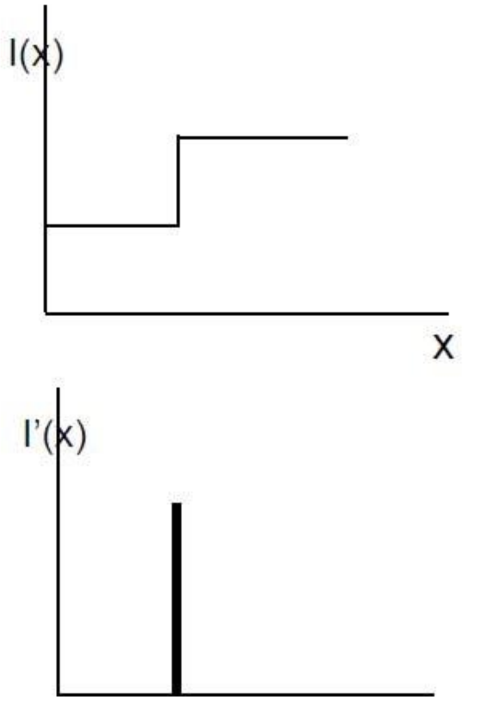
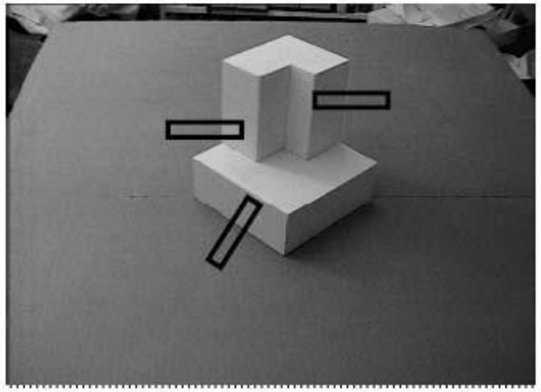
EDGES DETECTION

Nada Fitrieyatul Hikmah

Characteristics of an Edge (1)

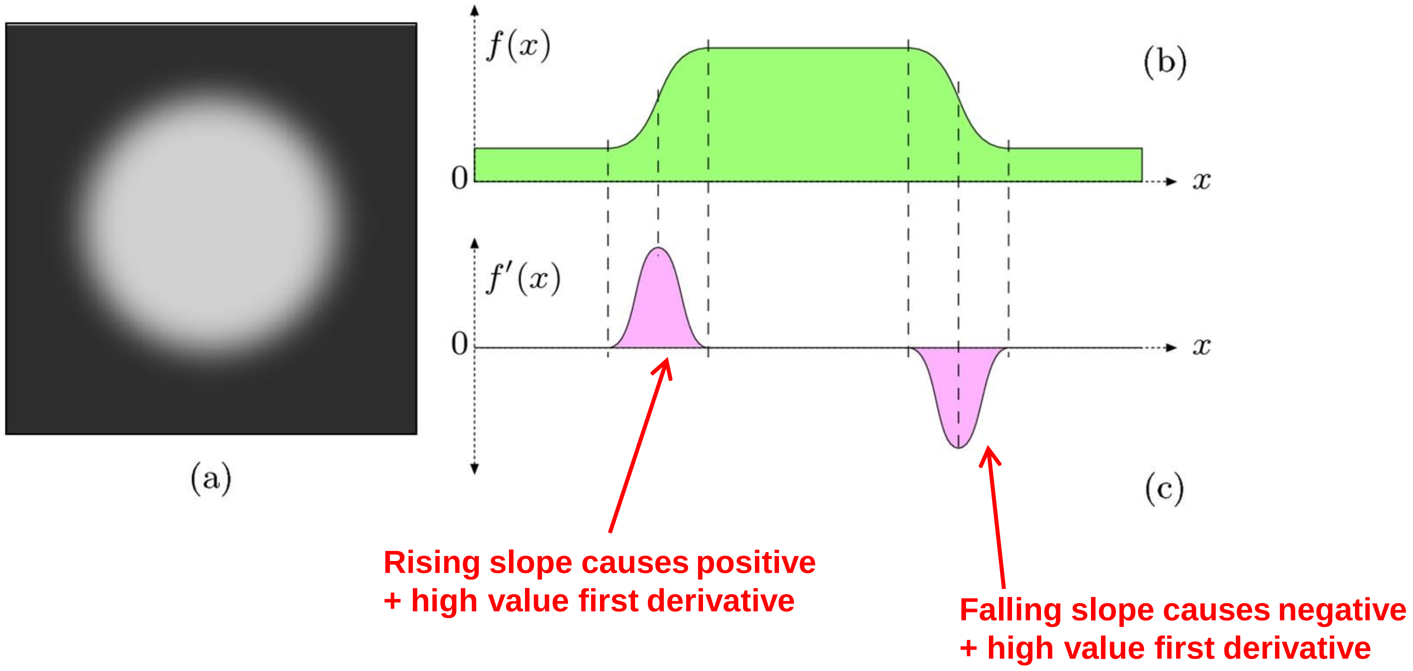
- Edge: A sharp change in brightness
- Ideal edge is a step function in some direction





Characteristics of an Edge (2)

- Real (non-ideal) edge is a slightly blurred step function
- Edges can be characterized by high value first derivative

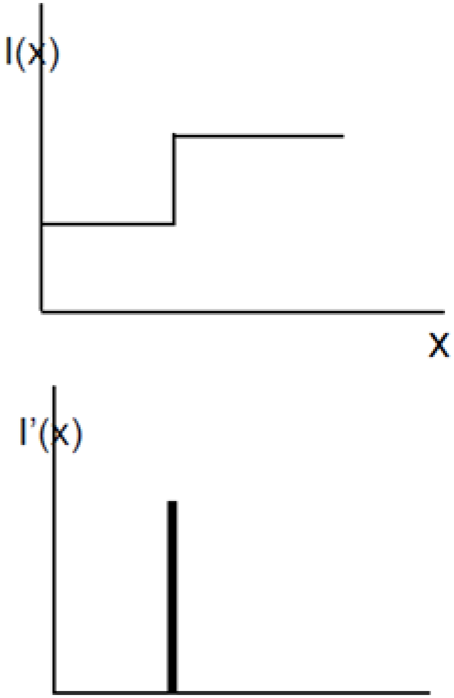


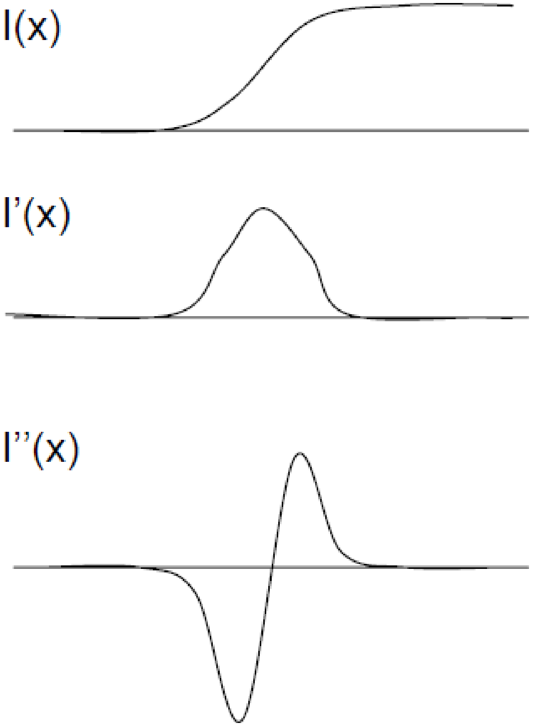
$$f'(x) = \frac{df}{dx}(x)$$

Characteristics of an Edge (3)

- Ideal edge is a step function in certain direction.
- First derivative of I(x) has a **peak** at the edge
- Second derivative of I(x) has a **zero crossing** at edge

Ideal edge





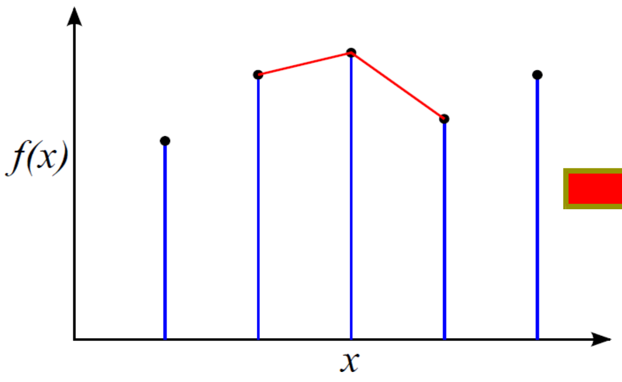
Real edge

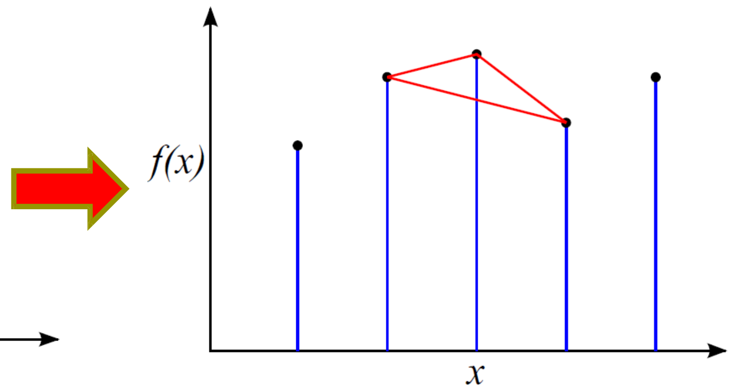
First derivative shows peak

Second derivative shows zero crossing

Finite Differences

- Left and right slope may not be same
- Solution? Take average of left and right slope





- Forward difference (right slope)

$$\Delta_+ f(x) = f(x+1) - f(x)$$

- Backward difference (left slope)

$$\Delta_- f(x) = f(x) - f(x-1)$$

- Central Difference (average slope)

$$\Delta f(x) = \frac{1}{2}\left(f(x+1) - f(x-1)\right)$$

Definition: Function Gradient

- Let $f(x,y)$ be a 2D function
- **Gradient:** Vector whose direction is in direction of maximum rate of change of f and whose magnitude is maximum rate of change of f
- Gradient is perpendicular to edge contour

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

- magnitude $= [(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2]^{1/2}$
- direction $= \tan^{-1}(\frac{\partial f / \partial y}{\partial f / \partial x})$

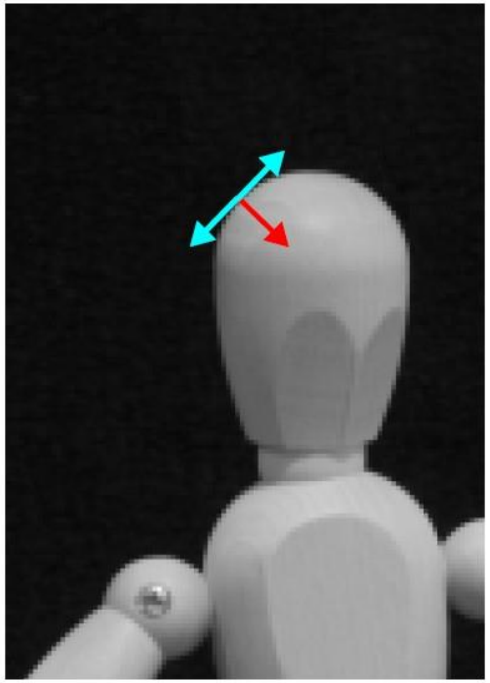


Image Gradient

- Image is 2D discrete function
- Image derivatives in horizontal and vertical directions

$$\frac{\partial I}{\partial u}(u,v) \quad \text{and} \quad \frac{\partial I}{\partial v}(u,v)$$

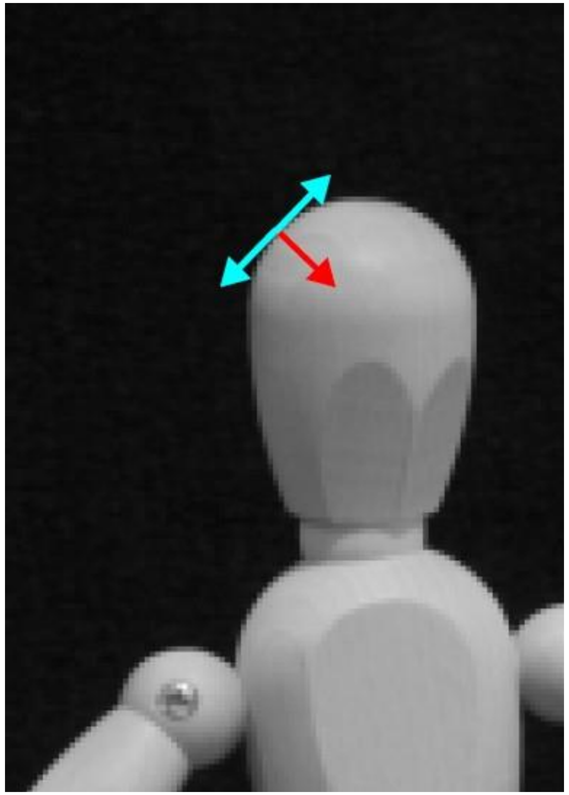
- Image gradient at location (u,v)

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

- Gradient magnitude

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

- Magnitude is invariant under image rotation, used in edge detection



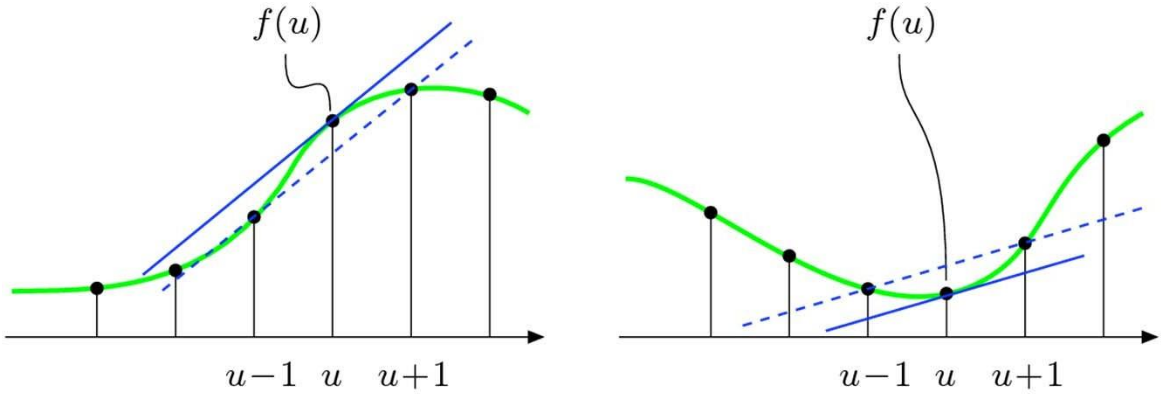
Derivative Filters

- We can compute derivative of discrete function as

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$

- Can we make linear filter that computes central differences

$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$



Finite Differences as Convolutions (1)

- Forward difference

$$\Delta_+ f(x) = f(x+1) - f(x)$$

- Take a convolution kernel $\boldsymbol{H} = [0 \quad -1 \quad 1]$

$$\Delta_+ f = f * H$$

Finite Differences as Convolutions (2)

- Central difference

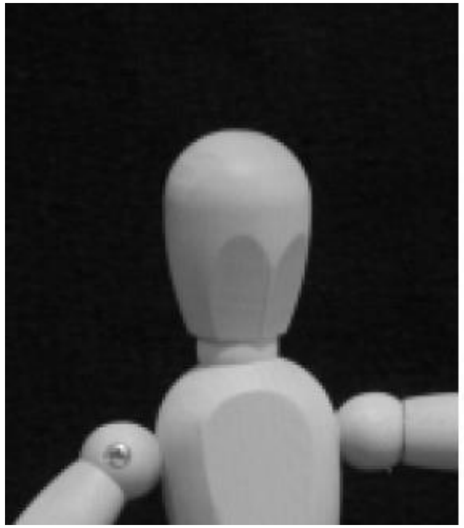
$$\Delta f(x) = \frac{1}{2}\left(f(x+1) - f(x-1)\right)$$

- Convolution kernel is: ***H* =** $\begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix}$

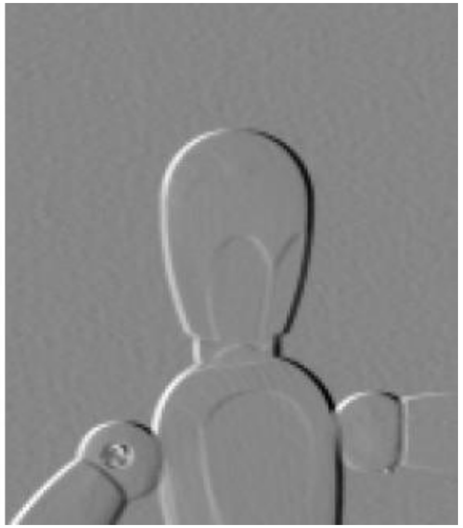
$$\Delta f(x) = f * H$$

- **Notice:** Derivative kernels sum to zero

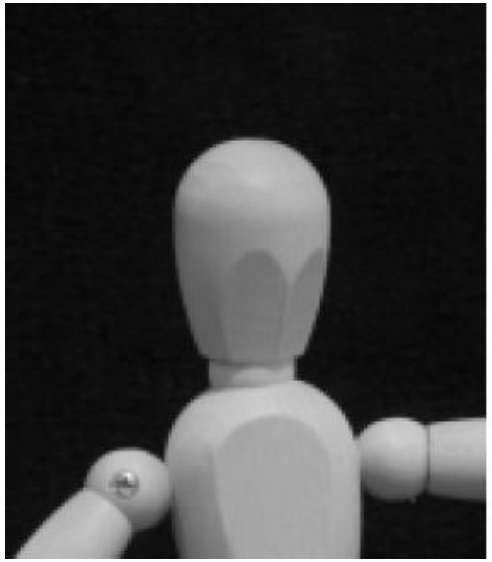
x-Derivative of Image using Central Difference



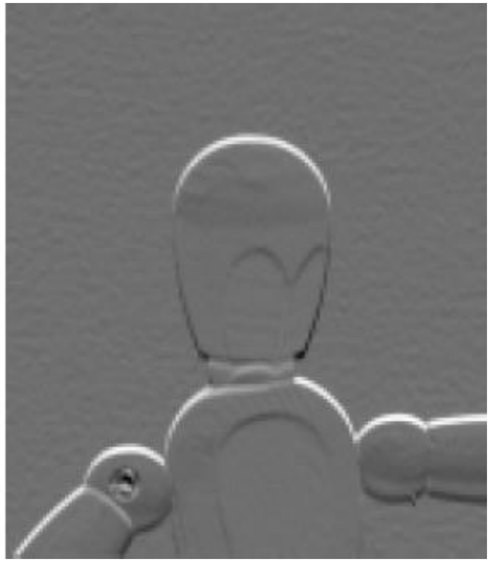
$* \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} =$



y-Derivative of Image using Central Difference



$$* \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} =$$

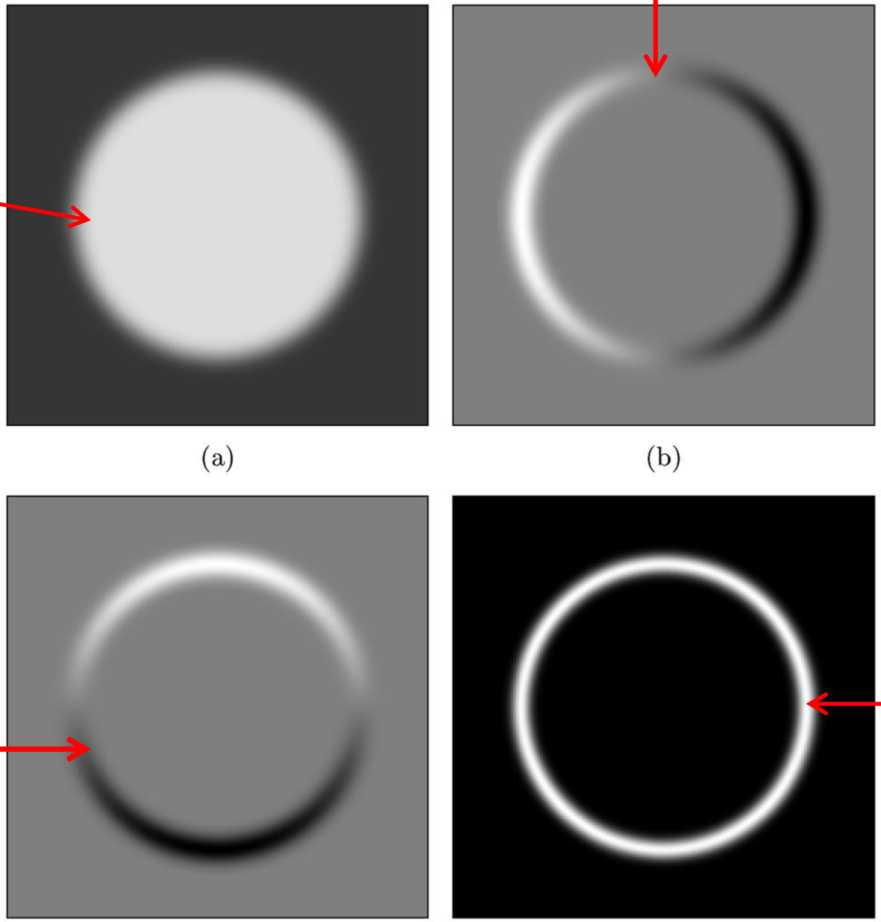


Derivative Filters

Gradient slope in
horizontal direction

$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$

A synthetic
image



(a) (b) (c) (d)

$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

Gradient slope in
vertical direction

Magnitude of
gradient

Edge Operators

- Approximating local gradients in image is basis of many classical edge-detection operators
- Main differences?
  - Type of filter used to estimate gradient components
  - How gradient components are combined
- We are typically interested in
  - Local edge direction
  - Local edge magnitude

Partial Image Derivatives

- Partial derivatives of images replaced by finite differences

$$\Delta_x f = f(x,y) - f(x-1,y) \qquad \boxed{\begin{matrix} -1 & 1 \end{matrix}} \qquad \boxed{\begin{matrix} 1 \\ -1 \end{matrix}}$$

$$\Delta_y f = f(x,y) - f(x,y-1)$$

- Alternatives are:

$$\Delta_{2x} f = f(x+1,y) - f(x-1,y) \qquad \boxed{\begin{matrix} -1 & 0 & 1 \end{matrix}} \qquad \boxed{\begin{matrix} 1 \\ 0 \\ -1 \end{matrix}}$$

$$\Delta_{2y} f = f(x,y+1) - f(x,y-1)$$

- Robert's gradient

$$\Delta_+ f = f(x+1,y+1) - f(x,y) \rightarrow \boxed{\begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}}$$

$$\Delta\_ f = f(x,y+1) - f(x+1,y) \rightarrow \boxed{\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}}$$

$$\boxed{\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}} \qquad \boxed{\begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}}$$

Prewitt

$$\boxed{\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}} \qquad \boxed{\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}}$$

Sobel

Prewitt and Sobel Edge Operators

- Prewitt Operator

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Written in separable form

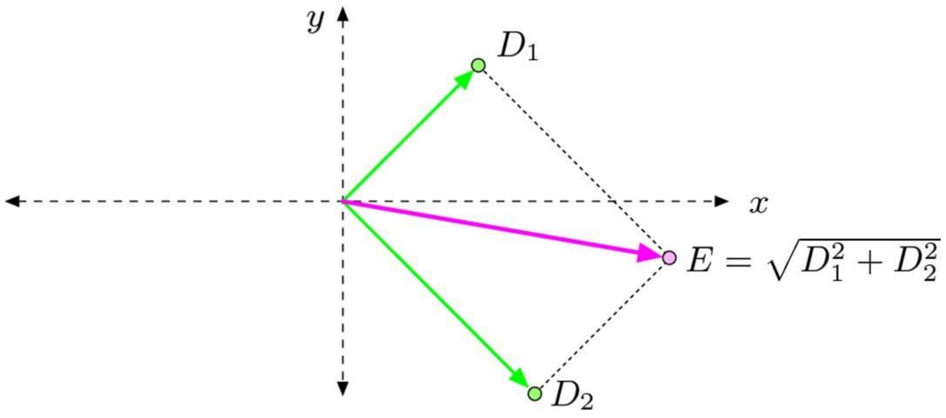
$$H_x^P = \begin{bmatrix} 1 \\ \mathbf{1} \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} 1 & \mathbf{1} & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

- Sobel Operator

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Roberts Edge Operators (1)

- Estimates directional gradient along 2 image diagonals
- Edge strength E(u,v): length of vector obtained by adding 2 orthogonal gradient components $D_1(u,v)$ and $D_2(u,v)$



- Filters for edge components $H_1^R = \begin{bmatrix} 0 & \mathbf{1} \\ -1 & 0 \end{bmatrix}$ and $H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & \mathbf{1} \end{bmatrix}$

Roberts Edge Operators (2)

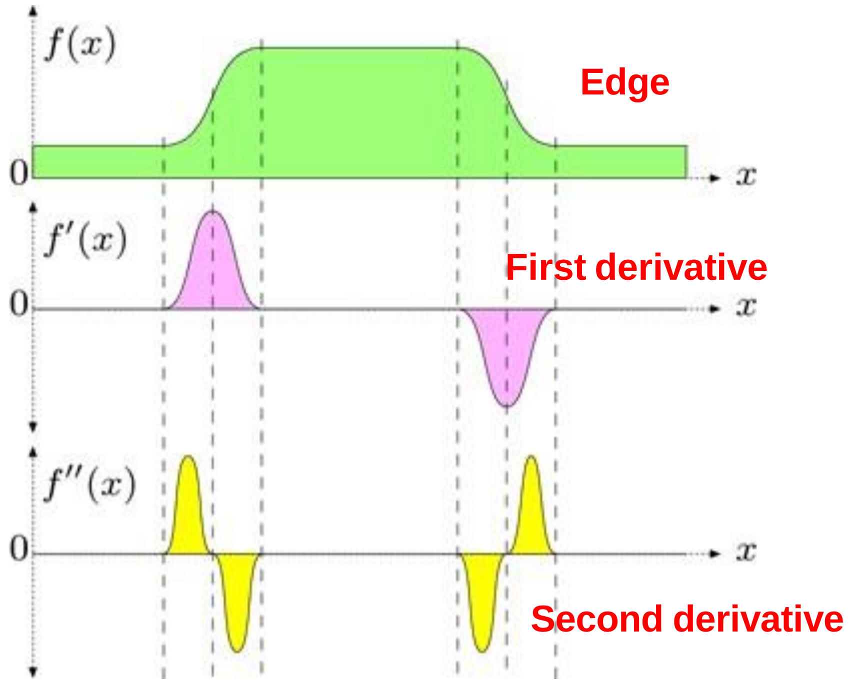
- Diagonal gradient components produced by 2 Robert filters

$D_1 = I * H_1^R$

$D_2 = I * H_2^R$

Other Edge Operators

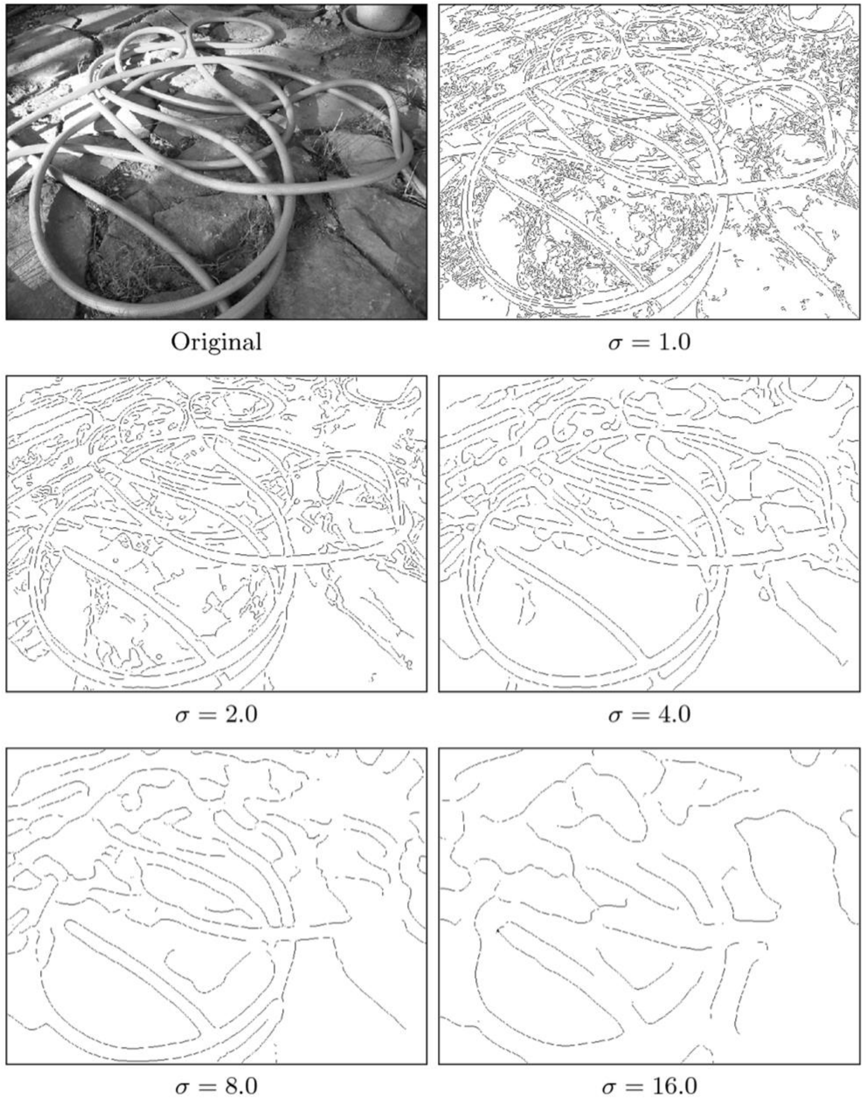
- Problem with edge operators based on first derivatives:
  - Edge is proportional to underlying intensity transition
  - Edges may be difficult to localize precisely
- Solution? Use second derivative
- **Recall:** An edge corresponds to a zero crossing of the 2nd derivative
- Since 2nd derivatives amplify image noise, pre-smoothing filters used first



Canny Edge Detector (1)

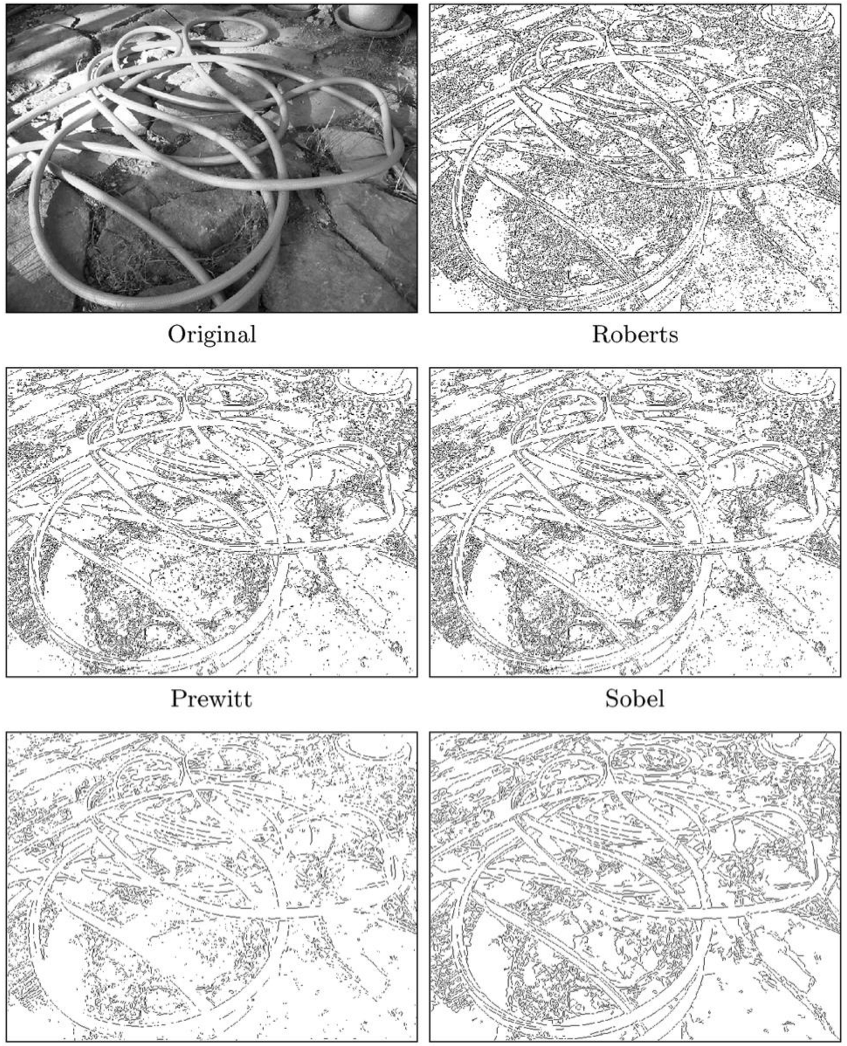
- Popular edge detector that operates at different scales, then combines results into common edge map. Tries to:
  1. Minimize number of false edge points
  2. Achieve good localization of edges
  3. Deliver only a single mark on each edge
- Essentially gradient based using zero crossings of second derivative
- Typically, a single scale implementation (1 image) used with adjustable filter radius (smoothing parameter σ)

Canny Edge Detector (2)



Resulting edge maps for different settings of the smoothing (scale) parameter σ

Comparison of Various Edge Operators



Original — Roberts

Prewitt — Sobel

Laplacian of Gaussian — Canny ($\sigma = 1.0$)

Image Sharpening

- Blurring may occur during image scanning or scaling
- Sharpening reduces effects of blurring
- How? Amplify high frequency components
- High frequencies occur at edges
- We need to sharpen edges
- Two main approaches:
  - Using Laplace filter
  - Unsharp masking

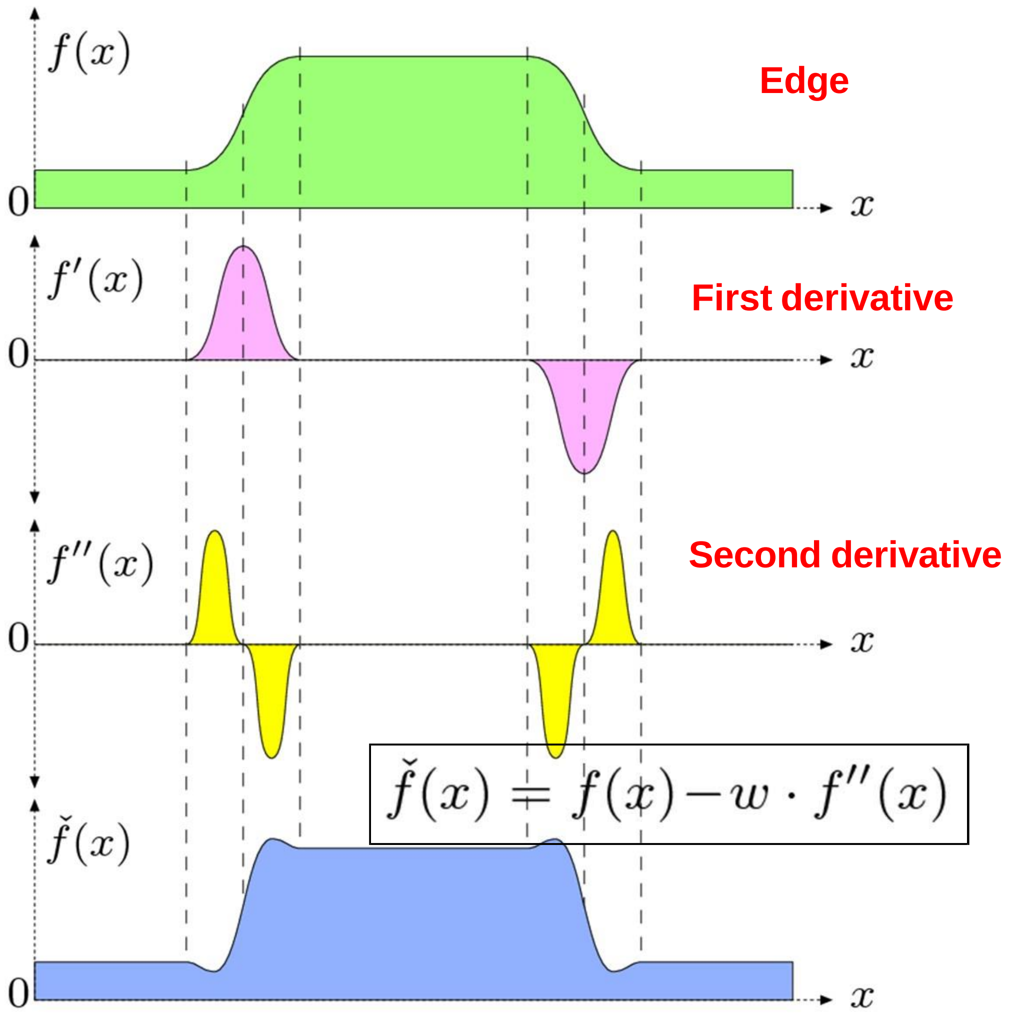
Edge Sharpening using Laplace Filter

Image intensity

$$\check{f}(x) = f(x) - w \cdot f''(x)$$

Weight

2nd derivative of intensity



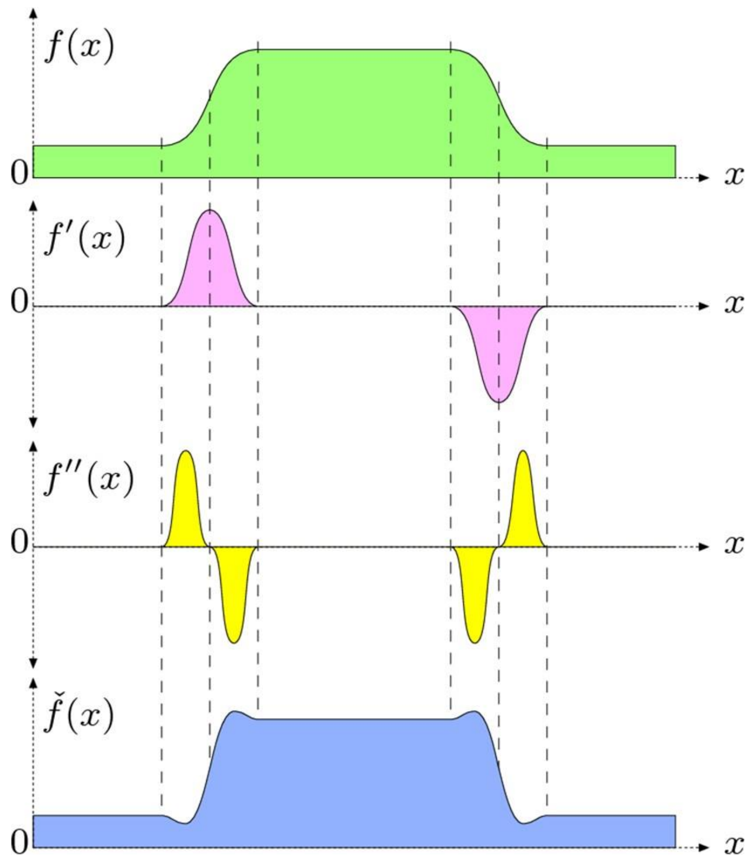
Laplace Operator

- **2D Laplace operator:** combines 2nd derivatives in horizontal and vertical directions
- Laplace operator defined as:

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

2nd derivative of intensity in x direction

2nd derivative of intensity in y direction



Laplacian Operator (1)

- Laplacian: $$(\nabla^2 f)(x,y) = \frac{\partial^2 f}{\partial^2 x}(x,y) + \frac{\partial^2 f}{\partial^2 y}(x,y)$$
- Digital approximation of laplacian is:

$$\nabla^2 f(x,y) = [f(x+1,y) - f(x,y)] - [f(x,y) - f(x-1,y)] + [f(x,y+1) - f(x,y)] - [f(x,y) - f(x,y-1)]$$

$$= [f(x+1,y) + f(x-1,y) + f(x,y+1) - f(x,y-1)] - 4f(x,y)$$

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Laplacian Operator (2)

- Laplacian: $$(\nabla^2 f)(x,y) = \frac{\partial^2 f}{\partial^2 x}(x,y) + \frac{\partial^2 f}{\partial^2 y}(x,y)$$

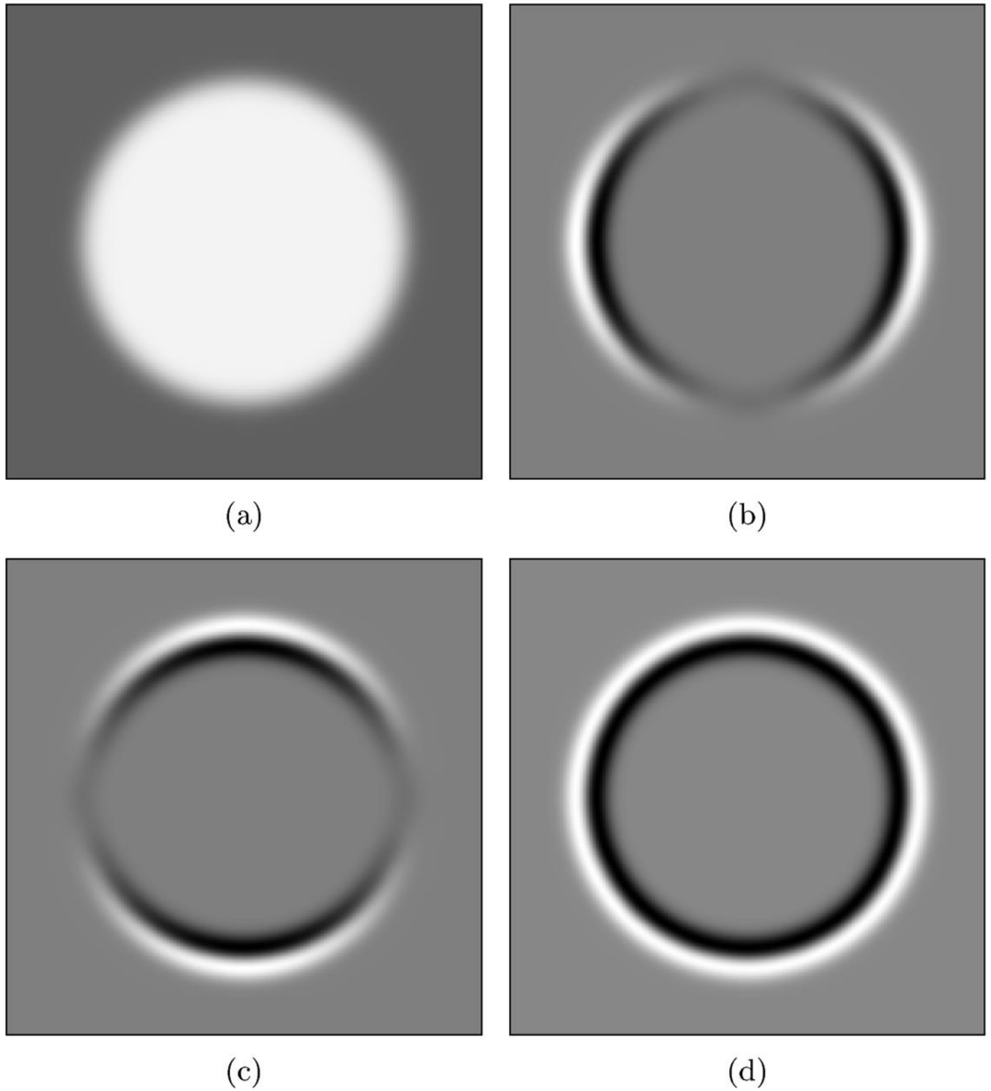
1d filters that estimate 2nd derivatives along x and y directions

$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & -\mathbf{2} & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -\mathbf{2} \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -\mathbf{4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2-dimensional Laplace filter

Results for Laplacian Operator



Synthetic test

Second partial derivative in horizontal direction

Second partial derivative in Vertical direction

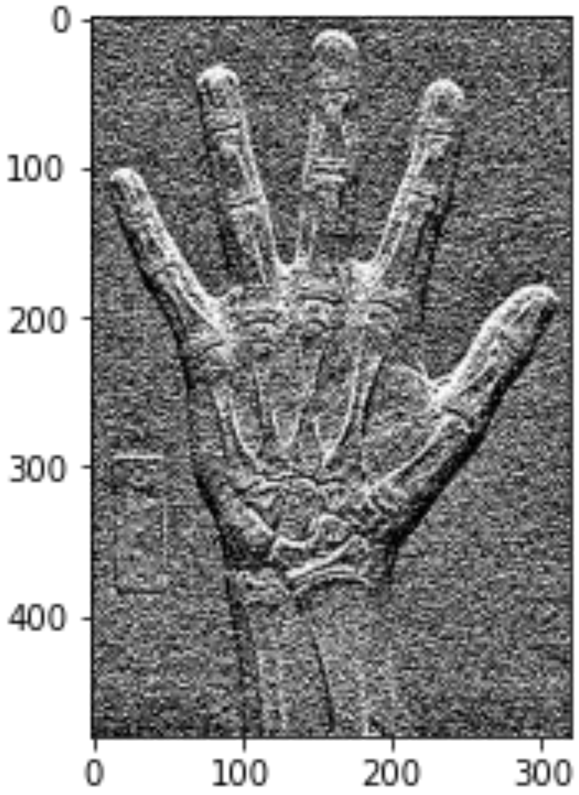
Laplace filter

How to Computer Edge Detection?

Gradient Vertical

```
vertical_kernel = np.array([[-1],[0],[1]])

gradient_vertical = ndi.convolve(imm,vertical_kernel)
fig, ax = plt.subplots()
ax.imshow(gradient_vertical, cmap='gray');
vertical_kernel
```

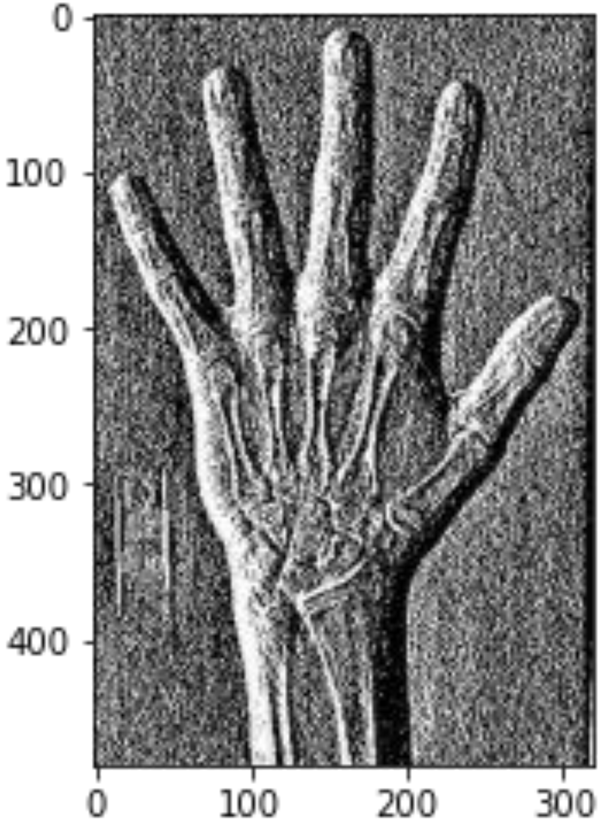
```
array([[-1],
       [ 0],
       [ 1]])
```



Gradient Horizontal

```
horizontal_kernel = vertical_kernel.T
gradient_horizontal = ndi.convolve(imm,horizontal_kernel)
fig, ax = plt.subplots()
ax.imshow(gradient_horizontal, cmap='gray');
horizontal_kernel
#print('dtype:', gradient_horizontal.dtype)
```

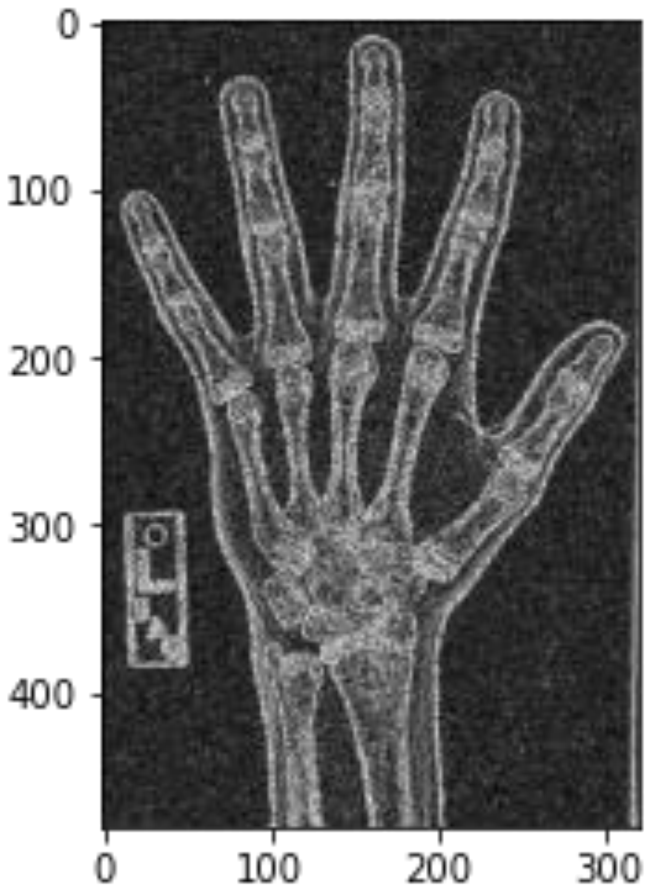
```
array([[-1,  0,  1]])
```



Gradient Magnitude

```python
gradient_mag = np.sqrt(gradient_vertical**2 + gradient_horizontal**2)
print('dtype:', gradient_mag.dtype)
gradient_mag

print('Min. value:', gradient_mag.min())
print('Max value:', gradient_mag.max())

#-------Untuk membenarkan clipping---#
im_eq = gradient_mag/np.amax(gradient_mag)  #
im_eq = np.clip(im_eq, 0, 255)  #

print('Min. value:', im_eq.min())
print('Max value:', im_eq.max())
print('dtype:', im_eq.dtype)

from skimage import img_as_ubyte
gradient_mag2 = img_as_ubyte(im_eq)  #convert to uint8
print('dtype:', gradient_mag2.dtype)

fig, ax = plt.subplots()
ax.imshow(gradient_mag2, cmap='gray')
plt.show()
```

```
dtype: float16
Min. value: 0.0
Max value: 15.91
Min. value: 0.0
Max value: 1.0
dtype: float16
dtype: uint8
```



Sobel Filter

```
# Apply Sobel filter along both axes
sobel_ax0 = ndi.sobel(imm, axis=0) #horizontal axis=0
sobel_ax1 = ndi.sobel(imm, axis=1) #vertikal axis=1
#sobel_horizontal_kernel = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
#sobel_vertical_kernel = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
#sobel_ax0 = ndi.convolve(imm,sobel_horizontal_kernel)
#sobel_ax1 = ndi.convolve(imm,sobel_vertical_kernel)

print('dtype:', sobel_ax0.dtype)

# Calculate edge magnitude
#edges = np.sqrt(np.square(sobel_ax0)+np.square(sobel_ax1))
edges = np.sqrt(sobel_ax0**2 + sobel_ax1**2)
print('dtype mag sobel:', edges.dtype)

print('Min. value:', edges.min())
print('Max value:', edges.max())
```

```
#-------Untuk membenarkan clipping---#
im_edges = edges/np.amax(edges)  #
im_edges = np.clip(im_edges, 0, 255)  #

print('Min. value:', im_edges.min())
print('Max value:', im_edges.max())
print('dtype:', im_edges.dtype)

from skimage import img_as_ubyte
edges2 = img_as_ubyte(im_edges)  #convert to uint8
print('dtype:', edges2.dtype)

# Plot edge magnitude
plt.imshow(edges2,cmap='gray')
```

```
dtype: uint8
dtype mag sobel: float16
Min. value: 0.0
Max value: 15.81
Min. value: 0.0
Max value: 1.0
dtype: float16
dtype: uint8
```