ABV-Indian Institute of Information Technology and Management Gwalior

==========

Edge Detection (ITIT-9507)

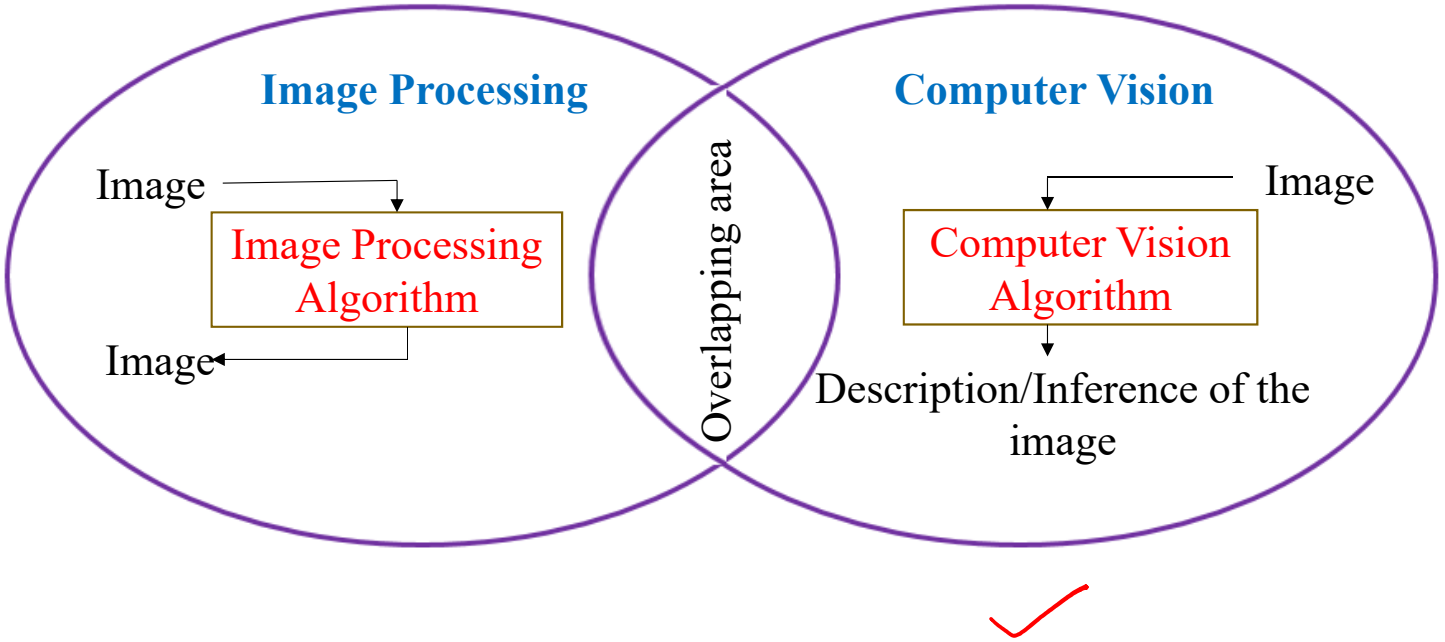
==========

Instructor – Dr. Sunil Kumar

Office – 206, F-Block (V), Tel No – 0751-2449710 (O), Email - snk@iiitm.ac.in

Mob - 8472842090

Image Processing Vs. Computer Vision



Motivation to Edge Detection

Motivation to Edge Detection



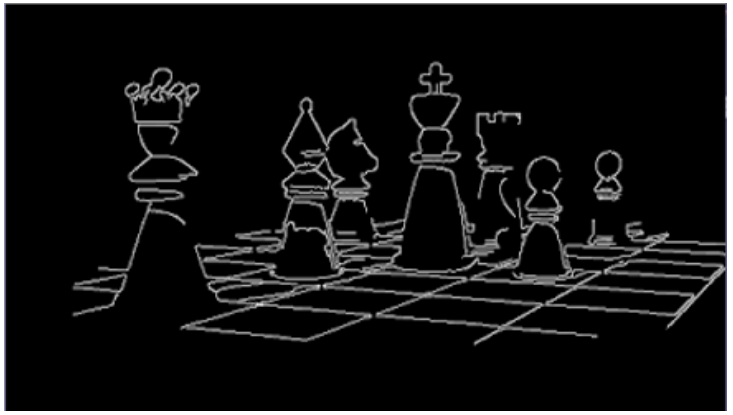
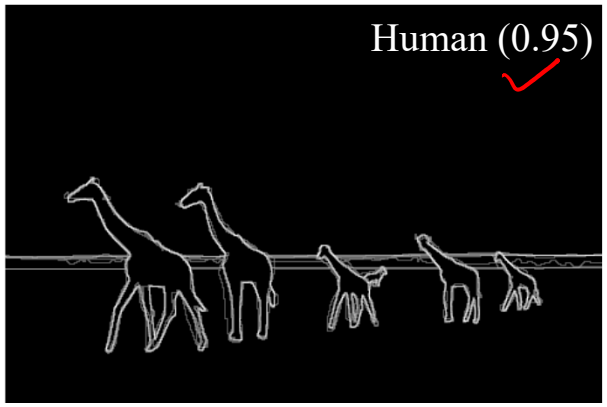
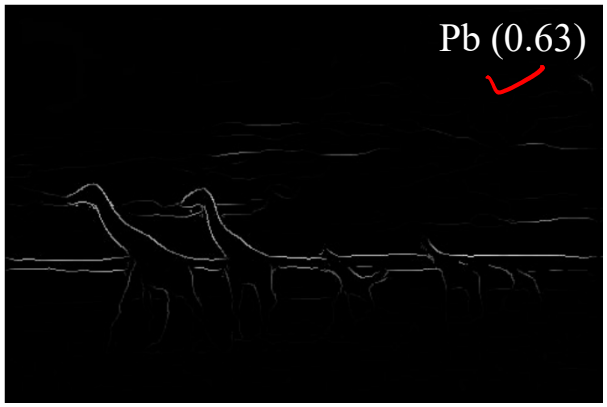
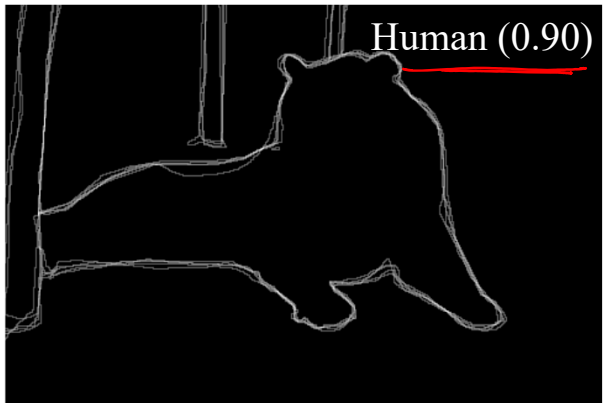



Image source : https://learnopencv.com/edge-detection-using-opencv/
http://opencv-cpp.blogspot.com/2016/11/canny-edge-detection-on-webcam.html

Motivation to Edge Detection





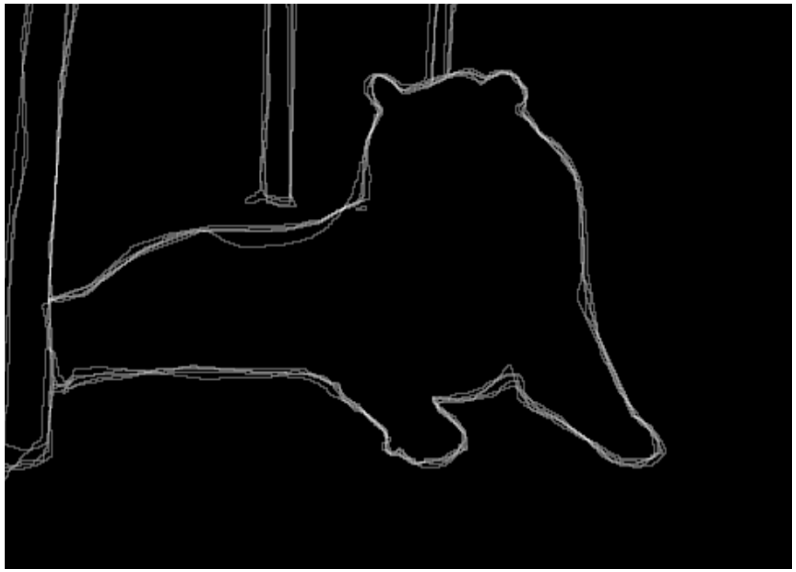


Slide Credit: James Hays

Key Applications

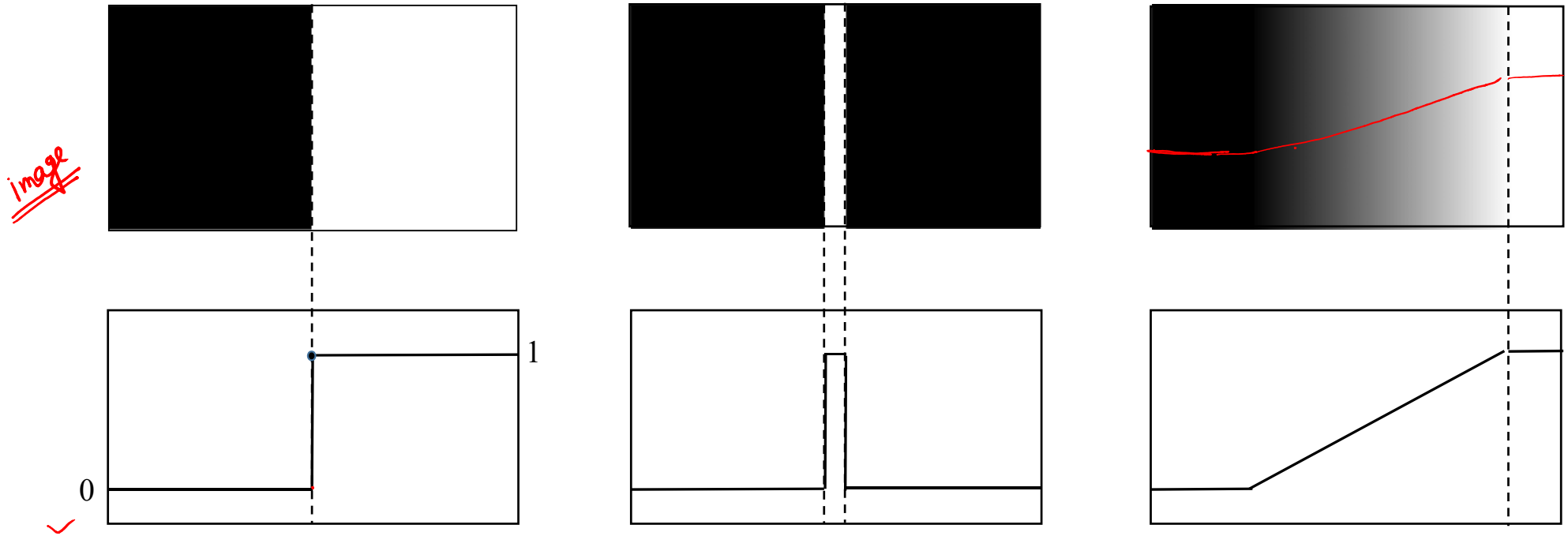
- Object detection / recognition

- Image segmentation



Edges in an Image

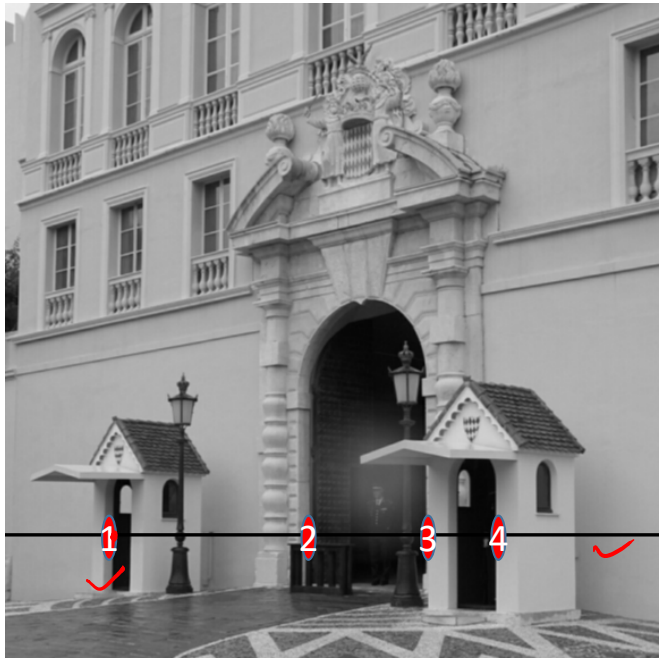
- What is an edge ?
  - Edges are pixels where there is a "change in intensity".

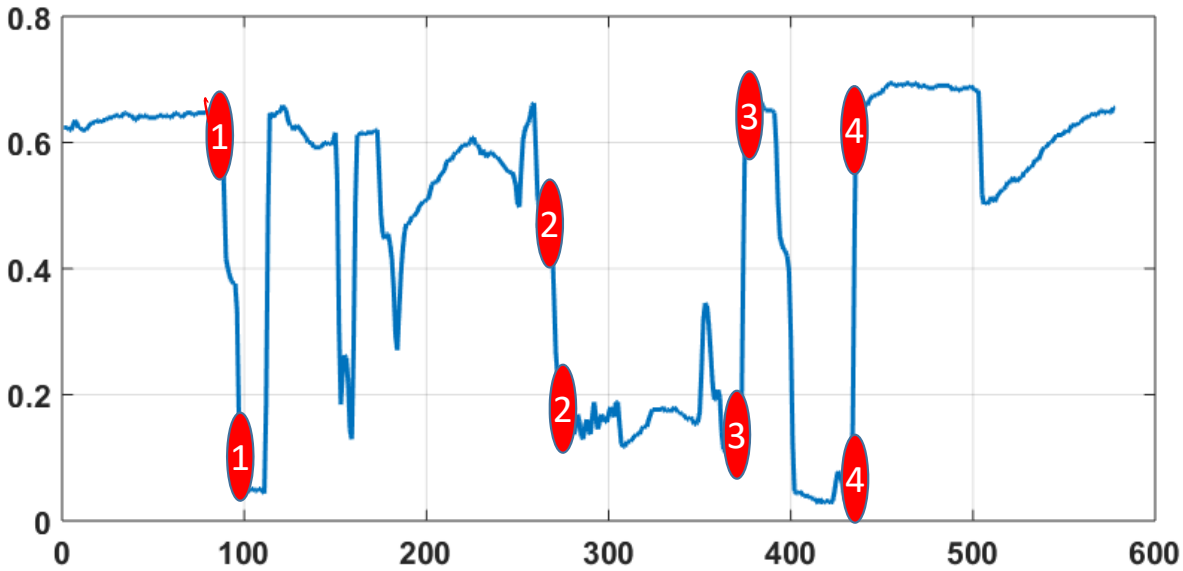


(a) Step-edge (b) Spike-edge (c) Ramp-edge

Edges Detection

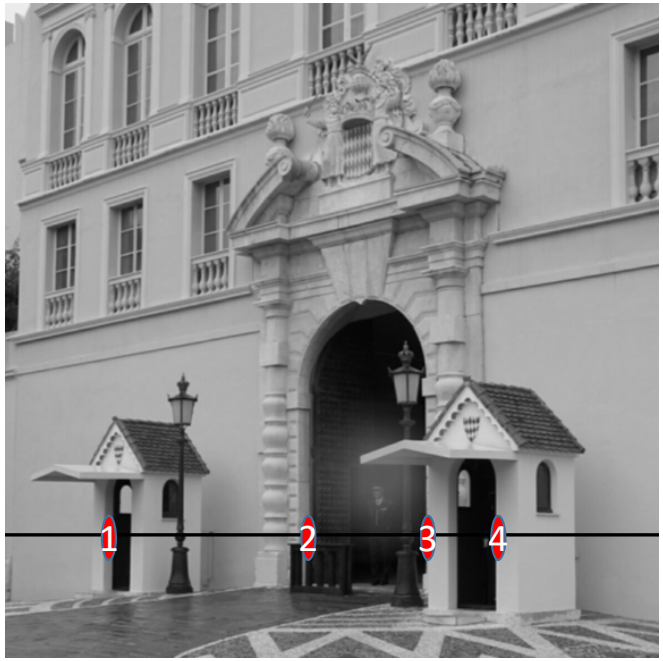
- "Change in intensity" or discontinuities in an image can be detected using "Image Derivative Filters".
- Goal : is to find sudden change (discontinuities) in a given image.

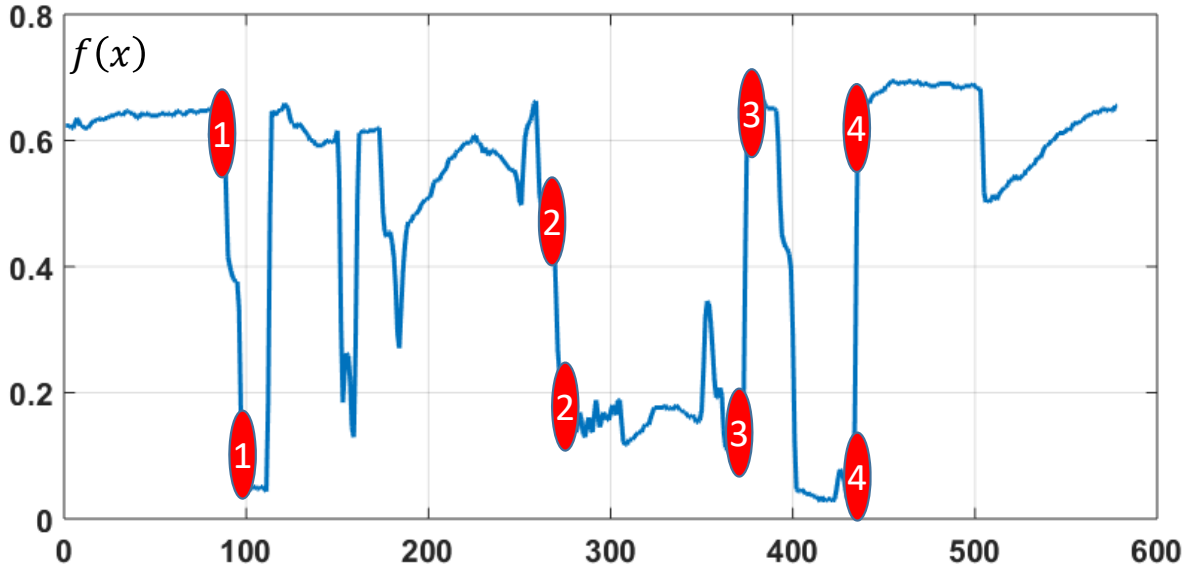


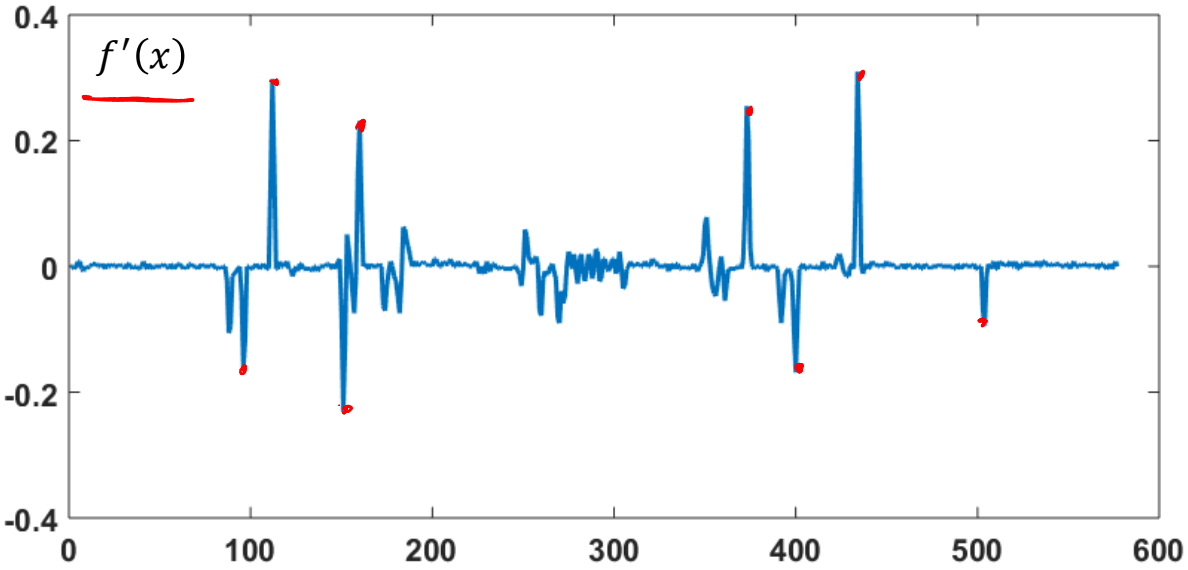


- Magnitude of change is high at the point of discontinuity

Edges Detection







Edges Detection

- "Change in intensity" or discontinuities in an image can be detected using "Image Derivative Filters".
- Goal : is to find sudden change (discontinuities) in a given image.

- 1D-derivatives : $\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h}$

- 2D-derivatives : an image $f(x, y)$ is a function of two independent space variables x and y.
  - **Case-1** : Analog image

$$\frac{\delta f}{\delta x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h} \qquad \frac{\delta f}{\delta y} = \lim_{h \to 0} \frac{f(x, y+h) - f(x, y)}{h}$$

Image Derivatives

- 2D-derivatives : an image $f(x, y)$ is a function of two independent space variables x and y.
  - **Case-2** : Digital image

$$\frac{\delta f}{\delta x} = f(x+1, y) - f(x, y)$$

Convolution kernel for f_x

−1	1

$$\frac{\delta f}{\delta y} = f(x, y+1) - f(x, y)$$

Convolution kernel for f_y

-1
1

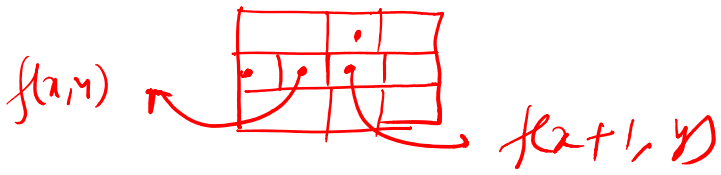


Image Derivatives

- 2D-derivatives : an image $f(x, y)$ is a function of two independent space variables x and y.
  - **Case-2** : Digital image

Convolution kernel

- Forward difference : $f_x = f(x, y+1) - f(x, y)$

-1	1

- Backward difference : $f_x = f(x, y) - f(x, y-1)$

-1	1

- Central difference : $f_x = f(x, y+1) - f(x, y-1)$

-1	0	1

- Similarly, find kernels for f_y

Edge Detectors

- Gradient-operator-based
  - Prewit
  - Sobel
- Laplacian of Gaussian
- Canny (Gradient of Gaussian)

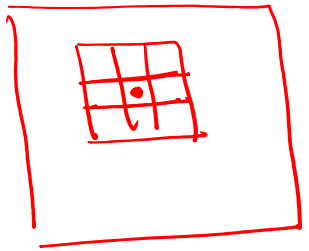
Prewit & Sobel Edge Detectors

- Problem : Given an image $f(x, y)$, find the edge pixel(s) in $f(x, y)$

- Steps for Prewit/Sobel edge detector:
  - Compute $f_x = \frac{\delta f}{\delta x} = f(x, y) * h_x$
  - Compute $f_y = \frac{\delta f}{\delta y} = f(x, y) * h_y$
  - Compute gradient magnitude : $\Delta f(x, y) = \sqrt{(f_x)^2 + (f_y)^2}$
  - Apply Threshold i.e., if $\Delta f(x, y) > T$ then $f(x, y)$ is an edge pixel else NOT.

Prewit & Sobel Edge Detectors

- Problem : Given an image $f(x, y)$, find the edge pixel(s) in $f(x, y)$



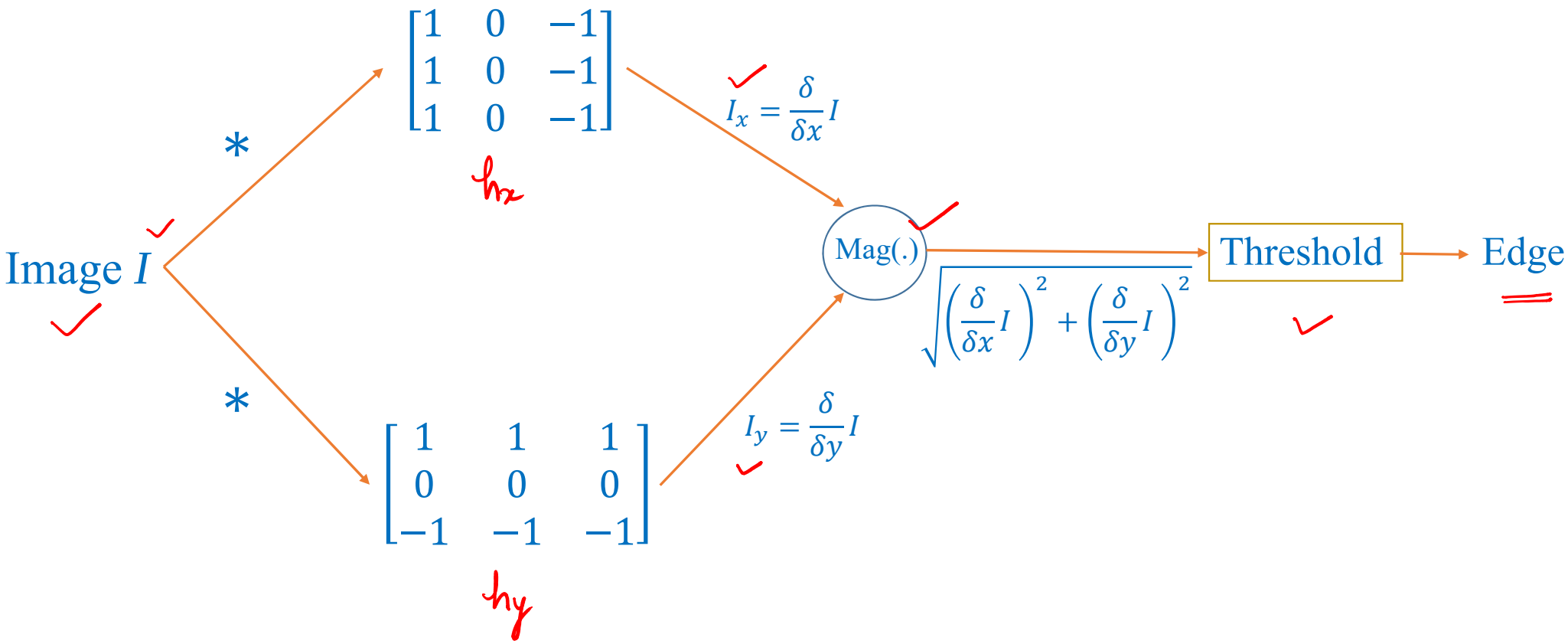
- For Prewit edge detector:

$$h_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \quad h_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

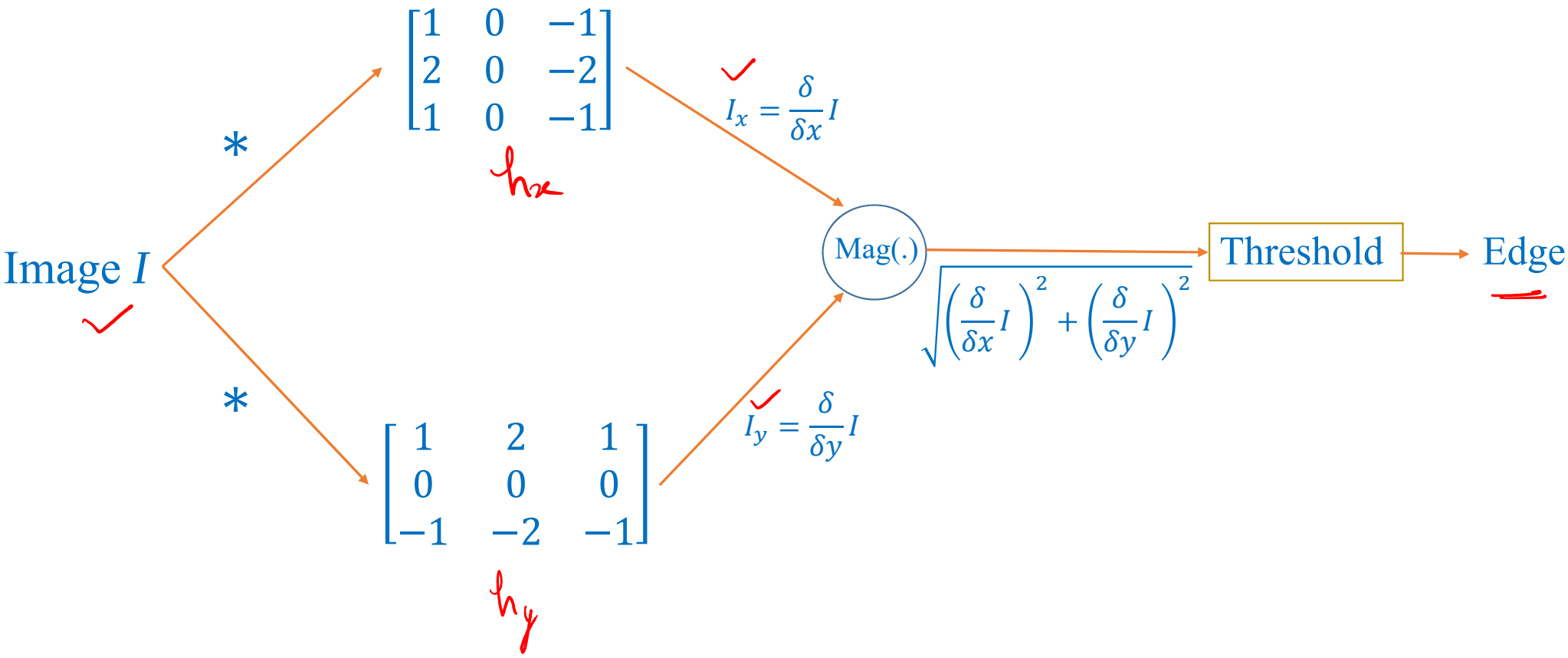
- For Sobel edge detector:

$$h_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad h_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Prewit Edge Detector



Sobel Edge Detector



Prewit Edge Detector : Result

$*$

I_x

$I_x = \frac{\delta}{\delta x} I$

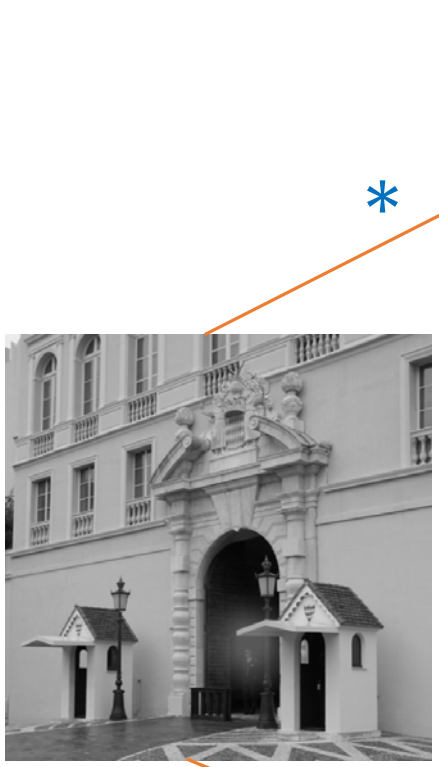
$*$

$I_y = \frac{\delta}{\delta y} I$

Mag(.)

>T

Sobel Edge Detector : Result



$*$

$I_x = \frac{\delta}{\delta x} I$

$*$

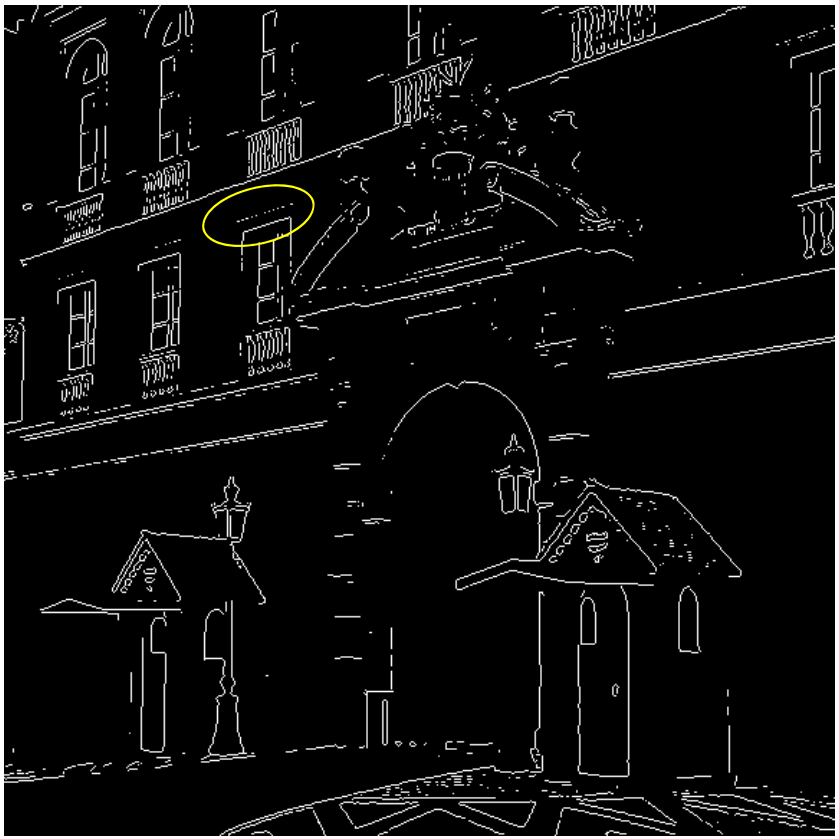
$I_y = \frac{\delta}{\delta y} I$

Mag(.)

>T

Prewit/Sobel Edge Detector : Comparison

"Prewit"



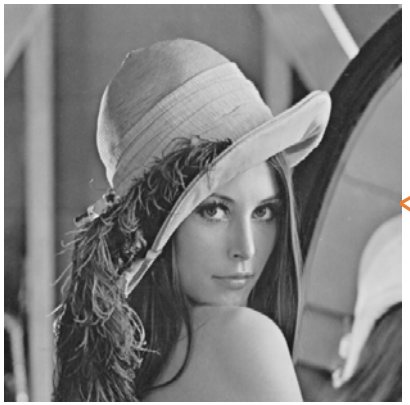
"Sobel"

Prewit/Sobel Edge Detector : Comparison

"Prewit"

"Sobel"

Prewit/Sobel Edge Detector : Comparison



$$\Delta = \sqrt{\left(\frac{\delta}{\delta x} I\right)^2 + \left(\frac{\delta}{\delta y} I\right)^2}$$

"Sobel"

$\Delta > 100$

$$\Delta = \sqrt{\left(\frac{\delta}{\delta x} I\right)^2 + \left(\frac{\delta}{\delta y} I\right)^2}$$

"Prewit"

$\Delta > 100$

Effect of Noise

$I_{original}(x, y)$

$I_{original}(x, y) + \eta_2(x, y) \sim g(0, 0.01)$

"Prewit"

False positive edges

"Sobel"

- Noise creates a lot of problems in order to get true edges.

Effect of Noise

$I_{original}(x, y) + \eta_2(x, y)$

Gaussian filtered image

"Prewit"

"Sobel"

False positive/negative edges got reduced

ABV-Indian Institute of Information Technology and Management Gwalior

==========

Edge Detection (ITIT-9507)

==========

Instructor – Dr. Sunil Kumar

Office – 206, F-Block (V), Tel No – 0751-2449710 (O), Email - snk@iiitm.ac.in

Mob - 8472842090

Marr Hildreth Edge Detector

- Based on Laplacian ($\nabla^2 = \frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2}$) of Gaussian
- Steps of Marr Hildreth Edge Detector:
  - Input image : $I_{original}$ (I_o)
  - Smooth I_o using Gaussian filter "g" : $I_s = I_o * g$
  - Apply Laplacian (∇^2) to I_s

    i.e., $\nabla^2 I_s = \nabla^2 (I_o * g) = I_o * \nabla^2 g$ (3)
  - Find zero-crossing :

    $\{+,-\}, \{-,+\}, \{+,\mathbf{0},-\}, \{-,\mathbf{0},+\}$
  - Find slope of zero-crossing {a,-b} =| a + b |
  - If slope is > Threshold (T) then mark that pixel as edge pixel (do for the pixels)

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

$\therefore f(x, y, z)$

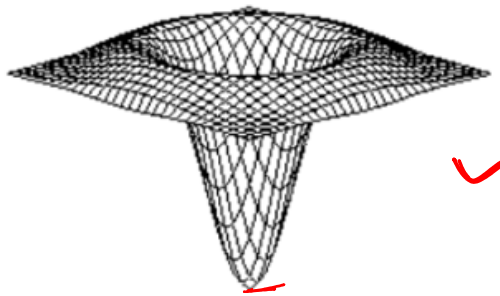
Laplacian (∇^2) of Gaussian

- Based on Laplacian (∇^2) of Gaussian
- $\nabla^2 f = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2} = f_{xx} + f_{yy}$
- Example-1: $f(x,y) = 3x^2y^3 + e^{xy}$ ← Analog image

$$\Rightarrow \nabla^2 f = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}$$

$$= [6y^3 + y^2 e^{xy}] + [36xy + x^2 e^{xy}]$$

- Example-2: $g(x,y) = e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$ ← 2D- Gaussian fn.

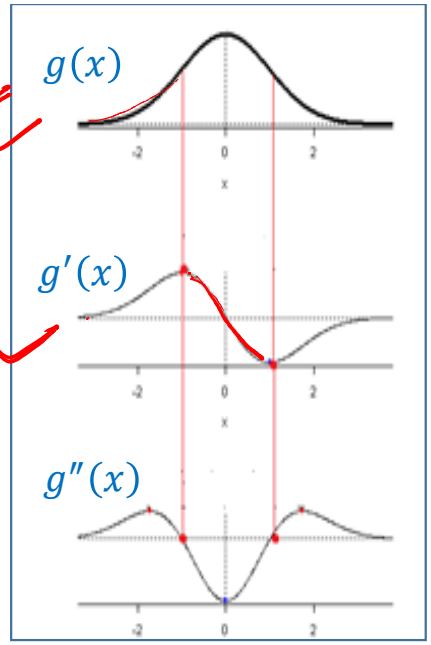


$\frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \Rightarrow \nabla^2 g = \frac{1}{2\sigma^2}\left[\frac{x^2+y^2}{2\sigma^2} - 2\right]$

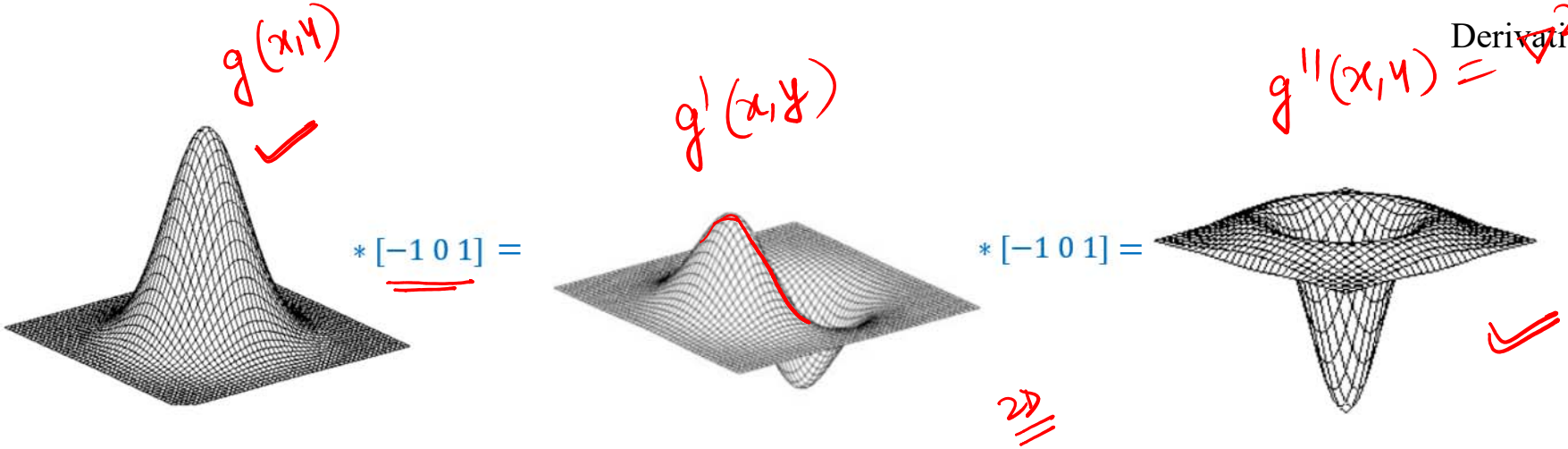
Laplacian (∇^2) of Gaussian in 2D

- Example-2: $g(x,y) = e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$

$$\Rightarrow \nabla^2 g = \frac{1}{2\sigma^2}\left[\frac{x^2+y^2}{2\sigma^2} - 2\right]$$



Derivatives of Gaussian in 1D



Laplacian (∇^2) of Gaussian Mask in 2D

- $\nabla^2 g = \frac{1}{2\sigma^2}\left[\frac{x^2+y^2}{2\sigma^2} - 2\right]$

3 × 3 LoG Mask

0.4038	0.8021	0.4038
0.8021	-4.8233	0.8021
0.4038	0.8021	0.4038

5 × 5 LoG Mask

0.0448	0.0468	0.0564	0.0468	0.0448
0.0468	0.3167	0.7146	0.3167	0.0468
0.0564	0.7146	-4.9048	0.7146	0.0564
0.0468	0.3167	0.7146	0.3167	0.0468
0.0448	0.0468	0.0564	0.0468	0.0448

7 × 7 LoG Mask

0.0228	0.0228	0.0228	0.0229	0.0228	0.0228	0.0228
0.0228	0.0229	0.0249	0.0345	0.0249	0.0229	0.0228
0.0228	0.0249	0.2948	0.6927	0.2948	0.0249	0.0228
0.0229	0.0345	0.6927	-4.9267	0.6927	0.0345	0.0229
0.0228	0.0249	0.2948	0.6927	0.2948	0.0249	0.0228
0.0228	0.0229	0.0249	0.0345	0.0249	0.0229	0.0228
0.0228	0.0228	0.0228	0.0229	0.0228	0.0228	0.0228

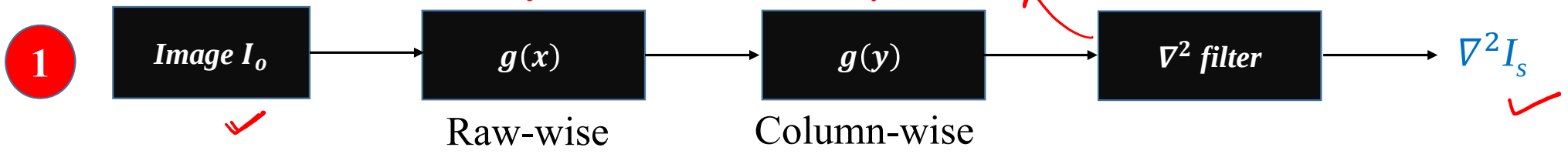
Convolution of Image with Laplacian (∇^2) of Gaussian Mask in 2D

- Let I is the image
- Output image : $\nabla^2 I_s = \nabla^2 (I_o * g) = I_o * \nabla^2 g$ → *Requires n^2 multiplication* (1)

$$= I_o * \nabla^2 g(x) g(y)$$

$$= [I_o * g_{xx}(x)] * g(y) + [I_o * g_{yy}(y)] * g(x)$$ (2)

→ *Requires 4n multiplication*



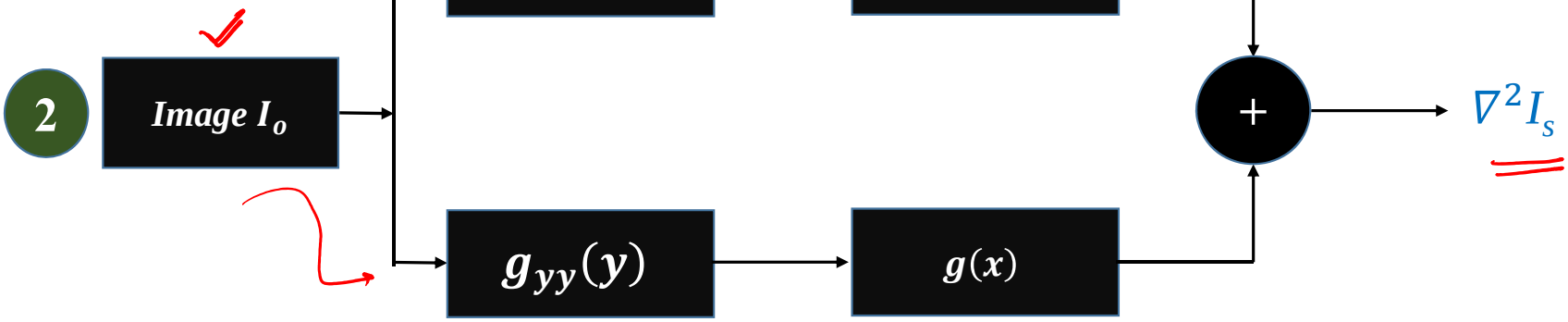
Convolution of Image with Laplacian (∇^2) of Gaussian Mask in 2D

- Let I is the image
- Output image : $\nabla^2 I_s = \nabla^2 (I_o * g) = I_o * \nabla^2 g$ → *Requires n^2 multiplication* (1)

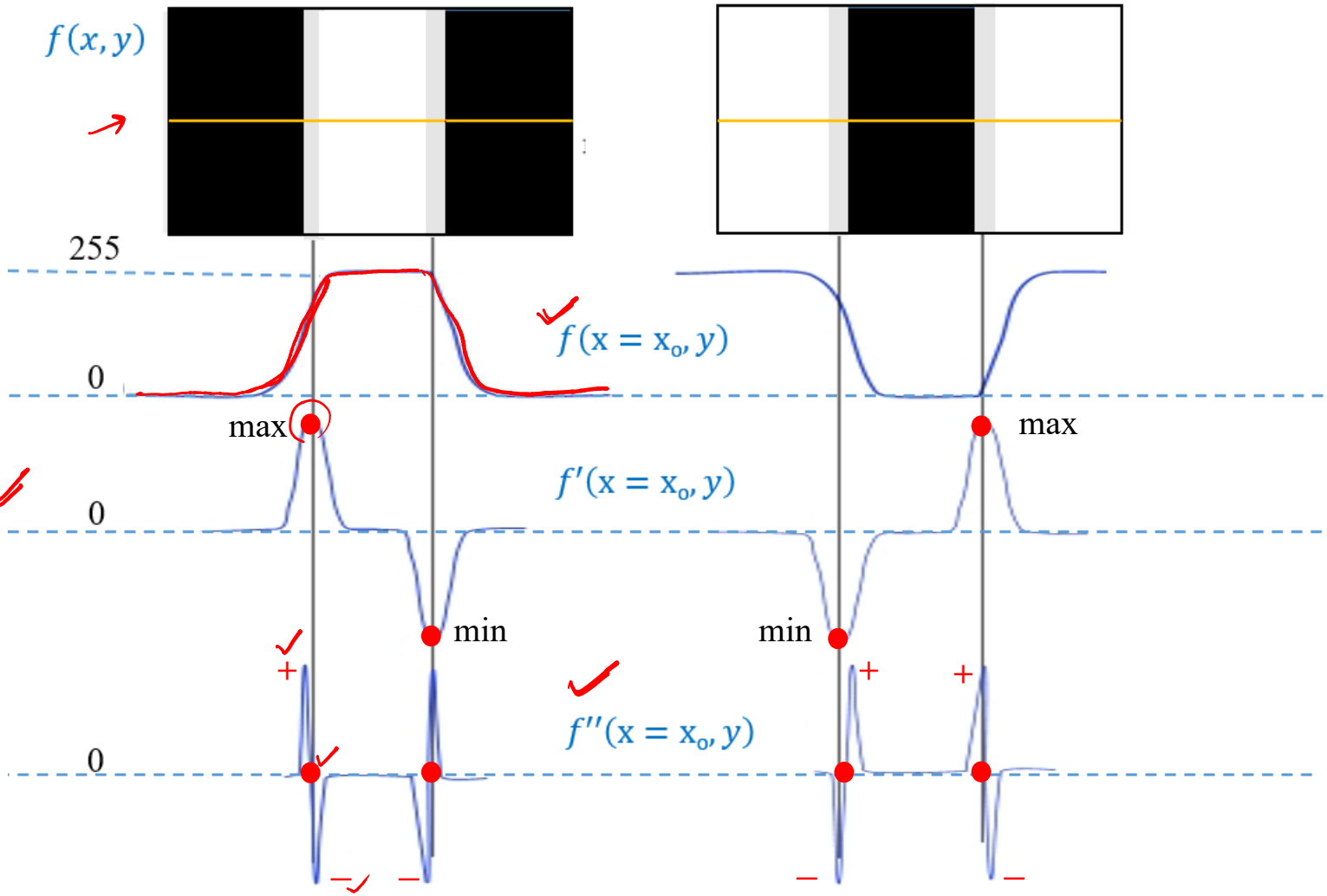
$$= I_o * \nabla^2 g(x) g(y)$$

$$= [I_o * g_{xx}(x)] * g(y) + [I_o * g_{yy}(y)] * g(x) \quad (3)$$

→ *Requires 4n multiplication*



Zero-Crossing : {+,-},{+,0,-},{-,+},{-,0,+}

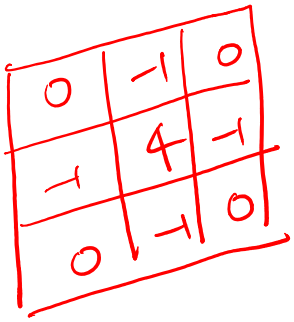


Slope of Zero-Crossing {a, -b}=|a + b|

- Find slope of zero-crossing {a,-b} =|a + b|
- At zero-crossing {a, -b} : if |a + b| > Threshold (*T*) then mark that pixel as edge pixel (do for all pixels).

Laplacian Filter

- $\nabla^2 f = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}$



- $\frac{\delta^2 f}{\delta x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$
- $\frac{\delta^2 f}{\delta y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$
- $\nabla^2 f = f(x+1,y) + f(x-1,y) - 4f(x,y) + f(x,y+1) + f(x,y-1)$

Example-

a	b	c
d	e	f
g	h	i

Image

$*$

0	1	0
1	−4	1
0	1	0

Laplacian kernel

$=$

	z	

Output

$z = b + d + f + h - 4e$

Marr Hildreth Edge Detector : Results

- **Experiment-1 :**

"Original"

"Prewit"

"Sobel"

"LoG"

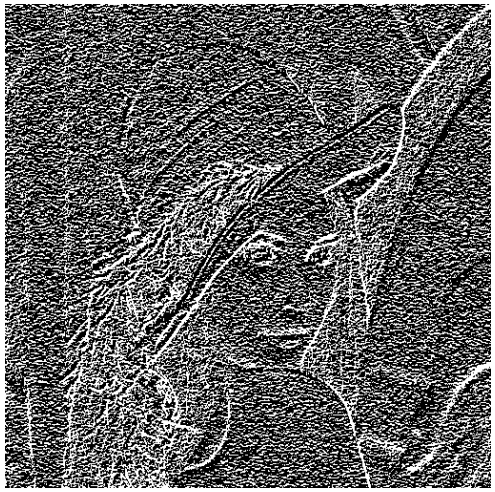
Marr Hildreth Edge Detector : Results

- **Experiment-2 :**

"Original"

$I_{original}(x, y) + \eta_2(x, y)$

"Prewit"

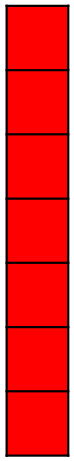


"Sobel"

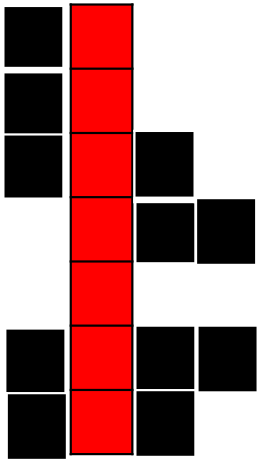
"LoG"

Canny Edge Detector

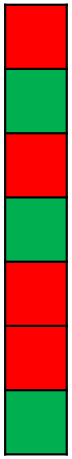
- What are the characteristics of a good edge detector?
  - Criteria – 1 : Minimum false positive and false negative edges



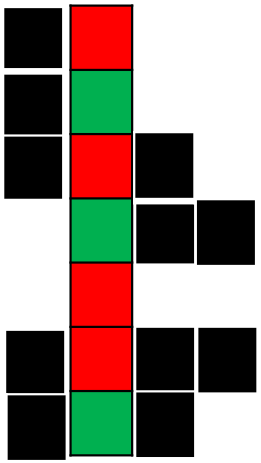
True Edge Pixels



False Positive Edge Pixels



False Negative Edge Pixels

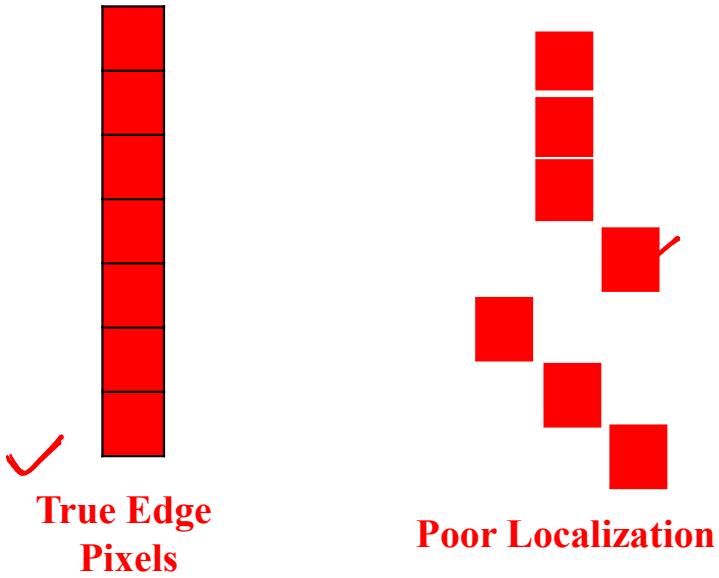


Both, False Negative and False Negative responses

Canny Edge Detector

- What are the characteristics of a good edge detector?

  - Criteria – 2 : A optimal edge detector should have good localization.



Credit - Alper Yilmaz, Mubarak Shah

Five Steps of Canny Edge Detector

- Let I_o is the input image
  - Step-1 : Smooth I_o using Gaussian filter "g" : $I_s = I_o * g$
  - Step-2 : Compute derivatives of filtered image I_s i.e., $\nabla I_s = \left[\frac{\delta}{\delta x} I_s \quad \frac{\delta}{\delta y} I_s\right]'$

    Let $I'_{sx} = \frac{\delta}{\delta x} I_s$ and $I'_{sy} = \frac{\delta}{\delta x} I_s$
  - Step-3 : Compute gradient magnitude and orientation :

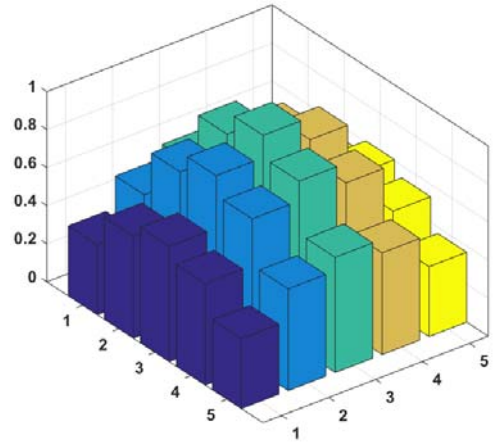
    $$|\nabla I_s| = \sqrt{(I'_{sx})^2 + (I'_{sy})^2} \text{ and } \phi = \tan^{-1}\left(\frac{I'_{sy}}{I'_{sx}}\right)$$
  - Step-4 : Apply "non-maximum suppression"
  - Step-5 : Apply "Hysteresis Threshold (Double Threshold)"

Five Steps of Canny Edge Detector

- Let I_o is the input image
  - Step-1 : Smooth I_o using Gaussian filter "g" : $I_s = I_o * g$

Input image : I_o

$*$



5 × 5 Gaussian Mask

g

$=$

$I_s = I_o * g$

Five Steps of Canny Edge Detector

❖ Step-2 : Compute derivatives of filtered image I_s i.e.,∇I_s

$$\nabla I_s = \nabla(I_o * g) = \nabla(g * I_o) = (\nabla g) * I_o = (\nabla I_o) * g = \left[\frac{\delta}{\delta x} I_s \quad \frac{\delta}{\delta y} I_s\right]'$$

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

$\frac{\delta}{\delta x} I_s$

Gradient along x-direction

Five Steps of Canny Edge Detector

- Step-2 : Compute derivatives of filtered image I_s i.e.,∇I_s

$$\nabla I_s = \nabla(I_o * g) = \nabla(g * I_o) = (\nabla g) * I_o = (\nabla I_o) * g = \left[\frac{\delta}{\delta x} I_s \quad \frac{\delta}{\delta y} I_s\right]'$$

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

$\frac{\delta}{\delta y} I_s$

Gradient along y-direction

Five Steps of Canny Edge Detector

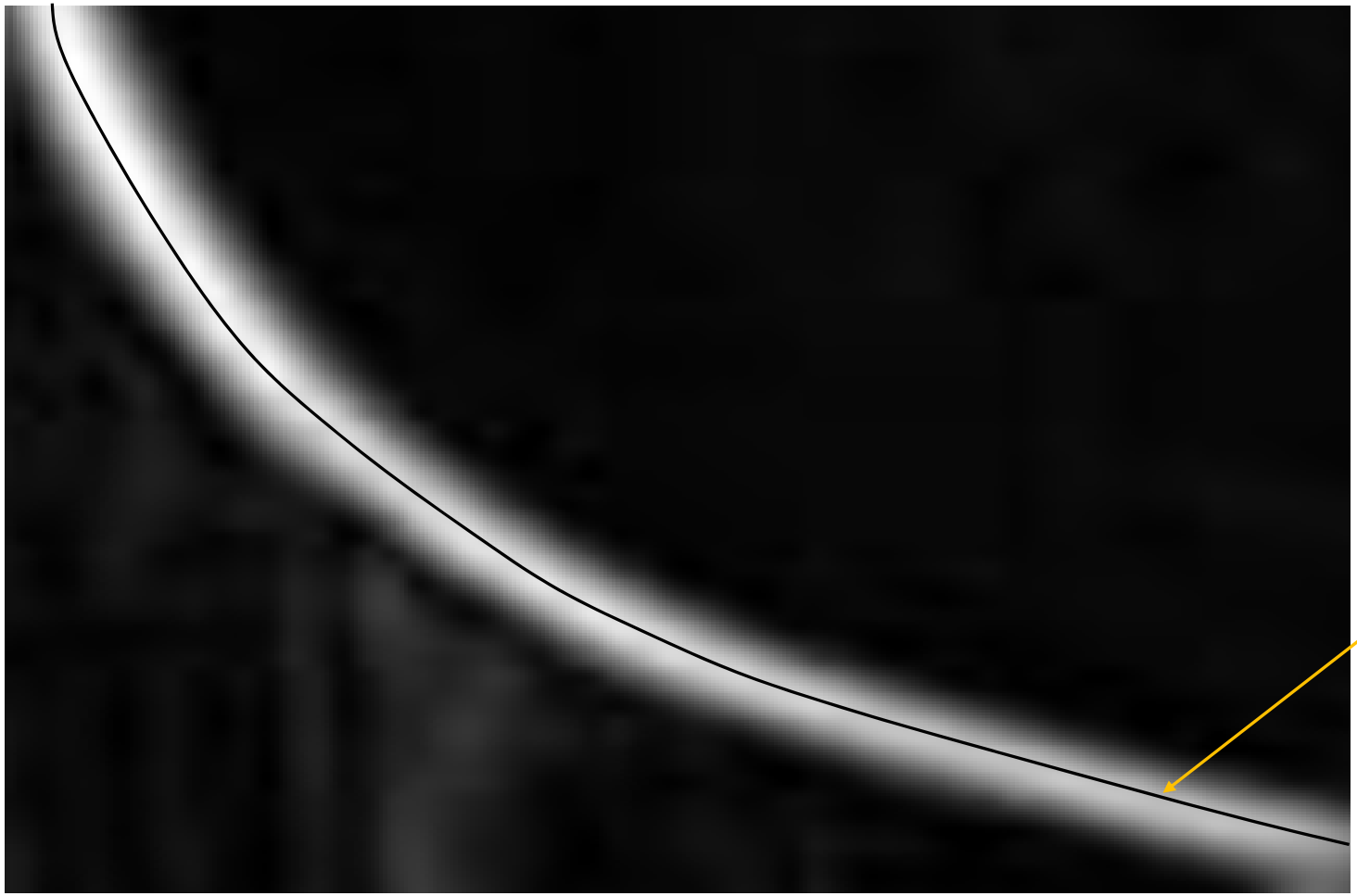
- Step-3 : Compute gradient magnitude and orientation :

$$|\nabla I_s| = \sqrt{(I'_{sx})^2 + (I'_{sy})^2} \text{ and } \phi = \tan^{-1}\left(\frac{I'_{sy}}{I'_{sx}}\right)$$

$|\nabla I_s|$

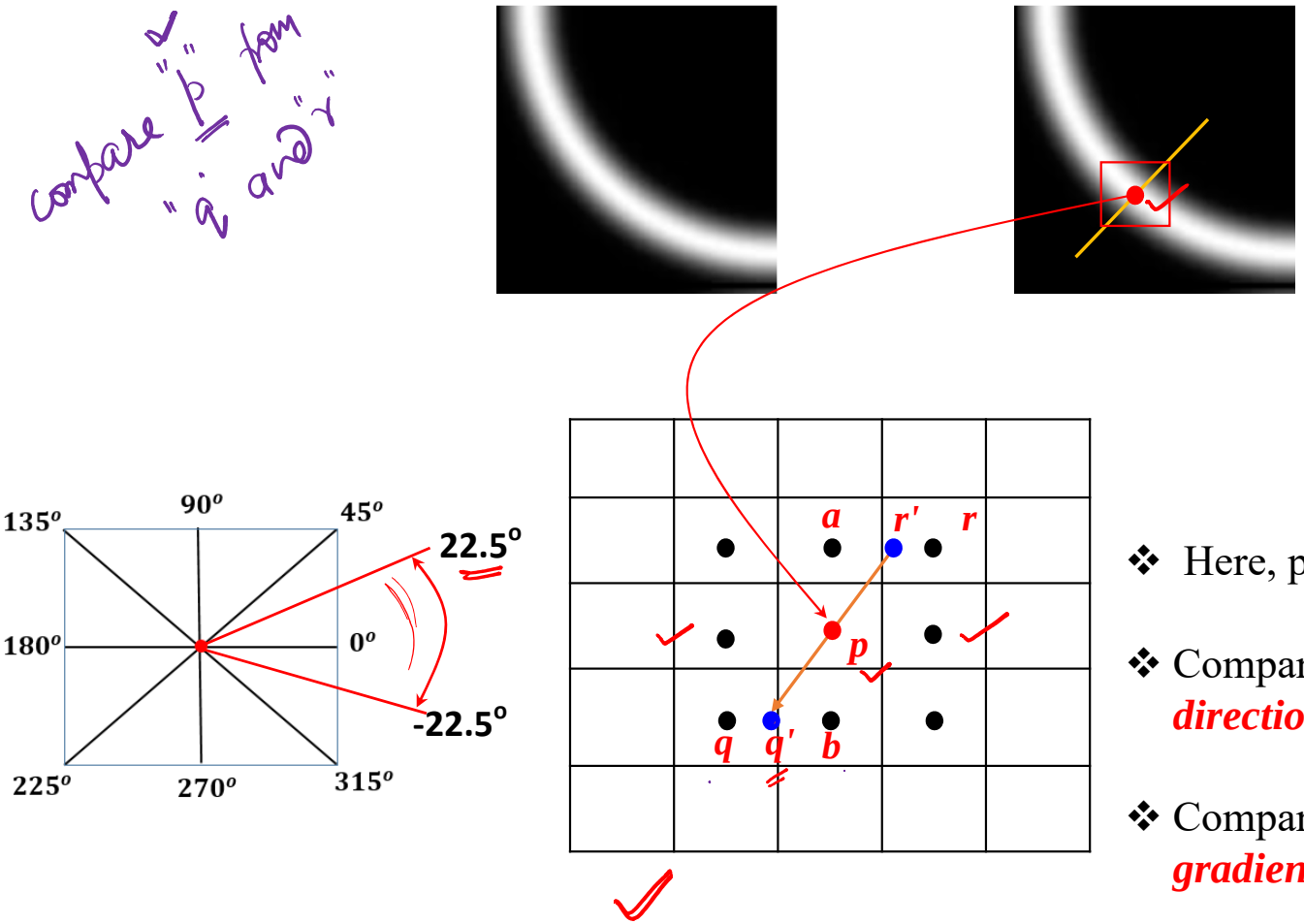
- In gradient magnitude, edges are thick. However, the true edges are actually thin.
- Thinning of edges are performed using "Non-maximum Suppression"

Five Steps of Canny Edge Detector



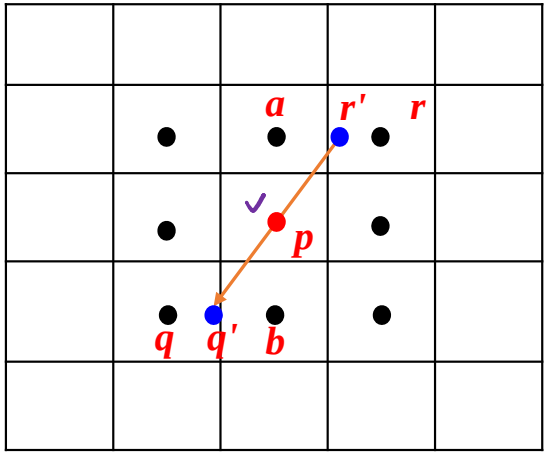
$|\nabla I_s|$

Five Steps of Canny Edge Detector



- Here, pixels represent gradient magnitude $|\nabla I_s|$
- Compare ***p*** with ***r and q (approximate gradient direction)***
- Compare ***p*** with ***r' and q' (interpolated gradient magnitudes)***

Five Steps of Canny Edge Detector



p > q'
p > r'

❑ Let $p \equiv (x, y)$, $q' \equiv (x', y')$, and $r' \equiv (x'', y'')$

$$M(x,y) = \begin{cases} |\nabla I_s|_{(x,y)} & ; |\nabla I_s|_{(x,y)} > |\nabla I_s|_{(x',y')} \text{ and } |\nabla I_s|_{(x,y)} > |\nabla I_s|_{(x'',y'')} \\ 0 & ; \text{otherwise} \end{cases}$$

i.e., If gradient magnitude of a pixel is greater than the local gradient magnitude of local pixels directed along gradient direction at pixel of concern than "*RETAIN*" that pixel else make it zero.

Five Steps of Canny Edge Detector

Before “Non-maximum Suppression”

After “Non-maximum Suppression”

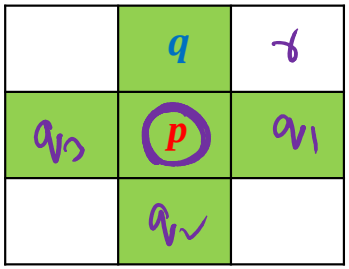
Five Steps of Canny Edge Detector

- Step-5 : Apply "Hysteresis Threshold (Double Threshold)"
  - Let T_h and T_l are two thresholds such that $T_h > T_l$
  - Case-1: If $|\nabla I_s|_{(x,y)} > T_h$ then declare pixel at (x, y) as an "edge pixel"
  - Case-2: If $|\nabla I_s|_{(x,y)} < T_l$ then declare pixel at (x, y) as a "Non-edge pixel"
  - Case-3: If $T_l < |\nabla I_s|_{(x,y)} < T_h$ then :
    - If the pixel at (x, y) is "connected" to any other pixel that has gradient magnitude greater than higher threshold (T_h) then declare that pixel as an "edge pixel".
    - If "NOT CONNECTED" then declare that pixel as an "Non-edge pixel".

Five Steps of Canny Edge Detector

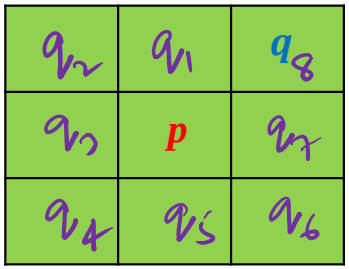
- Step-5 : Apply "Hysteresis Threshold (Double Threshold)"

4-connectivity



$q \in N_4(p)$

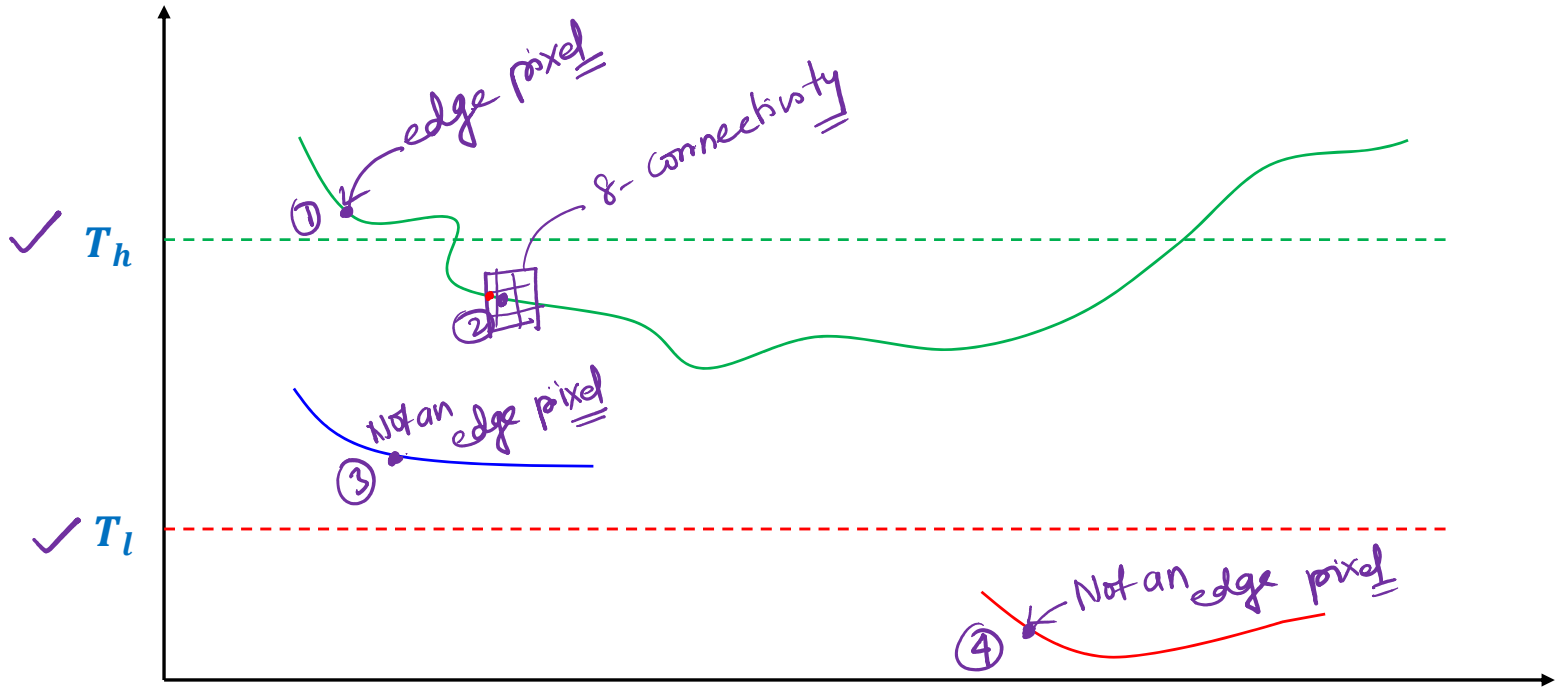
8-connectivity



$q \in N_8(p)$

Five Steps of Canny Edge Detector

- Step-5 : Apply "Hysteresis Threshold (Double Threshold)"



Five Steps of Canny Edge Detector

- Step-5 : Apply "Hysteresis Threshold (Double Threshold)"

Before "Hysteresis Thresholding"

After "Hysteresis Thresholding"

Slide Credit: James hays and Justin Liang

Five Steps of Canny Edge Detector

"Original"

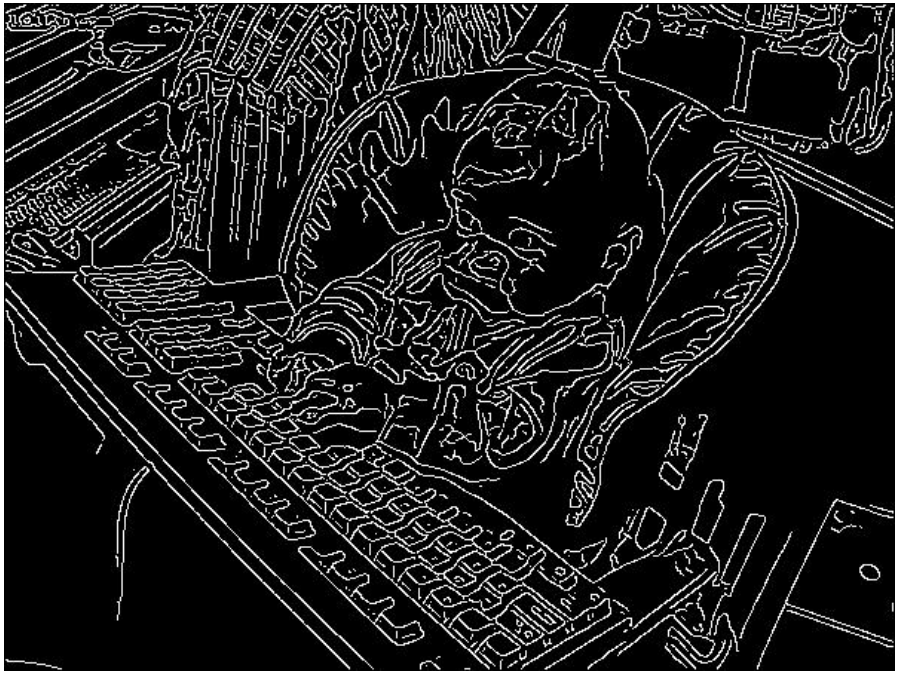
Canny

"Prewit"

"Sobel"

"LoG"

Canny Edge Detector : More Results



Slide Credit: Justin Liang

https://justin-liang.com/tutorials/canny/

Reference

- Richard Szeliski, Computer Vision: Algorithms and Applications, Springer, 2010 (online draft),
- Mubarak Shah, "Fundamentals of Computer Vision" (Online available)
- Ian Goodfellow, Yoshua Bengio and Aaron Courville, "Deep Learning" (Online available)

Edges Detection

- "Change in intensity" or discontinuities in an image can be detected using "Image Derivative Filters".
- Goal : is to find sudden change (discontinuities) in a given image.

Prewit Edge Detector

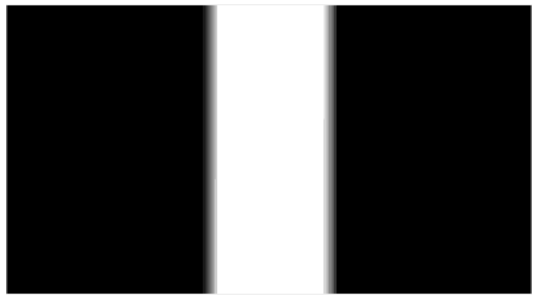


Image I

$*$

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

$$I_x = \frac{\delta}{\delta x} I$$

$*$

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

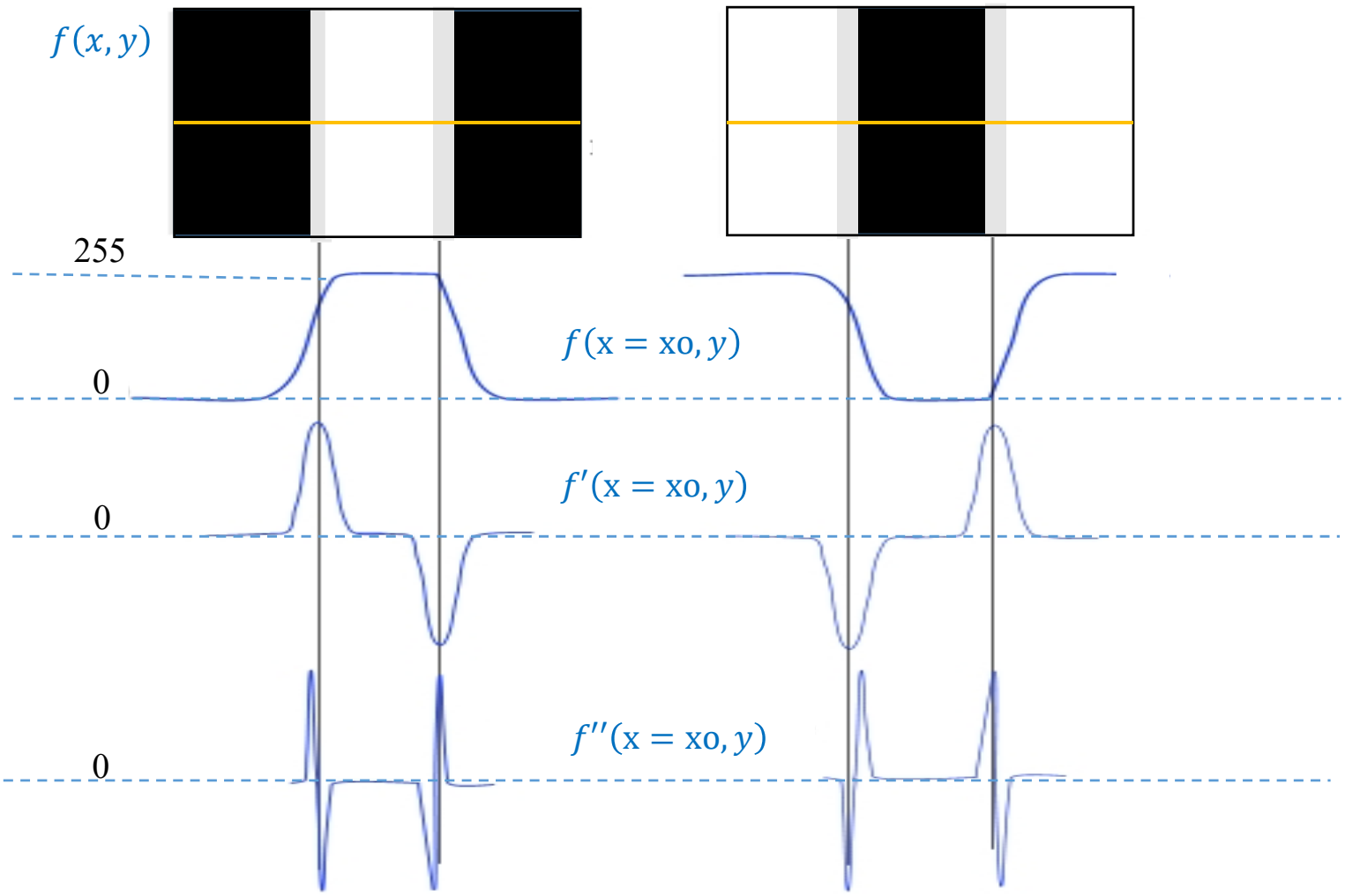
$$I_y = \frac{\delta}{\delta y} I$$

Mag(.)

$$\sqrt{\left(\frac{\delta}{\delta x} I\right)^2 + \left(\frac{\delta}{\delta y} I\right)^2}$$

Threshold → Edge

Zero-Crossing : {+,-},{+,0,-},{-,+},{-,0,+}



Acknowledgement!

Sources for this lecture include materials from works by Szeliski, Abhijit Mahalanobis, Sedat Ozer, Ulas Bagci, Mubarak Shah, Antonio Torralba, D. Hoiem, Justin Liang, and others. References are given for the source image contents.

Queries!