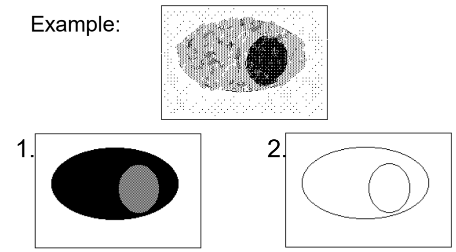
Computer Vision
and
Image Understanding
(Segmentation: Edge detection)

Bhabatosh Chanda
bchanda57@gmail.com

Segmentation

Example:



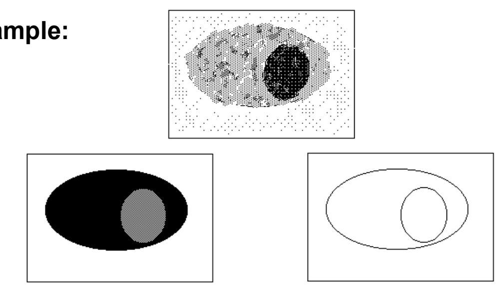
1. Region extraction (based on some measure of homogeneity).
2. Edge detection (based on abrupt change in some feature).

Outline

- Edge detection
  - Gradient magnitude and direction
- First and second derivative
  - 4-neighbour, Prewitt, Sobel operators
  - Convolution with Gaussian
  - Marr and Hildreth operator
  - Canny’s edge detector
- Hough transform

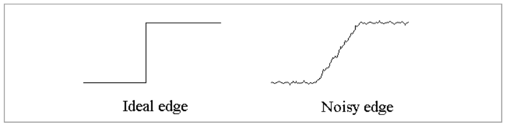
Segmentation

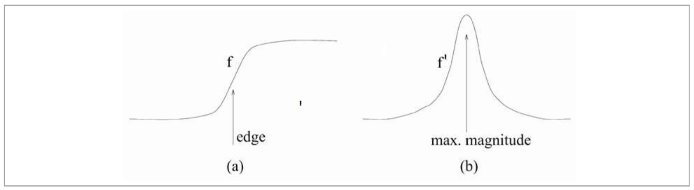
Example:



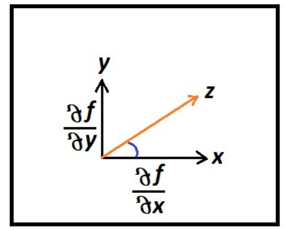
1. Region extraction (based on some measure of homogeneity)
2. Edge detection (based on detection of some abrupt change)

Edge in image





First derivative in 2D



- First derivative along any arbitrary direction z making an angle θ is
$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial x} cos\theta + \frac{\partial f}{\partial y} sin\theta$$

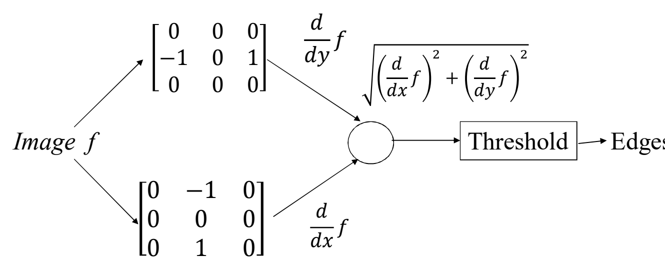
where

- Magnitude = $\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$
- Angle $\theta = tan^{-1}\left[\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right]$

First derivative in 2D

Detecting edge by first derivative

- 4-neighbour Edge Detector



Detecting edge by first derivative

- Prewitt Edge Detector

$$\text{Image } f \rightarrow \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \frac{d}{dy}f$$

$$\text{Image } f \rightarrow \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow \frac{d}{dx}f$$

$$\sqrt{\left(\frac{d}{dx}f\right)^2 + \left(\frac{d}{dy}f\right)^2} \rightarrow \boxed{\text{Threshold}} \rightarrow \text{Edges}$$

Detecting edge by first derivative

- Sobel Edge Detector

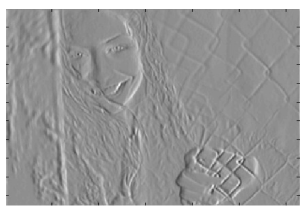
$$\text{Image } f \rightarrow \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \frac{d}{dy}f$$

$$\text{Image } f \rightarrow \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \rightarrow \frac{d}{dx}f$$

$$\sqrt{\left(\frac{d}{dx}f\right)^2 + \left(\frac{d}{dy}f\right)^2} \rightarrow \boxed{\text{Threshold}} \rightarrow \text{Edges}$$

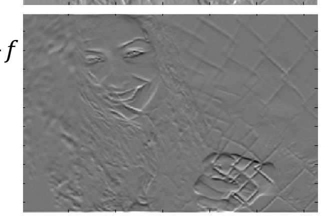
Detecting edge by Sobel operator

$\frac{d}{dy}f$



f

$\frac{d}{dx}f$



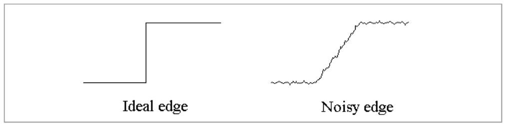
Detecting edge by Sobel operator

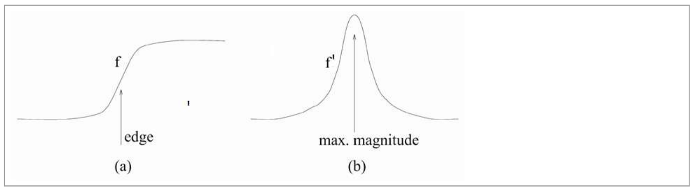
$$\Delta = \sqrt{\left(\frac{d}{dx}f\right)^2 + \left(\frac{d}{dy}f\right)^2}$$

f

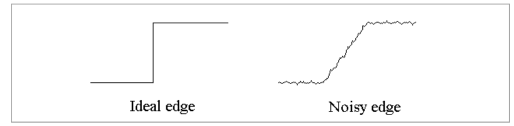
$\Delta \geq Threshold = 100$

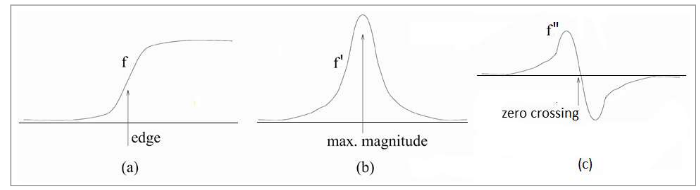
Edge in image





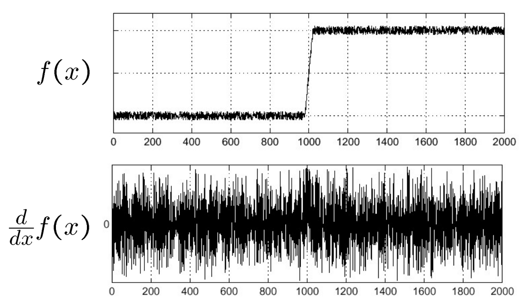
Edge in image





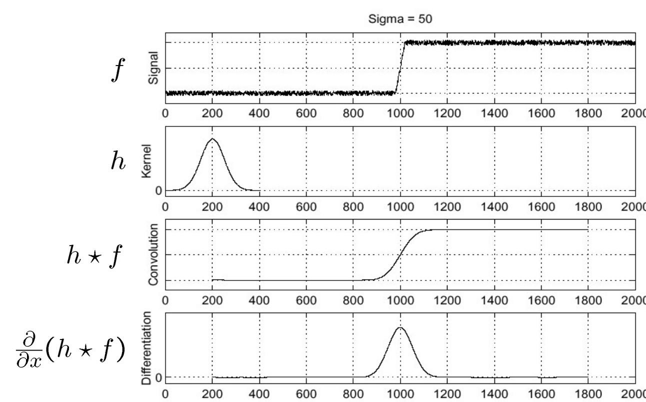
Effects of Noise

- 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

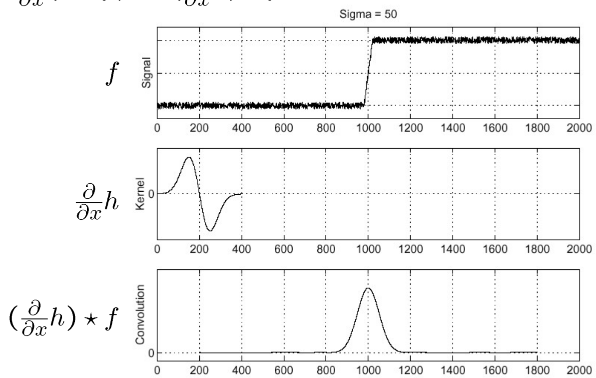


Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(f * h)$

Derivative Theorem of Convolution

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

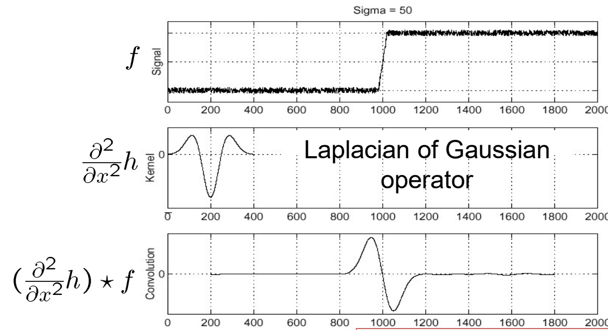
…saves us one operation.



Laplacian of Gaussian (LoG)

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

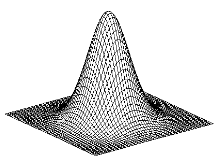
Laplacian of Gaussian



Where is the edge?

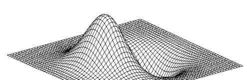
Zero-crossings of bottom graph !

2D Gaussian Edge Operators



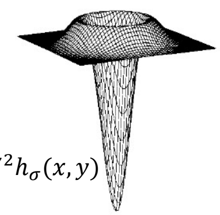
$$h_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian



$$\frac{\vartheta}{\vartheta x} h_\sigma(x,y)$$

Derivative of Gaussian (DoG)



$$\nabla^2 h_\sigma(x,y)$$

Laplacian of Gaussian

Mexican Hat (Sombrero)

- ∇^2 is the **Laplacian** operator:

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

Marr and Hildreth Edge Operator

- Smooth by Gaussian

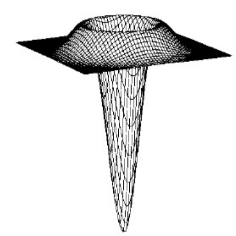
$$S = G_\sigma * f \qquad \text{where } G_\sigma = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Use Laplacian to find derivatives

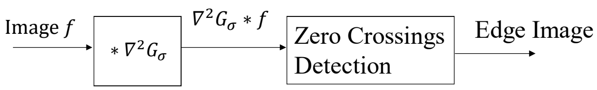
$$\nabla^2 S = \frac{\partial^2}{\partial x^2} S + \frac{\partial^2}{\partial y^2} S$$

$$\nabla^2 S = \nabla^2 (G_\sigma * f) = \nabla^2 G_\sigma * f$$

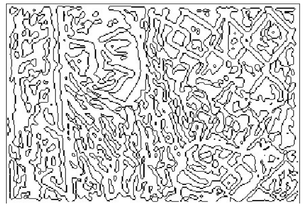
$$\nabla^2 G_\sigma = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2 - \frac{x^2+y^2}{\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Marr and Hildreth Edge Operator

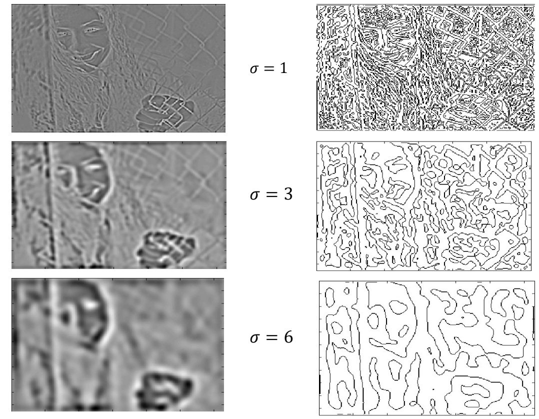


$\nabla^2 G_\sigma * f$

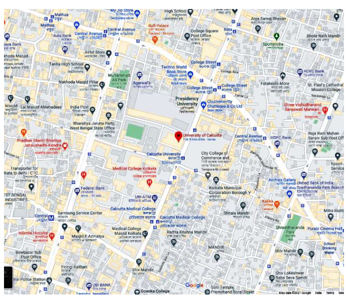


Zero Crossings

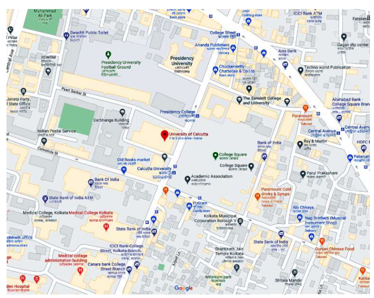
Marr and Hildreth Edge Operator



Idea of scale

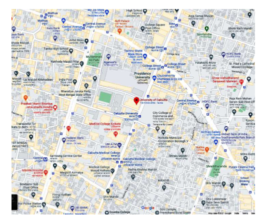


S1

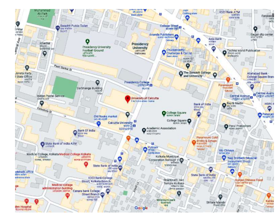


S2

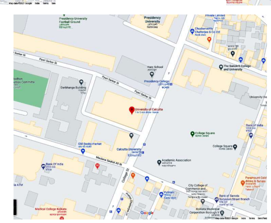
Idea of scale



S1



S2



S3

Canny Edge Detector

- Criterion 1. ***Good Detection:*** The optimal detector must minimize the probability of false positives as well as false negatives.
- Criterion 2. ***Good Localization:*** The edges detected must be as close as possible to the true edges.
- Criterion 3: ***Single Response Constraint:*** The detector must return one point only for each edge point.

Canny Edge Detector: Steps

- Convolution with derivative of Gaussian
- Non-maximum Suppression
- Hysteresis Thresholding

Canny Edge Detector

- Smooth by Gaussian

$$S = G_\sigma * f \qquad G_\sigma = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Compute *x* and *y* derivatives

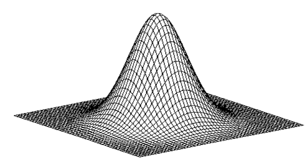
$$\nabla S = \left[\frac{\partial}{\partial x} S \quad \frac{\partial}{\partial y} S\right]^T = [S_x \quad S_y]^T$$

- Compute gradient magnitude and orientation

$$|\nabla S| = \sqrt{S_x^2 + S_y^2} \qquad \theta = \tan^{-1}\frac{S_y}{S_x}$$

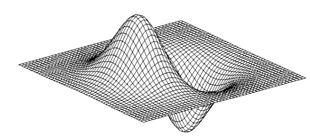
Canny Edge Operator

$$\nabla S = \nabla(G_\sigma * f) = \nabla G_\sigma * f$$



$$\nabla G_\sigma = \left[\frac{\partial G_\sigma}{\partial x} \quad \frac{\partial G_\sigma}{\partial y}\right]^T$$

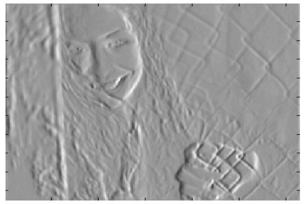
$$\nabla S = \left[\frac{\partial G_\sigma}{\partial x} * f \quad \frac{\partial G_\sigma}{\partial y} * f\right]^T$$



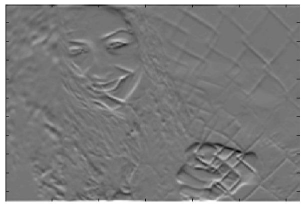
Canny Edge Detector

f

S_x



S_y



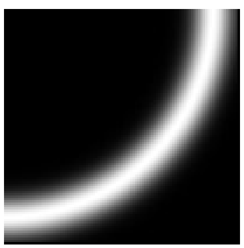
Canny Edge Detector

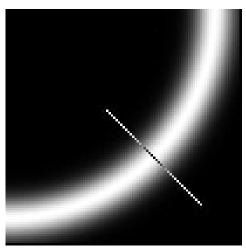
f

$|\nabla S| = \sqrt{S_x^2 + S_y^2}$

$|\nabla S| \geq Threshold = 25$

Non-Maximum Suppression



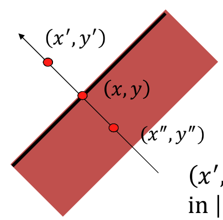


- We wish to mark points along the curve where the magnitude is largest.
- We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression).

Non-Maximum Suppression

- Suppress the pixels in 'Gradient Magnitude Image' which are not local maximum

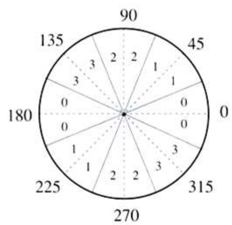
$$M(x,y) = \begin{cases} |\nabla S|(x,y) & \text{if } |\nabla S|(x,y) > |\nabla S|(x',y') \\ & |\nabla S|(x,y) > |\nabla S|(x'',y'') \\ 0 & \text{otherwise} \end{cases}$$

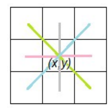


(x',y') and (x'',y'') are the neighbors of (x,y) in $|\nabla S|$ along the direction normal to an edge

Gradient Orientation

- Reduce angle of Gradient *ϑ(x,y)* to one of the 4 sectors





- Check the 3x3 region of each *M(x,y)*
- If the value at the center is not greater than the 2 values along the gradient, then *M(x,y)* is set to 0

Non-Maxima Suppression

0	0	0	0	1	1	1	3
0	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	0
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

- Thin edges by keeping large values of Gradient
  - not always at the location of an edge
  - there are many points on thick edges

Non-Maxima Suppression

0	0	0	0	1	1	1	3
0	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	0
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

local maxima

removed

depends on condition

Non-Maxima Suppression

0	0	0	0	0	0	0	3
0	0	0	0	2	1	3	0
0	0	2	1	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	1	0	1
0	3	0	0	1	0	2	0

false edges

- The suppressed magnitude image will contain many false edges caused by noise or fine texture

Non-Maxima Suppression

- Apply thresholding (>2) on thin ridges in *M(x,y)* that are only one pixel wide.
- Obtain edge pixels on the object contour.

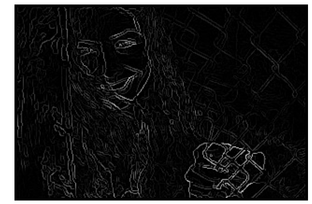
0	0	0	0	1	1	1	3
3	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	3
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

false edges

gaps

Non-Maximum Suppression

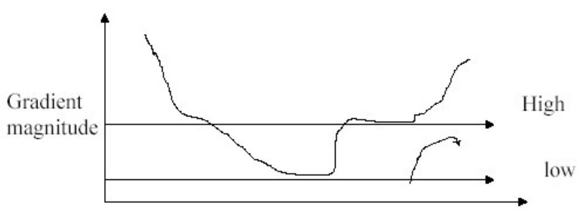
$$|\nabla S| = \sqrt{S_x^2 + S_y^2}$$



M

$$M \geq Threshold = 25$$

Hysteresis Thresholding



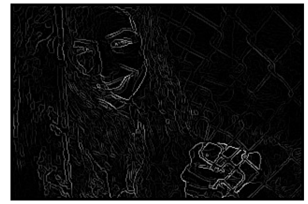
Hysteresis Thresholding

- If the gradient at a pixel is above '**High**',
  - declare it an 'edge pixel'
- If the gradient at a pixel is below '**Low**',
  - declare it a '**non-edge-pixel**'
- If the gradient at a pixel is between '**Low**' and '**High**' then
  - declare it an 'edge pixel' if and only if it is connected to an 'edge pixel'
- Iterate the third step until no change takes place.

Double Thresholding

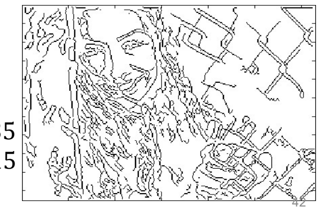
- Apply two thresholds in the suppressed image
  - Set $T_2 > T_1$
  - two images in the output
  - the image from T_2 contains fewer edges but has gaps in the contours
  - the image from T_1 has many false edges
  - Then combine the results from T_1 and T_2
  - link the edges of T_2 into contours until we reach a gap
  - link the edge from T_2 with edge pixels from a T_1 contour until a T_2 edge is found again

Hysteresis Thresholding



$|\nabla S| = \sqrt{S_x^2 + S_y^2}$

$|\nabla S| \geq Threshold = 25$



$High = 35$
$Low \ = 15$

The Canny Edge Detector

original image (Lena)

The Canny Edge Detector

magnitude of the gradient

The Canny Edge Detector

σ=1 σ=3

Canny Edge Operator

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

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

LINE DETECTION

Line detection kernels

-1	2	-1
-1	2	-1
-1	2	-1

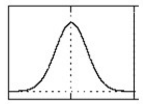
2	-1	-1
-1	2	-1
-1	-1	2

-1	-1	-1
2	2	2
-1	-1	-1

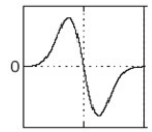
-1	-1	2
-1	2	-1
2	-1	-1

Similarity with 2nd order Gaussian

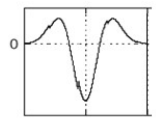
- $G(x) = \frac{1}{K}\exp(-\frac{x^2}{2\sigma^2})$



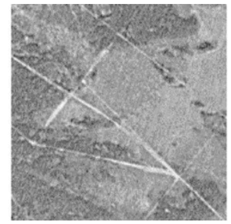
- $G'(x) = \frac{dG}{dx}$

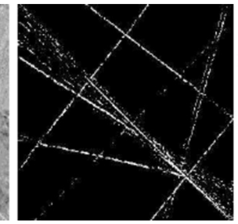


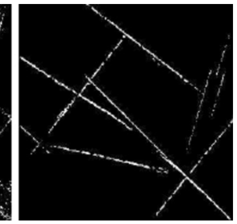
- $G''(x) = \frac{d^2G}{dx^2}$



Line detection results


Original image


Simple kernel


Gaussian Kernel

HOUGH TRANSFORM

Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

$y_i = mx_i + c \quad or \quad c = -x_i m + y_i$

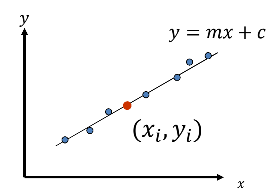
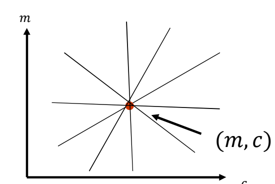


Image Space

Parameter space also called Hough Space



Parameter Space

Line Detection by Hough Transform

Algorithm:

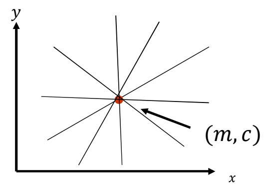
- Quantize Parameter Space (m, c)
- Create Accumulator Array $A(m, c)$
- Set $A(m, c) = 0 \quad \forall m, c$
- For each image edge (x_i, y_i) increment:

$$A(m, c) = A(m, c) + 1$$

- If (m, c) lies on the line:

$$c = -x_i m + y_i$$

- Find local maxima in $A(m, c)$



Parameter Space

$A(m, c)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Better Parameterization

NOTE: $-\infty \leq m \leq \infty$

Large Accumulator

More memory and computations

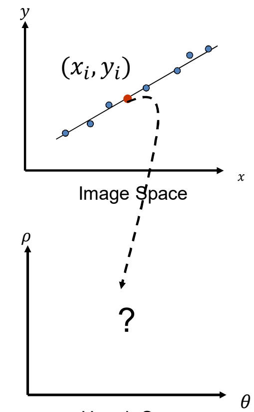
Improvement: (Finite Accumulator Array Size)

Line equation: $\rho = -x\cos\theta + y\sin\theta$

Here $0 \leq \theta \leq 2\pi$
$0 \leq \rho \leq \rho_{\max}$

Given points (x_i, y_i) find (ρ, θ)

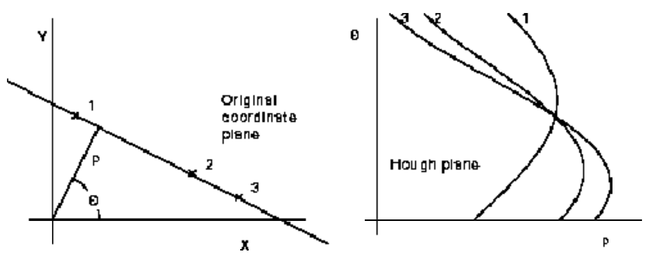
Hough Space Sinusoid



Hough Transform for Straight Line Detection

- A more useful representation in this case is

$$x\sin\theta + y\cos\theta = \rho$$



Hough Transform for Straight Lines

- Advantages of Parameterization
  - Values of ‘ρ’ and ‘θ’ become bounded
- How to find intersection of the parametric curves
  - Use of accumulator arrays – concept of ‘Voting’
  - To reduce the computational load use Gradient information

Computational Load

- Image size = 512 X 512
- Maximum value of $\rho = 512 * 2\sqrt{2}$
- With a resolution of 1°, maximum value of $\theta = 360°$
- Accumulator size = $512 * 2\sqrt{2} * 360$
- Use of direction of gradient reduces the computational load by 1/360

Hough Transform for Straight Lines - Algorithm

- Quantize the Hough Transform space: identify the maximum and minimum values of ρ and θ
- Generate an accumulator array $A(\rho, \theta)$; set all values to zero
- For all edge points (x_i, y_i) in the image
  - Use gradient direction for θ
  - Compute ρ from the equation
  - Increment $A(\rho, \theta)$ by one
- For all cells in $A(\rho, \theta)$
  - Search for the maximum value of $A(\rho, \theta)$
  - Calculate the equation of the line
- To reduce the effect of noise more than one element (elements in a neighborhood) in the accumulator array are increased

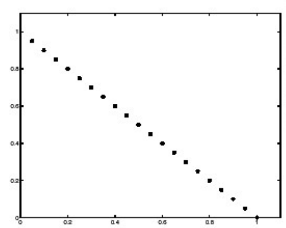
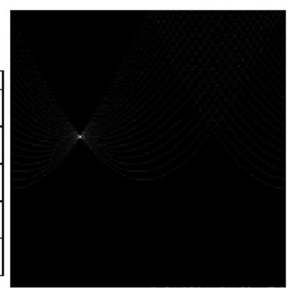


Image space



Votes

Horizontal axis is θ, vertical is rho.

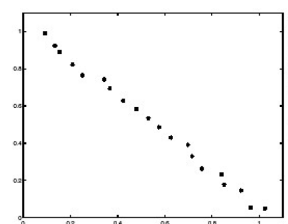
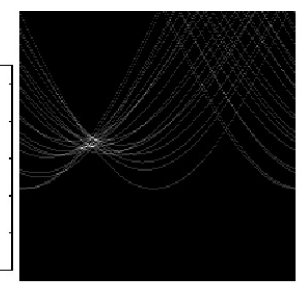
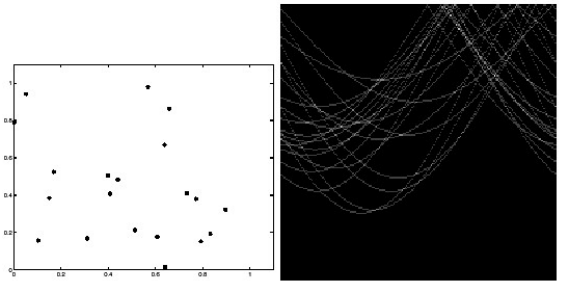


Image space



votes



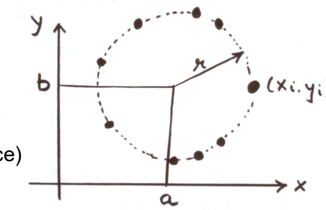
Finding Circles by Hough Transform

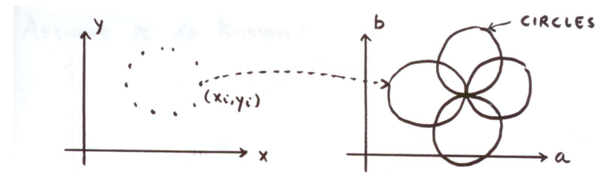
Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

If radius is known: (2D Hough Space)

Accumulator Array $A(a, b)$

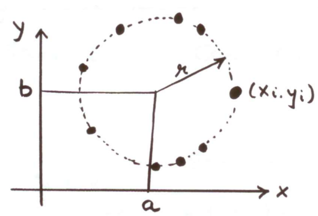




Finding Circles by Hough Transform

Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$



If radius is not known: 3D Hough Space!
Use Accumulator array $A(a, b, r)$

What is the surface in the hough space?

Thank you !

Any question?