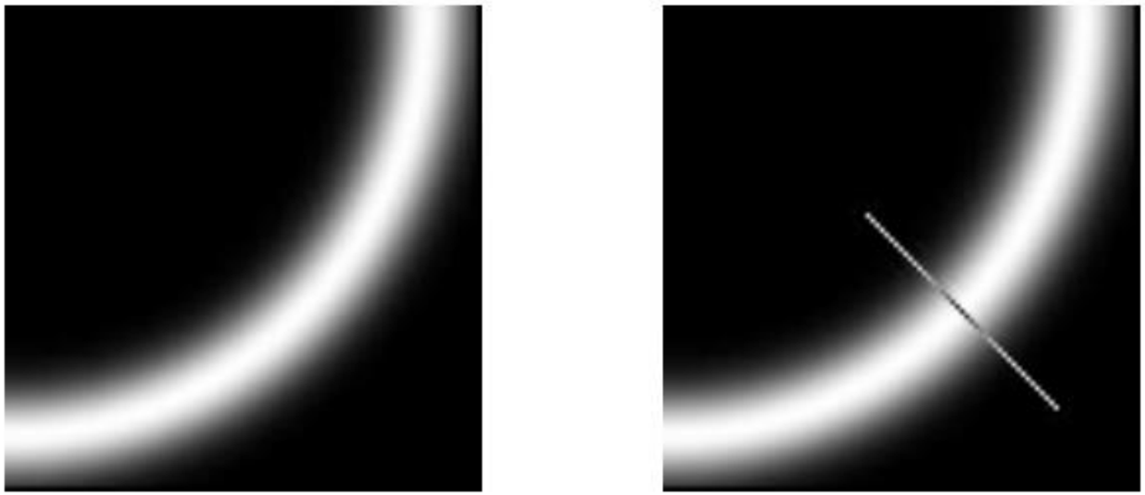


thresholding

The Canny edge detector

thinning
(non-maximum suppression)

Non-maxima Suppression

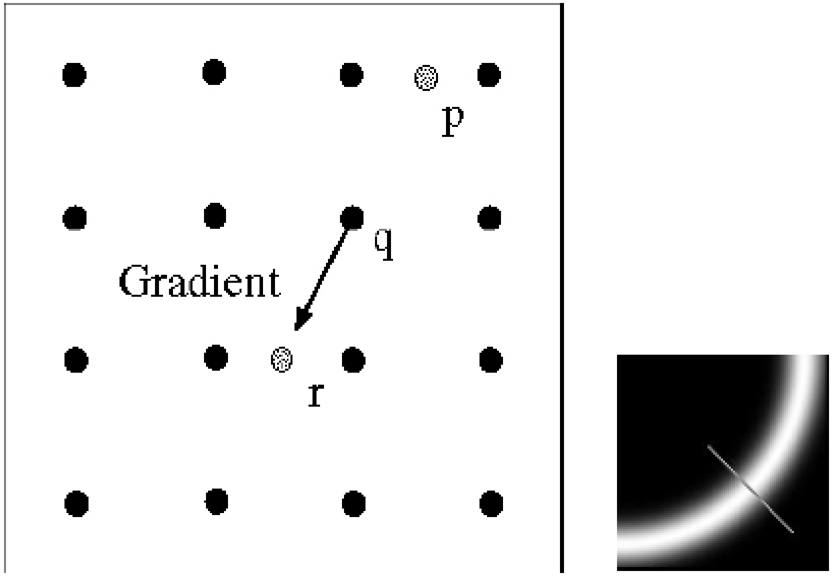


Forsyth & Ponce (1st ed.) Figure 8.11

Select the image **maximum point** across the width of the edge

Non-maxima Suppression

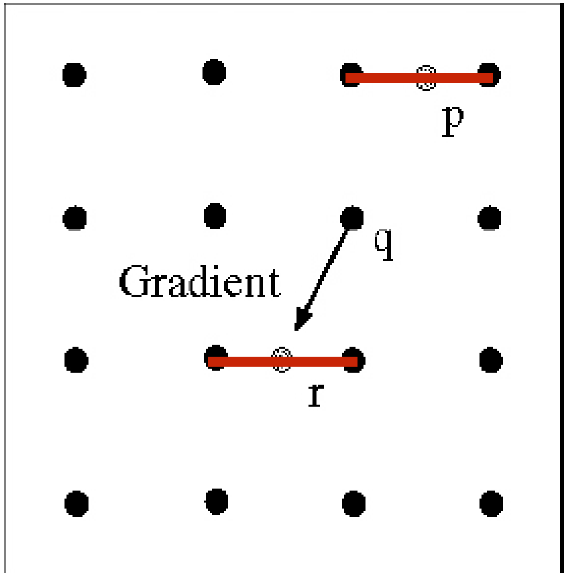
Value at *q* must be larger than interpolated values at *p* and *r*

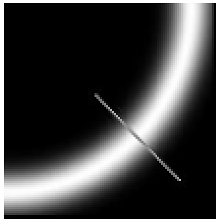


Forsyth & Ponce (2nd ed.) Figure 5.5 left

Non-maxima Suppression

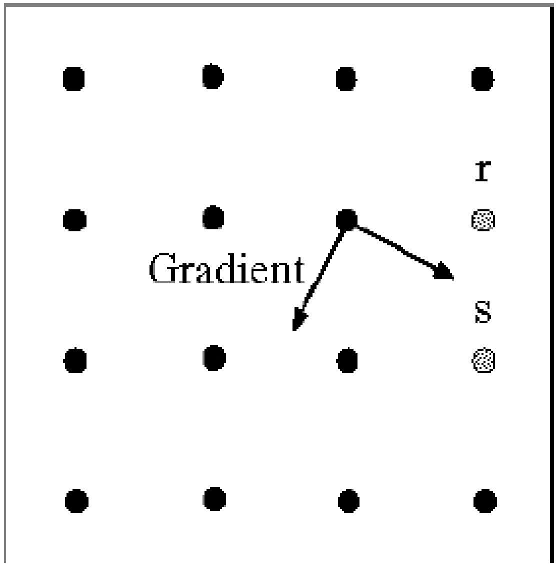
Value at *q* must be larger than interpolated values at *p* and *r*





Forsyth & Ponce (2nd ed.) Figure 5.5 left

Linking Edge Points



Forsyth & Ponce (2nd ed.) Figure 5.5 right

Assume the marked point is an **edge point**. Take the normal to the gradient at that point and use this to predict continuation points (either *r* or *s*)