Edge detection reduces the amount of data in an image while preserving the structural properties of an image.

Edge Detection Operators are of two types:

Gradient – based operator which computes first-order derivations in a digital image

Gaussian – based operator which computes second-order derivations in a digital image

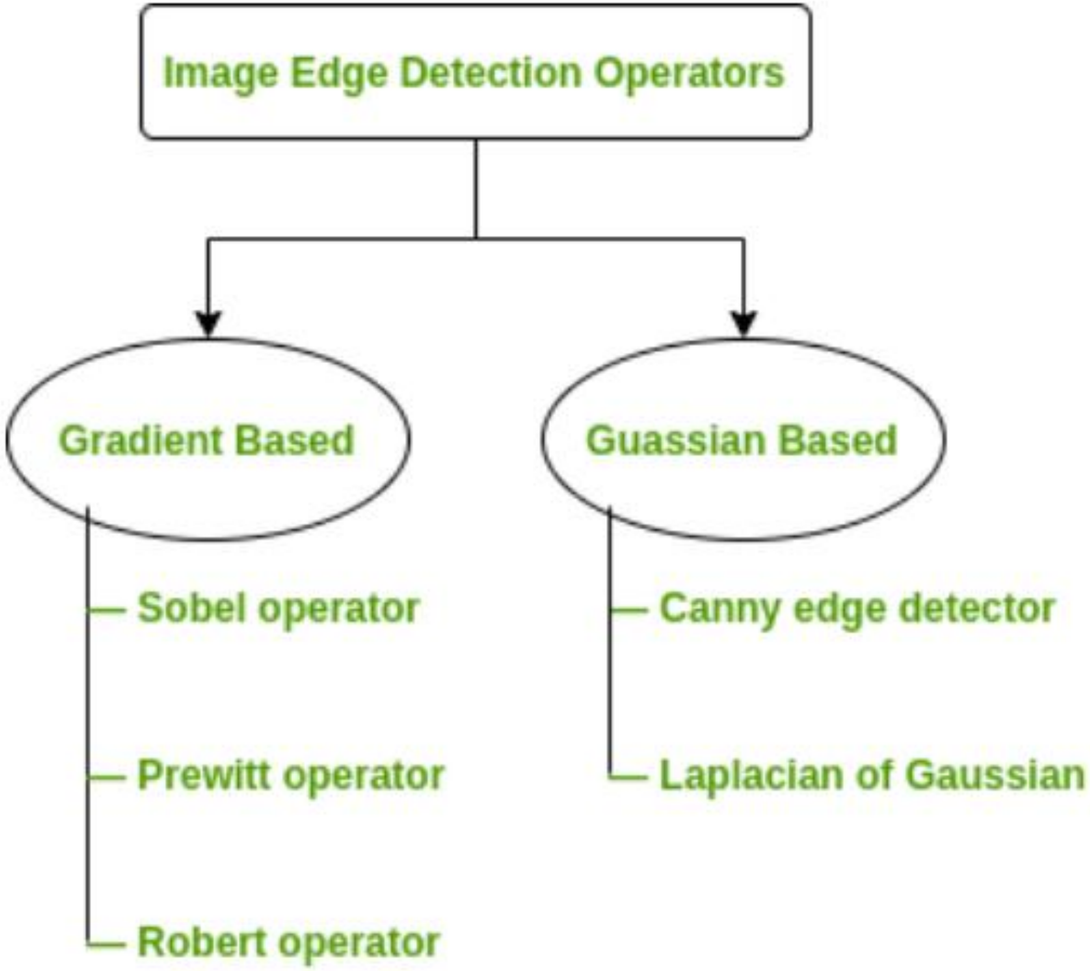
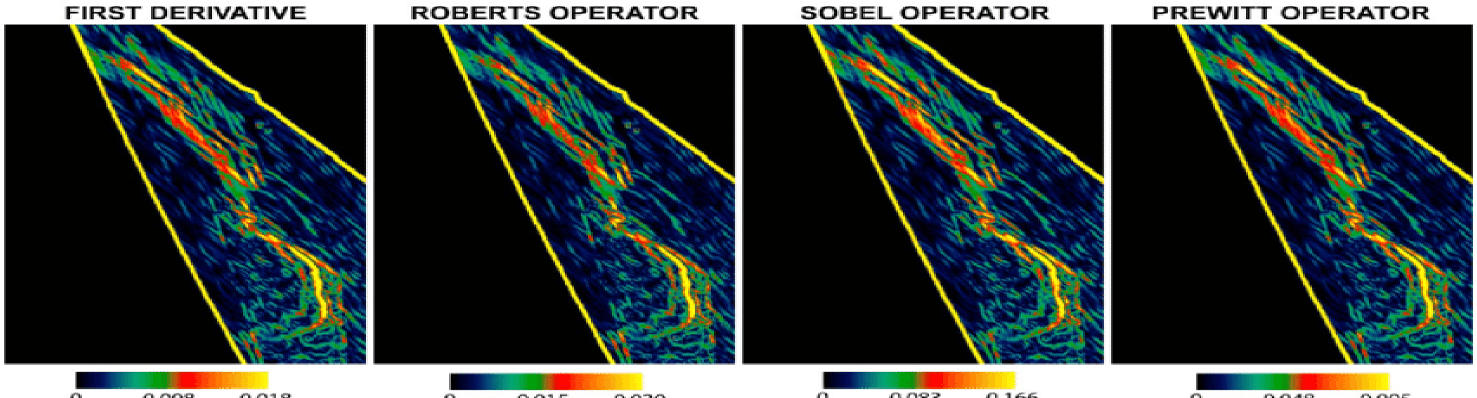


Image Edge Detection Operators
Gradient Based
Guassian Based
Sobel operator
Prewitt operator
Robert operator
Canny edge detector
Laplacian of Gaussian

Operators	Comparative Study	Parameter	Advantages	Disadvantages
1. Sobel	It produces thin edges. It does not offer detailed information.	Threshold	It is simple. Better approximation to gradient magnitude.	Sensitivity to Noise. Not accurate in locating edges. Accuracy descends when magnitude of the edges decreases.
2. Prewitt	More sensitive to horizontal and vertical edges.	Threshold	Detection of edges and their orientations are high.	Inaccurate. Size of the coefficient and kernel filter is fixed and cannot be changed to a given image.
3. Roberts	Works with binary images only. Does not detect edges when a minimal change in gray scale value.	Threshold	It is simple and fast. Detects thicker edges.	Localization is not good.Weak response to genuine edge.



Local Processing Approach

$(x', y') \in N_{(x,y)}$

(x', y') and (x, y) are similar if

$|\nabla f(x,y) - \nabla f(x',y')| < T$ Strength of gradient at location (x,y) and at location (x',y') must be close

$|\alpha(x,y) - \alpha(x',y')| < A$ Angle of gradient at location (x,y) and at location (x',y') must be close

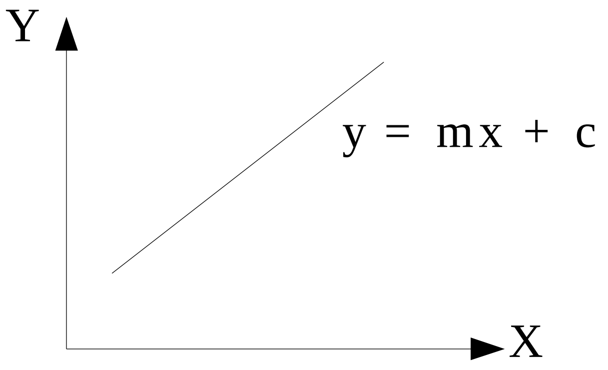
T is gradient strength threshold and A is gradient angle threshold

Global Processing Approach→ Linking

- Ideally discontinuity detection techniques should idetify pixels lying boundary
- Boundary is a region where transition from low intensity value to high intensity value
- But in practice often these boundary points are not connected due to poor illumination or noise
- Local processing is based on neighborhoods, but it is a very small region
- This may not link the pixels which are far away than neighborhood

Edge linking

- Hough Transform: <u>Mapping of a spatial domain into parameter domain</u>



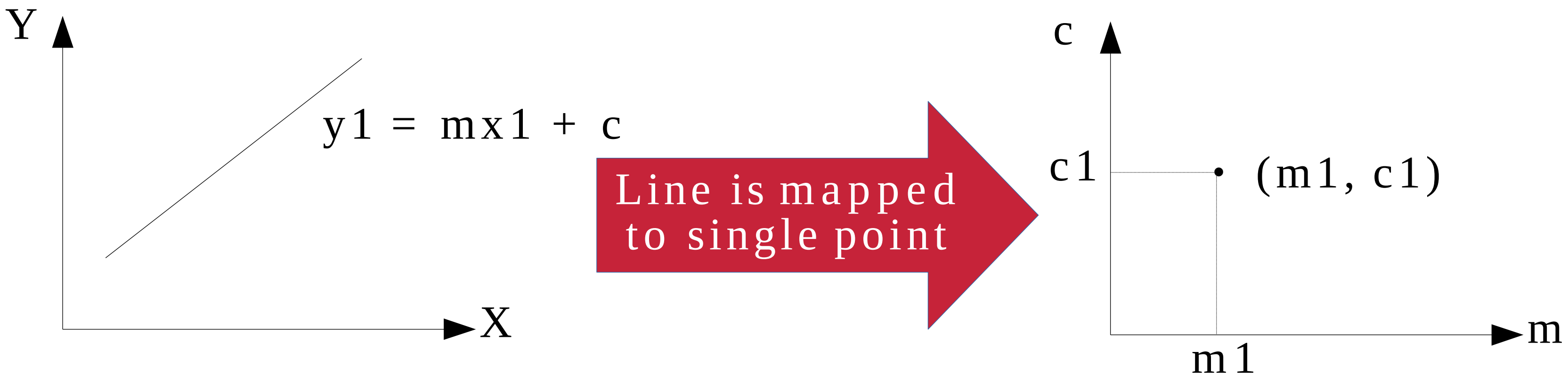
Slope intercept form

For a particular straight line slope and Intercept is constant y = m1x + c1

m1 is slope of the line and
c1 is the intercept of the line

Edge linking

- **Hough Transform:** Mapping of a spatial domain into parameter domain

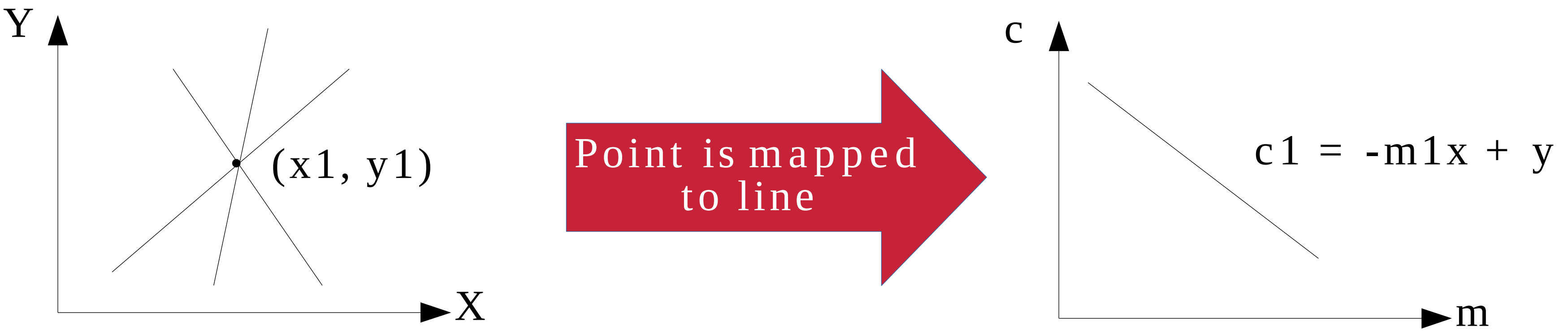


Slope intercept form

Parameter space

Edge linking

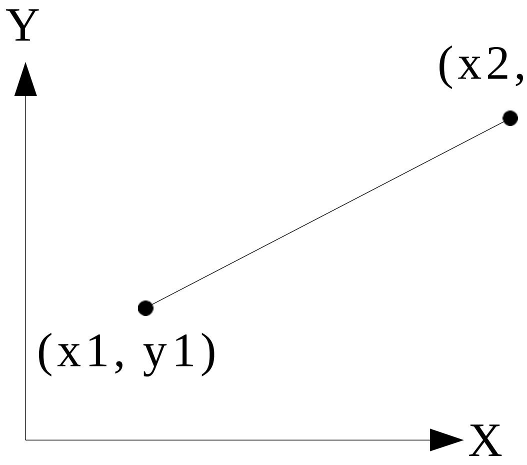
- Hough Transform: Mapping of a spatial domain into parameter domain



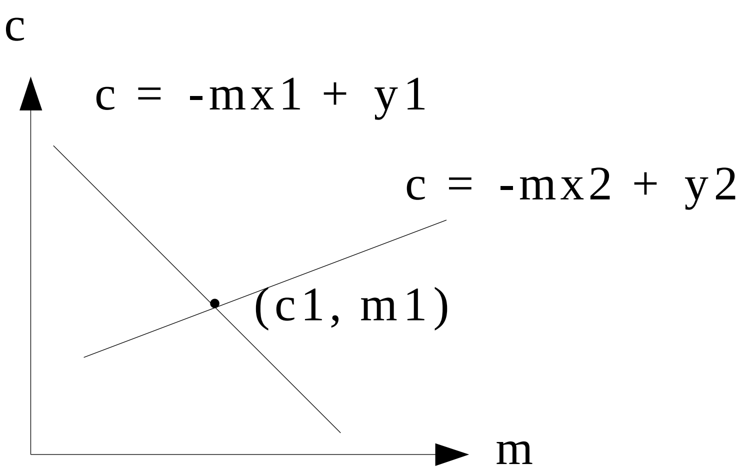
Slope intercept form

Parameter space

Edge linking

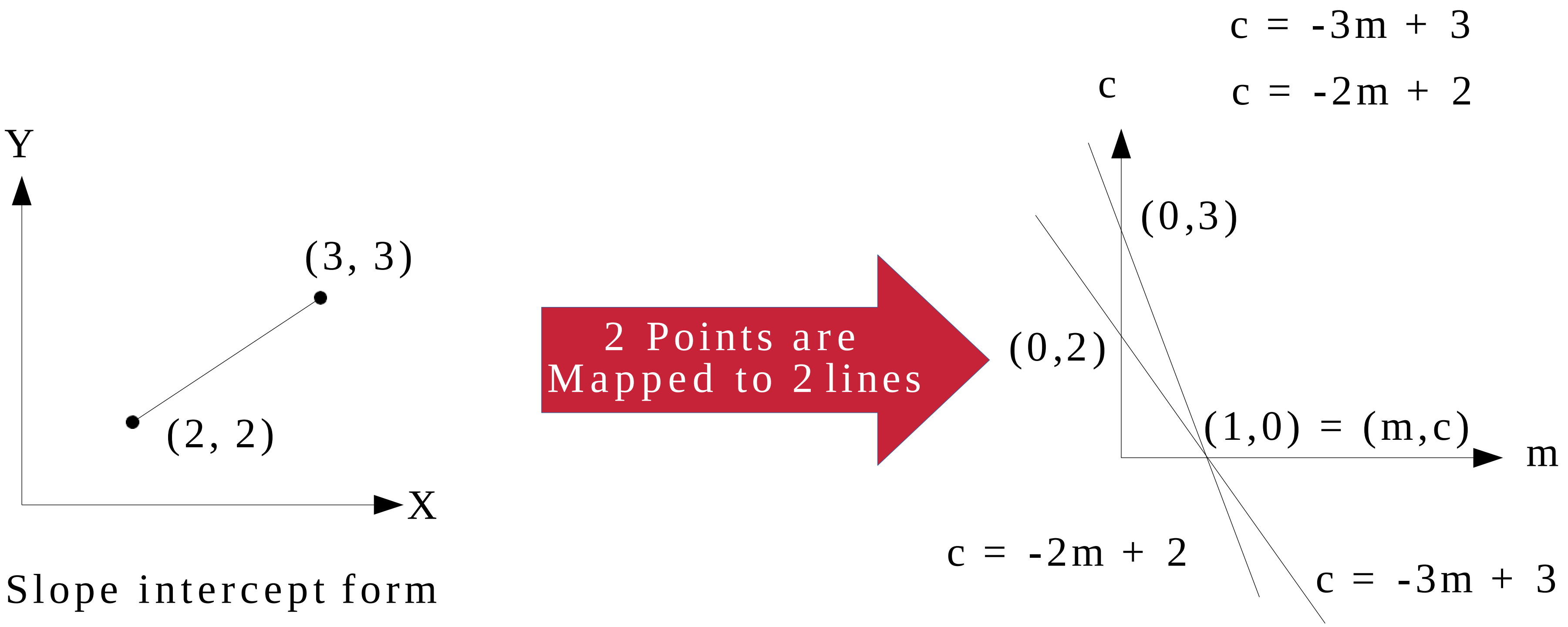


Slope intercept form



Parameter space

Edge linking



Edge linking

- Using Hough transform we find whether points are colinear or not.
- Problem:

  Check whether the points (1,1), (2,2), (3,3) are colinear or not

Edge linking

- Problem:
- Check whether the points (1,1), (2,2), (3,3) are colinear ?

Edge linking Hough Transform Example

- Step 1 Convert points from (x,y) plane to (m,c) plane

  Equation of line is $y = mx + c$

  given points (1,1), (2,2), (3,3)

  1. $(x,y) = (1,1) \Rightarrow 1 = 1m + c \Rightarrow c = -m + 1$

  if $m = 0$ then $c = 1$

  if $c = 0$ then $m = 1$ therefore $(m,c) = (1,1)$

Edge linking Hough Transform Example

- Step 1 Convert points from (x,y) plane to (m,c) plane

  Equation of line is $y = mx + c$

  given points (1,1), (2,2), (3,3)

  2. $(x,y) = (2,2) \Rightarrow 2 = 2m + c \Rightarrow c = -2m + 2$

  if $m = 0$ then $c = 2$

  if $c = 0$ then $m = 1$ therefore $(m,c) = (1,2)$

Edge linking Hough Transform Example

- Step 1 Convert points from (x,y) plane to (m,c) plane

  Equation of line is $y = mx + c$

  given points (1,1), (2,2), (3,3)

  3. $(x,y) = (3,3) \Rightarrow 3 = 3m + c \Rightarrow c = -3m + 3$

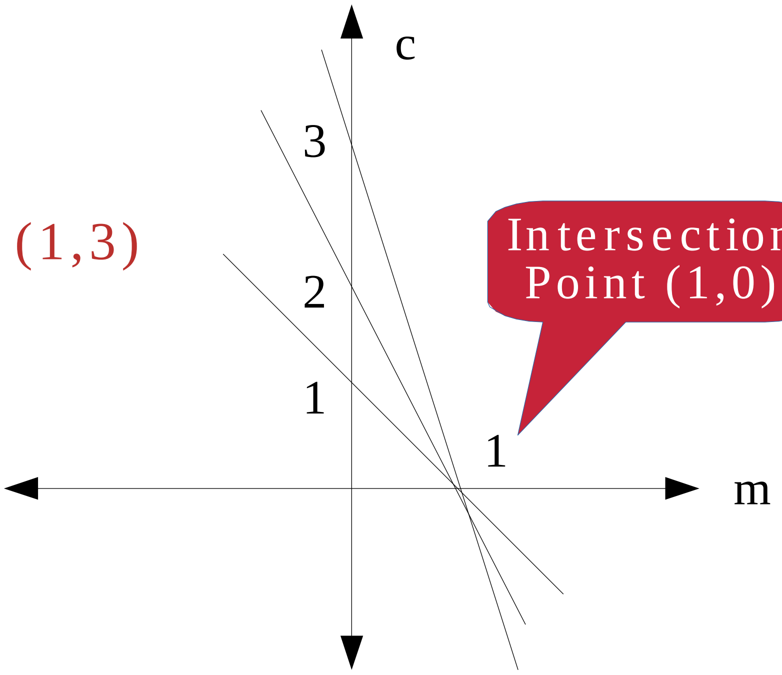
  if $m = 0$ then $c = 3$

  if $c = 0$ then $m = 1$ therefore $(m,c) = (1,3)$

Edge linking Hough Transform Example

- **Step 2:** Using (m,c) points draw lines in m-c plane

Points are
(1,1), (1,2), (1,3)



All lines intersect at
(m,c) = (1,0)

Edge linking Hough Transform Example

- Step 3: Put the value of $(m,c) = (1,0)$ in the original equation of line

  $y = mx + c \Rightarrow y = x$

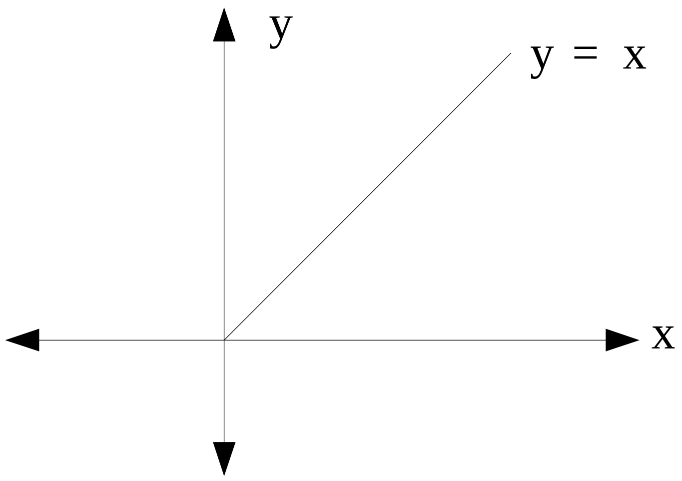
- Step 4: Check the original points (from x-y plane) satisfy the equation $y = x$

  $(1,1)$, $(2,2)$, $(3,3)$ points satisfy the equation hence these points are colinear

Edge linking Hough Transform Example

Step 4: Check the original points (from x-y plane) satisfy the equation $y = x$

(1,1), (2,2), (3,3) points satisfy the equation hence these points are colinear



Hough Transform

- Advantages:

1)Conceptually simple technique.

2)Handles missing occluded data gracefully.

3) Can be adapted for many other forms.

- Disadvantages:

1)Large storage space required.

2)Checks for only one type of object.

Although it is the commonly preferred method for lines & circle detection, the HT in general has several limitations making it challenging to detect anything other than lines and circles. This is especially the case when more parameters are needed to describe shapes, this add more complexity.

Techniques	Pros	Cons
Classical (Sobel, Prewitt, Robert)	Simpler to implement, while detection of edges and their orientation	Highly sensitive to image noise and inaccurate results.
Zero Crossing (Laplacian, Second directional derivative)	Detection of edges and their orientations have fixed characteristics among all the direction	Sensitive to image noise and re-responds to the some of the existing edges.
Laplacian of Gaussian (LOG) Marr Hilderath	Covers wider area around the pixels, meanwhile finds correct places of the edges	High chances of finding false edges and localization errors on the curve edges.
Gaussian (Canny)	Good edge detection in noisy images, while improving signal to noise ratio (SNR)	Complexity in computation, false zero crossing and time consuming

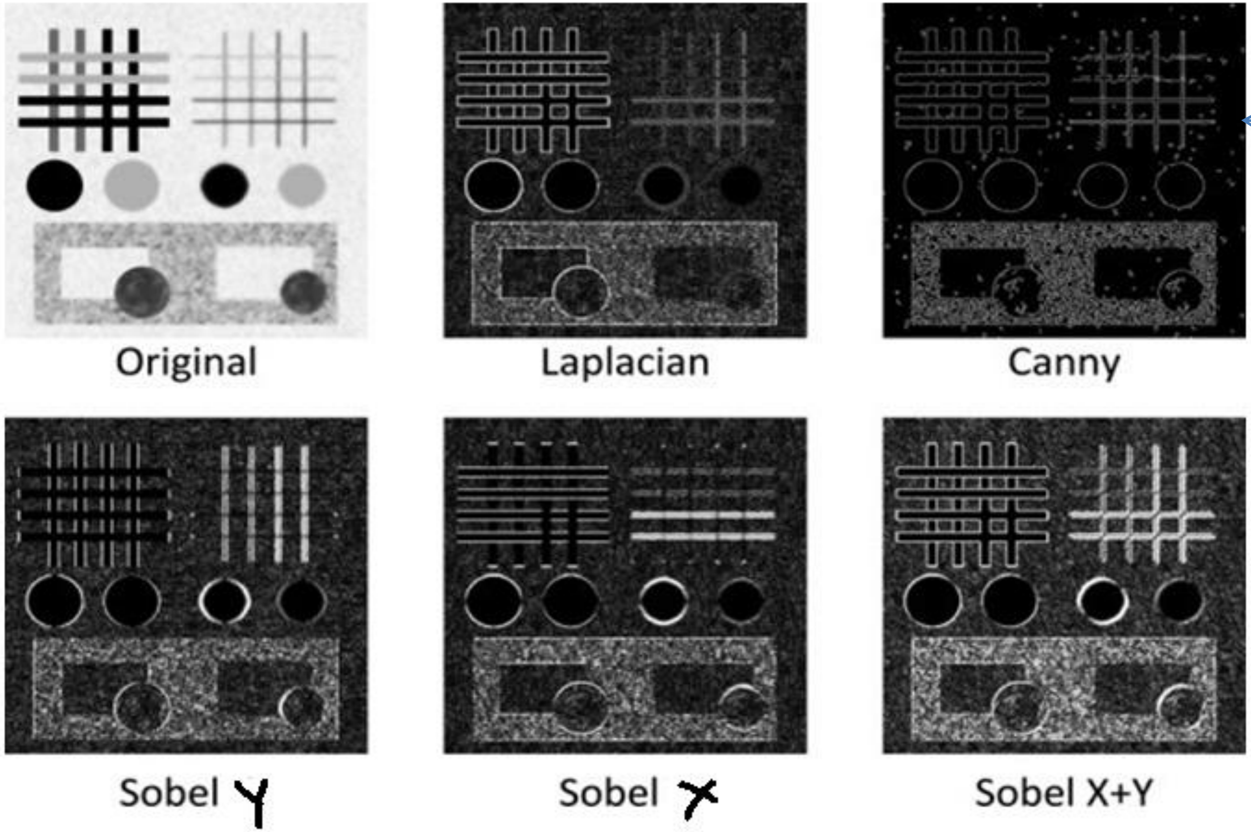
Canny edge detection algorithm (Canny, 1986) known as optimal edge detection algorithm and the most commonly used edge detection algorithm in practice.

Sobel operator finds more edges or make edges more visible as compared to Prewitt Operator. This is because in **Sobel operator allots more weight to the pixel intensities around the edges**.

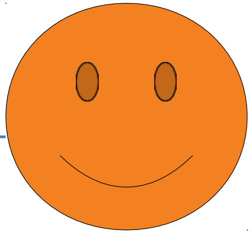
Prewitt operator is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images. However, unlike the Sobel, **this operator does not place any emphasis on the pixels that are near to the center of the mask**.

Canny Edge detection

Thick edges and noise



Optimal



Thick edges

Canny Edge Detection Algorithm

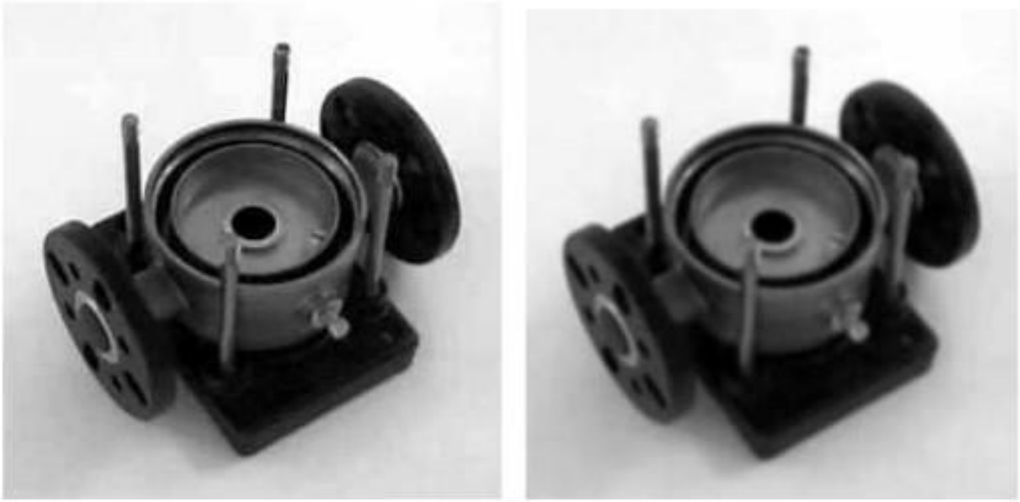
- Smoothing: Blurring of the image to remove noise.
- Finding gradients: The edges should be marked where the gradients of the image has large magnitudes.
- Non-maximum suppression: Only local maxima should be marked as edges.
- Double thresholding: Potential edges are determined by thresholding.
- Edge tracking by hysteresis: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

Canny Edge Detection Algorithm

1. Smoothing: Blurring of the image to remove noise.

- Eg. Apply Gaussian filter to smoothen the image

$$B = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$



(a) Original (b) Smoothed

Canny Edge Detection Algorithm

2. Finding gradients: To find the edges use sobel operators in X and Y direction G_x and G_y

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

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

Calculate strength of the gradient G

$$|G| = \left[G_x^2 + G_y^2\right]^{1/2}$$

// The horizontal gradient is **formed by taking the differences between column values.**//

Canny Edge Detection Algorithm

2. Finding gradients: Output showing Gradient magnitudes in image i.e edges in image

(a) Smoothed (b) Gradient magnitudes

Thick edges

Edges are very thick and broad
Hene exact location is unknown!!

Canny Edge Detection Algorithm

2. Finding gradients angle: To find the exact location of the edge angle of gradient isimportant

Calculate angle of the gradient G using

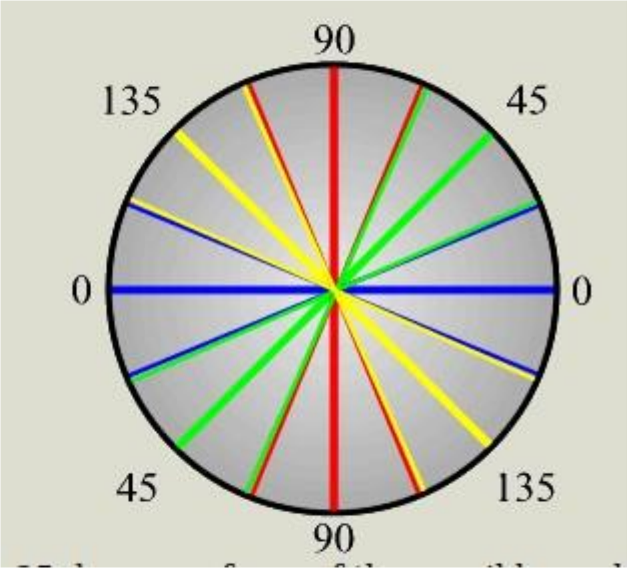
$$\theta = \tan^{-1}\frac{|G_y|}{|G_x|}$$

This angle information is used in non maximal suppression

Canny Edge Detection Algorithm

3. Non-maximum suppression: The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges

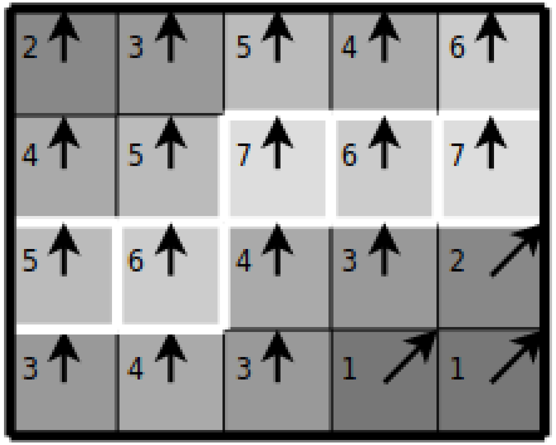
- Preserve local maxima in gradient image and remove everything else.
- Approximate angle of gradient at every pixel by its nearest 45 degree multiple

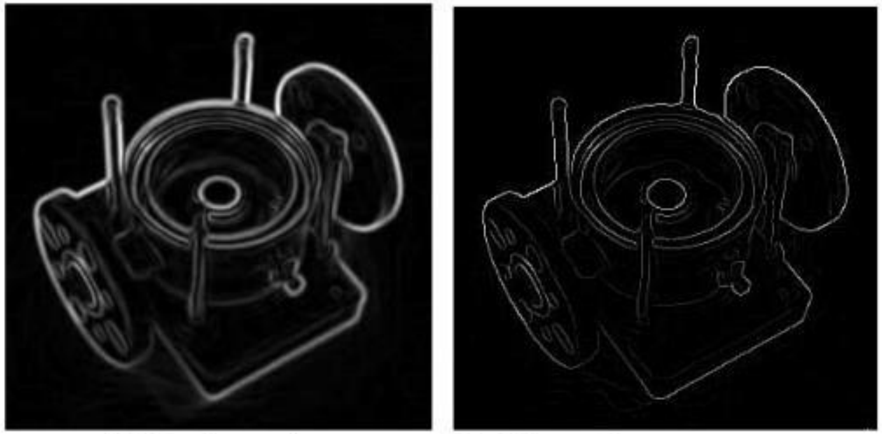


Canny Edge Detection Algorithm

3. Non-maximum suppression: Compare the pixels in +ve and -ve direction of gradient

- If maximum and its magnitude is greater than the upper threshold then mark it as a edge pixel





(a) Gradient values

(b) Edges after non-maximum suppression

Canny Edge Detection Algorithm

4. Double Thresholding: Many of the edges in the image after non maximal suppression are true but some are false due to noise inimage

- Edge pixels stronger than the high threshold are marked as strong edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

Canny Edge Detection Algorithm

5. Edge tracking by hysteresis: Weak edges are kept only if they are connected to strong edges

(a) Double thresholding (b) Edge tracking by hysteresis (c) Final output