Edge Linking using Hough Transform

Gonzalez, Rafael C. *Digital image processing*. Pearson education, 2009.

Introduction

- Work Experience:
  - June 2019-June 2020: Visiting Faculty, Dept. of CSE, IIT Jammu
  - July 2020-present: DST Inspire Faculty, Dept. of CSE, IIT Jammu
  - Dec 2019, Visiting Researcher, National Institute of Informatics (NII), Tokyo, Japan
  - July 2021, PDF position at NII Tokyo, online-part-time mode
- Major Research Interest:
  - Image Processing, Computer Vision, Biometrics, Visual Cryptography
  - Data privacy, Reinforcement Learning, Blockchain
- Education:
  - Dual Degree (MTech and PhD), PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, 2013-2018
  - PhD Thesis: Biometric Template Protection Cancelable Biometrics and Visual Cryptography

Edge Detection

Edge is detected by finding discontinuities in intensity positions.

(a) Original Image

(b) Edge Map using LoG

(c) Edge Map using Canny

Edge Linking and Boundary Detection

- Ideally, edge detection should yield sets of pixels lying only on edges.
- Edges detected are not continuous due to:
    - Presence of noise
    - Spurious edge points
    - Non uniform illumination

- Edge detection -> Edge linking algorithms
- Assemble edge pixels into meaningful edges and/or region boundaries.

Edge Linking and Boundary Detection

- Basic approaches
  - Local Processing (requires knowledge about edge points in a local region, e.g. 3x3 neighborhood),
  - Global Processing (works with an entire edge image) via Hough Transform

Local Processing

- **Input is a binary edge detected image**
- Analyze each pixel in small neighborhood of (x, y) and find neighbors (x', y') similar to (x, y)
- Similarity is measured by
  - Strength of the response of gradient operators (∇)
  - Direction of the gradient (α)
- Points (x, y) and (x', y') are similar if
  - $\nabla f(x, y) - \nabla f(x', y') \leq T$ and
  - $\alpha(x, y) - \alpha(x', y') \leq A$

  where $(x', y') \in N_{(x,y)}$
- Similar points are linked to form edges.
- The above strategy is expensive.
- A record has to be kept of all linked points by, for example, assigning a different label to every set of linked points.

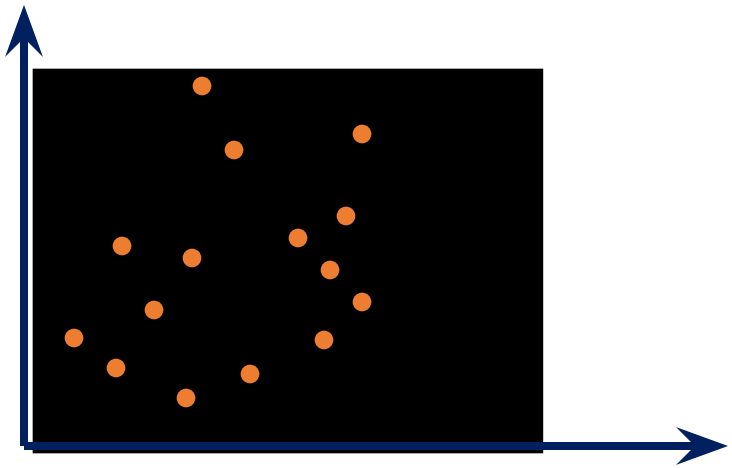
Binary Edge Map Image

Global Processing - Hough Transform

- Richard Duda and Peter Hart in 1972
- Developed an approach based on whether sets of pixels lie on curves of a specified shape.
- Once detected, these curves form the edges or region boundaries of interest.

Line Detection using Hough Transform

- Input to the Hough transform is a binary thesholded edge image
- We have 'n' edge pixels that partially define boundary of some object.

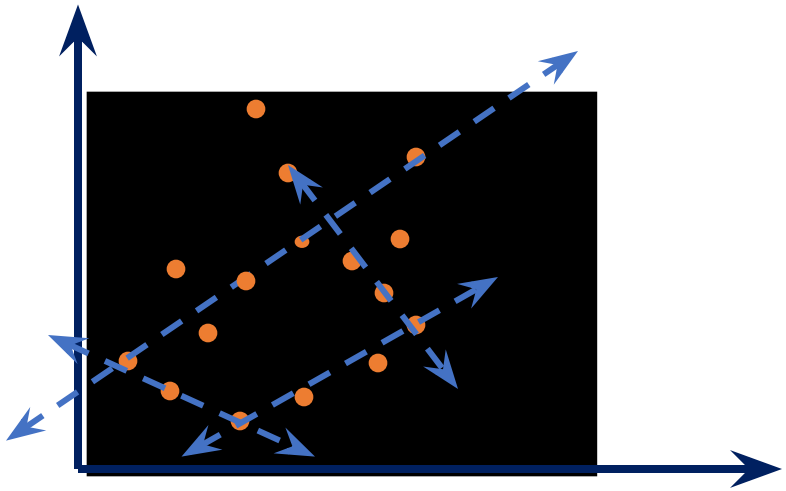


Binary Edge Map Image

- We wish to the find pixels that make up straight lines

Line Detection using Hough Transform

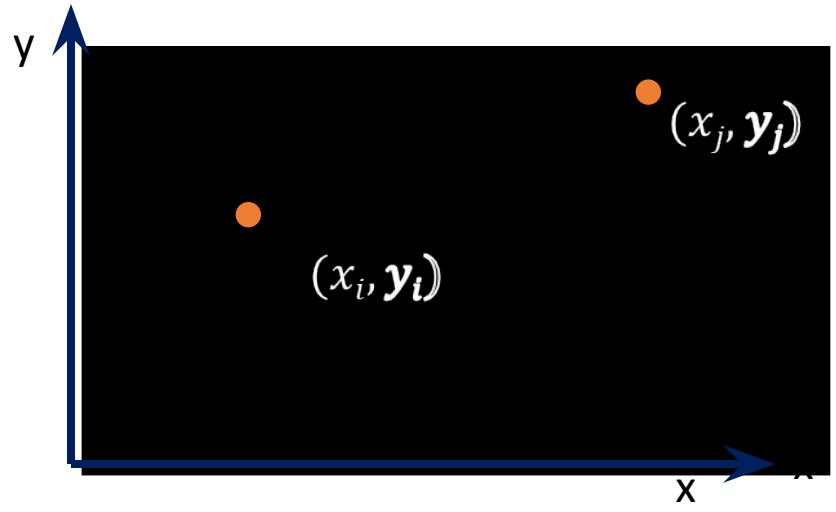
- Input to the Hough transform is a thesholded edge image
- We have 'n' edge pixels that partially define boundary of some object.



- We wish to the find pixels that make up straight lines

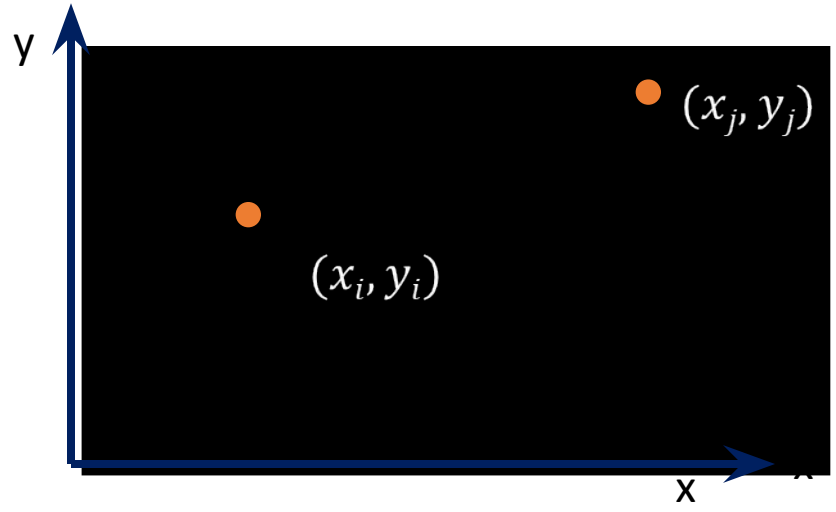
Line Detection using Hough Transform

- Transform coordinate space (x,y) to parameter space (m,c)



Line Detection using Hough Transform

- Transform coordinate space (x,y) to parameter space (m,c)



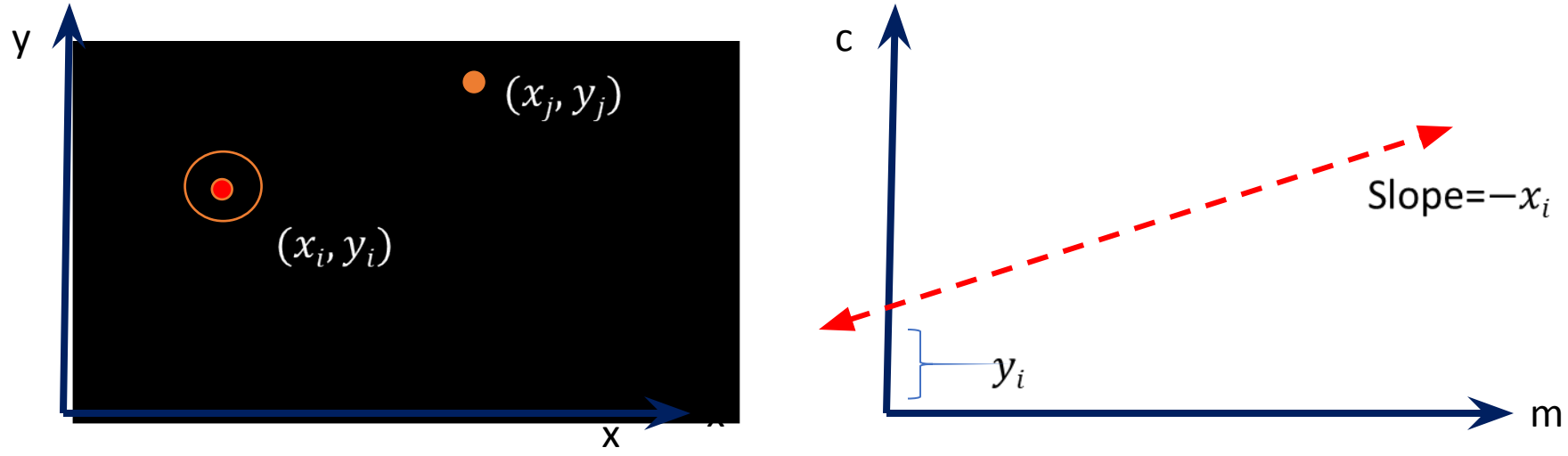
$$y_i = \boldsymbol{m}x_i + \boldsymbol{c} \quad \longleftrightarrow \quad c = -\boldsymbol{x_i}m + \boldsymbol{y_i}$$

Slope **m**, intercept ***c***

Slope $\mathbf{x_i}$, intercept $(\mathbf{-y_i})$

Line Detection using Hough Transform

- Transform coordinate space (x,y) to parameter space (m,c)

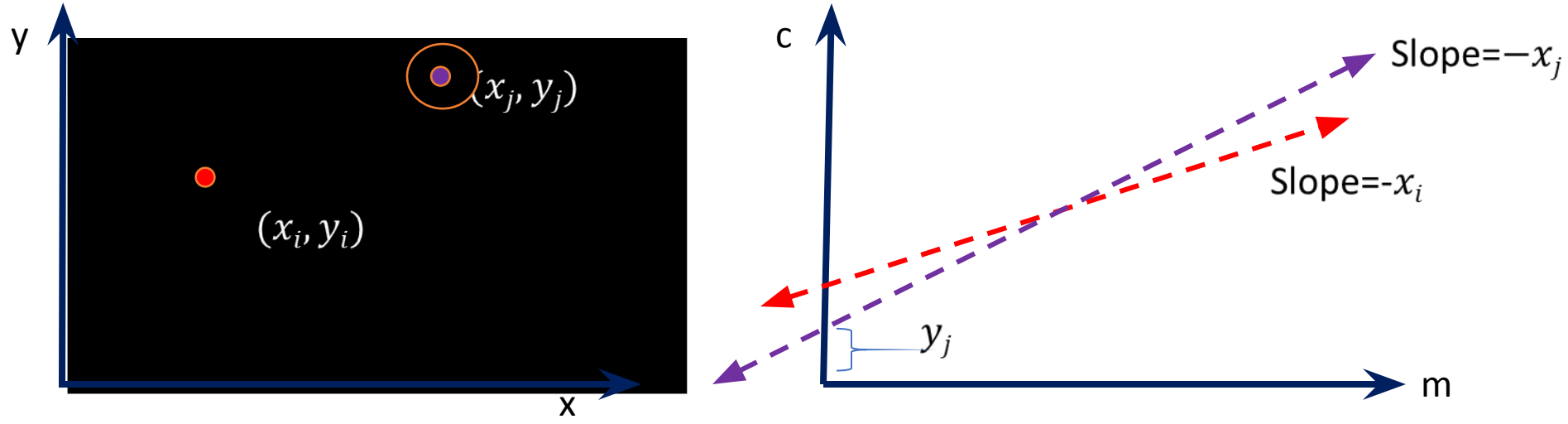


$$y_i = mx_i + c \quad \longleftrightarrow \quad c = -\boldsymbol{x_i}m + \boldsymbol{y_i}$$

- Observation 1: Point in (x, y) space becomes line in (m, c)space

Line Detection using Hough Transform

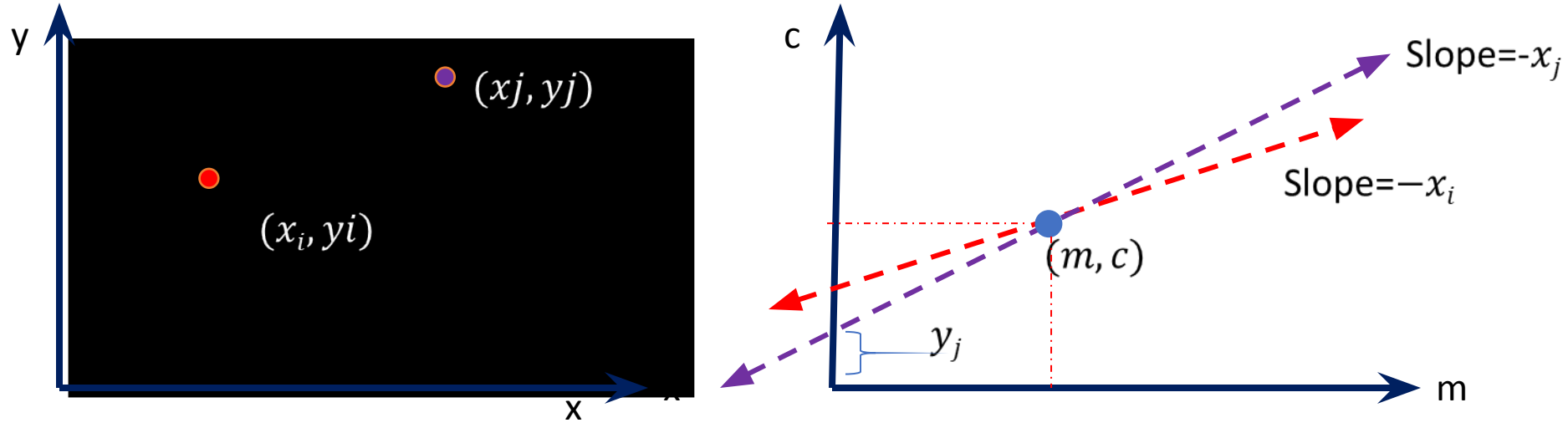
- Transform coordinate space (x,y) to parameter space (m,c)



$$y_j = \boldsymbol{m}x_j + \boldsymbol{c} \quad \longleftrightarrow \quad c = -\boldsymbol{x_j}m + \boldsymbol{y_j}$$

Line Detection using Hough Transform

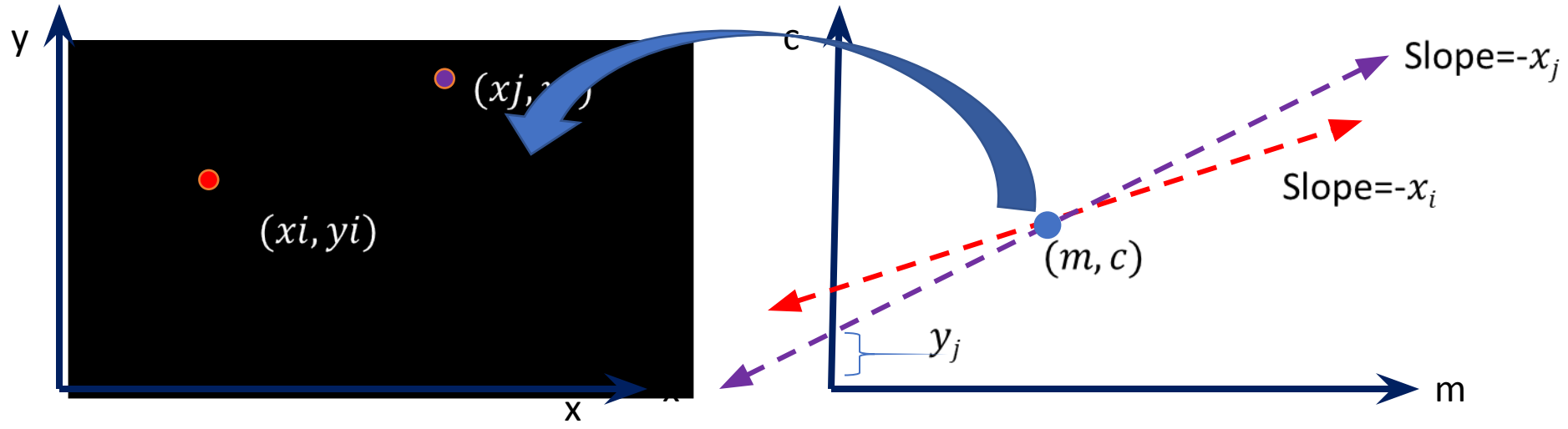
- Transform coordinate space (x,y) to parameter space (m,c)



$$c = -x_j m + y_j$$
$$c = -x_i m + y_i$$

Line Detection using Hough Transform

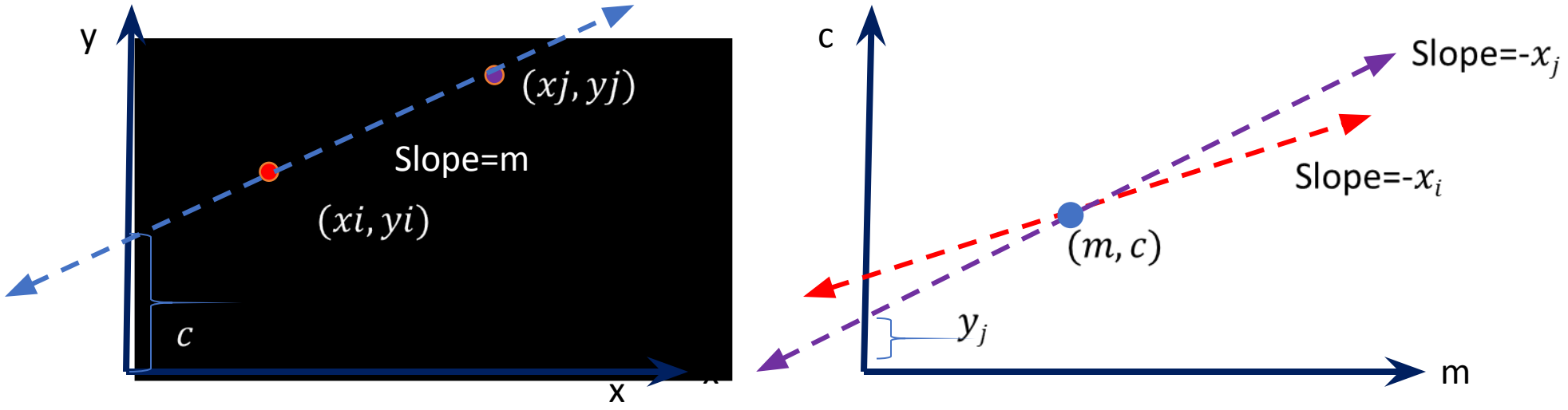
- Transform coordinate space (x,y) to parameter space (m,c)



$$c = -x_j m + y_j$$
$$c = -x_i m + y_i$$

Line Detection using Hough Transform

- Transform coordinate space (x,y) to parameter space (m,c)

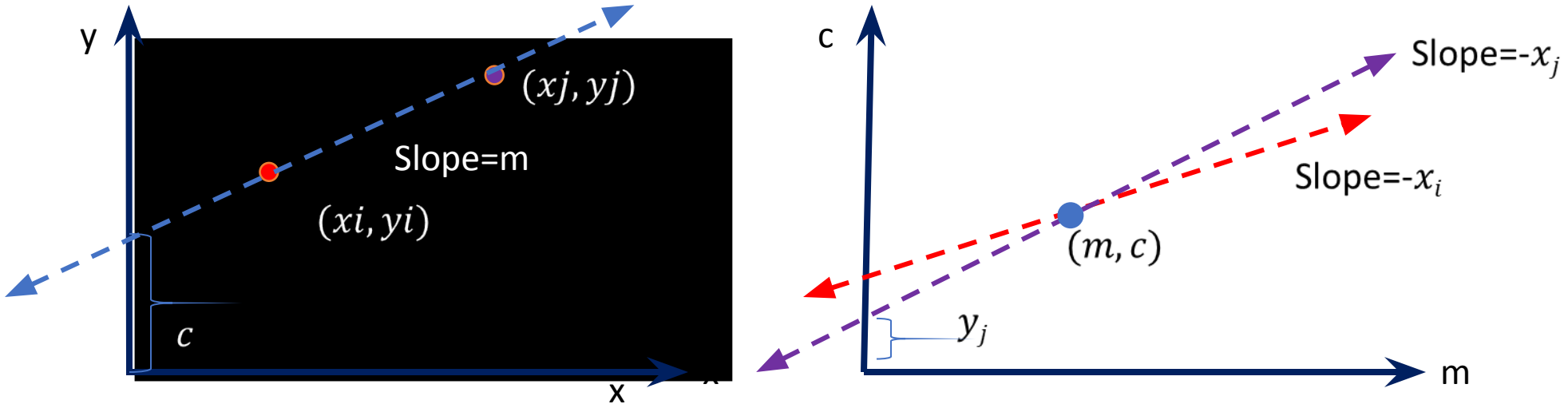


$$y_j = mx_j + c \qquad c = -x_j m + y_j$$
$$y_i = mx_i + c \qquad c = -x_i m + y_i$$

- Observation 2: Line in (x, y) space becomes point in (m, c) space

Line Detection using Hough Transform

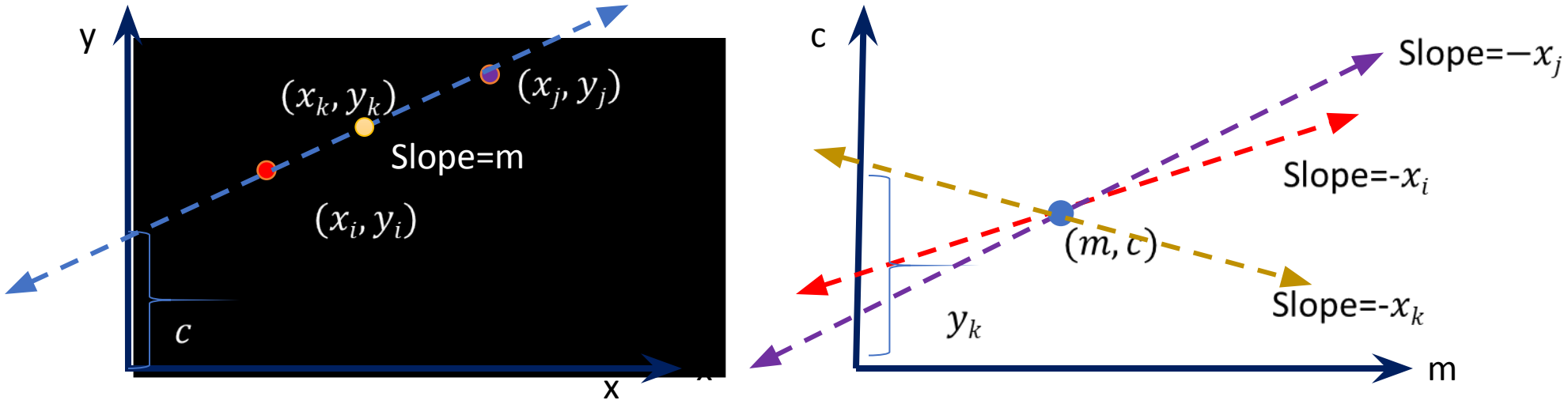
- Transform coordinate space (x,y) to parameter space (m,c)



- Observation 1: Point in (x,y) space ↔ line in (m,c) space
- Observation 2: Line in (x,y) space ↔ point in (m, c) space
- Observation 3: No. of points in same line in (x,y) space ↔ no. of intersecting lines in(m,c) space

Line Detection using Hough Transform

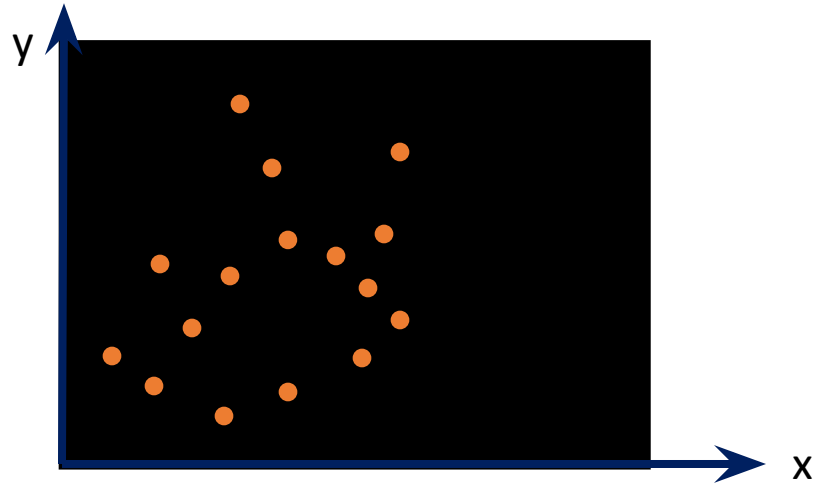
- Transform coordinate space (x,y) to parameter space (m,c)

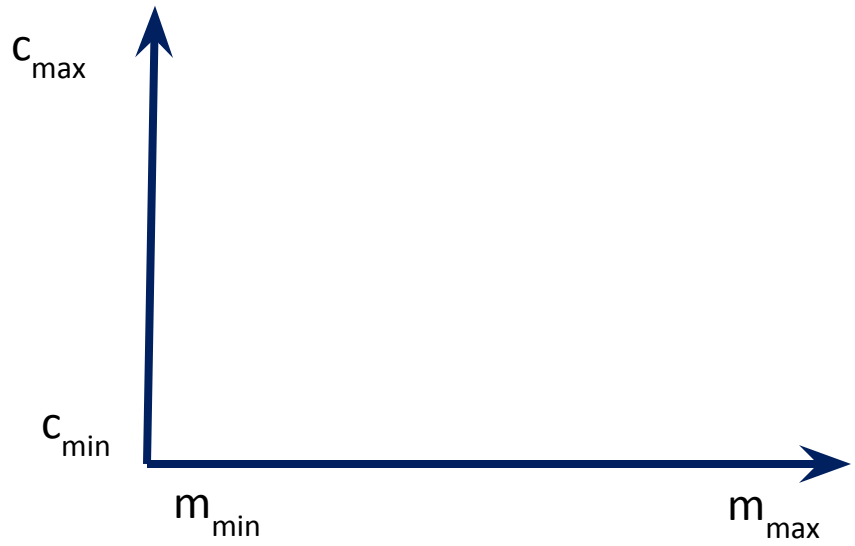


- Observation 1: Point in (x,y) space ↔ line in (m,c) space
- Observation 2: Line in (x,y) space ↔ point in (m, c) space
- Observation 3: No. of points in same line in (x,y) space ↔ no. of intersecting lines in(m,c) space

How is it useful?

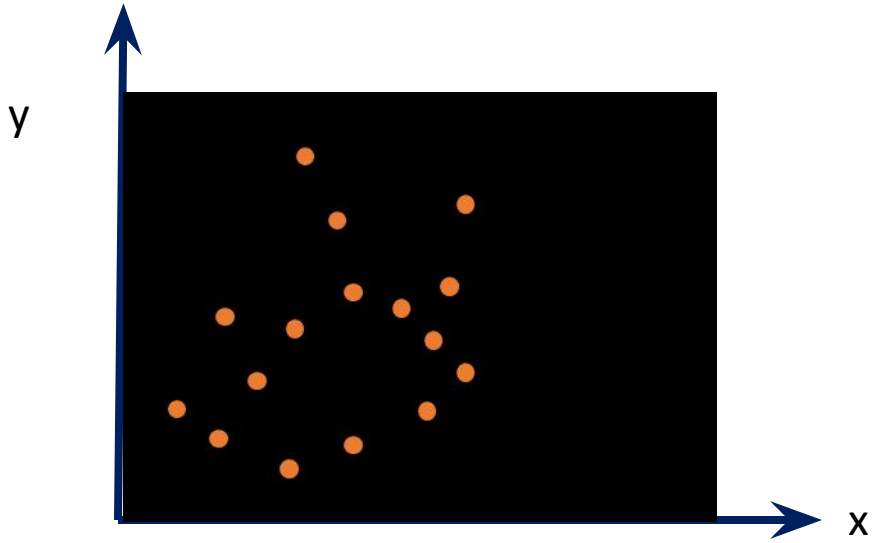
- Given a binary image I in (x,y) space

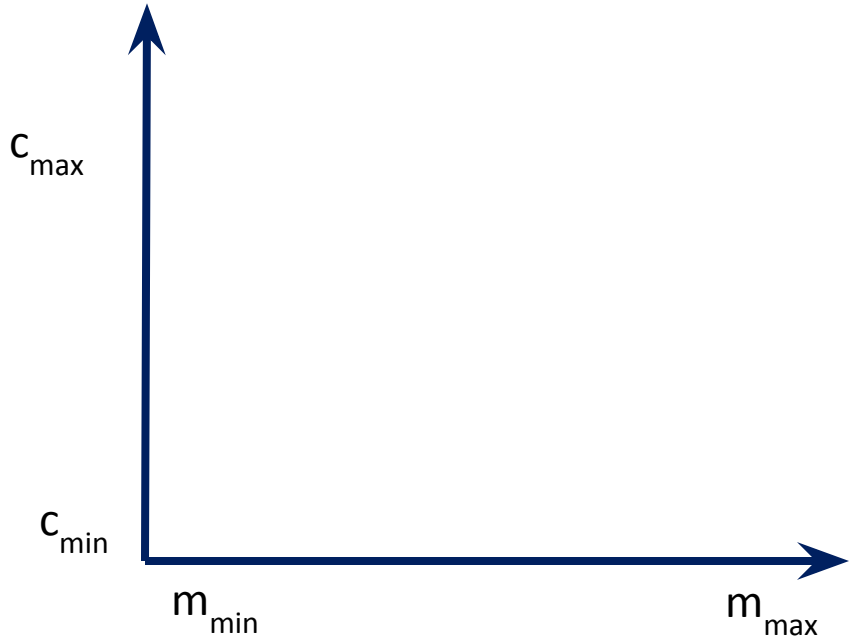




How is it useful?

- Given a binary image I in (x,y) space.

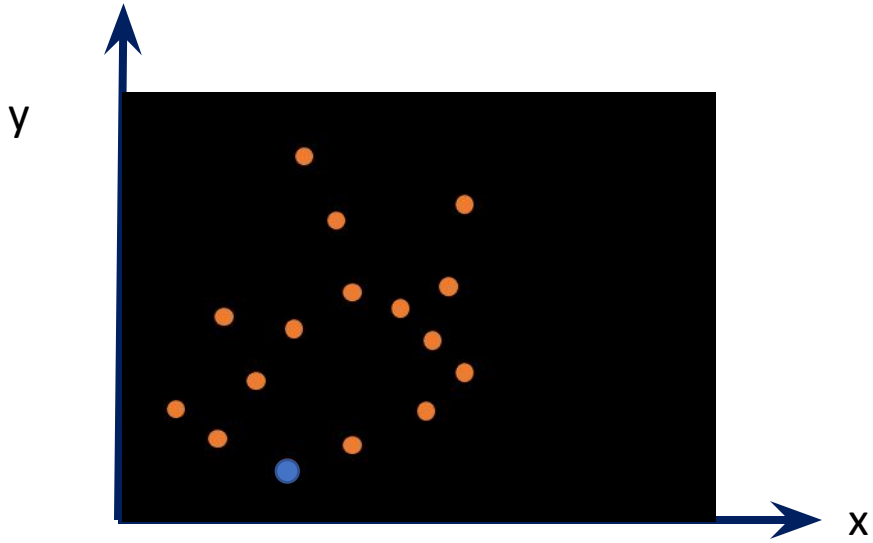




- Discretize and quantize (m,c) space to cells in range m_{min} - m_{max}, c_{min} - c_{max}

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
c_{min}	0	0	0	0	0	0	0
	m_{min}						m_{max}

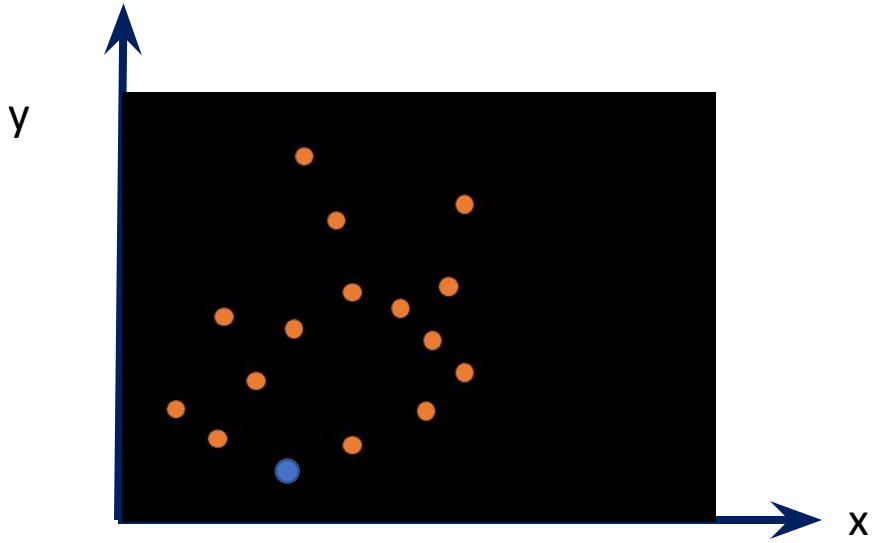
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i \boldsymbol{m_k} + (y_i)$, for all m_k, $\boldsymbol{m_{min} \leq m_k \leq m_{max}}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
c_{min}	1	0	0	0	0	0	0
	m_{min}						m_{max}

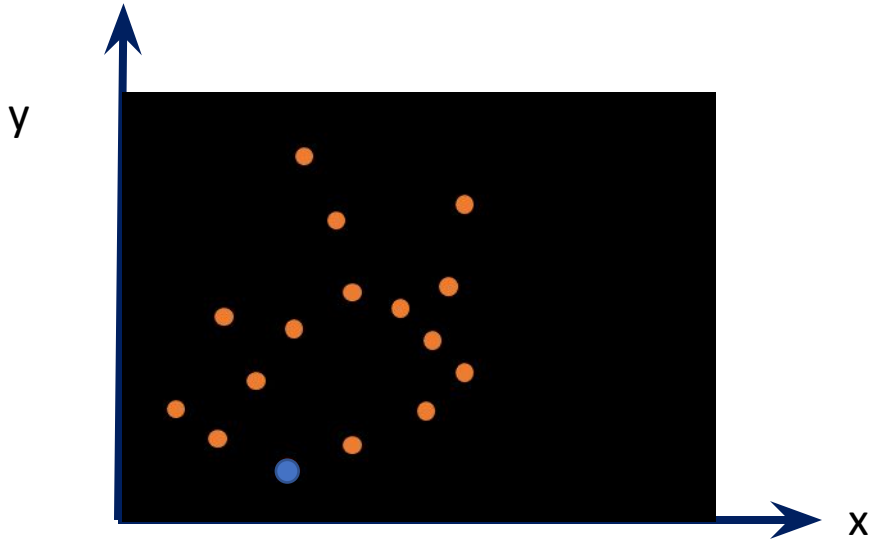
- Step 2: For each edge point (x_i, yi) ,

compute $\boldsymbol{c_k} = -x_i \boldsymbol{m_k} + (y_i)$, for all m_k, $\boldsymbol{m_{min} \leq m_k \leq m_{max}}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	1	0	0	0	0	0
c_{min}	1	0	0	0	0	0	0
	m_{min}						m_{max}

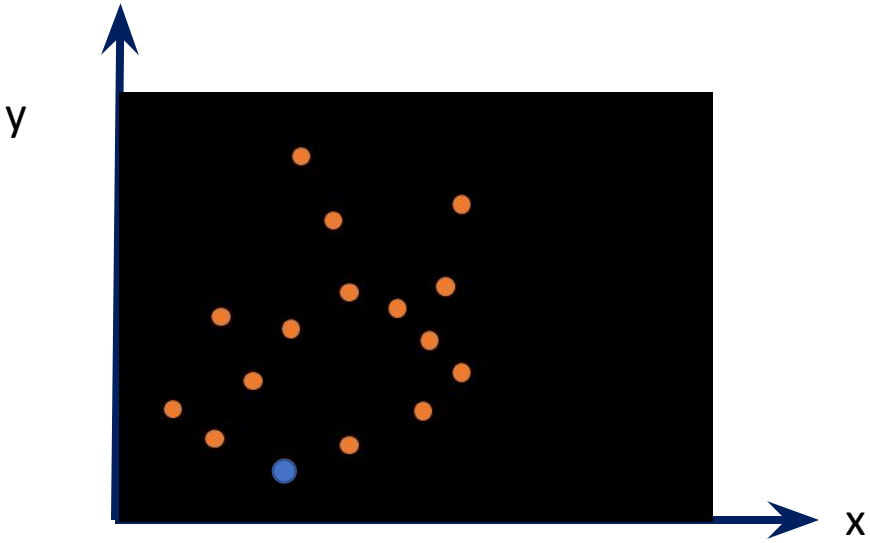
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + (y_i)$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	1	0	0	0
	0	0	1	0	0	0	0
	0	1	0	0	0	0	0
c_{min}	1	0	0	0	0	0	0
	m_{min}						m_{max}

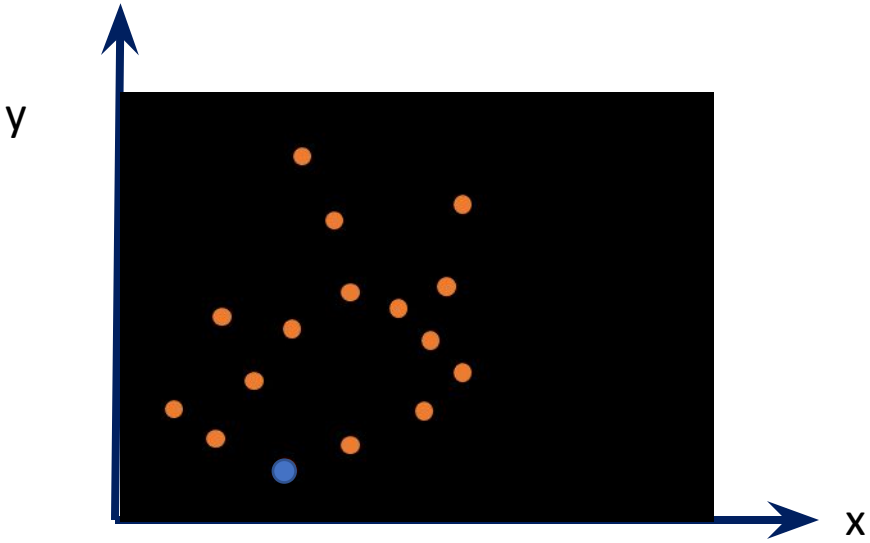
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	0	0	0	0	0	1	1
	0	0	0	0	1	0	0
	0	0	0	1	0	0	0
	0	0	1	0	0	0	0
	0	1	0	0	0	0	0
c_{min}	1	0	0	0	0	0	0
	m_{min}						m_{max}

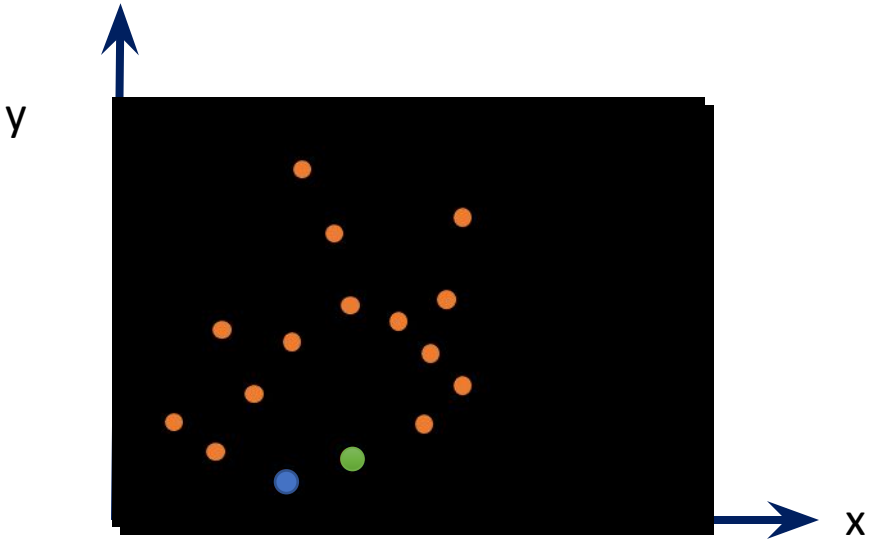
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}							
	0	0	0	0	0	1	1
	0	0	0	0	1	0	0
	0	0	0	1	0	0	0
	0	0	1	0	0	0	0
	0	1	0	0	0	0	0
c_{min}	1	0	0	0	0	0	0
	m_{min}						m_{max}

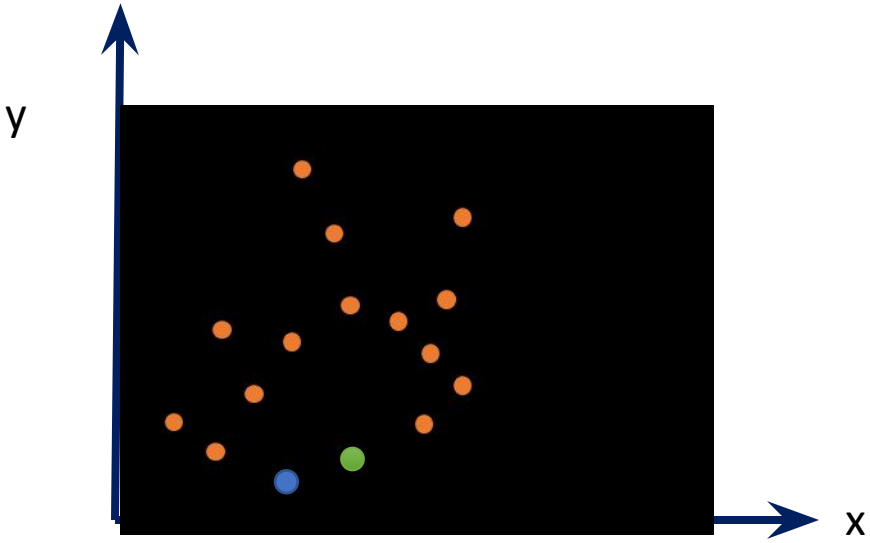
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}							
	1	0	0	0	0	1	1
	0	1	0	0	1	0	0
	0	0	0	1	0	0	0
	0	0	1	0	0	0	0
	0	1	0	0	0	0	0
c_{min}	1	0	0	0	0	0	0
	m_{min}						m_{max}

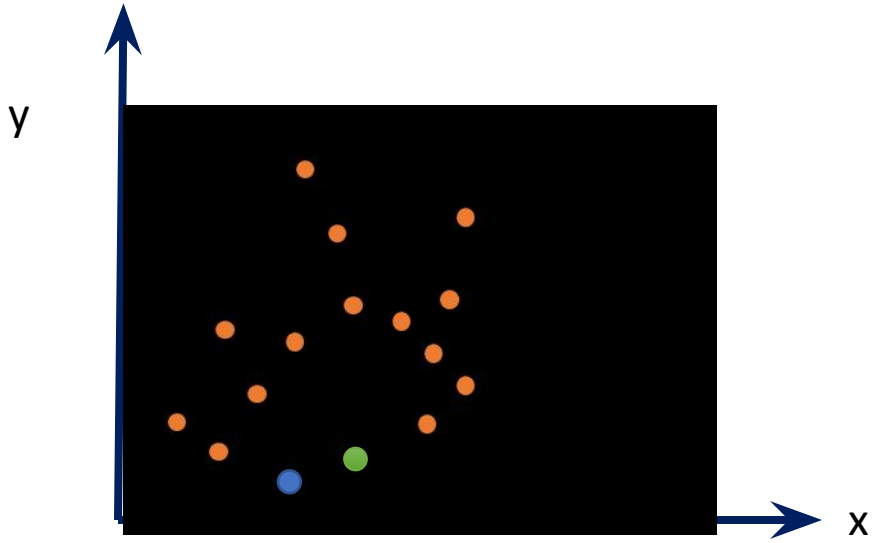
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	1	0	0	0	0	1	1
	0	1	0	0	1	0	0
	0	0	1	2	0	0	0
	0	0	1	0	0	0	0
	0	1	0	0	0	0	0
c_{min}	1	0	0	0	0	0	0
	m_{min}						m_{max}

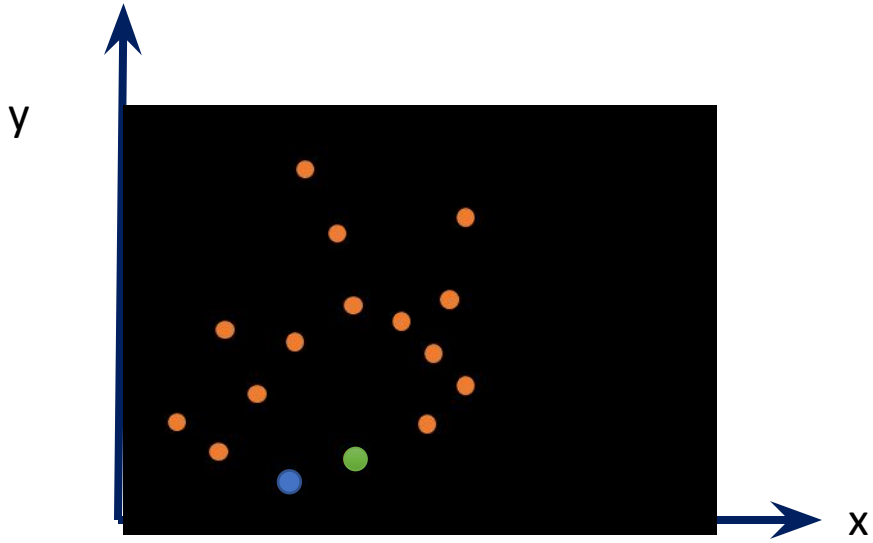
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	1	0	0	0	0	1	1
	0	1	0	0	1	0	0
	0	0	1	2	0	0	0
	0	0	1	0	1	0	0
	0	1	0	0	0	1	0
c_{min}	1	0	0	0	0	0	1
	m_{min}						m_{max}

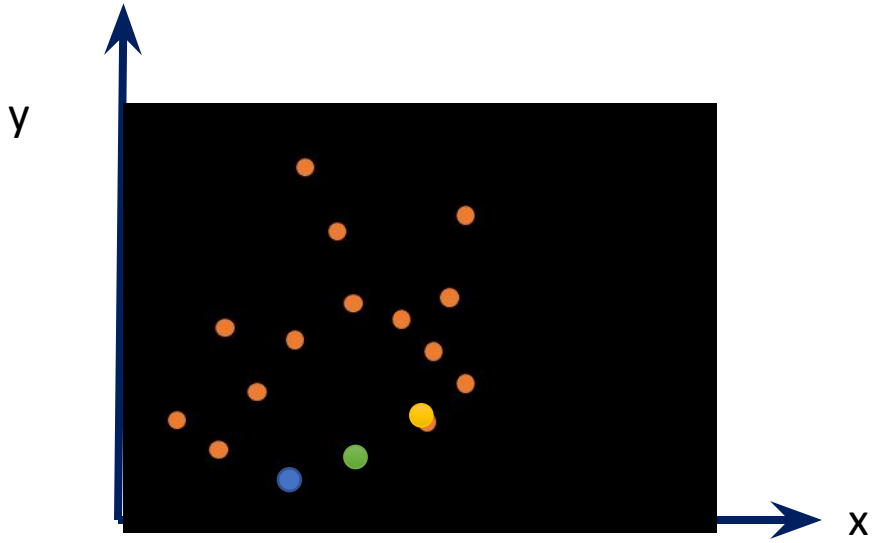
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}	1	0	0	0	0	1	1
	0	1	0	0	1	0	0
	0	0	1	2	0	0	0
	0	0	2	0	1	0	0
	1	2	0	0	0	1	0
c_{min}	1	0	0	0	0	0	1
	m_{min}						m_{max}

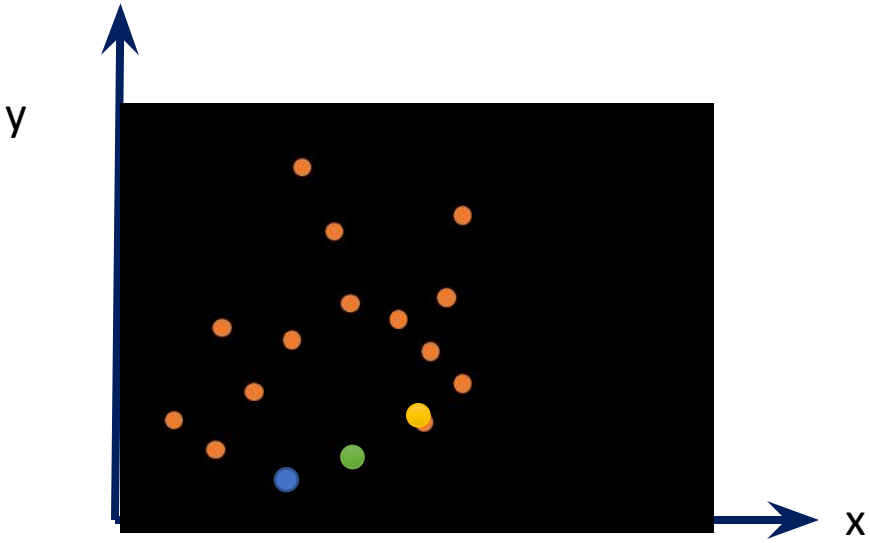
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}							
	1	0	0	0	0	1	2
	0	1	0	0	1	1	0
	0	0	1	3	1	0	0
	0	0	2	0	1	0	0
	1	2	0	0	0	1	0
c_{min}	1	0	0	0	0	0	1
	m_{min}						m_{max}

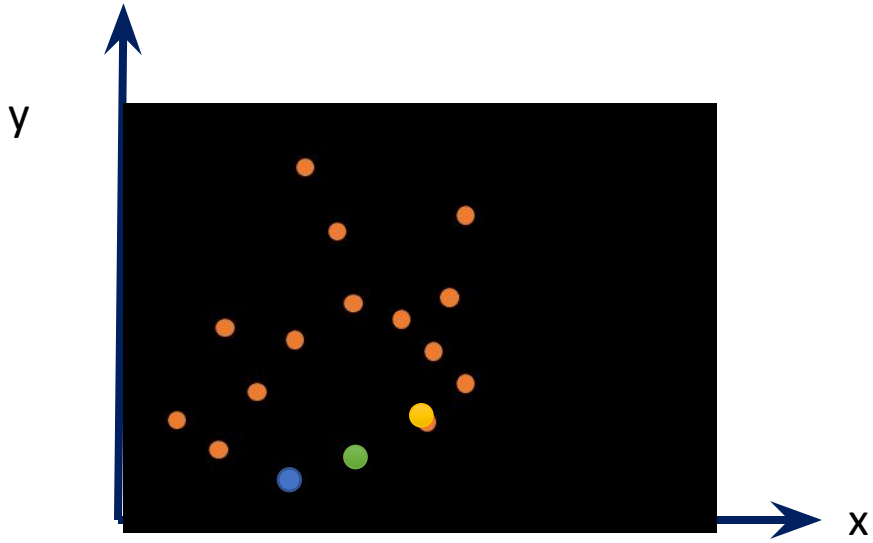
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{m_k}, Q_{c_k}) = A(P_{m_k}, Q_{c_k}) + 1$

How is it useful?

- Given a binary image I



$Accumulator$

c_{max}							
	1	0	0	0	0	1	2
	0	1	0	0	1	1	1
	0	0	1	4	3	2	1
	0	1	4	1	1	0	0
	2	3	0	0	0	1	0
c_{min}	2	0	0	0	0	0	1
	m_{min}						m_{max}

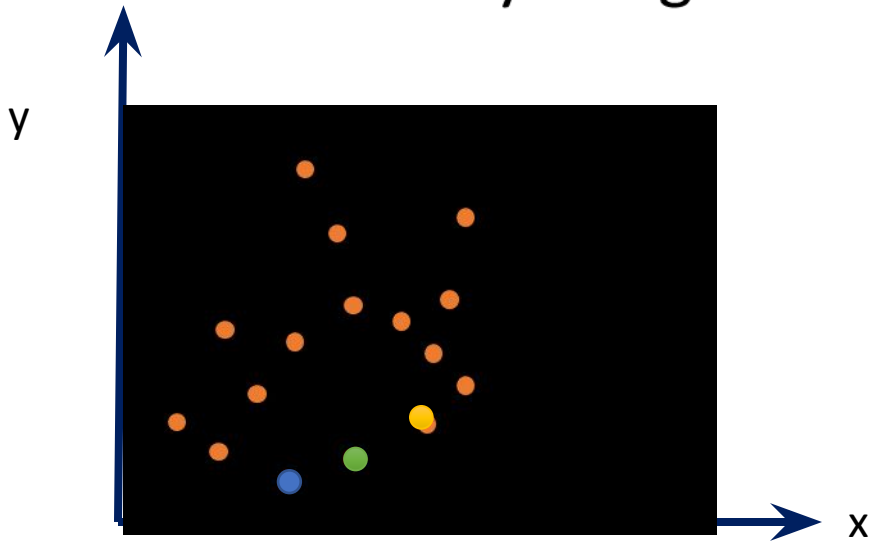
- Step 2: For each edge point (x_i, yi) ,

compute $c_k = -x_i m_k + y_i$, for all m_k, $m_{min} \leq m_k \leq m_{max}$

and increment $A(P_{mk}, Q_{ck}) = A(P_{mk}, Q_{ck}) + 1$

How is it useful?

- Given a binary image I



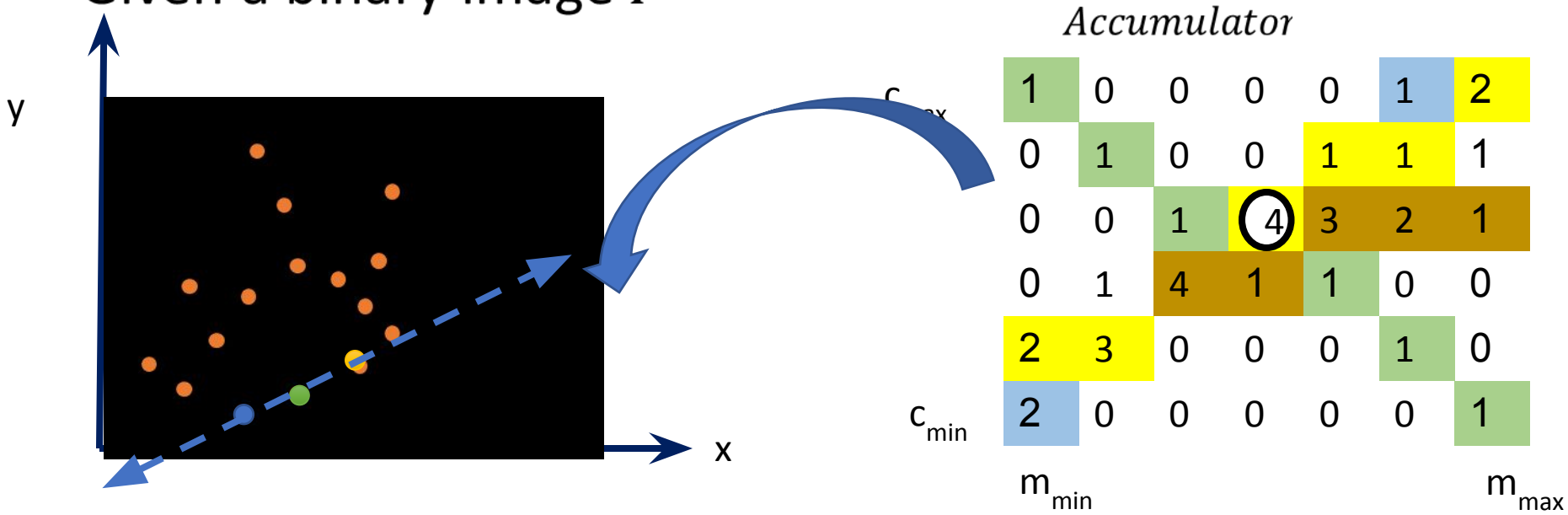
Accumulator

c_{max}	1	0	0	0	0	1	2
	0	1	0	0	1	1	1
	0	0	1	4	3	2	1
	0	1	4	1	1	0	0
	2	3	0	0	0	1	0
c_{min}	2	0	0	0	0	0	1
	m_{min}						m_{max}

- Step 3: Identify the cell having highest votes-> $A(P, Q)$

How is it useful?

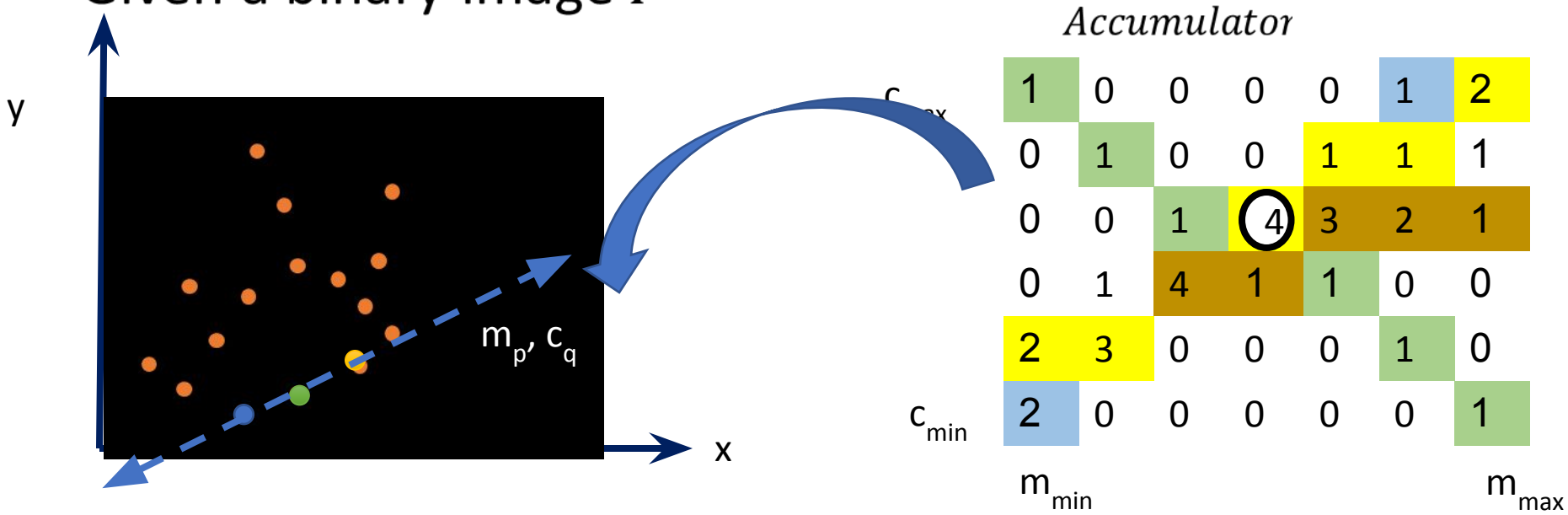
- Given a binary image I



- Step 3: Identify the cell having highest votes-> $A(P, Q)$
  - $A(P, Q)$ corresponds to slope m_p and intercept c_q

How is it useful?

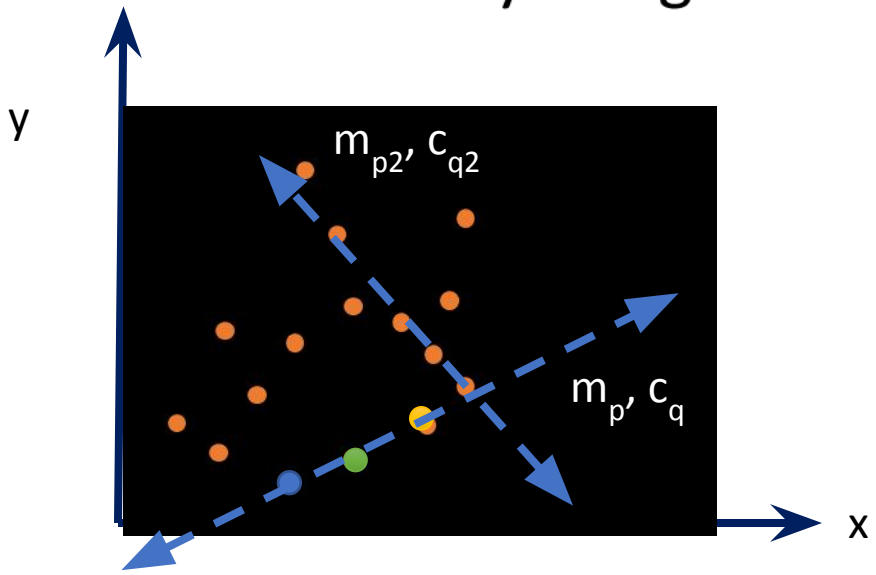
- Given a binary image I



- Step 3: Identify the cell having highest votes-> $A(P, Q)$
  - $A(P, Q)$ corresponds to slope m_p and intercept c_q

How is it useful?

- Given a binary image I



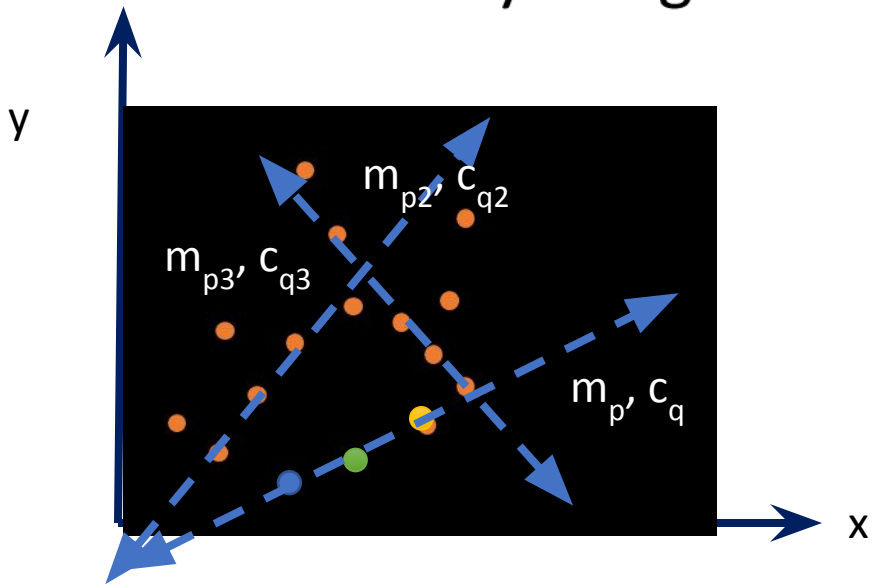
$Accumulator$

c_{max}							
	1	0	0	0	0	1	2
	0	1	0	0	1	1	1
	0	0	1	4	3	2	1
	0	1	4	1	1	0	0
	2	3	0	0	0	1	0
c_{min}	2	0	0	0	0	0	1
	m_{min}						m_{max}

- Step 3: Identify the cell having next highest votes-> $A(P, Q)$
  - $A(P, Q)$ corresponds to slope m_{p2} and intercept c_{q2}

How is it useful?

- Given a binary image I



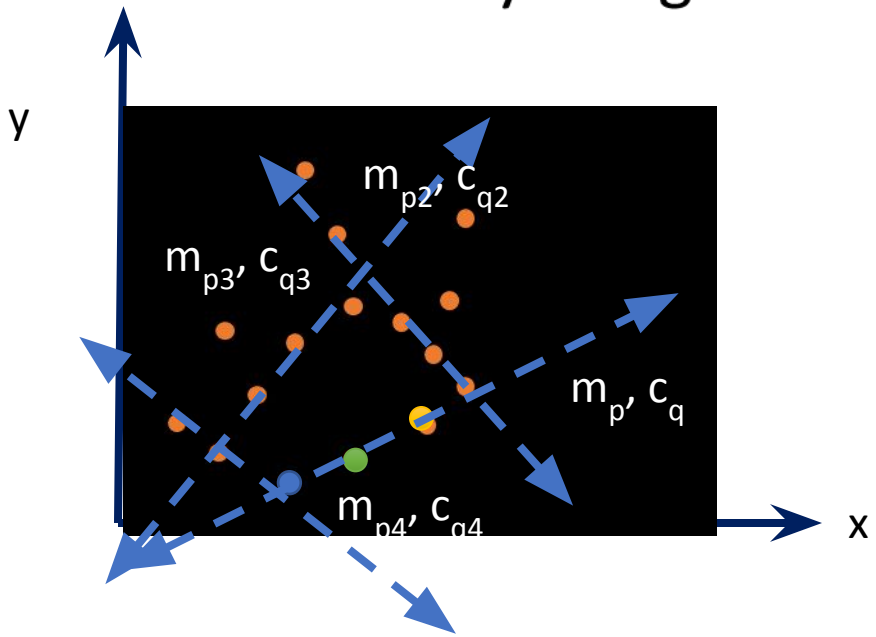
$Accumulator$

c_{max}							
	1	0	0	0	0	1	2
	0	1	0	0	1	1	1
	0	0	1	4	3	2	1
	0	1	4	1	1	0	0
	2	3	0	0	0	1	0
c_{min}	2	0	0	0	0	0	1
	m_{min}						m_{max}

- Step 3: Identify the cell having next highest votes-> $A(P, Q)$
  - $A(P, Q)$ corresponds to slope m_{p3} and intercept c_{q3}

How is it useful?

- Given a binary image I

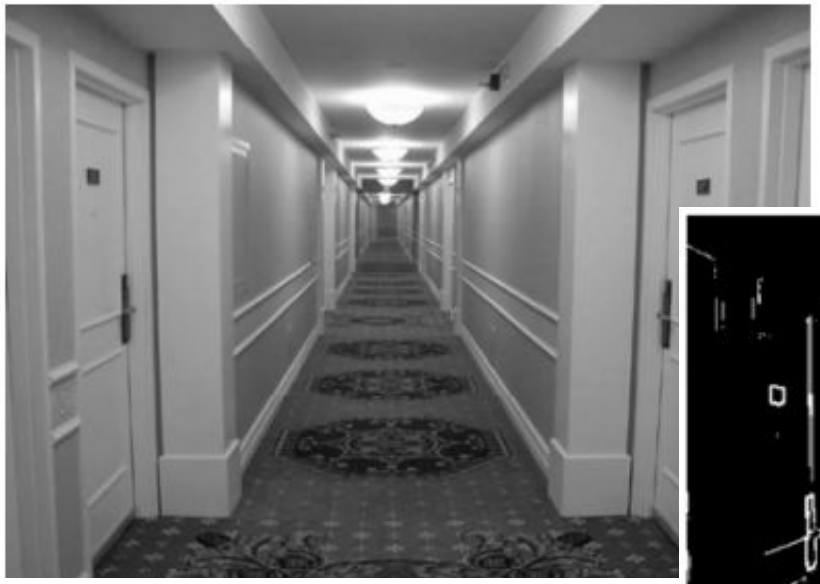


$Accumulator$

c_{max}							
	1	0	0	0	0	1	2
	0	1	0	0	1	1	1
	0	0	1	4	3	2	1
	0	1	4	1	1	0	0
	2	3	0	0	0	1	0
c_{min}	2	0	0	0	0	0	1
	m_{min}						m_{max}

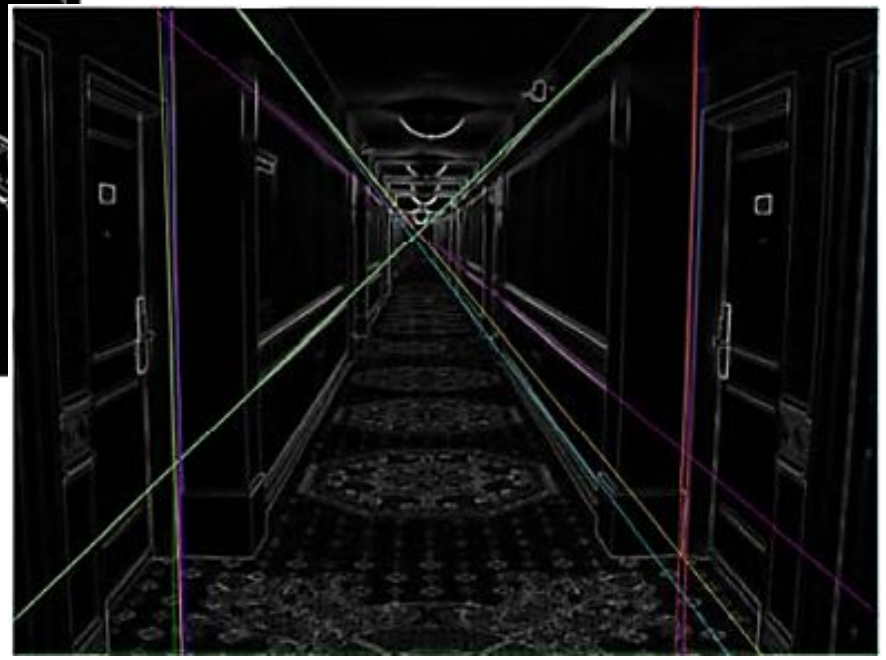
- Step 3: Identify the cell having next highest votes-> $A(P, Q)$
  - $A(P, Q)$ corresponds to slope m_{p4} and intercept c_{q4}

Results



Original image

Edge map Sobel

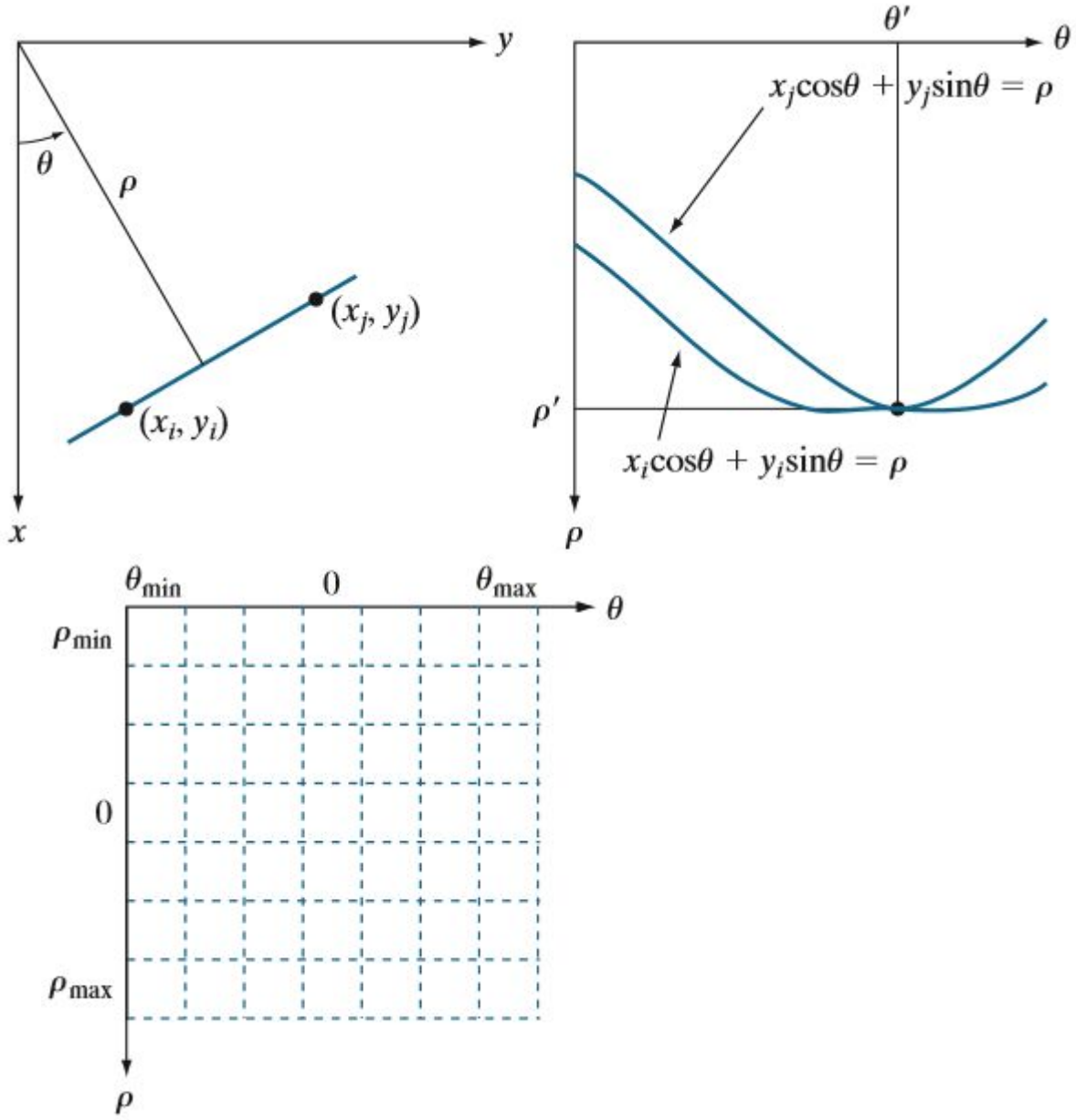


10 prominent lines by Hough

Question?

- There is one limitation here:
- Cannot detect vertical lines ?

Normal form of line



- Vertical lines can not be dealt with *mc*-plane
- Normal form of line equation is used:
- $\rho = x\cos\theta + y\sin\theta$
  - θ range is $\mp 90°$
  - ρ range is $\mp\sqrt{M^2 + N^2}$
- A point in xy-space gives a sinusoidal in $\theta\rho$-space

Advantages

- Conceptually simple.
- Easy implementation.
- Handles missing and occluded data very gracefully.
- Can be adapted to many types of forms, not just lines.