
**Computer Vision and Image Processing (CSEL-393)**

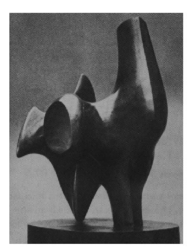
**Lecture**

Dr. Qurat ul Ain Akram
Assistant Professor
Computer Science Department (New Campus) KSK, UET, Lahore

# 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)

# Edge Detection



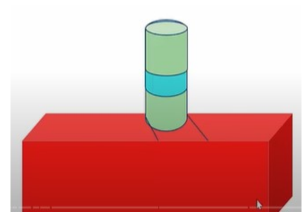
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

## Application

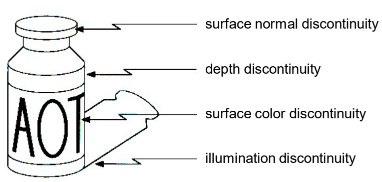
- What is an object
- How can we find it

## Edge Detection in images

- At edges intensity or color changes

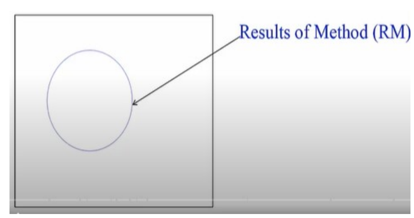


## Origin of Edges

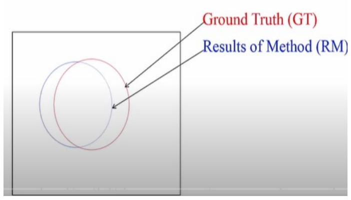



- Edges are caused by a variety of factors

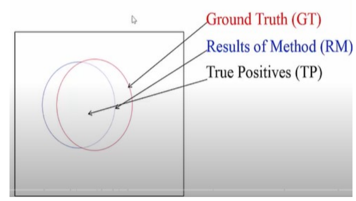
## Evaluation Metrics



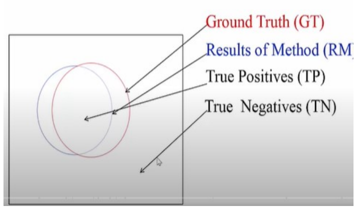
## Evaluation Metrics



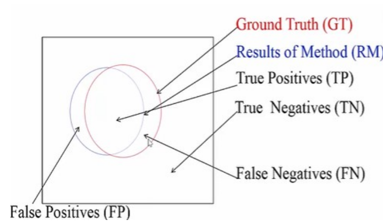

## Evaluation Metrics



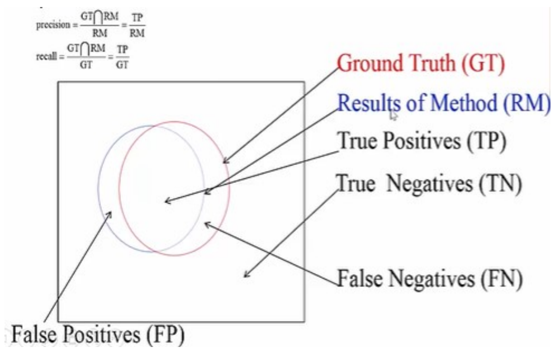

## Evaluation Metrics

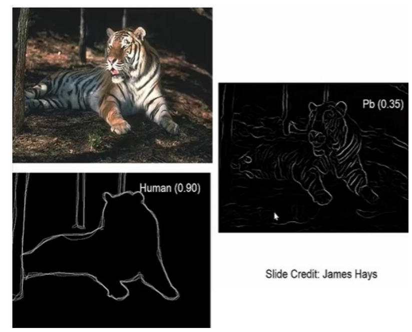



## Evaluation Metrics

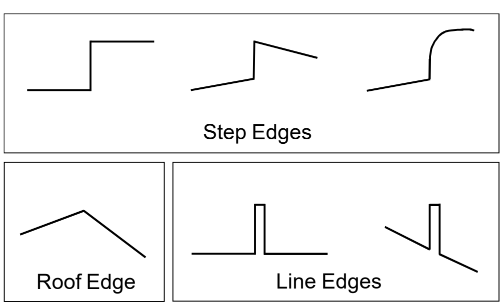


## Evaluation Metrics

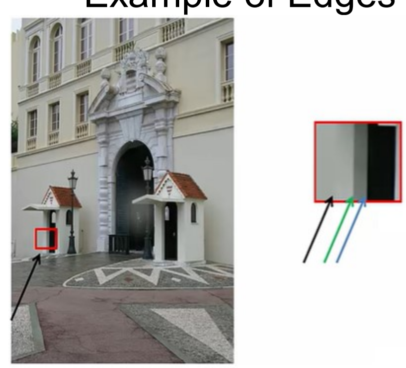


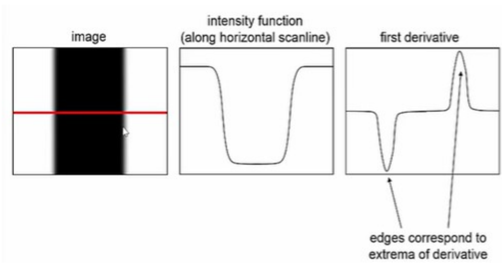

## Edge Types



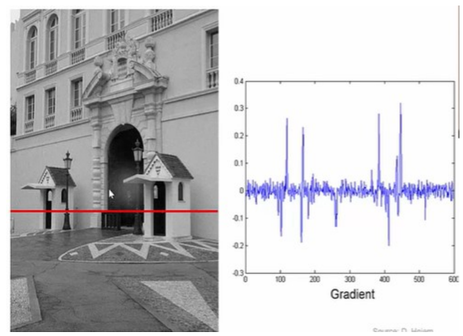

## Example of Edges



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

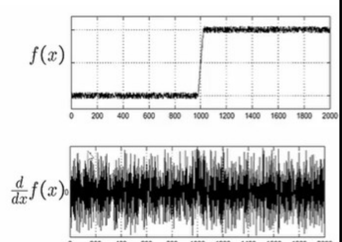



# Effect of noise




# Effect 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

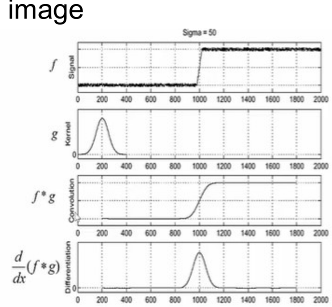



# Effects of Noise

- 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 can we do about it?

## Solution

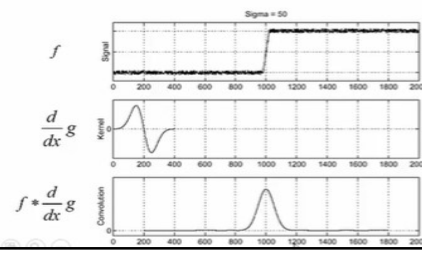
- First Smooth the image



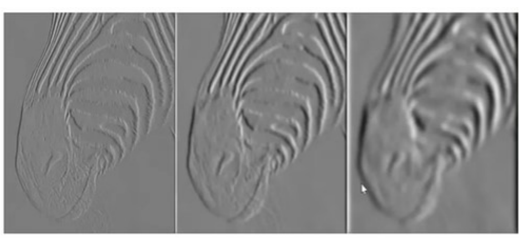

- To find edges, look for peaks in $\frac{d}{dx}(f*g)$

## Derivative Theorem of Smoothing

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




## Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

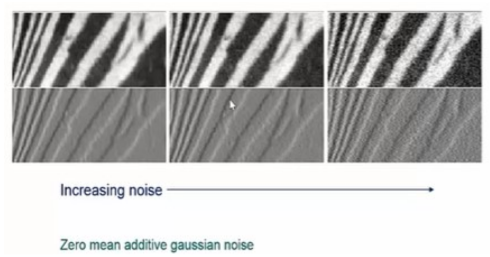
- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

## Derivatives and Noise

- Strongly affected by noise
  - obvious reason: image noise results in pixels that look very different from their neighbors
- The larger the noise is the stronger the response

- What is to be done?
  - Neighboring pixels look alike
  - Pixel along an edge look alike
  - Image smoothing should help
    - Force pixels different from their neighbors (possibly noise) to look like neighbors

## Derivatives and Noise



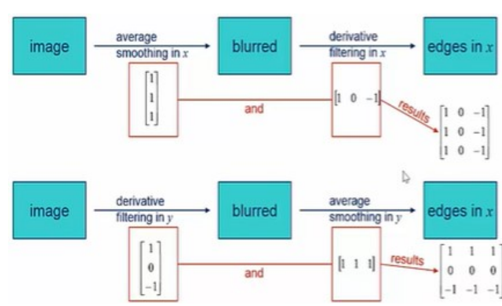

## Edge Detectors

- Gradient Operator
  - Prewitt
  - Sobel
- Laplacian of Gaussian
- Gradient of Gaussian (Canny Edge Detector)

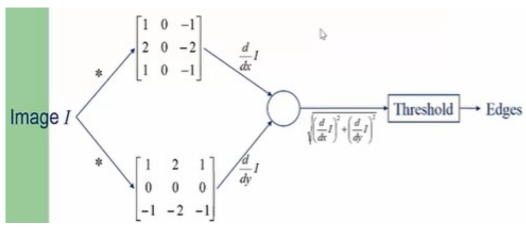
## Prewitt and Sobel Edge Detector

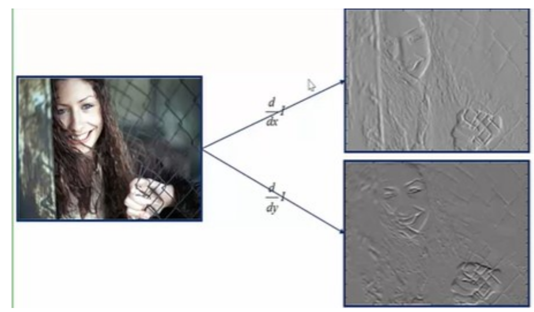
- Compute derivatives in x and y directions
- Find gradient magnitude
- Threshold gradient magnitude

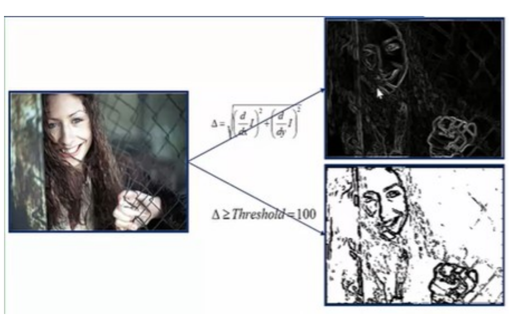

## Prewitt Edge Detector



## Sobel Edge Detector



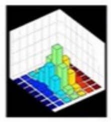

## Marr Hildreth Edge Detector

- Smooth image by Gaussian filter → S
- Apply Laplacian to S
  - Used in mechanics, electromagnetics, wave theory, quantum mechanics and Laplace equation
- Find zero crossings
  - Scan along each row, record an edge point at the location of zero-crossing
  - Repeat above step along each column

## Marr Hildreth Edge Detector

- Gaussian smoothing

$$\overbrace{S}^{\text{smoothed image}} = \overbrace{g}^{\text{Gaussian filter}} * \overbrace{I}^{\text{image}} \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



- Find Laplacian

$$\Delta^2 S = \overbrace{\frac{\partial^2}{\partial x^2} S}^{\text{secondorder derivative in } x} + \overbrace{\frac{\partial^2}{\partial y^2} S}^{\text{secondorder derivative in } y}$$

- $\nabla$ is used for gradient (first derivative)
- $\Delta^2$ is used for Laplacian (Secondt derivative

## Laplacian of Gaussian

- Deriving the Laplacian of Gaussian (LoG)

$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$g_x = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \left( -\frac{2x}{2\sigma^2} \right)$$

$$\Delta^2 g = -\frac{1}{\sqrt{2\pi}\sigma^3} \left( 2 - \frac{x^2+y^2}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

## Finding Zero Crossing

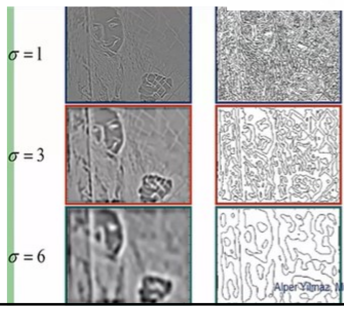
- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}
- Slope of zero-crossing {a, -b} is |a+b|.
- To mark an edge
  - compute slope of zero-crossing
  - Apply a threshold to slope

## Example

$I$ $\qquad$ $I * (\Delta^2 g)$ $\qquad$ Zero crossings of $\Delta^2 S$

## Example

$$\Delta^2 G_\sigma = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2-\frac{x^2+y^2}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$




## LOG Algorithm

- Apply LOG to the image
- Find Zero crossings of each row
- Find slope of zero crossing
- Apply threshold to the slope and mark edges

## Canny Edge Detection

- Canny Edge Detector Steps
  1. Smooth image with Gaussian filter
  2. Compute derivative of filtered image
  3. Find magnitude and orientation of gradient
  4. Apply "Non-maximum Suppression"
  5. Apply "Hysteresis Threshold" (use range between low and high)

## Home assignment

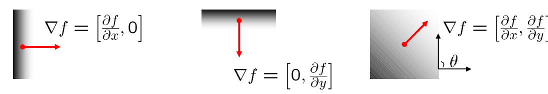
- Write a python code
  - Read an image
  - Find edges using
    1. Prewitt and sobel
    2. Laplacian of Gaussian (LOG)
    3. Canny
- Write image having marked edges on drive

# Readings

- Chapter
- Richard Szeliski, Computer Vision, Algorithms and Applications, 2nd Ed, https://szeliski.org/Book/

# Gradient

- Gradient equation: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Represents direction of most rapid change in intensity




- Gradient direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The *edge strength* is given by the gradient magnitude

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

# Discrete Edge Operators

- How can we differentiate a ***discrete*** image?

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$
$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$

| | |
|---|---|
| $I_{i,j+1}$ | $I_{i+1,j+1}$ |
| $I_{i,j}$ | $I_{i+1,j}$ |

$\varepsilon$

Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline\end{array} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & -1 \\ \hline\end{array}$$

# Discrete Edge Operators

- First order partial derivatives:
- 

$$\frac{\partial I}{\partial x} \approx \begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline\end{array} \qquad \frac{\partial I}{\partial y} \approx \begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & -1 \\ \hline\end{array}$$

- Second order partial derivatives:

| | | |
|---|---|---|
| $I_{i-1,j+1}$ | $I_{i,j+1}$ | $I_{i+1,j+1}$ |
| $I_{i-1,j}$ | $I_{i,j}$ | $I_{i+1,j}$ |
| $I_{i-1,j-1}$ | $I_{i,j-1}$ | $I_{i+1,j-1}$ |

- **Laplacian** :

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Convolution masks :

$$\nabla^2 I \approx \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array} \text{ or } \begin{array}{|c|c|c|}\hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline\end{array} \text{ (more accurate)}$$

## The Sobel Operators

- Better approximations of the gradients exist
  - The *Sobel* operators below are commonly used

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

$s_x$

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

$s_y$

## Comparing Edge Operators

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

Good Localization
Noise Sensitive
Poor Detection

Roberts (2 x 2):

| | |
|---|---|
| 0 | 1 |
| -1 | 0 |

| | |
|---|---|
| 1 | 0 |
| 0 | -1 |

Sobel (3 x 3):

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | 1 |

Sobel (5 x 5):

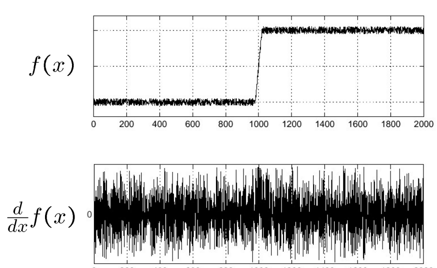
| | | | | |
|---|---|---|---|---|
| -1 | -2 | 0 | 2 | 1 |
| -2 | -3 | 0 | 3 | 2 |
| -3 | -5 | 0 | 5 | 3 |
| -2 | -3 | 0 | 3 | 2 |
| -1 | -2 | 0 | 2 | 1 |

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 2 | 1 |
| 2 | 3 | 5 | 3 | 2 |
| 0 | 0 | 0 | 0 | 0 |
| -2 | -3 | -5 | -3 | -2 |
| -1 | -2 | -3 | -2 | -1 |

Poor Localization
Less Noise Sensitive
Good Detection

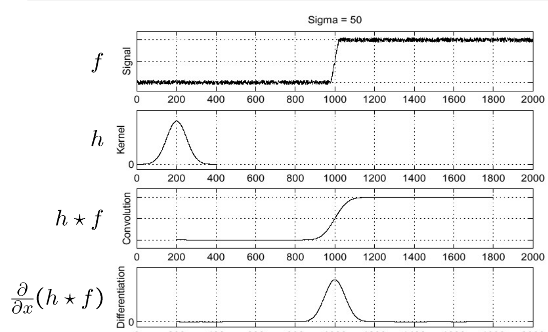
## Effects of Noise

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




Where is the edge??

## Solution: Smooth First




Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

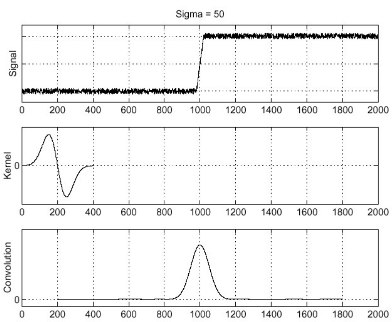
## Derivative Theorem of Convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$ …saves us one operation.

$f$

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$




## Laplacian of Gaussian (LoG)

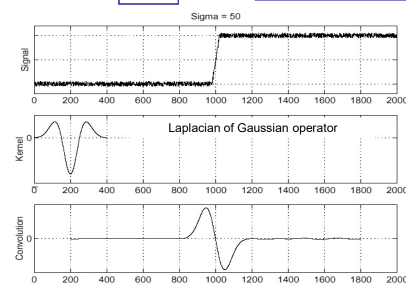
$$\frac{\partial^2}{\partial x^2}(h * f) = \boxed{\left(\frac{\partial^2}{\partial x^2} h\right)} * f$$

Laplacian of Gaussian

$f$

$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$




Where is the edge?

Zero-crossings of bottom graph !

## Canny Edge Operator

- Smooth image *I* with 2D Gaussian: $G * I$
- Find local edge normal directions for each pixel

$$\bar{\mathbf{n}} = \frac{\nabla(G * I)}{|\nabla(G * I)|}$$

- Compute edge magnitudes $|\nabla(G * I)|$
- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2 (G * I)}{\partial \bar{\mathbf{n}}^2} = 0$$

## The Canny Edge Detector

original image (Lena)

## The Canny Edge Detector

magnitude of the gradient

## The Canny Edge Detector

After non-maximum suppression

## Canny Edge Operator

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

- The choice of $\sigma$ depends on desired behavior
  - large $\sigma$ detects large scale edges
  - small $\sigma$ detects fine features