Segmentation

Detection Of Discontinuities

- Are based on detecting sharp changes in intensity in neighbouring pixels
- Discontinuities are
  - Edges

    Intensity changes abruptly from current pixel to next pixels along horizontal/ vertical/diagonal lines

    Connected set of edge pixels belong to edge
  - Lines

    Intensity of background on either side of the edge pixel is either higher or lower than edge pixels
  - Isolated points

    Intensity changes abruptly from current pixel to next pixel

    Line whose length and width is 1 pixel

Detection of isolated points using Laplacian

Laplacian mask, w_k

1	1	1
1	-8	1
1	1	1

Center pixel of filtered image, R(x, y)
If |R(x,y)| >= T then g(x,y) = 255
else g(x,y) = 0

image, g_k

4	4	4
3	10	1
4	1	5

Filtered image

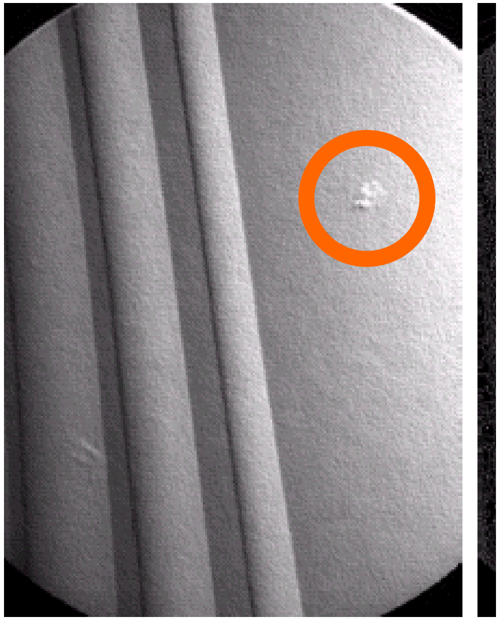
4	4	4
3	-54	1
4	1	5

For T = 15

4	4	4
3	255	1
4	1	5

Isolated point

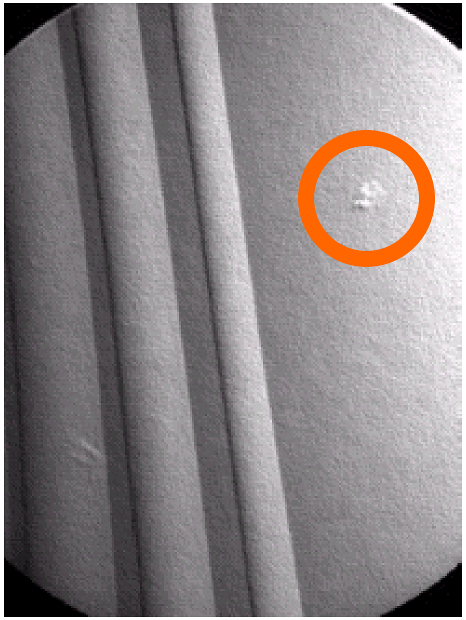
−1	−1	−1
−1	8	−1
−1	−1	−1



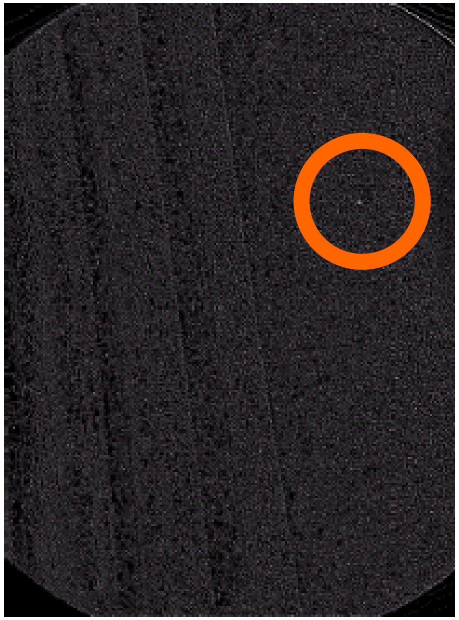
X-ray image of
a turbine blade

Isolated points

−1	−1	−1
−1	8	−1
−1	−1	−1



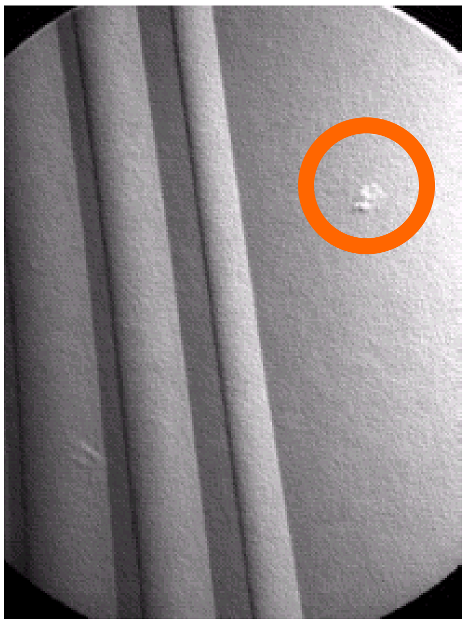
X-ray image of a turbine blade



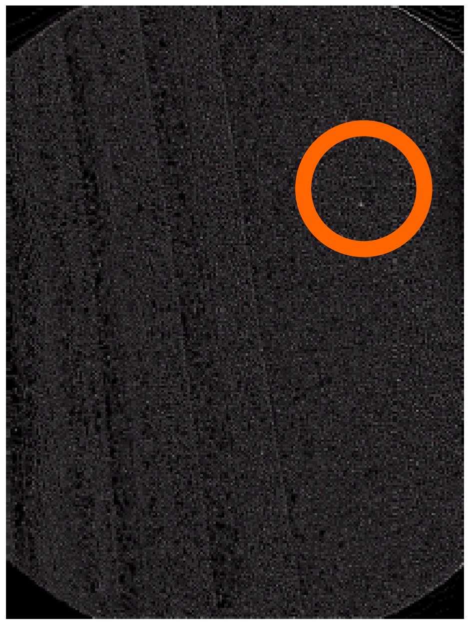
Result of point detection

Isolated points

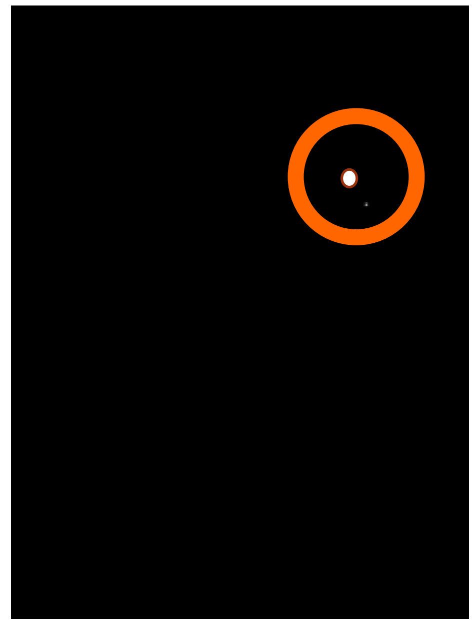
−1	−1	−1
−1	8	−1
−1	−1	−1



X-ray image of a turbine blade



Result of point detection



- Binary image
- Threshold at 90% of the highest intensity

Edge Detection using gradient operators

- The gradient of an image $f(x,y)$ at location (x,y) is defined as the vector:

$$\nabla \mathbf{f} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- The magnitude of this vector: $$\nabla f = \mathrm{mag}(\nabla \mathbf{f}) = \left[g_x^2 + g_y^2\right]^{1/2}$$

- The direction of this vector: $$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

$$\frac{\partial f}{\partial x} = g_x = f(x+1) - f(x) \qquad \frac{\partial f}{\partial y} = g_y = f(y+1) - f(y)$$

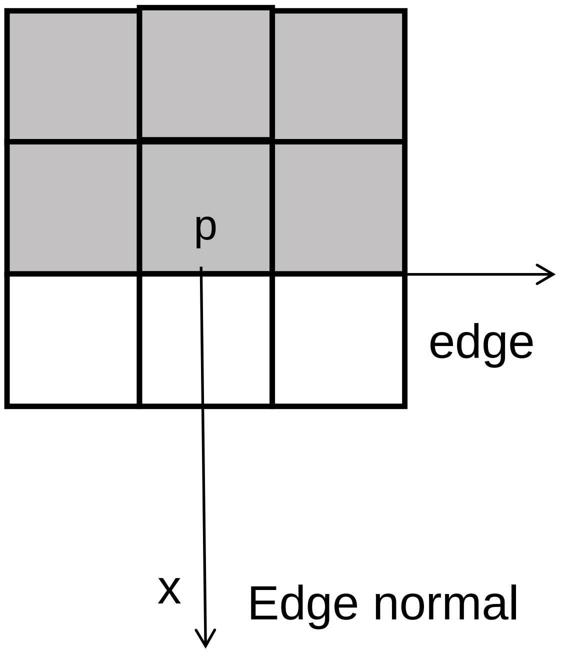
Gradient Vector

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[g_x^2 + g_y^2\right]^{1/2} \qquad \alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

- Points in the direction of the greatest rate of change intensity at location (x,y)
- Magnitude is the value of rate of change in the direction of gradient vector
- Direction is measured with respect to vertical axis
- Direction of gradient vector is also called edge normal

Gradient angle (edge normal)

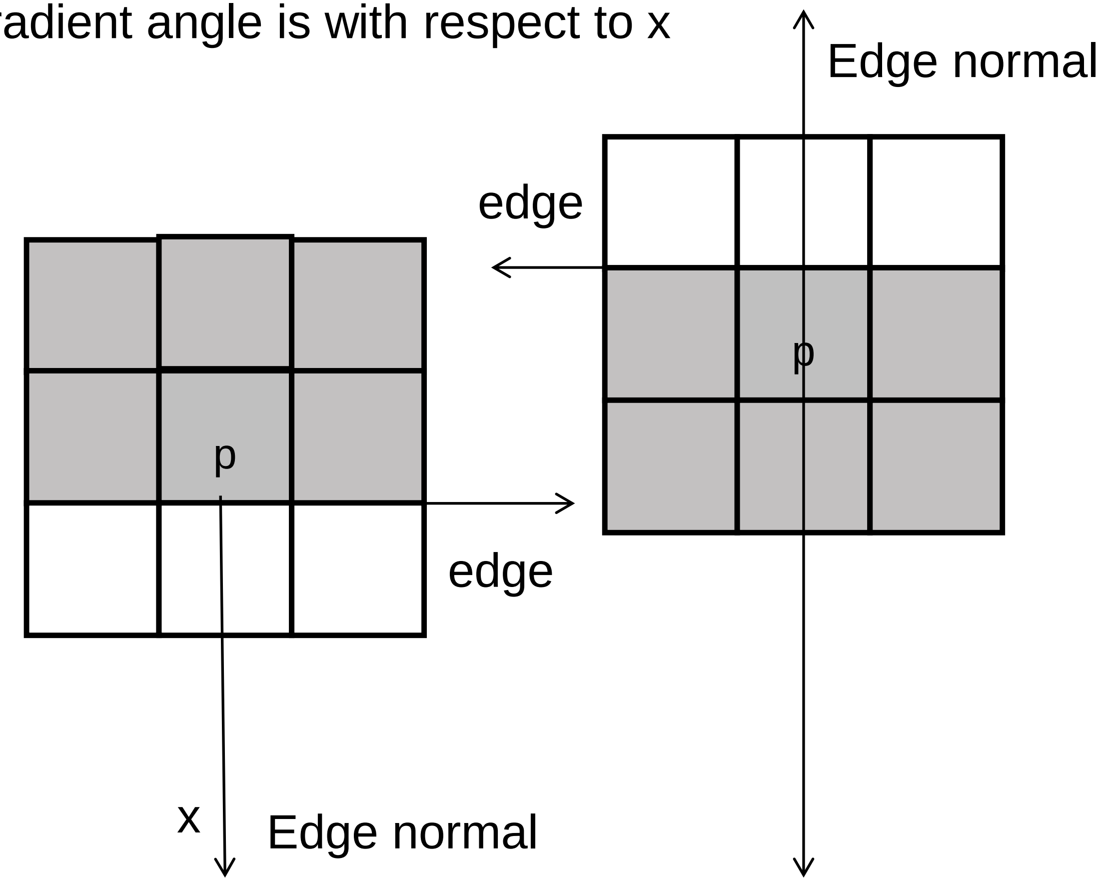
Gradient angle is with respect to x



$$\alpha(x, y) = 0$$

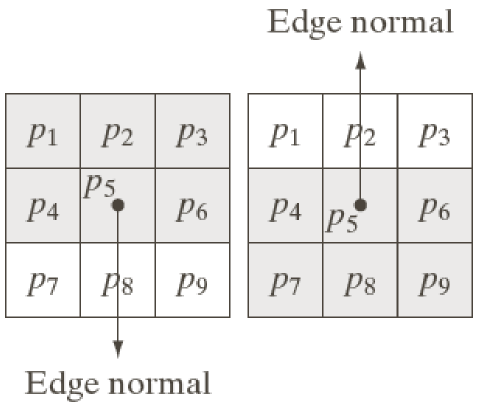
Gradient angle (edge normal)

Gradient angle is with respect to x

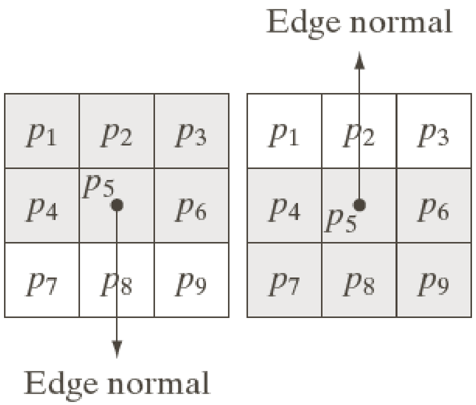


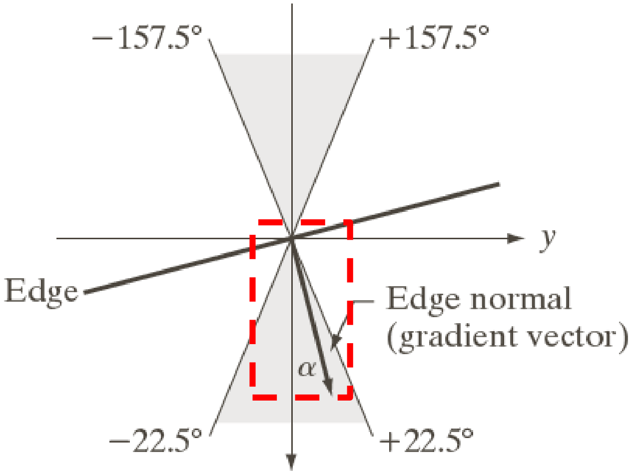
$$\alpha(x, y) = 0 \qquad \alpha(x, y) = \pi$$

Gradient angle



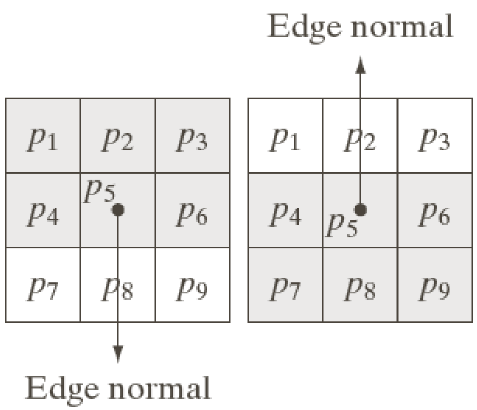
Gradient angle

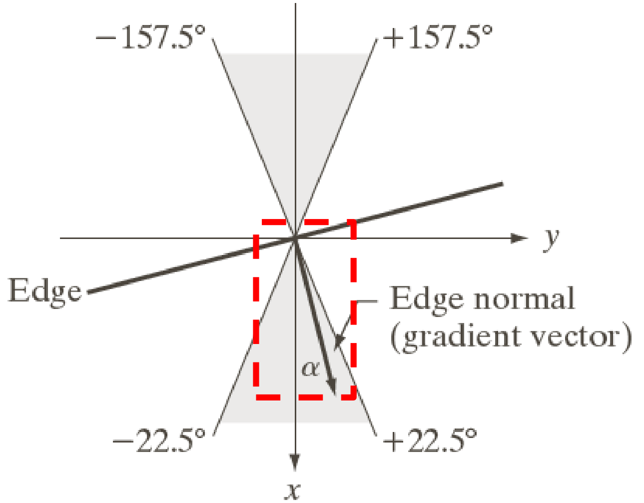


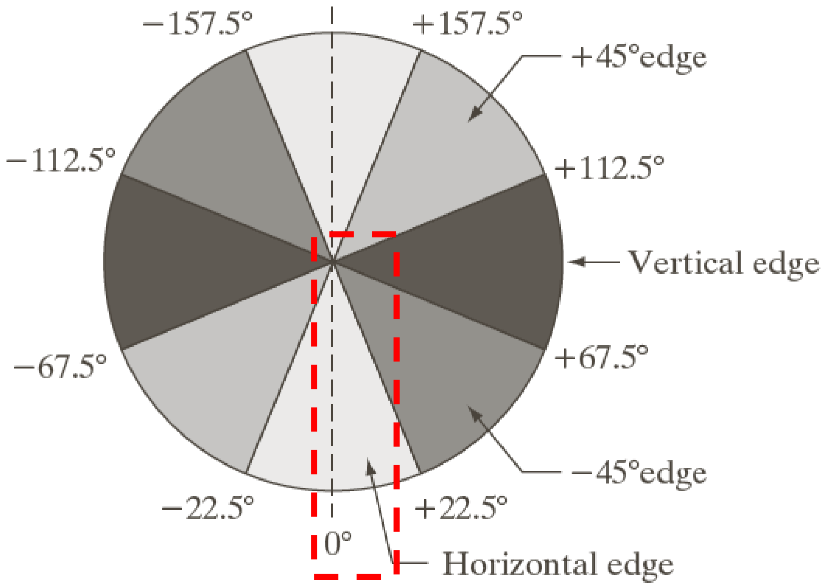


Angle of gradient vector = angle of edge with horizontal axis

Gradient angle

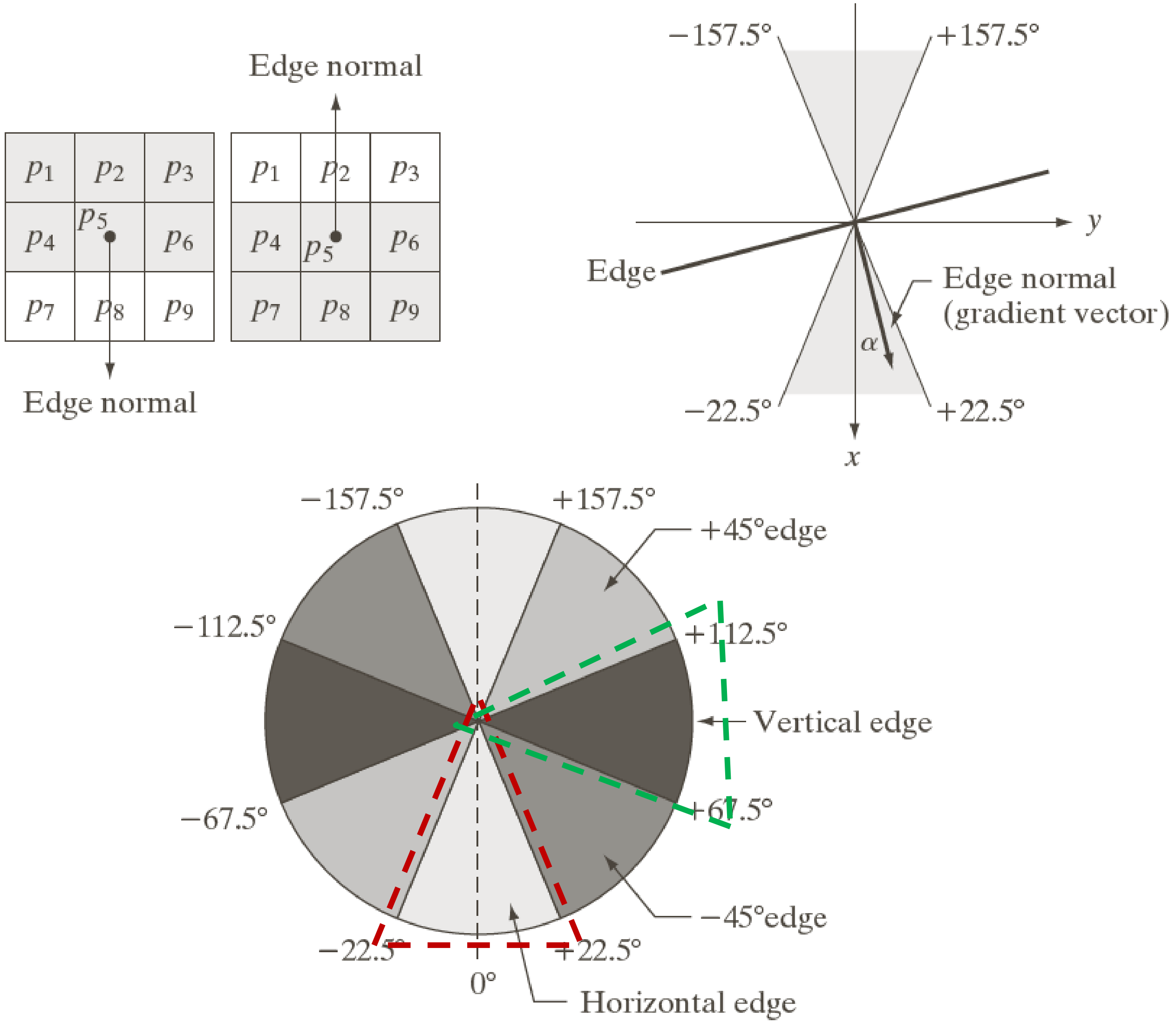






Angle of gradient vector = angle of edge with horizontal axis

Gradient angle



Angle of gradient vector = angle of edge with horizontal axis

Gradient vectors of Image

- Magnitude of gradient vector at pixel can be shown in the form of image
- Magnitude of gradient vector (gradient image) is given by

$$
\begin{aligned}
M(x, y) &= mag(\nabla \mathrm{f}) \\
&= \left[G_x^2 + G_y^2\right]^{1/2} \\
&= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}
\end{aligned}
$$

- Size of *M(x,y)* is same as image
- For practical reasons magnitude can be simplified as

$$M(x, y) \approx |G_x| + |G_y|$$

Common Edge Detectors

- Edge detection filters in spatial domain

−1	−1	−1		−1	0	1
0	0	0		−1	0	1
1	1	1		−1	0	1

Prewitt

−1	−2	−1		−1	0	1
0	0	0		−2	0	2
1	2	1		−1	0	1

Sobel

Sobel Operators (gradient vector)

image

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Mask for g_x

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

Mask for g_y

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

$$g_x = \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right|$$

$$g_y = \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

$$M(x, y) \approx \left|g_x\right| + \left|g_y\right|$$

Sobel Operators for gradient vector

$$g_x = \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right|$$

$$g_y = \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

image

10	10	50	10	10
10	10	50	10	10
10	10	50	10	10
10	10	50	10	10

Vertical mask, gy

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

Horizontal mask, gx

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

g_y

10	10	50	10	10
10	160	0	160	10
10	160	0	160	10
10	10	50	10	10

+

g_x

10	10	50	10	10
10	0	0	0	10
10	0	0	0	10
10	10	50	10	10

=

$g_y + g_x$

10	10	50	10	10
10	160	0	160	10
10	160	0	160	10
10	10	50	10	10

Edge detection

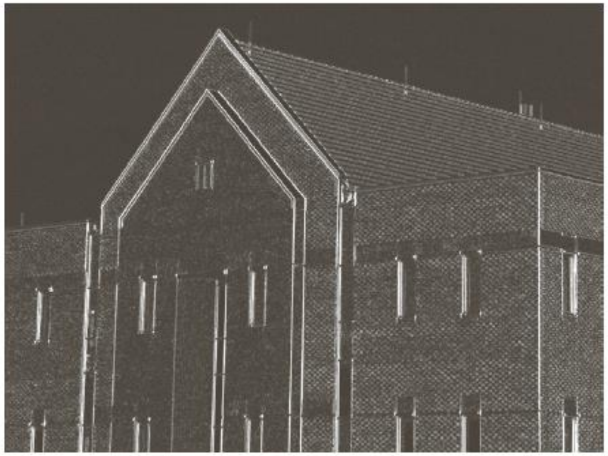
Edge detection

horizontal edges using sobel mask

Edge detection

horizontal edges using sobel mask



vertical edges using sobel mask

Edge detection

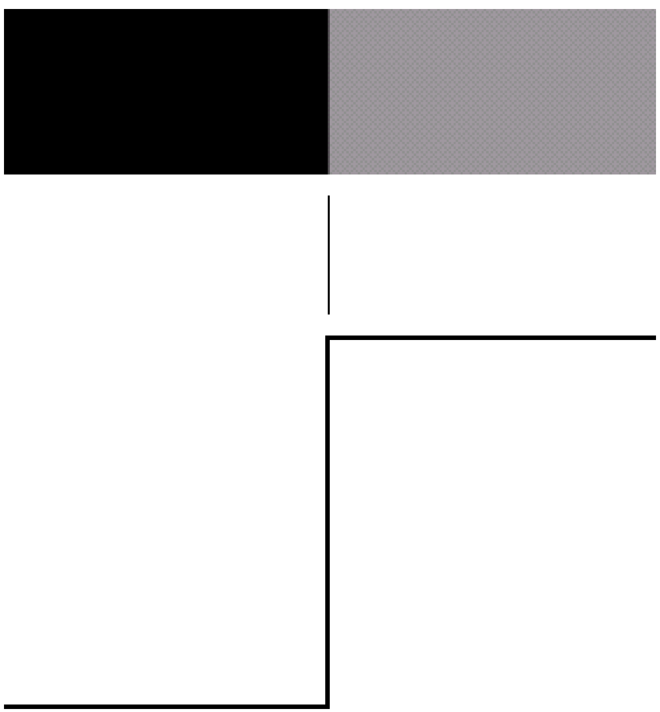
horizontal edges using sobel mask

vertical edges using sobel mask

Addition of vertical and horizontal edges

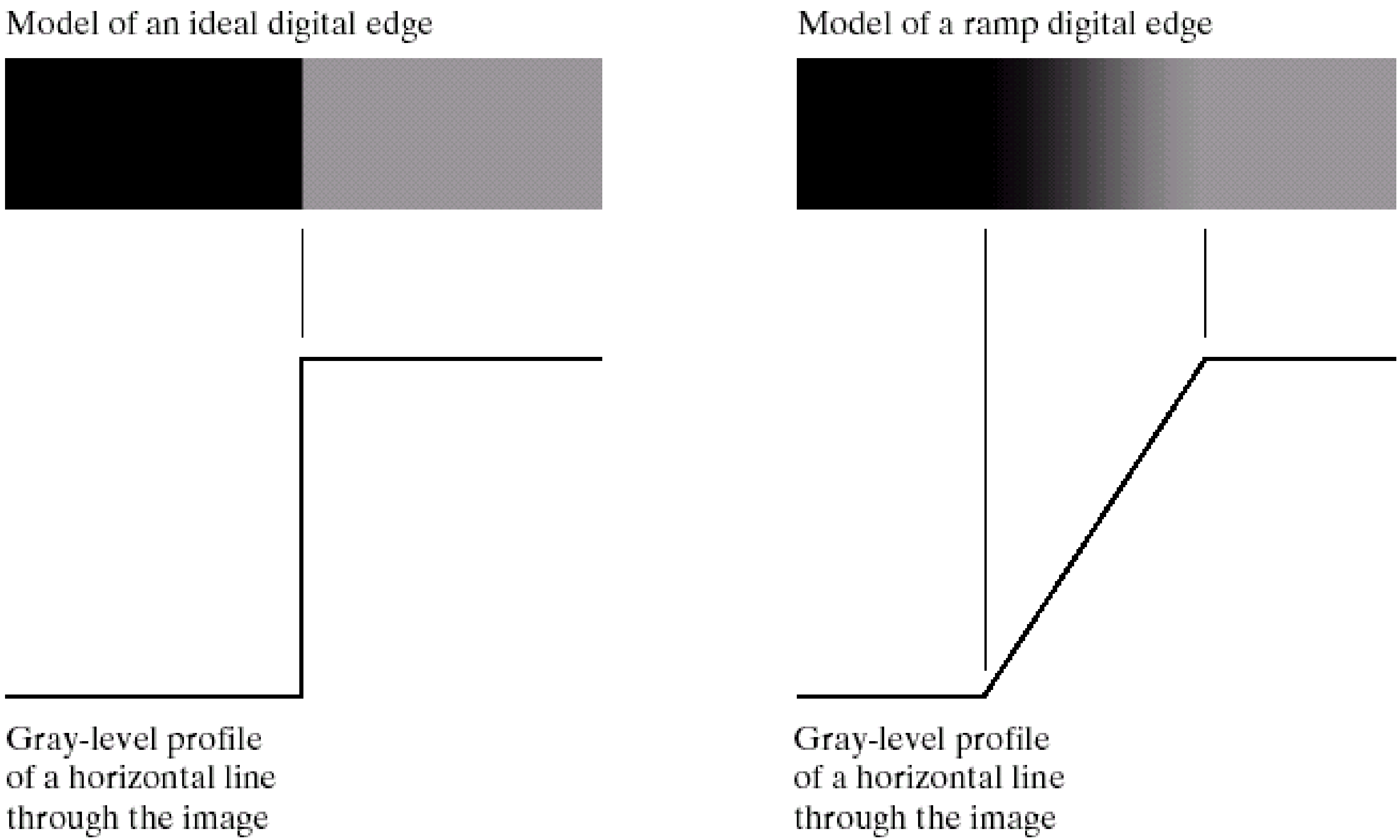
Edge Detection

Model of an ideal digital edge

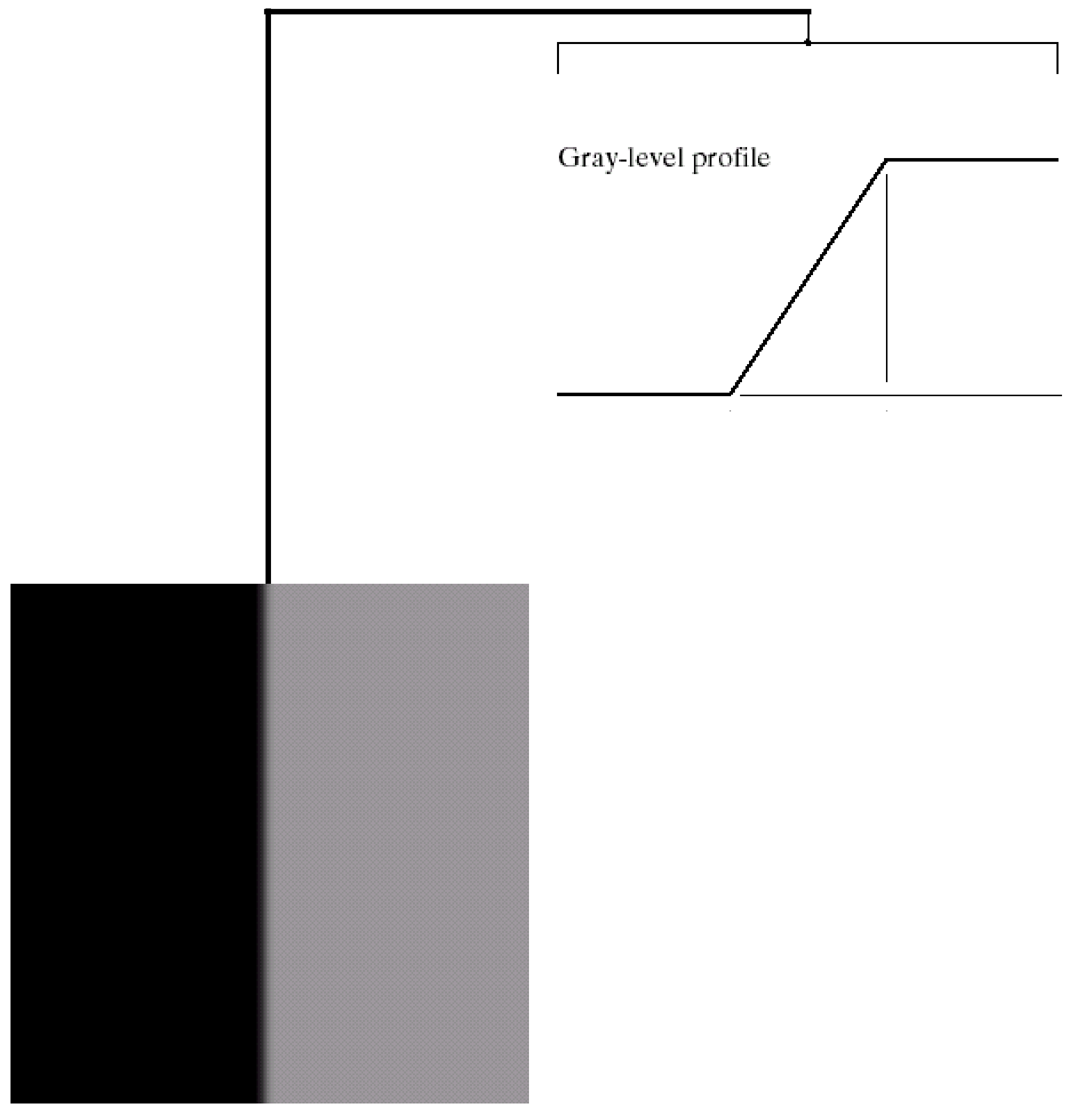


Gray-level profile
of a horizontal line
through the image

Edge Detection

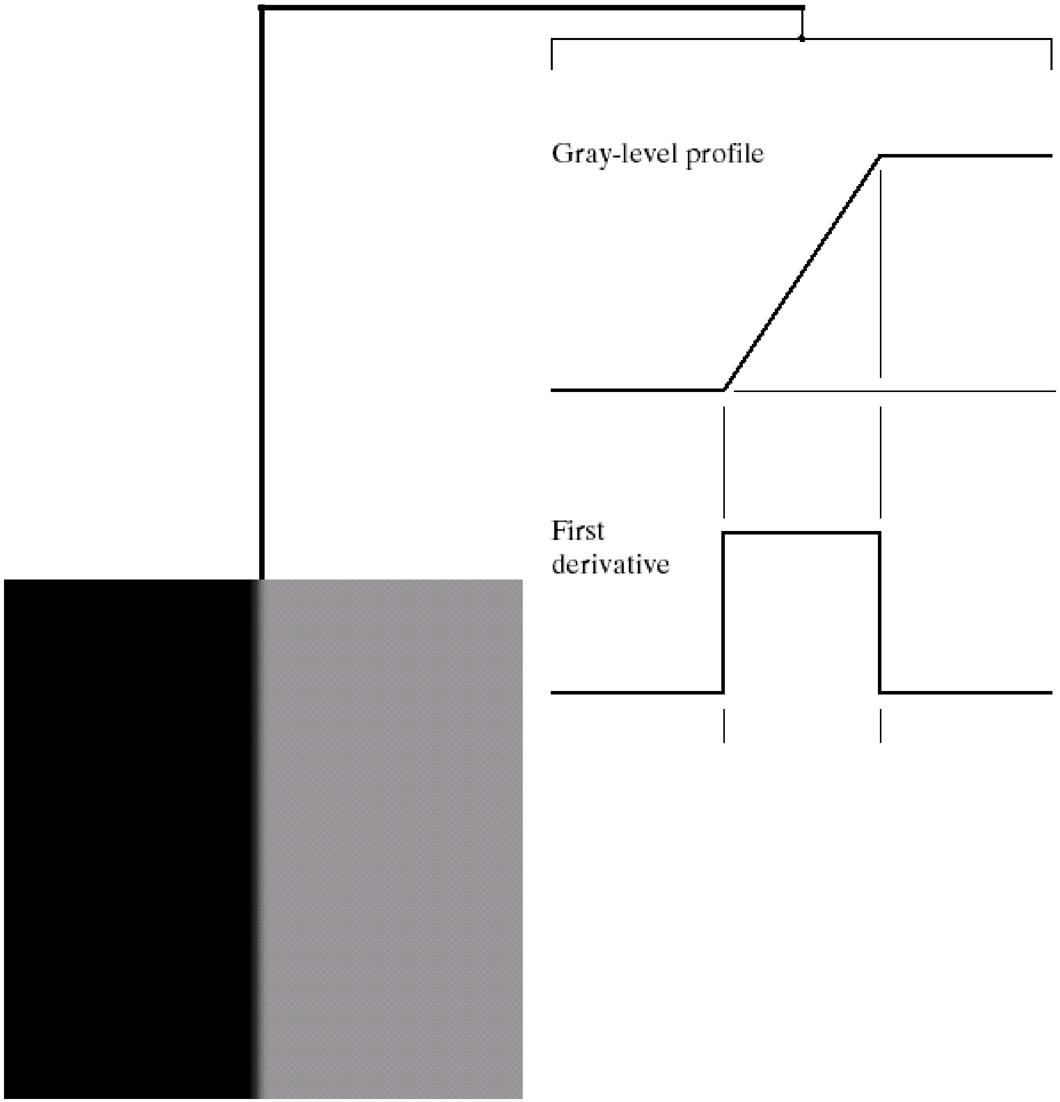


Edges & Derivatives



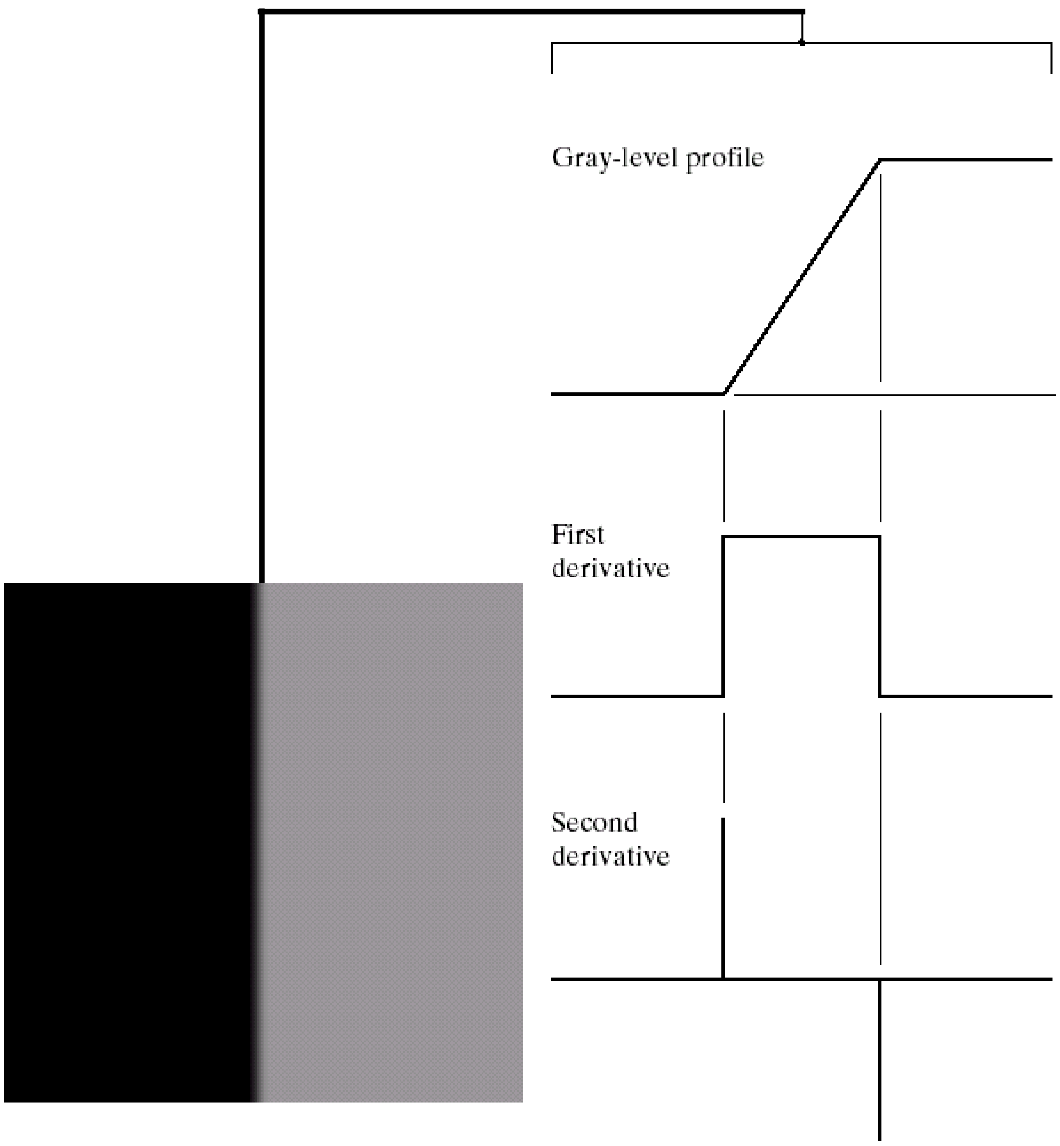
Edges & Derivatives

- 1st derivative denotes the location of edge



Edges & Derivatives

- 1st derivative denotes the location of edge
- 2nd derivative shows location of edge and its direction

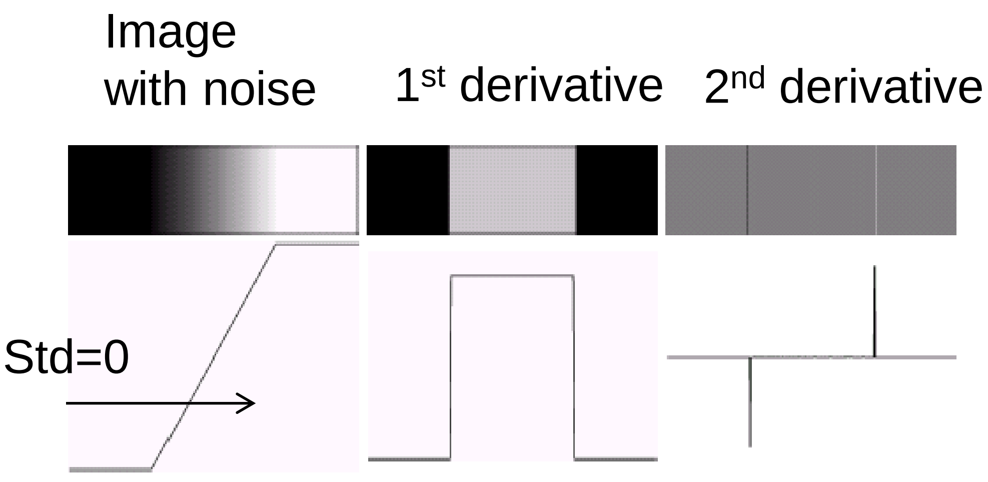


Derivatives & Noise

- Derivative based edge detectors are sensitive to noise

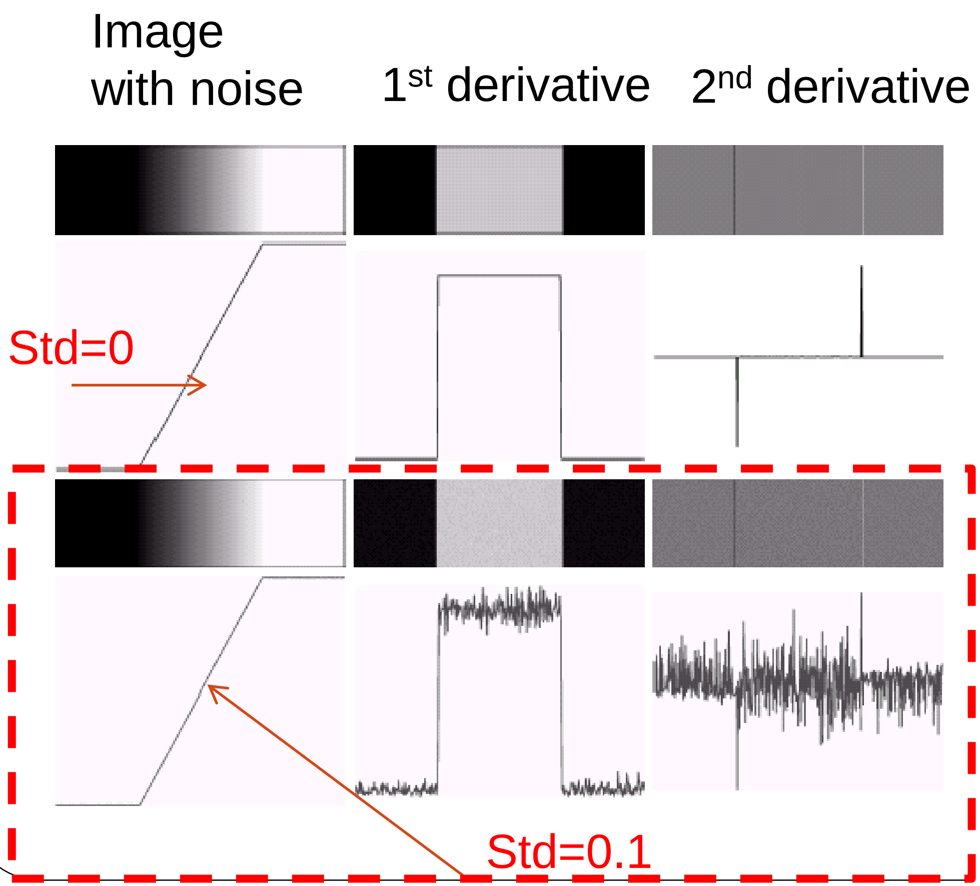
Derivatives & Noise

- Derivative based edge detectors are sensitive to noise



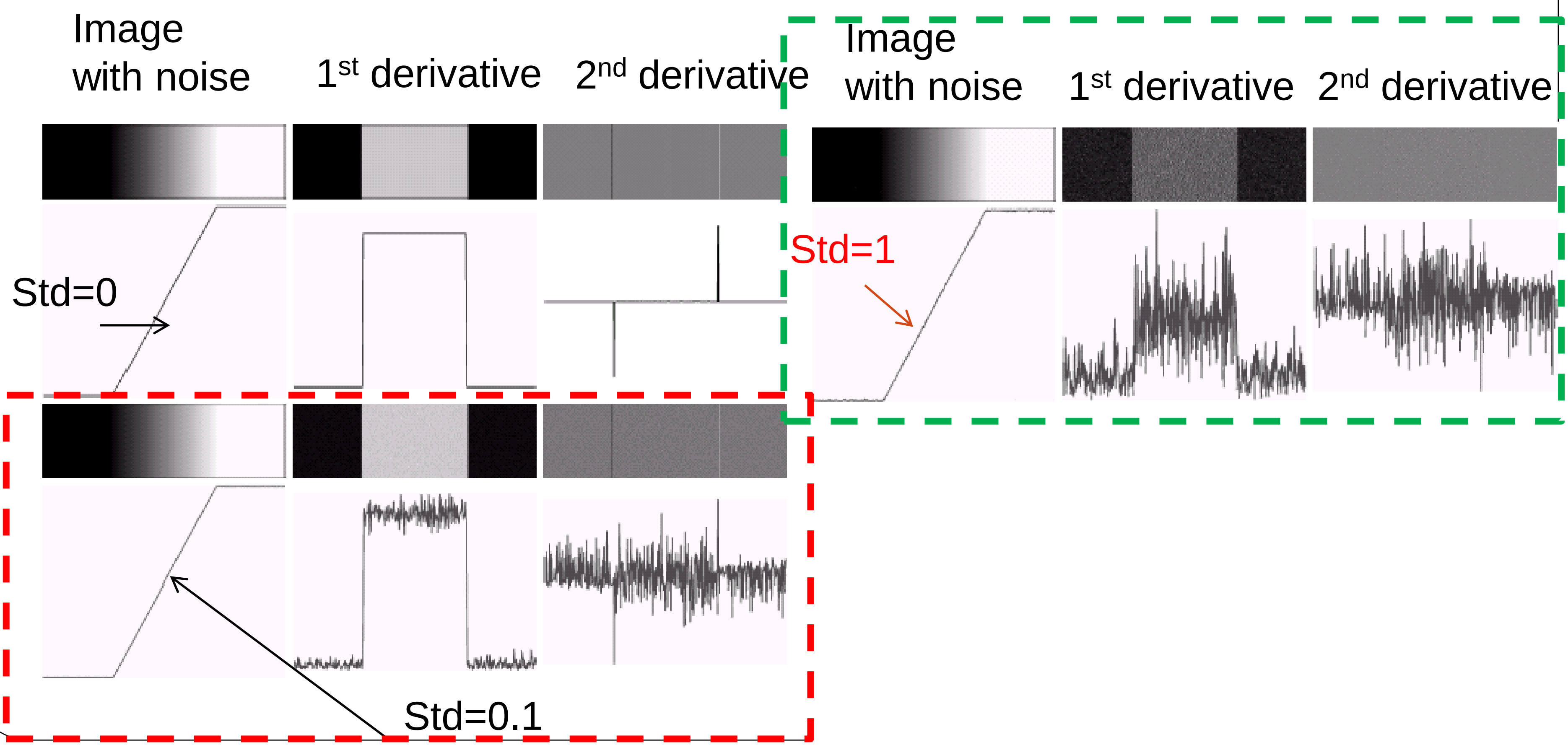
Derivatives & Noise

- Derivative based edge detectors are extremely sensitive to noise



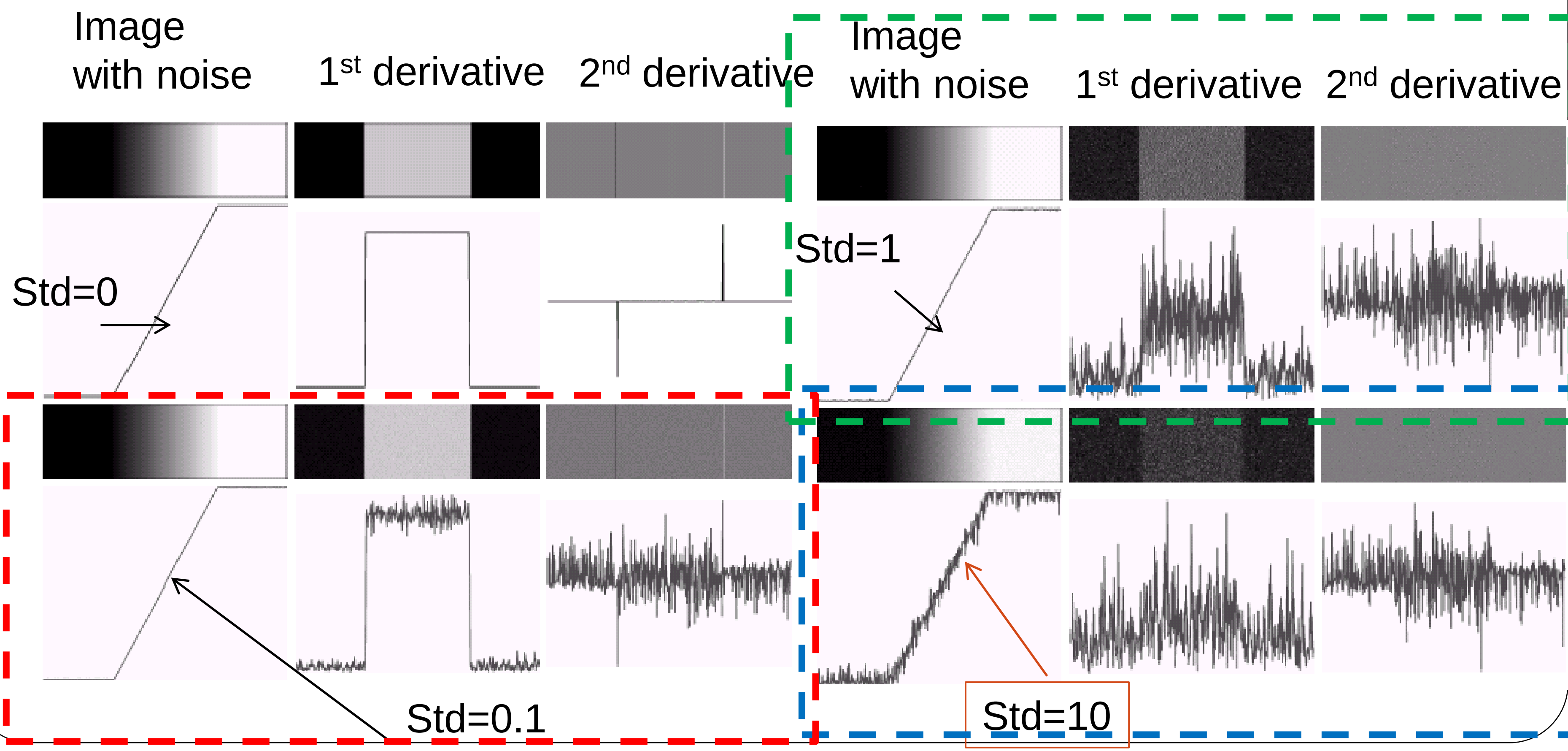
Derivatives & Noise

- Derivative based edge detectors are extremely sensitive to noise



Derivatives & Noise

- Derivative based edge detectors are extremely sensitive to noise



Edge Detection of Noisy Images

- Images can have noise pixels
- Image can also have too many details
- These details are considered as noise
- Ex: Brickwork of the house is considered as noise
- To overcome this, smooth(blur) images prior to edge detection

edge detection with and without smoothing

Apply 5x5 averaging filter before edge detection

without smoothing

with smoothing

Canny edge detector

- Low error rate
  - All edges are detected
  - There are no spurious edges
- Edge points are localized
  - Detected edges are close to the true edge
- Single edge point response
  - Detector return only one point for each true edge point
  - Does not identify multiple edge pixels where only single edge point exists

Canny edge detection algorithm

1. Smoothen the image by Gaussian low pass filter
2. Compute the gradient magnitude and angle at each pixel
3. Detect edge points if magnitude of gradient is above a threshold
4. If above threshold, determine edge direction
5. If neighboring pixel is also edge point and has same edge direction then add it as a edge point
6. Link edge points

Steps for Canny edge detection

1. Image is f(x,y)
2. G(x, y) denote Gaussian low pass filter

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

3. Filter image

F(x, y)=G(x, y) * f(x, y)

4. Determine magnitude of gradient vector for F(x,y)

$$M(x, y) = \sqrt{g_x^2 + g_y^2}$$

Steps for Canny edge detection

5. Determine angle of gradient vector

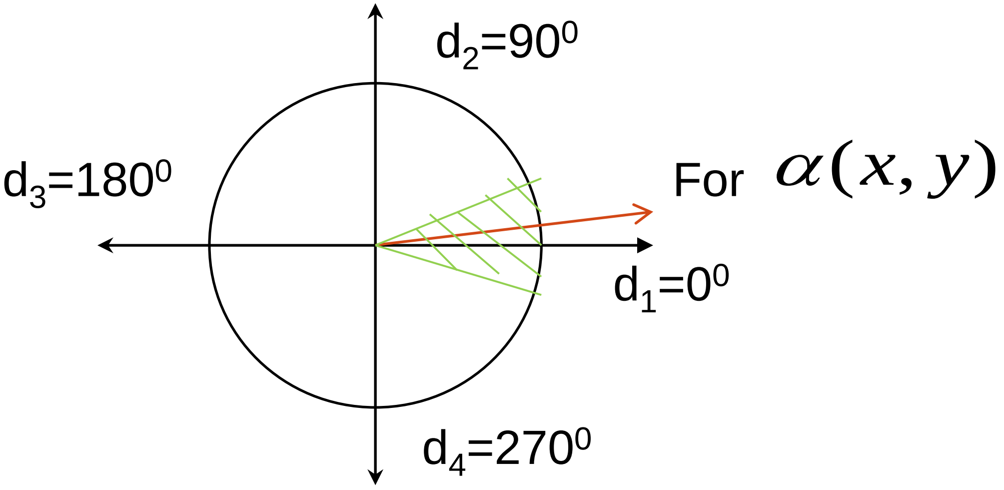
$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

6. Apply non-maxima suppression to generate image, $g_N(x,y)$

7. Reduce false edge points by using hysteresis threshold to generate $G_{NL}(x,y)$ and $G_{NH}(x,y)$

8. Fill gaps between strong edge points

Steps for non-maxima suppression



- d_1,d_2,d_3 and d_4 denote basic edge direction for 3x3 region
- Identify region of d_1,d_2, d_3 or d_4 that contains α(x,y) for each point
- Let M(x,y) denote magnitude of the gradient vector of the point for which either of d_1,d_2, d_3 or d_4 are identified
- Identified points are edge points

Steps for non-maxima suppression

- Identify 2 neighboring points on the same vector (same gradient angle) of identified edge points and determine their magnitudes, $M_1(x, y)$ and $M_2(x, y)$
- Assume that $g_N(x,y)$ denote image matrix for edge points
- If $M(x, y) < M_1(x, y)$ or $M_2(x, y)$ then $g_N(x,y) = 0$ because (x, y) is not on the edge.
  Else $g_N(x,y) = M(x, y)$
- $g_N(x,y)$ is non-maxima suppressed image and contains magnitude of edge points

Steps for Canny edge detection

5. Determine angle of gradient vector

$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

6. Apply non-maxima suppression to generate image, $g_N(x,y)$

7. Reduce false edge points by using hysteresis threshold to generate $g_{NL}(x,y)$ and $g_{NH}(x,y)$

8. Fill gaps between strong edge points

Apply hysteresis threshold

- $g_N(x,y)$ contains gradient magnitudes of edge pixels
- Generate two new images, $g_{NH}(x,y)$ and $g_{NL}(x,y)$ from $g_N(x,y)$
- $g_{NH}(x,y) = g_N(x,y)$ if $g_N(x,y) >= T_H$ else $g_{NH}(x,y) = 0$
  $g_{NH}(x,y)$ are strong and are valid edge pixels
- $g_{NL}(x,y) = g_N(x,y)$ if $g_N(x,y) >= T_L$ else $g_{NL}(x,y) = 0$
  $g_{NL}(x,y)$ are weak edge points
- $g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$
  $g_{NL}(x,y)$ contains pixels with gradient magnitude greater than lower threshold and less than high threshold

Steps for Canny edge detection contd

5. Determine angle of gradient vector

$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

6. Apply non-maxima suppression to generate image, $G_N(x,y)$

7. Reduce false edge points by using hysteresis threshold to generate weak edge points, $G_{NL}(x,y)$ and strong edge points, $G_{NH}(x,y)$

8. Fill gaps between strong edge points using weak edge points

Steps to fill gaps to form long edges

- Locate a pixel, p, in $g_{NH}(x,y)$
- Identify points in $g_{NL}(x,y)$ that are connected to p of $g_{NH}(x,y)$ using 8-connectivity
- 8-connectivity means difference in intensity of surrounding pixels and center pixel is <= threshold
- If a pixel of $g_{NL}(x,y)$ is 8-connected to a pixel $g_{NH}(x,y)$ then it can be a valid edge because its gradient magnitude is large
- Append these valid edge points of $g_{NL}(x,y)$ to $g_{NH}(x,y)$
- Repeat above steps for the remaining pixels in $g_{NH}(x,y)$
- Set remaining pixels of $g_{NL}(x,y)$ to 0

$g_{NL}(x,y)$

$g_{NH}(x,y)$

a	b	c
d	p	e
f	g	h

Whether

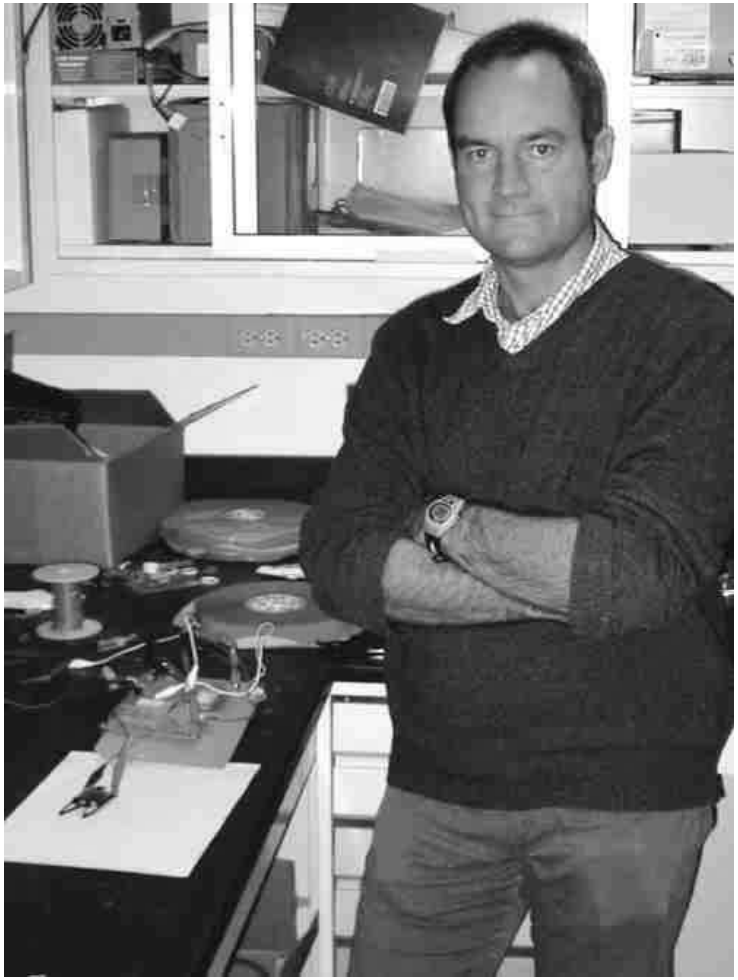
p - c >= T

p - e >= T

:

:

Canny edge detection



Canny edge detection

Edge Linking

- Edge detection characterize edge points
- Breaks may be present in the edges
- Edge linking is required to link edge points with gaps between two edge points
- Linking assembles edge pixels into meaningful edges and region boundaries

Edge Linking

- If edge point are close to each other, they may belong to one edge
- Two properties of edge points are useful for edge linking:
  - the strength (or magnitude) of the detected edge points
  - their directions (determined from gradient directions)
- Adjacent edge points with similar magnitude and direction are linked
- Example: Given an edge pixel with coordinates (x_0,y_0)
- Neighborhood of pixel at (x,y) is also an edge pixel if-

$$\left|M(x,y)-M(x_0,y_0)\right| \le E, \quad E: \text{a nonnegative threshold}$$

$$\left|\alpha(x,y)-\alpha(x_0,y_0)\right| < A, \quad A: \text{a nonegative angle threshold}$$

Edge Linking

- Compute strength and direction of gradient vector for pixel (x,y) on edge

$$\nabla \mathrm{f} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

$$M(x, y) = \sqrt{g_x^2 + g_y^2}$$

$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

- Compute magnitude of gradient for each neighboring pixel of (x,y)

Edge Linking

		(a,b)
	(x,y)	

- For the given threshold, E and A
- If $|M(a, b) - M(x, y)| <= T_E$

  and $\alpha(a, b) - \alpha(x, y) < = T_A$
- then the pixel with coordinates (a, b) is linked to the pixel at (x, y)

Edge Linking Algorithm

1. Compute gradient magnitude and angle arrays, $M(x, y)$ and $\alpha(x, y)$ of all pixels of image $f(x, y)$
2. Form binary image $g(x,y)$ such that
3. if $|M(a, b) - M(x, y)| <= T_E$
   and $\alpha(x,y) <= T_A$
   then $g(x,y) = 1$
   else $g(x,y) = 0$

1	1	0	1	0	1
1	0	0	0	1	0
0	0	0	1	1	1

Edge Linking

1	1	0	1	0	1
1	0	0	0	1	0
0	0	0	1	1	1

4. Assume link value, k
5. Scan each row of g(x,y) and identify 0s
6. If number consecutive zeros < = k
   then fill gaps with '1'

k = 2

1	1	1	1	1	1
1	0	0	0	1	0
0	0	0	1	1	1

Edge detection

Sobel filter

canny

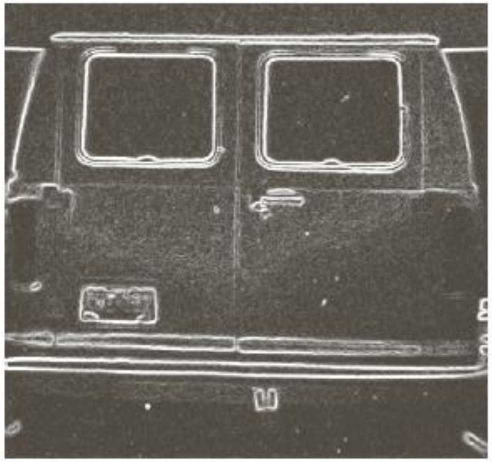
Edge linking
to fill gaps

Edge Detection and Linking

Edge Detection and Linking

Original image

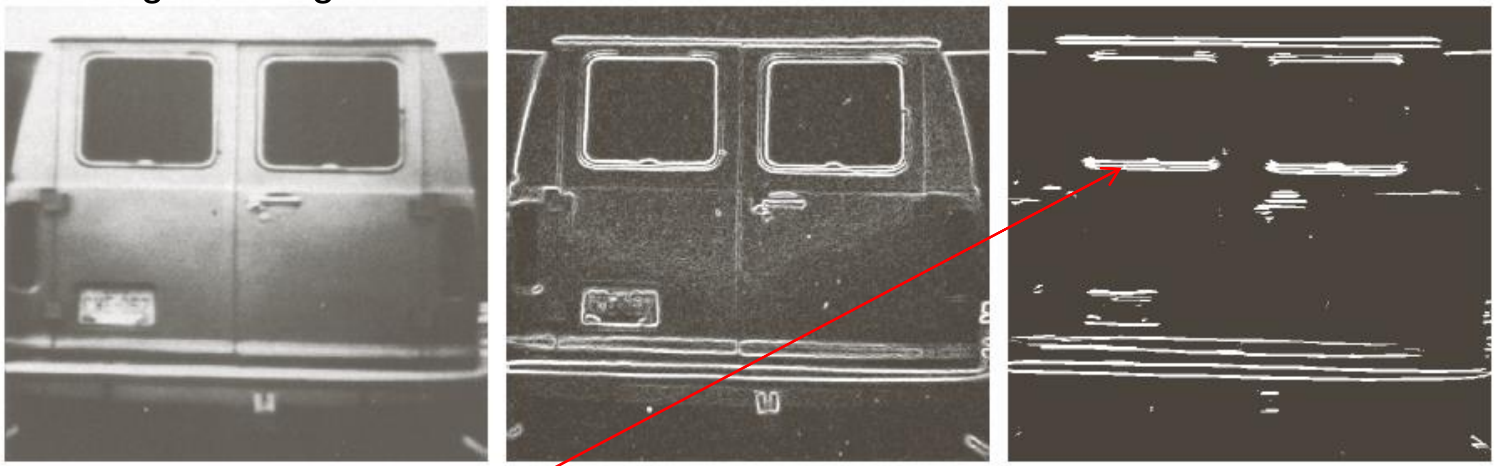
Gradient magnitude image



Edge Detection and Linking

Original image

Gradient magnitude image



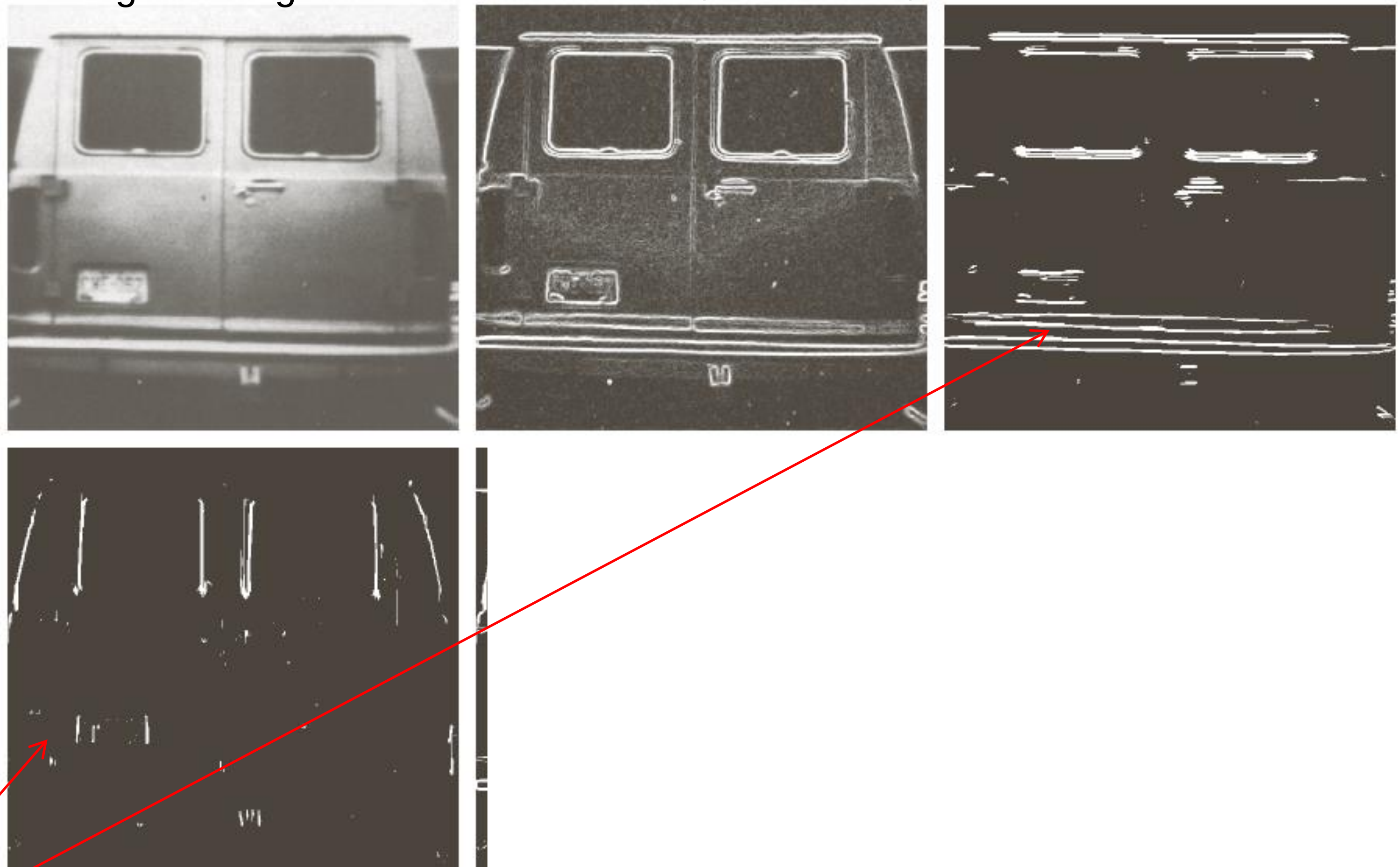
E=30% of max gradient magnitude

A = 90^0

$T_A = 45^0$

K = 25 for horizontal direction and vertical direction

Original image Gradient magnitude image



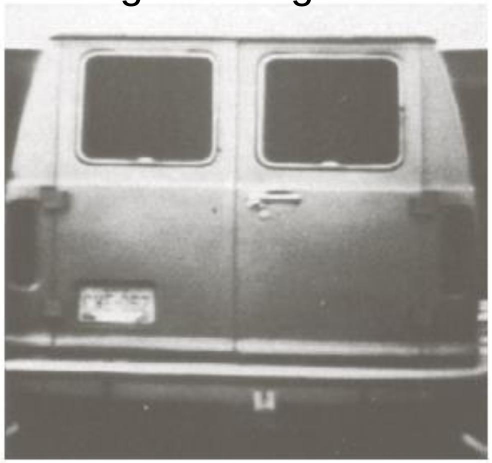
E=30% of max gradient magnitude

A = 90^0

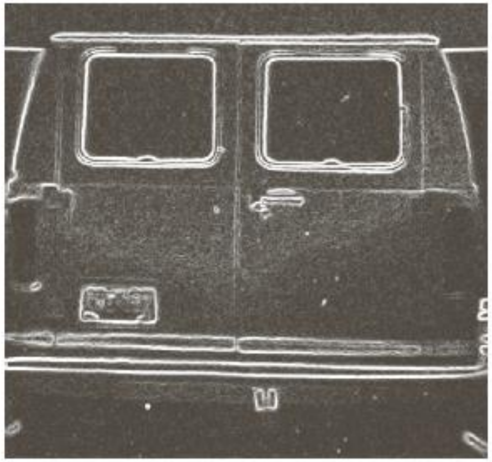
$T_A = 45^0$

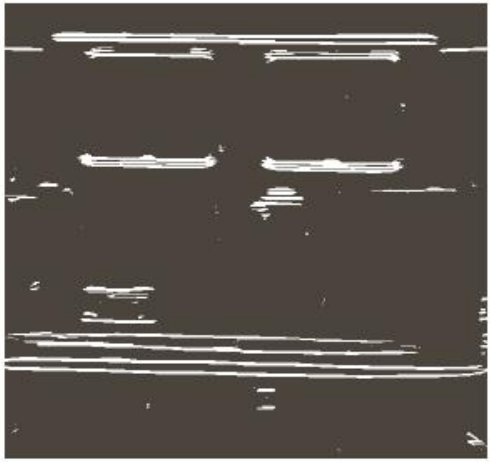
K = 25 for horizontal direction and vertical direction

Original image

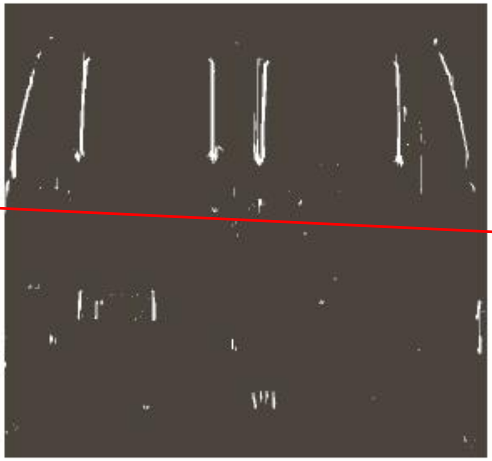


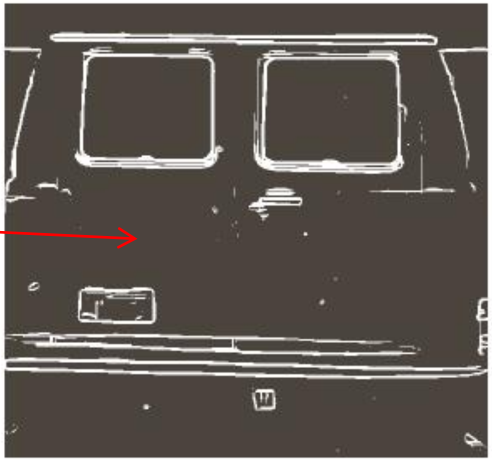
Gradient magnitude image

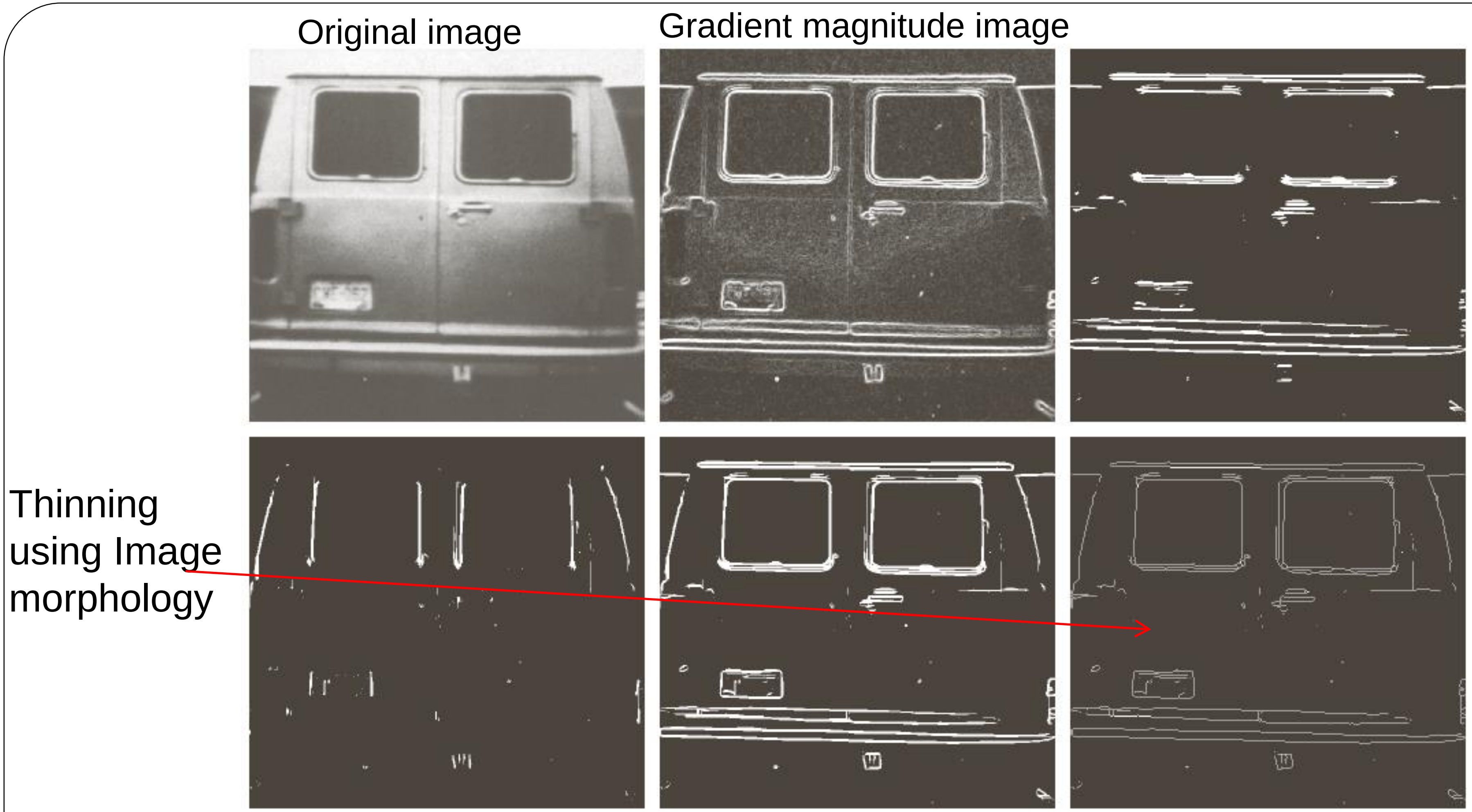




Addition of horizontal and vertical linked edges






Original image
Gradient magnitude image
Thinning
using Image
morphology

What is Image segmentation?

- Refers to the process of partitioning an image into groups of pixels which are homogeneous with respect to some criterion.
- Different groups must not intersect with each other, and adjacent groups must be heterogeneous
- Segmentation algorithms are area oriented instead of pixel-oriented
- The result of segmentation is the splitting up of the image into connected areas
- Thus segmentation is concerned with dividing an image into meaningful regions

CLASSIFICATION OF IMAGE-SEGMENTATION TECHNIQUES

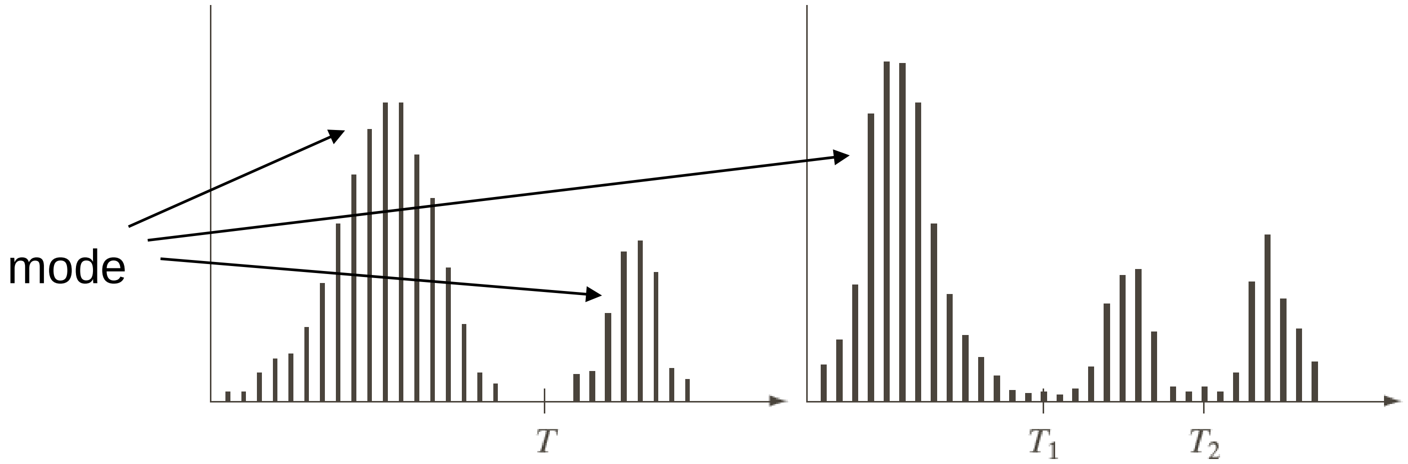
- Local Segmentation
  - Deals with segmenting sub-images which are small windows on a whole image
  - The number of pixels available to local segmentation is much lower than global segmentation
- Global Segmentation
  - is concerned with segmenting a whole image
  - deals mostly with segments consisting of a relatively large number of pixels
  - This makes estimated parameter values for global segments more robust

Approaches for segmentation

1. Region approach
   - Regions in an image are a group of connected pixels with similar properties
   - Each pixel is assigned to a particular object or region
2. Boundary approach
   - locates boundaries that exist between the regions.
3. Edge approach
   - edges are identified first, and then they are linked together to form required boundaries

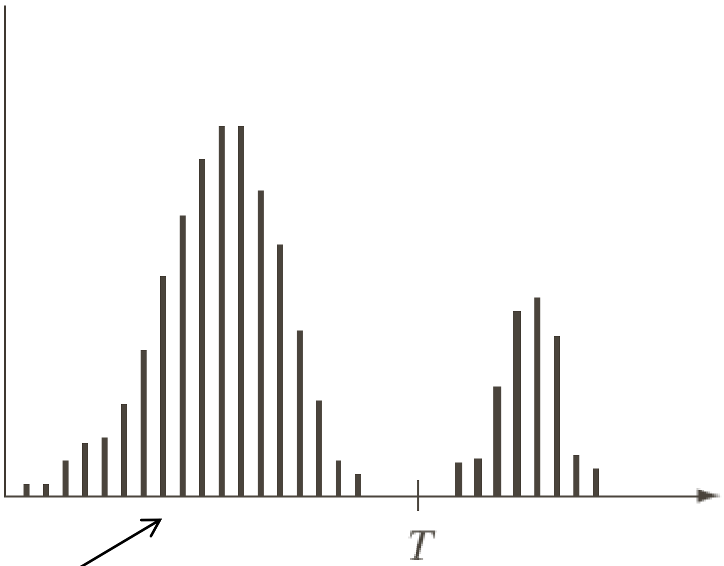
Intensity Thresholding for segmentation

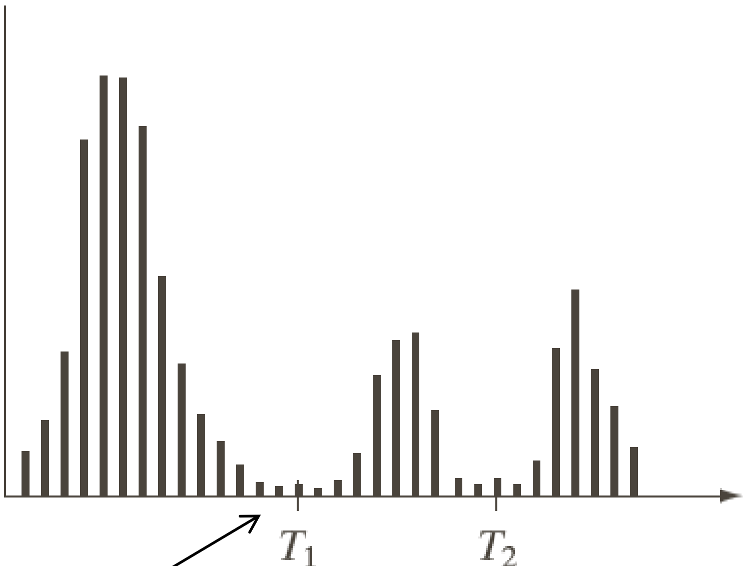
- Based on histogram
- Depends on width and depth of the valleys separating the modes in histogram
- Valleys depend on
  - Separation between peaks
  - Relative size of object against background
  - Noise contents
  - Uniformity of the illumination source



Thresholding

histograms





Two dominant modes

Three dominant modes

For thresholding,

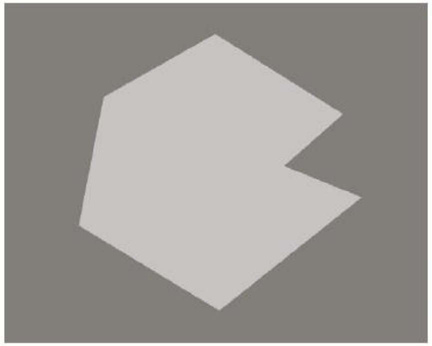
$$g(x,y)=\begin{cases}1\, if \;\; f(x,y)>T\\ 0\, if \;\; f(x,y)\leq T\end{cases}$$

$$g(x,y)=\begin{cases}a & \text{if } f(x,y)>T_2\\ b & \text{if } T_1<f(x,y)\leq T_2\\ c & \text{if } f(x,y)\leq T_1\end{cases}$$

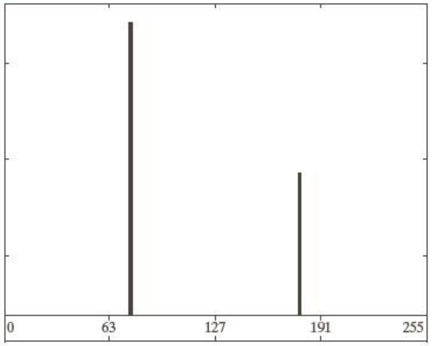
Effect of noise in image thresholding

Gaussian noise, std = 0

Image



histograms



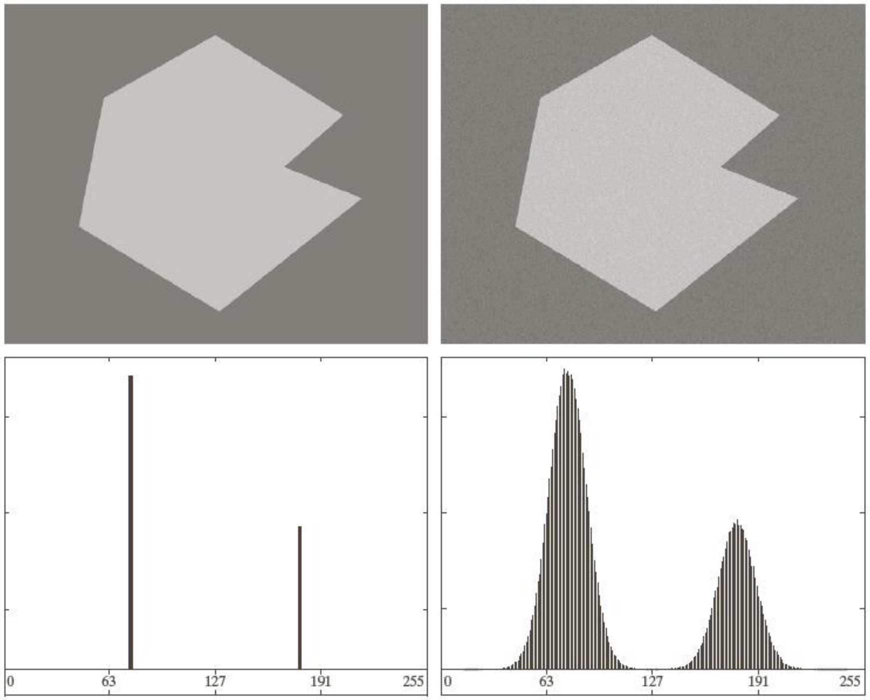
Threshold can be set as two modes can be separated easily

Effect of noise in image thresholding

Gaussian noise, std = 0 | Gaussian noise, std 10

Image

histograms



Threshold can be set as two modes can be separated easily

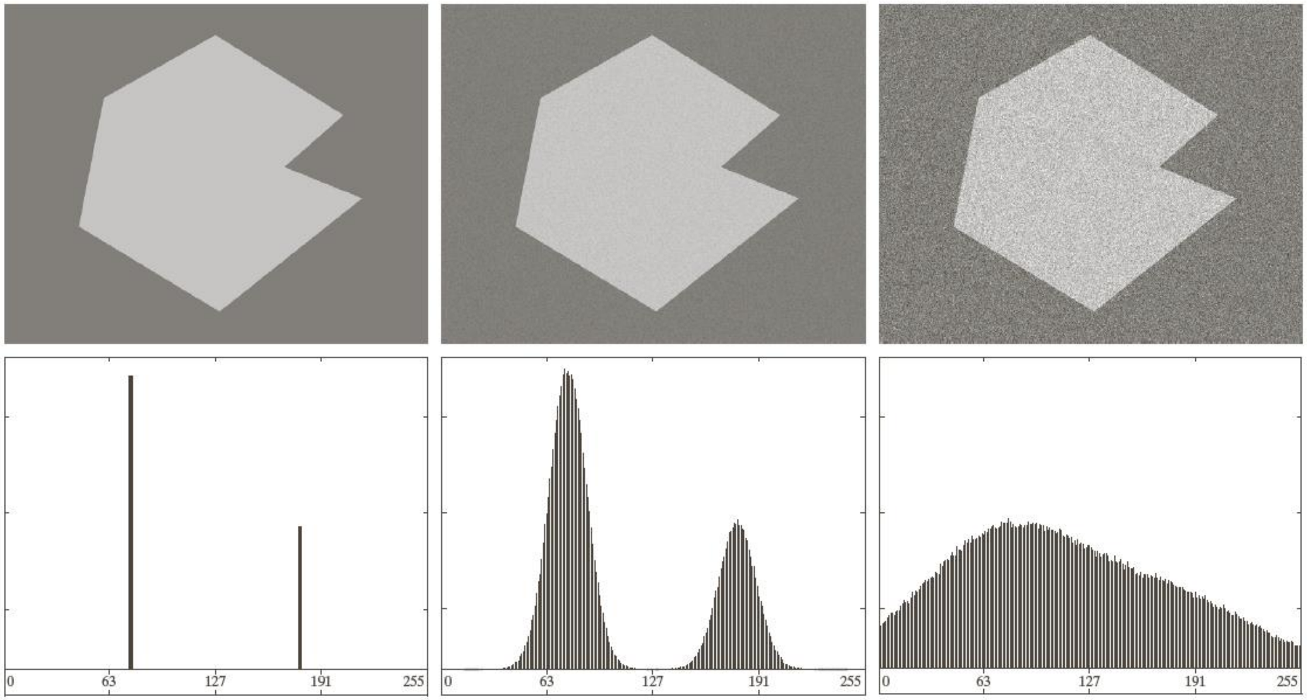
Threshold can be set as two modes can be separated though image is noisy

Effect of noise in image thresholding

Gaussian noise, std = 0 | Gaussian noise, std 10 | Gaussian noise std, 50

Image

histograms



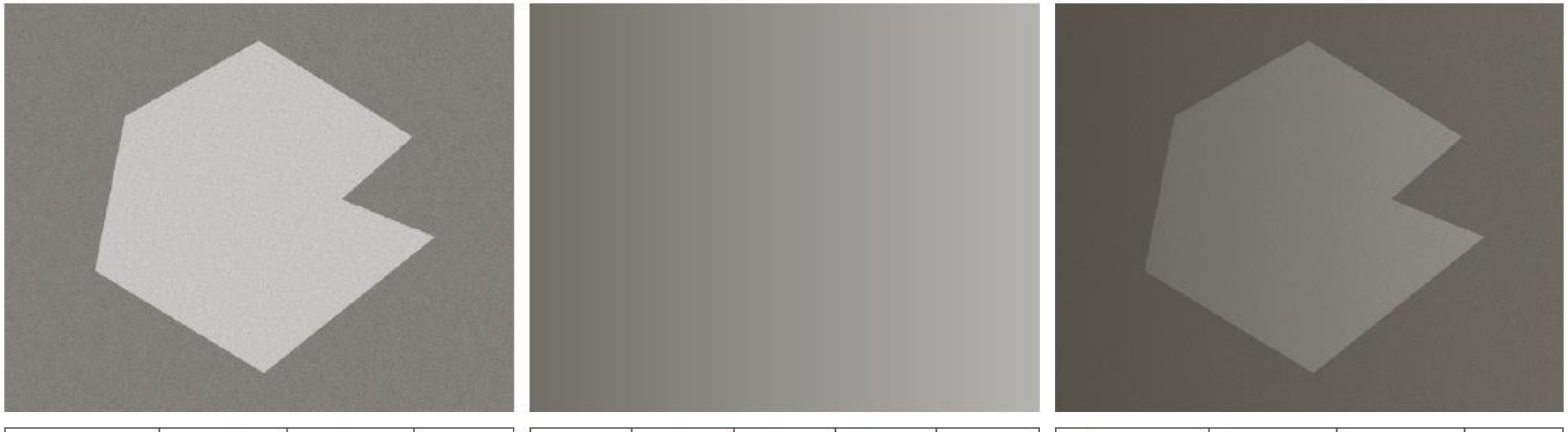
Threshold can be set as two modes can be separated easily

Threshold can be set as two modes can be separated though image is noisy

Thresholding is not possible as modes have merged

Role of illumination in image thresholding

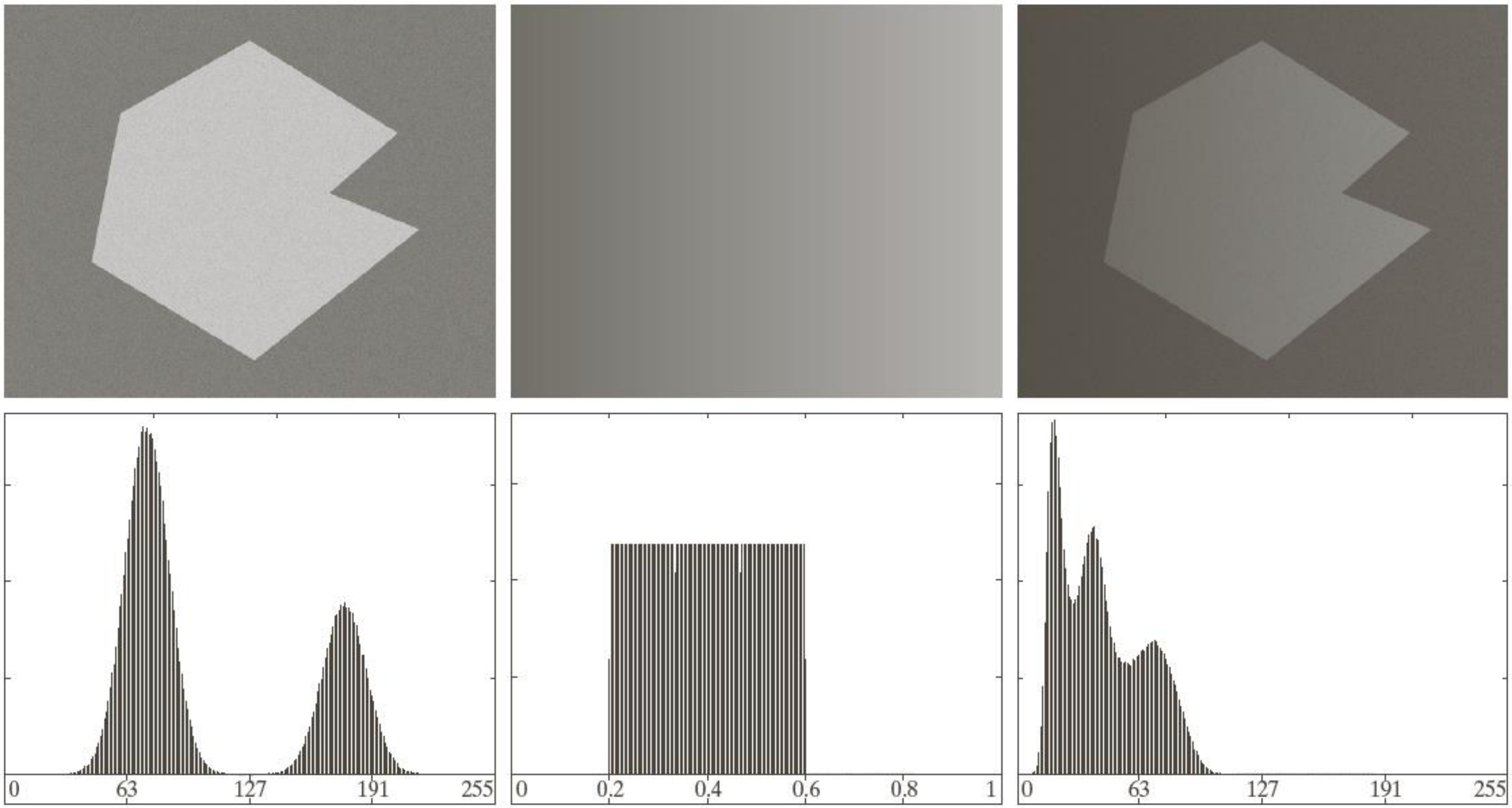
Image



Role of illumination in image thresholding

Image

histog
rams



Threshold can be set as two modes can be separated

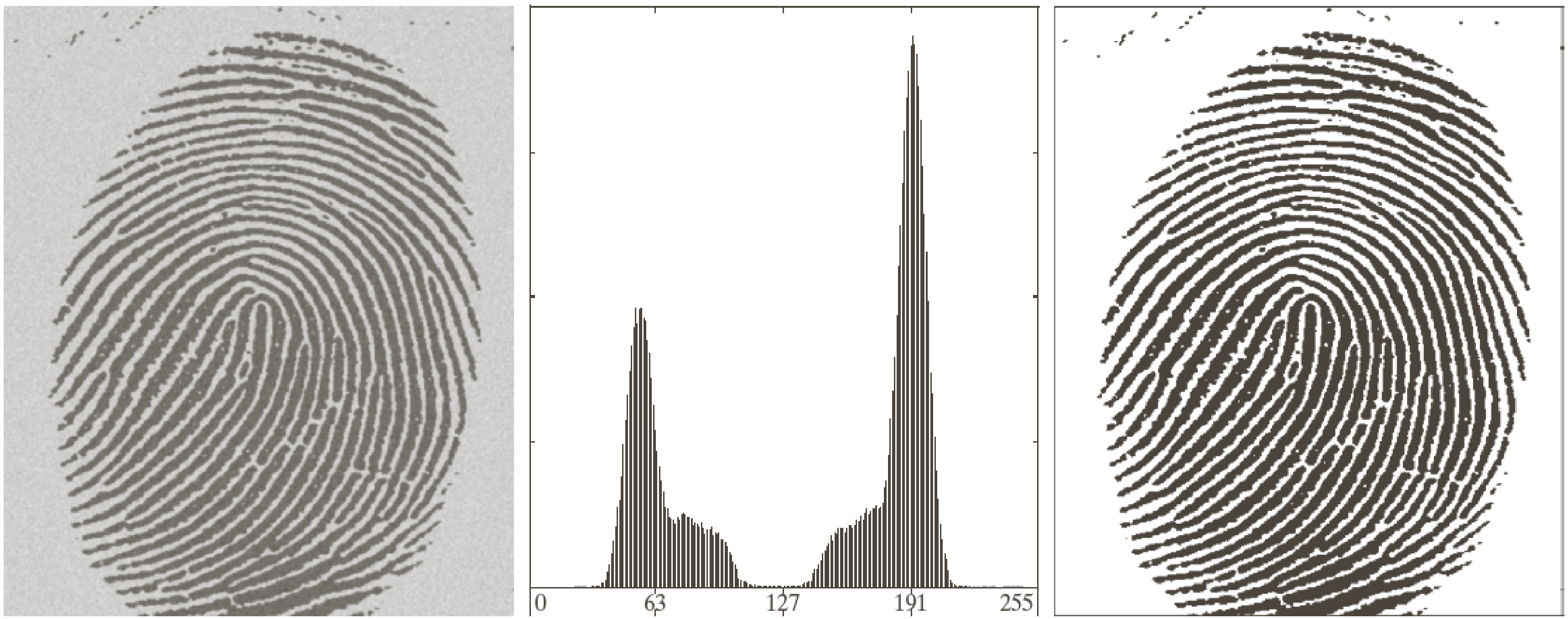
Thresholding is not possible as modes have merged

Thresholding is not possible as modes have merged

Basic Global Thresholding

1. Select an initial estimate for the global threshold, T
2. Segment image using T to form two groups, G_1 and G_2.
3. Compute mean intensity m_1 and m_2 for pixels in each group
4. Compute new threshold
   $T_{new}=(m_1+m_2)/2$
5. Determine $T- T_{new}$
6. If $T - T_{new} > \Delta$
   then $T = T_{new}$ go to step 5
   else end
5. Repeat step 2 through 7 until $T - T_{new} < \Delta$ (predefined)
6. Predefined difference controls number of iterations
7. Less is the value of Δ, more is the number of iterations

Segmentation based on global thresholding



Optimum global thresholding: Otsu's method

- Let {0,1,2,…,L-1} be L intensity levels and size of image is MxN
- For histogram, n_i is number of pixels with intensity, i
- For normalized histogram, $p_i=n_i/MN$
- Ex: 3-bit image has n = {4,2,8,1,0, 5,2,3}
  for intensity levels, i = {0,1,2,3,4,5,6,7}
- p = {4,2,8,1,0, 5,2,3}/25

- Select threshold, T(k)=k, 0 < k < L-1 to generate classes C_1 and C_2
- Ex: T(1) = 1

Optimum global thresholding: Otsu's method

Normalized Probability, p = {4,2,8,1,0, 5,2,3}/25
For threshold, T(k), probability of C_1 is $P_1(k)$
And for C_2 probability is $P_2(k)$

$$P_1(k) = \sum_{i=0}^{k} p_i$$

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

$P_1(1) = (4+2)/25$
$= 6/25$

$P_2(1) = (8+1+0+5+2+3)/25$
$= 19/25$

$$m_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i p_i$$

$$m_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i p_i$$

$m_1(1) = (25/6)\{(0 \times 4/25) + (1 \times 2/25)\}$
$= (1/3)$

$m_2(1) = (25/19)\{(2 \times 8/25) + (3 \times 1/25)$
$+ (5 \times 5/25) + (6 \times 2/25)$
$+ (7 \times 3/25)\}$
$= (77/19)$

Optimum global thresholding: Otsu's method

p = {4,2,8,1,0, 5,2,3}/25

Average intensity upto k

$$m(k) = \sum_{i=0}^{k} i p_i$$

m(1) = (0 × 4/25) +(1 × 2/25) = (2/25)

Global mean of entire image

$$m_G = \sum_{i=0}^{L-1} i p_i$$

m_G = (0 × 4/25) +(1 × 2/25)+…+ (7 × 3/25) = (79/25)

Optimum global thresholding: Otsu's method

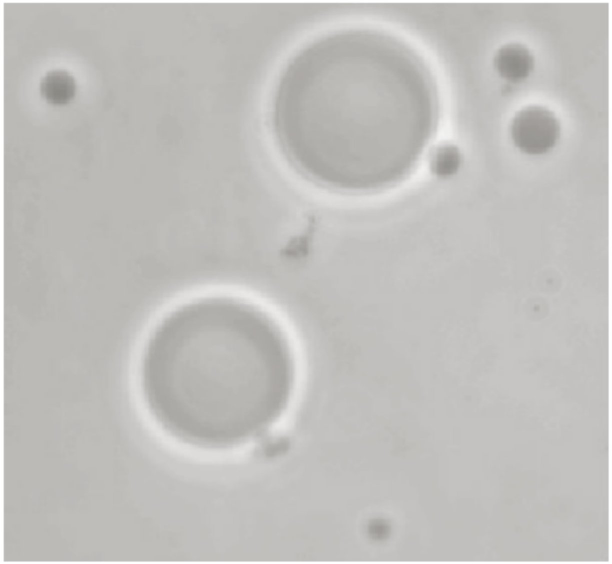
Global variance
$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$$

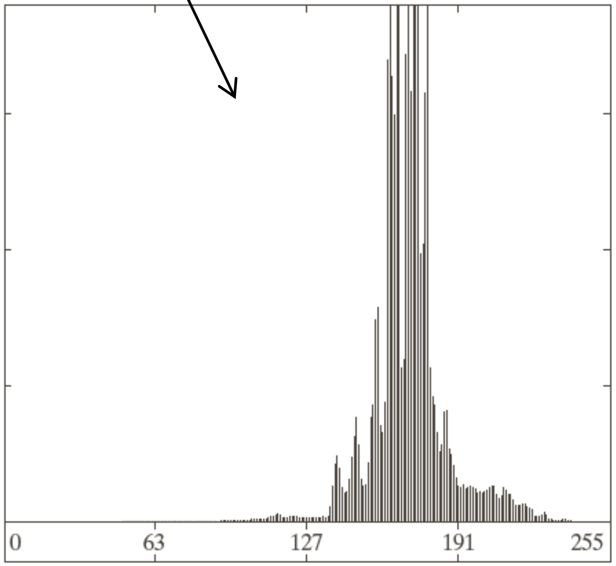
Between the class variance
$$\sigma_B^2(k) = P_1(k)P_2(k)(m_1 - m_2)^2$$

Metric for thresholding
$$\eta = \frac{\sigma_B^2}{\sigma_G^2}$$

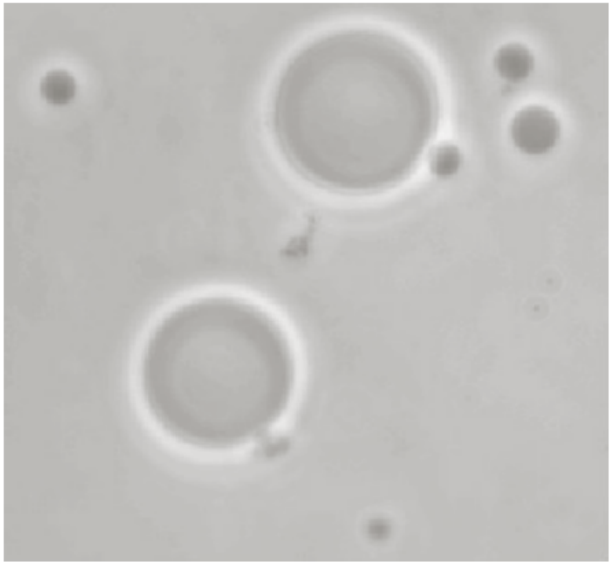
- Large value of metric represents large separability
- Choose k which provides maximum value of metric for thresholding

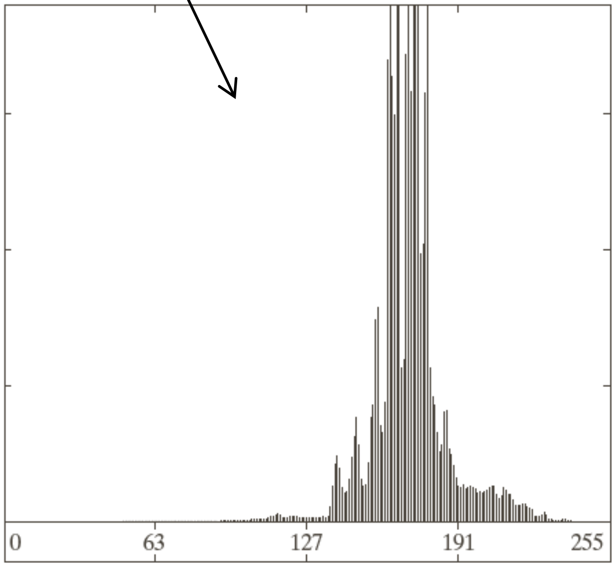
Does not have distinct valleys





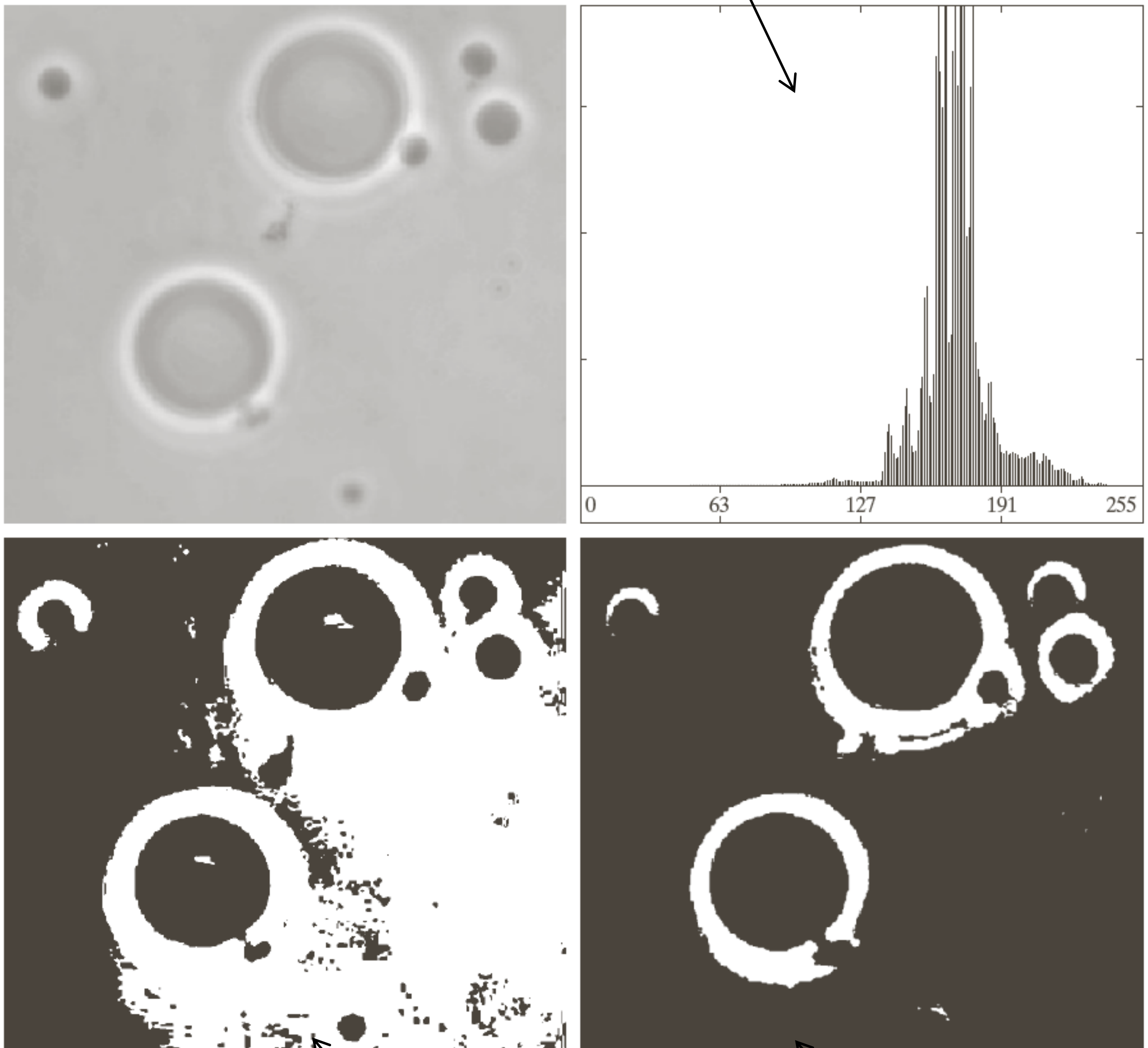
Does not have distinct valleys





Global thresholding, T=169

Does not have distinct valleys



global thresholding T=169

Otsu’s method, T=181
$\eta = 0.467$

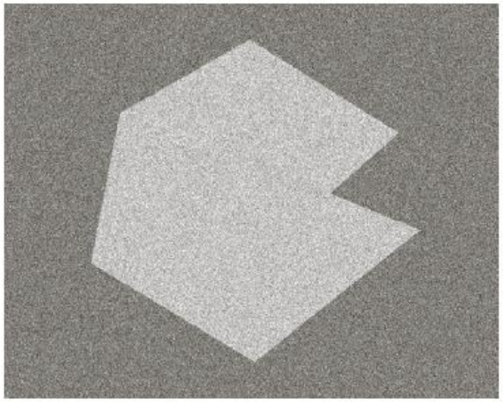
Image smoothing to before Otsu's thresholding

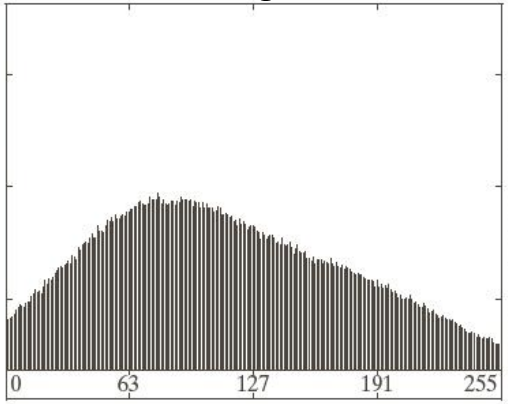
Image smoothing to before Otsu’s thresholding

Histogram

After threshoding

Noisy image





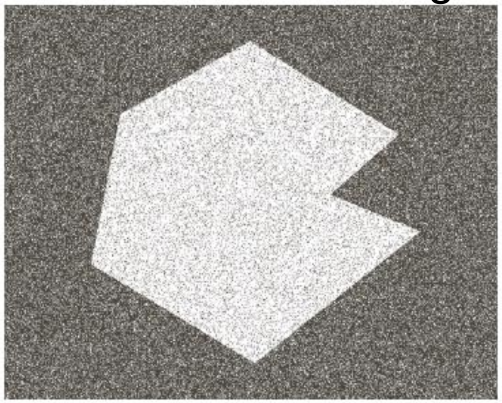
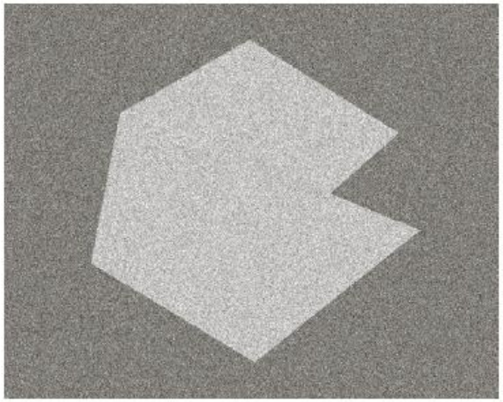


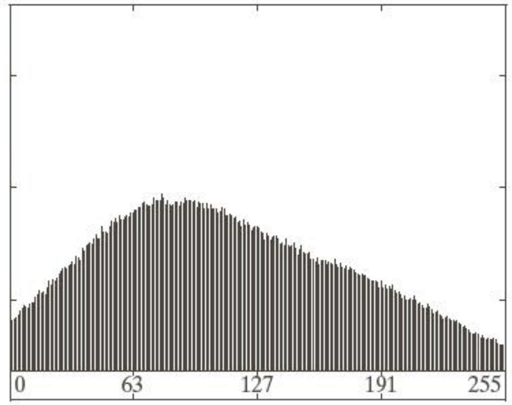
Image smoothing to improve global thresholding

Histogram

After threshoding

Noisy image





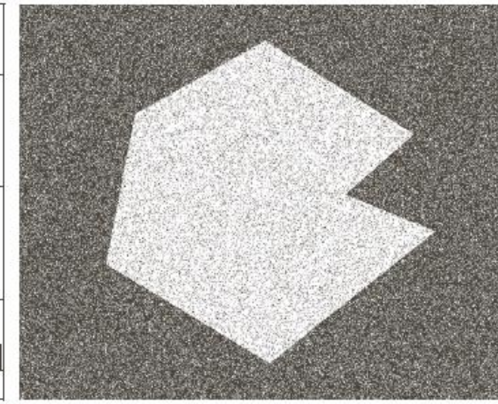
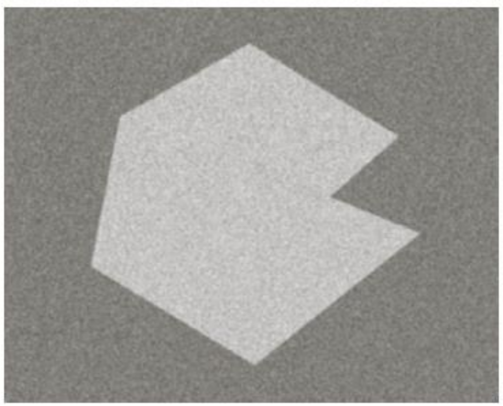
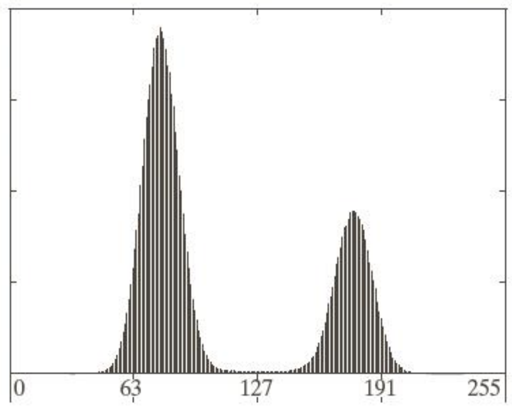
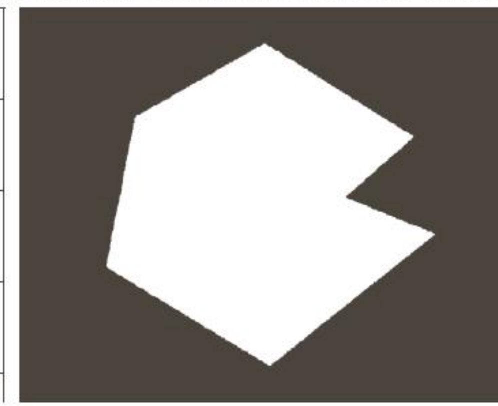


Image after
averaging
by 5x5 filter







Region based segmentation

What is a Region

- A group of connected pixels with similar properties
- Connected pixels correspond to objects in a scene
- Region of connected pixels correspond to objects or parts of an object

Region-based Approach

- Predicate is a condition to define region
- Pixels satisfying the condition of predicate are marked
- Group of these pixels defines an object
- Predicates are based on

  1. Value similarity

     -Gray value differences

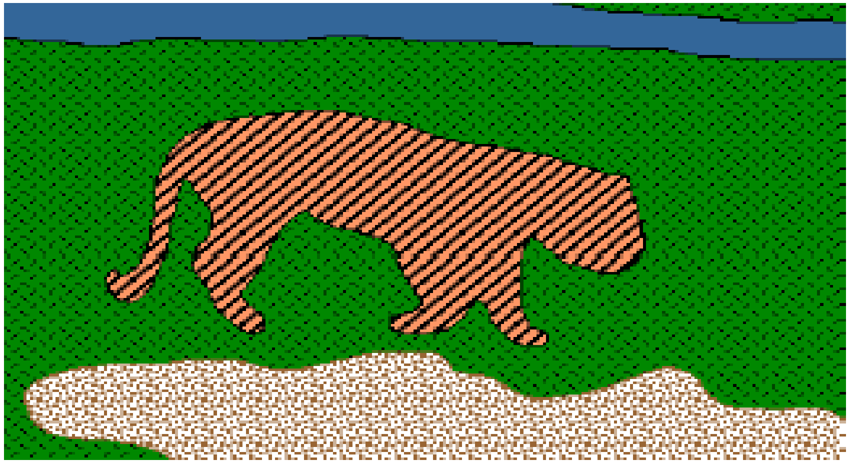
     -Gray value variance

  2. Spatial Proximity

     Euclidean distance

Example: Region Based Segmentation

Original Image



Segmented image

Region Based Segmentation

- Region Growing
- Region Split
- Region Merge
- Region Split and Merge

Region Growing

- Region growing requires a seed to begin with
- Seed could be a single pixel
- A new segment is grown from the seed
- Resultant segment is then removed from the process
- A new seed is chosen from the remaining pixels
- This continues until all pixels have been allocated to a segment
- The resulting segmentation could
  - depend on the initial seed chosen
  - order in which neighboring pixels are examined

Region Growing

- Region growing offers several advantages over conventional segmentation techniques
- The borders of regions found by region growing are perfectly thin and connected
- Stable with respect to noise
- first step is to choose a seed

Region growing Predicate: Example

Predicate: Difference between two pixels <= 10

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

If location of seed changes then regions change

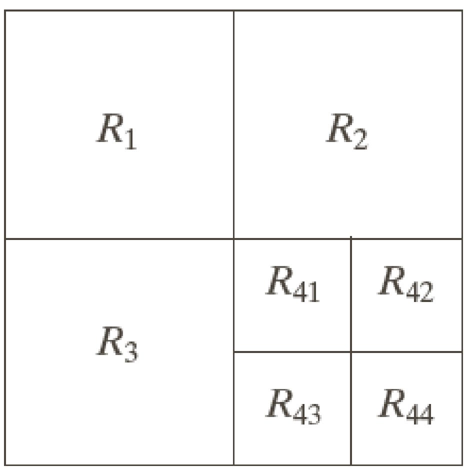
Region growing

Multiple seeds are used

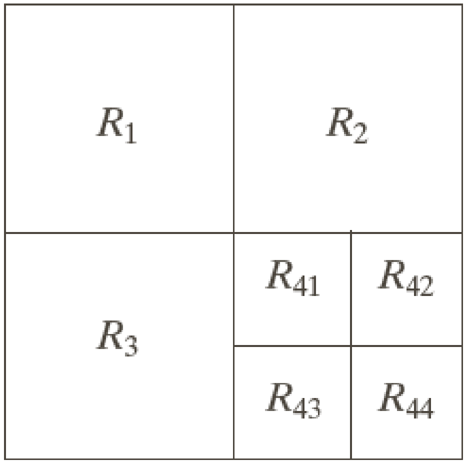
Region Splitting

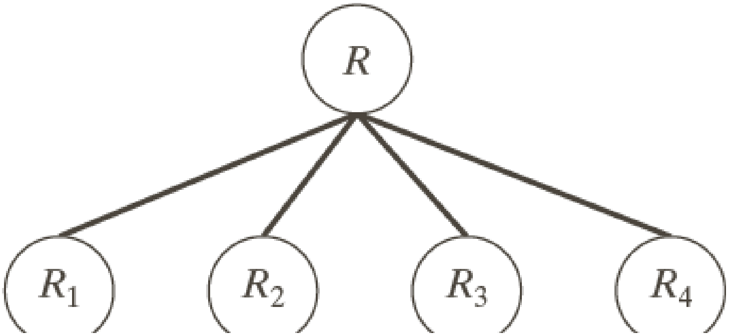
- Region growing starts from a set of seed points
- Opposite approach to region growing is region splitting
- It is a top-down approach and it starts with the assumption that the entire image is homogeneous
- If this is not true, the image is split into four sub images
- Splitting procedure is repeated recursively until we split the image into homogeneous regions

Region Splitting

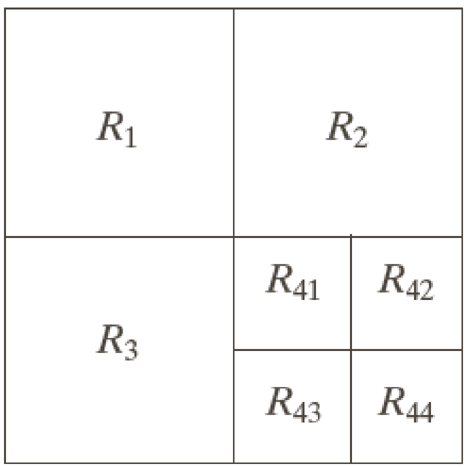


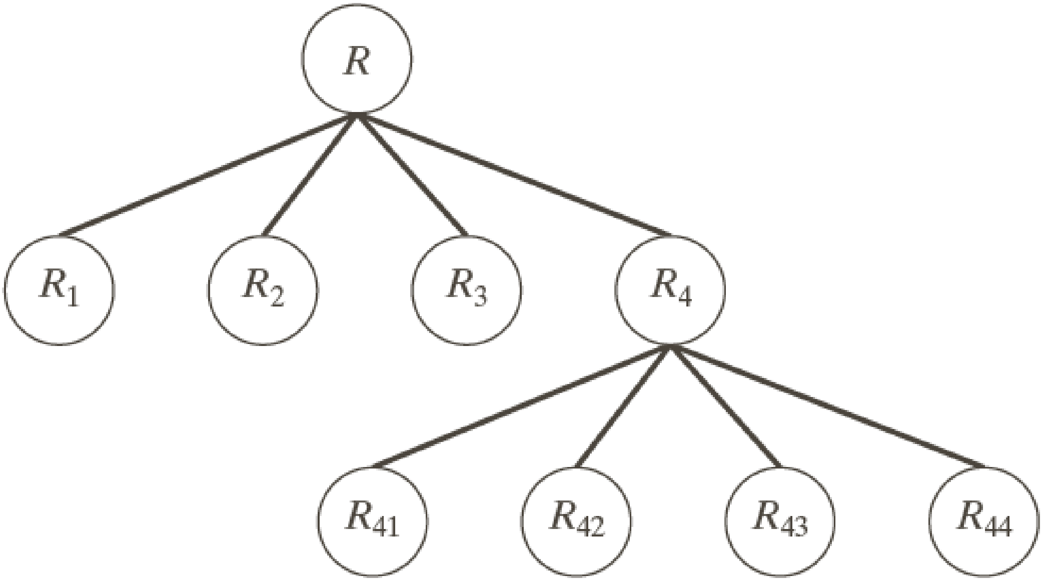
Region Splitting





Region Splitting



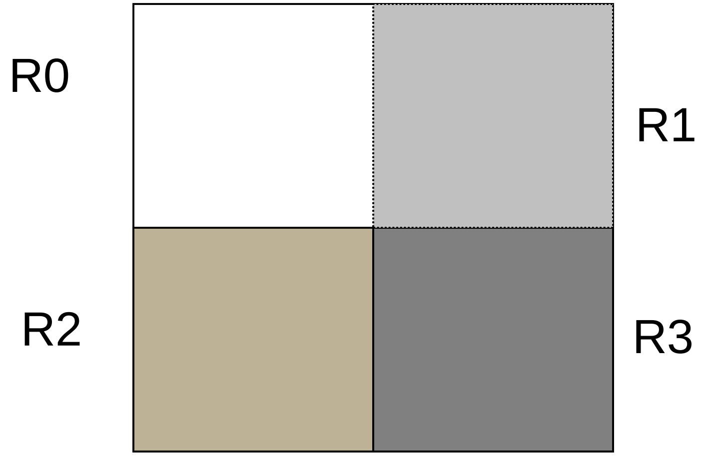


Region Splitting

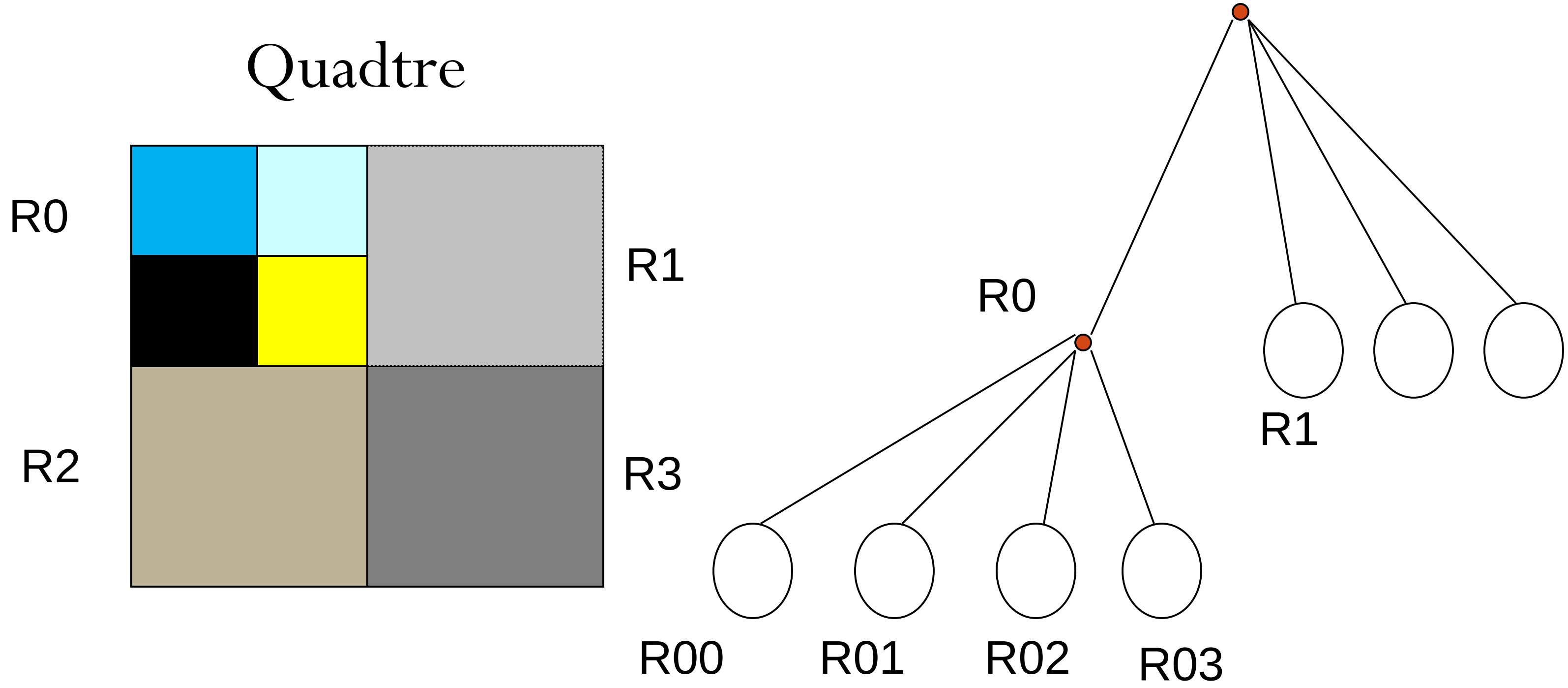
- If the original image has size M x N,
- Then $M = 2^n$ and $N = 2^n$
- Procedure of splitting is recursive
- Produces an image representation that can be described by a tree/subtree with four nodes
- Tree is called a Quadtree

Region Splitting: Example

Quadtre



Region Splitting: Example



Region Splitting Example

Minimum size after splitting 2x2

10	11	10	11	15	16	15	15
10	10	10	11	15	15	16	15
11	11	10	11	16	15	15	15
11	10	11	10	16	15	16	15
17	18	17	18	9	8	3	4
17	18	18	17	9	8	3	4
18	17	18	17	12	11	21	21
17	18	17	18	11	11	20	20

Condition:

Pixel difference between two pixels (Separation Threshold) <=2

All the pixels do not satisfy the condition

Region Splitting Example

Minimum size after splitting 2x2

	10	11	10	11	15	16	15	15	
	10	10	10	11	15	15	16	15	
R_0	11	11	10	11	16	15	15	15	R_1
	11	10	11	10	16	15	16	15	
	17	18	17	18	9	8	3	4	
R_2	17	18	18	17	9	8	3	4	R_3
	18	17	18	17	12	11	21	21	
	17	18	17	18	11	11	20	20	

Condition:

Pixel difference between two pixels (Separation Threshold) <=2

Pixels of R_3 do not satisfy the condition

Region Splitting Example

Minimum size after splitting 2x2

10	11	10	11	15	16	15	15
10	10	10	11	15	15	16	15
11	11	10	11	16	15	15	15
11	10	11	10	16	15	16	15
17	18	17	18	9	8	3	4
17	18	18	17	9	8	3	4
18	17	18	17	12	11	21	41
17	18	17	18	11	11	10	20

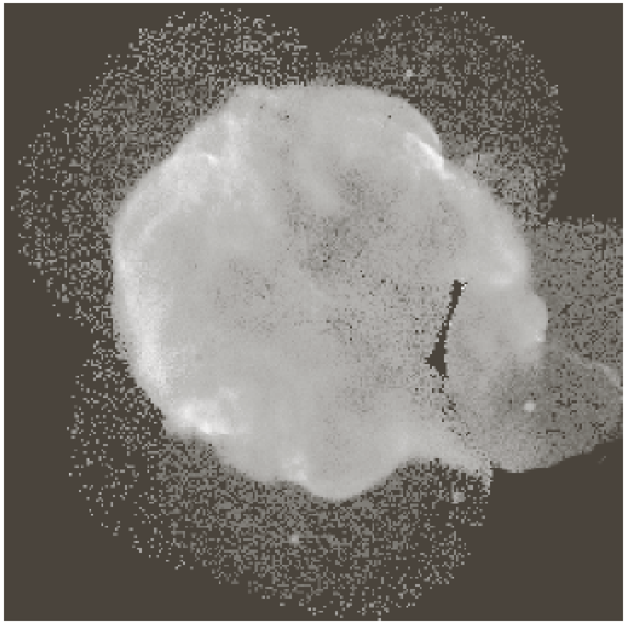
Condition:

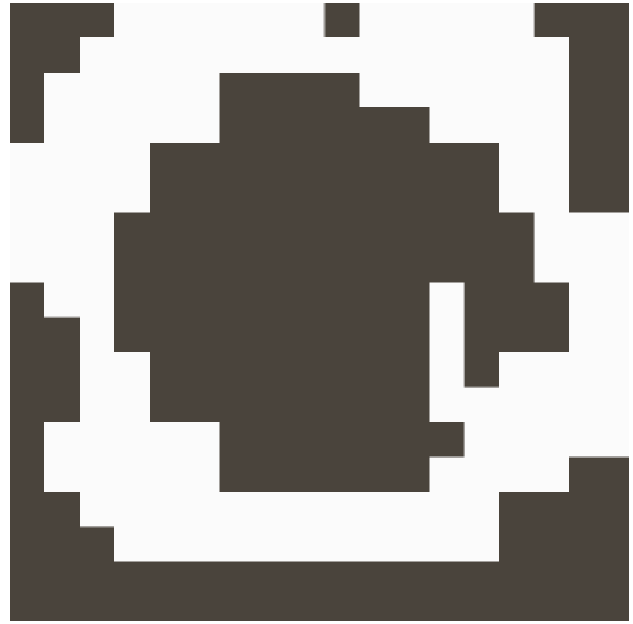
Pixel difference between two pixels (Separation Threshold) <=2

Results – Region Split

Splitting Ex: Condition: $\sigma > 10$ & $0 < m < 125$

σ is variance and m is mean

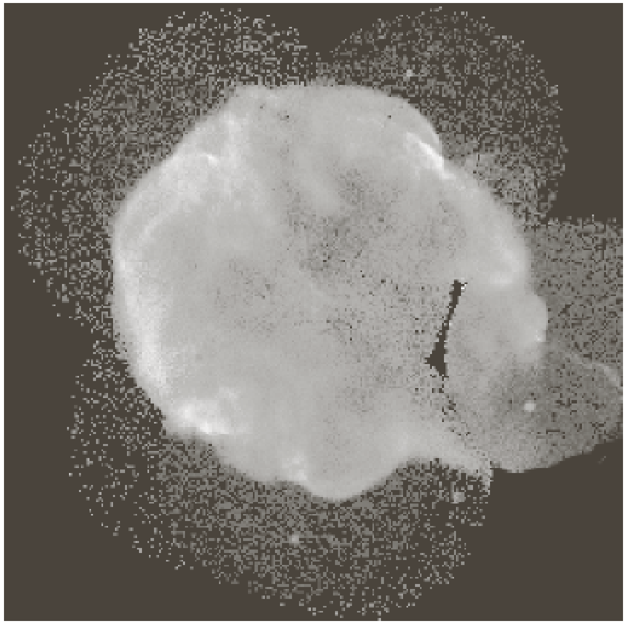


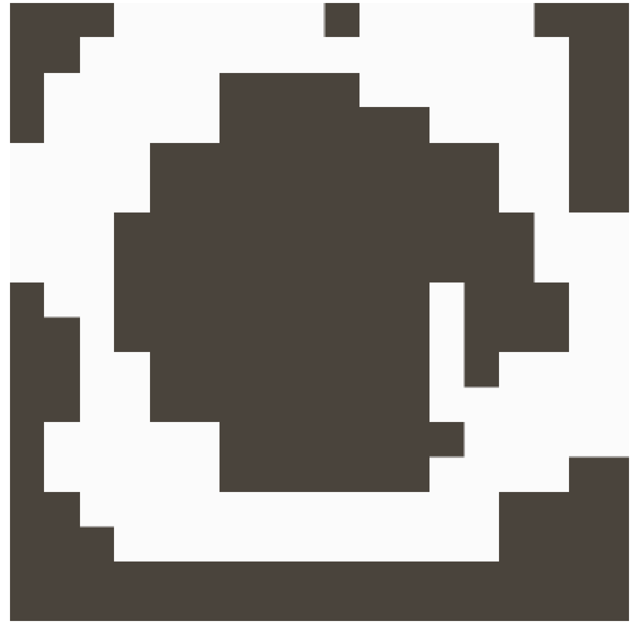


Min size 32x32

Splitting Ex: Condition: $\sigma > 10$ & $0 < m < 125$

σ is variance and m is mean



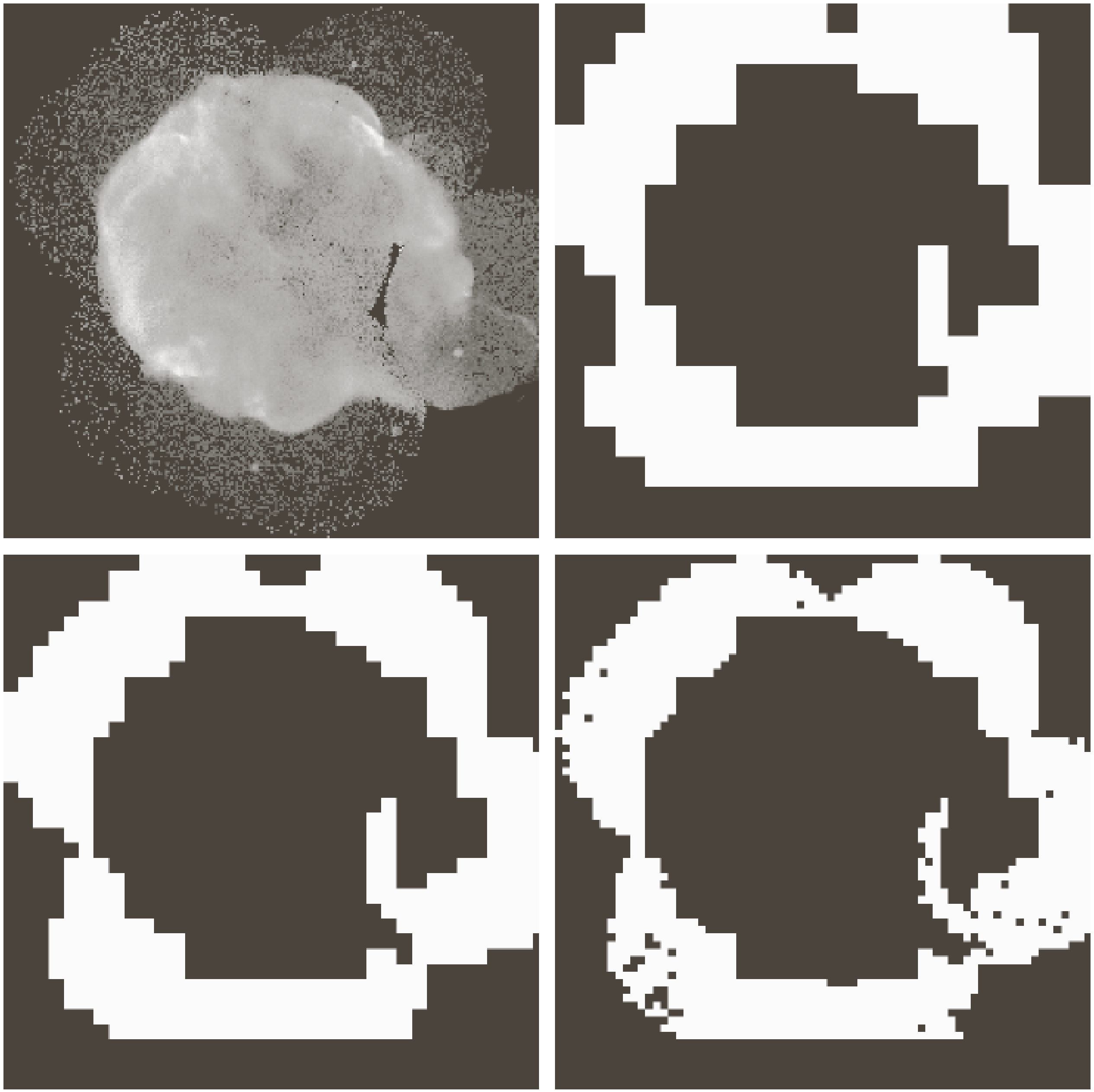


Min size 32x32

Min size 16x16

Splitting Ex: Condition: $\sigma > 10$ & $0 < m < 125$

σ is variance and m is mean



Min size 32x32

Min size 16x16

Min size 8x8

Region Merge Example

10	11	10	11	15	16	15	15
10	10	10	11	15	15	16	15
11	11	10	11	16	15	15	15
11	10	11	10	16	15	16	15
17	18	17	18	9	8	3	4
17	18	18	17	9	8	3	4
18	17	18	17	12	11	21	21
17	18	17	18	11	11	20	20

Minimum size 2x2

Condition:

Pixel difference between two pixels (Separation Threshold) <=2

- Divide image into subimage of minimum size

Region Merge Example

- Divide image into subimage of minimum size

10	**11**	**10**	**11**	**15**	**16**	**15**	**15**
10	10	10	11	15	15	16	15
11	11	10	11	16	15	15	15
11	10	11	10	16	15	16	15
17	18	17	18	9	8	3	4
17	18	18	17	9	8	3	4
18	17	18	17	12	11	21	21
17	18	17	18	11	11	20	20

Minimum size 2x2

Condition:

Pixel difference between two pixels (Separation Threshold) <=2

- Merge subimages if neighboring images satisfy the condition

Region Merge Example

- Divide image into subimage of minimum size
- Merge subimages if neighboring images satisfy the condition

10	**11**	**10**	**11**	**15**	**16**	**15**	**15**
10	10	10	11	15	15	16	15
11	11	10	11	16	15	15	15
11	10	11	10	16	15	16	15
17	18	17	18	9	8	3	4
17	18	18	17	9	8	3	4
18	17	18	17	12	11	21	21
17	18	17	18	11	11	20	20

Condition:

Pixel difference between two pixels (Separation Threshold) <=2

Split and Merge

- Split image
- Merge segments with similar predicate

Split and Merge: Example

Image after splitting

10	**11**	**10**	**11**	**15**	**16**	**15**	**15**
10	10	10	11	15	15	16	15
11	11	10	11	16	15	15	15
11	10	11	10	16	15	16	15
17	18	17	18	9	8	3	4
17	18	18	17	9	8	3	4
18	17	18	17	10	9	21	21
17	18	17	18	8	10	20	20

Condition:
Pixel difference between two consecutive pixels (Separation Threshold) <=2

Split and Merge: Example

Image after splitting followed by merging

10	**11**	**10**	**11**	**15**	**16**	**15**	**15**
10	10	10	11	15	15	16	15
11	11	10	11	16	15	15	15
11	10	11	10	16	15	16	15
17	18	17	18	9	8	3	4
17	18	18	17	9	8	3	4
18	17	18	17	10	9	21	21
17	18	17	18	8	10	20	20

Condition:
Pixel difference between two consecutive pixels (Separation Threshold) <=2

Split and Merge

Split

Original image

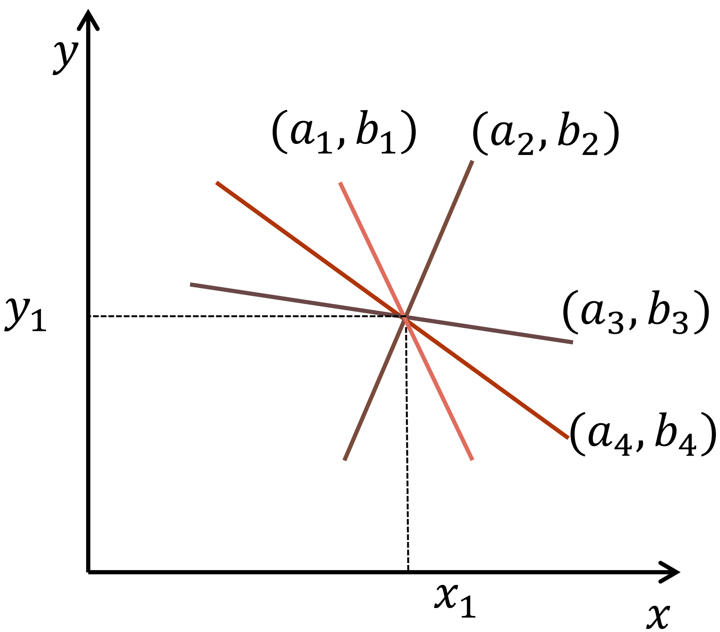
Split and Merge

Hough transform

- Useful for detecting curves or lines
- Relatively unaffected by gaps in lines and noise
- Given a set of edge points, Hough transform checks if these points lie on a straight line
- If yes, it draws a line joining all these points

Image with edge points

Hough transform to find lines



Slope: a
Intercept: b

- Consider a point (x_1, y_1)
- Equation of line passing through this point is

$$y_1 = ax_1 + b$$

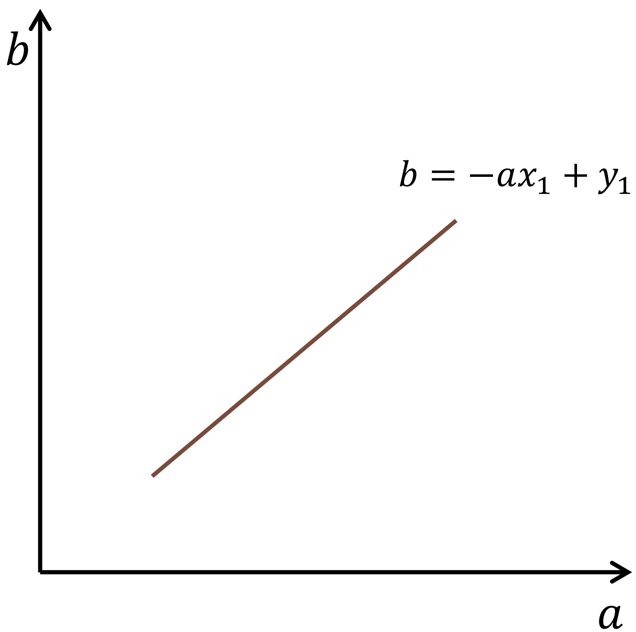
(slope intercept form)

- Using this equation and by varying value of $a \ and \ b$, infinite number of lines pass though this point (x_1, y_1)

Hough transform to find lines

- Equation of line passing through point (x_1, y_1) $y_1 = ax_1 + b$
- Rewriting above equation as $b = -ax_1 + y_1$
- This is equation of a line in ab plane

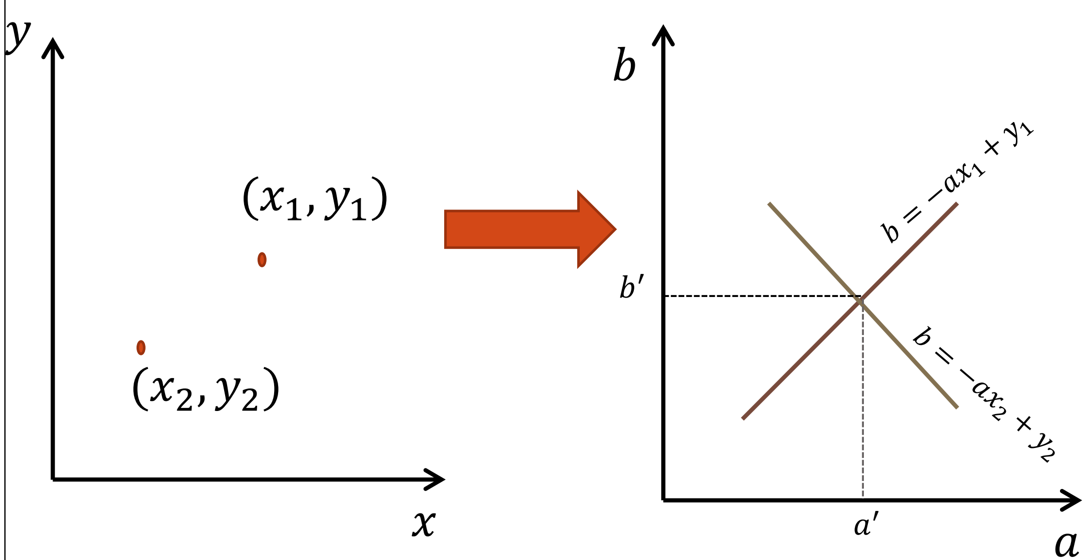
Hough transform to find lines



- Equation of line passing through point (x_1, y_1) $y_1 = ax_1 + b$
- Rewriting above equation as $b = -ax_1 + y_1$
- This is equation of a line in ab plane

A point in xy plane becomes a line in ab plane (Hough space)

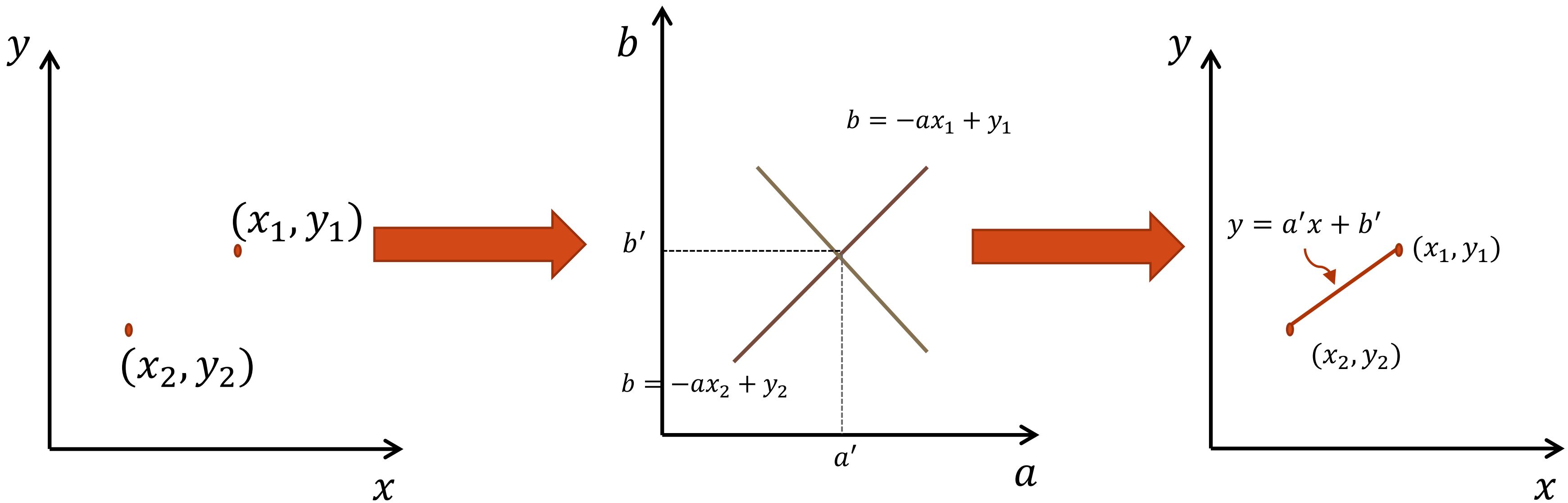
Hough transform to find lines



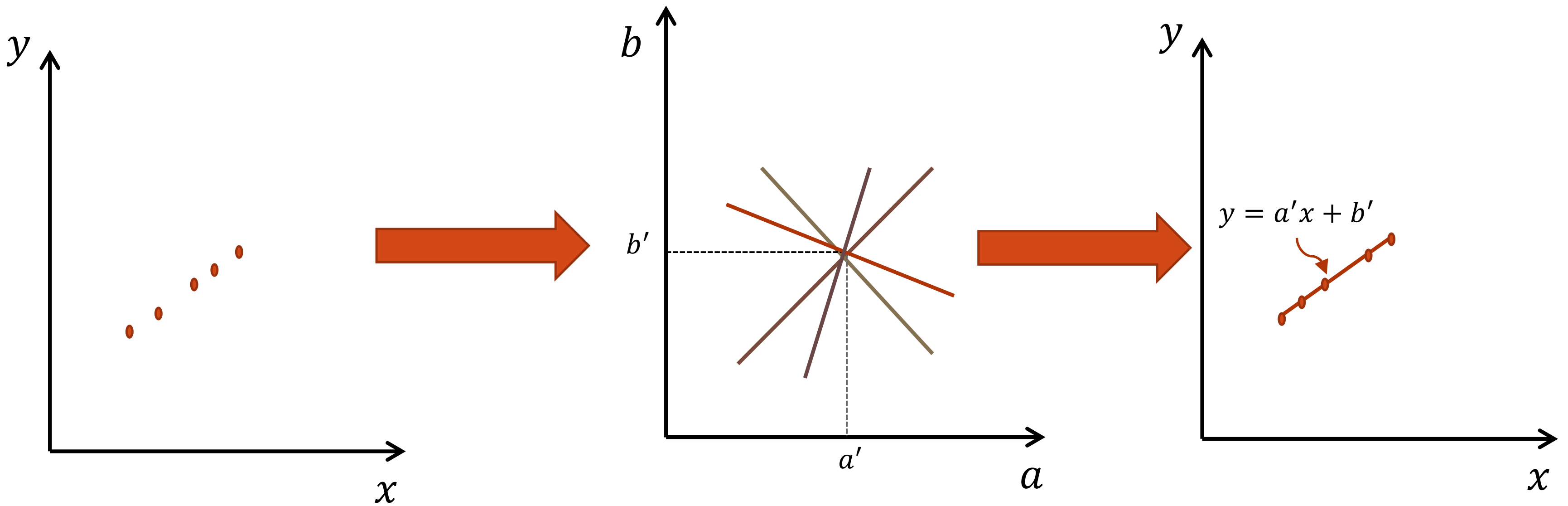
- Consider another point (x_2, y_2) in xy plane
- Line is $y_2 = ax_2 + b$
- Or, $b = -ax_2 + y_2$
- This is another line in $ab\ plane$
- These two lines intersect in ab plane if they are a part of straight line in xy plane
- Let the point of intersection in ab plane be (a', b')
- Line in slope intercept form is

  $y = a'x + b'$
- This line passes through points (x_1, y_1) and (x_2, y_2)

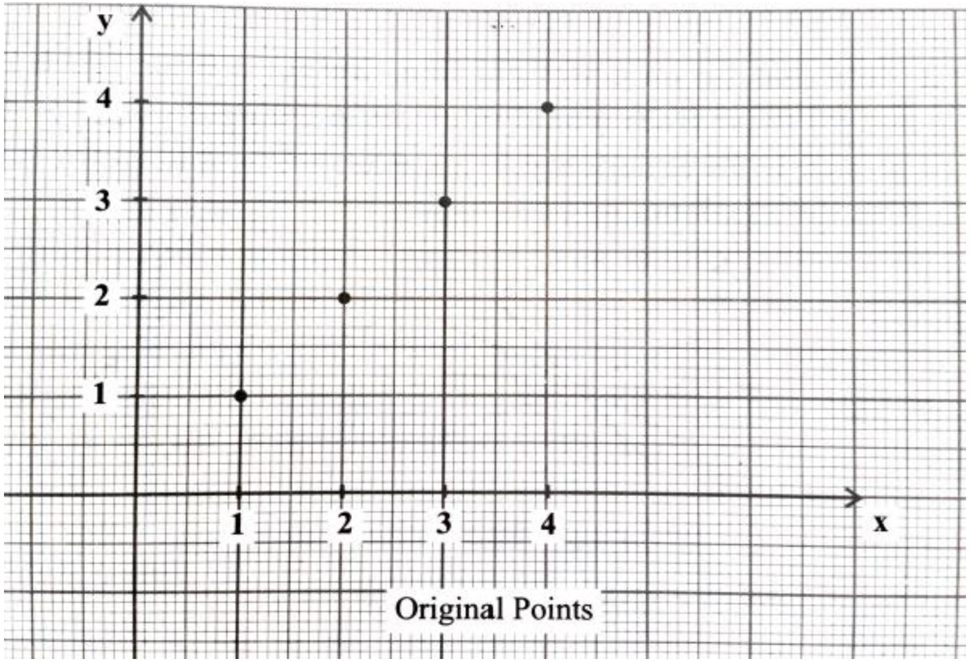
Hough transform to find lines



Hough Transform Example



Hough transform example



- Given four points in xy plane with the following coordinates (1,1), (2,2), (3,3), (4,4).
- Use Hough transform to join them

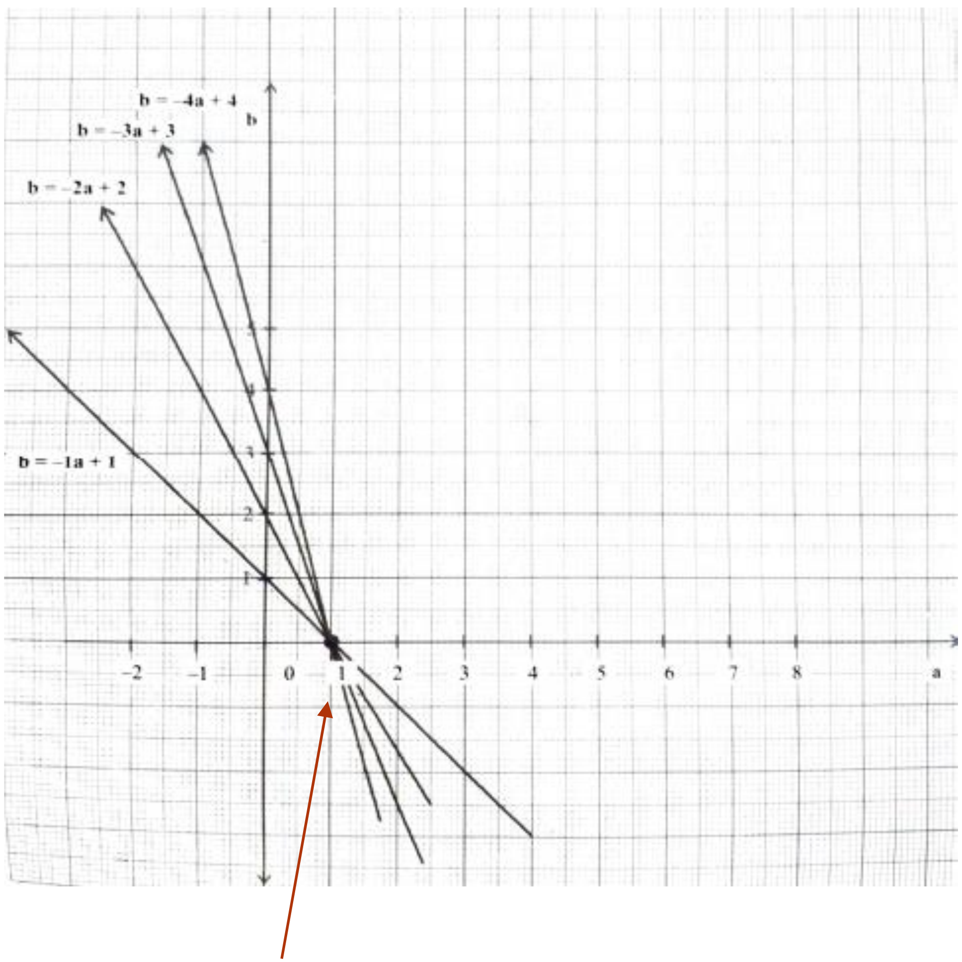
Hough transform example

- $b = -ax + y$
- Inserting values of x and y in the above equation
- Each line represents a point in xy plane

x	y
1	1
2	2
3	3
4	4

Hough transform example

ab plane

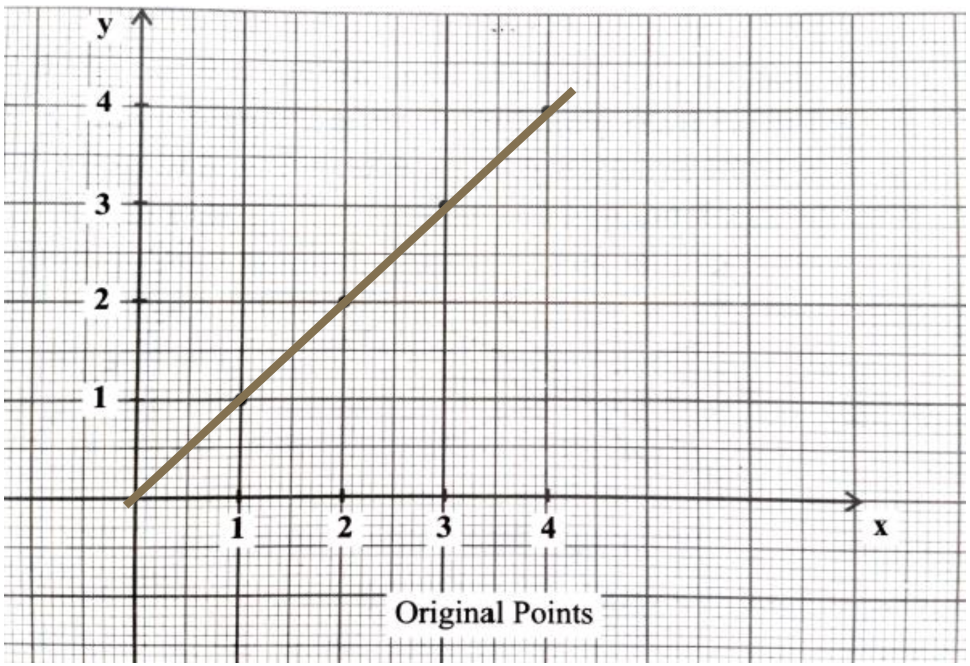


Intersecting point (1,0)

- $b = -ax + y$
- Inserting values of x and y in the above equation
- Each line represents a point in xy plane

x	y	$b = -ax + y$	
1	1	$b = -a + 1$	(1)
2	2	$b = -2a + 2$	(2)
3	3	$b = -3a + 3$	(3)
4	4	$b = -4a + 4$	(4)

Hough transform example



- Single intersection point which has coordinates (1,0)
- Therefore, $a = 1, b = 0$
- Draw a line in xy plane using equation $y = ax + b$
- Thus, $y = 1.x + 0$

  or $y = x$
- Line is passing through all the four points

Alternate Way to Represent a Line

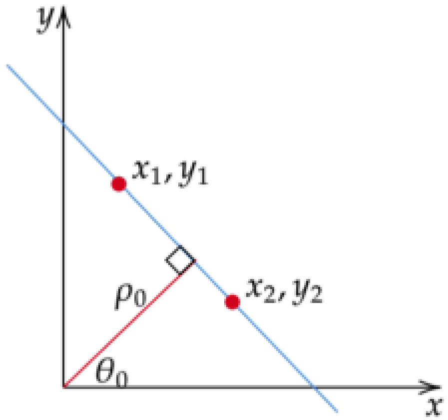
- $y = ax + b$ and the Hough Space with a and b
- Algorithm won't be able to detect vertical lines because the slope a is undefined (infinite) for vertical lines
- Computer would need an infinite amount of memory to represent all possible values of a.
- To avoid this issue, form of the normal line is

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

where ρ is the length of the normal line and θ is the angle between the normal line and the x axis

Alternate Way to Represent a Line

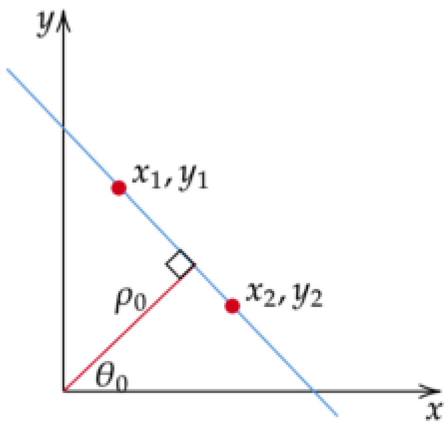
$\rho_0 = x \cos(\theta_0) + y \sin(\theta_0)$



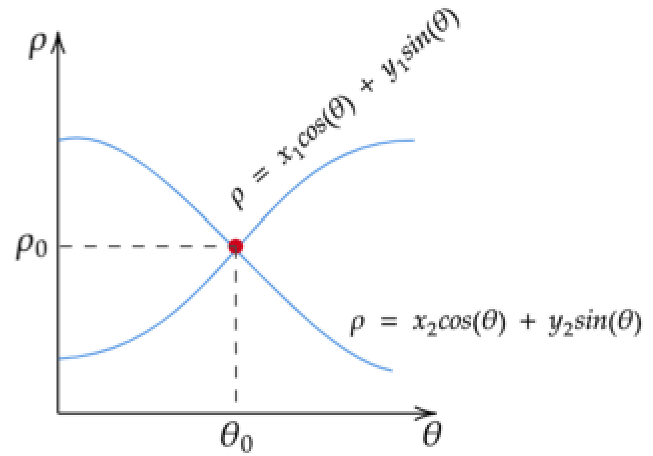
straight line passing
through two points

Alternate Way to Represent a Line

$\rho_0 = x\ cos(\theta_0) + y\ sin(\theta_0)$



straight line passing
through two points



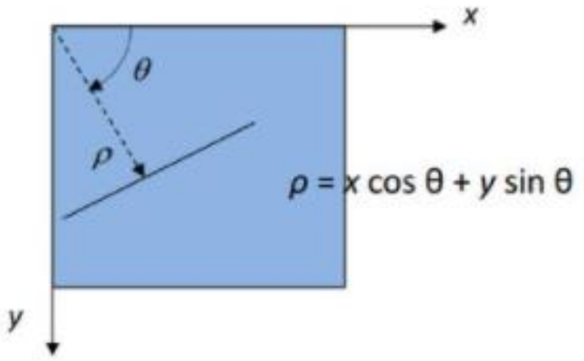
Hough Space

- Hough space for (x_1, y_2) and (x_1, y_2) or more
- Range of θ can be (0 to pi)
- More intersections represent more points of a line
- Thus a line can be detected by finding the number of intersections between curves
- In general, a threshold of the minimum number of intersections is used to detect a line

Polar Representation of a line

$\rho_0 = x \cos(\theta_0) + y \sin(\theta_0)$

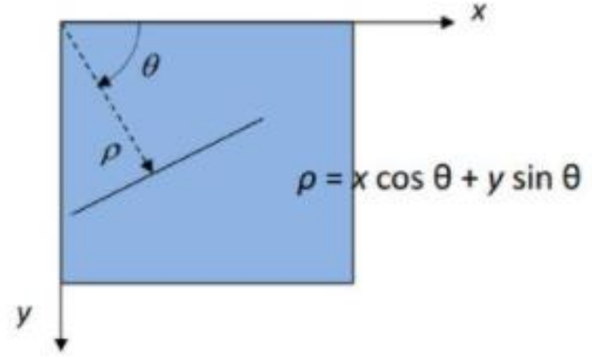
Angle range is -90^0 to $+90^0$
Rho range is $-\rho_{max}$ to $+\rho_{max}$
ρ_{max} is maximum possible distance
ρ_{max} is diagonal distance of image



Polar Representation of a line

$\rho_0 = x\cos(\theta_0) + y\sin(\theta_0)$

Angle range is -90^0 to $+90^0$
Rho range is $-\rho_{max}$ to $+\rho_{max}$
ρ_{max} is maximum possible distance
ρ_{max} is diagonal distance of image

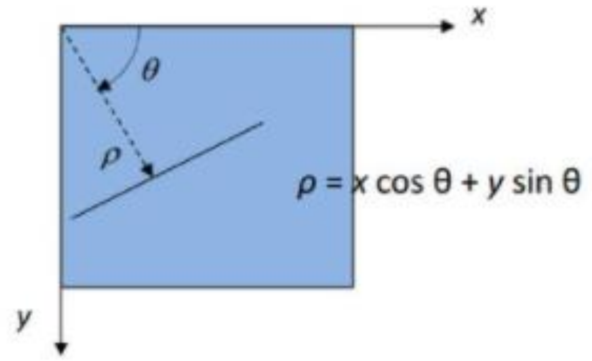


- Example of a point at (x,y) = (50,100)

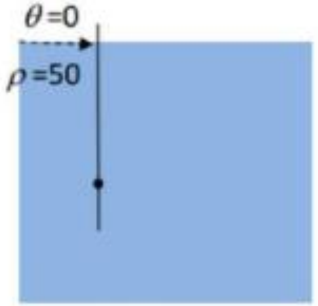
Polar Representation of a line

$\rho_0 = x\cos(\theta_0) + y\sin(\theta_0)$

Angle range is -90^0 to $+90^0$
Rho range is $-\rho_{max}$ to $+\rho_{max}$
ρ_{max} is maximum possible distance
ρ_{max} is diagonal distance of image



- Example of a point at (x,y) = (50,100)

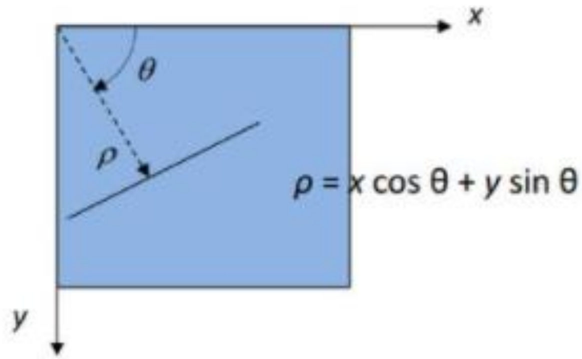


Lines passing point (50, 100)

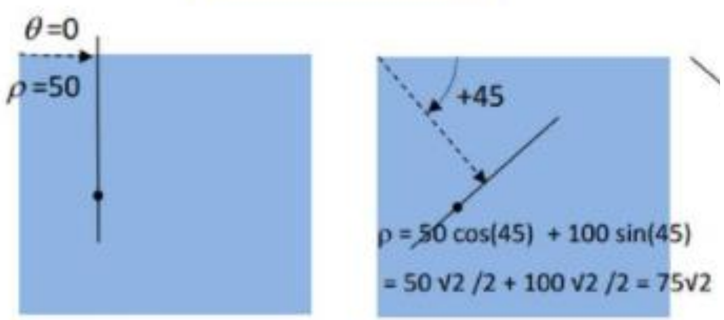
Polar Representation of a line

$\rho_0 = x \cos(\theta_0) + y \sin(\theta_0)$

Angle range is -90^0 to $+90^0$
Rho range is $-\rho_{max}$ to $+\rho_{max}$
ρ_{max} is maximum possible distance
ρ_{max} is diagonal distance of image



- Example of a point at (x,y) = (50,100)

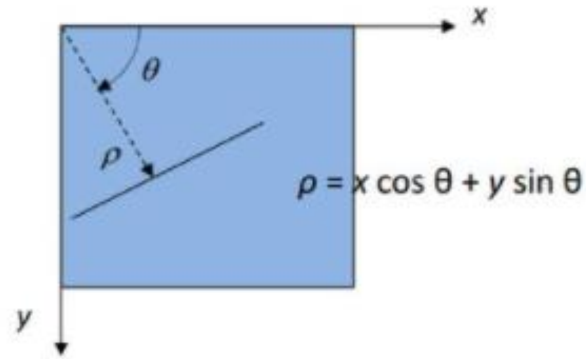


Lines passing point (50, 100)

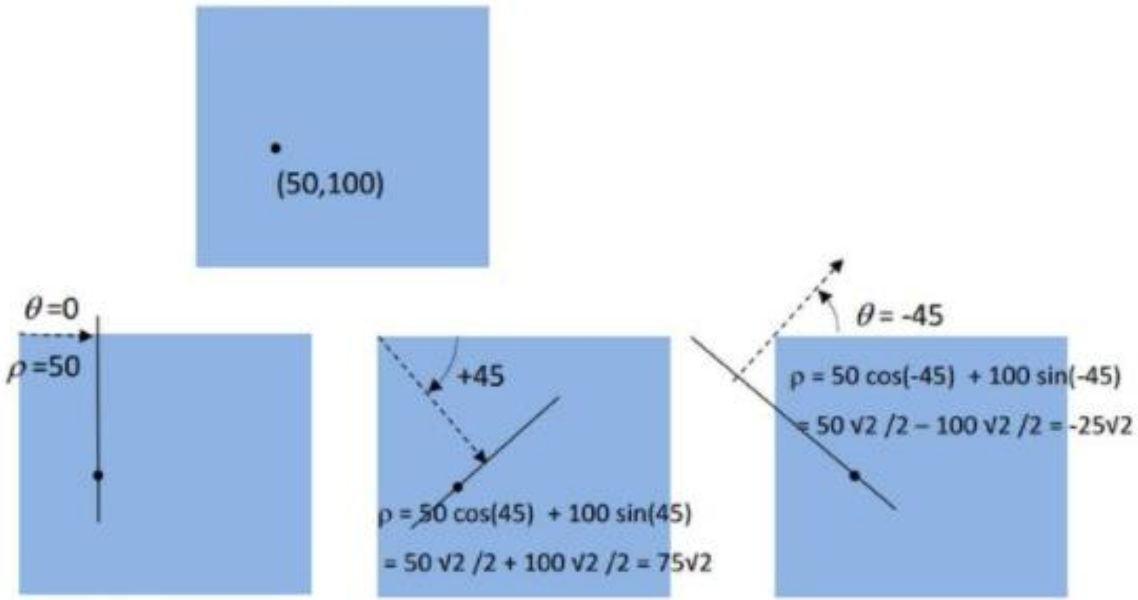
Polar Representation of a line

$\rho_0 = x\ cos(\theta_0) + y\ sin(\theta_0)$

Angle range is -90^0 to $+90^0$
Rho range is $-\rho_{max}$ to $+\rho_{max}$
ρ_{max} is maximum possible distance
ρ_{max} is diagonal distance of image



- Example of a point at (x,y) = (50,100)

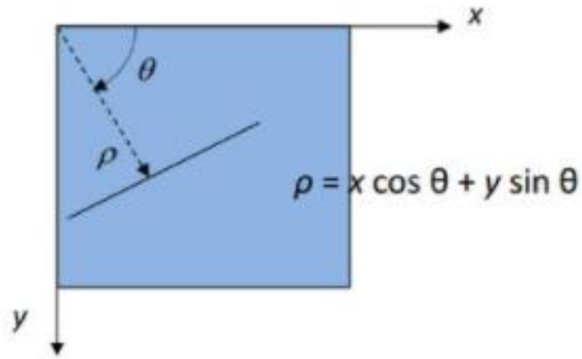


Lines passing point (50, 100)

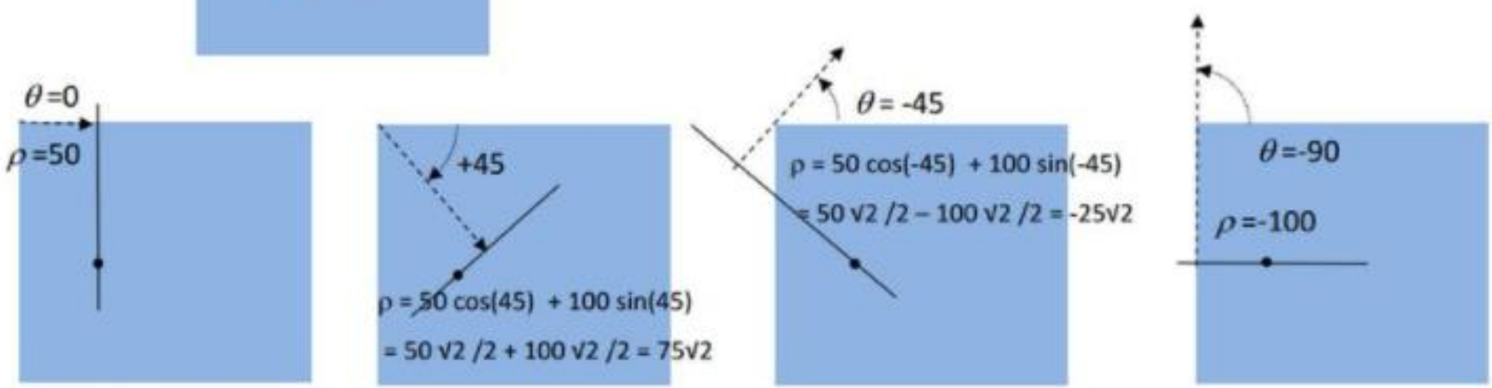
Polar Representation of a line

$\rho_0 = x\ cos(\theta_0) + y\ sin(\theta_0)$

Angle range is -90^0 to $+90^0$
Rho range is $-\rho_{max}$ to $+\rho_{max}$
ρ_{max} is maximum possible distance
ρ_{max} is diagonal distance of image



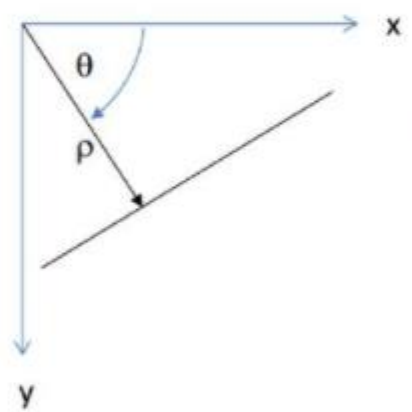
- Example of a point at (x,y) = (50,100)



Lines passing point (50, 100)

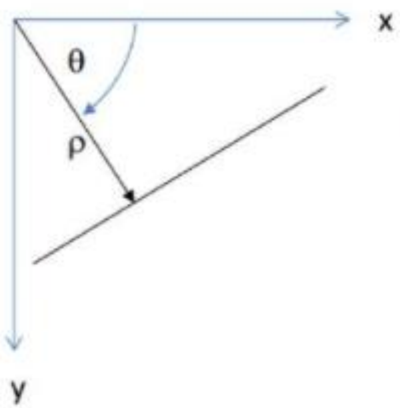
Accumulator for Polar Representation

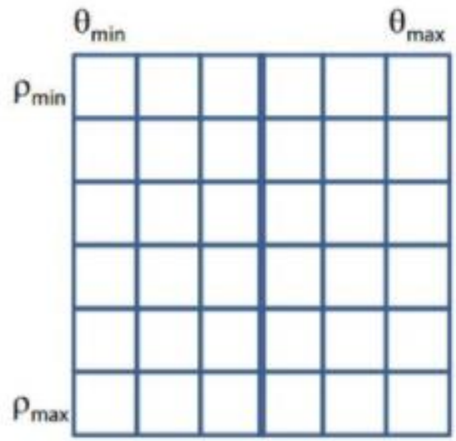
- $\rho = x \cos \theta + y \sin \theta$
  - Avoids infinite slope
  - Constant resolution



Accumulator for Polar Representation

- $\rho = x \cos \theta + y \sin \theta$
  - Avoids infinite slope
  - Constant resolution



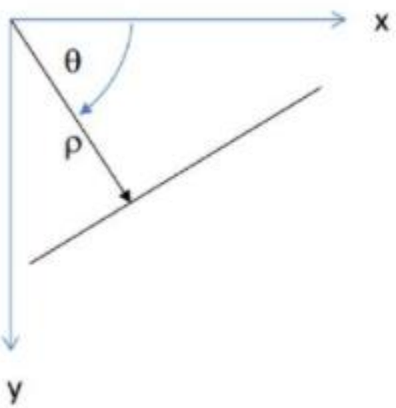


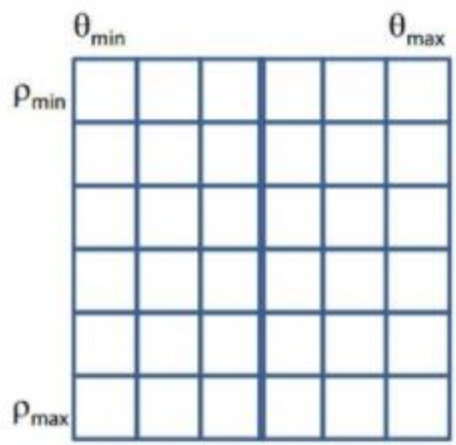
A(ρ,θ)

Accumulator for Polar Representation

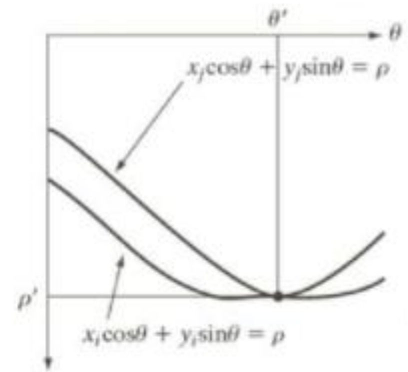
- $\rho = x \cos \theta + y \sin \theta$
  - Avoids infinite slope
  - Constant resolution

The parameter space transform of a point is a sinusoidal curve





A(ρ,θ)



Polar Representation of a line

- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image

  for θ = 0 to 180

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

  $H(\theta, \rho) = H(\theta, \rho) + 1$

  end

  end

θ →

ρ ↓	0^0	45^0	90^0	135^0	180^0
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Polar Representation of a line

- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image

  for θ = 0 to 180

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

  $H(\theta, \rho) = H(\theta, \rho) + 1$

  end

  end
- Find the value, s of (θ, ρ) where H(θ, ρ) is a local maximum
- If s > threshold, at (θ_o, ρ_0)
- Detected line in the image is given by

  $\rho_0 = x \cos\theta_o + y \sin\theta_o$
- slope, $a = \cos\theta_o / \sin\theta_o$
- intercept, $b = \rho / \sin\theta_o$
- $y = ax + b$

θ →

ρ ↓	0^0	45^0	90^0	135^0	180^0
0	0	3	0	1	1
1	2	1	3	0	0
2	1	3	6	0	0
3	1	2	2	2	1
4	2	4	0	0	2
5	1	8	1	0	1

Threshold = 5

Detects lines