Edge Detection

Computer Vision (CS0029)

Edges

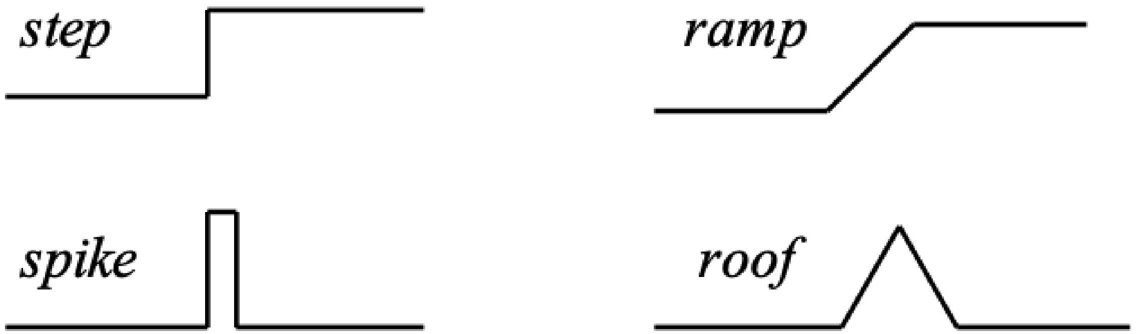
- Edge are points in an image where brightness change sharply
  - Changes in image brightness
- Many important things happen at an edge
  - Object boundaries
  - Reflectance change/patterns (e.g. zebra)
- Look at derivative in image to detect edges
  - Gradient
- Usually, the problem in edge detection is dealing with image noise

Gradients and Edges

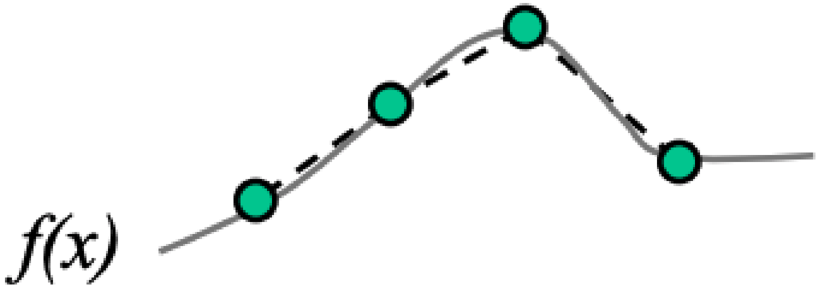
- High-contrast image points can be detected by computing intensity difference in local image regions
- Detect high-contrasts using neighborhood template or masks
  - Similar to noise removal
- We will study 1-D signal, then move to 2-D
  - 1-D signal could be a row or column of an image

Differencing 1-D Signals

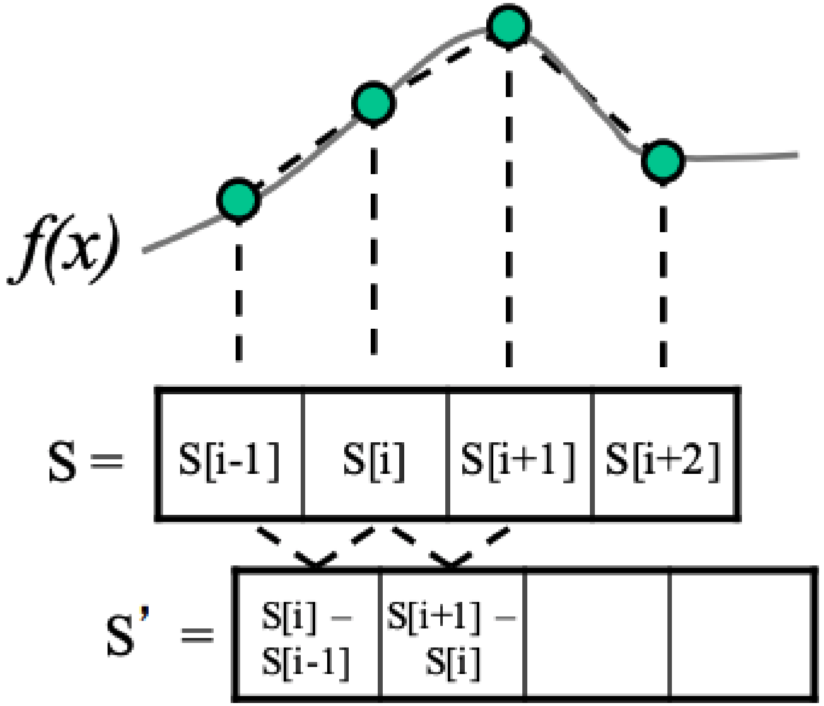
- Some types of 1-D edges



- Consider 1-D sampled signal



Difference/Derivative Mask



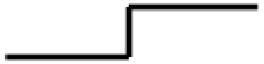
$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \approx f(x_i) - f(x_{i-1})$$

Difference Masks

M' =

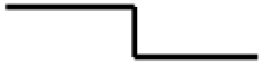
-1	**+1**

Examples of First Derivative



S_1	12	12	12	12	12	24	24	24	24	24
$S_1 \otimes M'$	-	0	0	0	0	**12**	0	0	0	0

S_1 is upward step edge

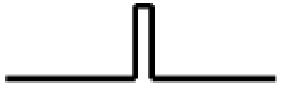


S_2	24	24	24	24	24	12	12	12	12	12
$S_2 \otimes M'$	-	0	0	0	0	**-12**	0	0	0	0

S_2 is downward step edge

S_3	12	12	12	12	15	18	21	24	24	24
$S_3 \otimes M'$	-	0	0	0	**3**	**3**	**3**	**3**	0	0

S_3 is upward ramp



S_4	12	12	12	12	24	12	12	12	12	12
$S_4 \otimes M'$	-	0	0	0	**12**	**-12**	0	0	0	0

S_4 is bright impulse

Smoothing

- The basic differences will give strong response to noise
  - Not a good way to estimate derivatives in real signals/images
- In practice, signal/image is almost always smoothed before calculating derivate
- Usually, Gaussian smoothing is used

Derivative of Gaussian

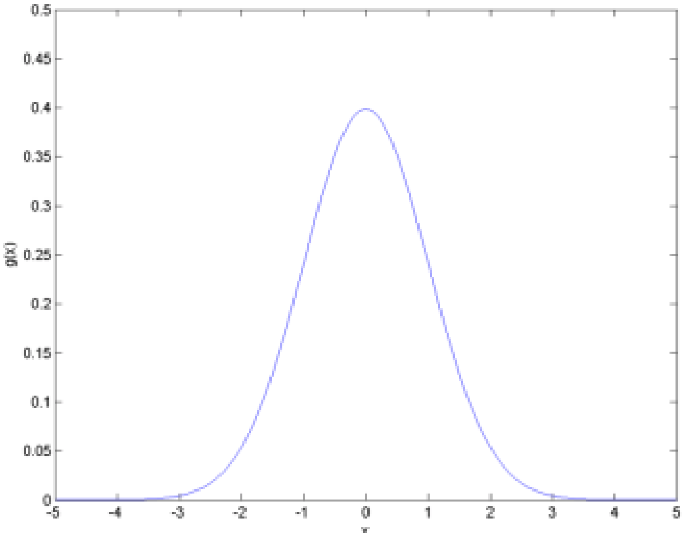
- Smoothing then differentiating is same as convolving with the derivative of a smoothing kernel
  - $M_D*(M_S*I) = (M_D*M_S)*I$

- Thus need only to convolve with a “derivative of the Gaussian” filter

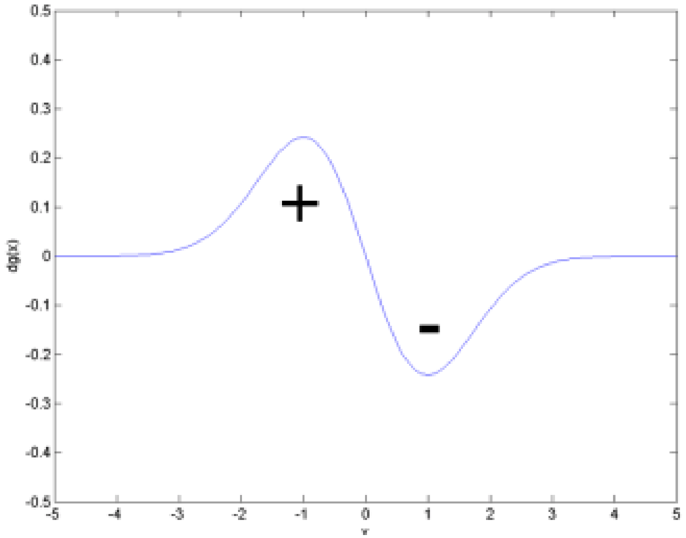
- Results in smaller noise responses from derivative estimates

Derivative of Gaussian

$$g(x;\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-x_o)^2}{2\sigma^2}} \longrightarrow$$



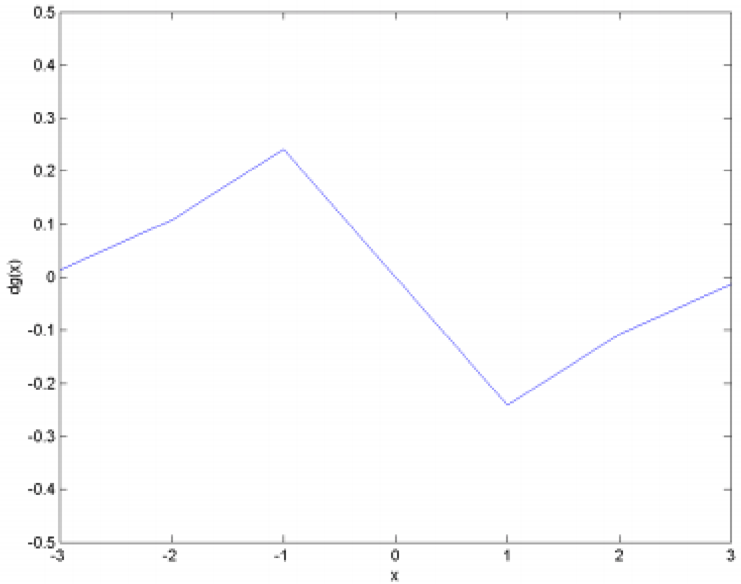
$$\frac{dg(x;\sigma)}{dx} = \frac{-(x-x_o)}{\sqrt{2\pi\sigma^3}} e^{-\frac{(x-x_o)^2}{2\sigma^2}} \longrightarrow$$



(σ controls the scale/spread)

Discrete Gaussian Derivative Mask

- Set mask size: ceil(3σ)*2 + 1
  - Examine values for $x = [-ceil(3\sigma): ceil(3\sigma)]$
- For $\sigma = 1$, yields a 7-tap filter
  - Mask=[0.01, 0.11, 0.24, 0, -0.24, -0.11, -0.01]



Second Derivative

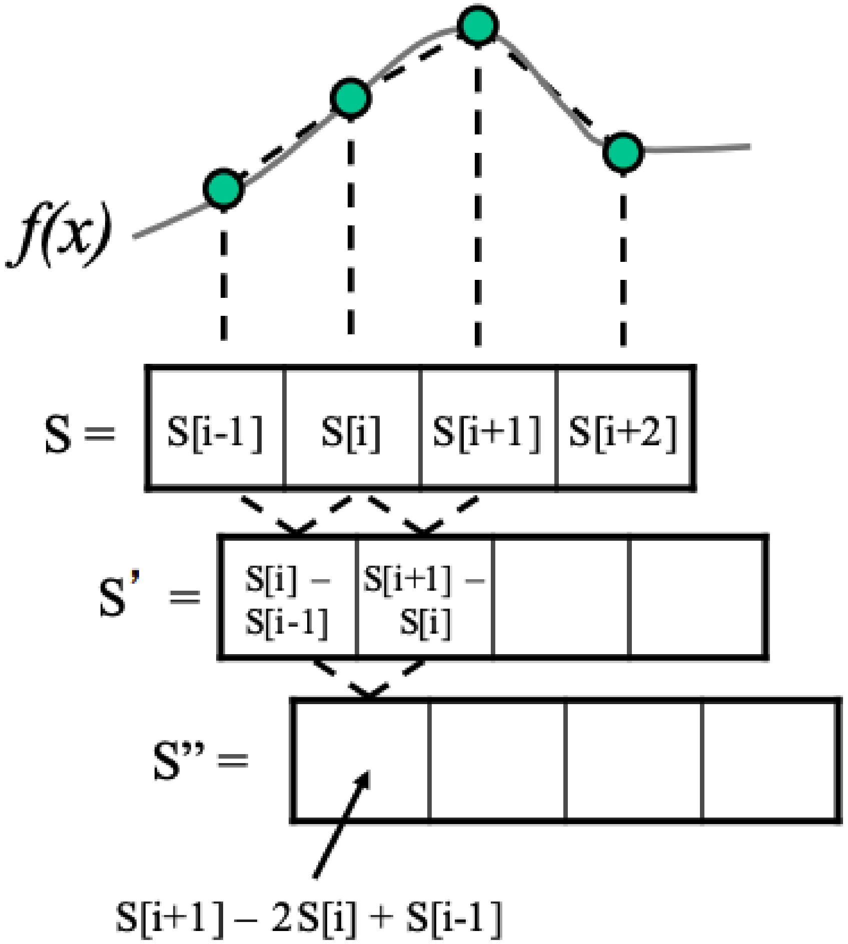
- Second derivative is zero when derivative magnitude is extremal (peak or valley)

- To find large changes (edge), good place to look is where the second derivative makes “zero-crossings”

- Look for a change from “+ to –” or “- to +” (or “0 to +/- ”, “+/- to 0”)

- Produces double-side edges

Difference/Derivative Masks



$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \approx f(x_i) - f(x_{i-1})$$

Difference Masks

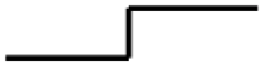
M' =

-1	**+1**

M'' =

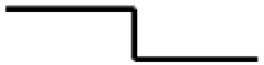
+1	**-2**	**+1**

Examples of Second Derivative



S_1	12	12	12	12	12	24	24	24	24	24
$S_1 \otimes M''$	-	0	0	0	**12**	**-12**	0	0	0	0

S_1 is upward step edge

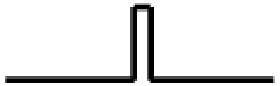


S_2	24	24	24	24	24	12	12	12	12	12
$S_2 \otimes M''$	-	0	0	0	**-12**	**12**	0	0	0	0

S_2 is downward step edge

S_3	12	12	12	12	15	18	21	24	24	24
$S_3 \otimes M''$	-	0	0	**3**	0	0	0	**-3**	0	0

S_3 is upward ramp

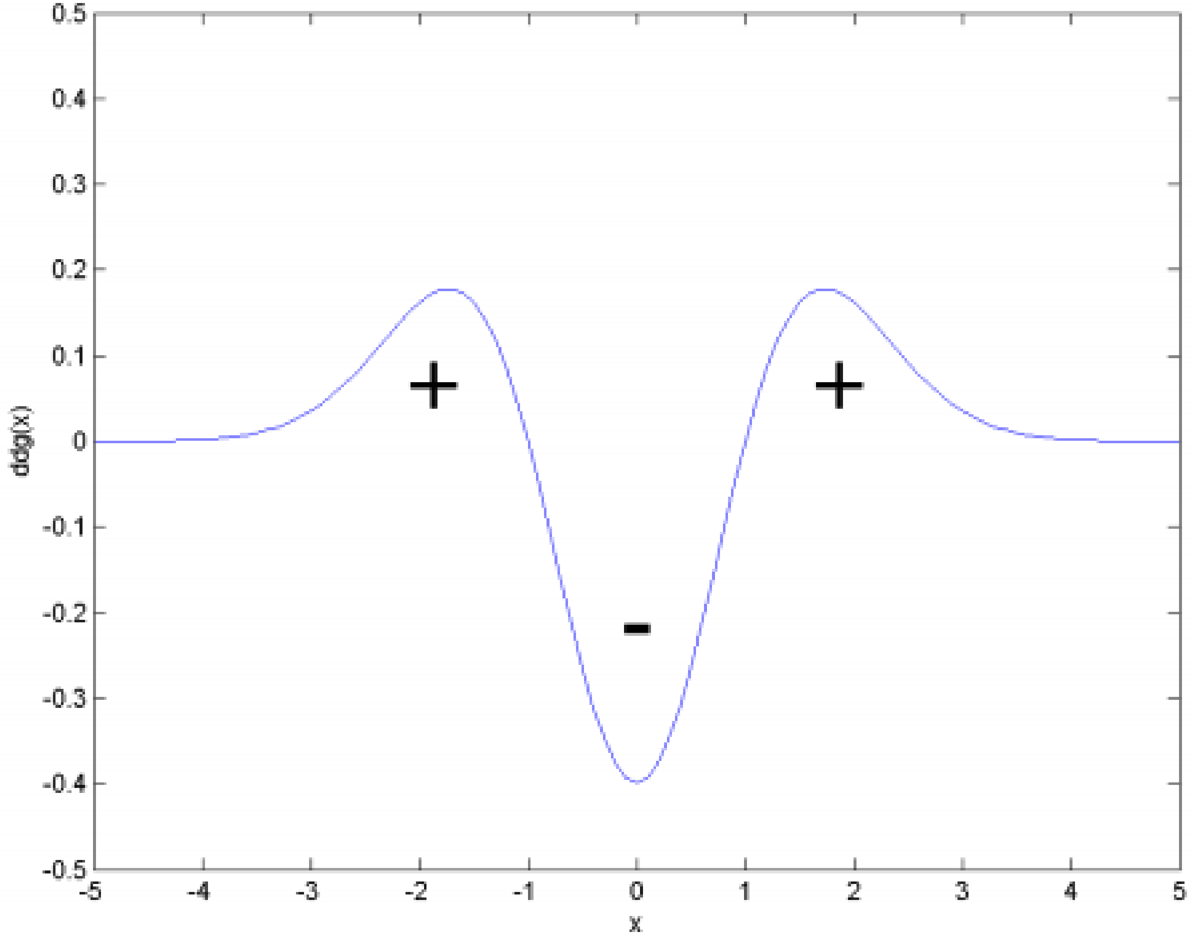


S_4	12	12	12	12	24	12	12	12	12	12
$S_4 \otimes M''$	-	0	0	**12**	**-24**	**12**	0	0	0	0

S_4 is bright impulse

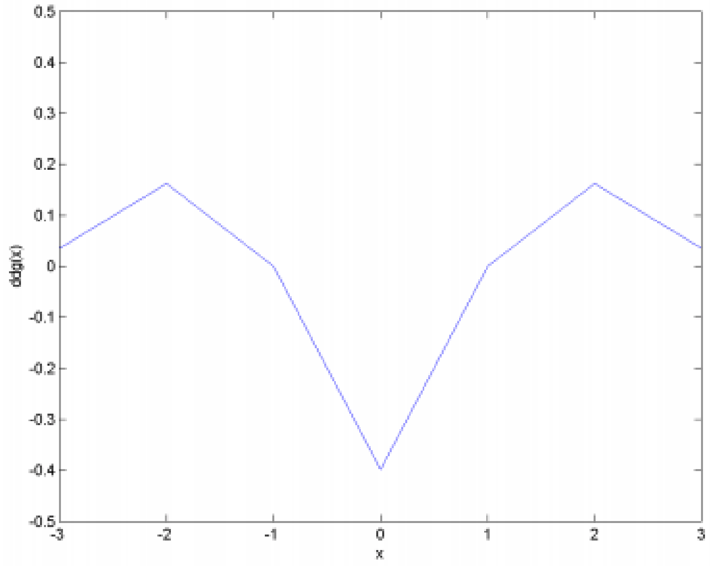
Gaussian Second Derivative

$$\frac{\partial^2 g(x;\sigma)}{\partial x^2} = \left(\frac{(x-x_o)^2}{\sqrt{2\pi}\sigma^5} - \frac{1}{\sqrt{2\pi}\sigma^3} \right) \cdot e^{-\frac{(x-x_o)^2}{2\sigma^2}}$$



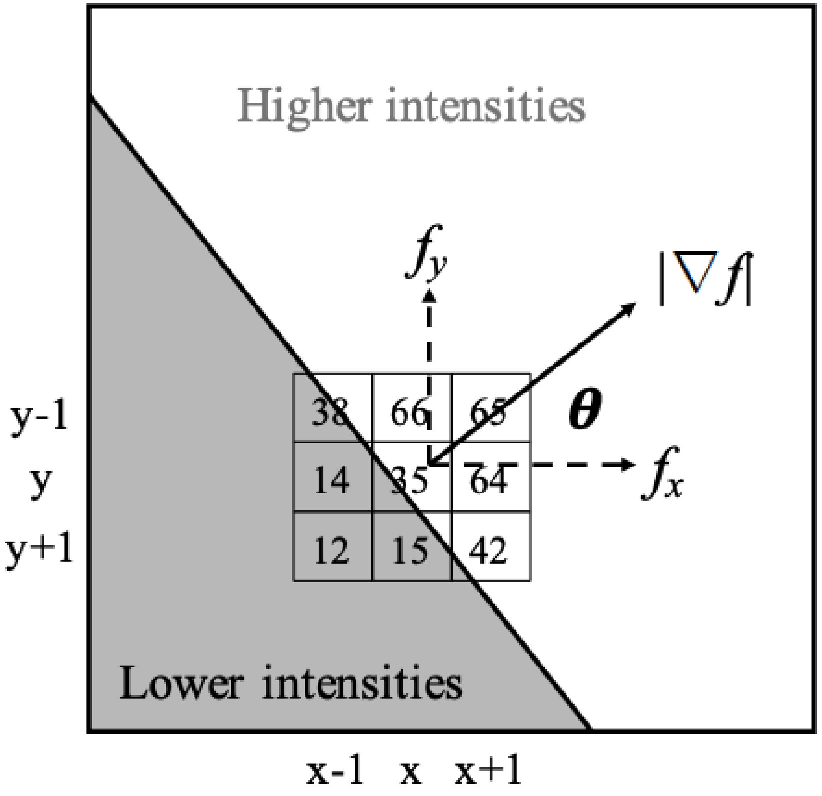
Gaussian Second Derivative Mask

- Set mask size: ceil(3σ)*2+1
  - Examine values for $x = [-ceil(3\sigma) : ceil(3\sigma)]$
- For $\sigma = 1$, yields a 7-tap filter
  - Mask = [0.04, 0.16, 0, -0.4, 0, 0.16, 0.04]



2-D Signals (Images)

- 2-D Difference “Gradient” Operators



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

$$\begin{aligned}\frac{\partial f}{\partial x} \equiv f_x \approx \quad & \frac{1}{3}[(I[x+1,y]-I[x-1,y])/2 \\ & +(I[x+1,y-1]-I[x-1,y-1])/2 \\ & +(I[x+1,y+1]-I[x-1,y+1])/2]\end{aligned}$$

$$\begin{aligned}\frac{\partial f}{\partial y} \equiv f_y \approx \quad & \frac{1}{3}[(I[x,y+1]-I[x,y-1])/2 \\ & +(I[x-1,y+1]-I[x-1,y-1])/2 \\ & +(I[x+1,y+1]-I[x+1,y-1])/2]\end{aligned}$$

$$\theta = \operatorname{atan}(f_y, f_x) \qquad |\nabla f| = \sqrt{f_x^2 + f_y^2}$$

Classic Gradient Masks

Prewitt:

$$F_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \;/6 \qquad F_y = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \;/6$$

<u>Sobel</u>:

$$F_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \;/8 \qquad F_y = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \;/8$$

Separability

Prewitt:

$$F_x = 1/6 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad F_y = 1/6 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

<u>Sobel</u>:

$$F_x = 1/8 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad F_y = 1/8 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Prewitt mask results

F_x F_y

Sobel mask results

F_x F_y

Gradient Magnitude

$\sqrt{}$

^{2} +

^{2} =

F_x

F_y

Different Gradient Magnitude Threshold

Use different thresholds

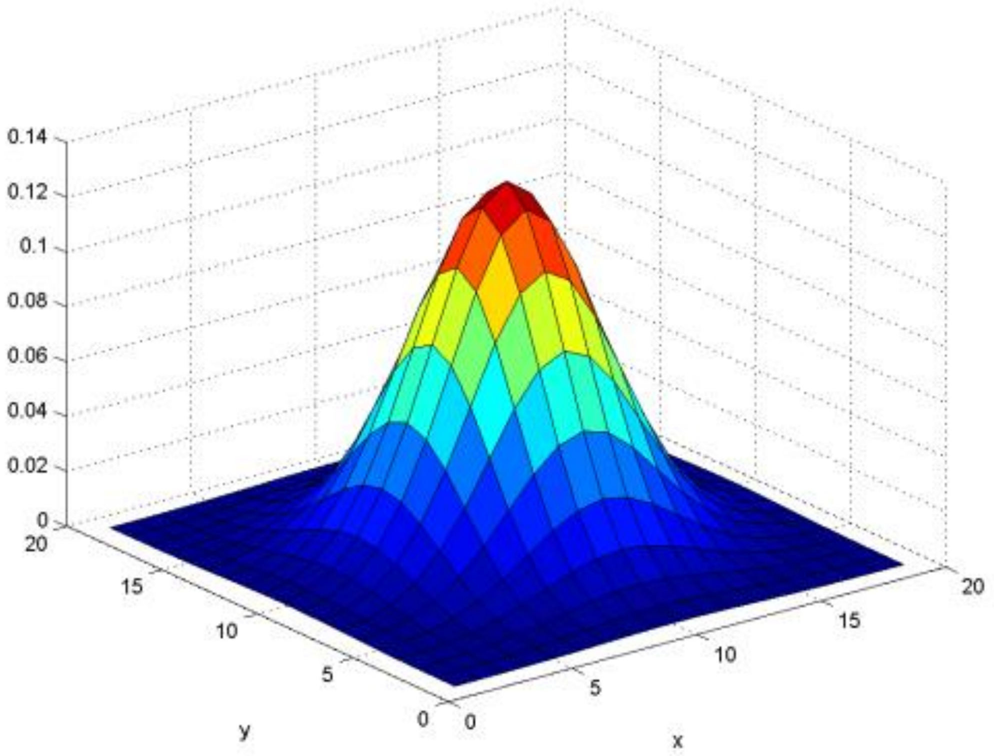
    

OpenCV - Sobel

```
import cv2
import numpy as np

img = cv2.imread('kcface.png', cv2.IMREAD_GRAYSCALE)

#X and Y direction sobel filter
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)
cv2.imwrite('kcFaceGrayFx.png', sobelx)
cv2.imwrite('kcFaceGrayFy.png', sobely)

#combine gradient X and Y
grad = np.sqrt( np.square(sobelx) + np.square(sobely) )
cv2.imwrite('kcFaceGraySobel.png', grad)

#thresholding
grad[grad < 250] = 0
grad[grad >= 250] = 255
cv2.imwrite('SobelThreshold250.png', grad)
```

2-D Gaussian

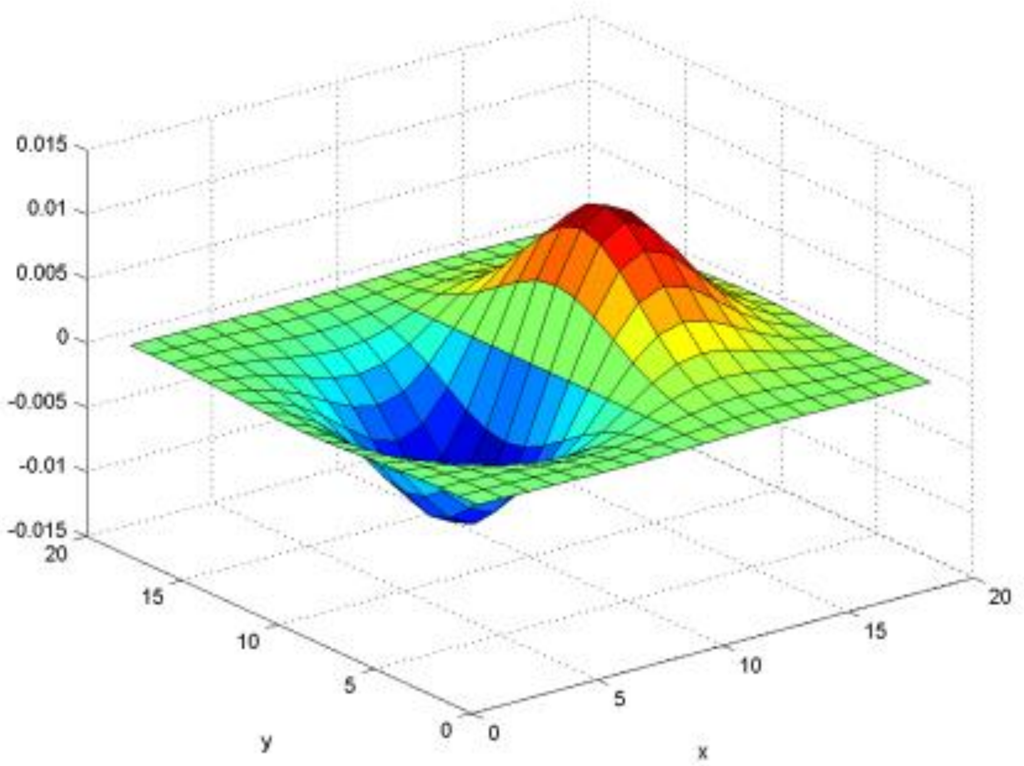
$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_o)^2 + (y-y_o)^2}{2\sigma^2}}$$

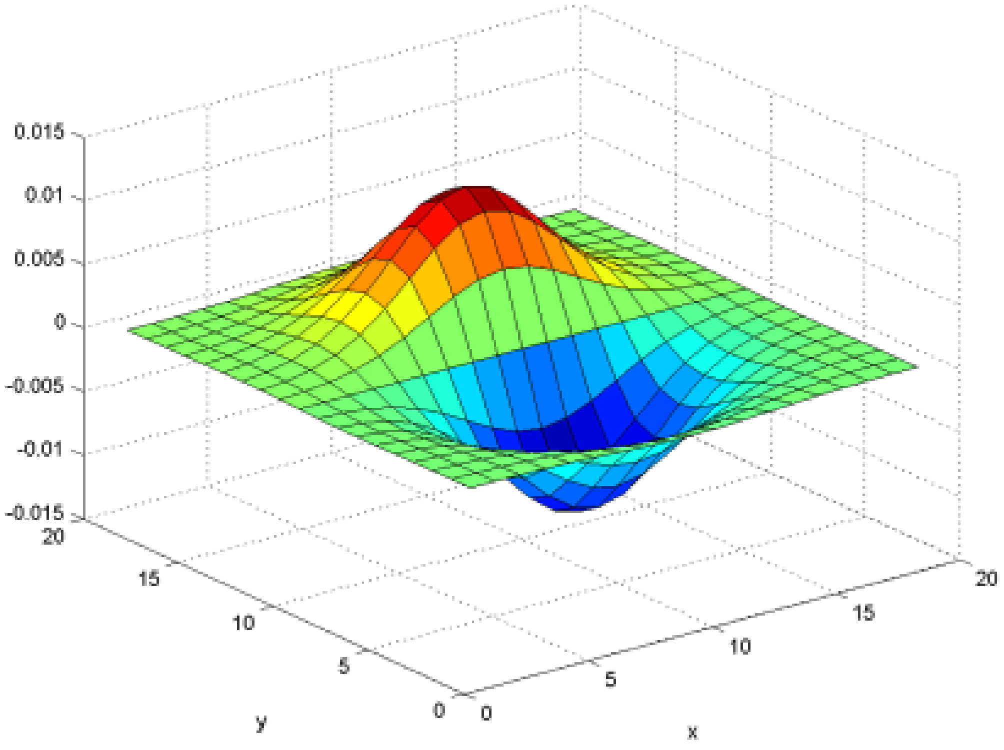


Gaussian Derivatives

$$\frac{\partial g(x,y;\sigma)}{\partial x} = \frac{-(x-x_o)}{2\pi\sigma^4} e^{-\frac{(x-x_o)^2+(y-y_o)^2}{2\sigma^2}}$$

$$\frac{\partial g(x,y;\sigma)}{\partial y} = \frac{-(y-y_o)}{2\pi\sigma^4} e^{-\frac{(x-x_o)^2+(y-y_o)^2}{2\sigma^2}}$$



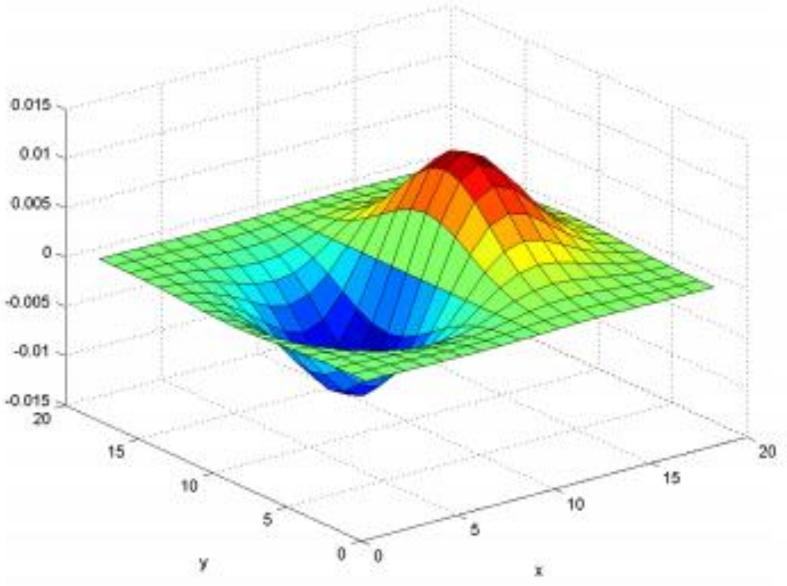


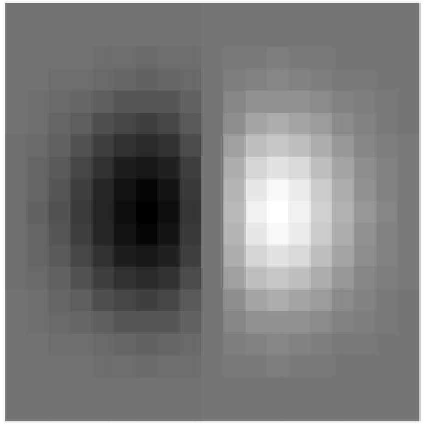
Creating Masks

- Gx mask
  - Fill mask values with $g_x(x, y, ; \sigma)$
    - x-range: $[-ceil(3\sigma){:}\, ceil(3\sigma)]$ (or 2 σ)
    - y-range: $[-ceil(3\sigma){:}\, ceil(3\sigma)]$

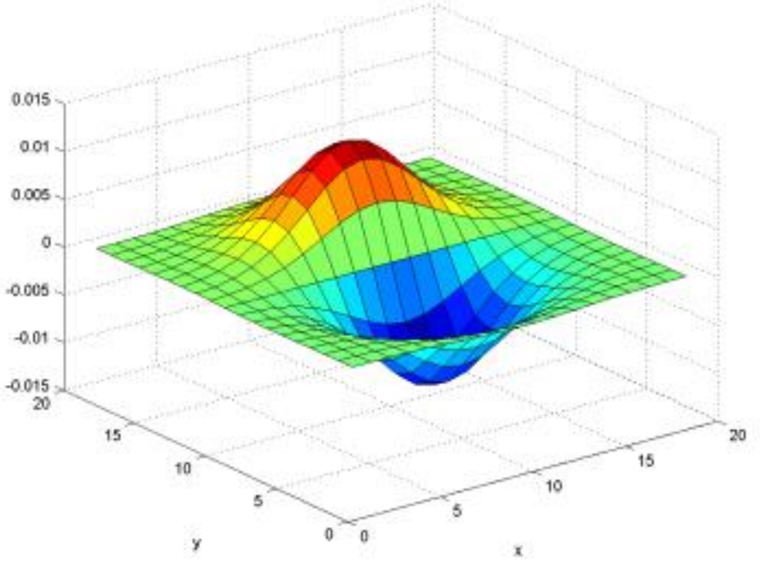
- Gy mask
  - Fill mask values with $g_y\ (x, y, ; \sigma)$
    - x-range: $[-ceil(3\sigma){:}\, ceil(3\sigma)]$ (or 2 σ)
    - y-range: $[-ceil(3\sigma){:}\, ceil(3\sigma)]$

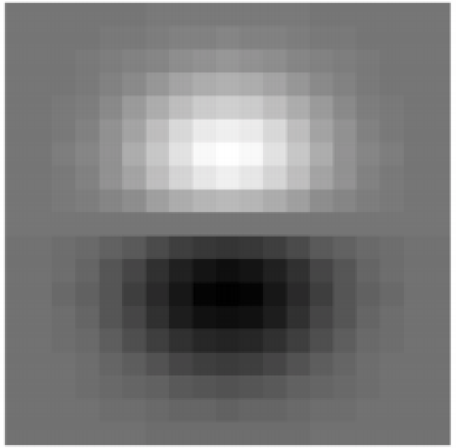
Gaussian Derivatives





G_x





G_y

Effect of Scale

$\sigma = 1$

$\sigma = 3$

$\sigma = 5$

Thicker edges (from left to right)

Laplacian of Gaussian

- Second derivative function of Gaussian
  - Zero-crossing are found at edge locations
  - Smoothing used to combat noise

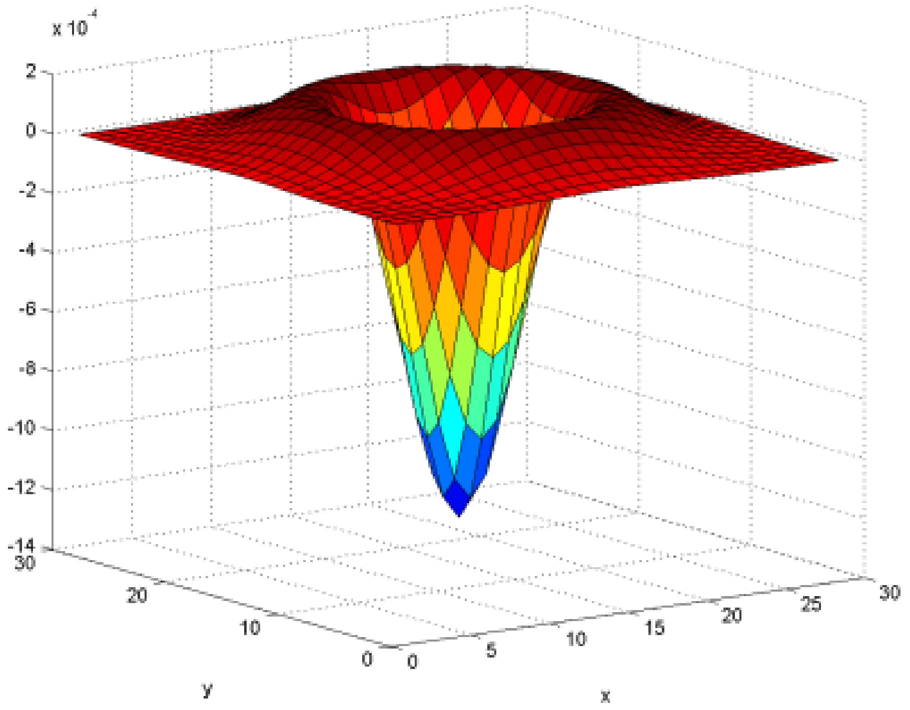
- Combine two orientations into one circular filter
  - Simply sum Gaussian second derivative in x direction and y direction
  - Non-oriented, the second derivative filter

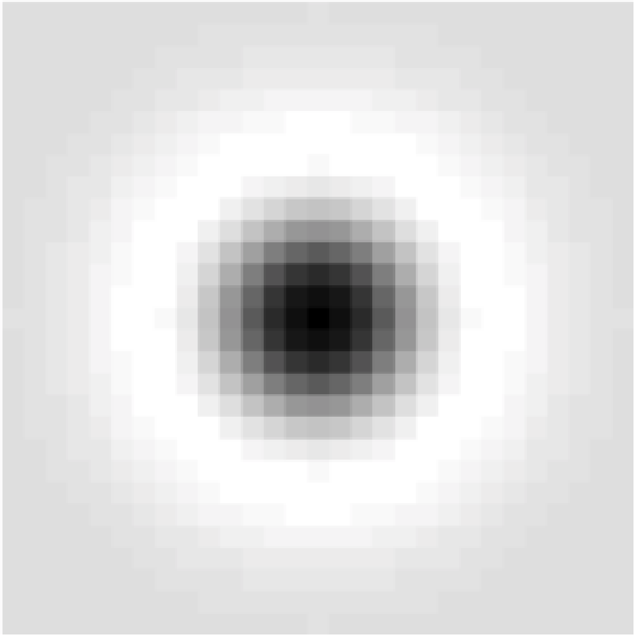
- Laplacian of Gaussian operator:

$$\nabla^2 g(x, y; \sigma) = \frac{\partial^2 g(x, y; \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y; \sigma)}{\partial y^2}$$

Laplacian of Gaussian

$$\nabla^2 g(x, y; \sigma) = \frac{\partial^2 g(x, y; \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y; \sigma)}{\partial y^2}$$





Zero-Crossings

$\sigma = 3$

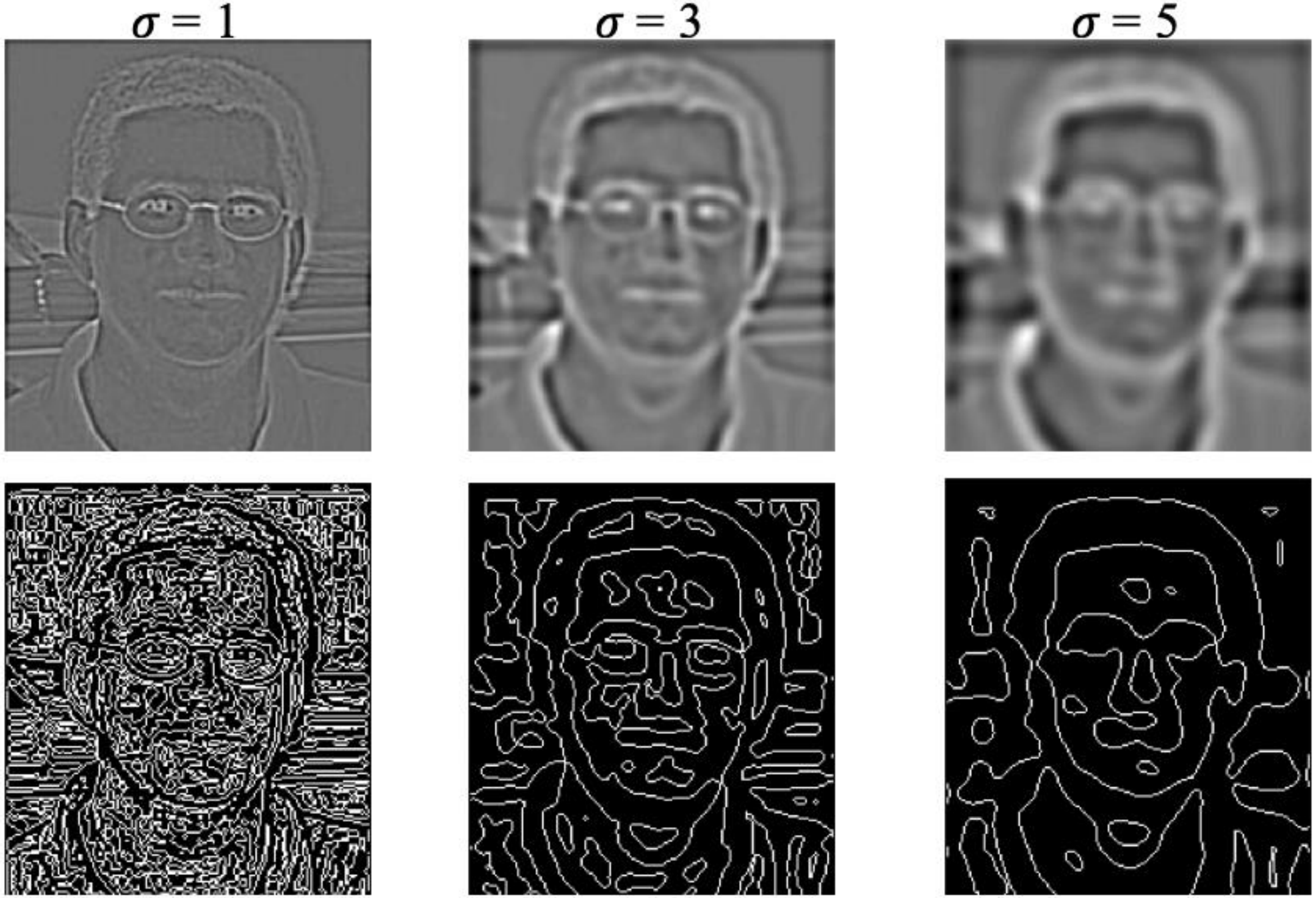
Selected zero-crossings

Effect of Scale

From left to right

$\sigma = 1, 3, 5$?
or
$\sigma = 5, 3, 1$?

Effect of Scale



Simple Laplacian Mask

- Small 3x3 mask approximation of LOG

$$\text{LOG} = \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Canny Edge Detector

- Very popular and effective method
- Produces extended contour/edge segments by “following” high gradient magnitudes within the smoothed image

Canny Edge Detection

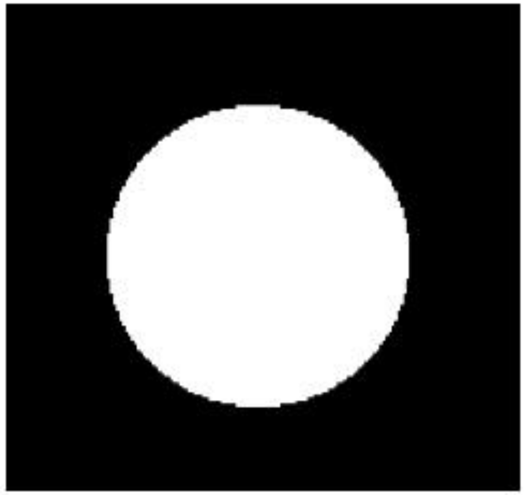
- 1. Smooth the image (Gaussian)

- 2. Compute the gradient
  - Calculate magnitude and orientation at each pixel (sobel operator)

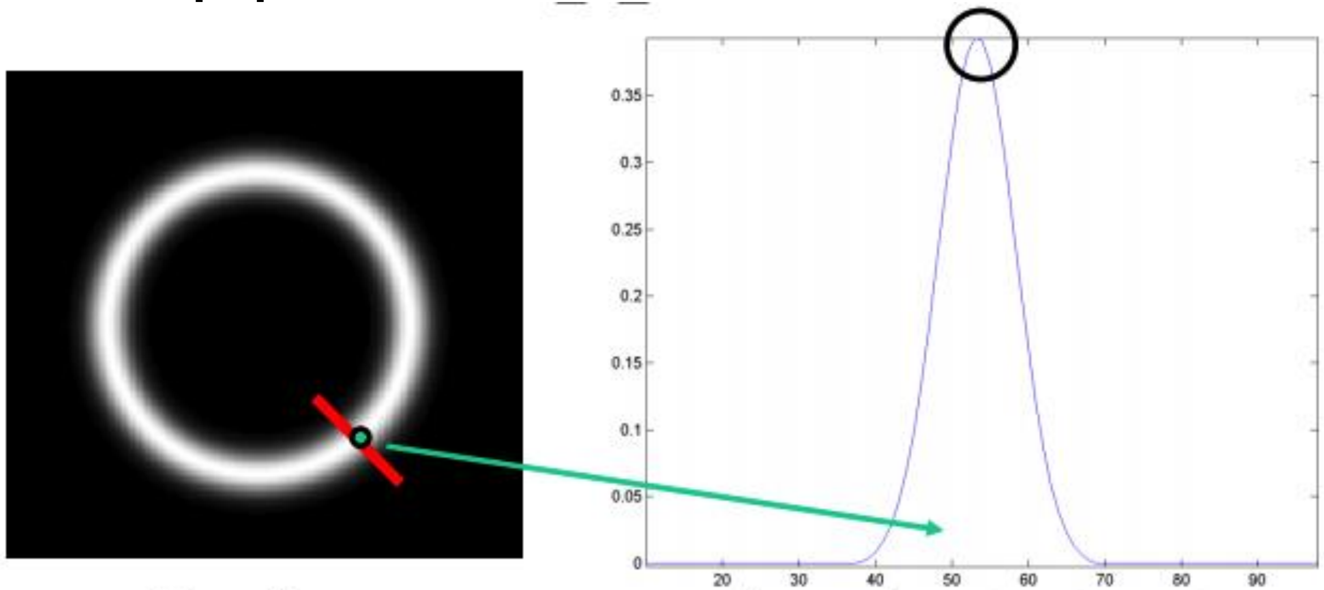
- 3. Suppress non-maximal gradients
  - Keep points where gradient magnitude is maximal along the direction of the gradient

- 4. Follow edge contours (edge linking)
  - Use upper and lower thresholds

Non-Maximal Suppression



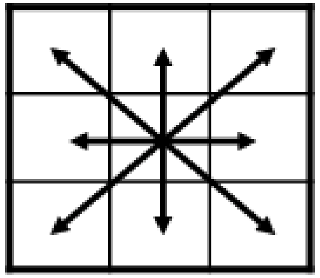
Input image



Gradient magnitude

Along gradient orientation at point

- Detect a pixel is the maximal between positive and negative gradient
  - Slice the gradient magnitude along gradient direction (perpendicular to edge)
    - Quantize gradient orientation by 45 degrees
    - Could also interpolate values
  - Mark points along slice that are maximal
    - E.g. 0 degree direction and gradient points left: if g[x-1]<g[x] or g[x]>g[x+1], then g[x]=1

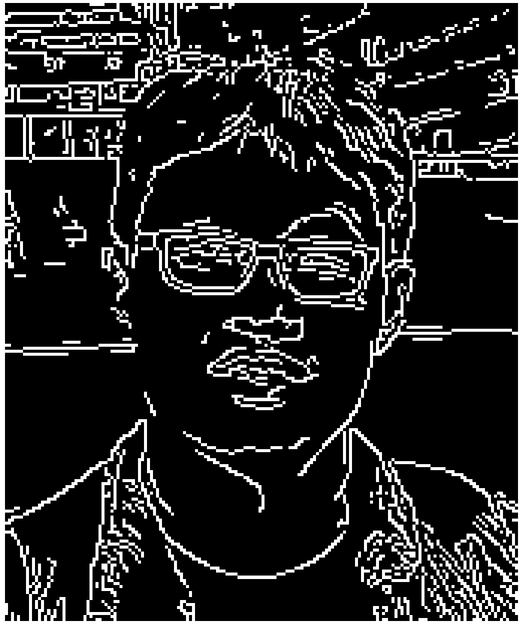


Edge Linking

- Sequentially follow continuous contour segments
- Initiate only on edge pixels where gradient magnitude meets high threshold (T_H)
- A single threshold can cause many broken edge segments
- Once started, follow through connect pixels whose gradient magnitude meet a lower threshold (T_L)
  - If $g[x] < T_L$, no edge at x
  - If $g[x] > T_H$, edge at x
  - If $T_H > g[x] > T_L$ and $g[x_n] > T_H$
    - x_n : any one of 8-connected neighbors

Canny Edge Detection Results

Original Image



Canny Edge Detection

Overlay

```
import cv2
import numpy as np

img = cv2.imread('kcface.png', cv2.IMREAD_GRAYSCALE)
canny = cv2.Canny(img,150,200)
```