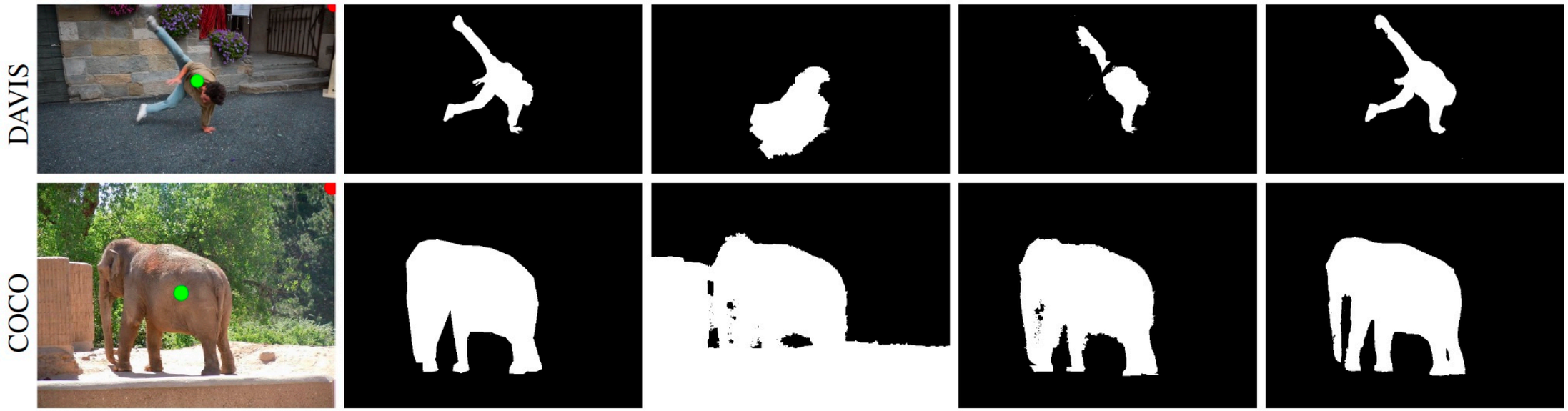
# Digital Image Processing

## - Image Segmentation : ‘Point’, ‘Line’, ‘Edge’ Detection -

**SNU Computer Graphics & Image Processing LAB**
**YongJin Jeon**

# Segmentation

- Segmentation

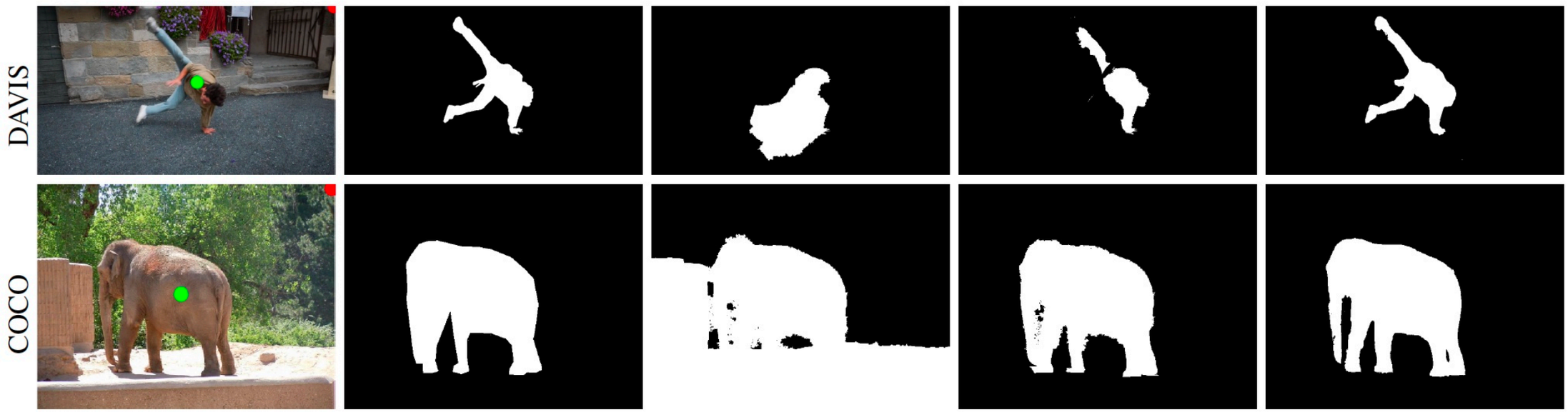



(a) Input (b) Ground truth (c) Graph cuts (d) DOS (e) Our approach

-> Subdivides an image into its constituent regions or objects

# Segmentation

- Segmentation




(a) Input (b) Ground truth (c) Graph cuts (d) DOS (e) Our approach

-> Subdivides an image into its constituent regions or objects

-> Basic Properties of Intensity values for Segmentation Algorithm

**Discontinuity** **Similarity**

# Fundamentals

- Preceding Conditions for partitioning the image into region

# Fundamentals

- Preceding Conditions for partitioning the image into region

$$1)\ \bigcup_{i=1}^{n} R_i = R$$

$$2)\ R_i \text{ is a connected set, } (i = 1,2,\ldots,n)$$

$$3)\ R_i \bigcap R_j = \varnothing \text{ for all } i \text{ and } j,\ i \neq j$$

$$4)\ Q(R_i) = TRUE,\ (i = 1,2,\ldots,n)$$

$$5)\ Q(R_i \bigcup R_j) = FALSE \text{ for any adjacent regions } R_i,\ R_j$$

R : Entire Spatial Region occupied by an Image
Ri : Partitioned Subregions

# Fundamentals

- Preceding Conditions for partitioning the image into region

$$1)\ \bigcup_{i=1}^{n} R_i = R$$

Points in a region be connected

$$2)\ R_i \text{ is a connected set, } (i = 1,2,\ldots,n)$$

$$3)\ R_i \bigcap R_j = \varnothing \text{ for all } i \text{ and } j,\ i \neq j$$

$$4)\ Q(R_i) = TRUE,\ (i = 1,2,\ldots,n)$$

$$5)\ Q(R_i \bigcup R_j) = FALSE \text{ for any adjacent regions } R_i,\ R_j$$

R : Entire Spatial Region occupied by an Image
Ri : Partitioned Subregions

# Fundamentals

- Preceding Conditions for partitioning the image into region

$$1)\ \bigcup_{i=1}^{n} R_i = R$$

$$2)\ R_i \text{ is a connected set, } (i = 1,2,\ldots,n)$$

$$3)\ R_i \bigcap R_j = \varnothing \text{ for all } i \text{ and } j,\ i \neq j$$

All pixels in Ri have the same intensity level

$$4)\ Q(R_i) = TRUE,\ (i = 1,2,\ldots,n)$$

$$5)\ Q(R_i \bigcup R_j) = FALSE \text{ for any adjacent regions } R_i,\ R_j$$

R : Entire Spatial Region occupied by an Image
Ri : Partitioned Subregions

# Fundamentals

- Preceding Conditions for partitioning the image into region

$$1)\ \bigcup_{i=1}^{n} R_i = R$$

$$2)\ R_i \text{ is a connected set, } (i = 1,2,\ldots,n)$$

$$3)\ R_i \cap R_j = \varnothing \text{ for all } i \text{ and } j,\ i \neq j$$

$$4)\ Q(R_i) = TRUE,\ (i = 1,2,\ldots,n)$$

$$5)\ Q(R_i \cup R_j) = FALSE \text{ for any adjacent regions } R_i,\ R_j$$

R : Entire Spatial Region occupied by an Image
Ri : Partitioned Subregions

Intensity of Pixels in two adjacent regions should be different

# Fundamentals

- Preceding Conditions for partitioning the image into region

$$1)\ \bigcup_{i=1}^{n} R_i = R$$

$$2)\ R_i \text{ is a connected set, } (i = 1,2,\ldots,n)$$

$$3)\ R_i \bigcap R_j = \varnothing \text{ for all } i \text{ and } j,\ i \neq j$$

$$4)\ Q(R_i) = TRUE,\ (i = 1,2,\ldots,n)$$

$$5)\ Q(R_i \bigcup R_j) = FALSE \text{ for any adjacent regions } R_i,\ R_j$$

R : Entire Spatial Region occupied by an Image
Ri : Partitioned Subregions

-> Basic Categories dealing with properties of intensity values

* Discontinuity : Partitioning the image into regions by detecting boundary based on **local discontinuities in intensity**

**Edge-base Segmentation**

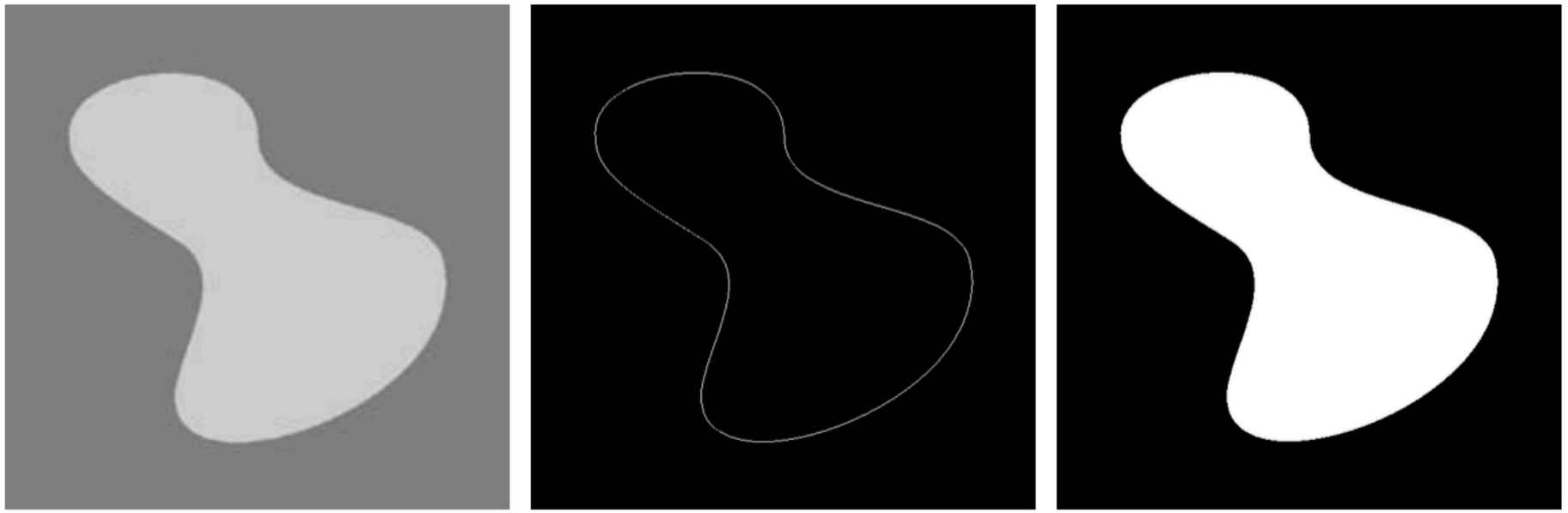
* Similarity : Partitioning the image into regions that are similar according to **a set of predefined criteria**

**Region-base Segmentation**

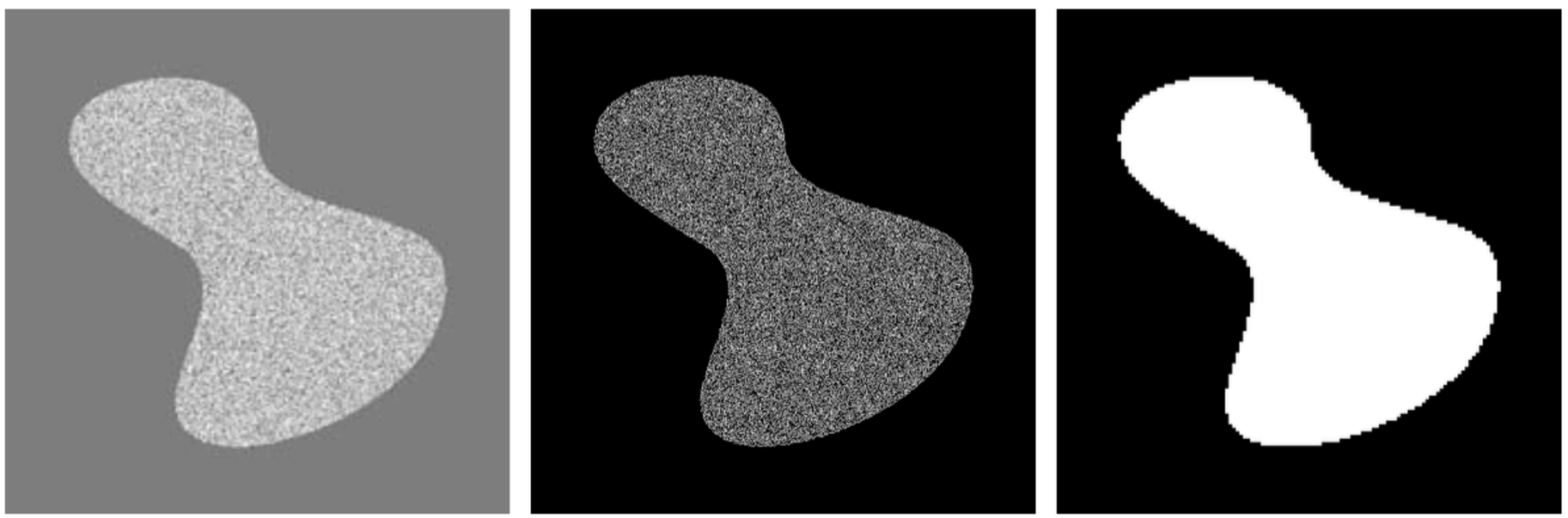
# Fundamentals

- Partitioning Result of monochrome image

**Partitioning with Discontinuity**



**Partitioning with Similarity**

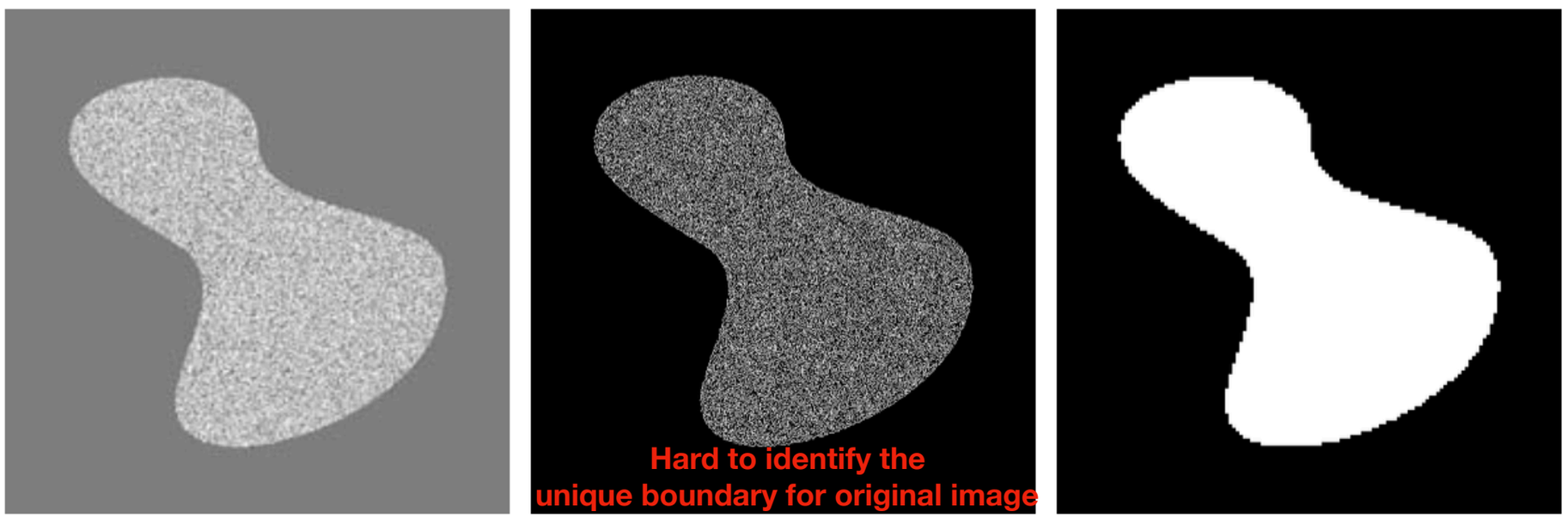


# Fundamentals

- Partitioning Result of monochrome image

**Partitioning with Discontinuity**


**Partitioning with Similarity**

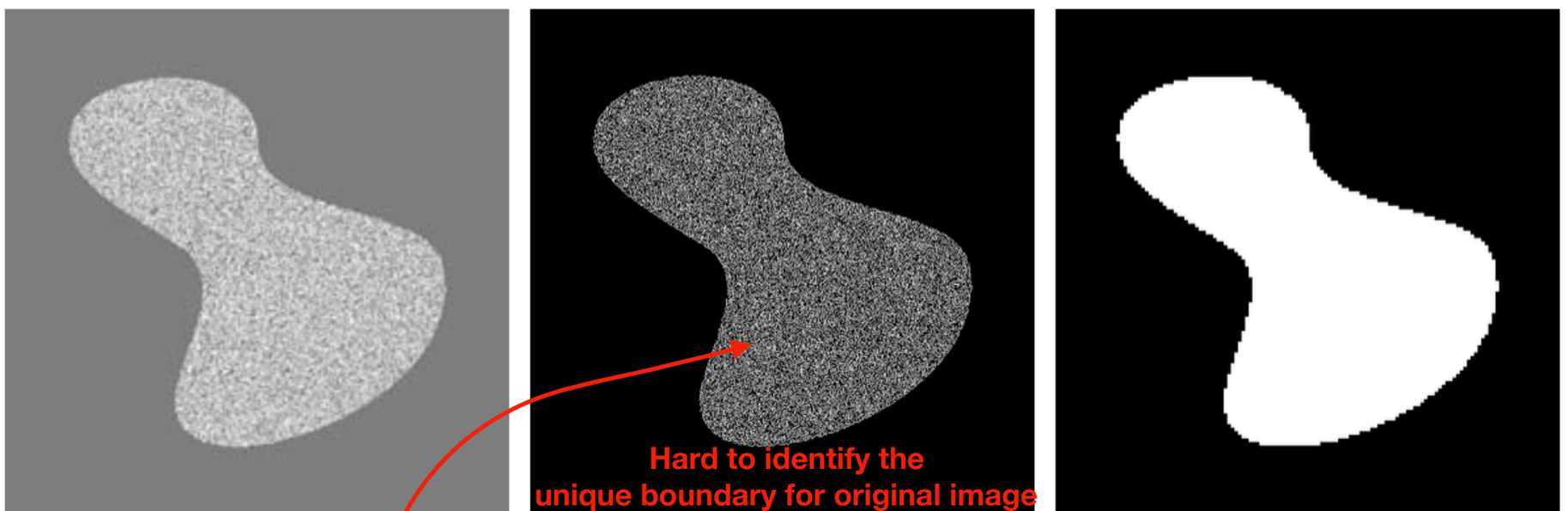


# Fundamentals

- Partitioning Result of monochrome image

**Partitioning with Discontinuity**


**Partitioning with Similarity**

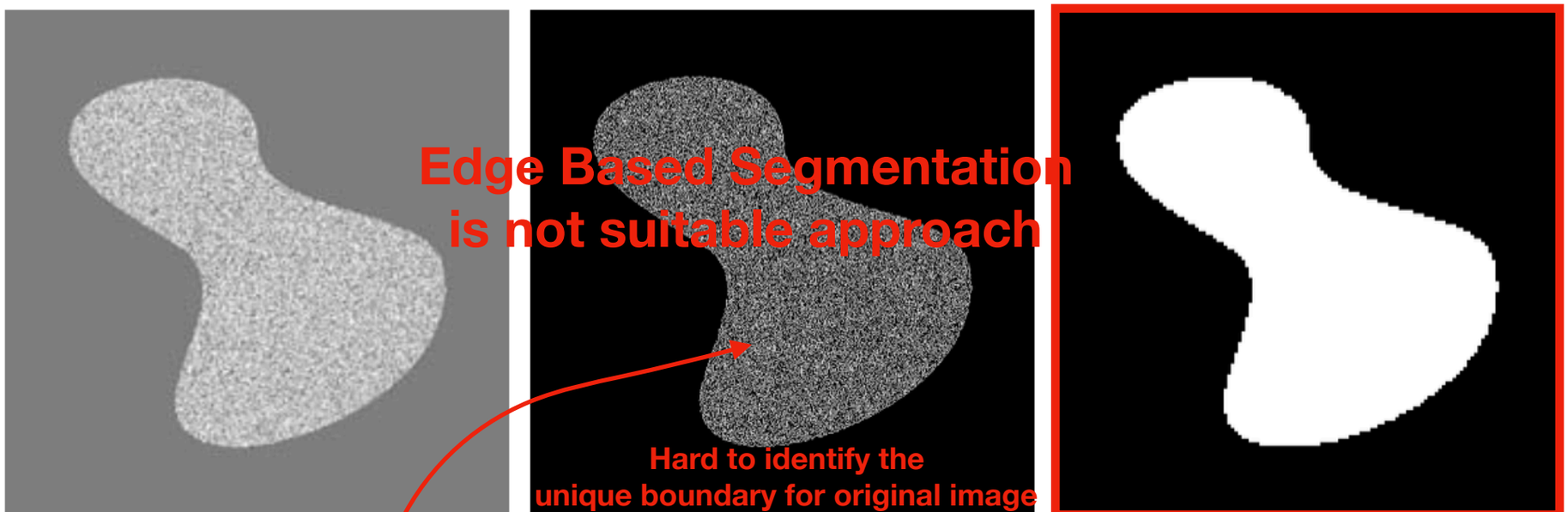


# Fundamentals

- Partitioning Result of monochrome image

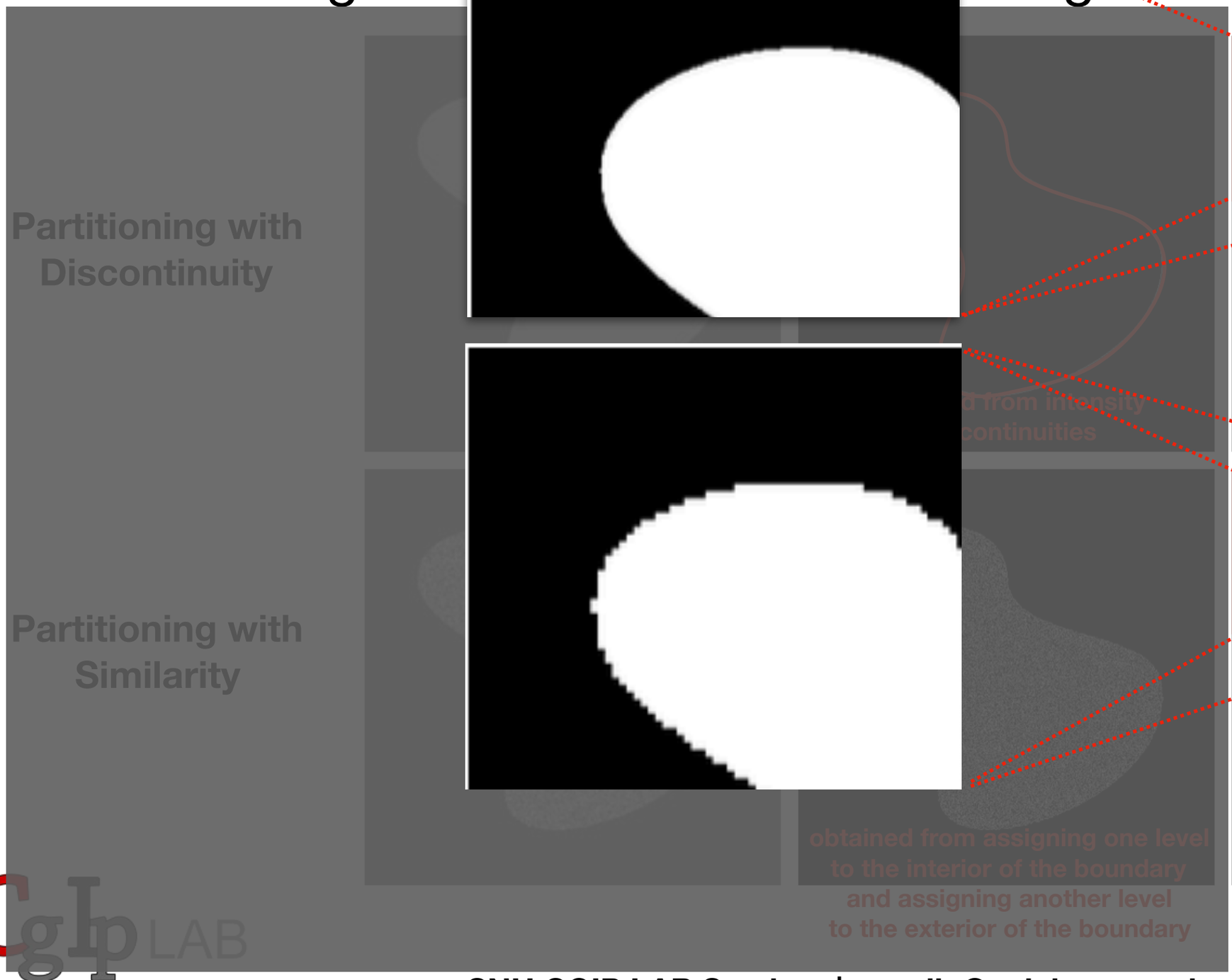
**Partitioning with Discontinuity**


**Partitioning with Similarity**



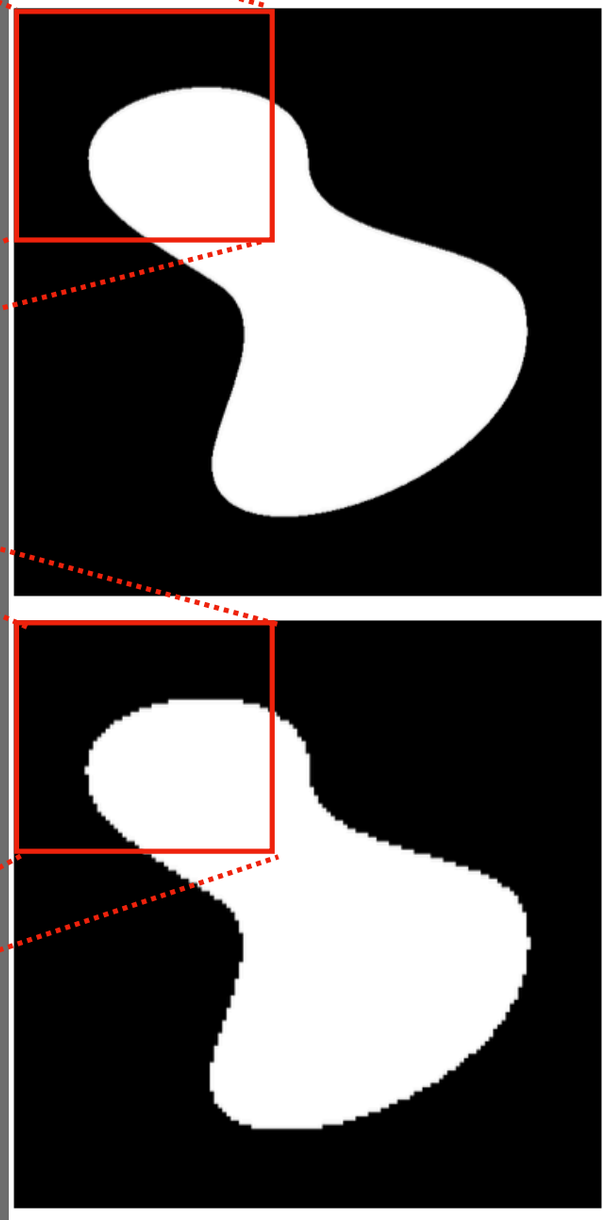
# Fundamentals

- Partitioning Res image







# Point, Line, Edge Detection

- Segmentation by Point, Line and Edge Detection
  -> Focus on the methodology based on detecting sharp, local changes in intensity

# Point, Line, Edge Detection

- Segmentation by Point, Line and Edge Detection
  -> Focus on the methodology based on detecting sharp, local changes in intensity

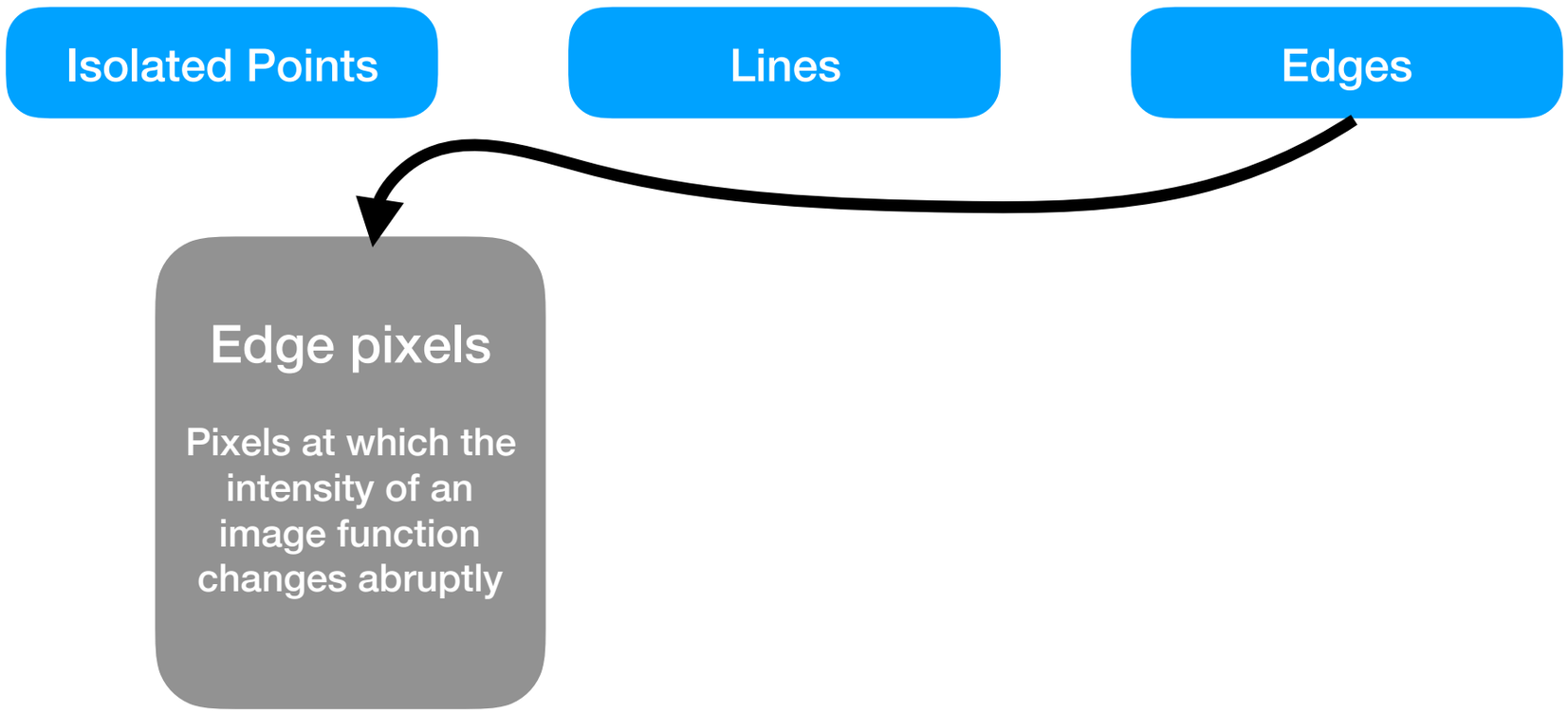
  -> Three types of Image features

Isolated Points | Lines | Edges

# Point, Line, Edge Detection

- Segmentation by Point, Line and Edge Detection
  -> Focus on the methodology based on detecting sharp, local changes in intensity

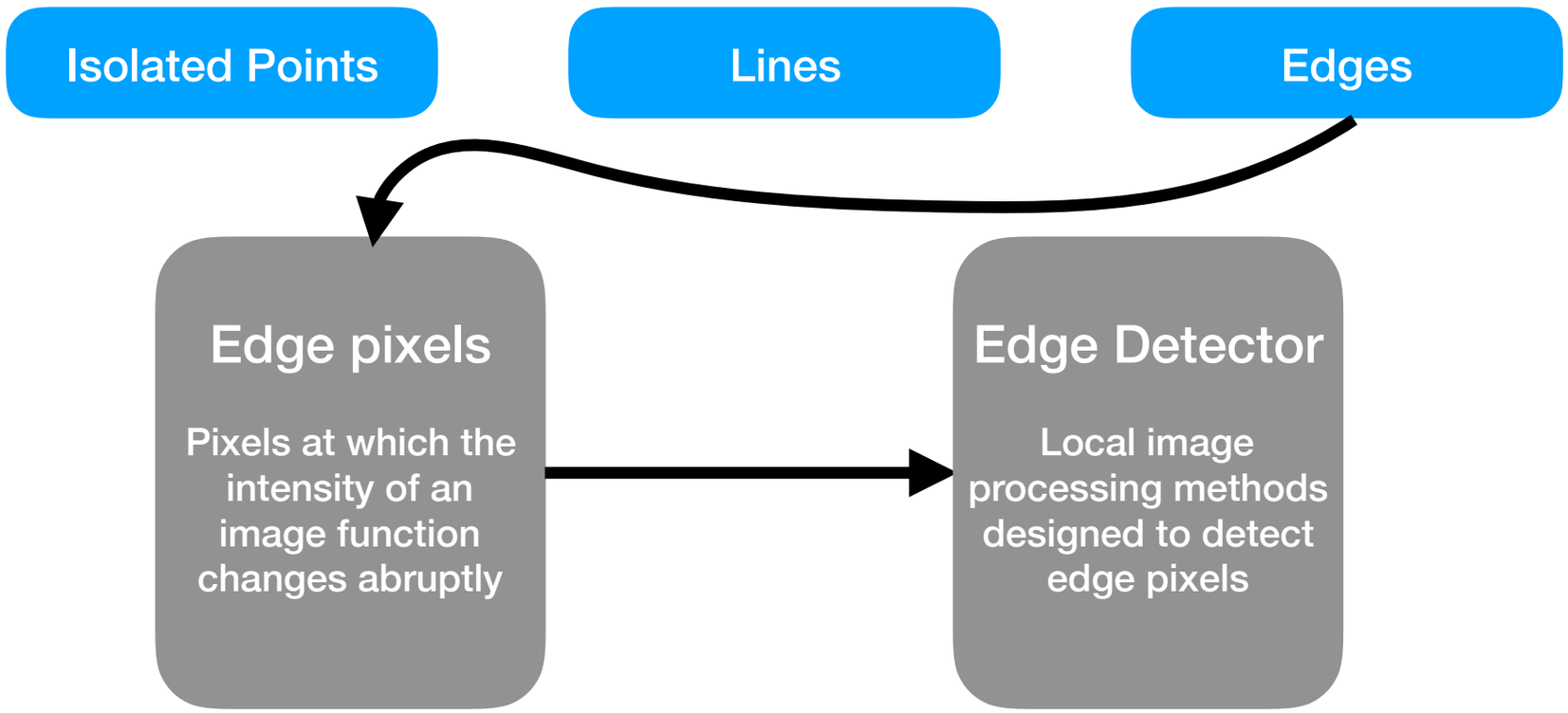
  -> Three types of Image features



# Point, Line, Edge Detection

- Segmentation by Point, Line and Edge Detection
  -> Focus on the methodology based on detecting sharp, local changes in intensity

  -> Three types of Image features



# Point, Line, Edge Detection

- How could we detect the local changes in intensity?

# Point, Line, Edge Detection

- How could we detect the local changes in intensity?
  -> Using Derivatives

**First Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset of an intensity step or ramp
3) Nonzero at points along the intensity ramp

**Second Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset and end of an intensity step or ramp
3) Zero along intensity ramp

# Point, Line, Edge Detection

- How could we detect the local changes in intensity?
  -> Using Derivatives

**First Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset of an intensity step or ramp
3) Nonzero at points along the intensity ramp

**Second Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset and end of an intensity step or ramp
3) Zero along intensity ramp

**Partial derivative : consider an image function**

$$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial f'(x)}{\partial x} = f'(x+1) - f'(x)$$

$$= f(x+2) - f(x+1) - f(x+1) + f(x)$$

$$= f(x+2) - 2f(x+1) + f(x)$$

**Expansion in to x+1**

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) - 2f(x) + f(x-1)$$

**subtract 1 from the arguments**

# Point, Line, Edge Detection

- Several Types of Edges

Ramp Edges

Step Edges

Roof Edges

# Point, Line, Edge Detection

- Several Types of Edges

Ramp Edges

Step Edges

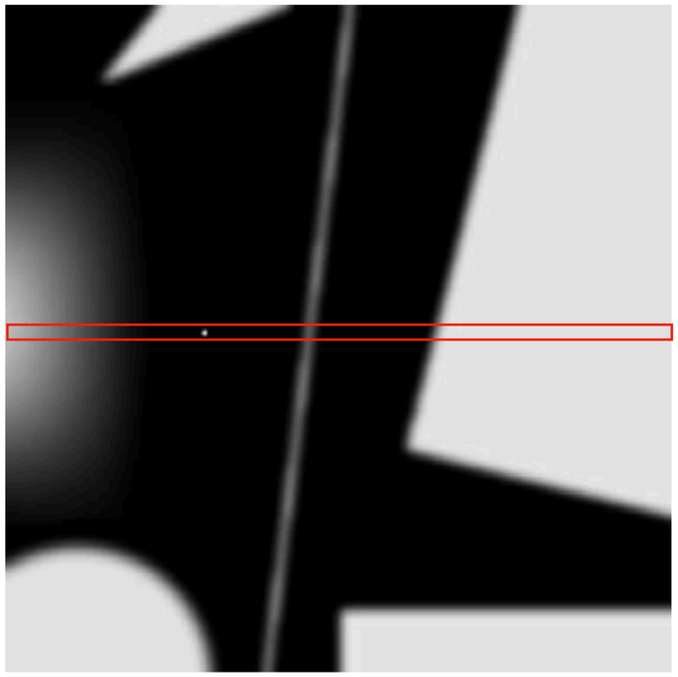
**Intensity transitions between solid objects and background**

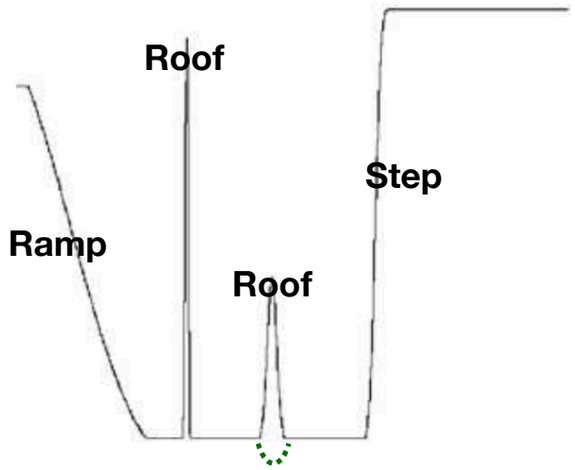
Roof Edges

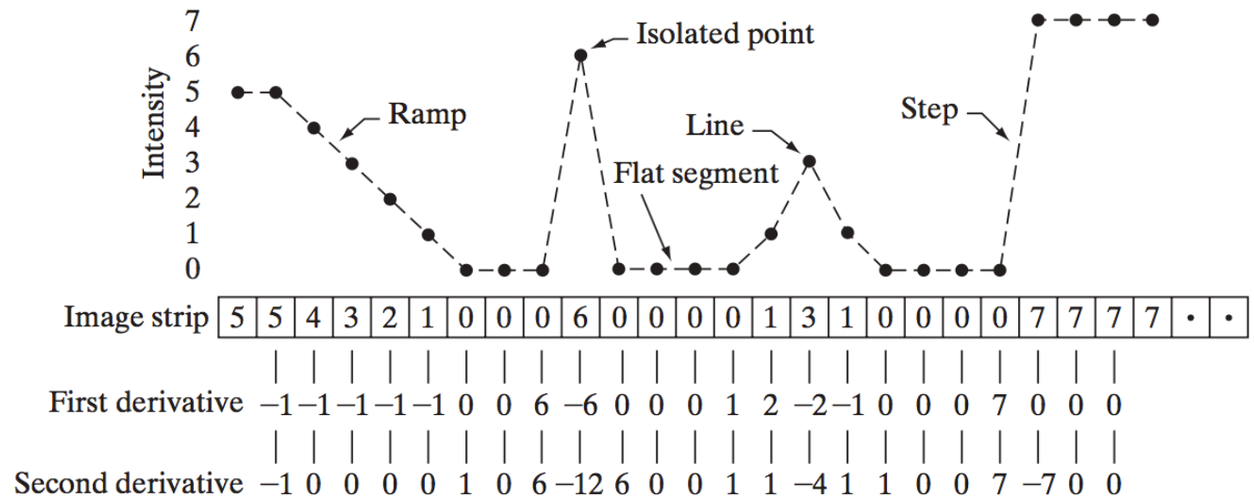
**Intensity transitions involving thin objects**

# Point, Line, Edge Detection

- Several Types of Edges

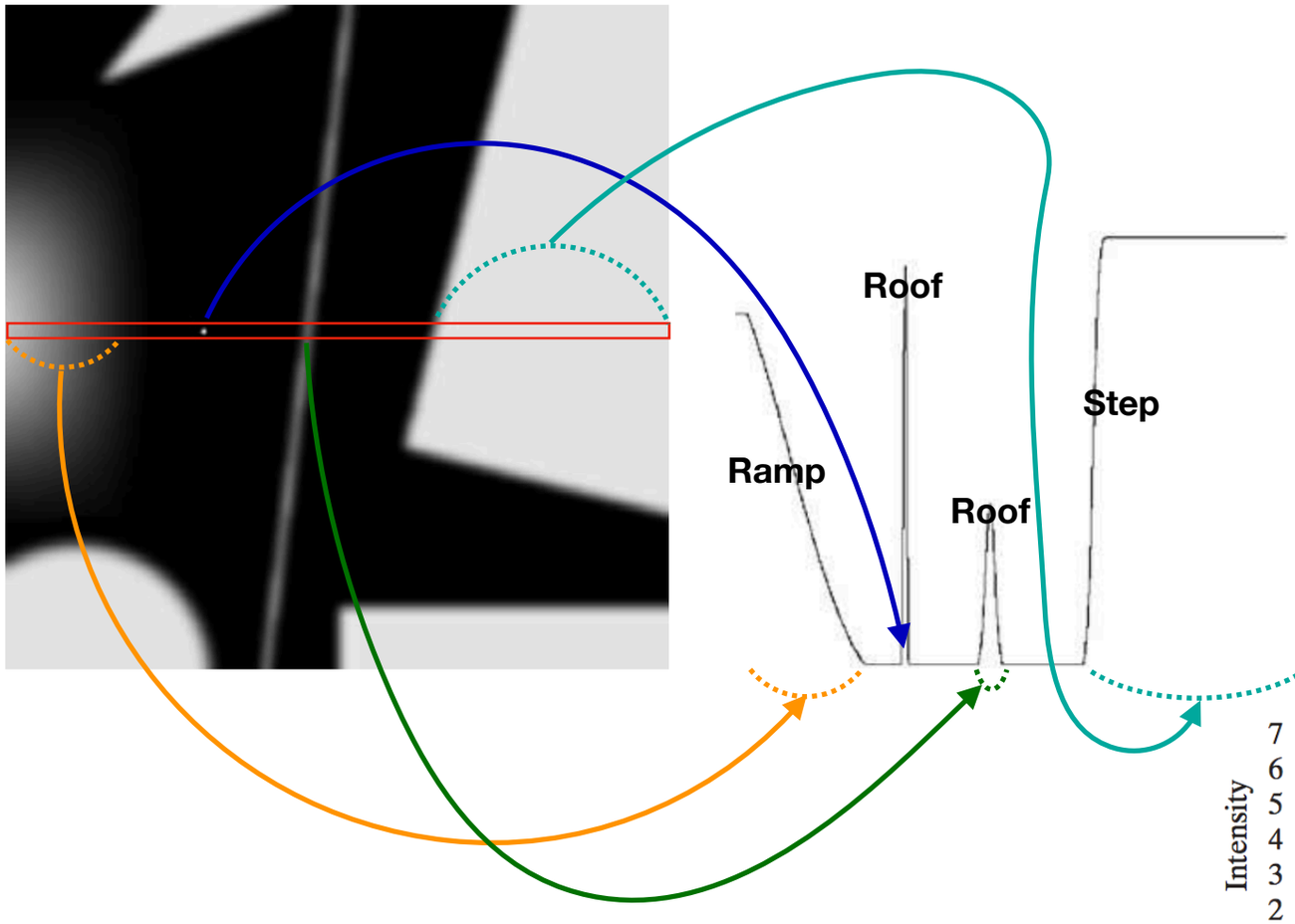


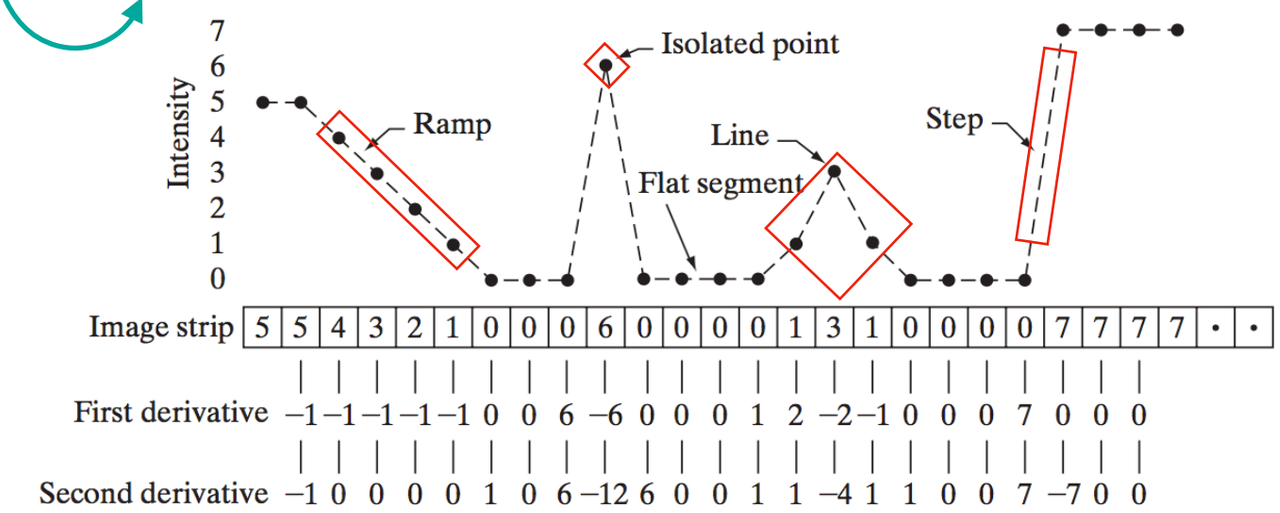


# Point, Line, Edge Detection

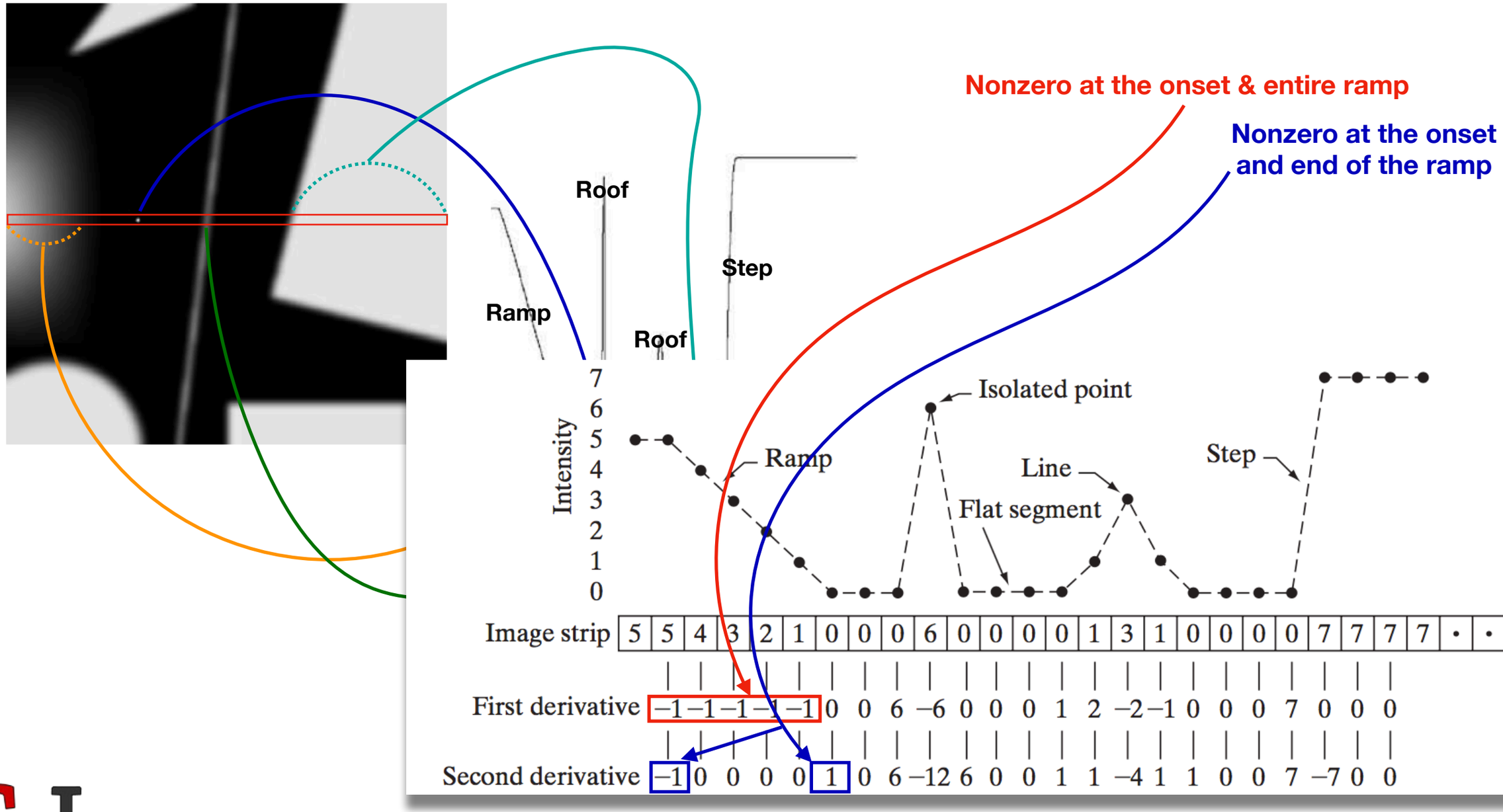
- Several Types of Edges



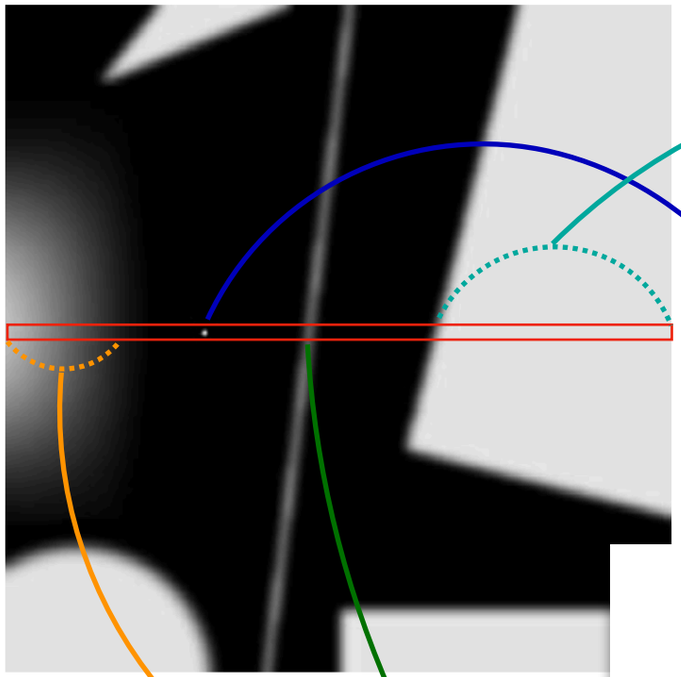
# Point, Line, Edge Detection

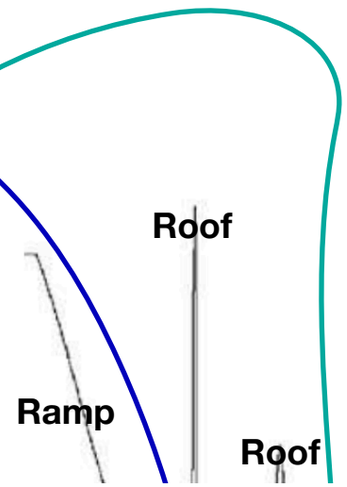
- Several Types of Edges



# Point, Line, Edge Detection

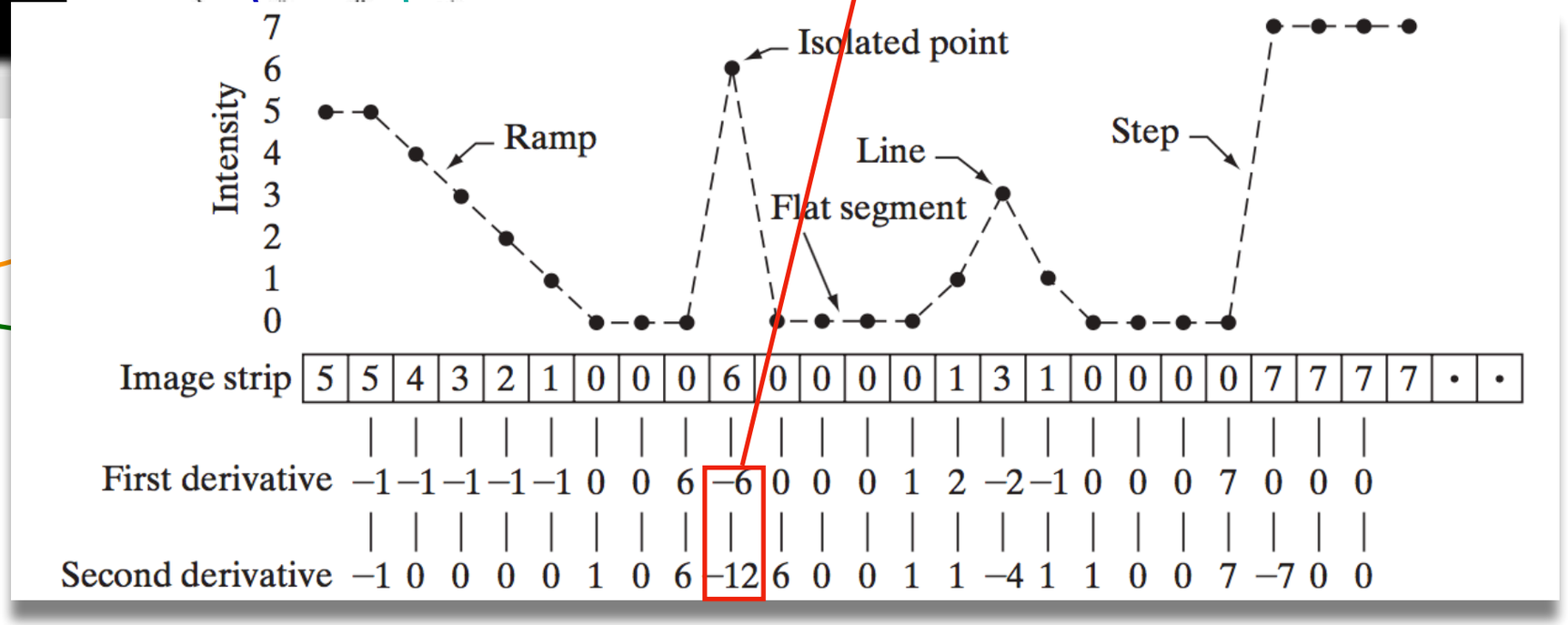
- Several Types of Edges





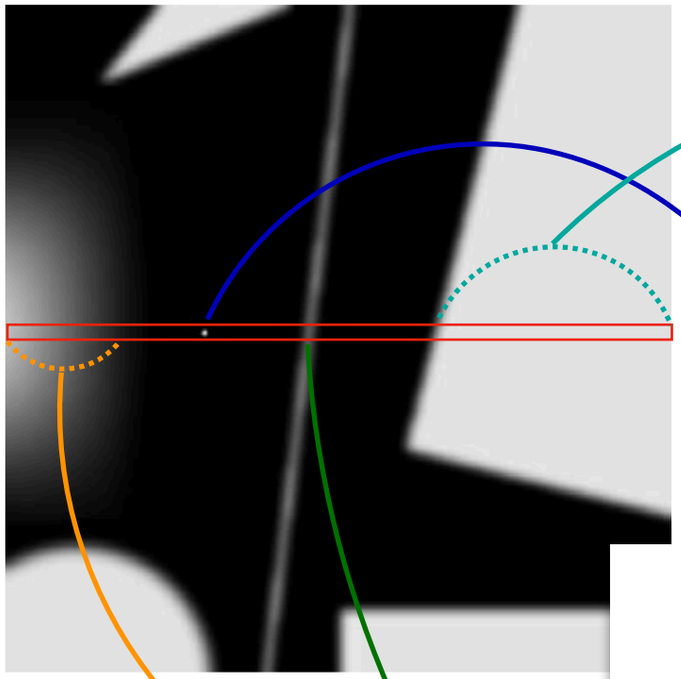

Magnitude of the response at the Isolated point

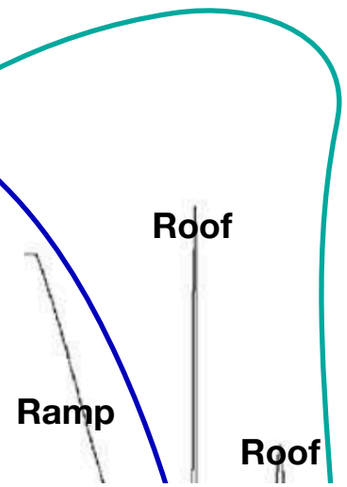
1st order derivative < 2nd order derivative



# Point, Line, Edge Detection

- Several Types of Edges



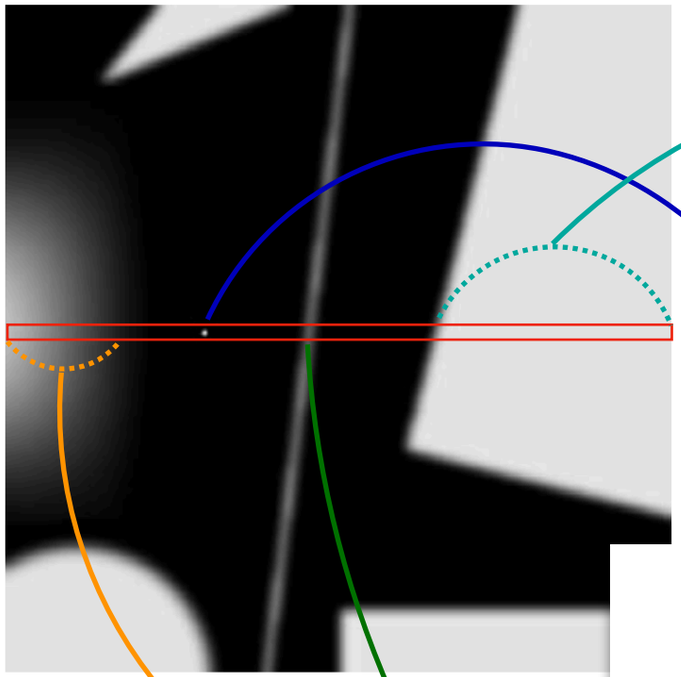


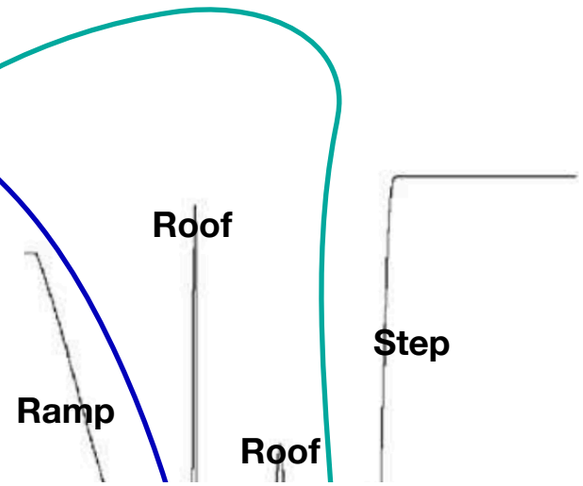

Magnitude of the response at the Isolated point

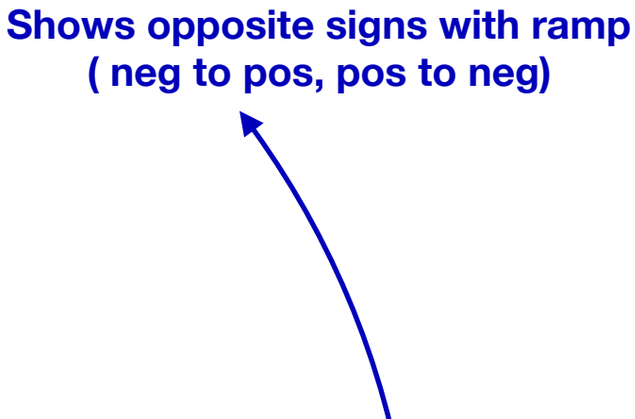
1st order derivative < 2nd order derivative

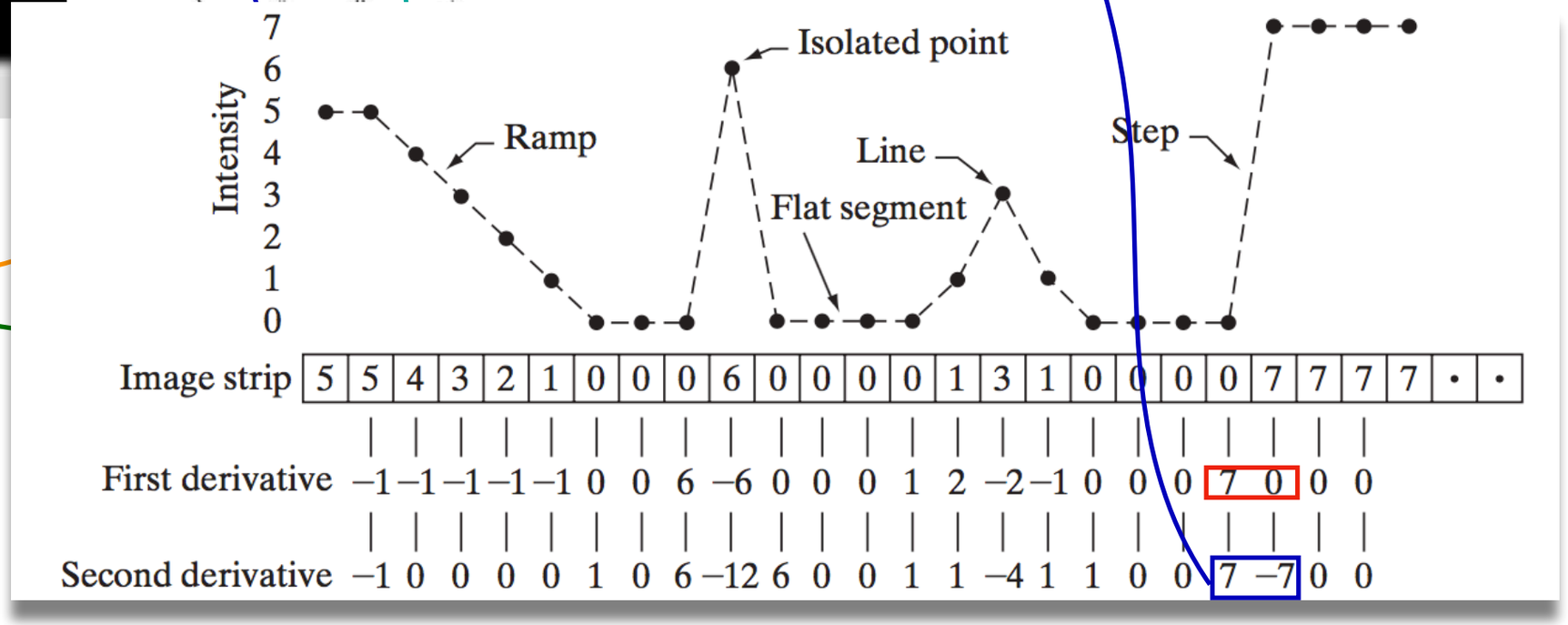
# Point, Line, Edge Detection

- Several Types of Edges



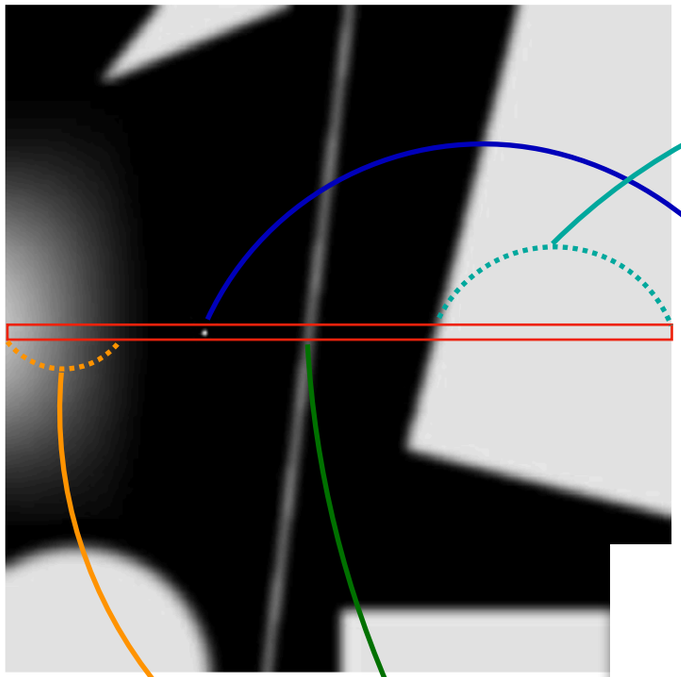


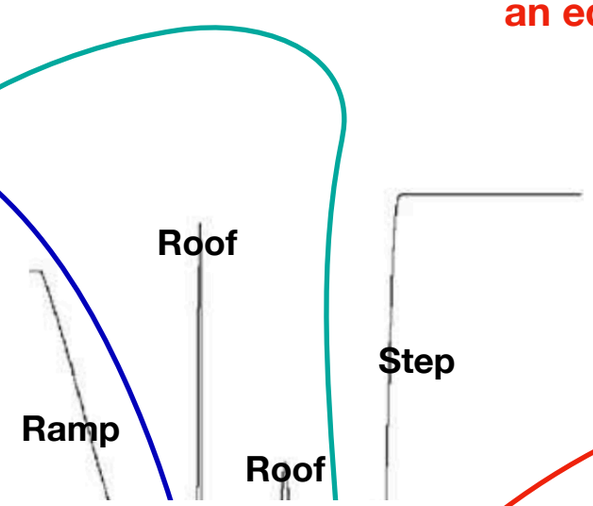



# Point, Line, Edge Detection

- Several Types of Edges






**Double Edge effect**
**(Using second-order derivative to determine whether an edge is a transition from light to dark or dark to light)**

# Point, Line, Edge Detection

- Conclusions from former clues

**First Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset of an intensity step or ramp
3) Nonzero at points along the intensity ramp

**Second Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset and end of an intensity step or ramp
3) Zero along intensity ramp

-> First order derivative generally produce thicker edges in an image
-> Second order derivative have a stronger response to fine detail (Lines, Isolated points, Noise, ..)
-> Second order derivative produce double-edge response ramp and step transition in intensity
-> Sign of the second derivative can be used to determine whether transition into edge from light to dark or dark to light

# Point, Line, Edge Detection

- First-order derivatives VS Second-order derivatives

**First Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset of an intensity step or ramp
3) Nonzero at points along the intensity ramp

**Second Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset and end of an intensity step or ramp
3) Zero along intensity ramp

# Point, Line, Edge Detection

- First-order derivatives VS Second-order derivatives

**First Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset of an intensity step or ramp
3) Nonzero at points along the intensity ramp

**Second Order Derivatives**

1) Zero in areas of constant intensity
2) Nonzero at the onset and end of an intensity step or ramp
3) Zero along intensity ramp

-> Use Spatial Filters

=> Laplacian based Isotropic filter

=> Gradient based Robert-cross gradient operator

=> Gradient based Sobel operator

**-> GO TO Chap 3!**

# Point, Line, Edge Detection

- Isolated Point Detection

  -> Based on the preceding content, Point detection should be based on the second derivative (Laplacian)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

# Point, Line, Edge Detection

- Isolated Point Detection

  -> Based on the preceding content, Point detection should be based on the second derivative (Laplacian)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

with former equations in p21,

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

# Point, Line, Edge Detection

- Isolated Point Detection

  -> Based on the preceding content, Point detection should be based on the second derivative (Laplacian)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

with former equations in p21,

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

-> With Laplacian masking, point detected at the location(x,y) on which the mask is centered if absolute value of the response of the mask at that point exceeds a specific threshold

$$g(x,y) = \begin{cases} 1, & if \ |R(x,y)| \geq T \\ 0, & otherwise \end{cases}$$

Treshold value

Response given by spatial filtering

output image

# Point, Line, Edge Detection

- Isolated Point Detection

  -> Based on the preceding conten

  second derivative (Laplacian)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

  with former equations in p21,

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x$$

  -> With Laplacian masking, point d

  the mask is centered if absolute

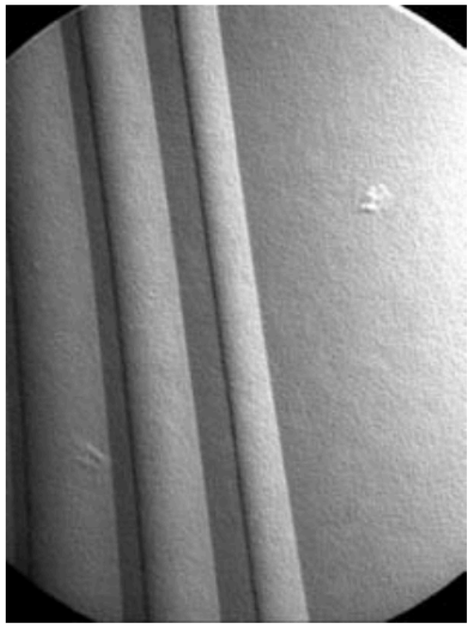
  exceeds a specific threshold

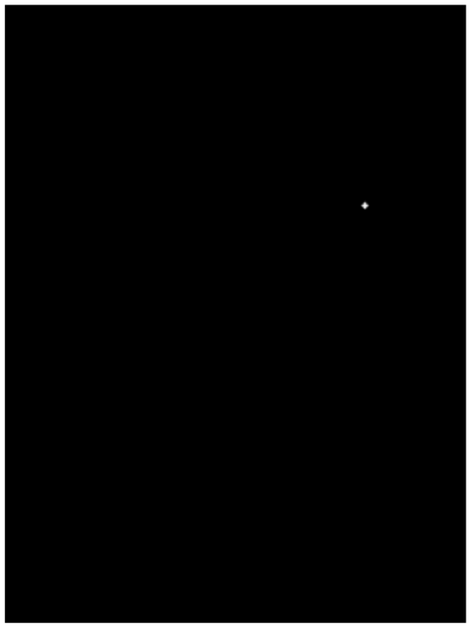
$$g(x,y) = \begin{cases} 1 \ , & if \ |R(x,y)| \geq T \\ 0 \ , & otherwise \end{cases}$$

output image

Response

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |





# Point, Line, Edge Detection

- Isolated Point Detection

  -> Based on the preceding conten

  second derivative (Laplacian)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

  with former equations in p21,

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x$$

  -> With Laplacian masking, point d

  the mask is centered if absolute

  exceeds a specific threshold

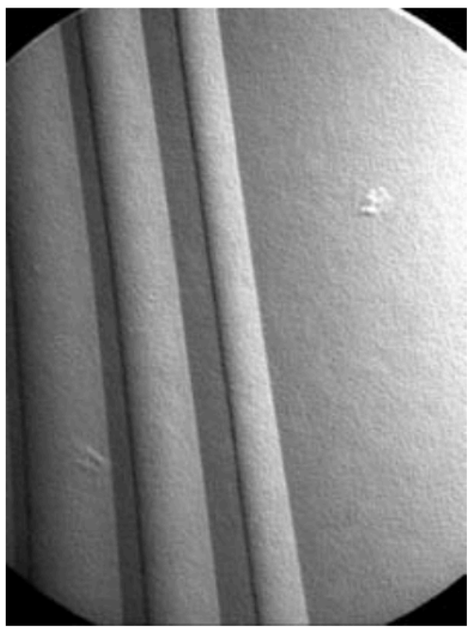
$$g(x,y) = \begin{cases} 1 \ , & if \ |R(x,y)| \geq T \\ 0 \ , & otherwise \end{cases}$$

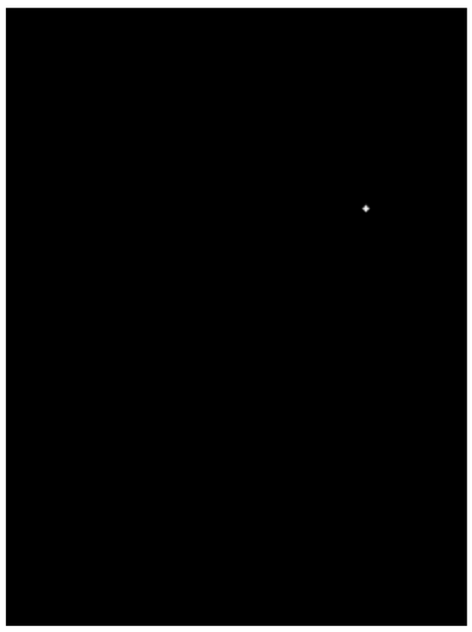
output image

Response

**Coefficients sum to zero, for make mask response will become zero in areas of constant intensity**

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |





# Point, Line, Edge Detection

- Isolated Point Detection

  -> Based on the preceding conten

  second derivative (Laplacian)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

  with former equations in p21,

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x$$

  -> With Laplacian masking, point c

  the mask is centered if absolute

  exceeds a specific threshold

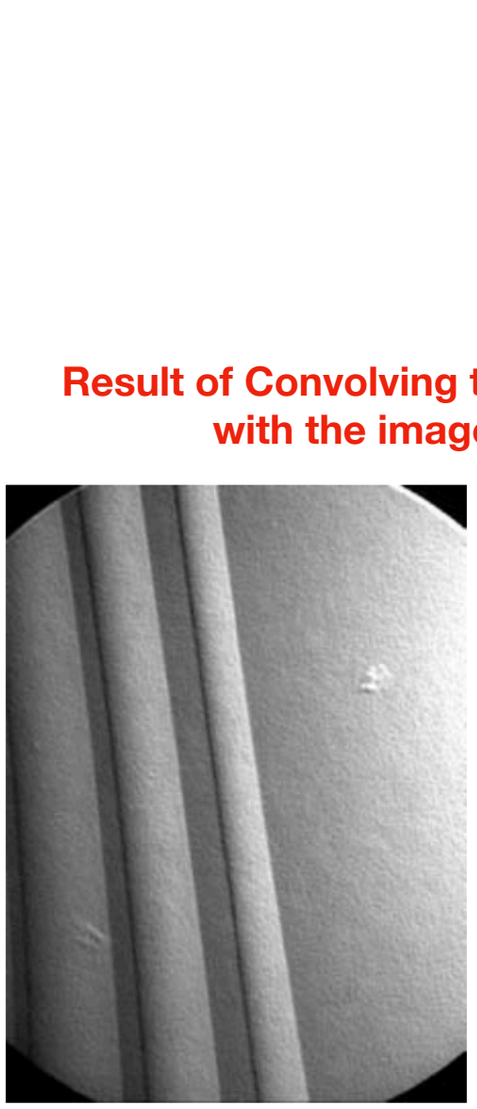
$$g(x,y) = \begin{cases} 1 \ , & if \ |R(x,y)| \geq T \\ 0 \ , & otherwise \end{cases}$$

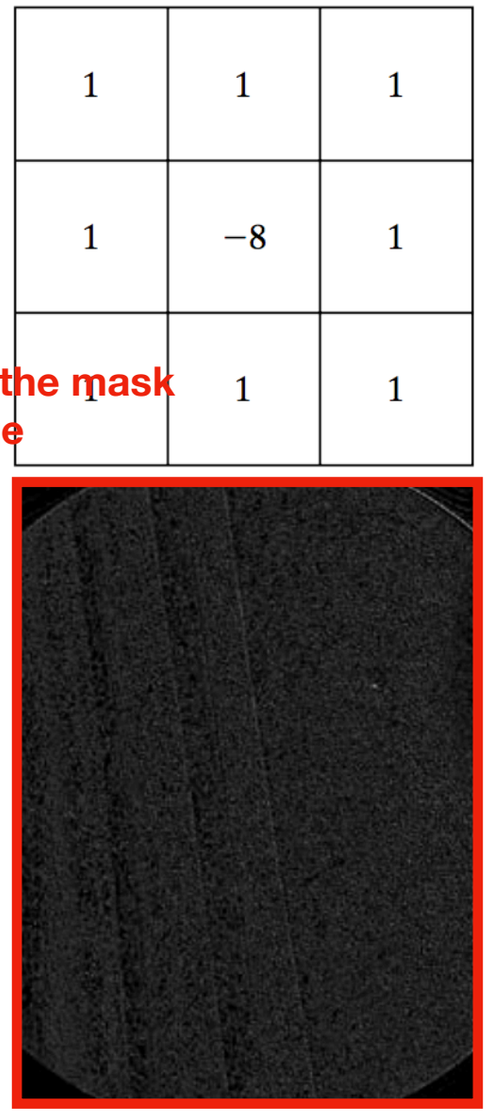
output image

Response

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |

Result of Convolving the mask with the image





# Point, Line, Edge Detection

- Isolated Point Detection

  -> Based on the preceding conten

  second derivative (Laplacian)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

  with former equations in p21,

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x$$

  -> With Laplacian masking, point c

  the mask is centered if absolute

  exceeds a specific threshold

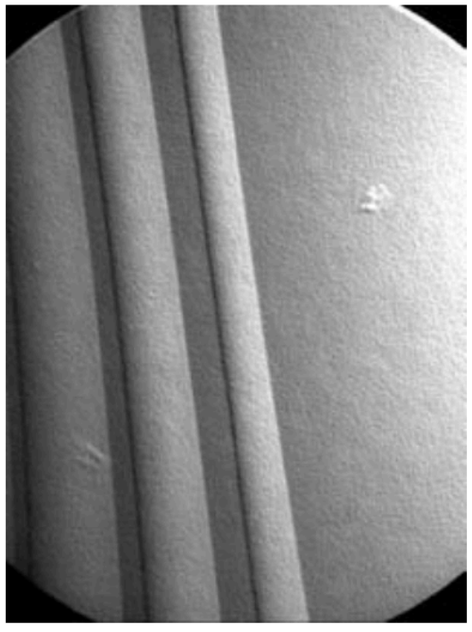
$$g(x,y) = \begin{cases} 1 \ , & if \ |R(x,y)| \geq T \\ 0 \ , & otherwise \end{cases}$$

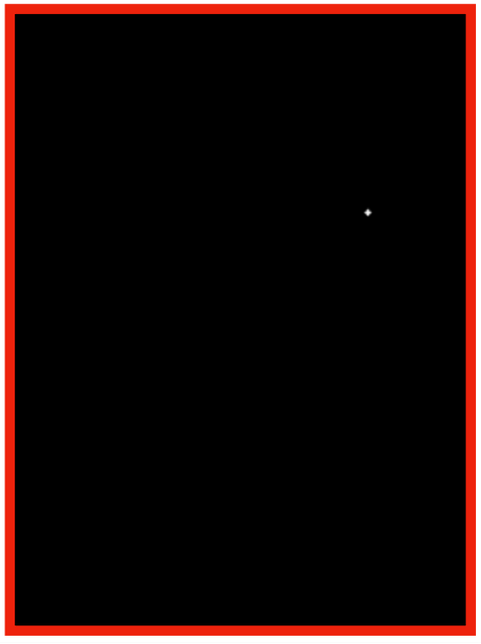
output image

Response

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |

Result of using point detecting by Thresholding





# Point, Line, Edge Detection

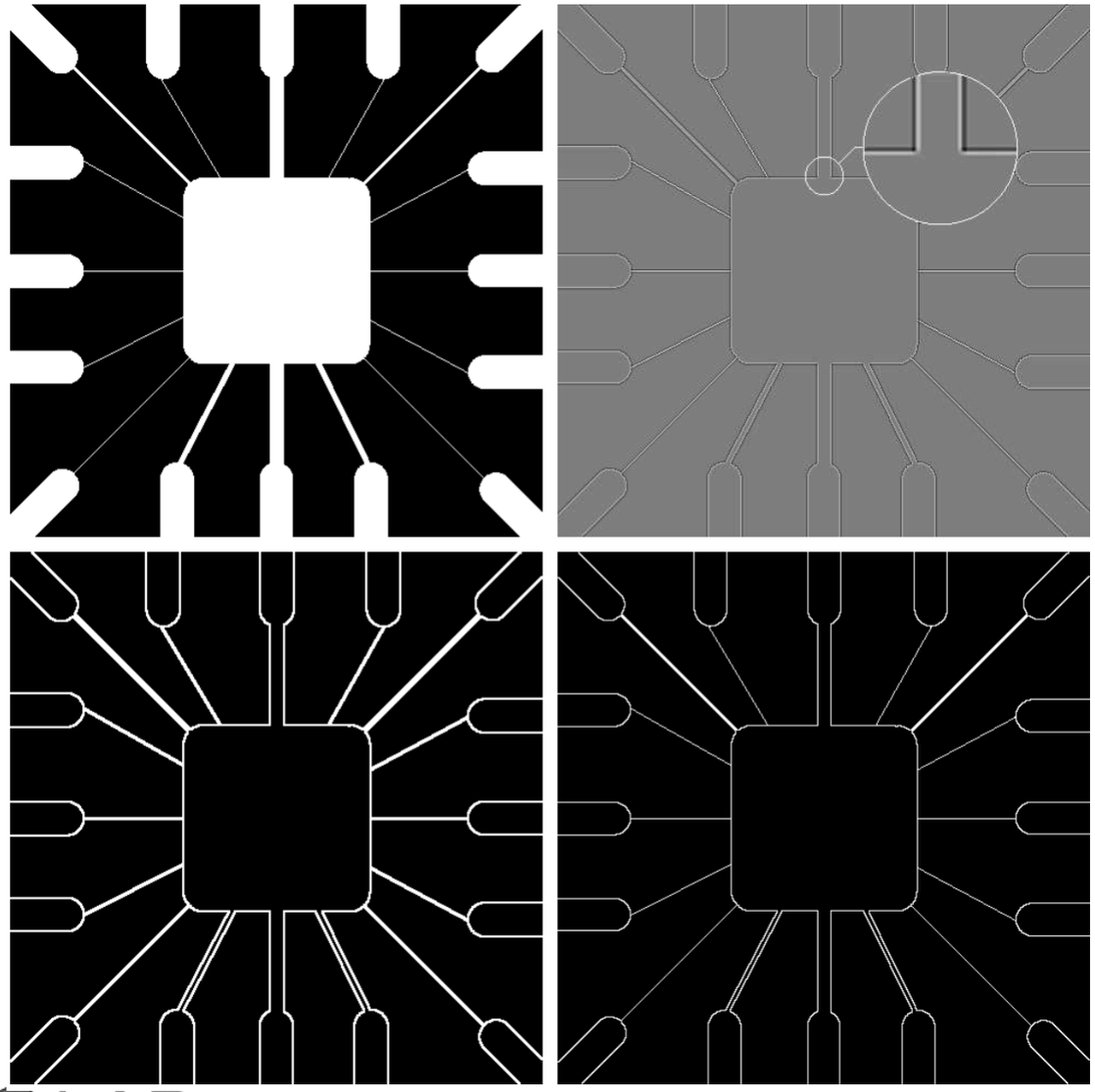
- Line Detection

  -> Based on the preceding content, we can expect second derivative (Laplacian) to result in a stronger response and produce thinner lines than first derivative

# Point, Line, Edge Detection

- Line Detection

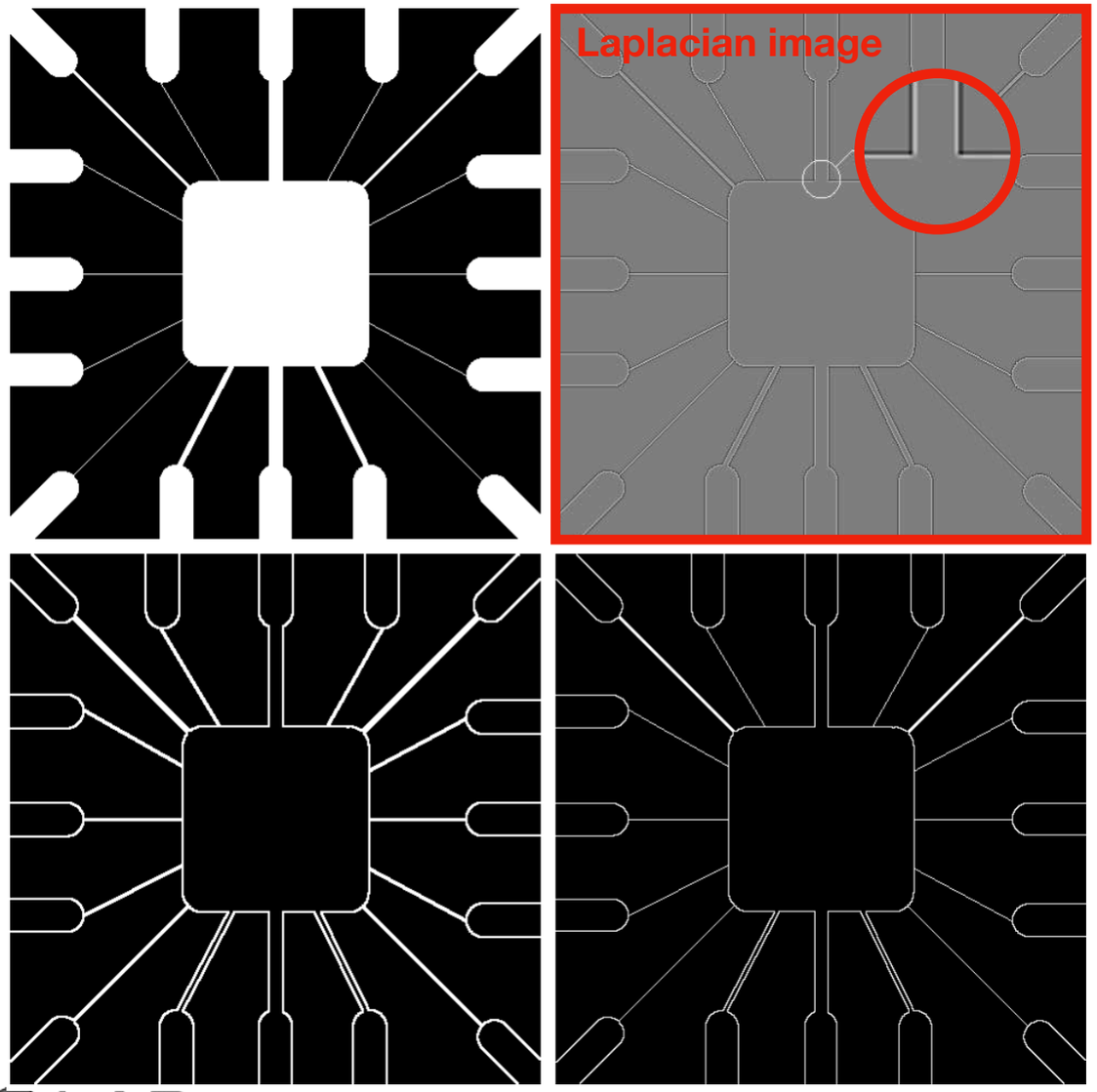
  -> Based on the preceding content, we can expect second derivative (Laplacian) to result in a stronger response and produce thinner lines than first derivative



# Point, Line, Edge Detection

- Line Detection

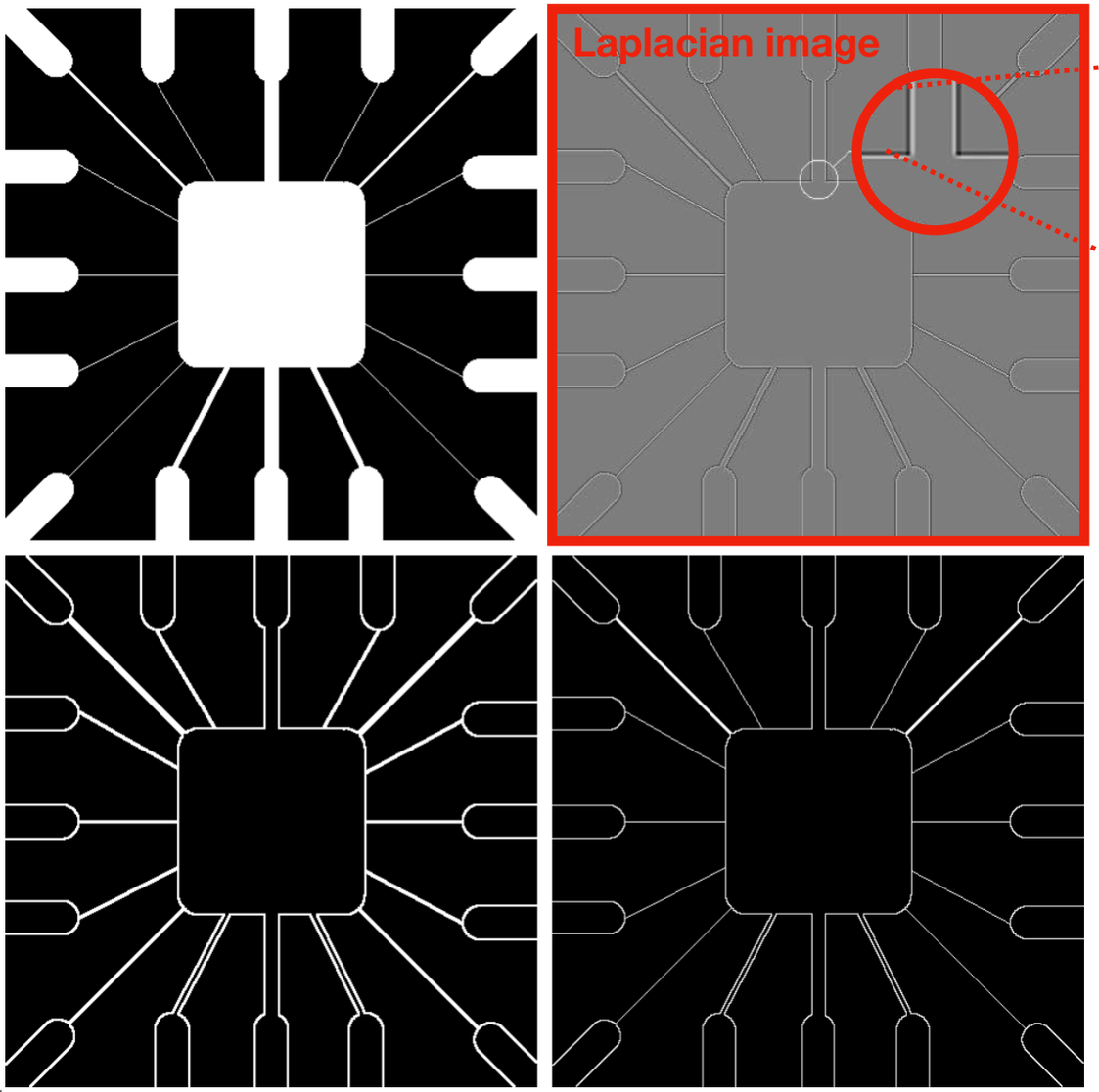
  -> Based on the preceding content, we can expect second derivative (Laplacian) to result in a stronger response and produce thinner lines than first derivative

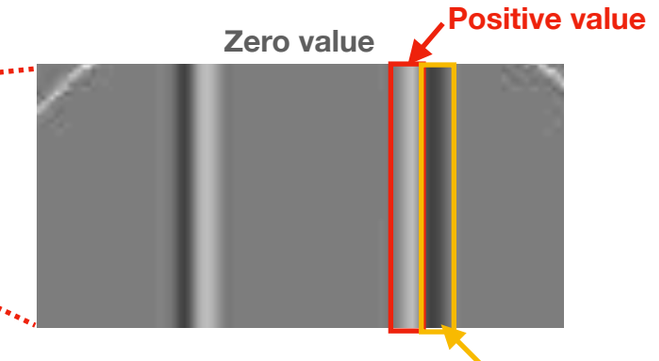


# Point, Line, Edge Detection

- Line Detection

  -> Based on the preceding content, we can expect second derivative (Laplacian) to result in a stronger response and produce thinner lines than first derivative



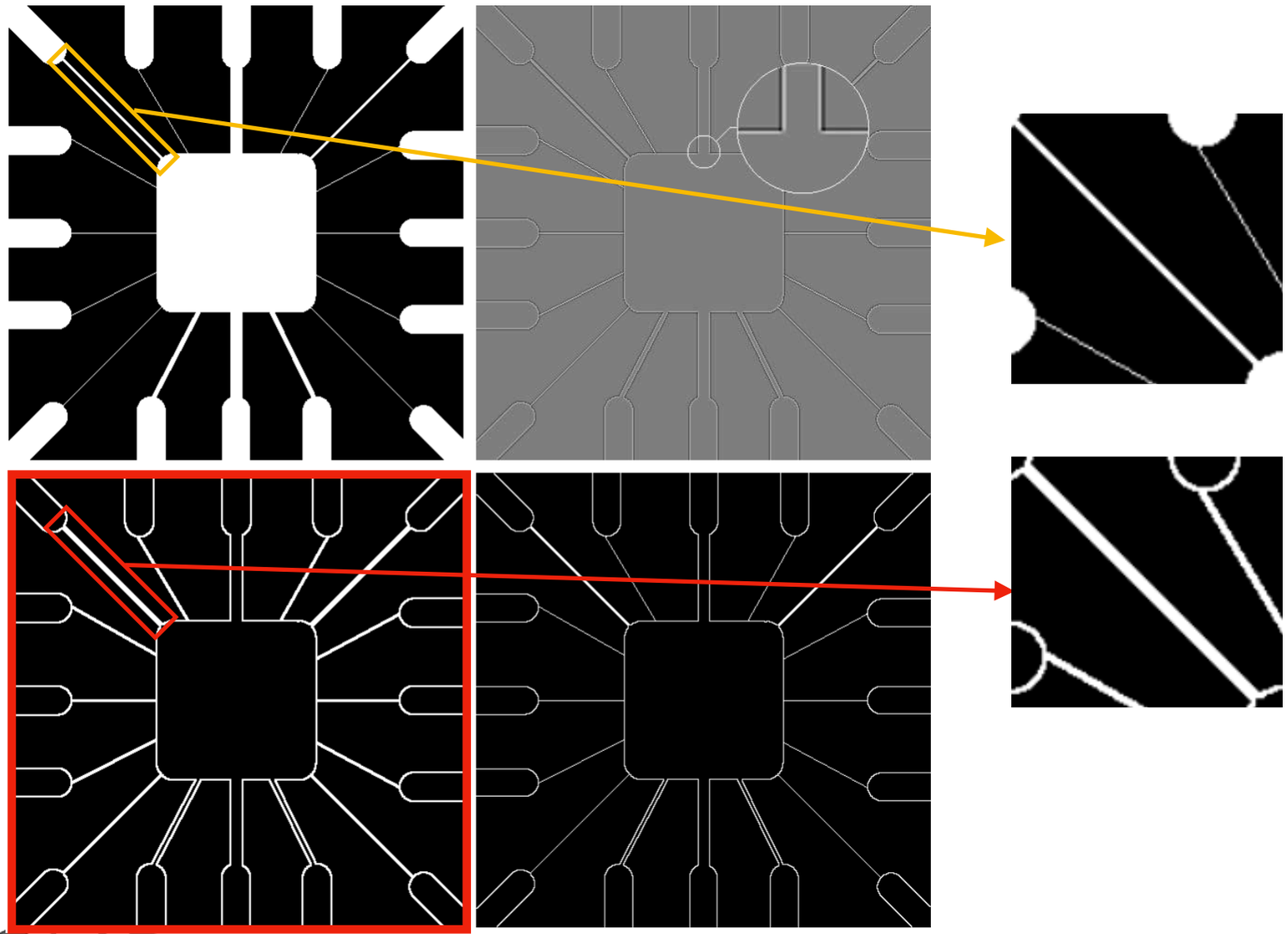




**Double Line effect**
**-> Caused by negative value contained in Laplacian image**

# Point, Line, Edge Detection

- Line Detection

  -> Based on the preceding content, we can expect second derivative (Laplacian) to result in a stronger response and produce thinner lines than first derivative

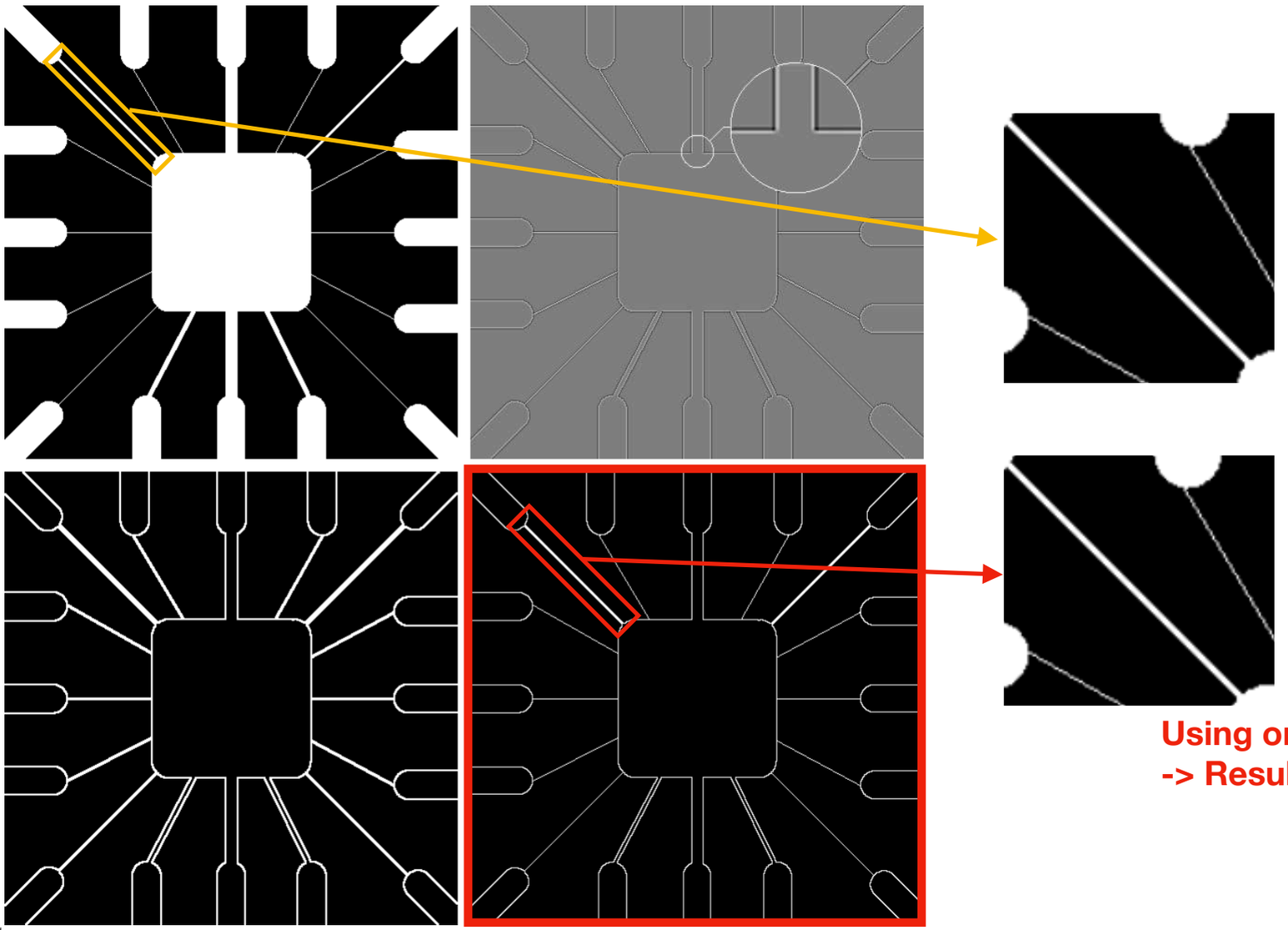


**Absolute Value of Laplacian**
**-> Double Thickness Problem**

# Point, Line, Edge Detection

- Line Detection

  -> Based on the preceding content, we can expect second derivative (Laplacian) to result in a stronger response and produce thinner lines than first derivative

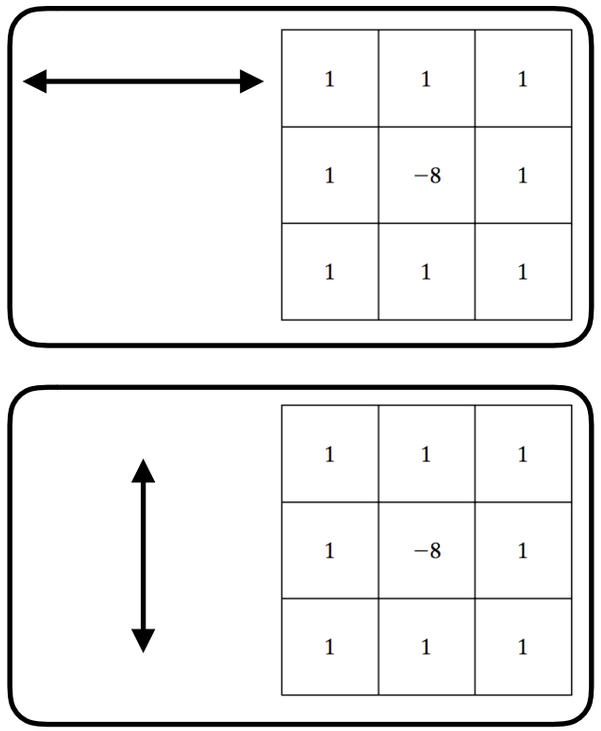


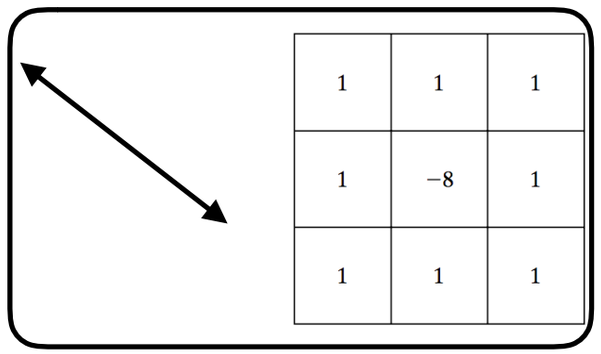
**Using only the Positive Values of Laplacian**
**-> Results in thinner lines**

# Point, Line, Edge Detection

- Line Detection
  -> Laplacian detector has Isotropic property,
  so its response is independent of direction

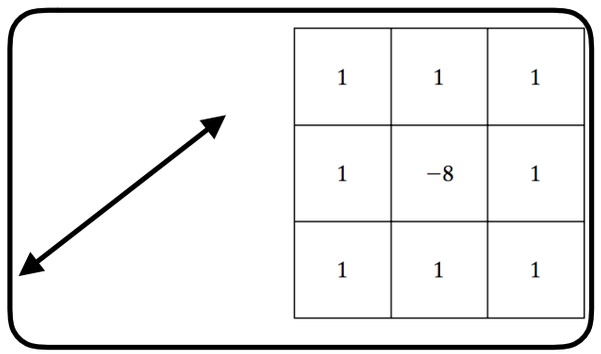
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |





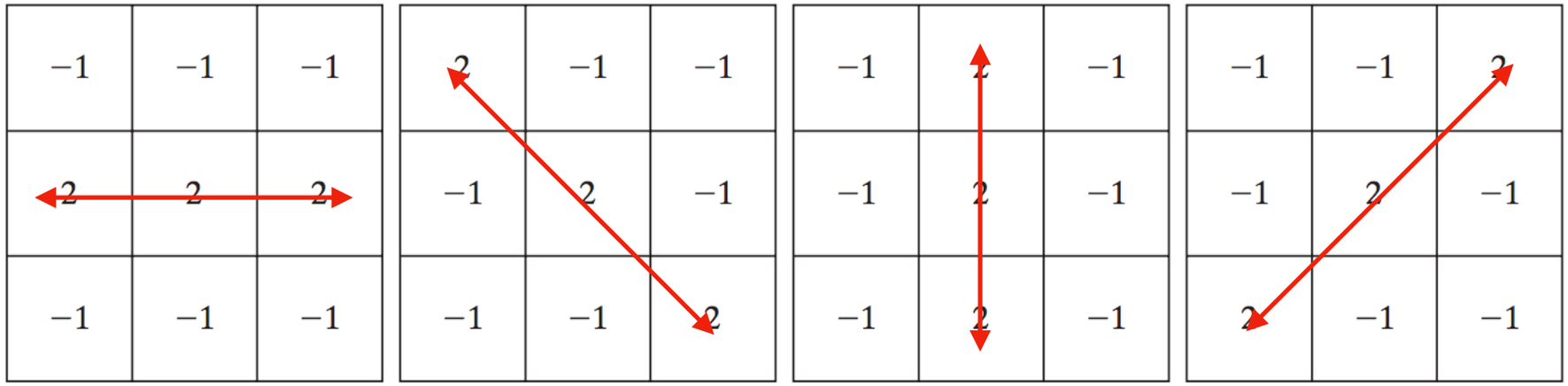




# Point, Line, Edge Detection

- Line Detection
  -> Laplacian detector has Isotropic property,
  so its response is independent of direction

  -> So, we can consider underlying masks
  if interest in detecting lines in specified directions

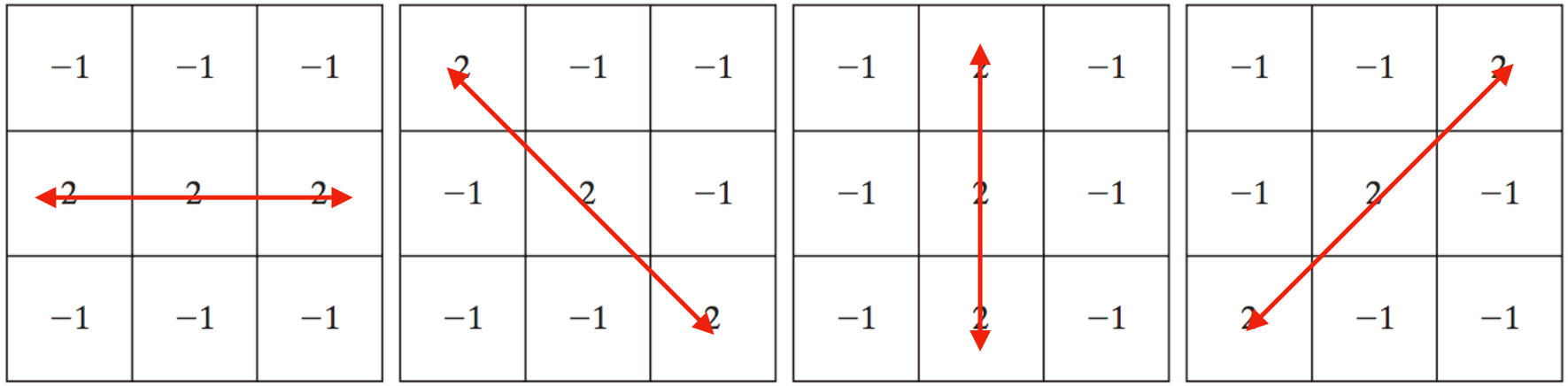


# Point, Line, Edge Detection

- Line Detection

  -> Laplacian detector has Isotropic property,
  so its response is independent of direction

  -> So, we can consider underlying masks
  if interest in detecting lines in specified directions




$|R_k| > |R_j|, \quad for \ all \ j \neq k$ → **The point is associated with a line in the direction of k**

# Point, Line, Edge Detection

- Line Detection

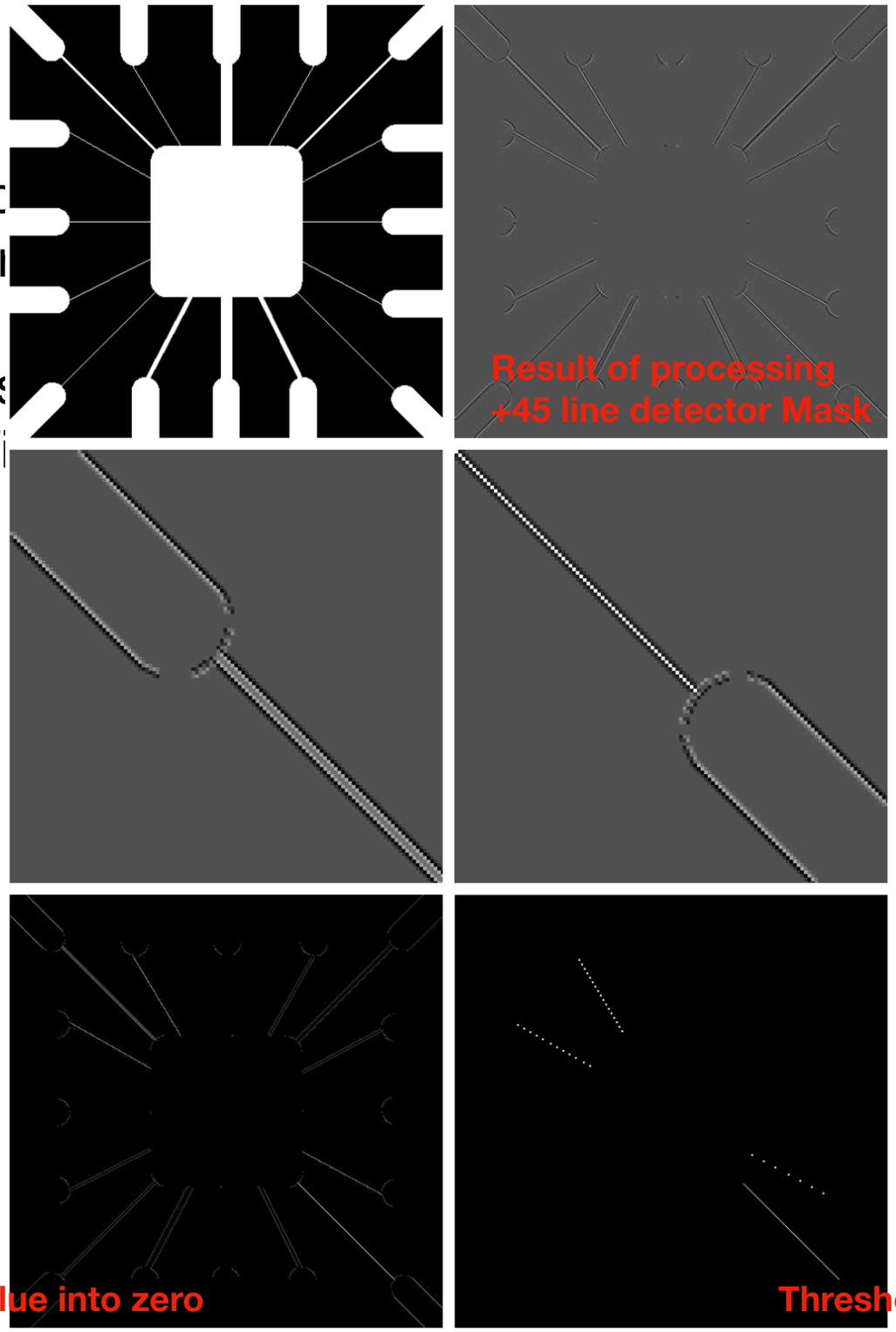
  -> Laplacian detector has Isotropic pro
  so its response is independent of dir

  -> So, we can consider underlying mas
  if interest in detecting lines in specifi

| | | |
|---|---|---|
| 2 | −1 | −1 |
| −1 | 2 | −1 |
| −1 | −1 | 2 |

+45°

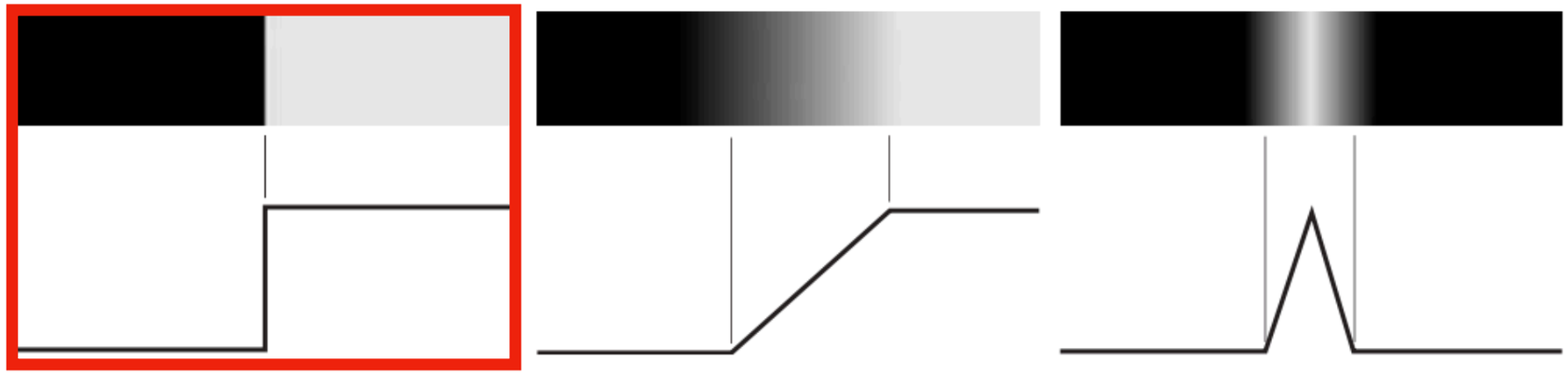
**Use '+45 degrees' Mask to detect the lines rotated into +45 degrees**



# Point, Line, Edge Detection

- Edge Model

Step Edges



Ramp Edges

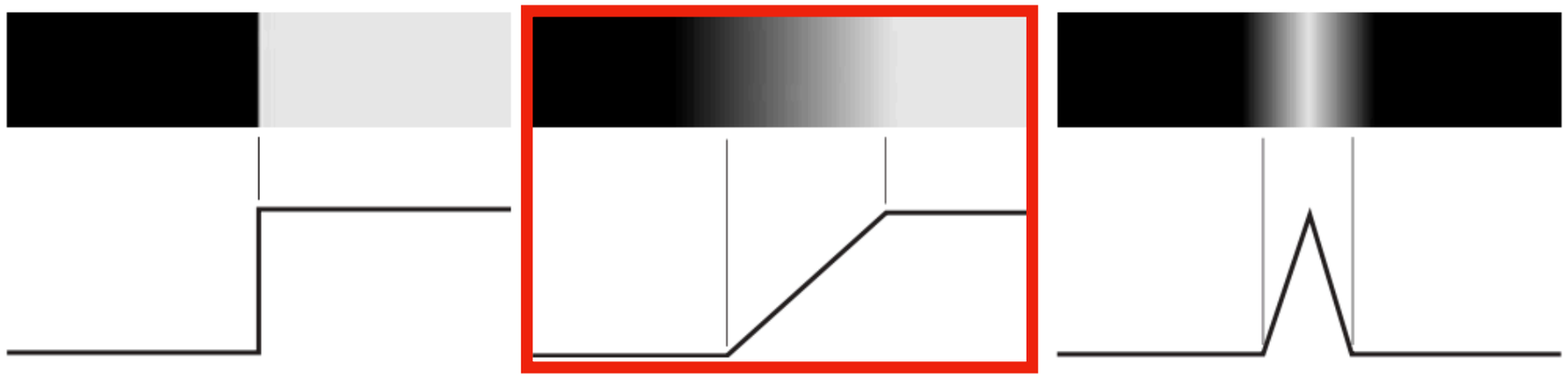
Roof Edges

- Involve transition between two intensity levels
- Occurring ideally over the distance of 1 pixel
- Generated by a computer for use in area such as solid modeling and animation

# Point, Line, Edge Detection

- Edge Model

Step Edges



Ramp Edges

- Edges that are blurred and noisy

Roof Edges

- Degree of blurring determined principally by limitation in focusing mechanism

- Degree of noise level determined principally by the electronic components of the imaging system

- Slope of the ramp is inversely proportional to the degree of blurring

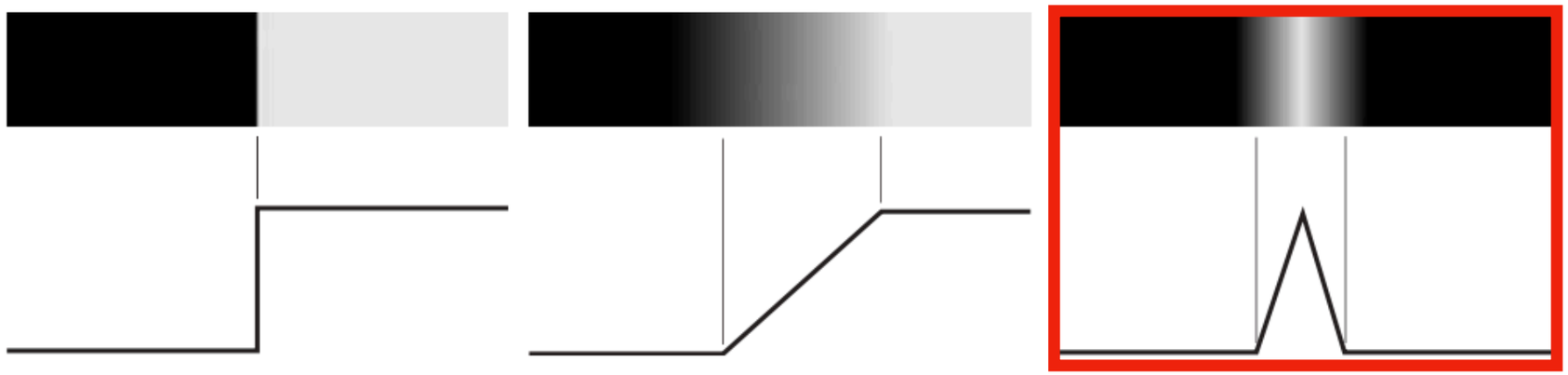
# Point, Line, Edge Detection

- Edge Model

Step Edges

Ramp Edges

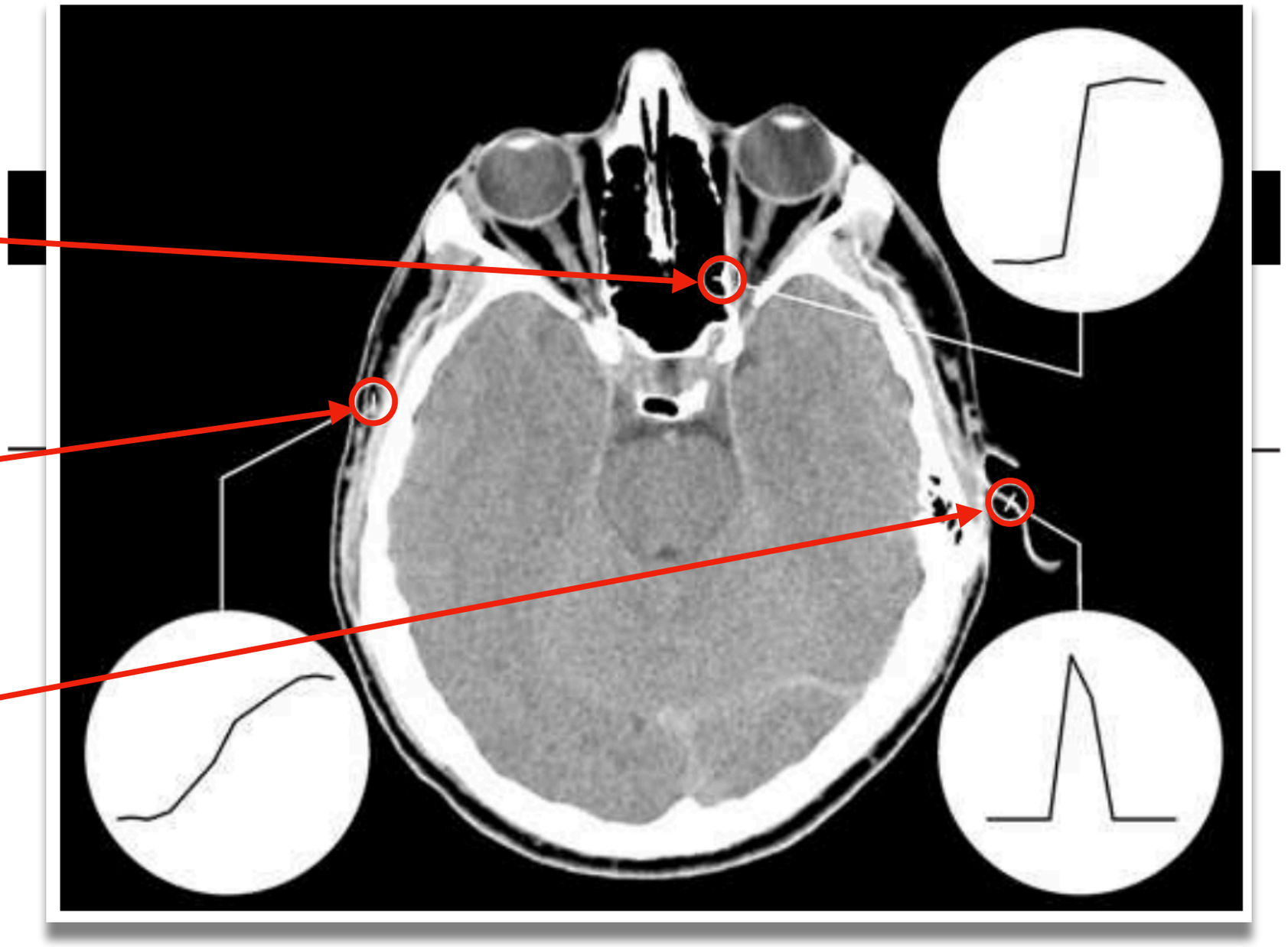
Roof Edges



- Roof edges are models of lines through of region with the base of roof edge being determined by thickness and sharpness of the line

# Point, Line, Edge Detection

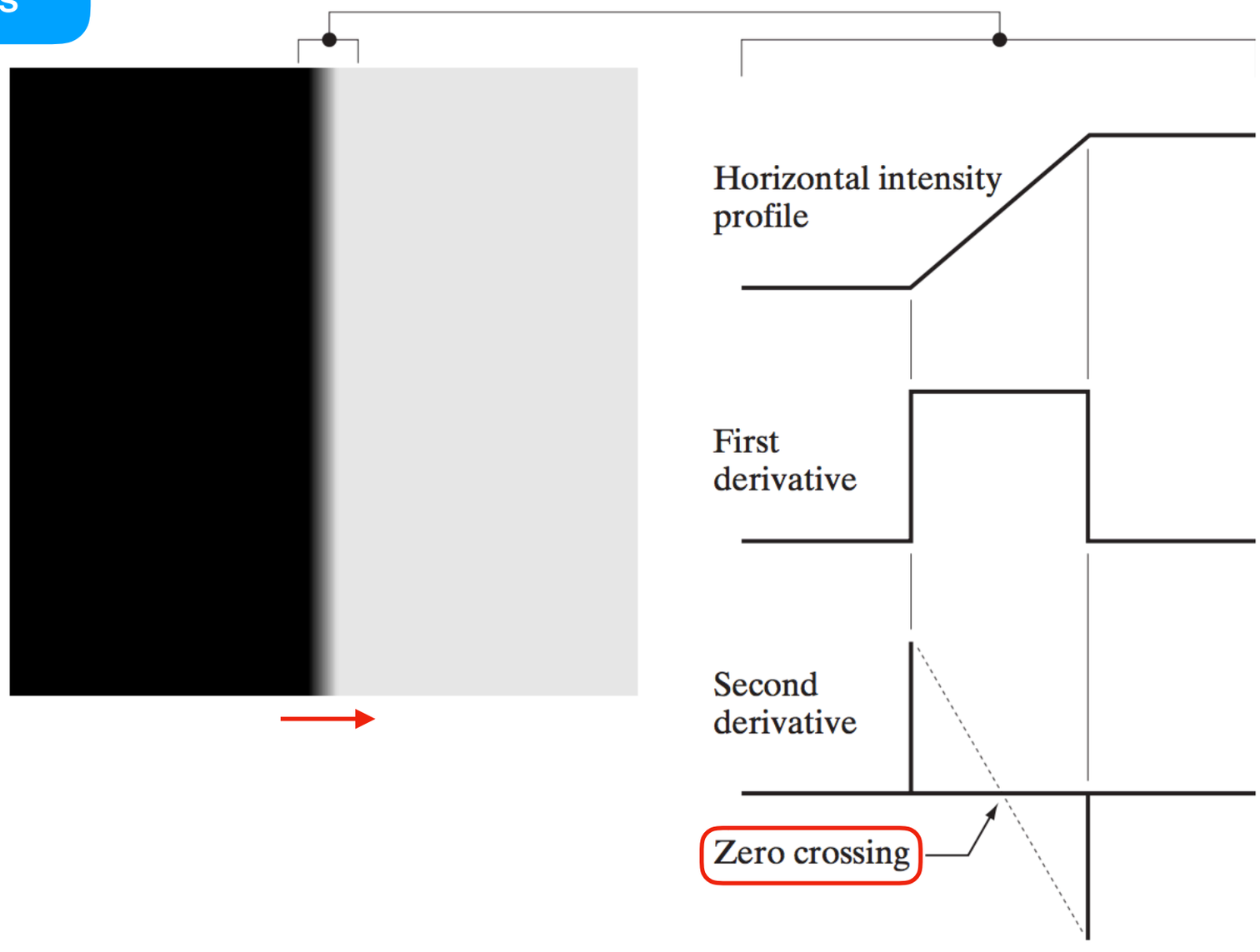
- Edge Model



# Point, Line, Edge Detection

- Edge Model

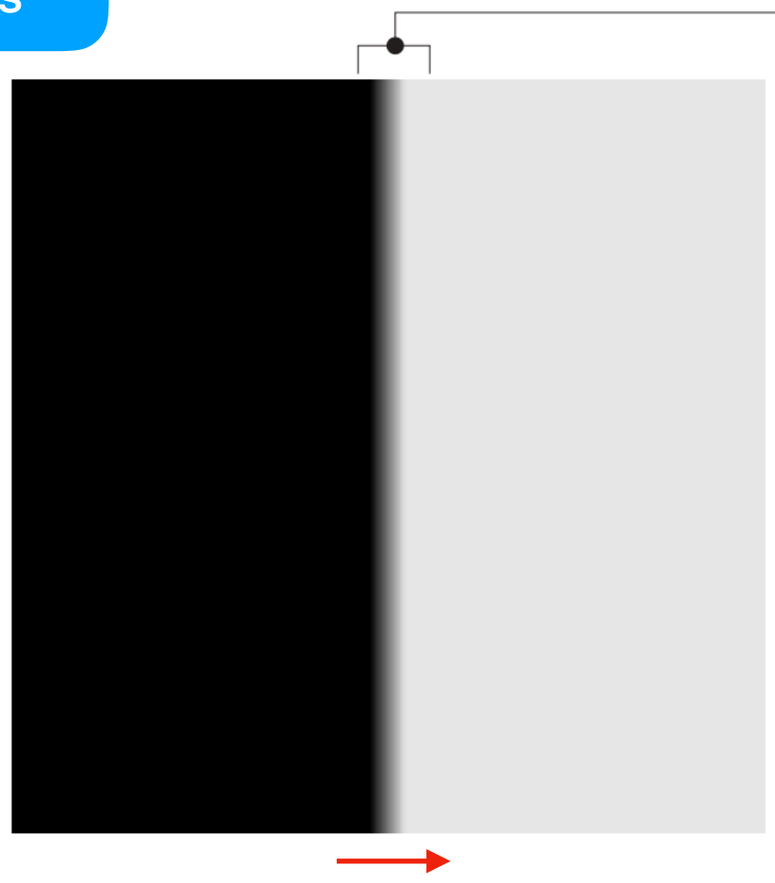
Ramp Edges

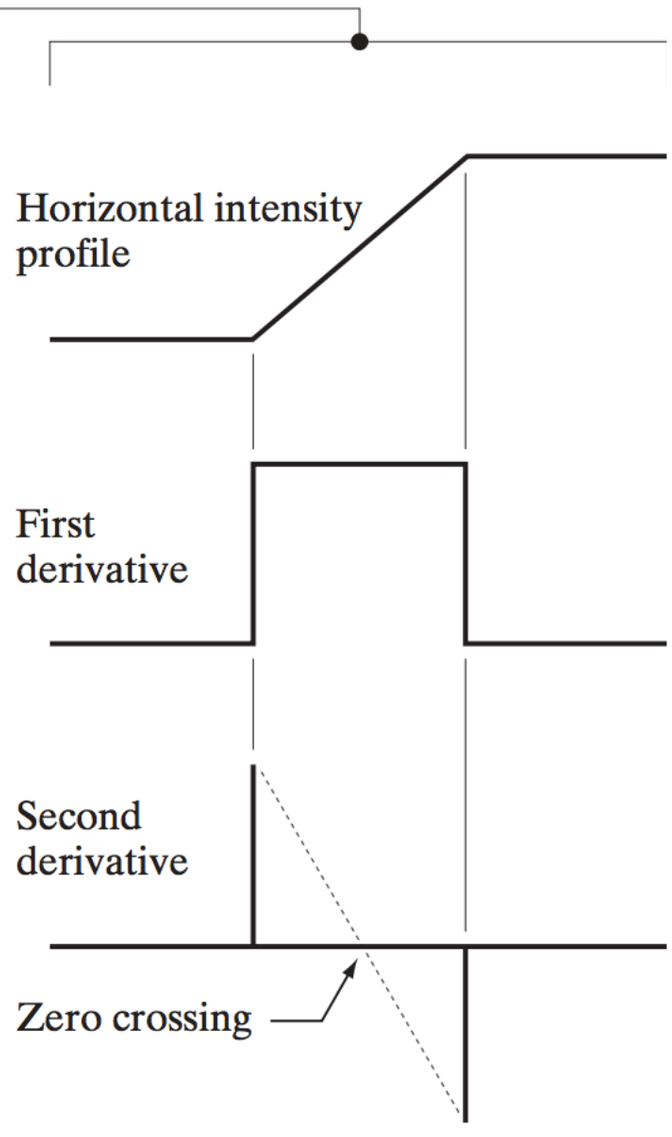


# Point, Line, Edge Detection

- Edge Model

Ramp Edges





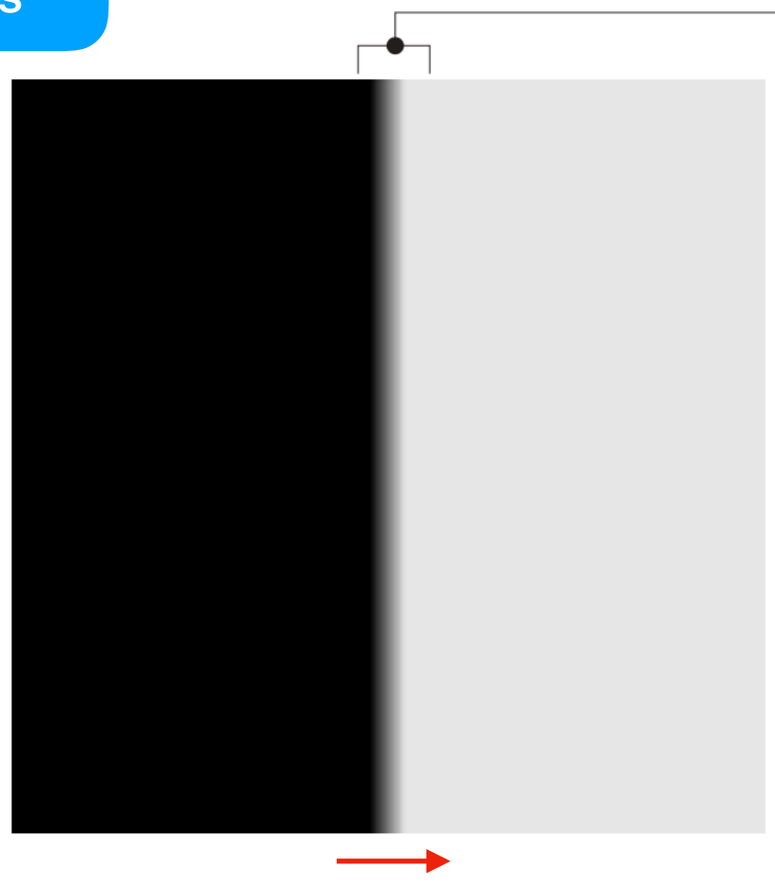

**Magnitude of the first derivative can be used to detect the presence of an edge at the point in an image**

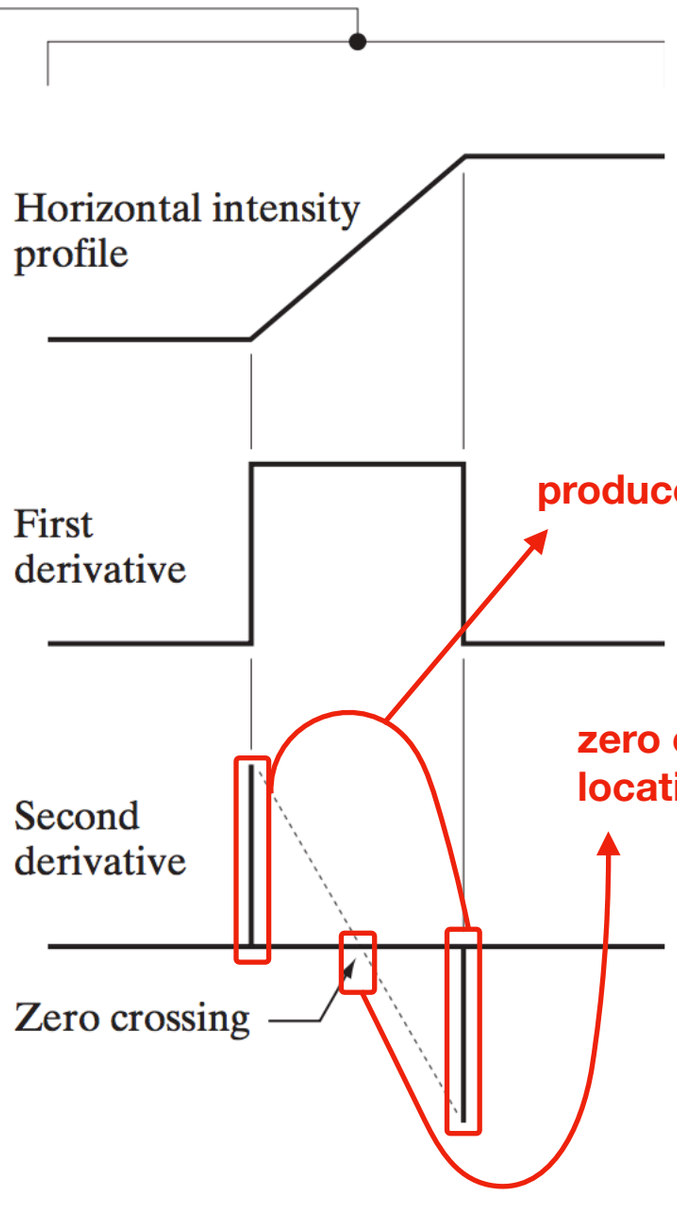
**Sign of second derivative can be used to determine whether an edge pixel lies on the dark or light side of an edge**

# Point, Line, Edge Detection

- Edge Model

Ramp Edges

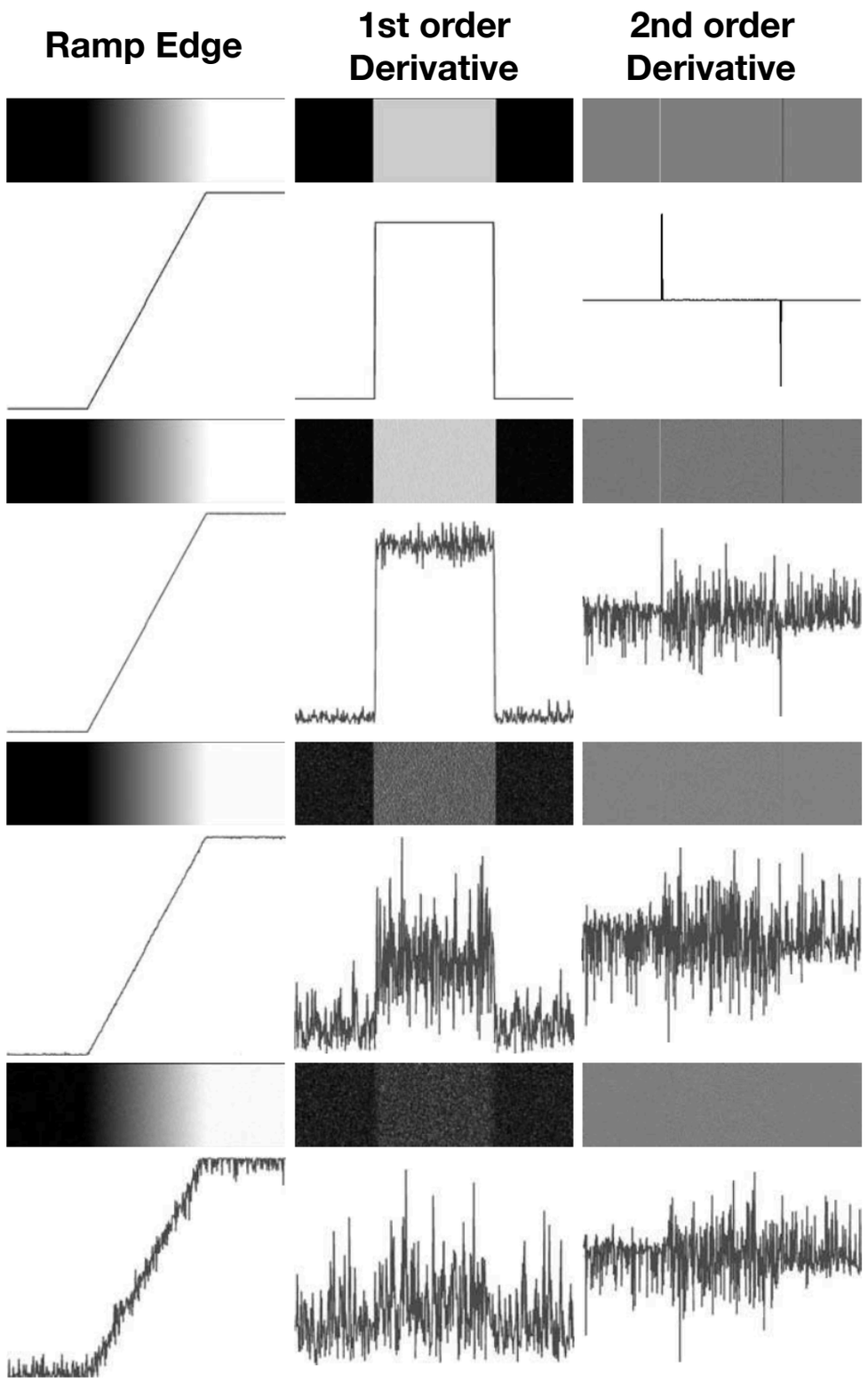




# Point, Line, Edge Detection

- Edge Model

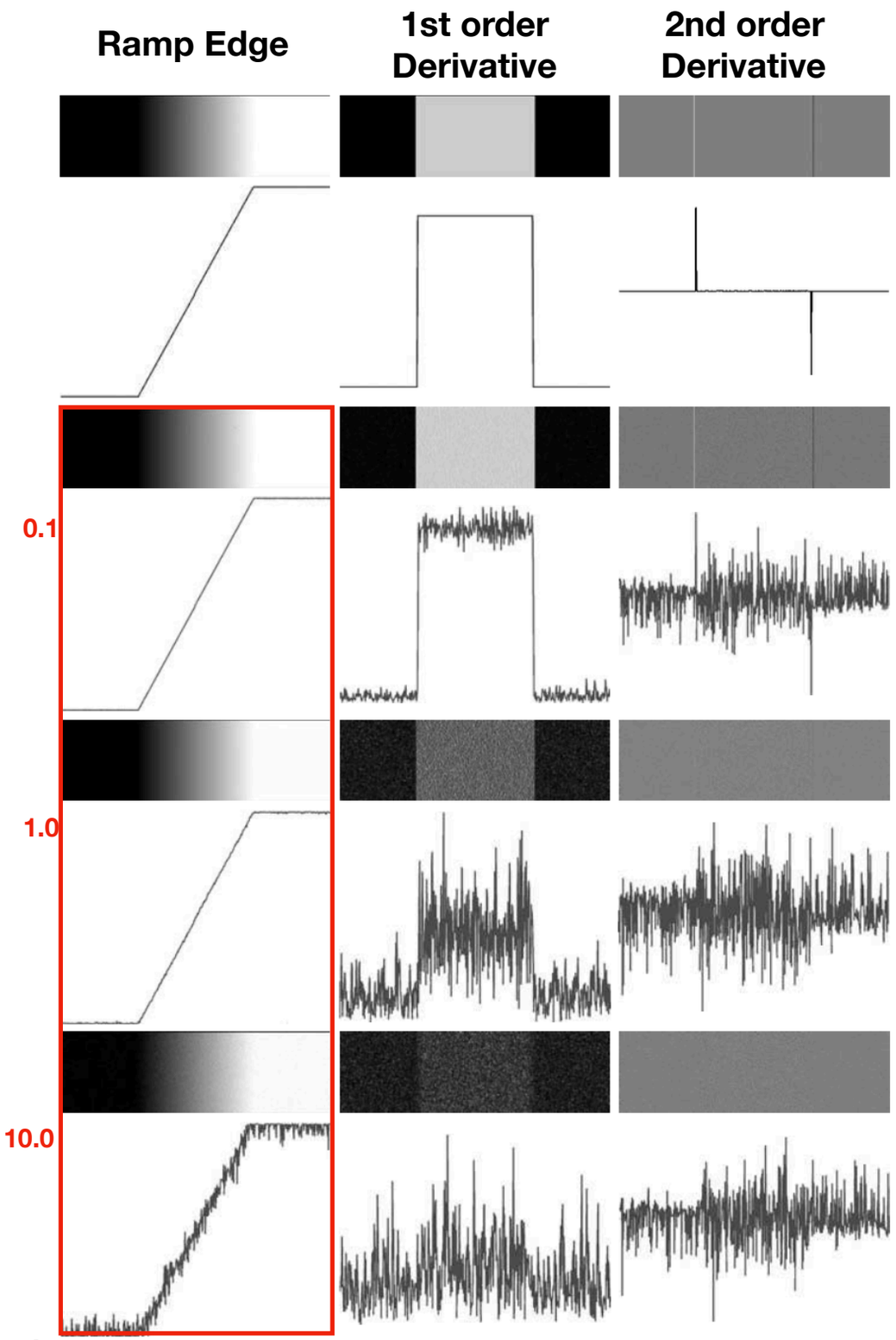
  Ramp Edges



# Point, Line, Edge Detection

- Edge Model

Ramp Edges

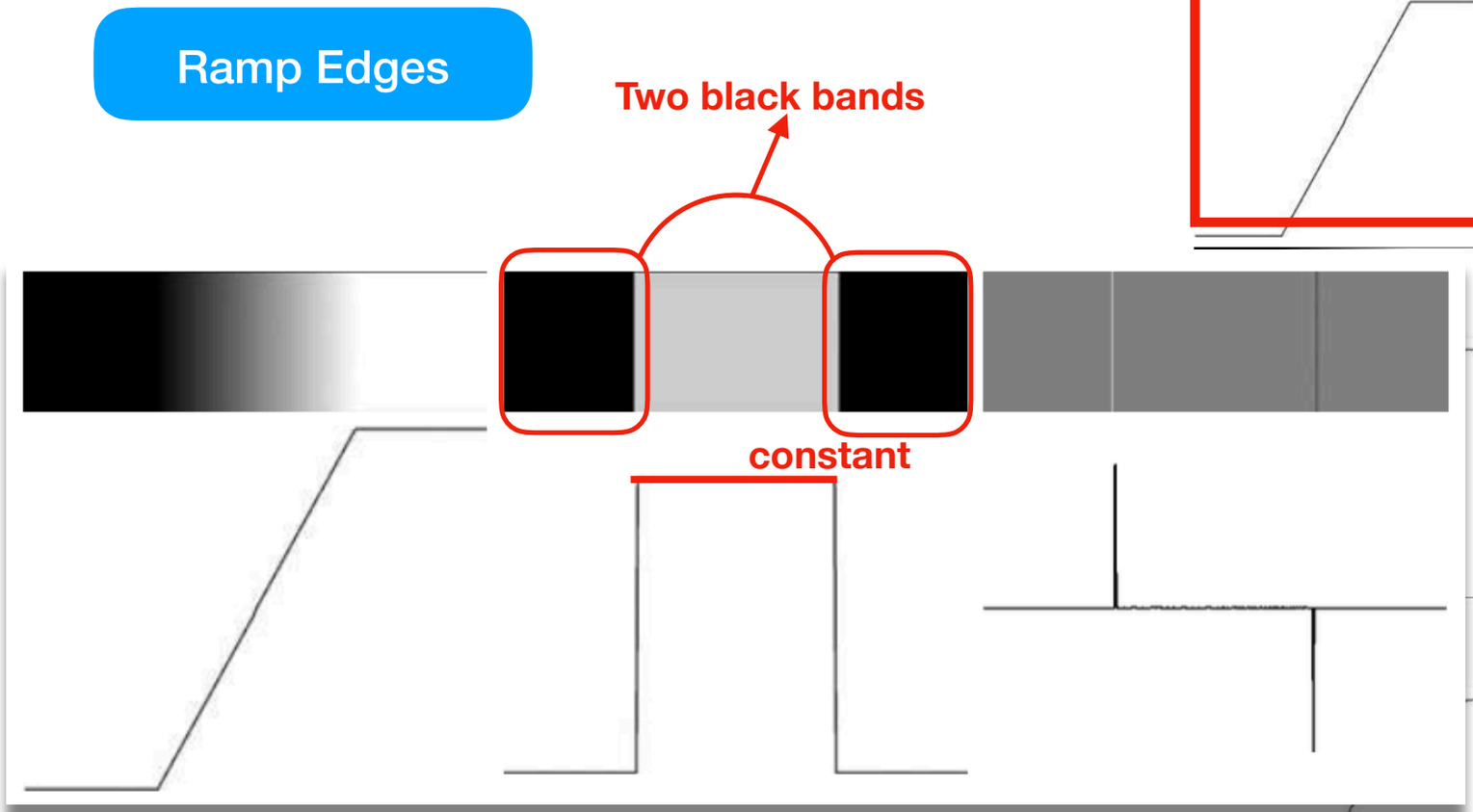


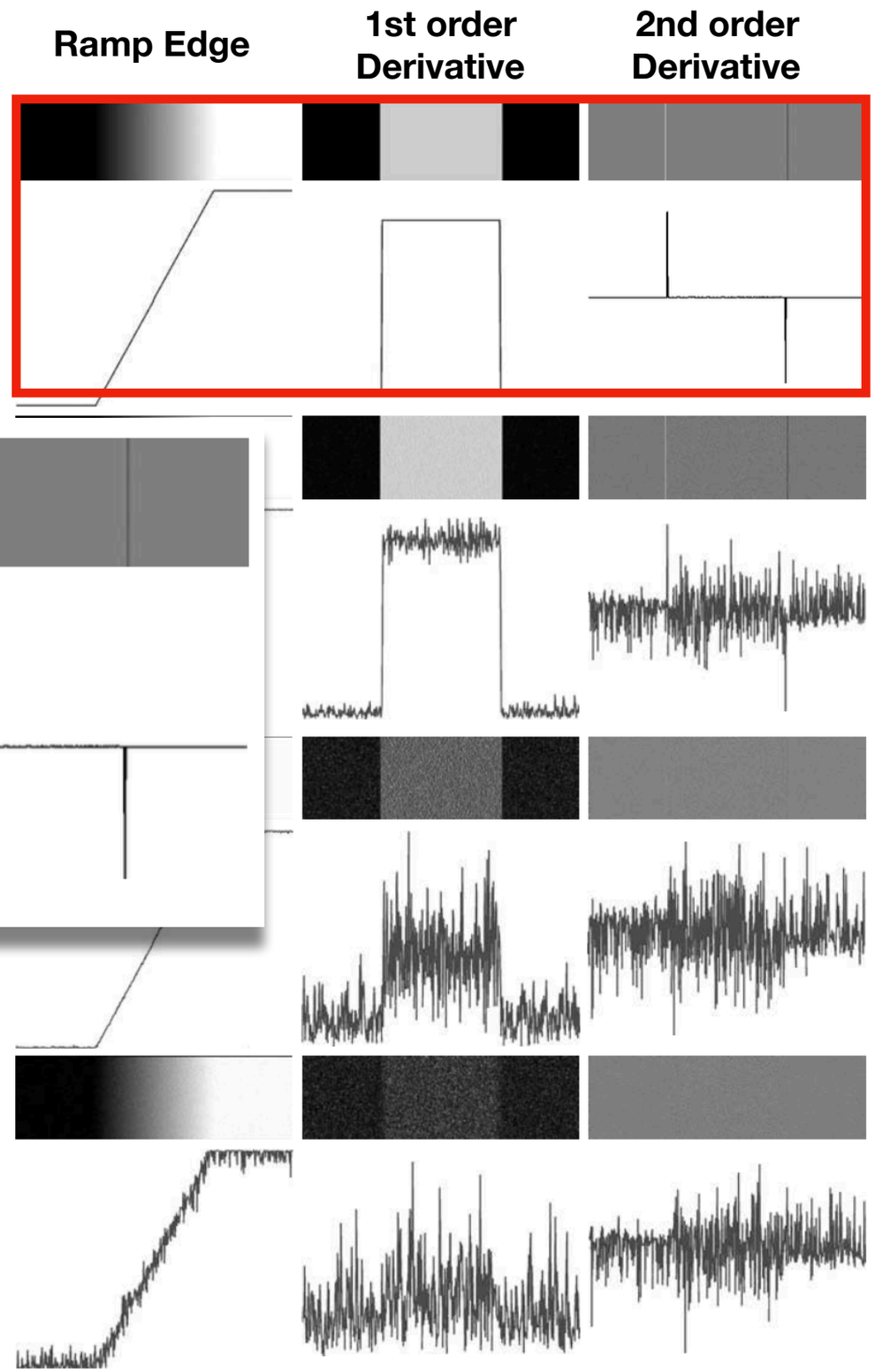

**corrupted by additive Gaussian Noise with mean and std of case 0.1, 1.0, 10.0**

# Point, Line, Edge Detection

- Edge Model

Ramp Edges

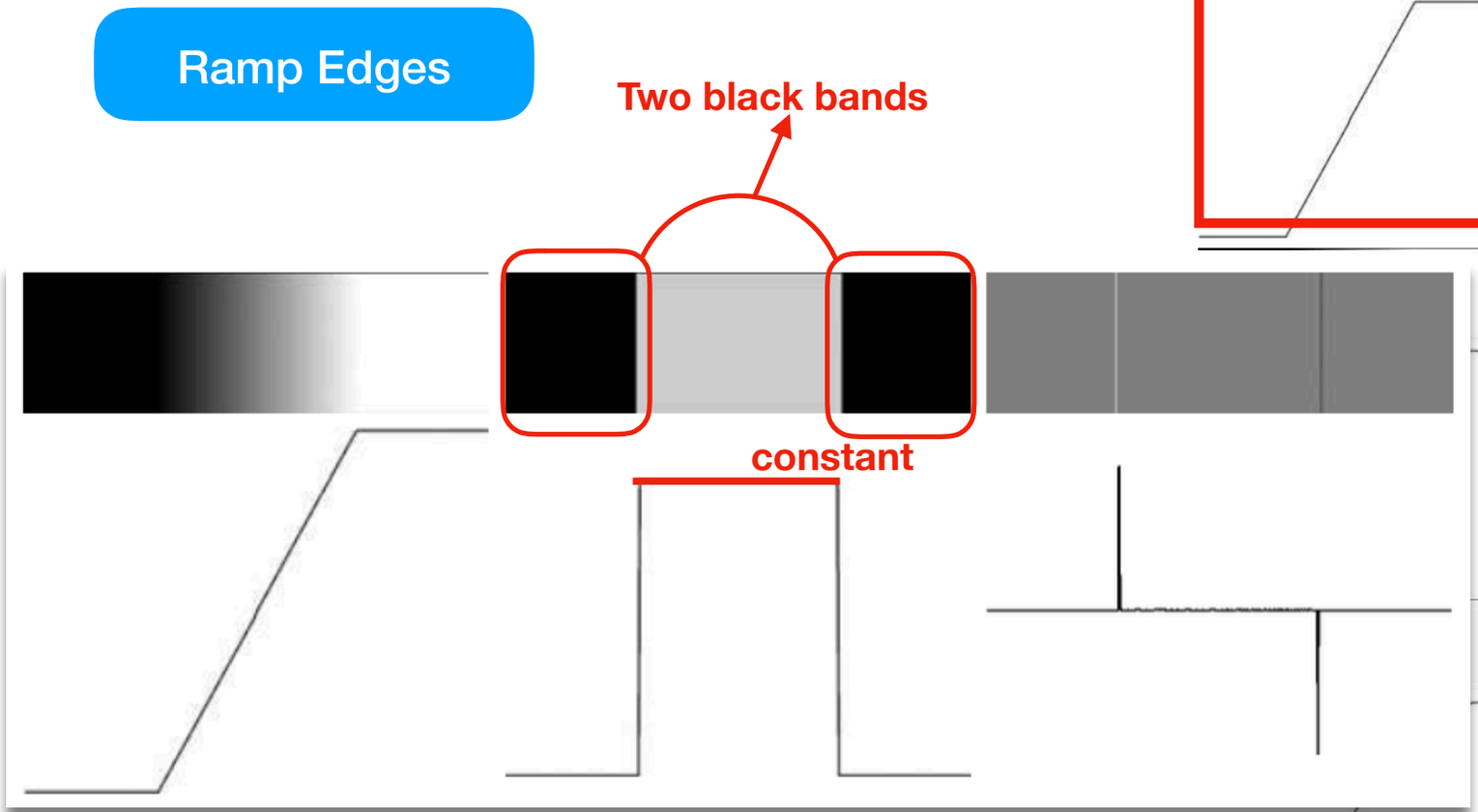


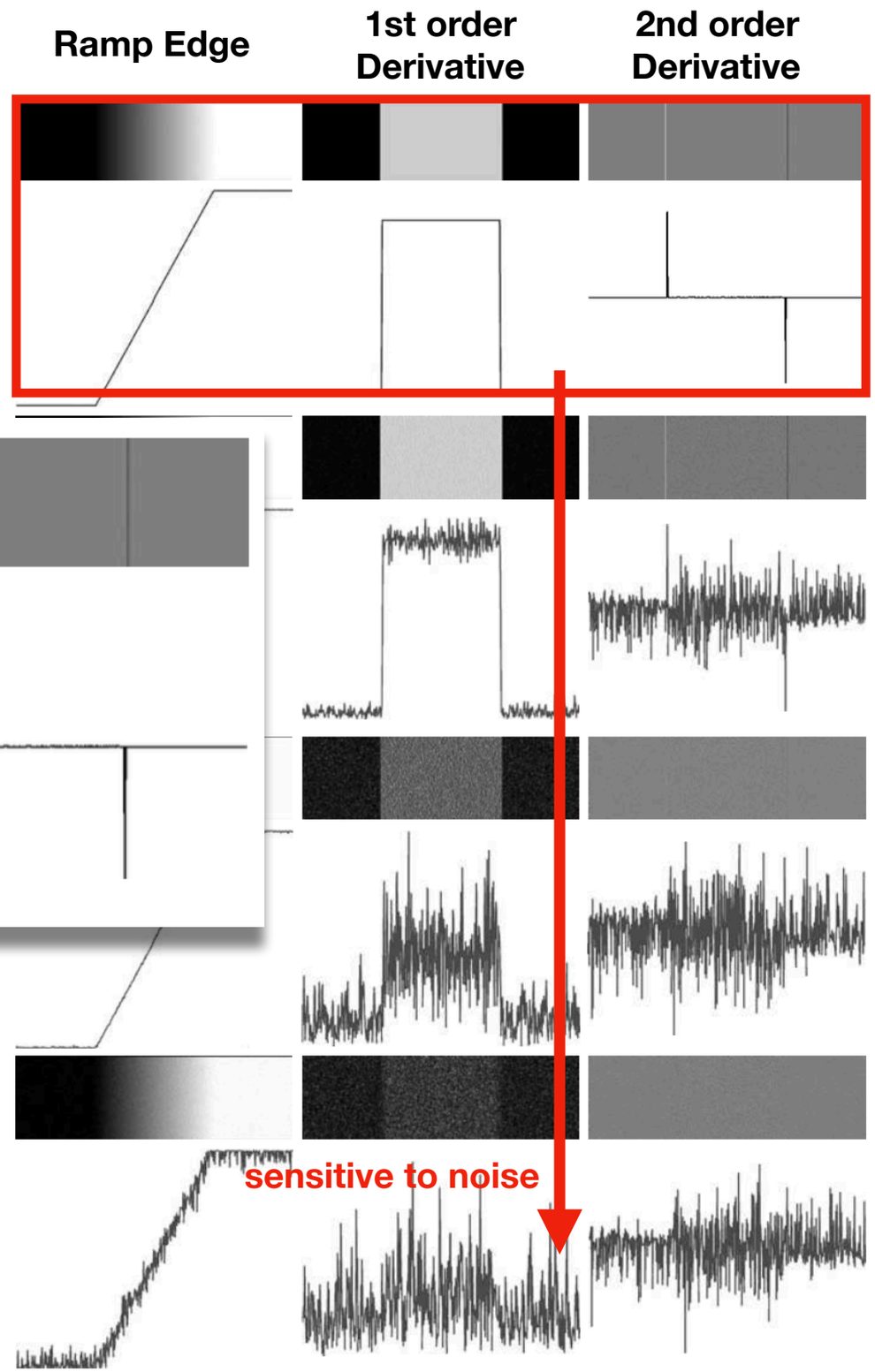



# Point, Line, Edge Detection

- Edge Model

Ramp Edges

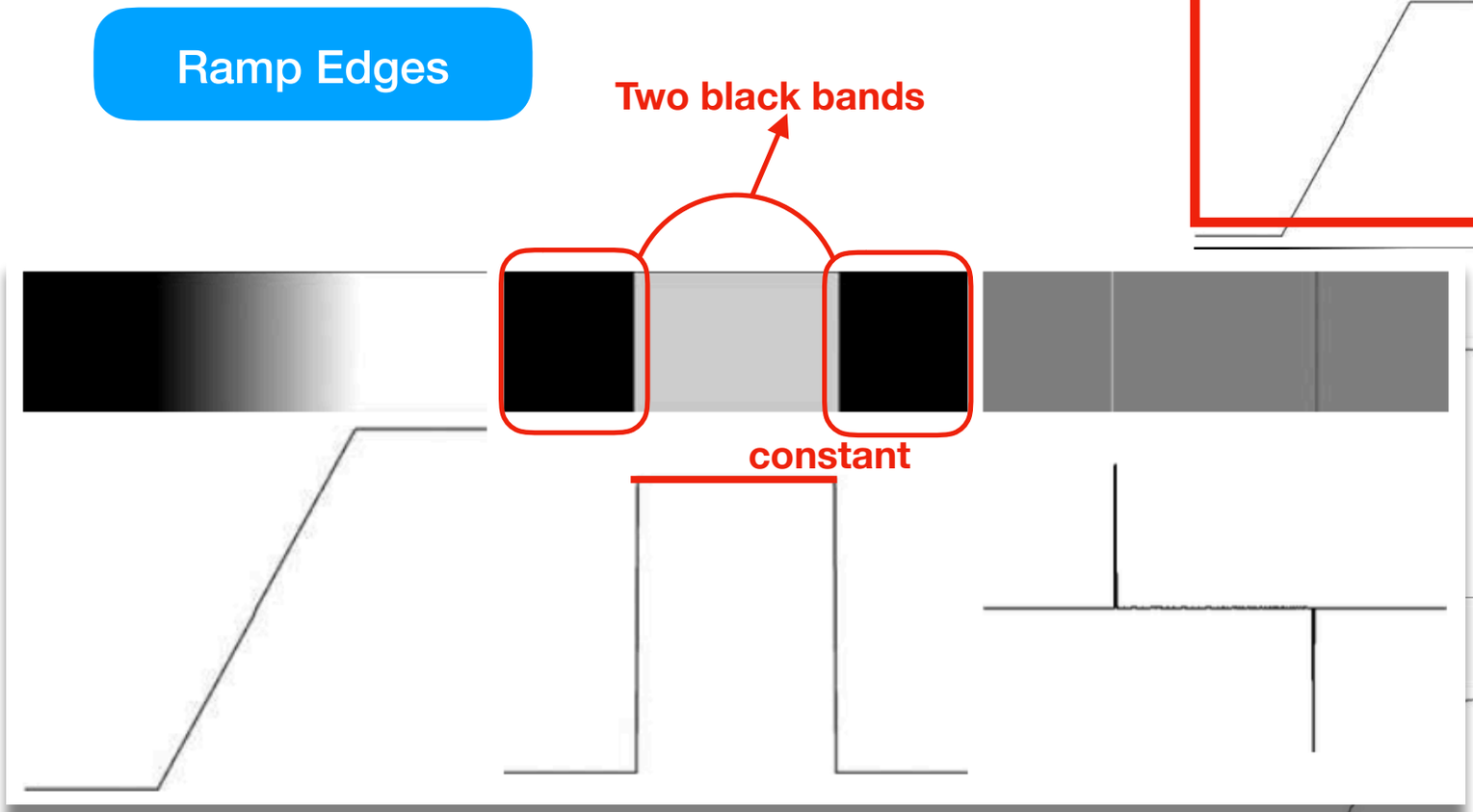


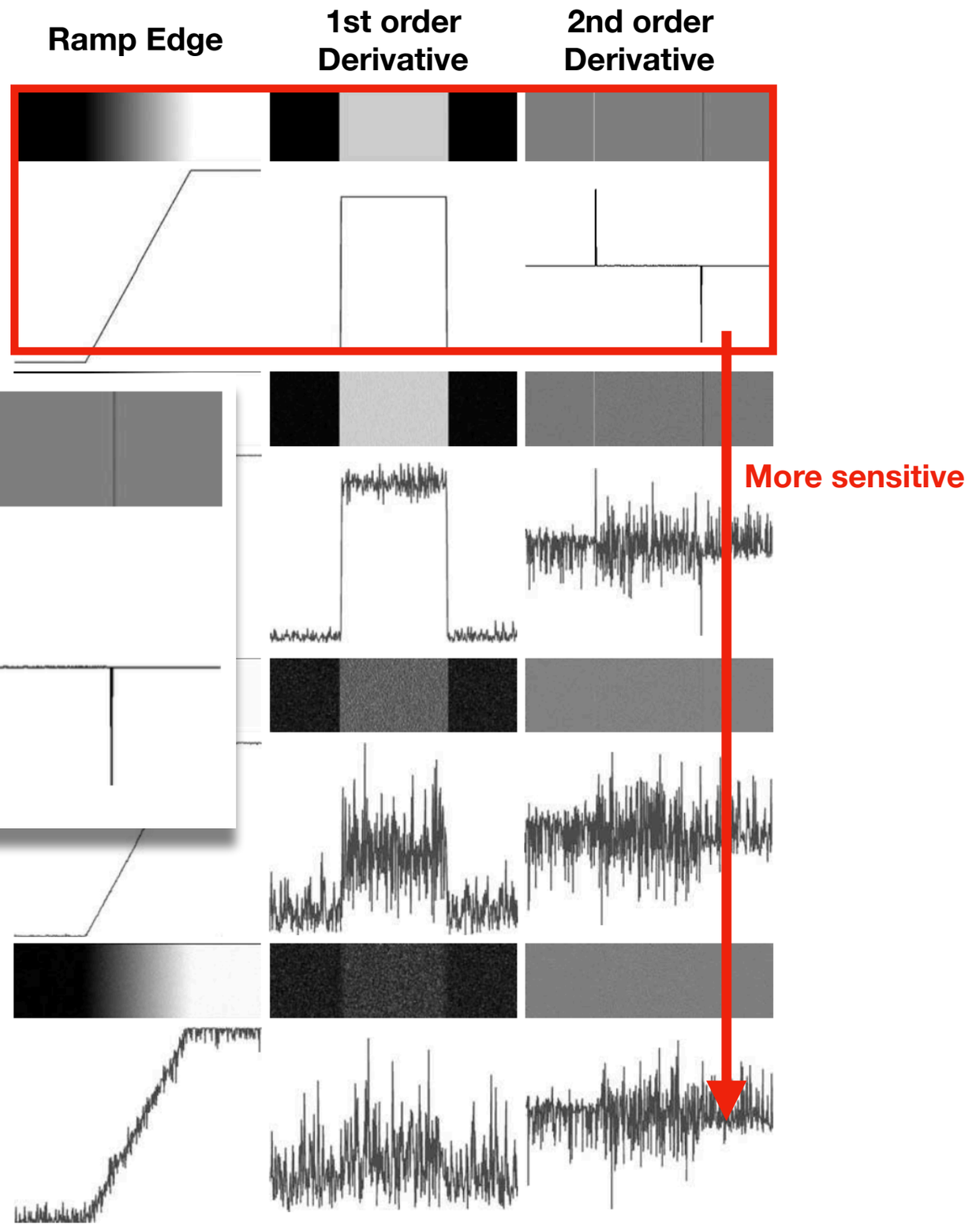



# Point, Line, Edge Detection

- Edge Model

Ramp Edges

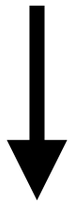


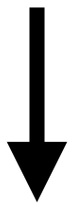
# Point, Line, Edge Detection

- Edge Model
  -> Three fundamental steps performed in edge detection

**Image Smoothing for Noise reduction**



**Detection of Edge points**



**Edge Localization**

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Discuss about First-order Derivatives

  => For finding edge strength and direction at location (x,y) with gradient of f,

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- **Magnitude (length)** of vector Nebular f

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

- **Direction** of the gradient vector

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

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Discuss about First-order Derivatives

  => For finding edge strength and direction at location (x,y) with gradient of f,

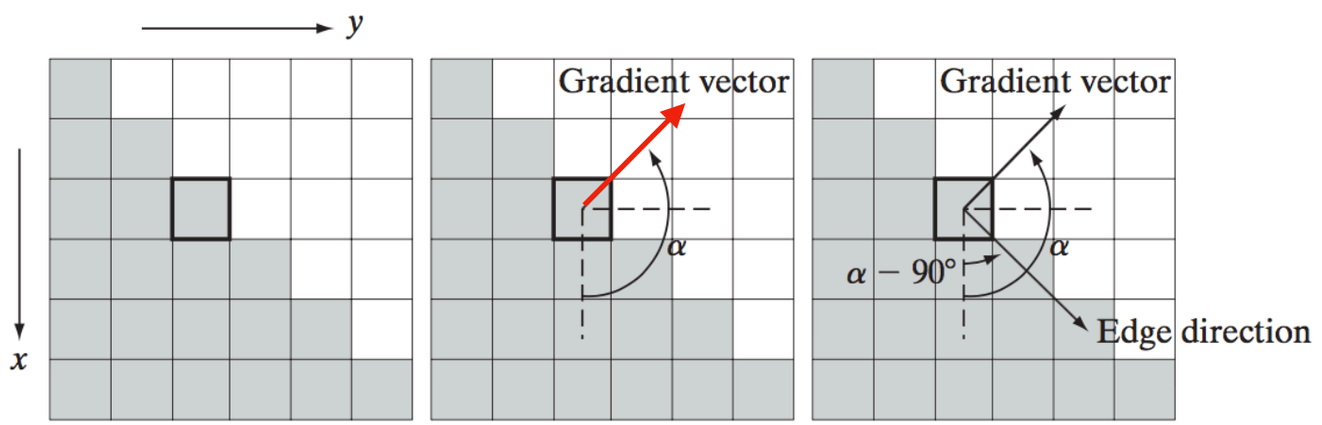
$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- **Magnitude (length)** of vector Nebular f

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

- **Direction** of the gradient vector

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



# Point, Line, Edge Detection

- Basic Edge Detection

  -> Discuss about First-order Derivatives

  => For finding edge strength and direction at location (x,y) with gradient of f,

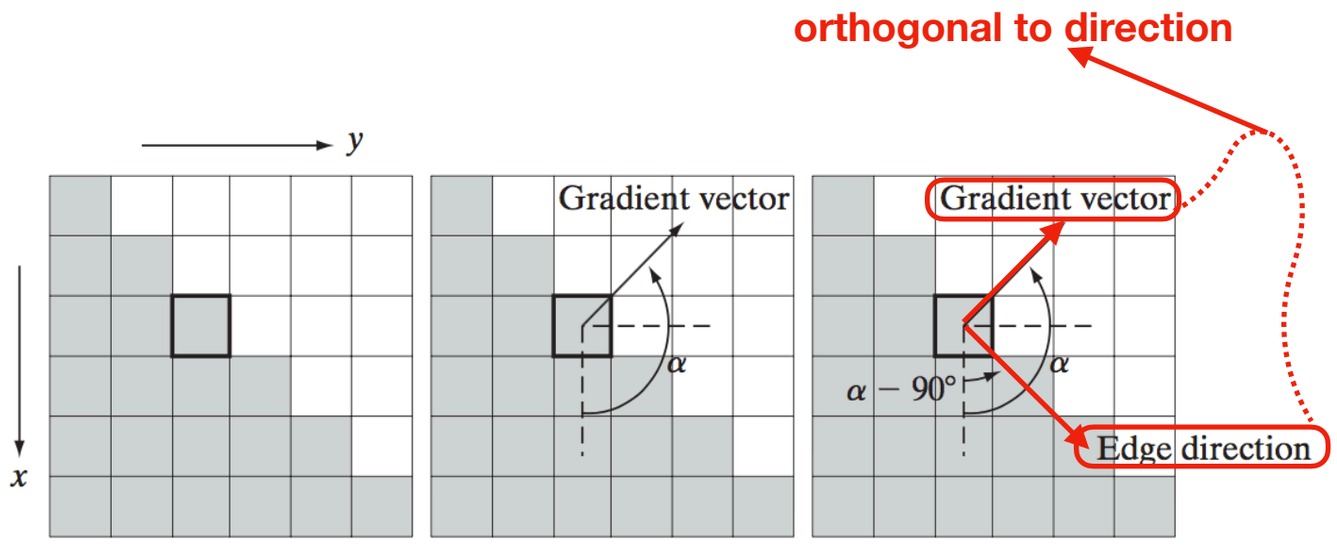
$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- **Magnitude (length)** of vector Nebular f

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

- **Direction** of the gradient vector

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



# Point, Line, Edge Detection

- Basic Edge Detection

  -> Discuss about First-order Derivatives

  => For finding edge strength and direction at location (x,y) with gradient of f,

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

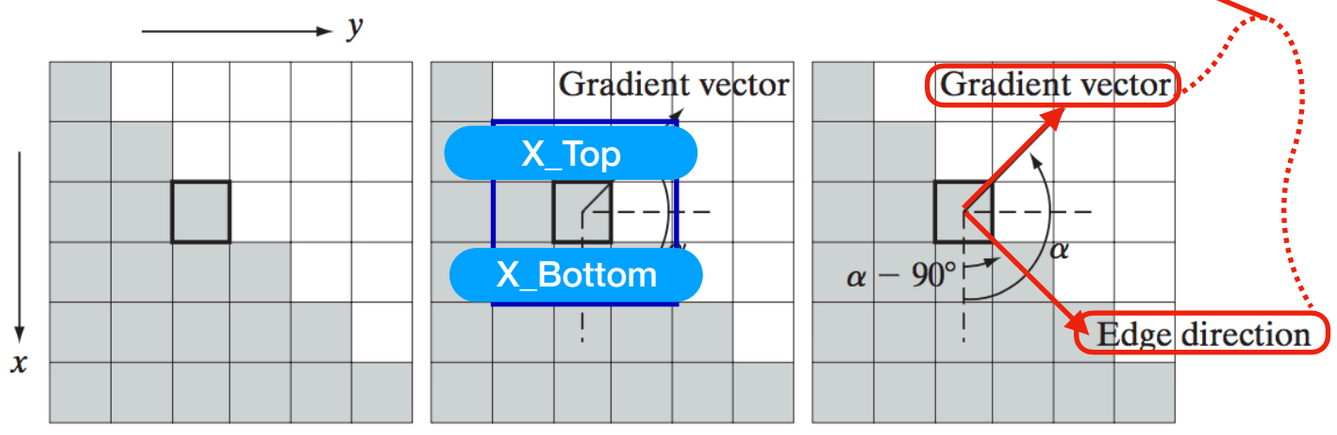
orthogonal to direction

- **Magnitude (length)** of vector Nebular f

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

- **Direction** of the gradient vector

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




**Suppose of pixels in gray have value of 0 and white have value of 1 and Computing derivatives in the x- and y-direction using 3x3 neighborhood centered**

**partial derivative in the x direction** = X_Bottom - X_Top

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Discuss about First-order Derivatives

  => For finding edge strength and direction at location (x,y) with gradient of f,

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

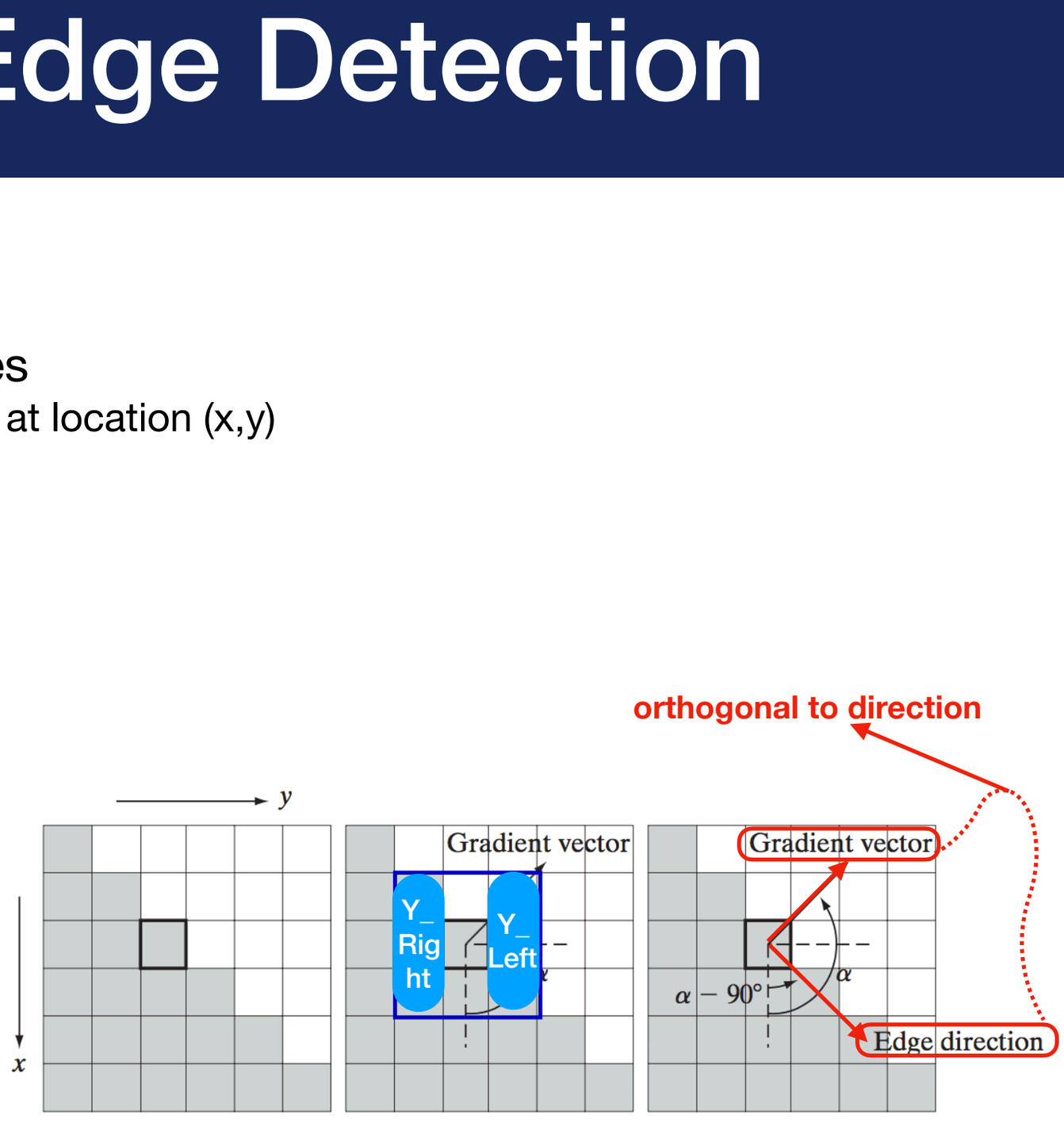
- **Magnitude (length)** of vector Nebular f

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

- **Direction** of the gradient vector

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

orthogonal to direction




**Suppose of pixels in gray have value of 0 and white have value of 1 and Computing derivatives in the x- and y-direction using 3x3 neighborhood centered**

**partial derivative in the Y direction** = Y_Left - Y_Right

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Discuss about First-order Derivatives

  => For finding edge strength and direction at location (x,y) with gradient of f,

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

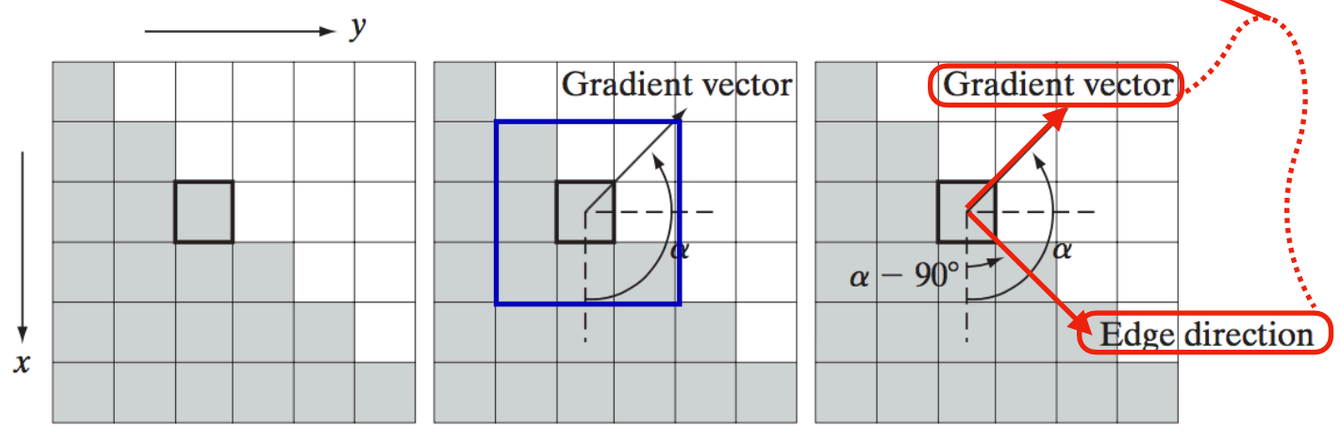
orthogonal to direction

- **Magnitude (length)** of vector Nebular f

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

- **Direction** of the gradient vector

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




$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

# Point, Line, Edge Detection

- Basic Edge Detection

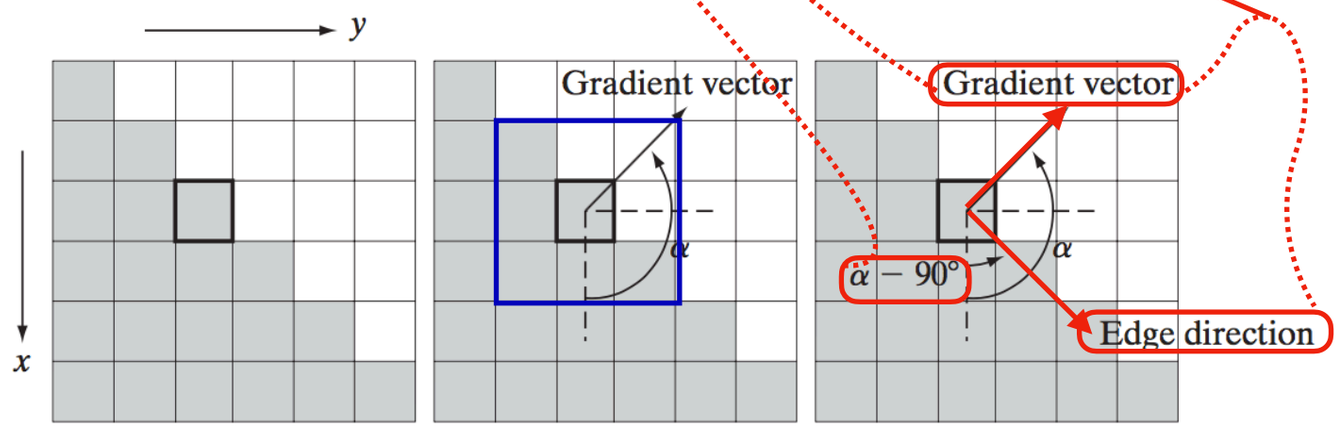
  -> Discuss about First-order Derivatives

  => For finding edge strength and direction at location (x,y) with gradient of f,

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

**Gradient vector is called edge normal**

**orthogonal to direction**




- **Magnitude (length)** of vector Nebular f

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

- **Direction** of the gradient vector

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

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

$$M(x,y) = \sqrt{8} = 2\sqrt{2}$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{g_y}{g_x}\right) = \tan^{-1}(-1) = -45^\circ$$

# Point, Line, Edge Detection

- Basic Edge Detection
  -> Gradient Operators

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

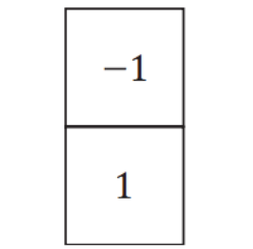
$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,\ y+1) - f(x,y)$$

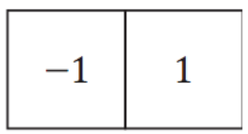
# Point, Line, Edge Detection

- Basic Edge Detection
  -> Gradient Operators

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

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



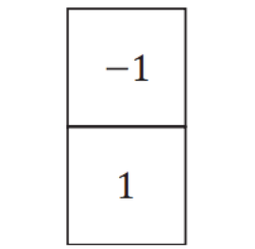

→ **For 1-D Masking**

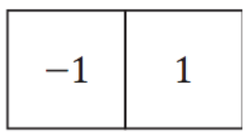
# Point, Line, Edge Detection

- Basic Edge Detection
  -> Gradient Operators

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

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




→ **For 1-D Masking**

-> When diagonal edge direction is of interest,

# Point, Line, Edge Detection

- Basic Edge Detection
  -> Gradient Operators

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

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

| −1 |
|---|
| 1 |

| −1 | 1 |
|---|---|

→ **For 1-D Masking**

-> When diagonal edge direction is of interest,
  => Use 2-D Masking (Roberts operators)

$$g_x = \frac{\partial f(x,y)}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = (z_8 - z_6)$$

| $z_1$ | $z_2$ | $z_3$ |
|---|---|---|
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

→

**Derivatives are implemented by**

| −1 | 0 |
|---|---|
| 0 | 1 |

| 0 | −1 |
|---|---|
| 1 | 0 |

# Point, Line, Edge Detection

- Basic Edge Detection

  -> 2x2 sized Mask are not useful for computing edge direction as masks that are symmetric about the center point computing edge direction as 3x3 sized Mask

| | | | | | |
|---|---|---|---|---|---|
| −1 | −1 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −1 | 0 | 1 |
| 1 | 1 | 1 | −1 | 0 | 1 |

Prewitt

$$g_x = \frac{\partial f(x,y)}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

| | | | | | |
|---|---|---|---|---|---|
| −1 | −2 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −2 | 0 | 2 |
| 1 | 2 | 1 | −1 | 0 | 1 |

Sobel

$$g_x = \frac{\partial f(x,y)}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

# Point, Line, Edge Detection

- Basic Edge Detection

  -> 2x2 sized Mask are not useful for computing edge direction as masks that are symmetric about the center point computing edge direction as 3x3 sized Mask

| | | | | | |
|---|---|---|---|---|---|
| −1 | −1 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −1 | 0 | 1 |
| 1 | 1 | 1 | −1 | 0 | 1 |

Prewitt

$$g_x = \frac{\partial f(x,y)}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

**Simpler to Implement**

| | | | | | |
|---|---|---|---|---|---|
| −1 | −2 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −2 | 0 | 2 |
| 1 | 2 | 1 | −1 | 0 | 1 |

Sobel

$$g_x = \frac{\partial f(x,y)}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

**Better Noise-Suppresion**

Smoothing effect

# Point, Line, Edge Detection

- Basic Edge Detection

  -> 2x2 sized Mask are not useful for computing edge direction as masks that are symmetric about the center point computing edge direction as 3x3 sized Mask

| | | | | | |
|---|---|---|---|---|---|
| −1 | −1 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −1 | 0 | 1 |
| 1 | 1 | 1 | −1 | 0 | 1 |

Prewitt

$$g_x = \frac{\partial f(x,y)}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

**Simpler to Implement**

$$g_y = \frac{\partial f(x,y)}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

| | | | | | |
|---|---|---|---|---|---|
| −1 | −2 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −2 | 0 | 2 |
| 1 | 2 | 1 | −1 | 0 | 1 |

Sobel

$$g_x = \frac{\partial f(x,y)}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

**Better Noise-Suppresion**

Smoothing effect

$$g_y = \frac{\partial f(x,y)}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Magnitude of the gradients from two partial derivatives used to estimate edge strength and direction requires **gx and gy be combined**

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

  -> Combining **requires square and square root computation**, so approach of **approximating** the magnitude of the gradient by absolute value is used

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

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Magnitude of the gradients from two partial derivatives used to estimate edge strength and direction requires **gx and gy be combined**

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

  -> Combining **requires square and square root computation**, so approach of **approximating** the magnitude of the gradient by absolute value is used

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

  => Resulting filters will not be isotropic in general

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Magnitude of the gradients from two partial derivatives used to estimate edge strength and direction requires **gx and gy be combined**

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

  -> Combining **requires square and square root computation**, so approach of **approximating** the magnitude of the gradient by absolute value is used

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

  => Resulting filters will not be isotropic in general

  => Solve this problem by using Prewitt or Sobel masks which **gives isotropic results only for vertical and horizontal edges**

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Magnitude of the gradients from two partial derivatives used to estimate edge strength and direction requires **gx and gy be combined**

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

  -> Combining **requires square and square root computation**, so approach of **approximating** the magnitude of the gradient by absolute value is used

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

  => Resulting filters will not be isotropic in general

  => Solve this problem by using Prewitt or Sobel masks which **gives isotropic results only for vertical and horizontal edges**

  **-> And also we can modify the <Prewitt,Sobel >Mask to take the strongest responses along the diagonal direction**

# Point, Line, Edge Detection

- Basic Edge Detection

-> Magnitude of the gradients from two partial derivatives used to estimate edge strength and d**bined**

$$M(x,y) = mag(\nabla f) =$$

-> Combining **requires** **tion**, so approach of **app** dient by absolute value is used

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

=> Resulting filters will r

=> Solve this proble **ives isotropic results only for vertical and**

**-> And also we take the strongest**

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 1 | −1 | −1 | 0 |
| −1 | 0 | 1 | −1 | 0 | 1 |
| −1 | −1 | 0 | 0 | 1 | 1 |

Prewitt

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | −2 | −1 | 0 |
| −1 | 0 | 1 | −1 | 0 | 1 |
| −2 | −1 | 0 | 0 | 1 | 2 |

Sobel

# Point, Line, Edge Detection

- Basic Edge Detection

  -> Magnitude of the gradients from two partial derivatives used to estimate edge strength and d**bined**

  $$M(x,y) = mag(\nabla f) =$$

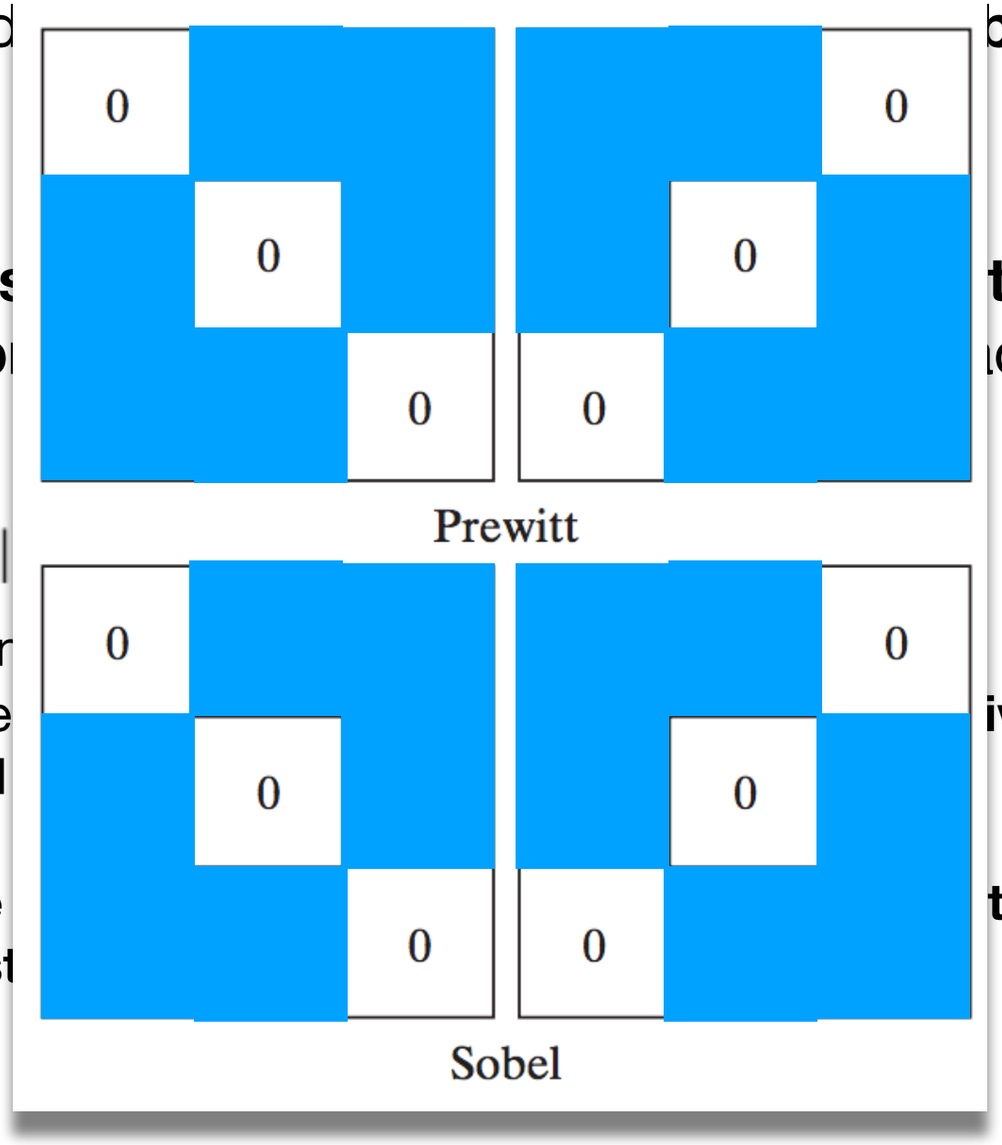
  -> Combining **requires** **tion**, so approach of **app**dient by absolute value is used

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

  => Resulting filters will r

  => Solve this proble**ives isotropic results only for vertical and**

  **-> And also we take the strongest**



# Point, Line, Edge Detection

- Basic Edge Detection

-> Magn … :e
edge …

$M(x,y)$

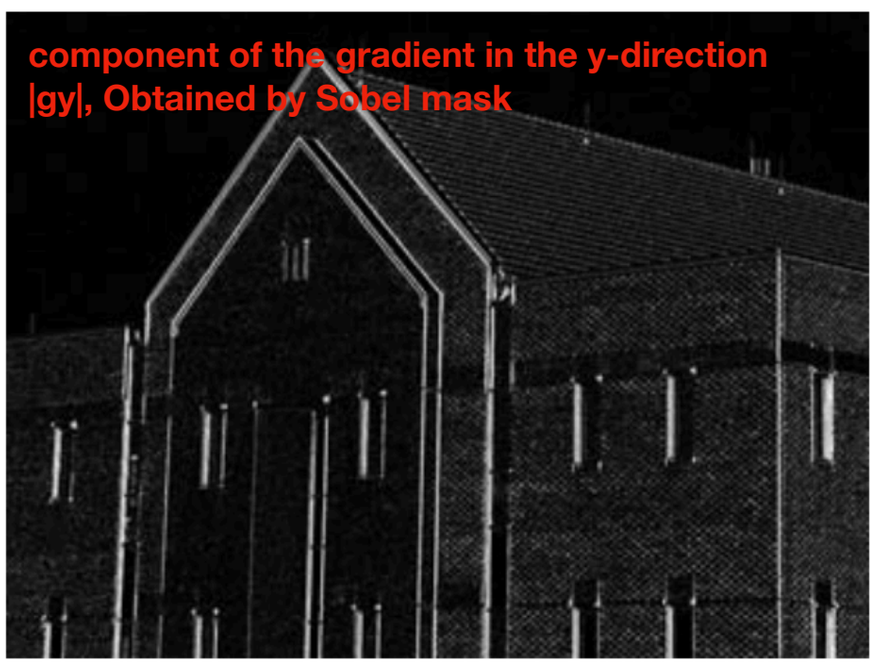
-> Com …
so ap … ute value
is use …

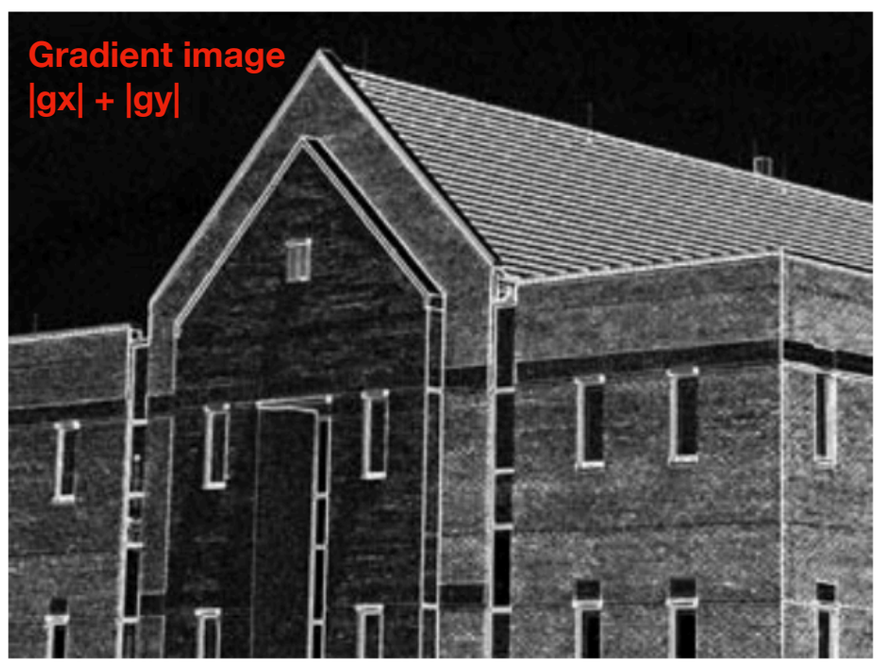
$M(x,y)$

=> Res … sults only
=> … t responses



Gradient Magnitude Image



# Point, Line, Edge Detection

- Basic Edge Detection

-> Magn :e
edge

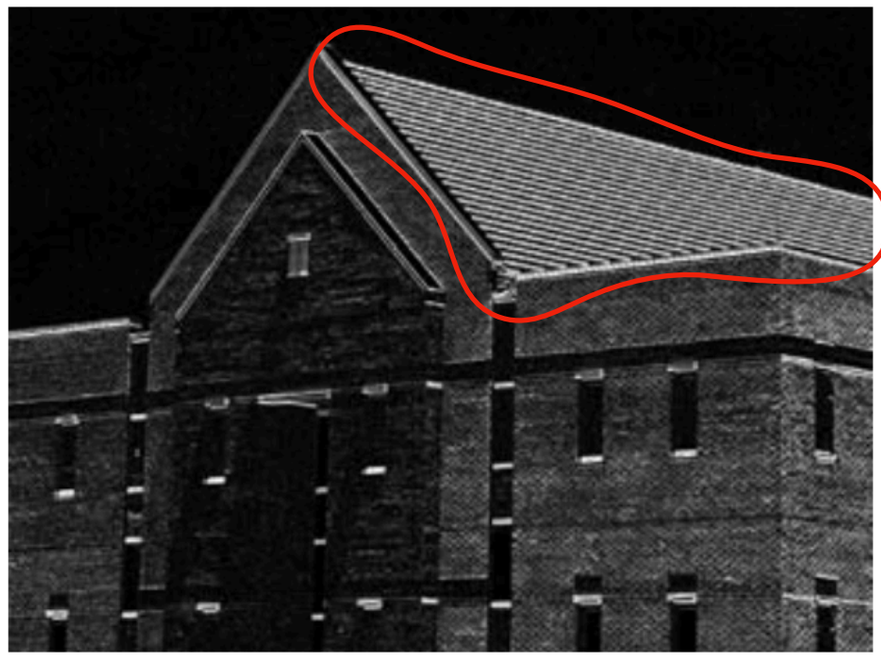
$M(x,y)$

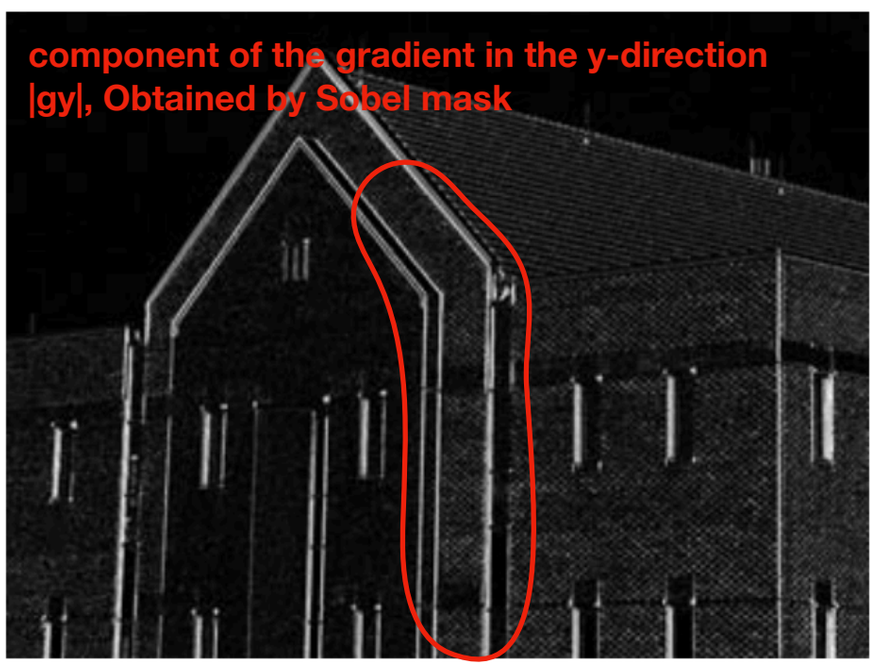
-> Com
so ap ute value
is use

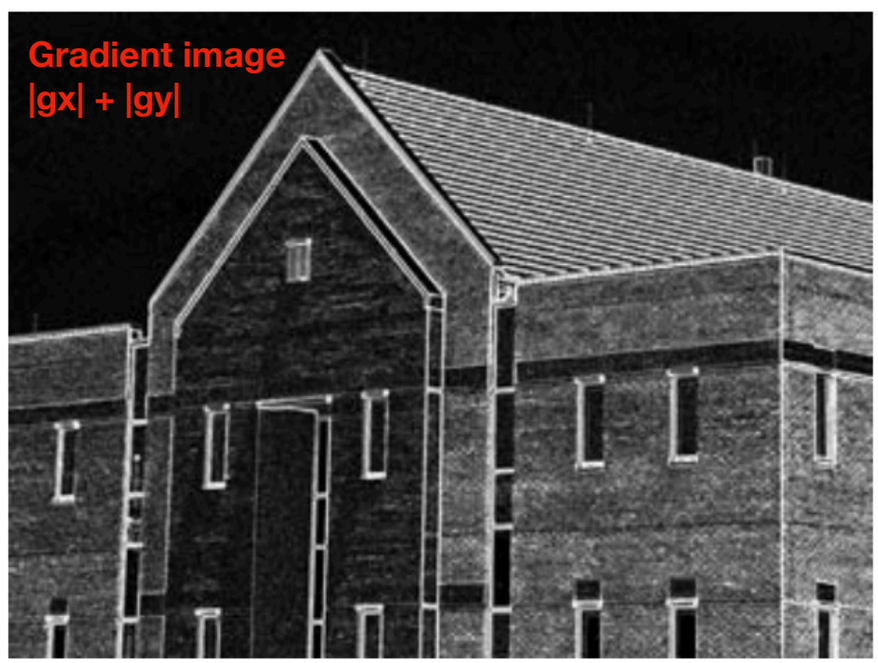
$M(x,y)$

=> Re
=> sults only

t responses

# Point, Line, Edge Detection

- Basic Edge Detection

-> Magnitud … timate edge strer …

$$M(x,y) = n$$

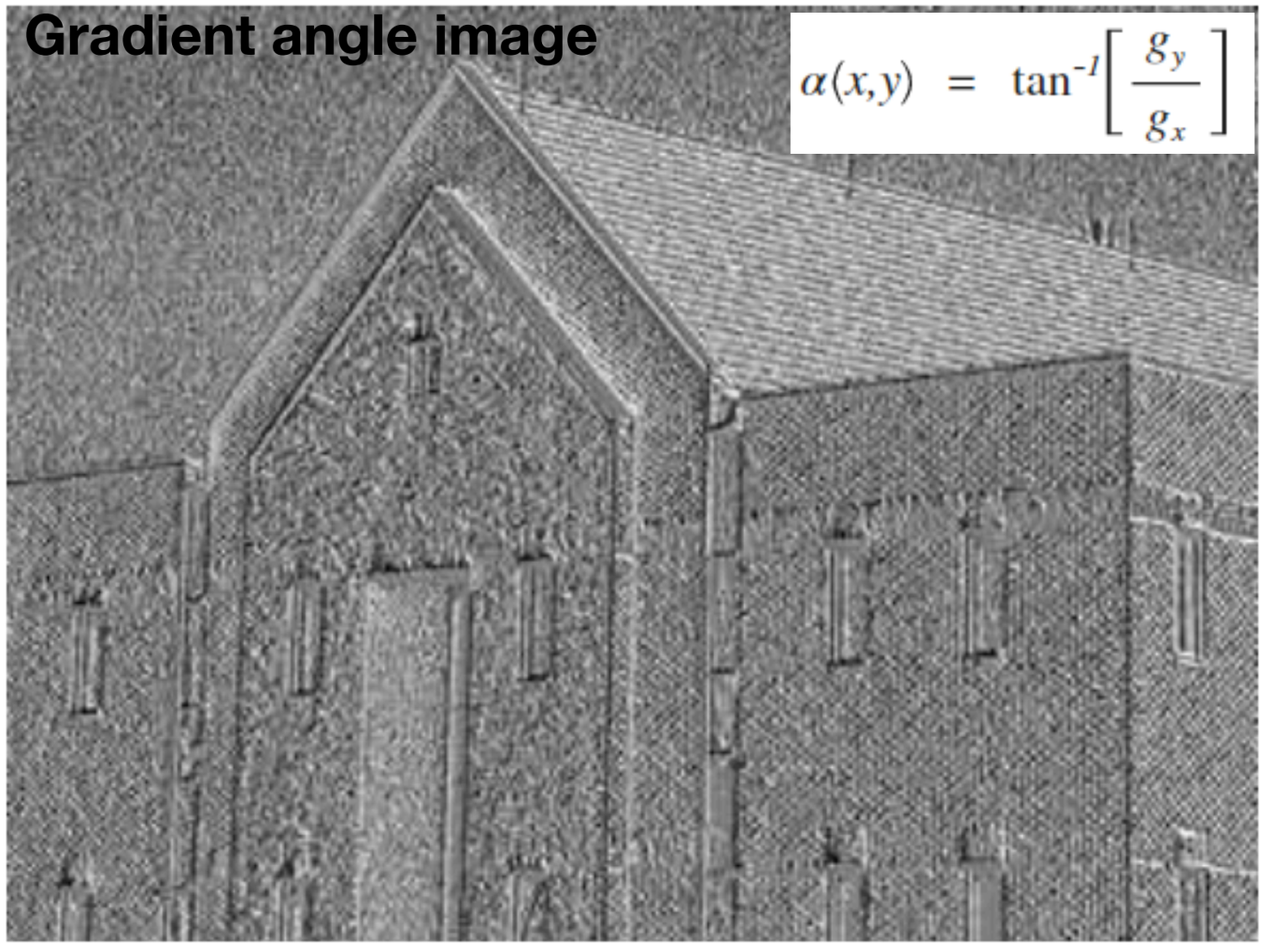
-> Combinin … so approa … bsolute value is used …

$$M(x,y) \approx |$$

=> Resulting …

=> Solve … **ic results only for v …**

**-> S … ngest responses a …**



# Point, Line, Edge Detection

- Basic Edge Detection

  -> Magn **Reduce Detail, Smoothing the image** om two partial derivatives used to estimate

$$= \sqrt{g_x^2}$$

res squ

pproxim

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

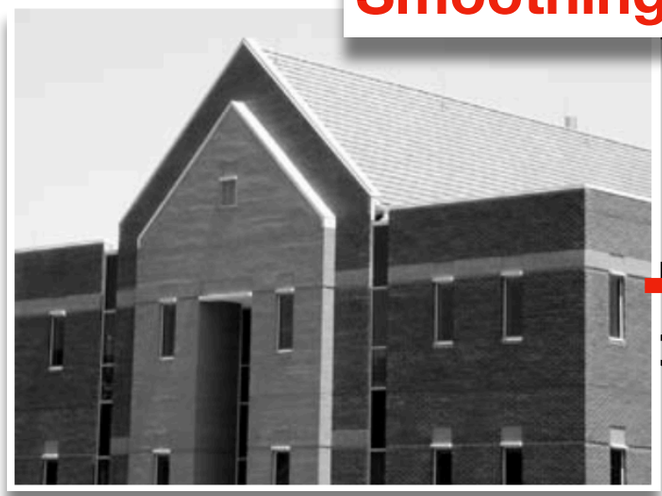
=> Resulting filters will not be i

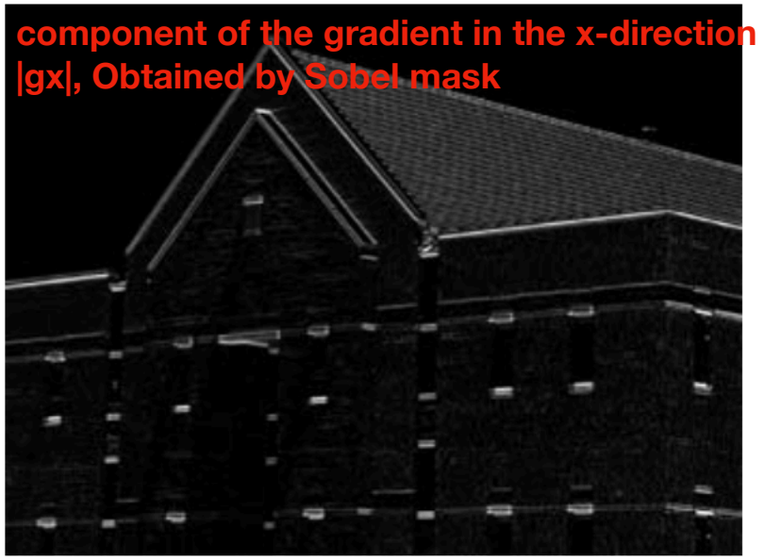
=> Solve this problem by u

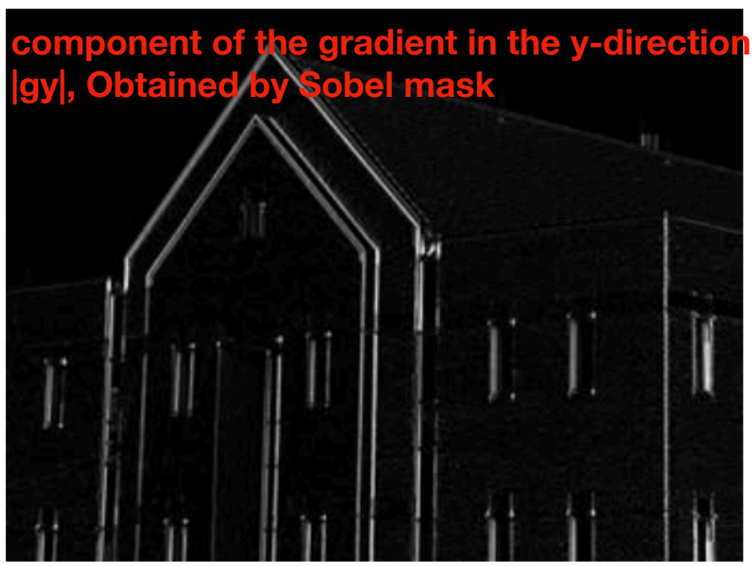
**for vertical and horizo**

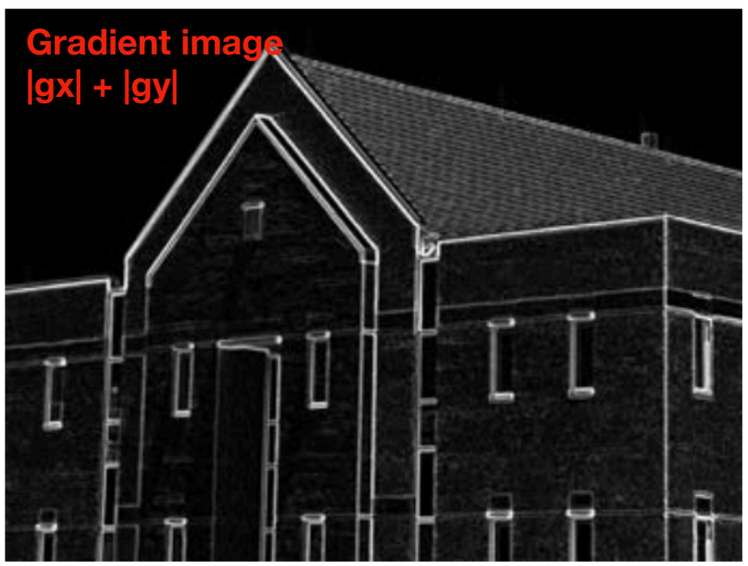
**-> So we modify the <**

**along the diagonal**



# Point, Line, Edge Detection

- Basic Edge Detection

-> Magnitude of the gradients from two partial derivatives used to e

edge stre

$$M(x,y) = r$$

-> Combinir

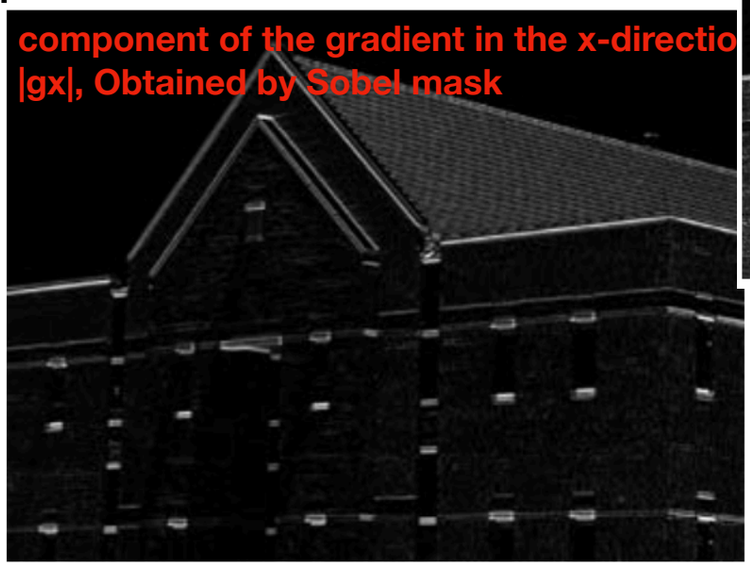
so approa absolute value

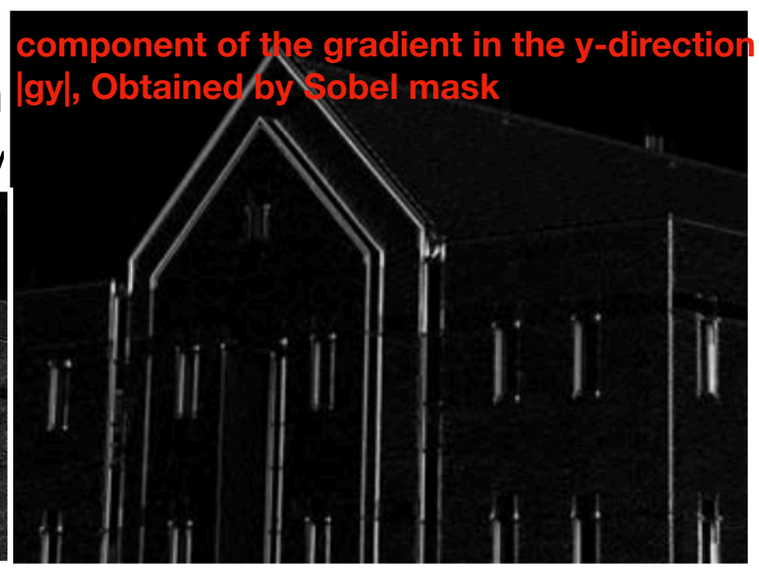
is used

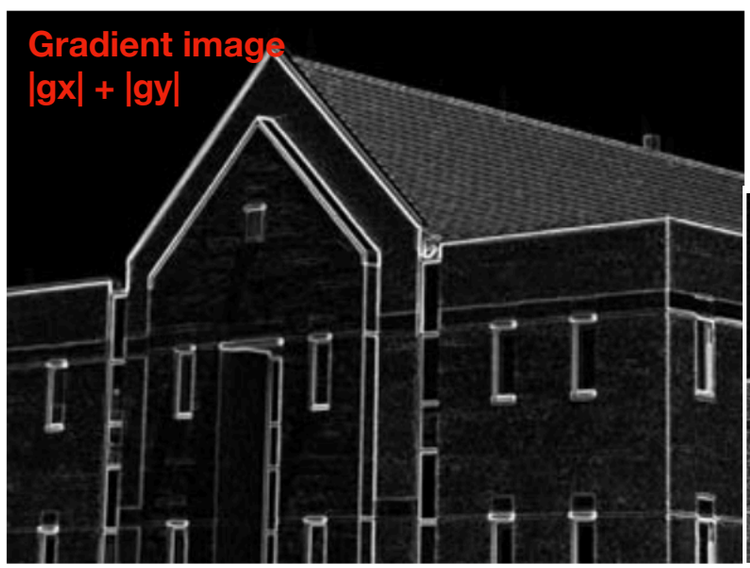
$$M(x,y) \approx$$

=> Resultin

=> Solv pic results only


# Point, Line, Edge Detection

- Basic Edge Dete

-> Magnitude of the g sed to estimate edge strength and **ined**

$M$

| | | |
|---|---|---|
| 0 | 1 | 2 |
| −1 | 0 | 1 |
| −2 | −1 | 0 |

| | | |
|---|---|---|
| −2 | −1 | 0 |
| −1 | 0 | 1 |
| 0 | 1 | 2 |

-> C **i**

s **d** value

is

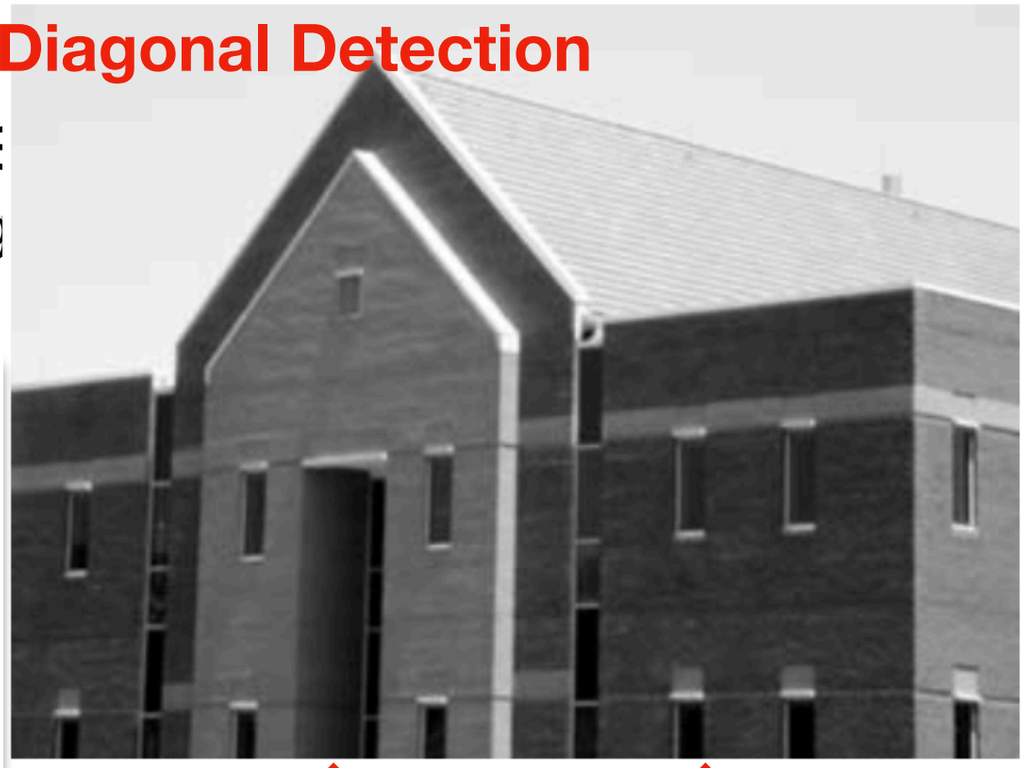
$M(x,$ $\approx$ $|$

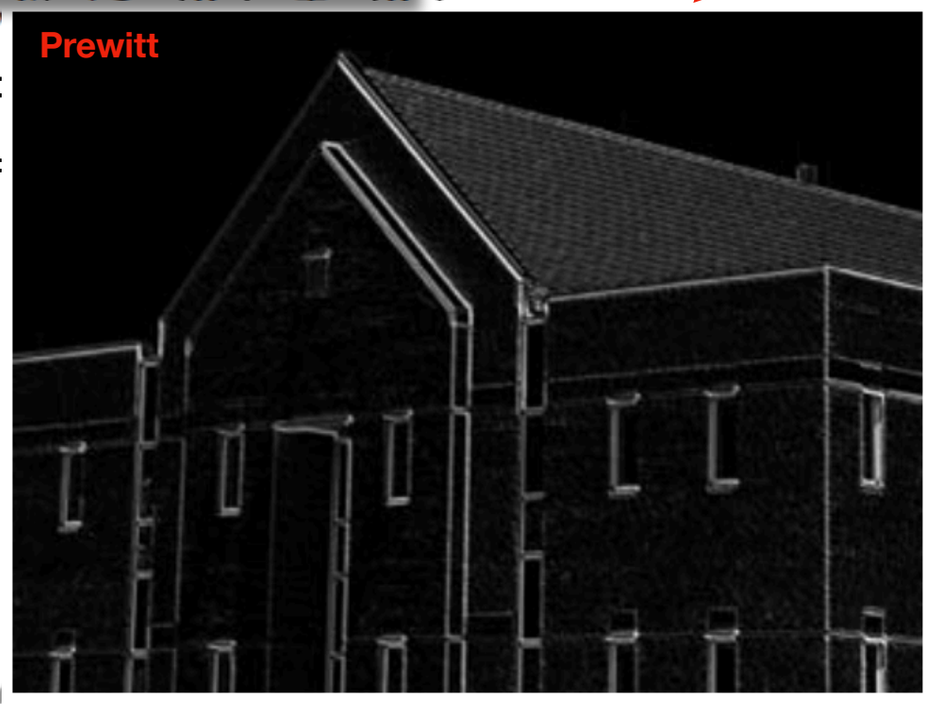
=> F

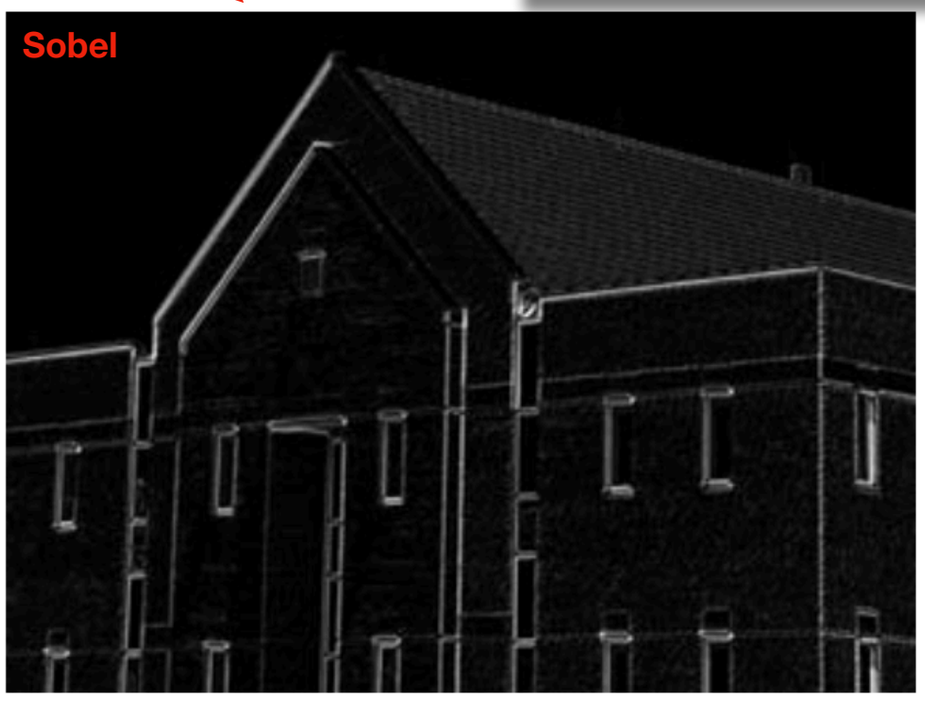
: **ts only**

**esponses**

**Diagonal Detection**





# Point, Line, Edge Detection

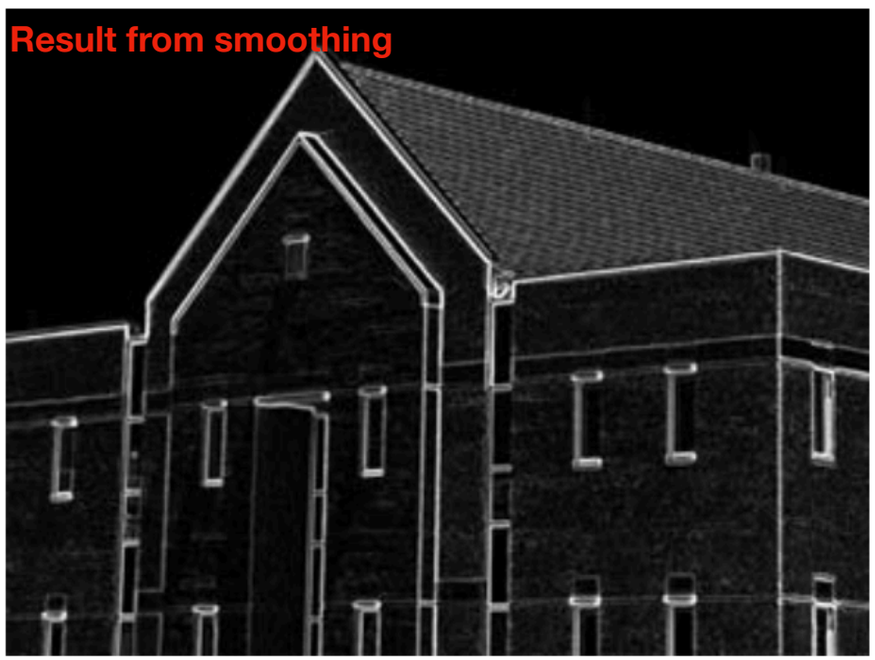
**Thresholding (selected 33% of highest value in image)**

- Basic

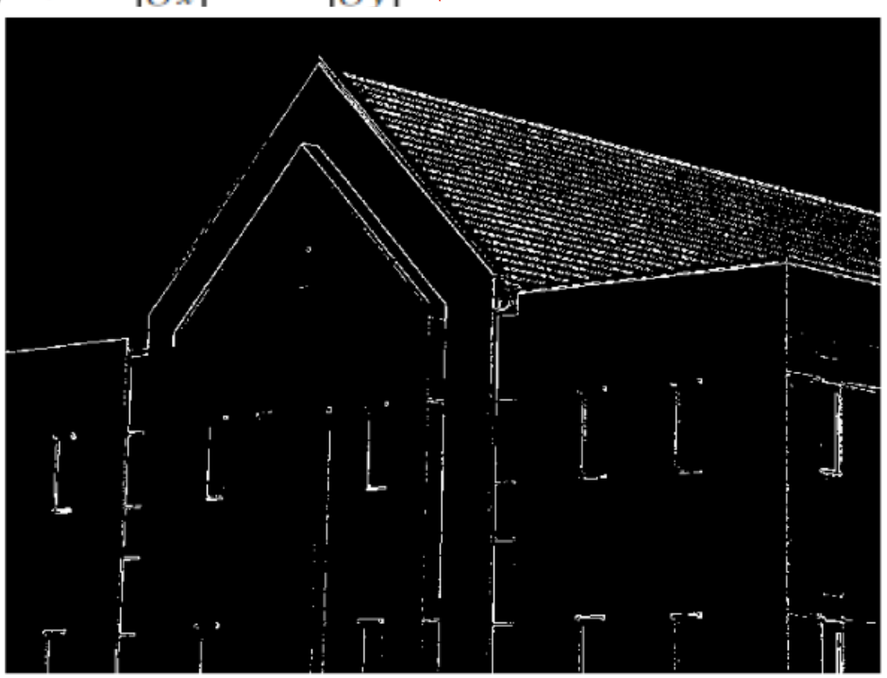
-> Mag p e
edge

$M(x,y$

-> Com a
so ap a te value
is us


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



=> Re

=> ults only

responses

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Improve edge detecting methodologies by taking into account factors such as image noise and the nature of edge themselves

Marr-Hildreth Edge Detector

Canny Edge Detector

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

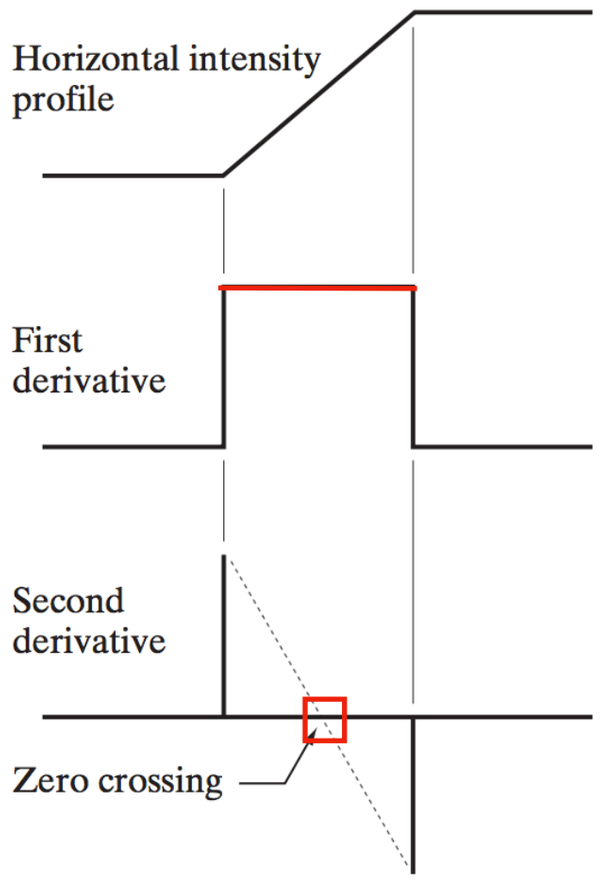
  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**



# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

**=> These two ideas suggest that an operator used for edge detection should have two salient features**

- Should be a differential operator, capable of computing digital approximation of the first & second derivatives at every point in an image

- Should be capable of being tuned to act at any desired scale
  * Large operator -> used for detecting blurring edge
  * Small operator -> used for detecting sharply focused fine detail

**Then, which operator satisfy these conditions?**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

**=> These two ideas suggest that an operator used for edge detection should have two salient features**

- Should be a differential operator, capable of computing digital approximation of the first & second derivatives at every point in an image

- Should be capable of being tuned to act at any desired scale
  * Large operator -> used for detecting blurring edge
  * Small operator -> used for detecting sharply focused fine detail

**Then, which operator satisfy these conditions?**
**!! Laplacian of a Gaussian function (LoG) !!**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

     => Laplacian of Gaussian (LoG)

$$\nabla^2 G$$

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

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

**2-D Gaussian function with std**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

=> Laplacian of Gaussian (LoG)

$$\nabla^2 G$$

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

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

**2-D Gaussian function with std**

$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

$$= \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

$$= \frac{\partial}{\partial x}\left[-\frac{x}{\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[-\frac{y}{\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}\right]$$

$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2 + y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

# Point, Line, Edge Detection

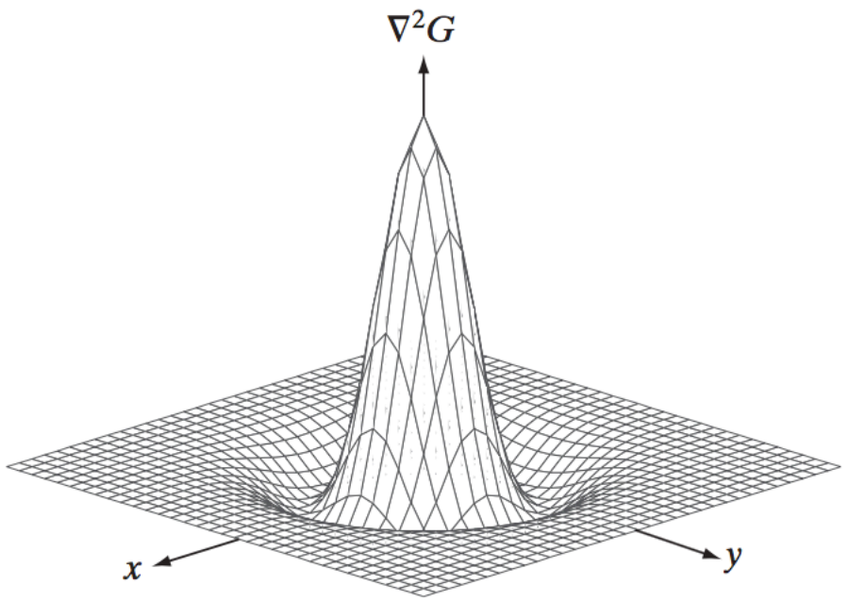
- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

  => Laplacian of Gaussian (LoG)

$$\nabla^2 G(x,y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$$




**Mexican hat operator**

# Point, Line, Edge Detection

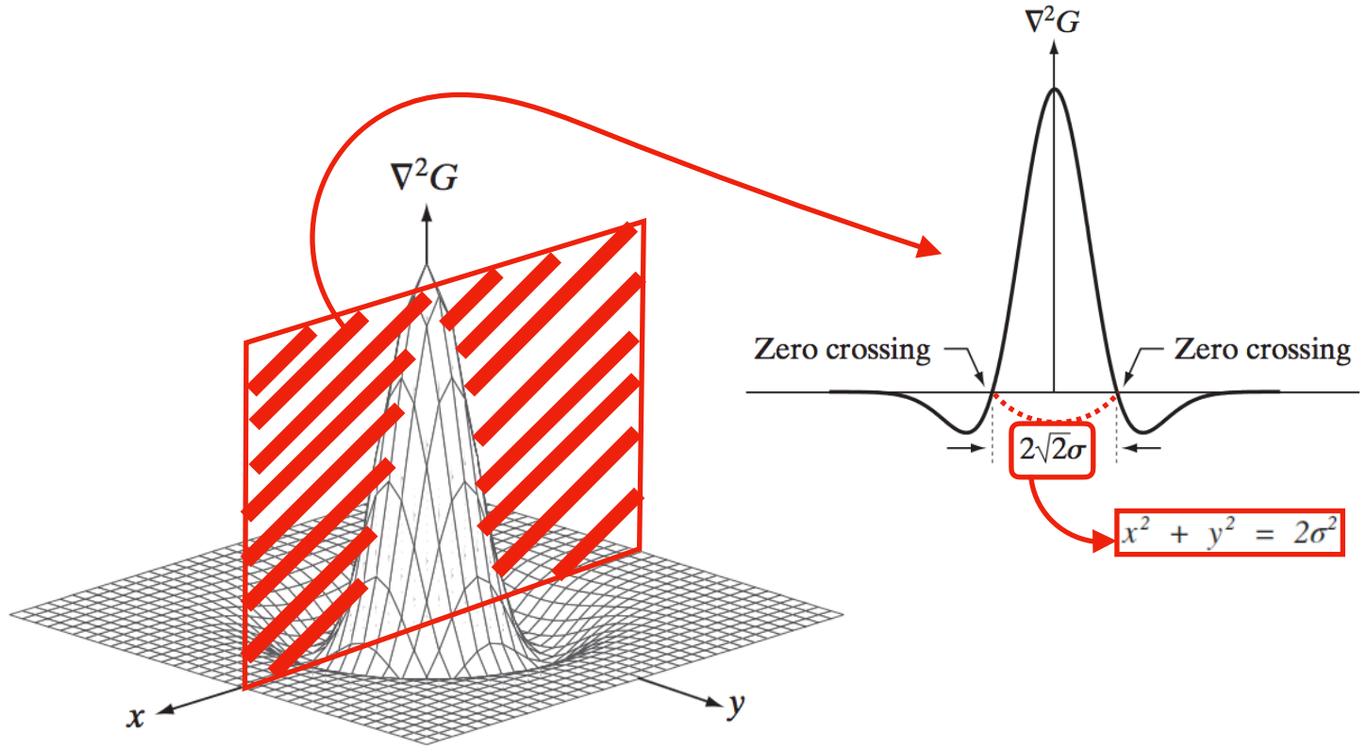
- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

  => Laplacian of Gaussian (LoG)

$$\nabla^2 G(x,y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$$



# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

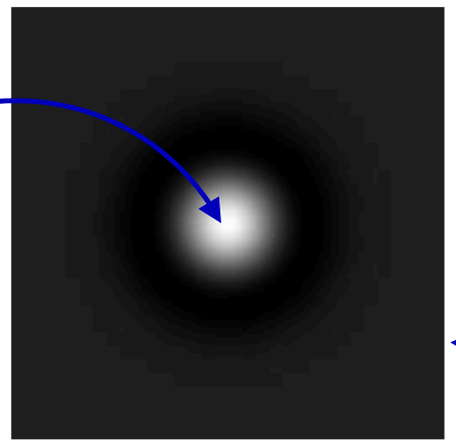
  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

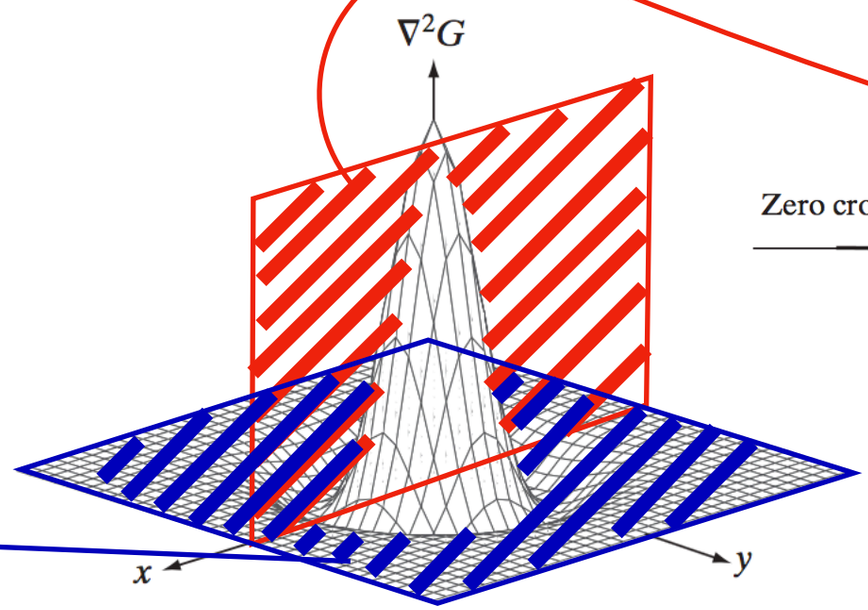
  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

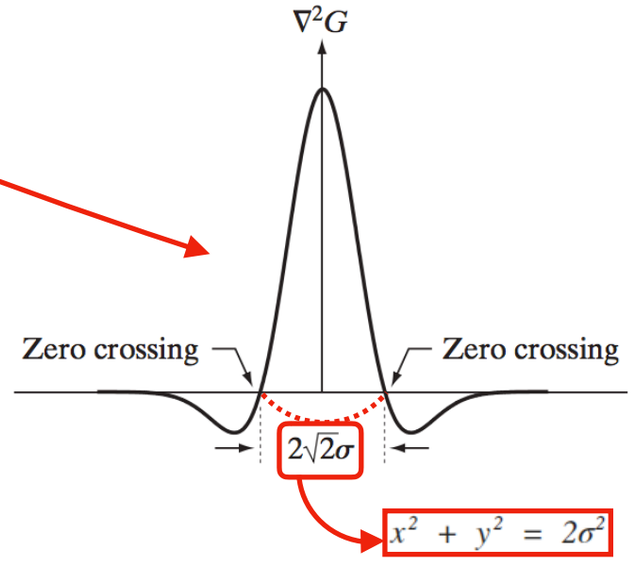
  => Laplacian of Gaussian (LoG)

$$\nabla^2 G(x,y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

| | | | | |
|---|---|---|---|---|
| 0 | 0 | −1 | 0 | 0 |
| 0 | −1 | −2 | −1 | 0 |
| −1 | −2 | 16 | −2 | −1 |
| 0 | −1 | −2 | −1 | 0 |
| 0 | 0 | −1 | 0 | 0 |






**Coefficients are sum into zero for response of the mask is zero in areas of constant intensity**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth Edge Detector

  1) **Intensity changes are not independent of image scale** and so their detection requires the use of operators of different sizes

  2) Sudden intensity change will **give rise to a [peak <or> trough]** in the first derivative or to a **zero-crossing in the second derivative**

  => Laplacian of Gaussian (LoG)

  Two Fundamental ideas behind the selection of the LoG operator

  - **Gaussian part of the operator blurs the image, thus reducing the intensity of structures at scales much smaller than std value**
    *** Smooth in both the spatial and frequency domains, with less likely to introduce artifacts**

  - **Second derivative filter**
    *** Important advantage of being isotropic (responds equally to changes in intensity in any mask direction)**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth edge-detection algorithm

1) Filter the input image with an nxn Gaussian low pass filter obtained by sampling $G(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$

2) Compute the Laplacian of the image resulting from step1 using nxn mask

$$g(x,y) = \nabla^2[G(x,y) * f(x,y)]$$

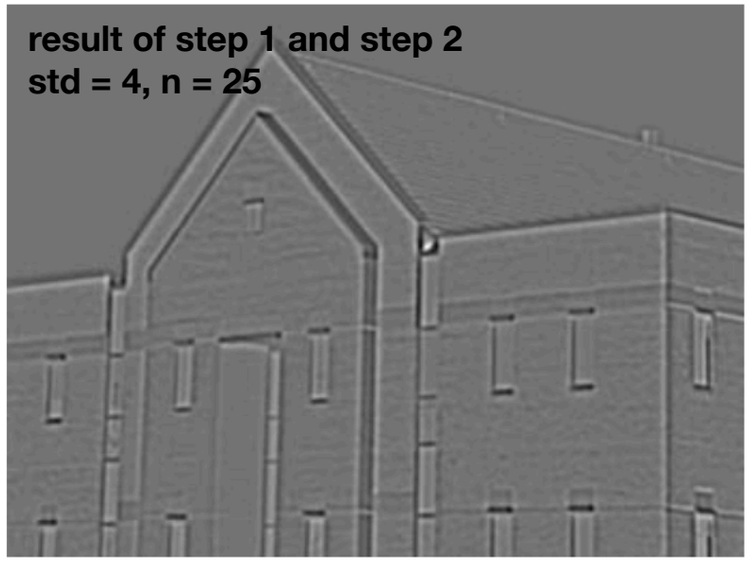
**n should be smallest odd integer greater than or equal to 6*std**

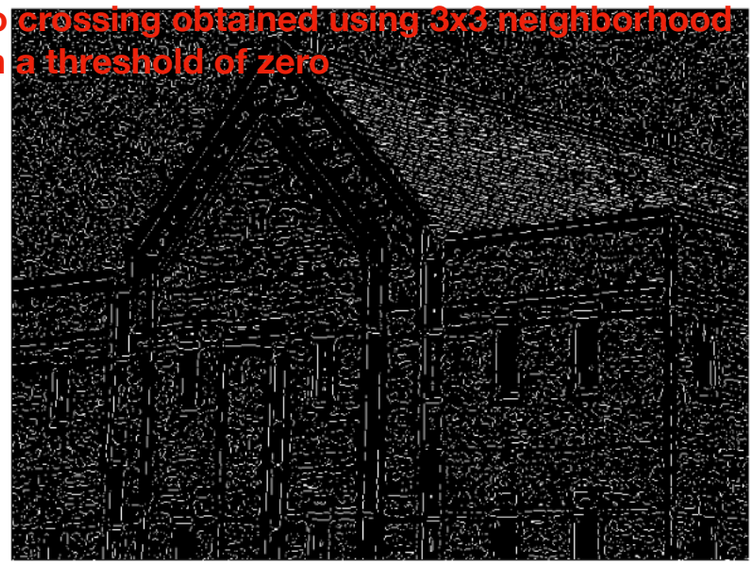
3) Find the zero-crossings of the image from step2

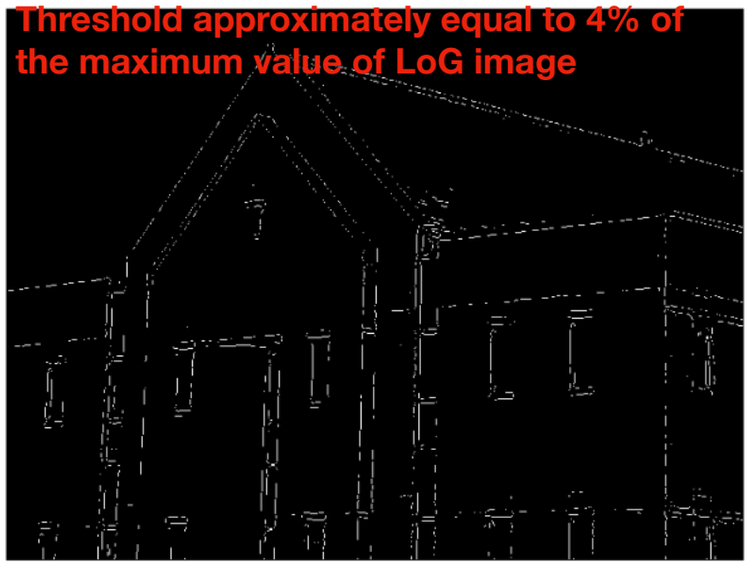
=> For finding the zero crossing at any pixel p, of the filtered image g(x,y),
Use mxm neighborhood centered at p

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth edge-detection algorithm

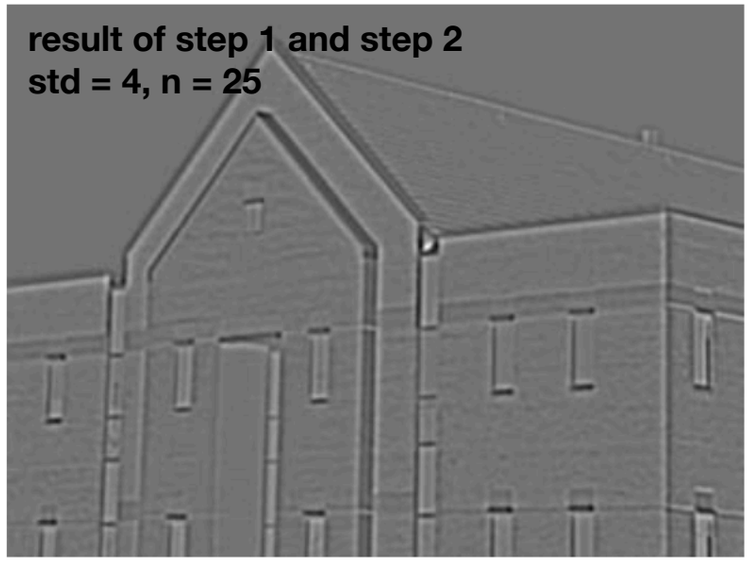



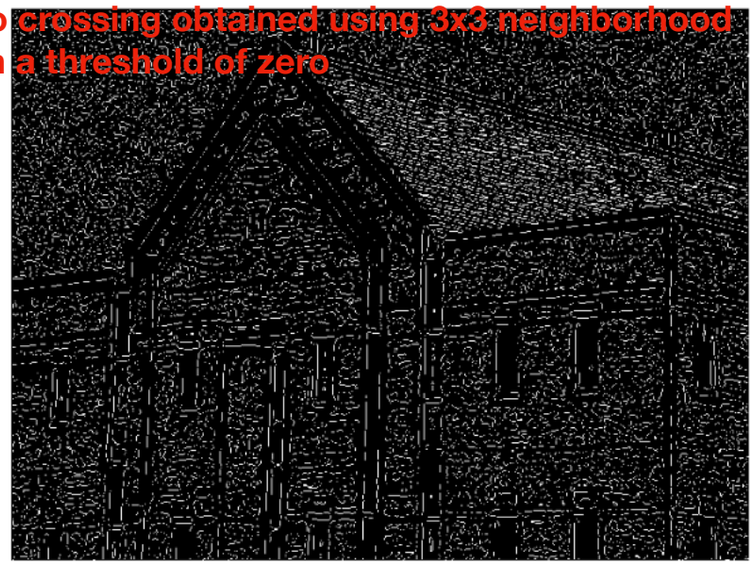



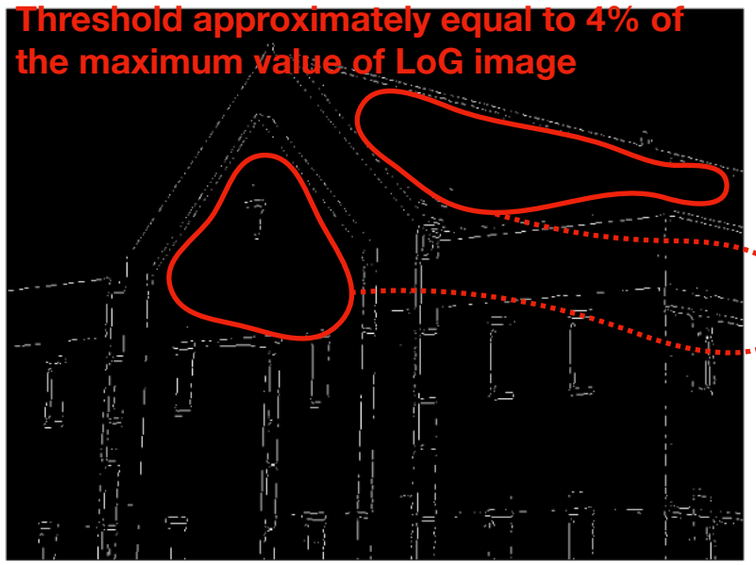



# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Marr-Hildreth edge-detection algorithm


irrelevant features, such as the edges due to the bricks and tile roof were filtered out

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Approximate LoG filter by DoG(Difference of Gaussians)

$$\nabla^2 G(x,y) = \left[ \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

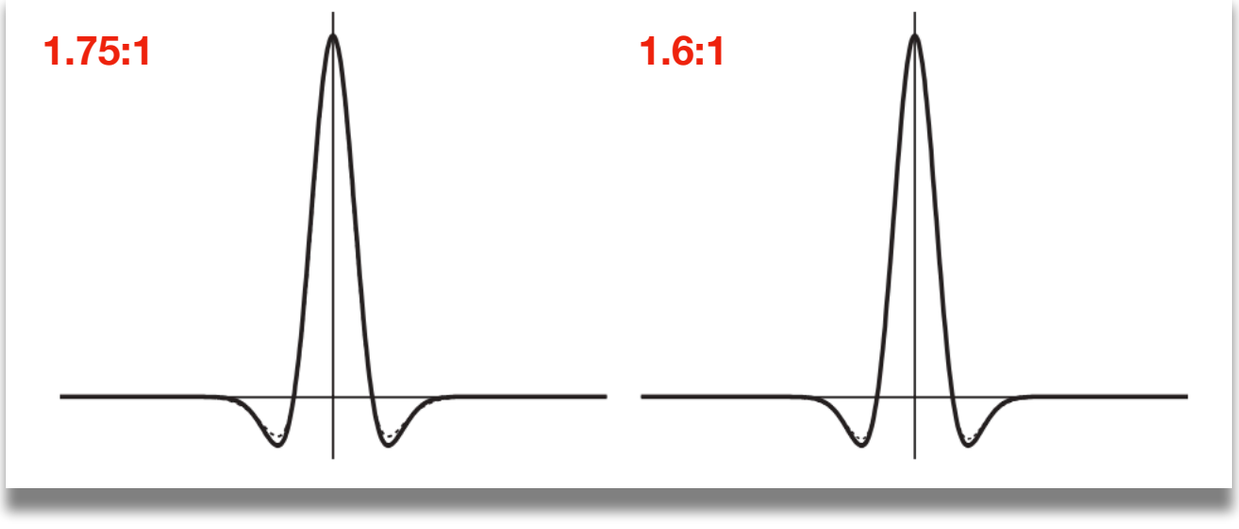
$$DoG(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2 + y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2 + y^2}{2\sigma_2^2}}, \quad (\sigma_1 > \sigma_2)$$

=> From human vision system, we set the ratio of two deviations with 1.75:1
=> Marr and Hildreth suggest using the **ratio of 1.6:1**
  * they argue the ratio of 1.6:1 preserved the basic characteristics of these observations and closer to the approximation to the LoG function

std from LoG

$$\sigma^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[ \frac{\sigma_1^2}{\sigma_2^2} \right]$$

std from DoG



# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Approximate LoG filter by DoG(Difference of Gaussians)

$$\nabla^2 G(x,y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$DoG(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}, \quad (\sigma_1 > \sigma_2)$$

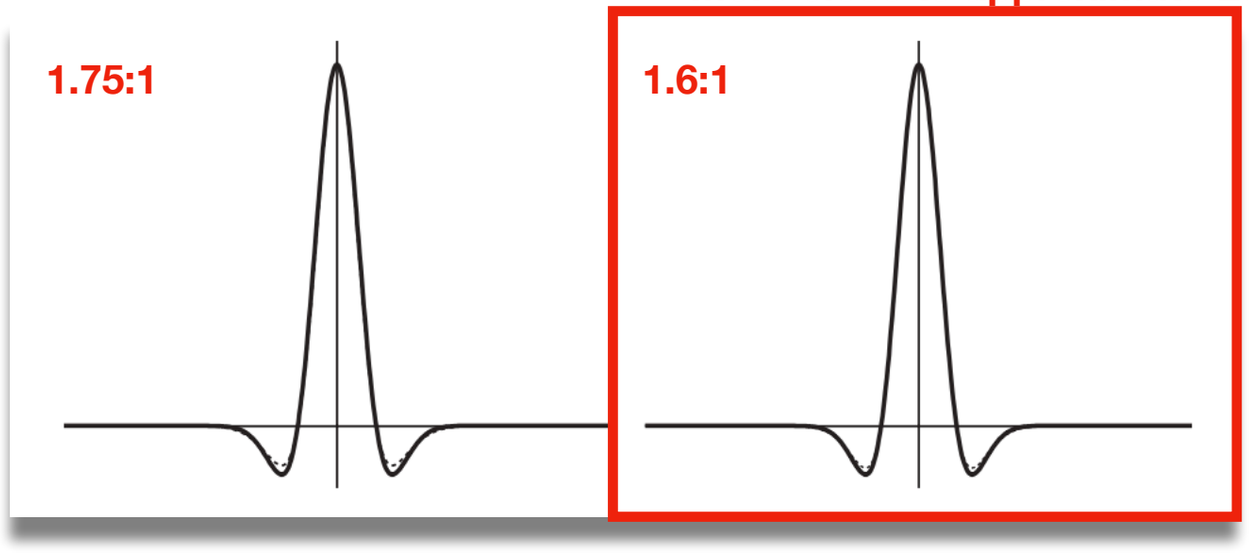
=> From human vision system, we set the ratio of two deviations with 1.75:1
=> Marr and Hildreth suggest using the **ratio of 1.6:1**
  * they argue the ratio of 1.6:1 preserved the basic characteristics of these observations and closer to the approximation to the LoG function

std from LoG

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right]$$

std from DoG

More close to approximate LoG



# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection

  -> Canny Edge Detector

  => More complex algorithm

  => Performance is superior in general to the edge detectors thus far

*** Basic Objectives of Canny edge detection**

- **Low error rate**
  **: Detected edges must be as close as possible to the true edges**

- **Edge points should be well localized**
  **: Located edges must be as close as possible to the true edges**

- **Single edge point response**
  **: Detector should return only one point for each true edge point**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

  1) Smooth the image with a Gaussian filter

  2) Compute the gradient magnitude and angle images

  3) Apply non maxima suppression to the gradient magnitude image

  4) Use double thresholding and connectivity analysis to detect and link edges

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

  1) Smooth the image with a Gaussian filter

Gaussian low pass filter

$$f_s(x,y) = G(x,y) * f(x,y)$$

Smoothed image  Original image

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

  2) Compute the gradient magnitude and angle images

**Gradient Direction**

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

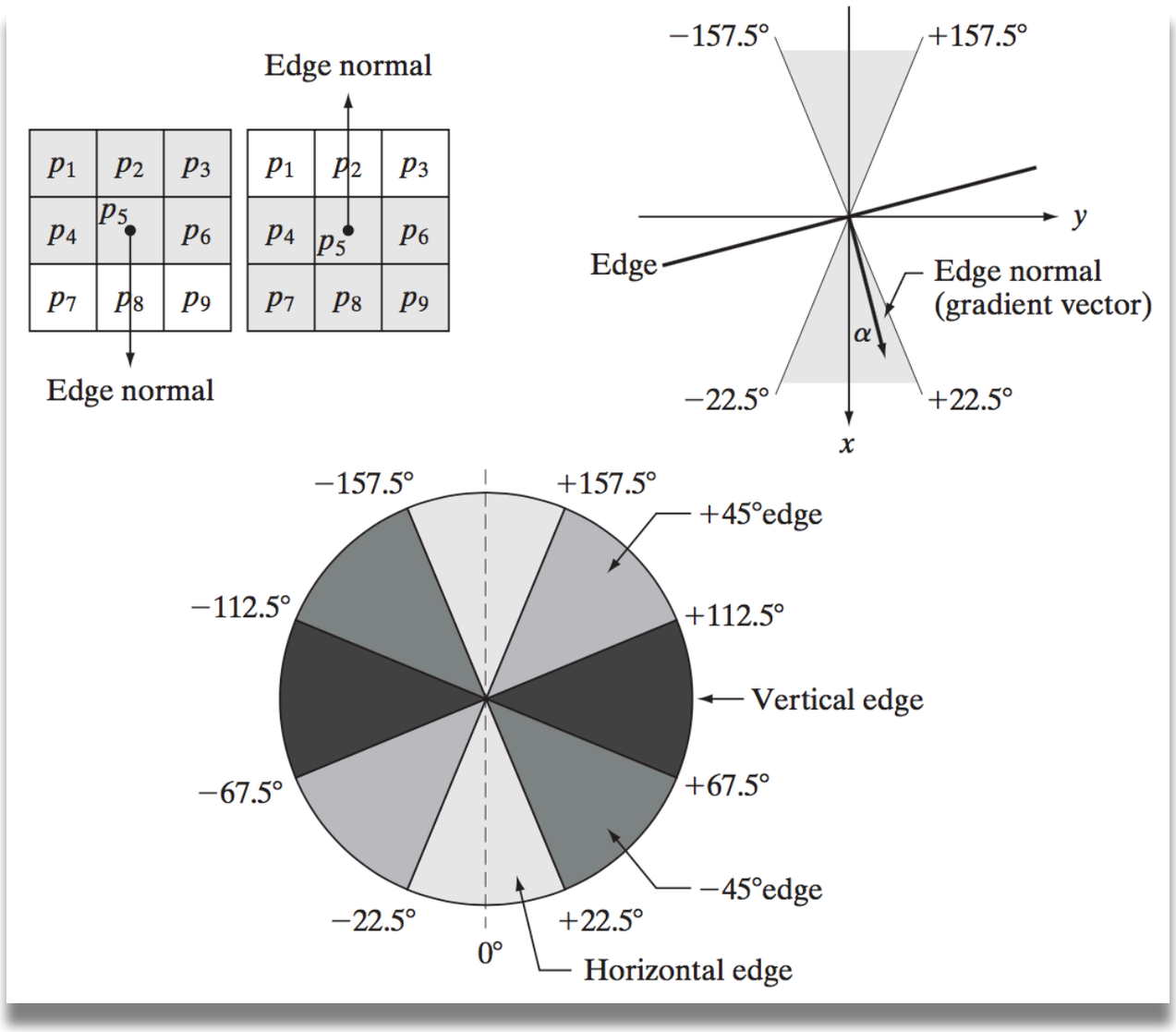
# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

  2) Compute the gradient magnitude and angle images

**Gradient Direction**

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



# Point, Line, Edge Detection

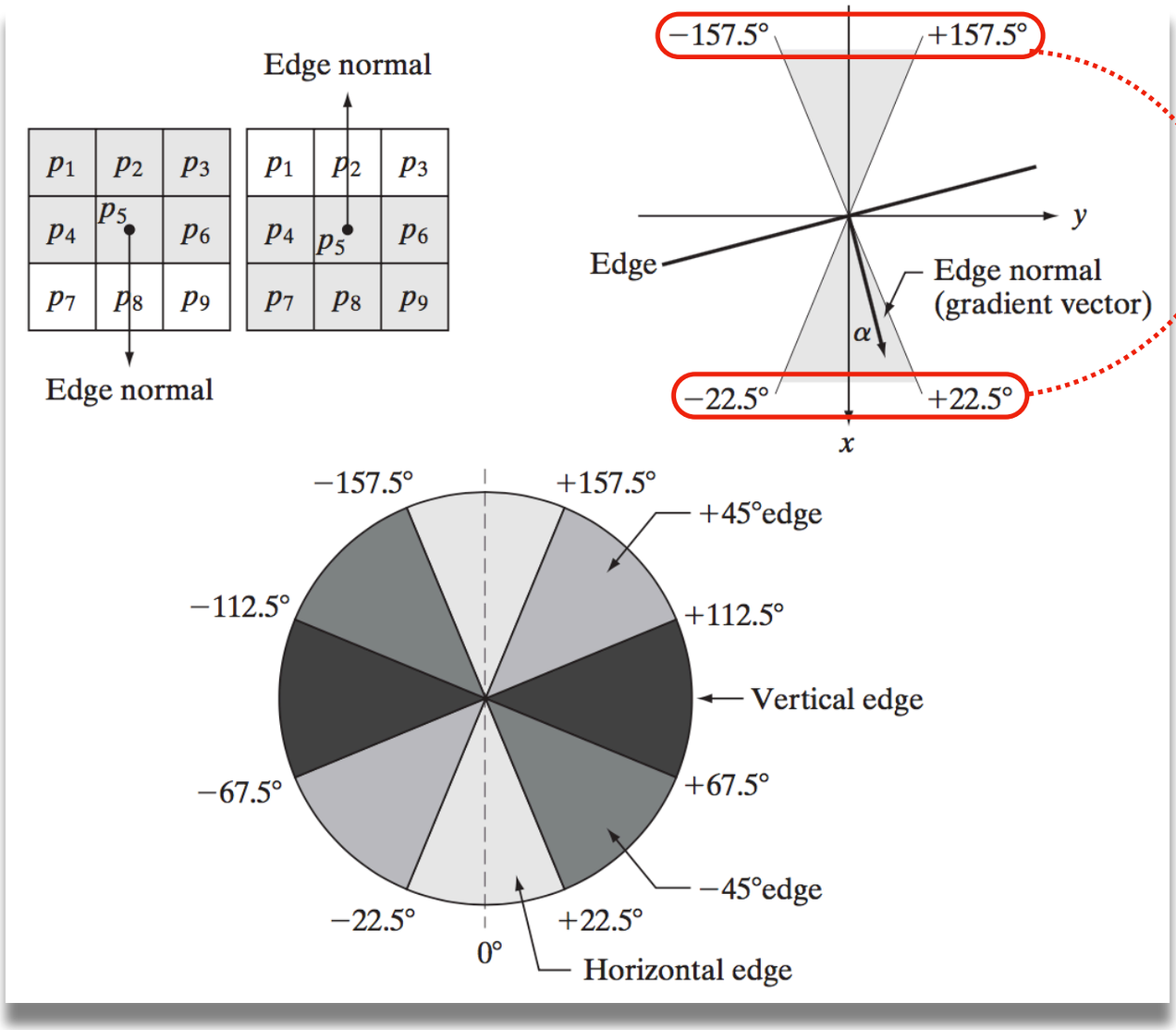
- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

2) Compute the gradient magnitude and angle images

**Gradient Direction**

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

**Find the direction dk that is closest to a(x,y)**



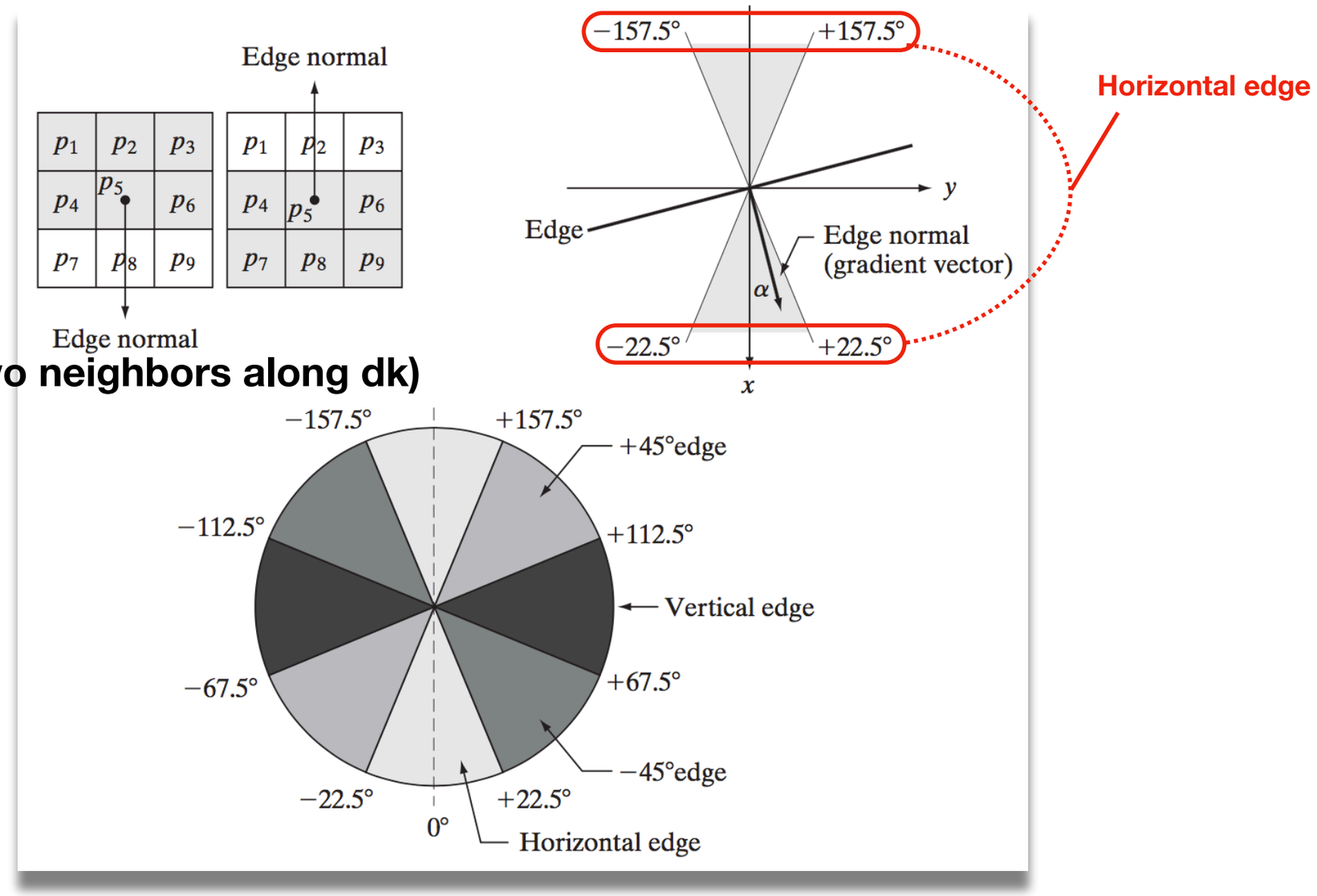
# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

3) Apply non maxima suppression to the gradient magnitude image

**Gradient Magnitude Image**

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




**If (M(x,y) < at least one of its two neighbors along dk)**
**let gn(x,y) = 0 ; (suppression)**
**otherwise**
**let gn(x,y) = M(x,y)**

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

  **To reduce false edge points**

  4) Use double thresholding and connectivity analysis to detect and link edges

  => Hysteresis Thresholding : Using two thresholds
  (Address the problem comes from the size of thresholding value)
  * Too small -> Still be some false edges
  * Too big -> Eliminating actual valid edge points

# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm

**To reduce false edge points**

4) Use double thresholding and connectivity analysis to detect and link edges

=> Hysteresis Thresholding : Using two thresholds
(Address the problem comes from the size of thresholding value)
* Too small -> Still be some false edges
* Too big -> Eliminating actual valid edge points

(1) Create two additional images with two threshold values (TL, TH)

$$g_{NH}(x,y) = g_N(x,y) \geq T_H$$

$$g_{NL}(x,y) = g_N(x,y) \geq T_L$$

(2) Eliminate from gNL(x,y) all the nonzero pixels from gNH(x,y) by

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$$

**Weak edge pixels** **Strong edge pixels**

(3) Take all strong pixels to be valid edge pixels

**a) Locate the next unvisited edge pixel in p in gNH(x,y)**

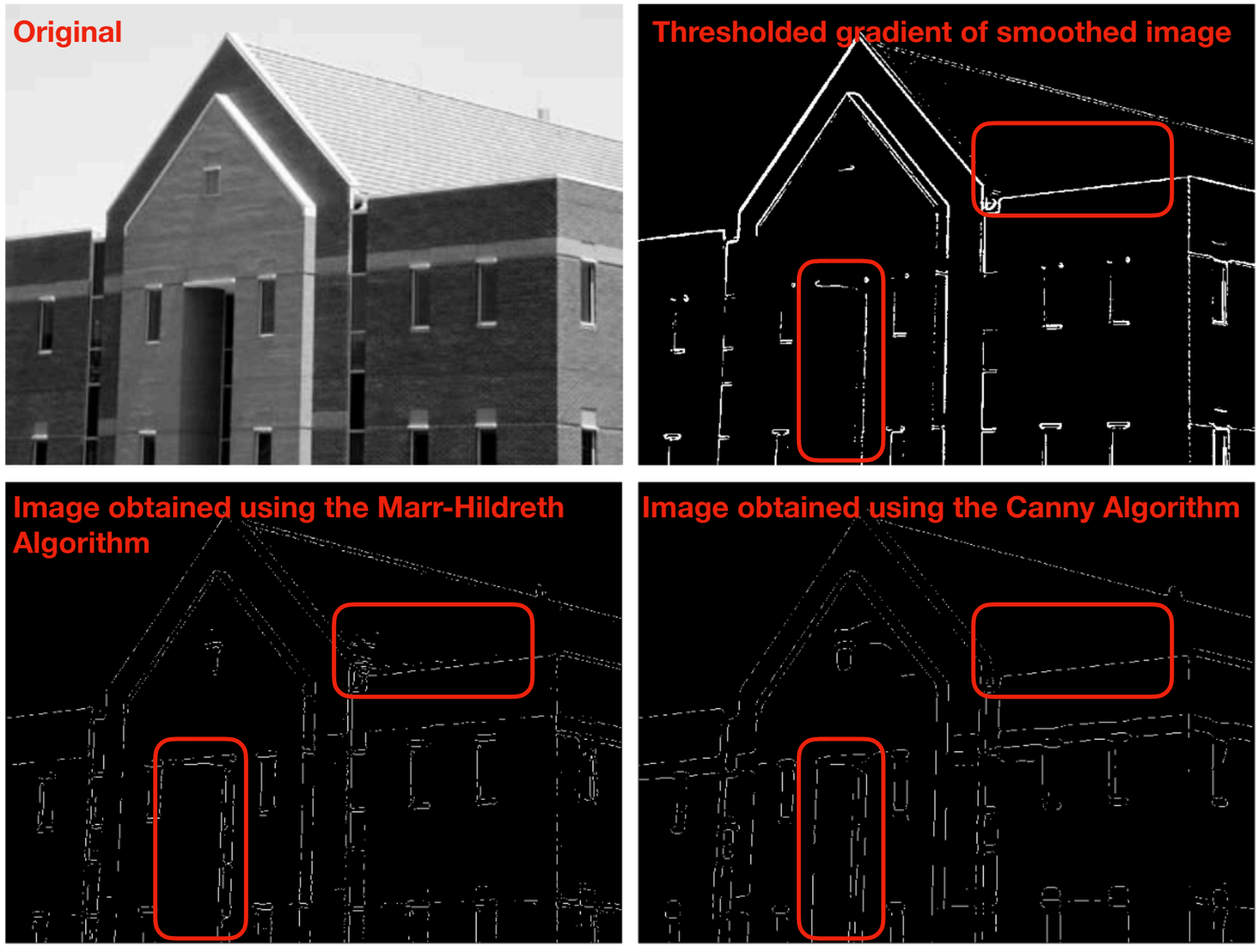
**b) Mark as valid edge pixels all the weak pixels in gNL(x,y) that are connected to p using 8-connectivity**

**c) If nonzero pixels in gNH(x,y) have been visited**
**-> goto step d**
**else return to a**

**d) Set to zero all pixels in gNL(x,y) that are not marked as valid edge pixels**

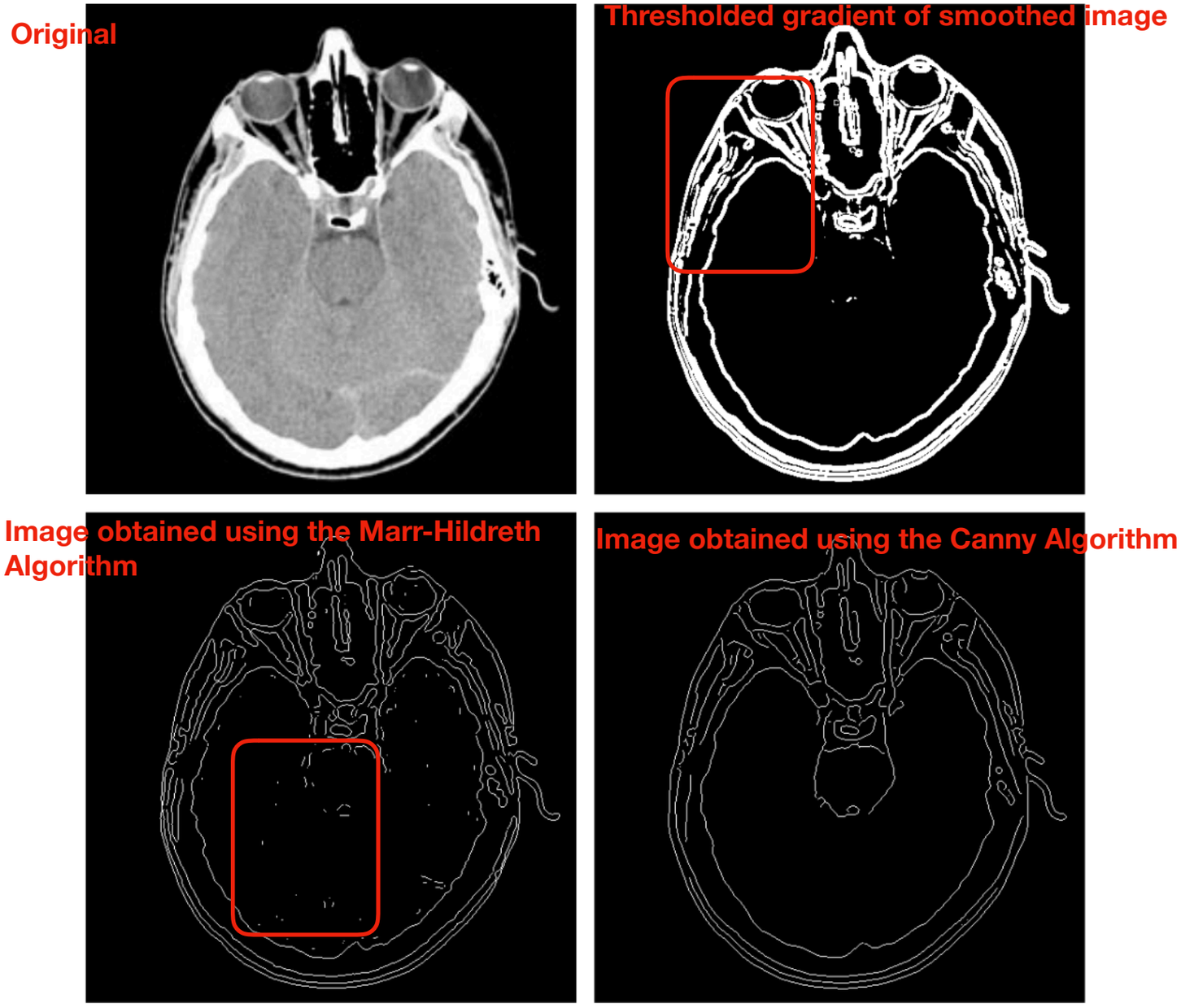
# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm



# Point, Line, Edge Detection

- More Advanced Techniques for Edge Detection
  -> Canny edge detection algorithm



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

**In practice, pixels seldom characterize edges completely**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

**In practice, pixels seldom characterize edges completely**

**Why?**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

**In practice, pixels seldom characterize edges completely**

**Why?**

Noise

Nonuniform illumination

Cause Spurious Discontinuities

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

**In practice, pixels seldom characterize edges completely**

**Why?**

Noise

Nonuniform illumination

Cause Spurious Discontinuities

**=> Address these problems by linking algorithms**

**Assemble edge pixels into meaningful edges and/or region boundaries …**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Local Processing

  => Analyzing the characteristics of pixels in a small neighborhood about every point (x,y) that has been declared an edge point

  => Use two principal properties for establish similarity

  * Strength (Magnitude) of gradient vector

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

  * Direction of gradient vector

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

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Local Processing

  => Analyzing the characteristics of pixels in a small neighborhood about every point (x,y) that has been declared an edge point

  => Use two principal properties for establish similarity

  * Strength (Magnitude) of gradient vector

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \longrightarrow \boxed{|M(s,t) - M(x,y)| \leq E}$$

  * Direction of gradient vector

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \longrightarrow \boxed{|\alpha(s,t) - \alpha(x,y)| \leq A}$$

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Local Processing

  => Analyzing the characteristics of pixels in a small neighborhood about every point (x,y) that has been declared an edge point

  => Use two principal properties for establish similarity

  * Strength (Magnitude) of gradient vector

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

$$|M(s,t) - M(x,y)| \leq E$$

Positive threshold

  * Direction of gradient vector

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

$$|\alpha(s,t) - \alpha(x,y)| \leq A$$

Positive angle threshold

**Pixel with coord (s,t) in Sx,y is lined to the pixel at (x,y) if both condition are satisfied**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Local Processing

  => Analyzing the characteristics of pixels in a small neighborhood about every point (x,y) that has been declared an edge point

=> Use two principal properties for establish similarity

* Strength (Magnitude) of gradient vector

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

$$|M(s,t) - M(x,y)| \leq E$$

Positive threshold

* Direction of gradient v

**Computationally Expensive!**

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

$$|\alpha(s,t) - \alpha(x,y)| \leq A$$

Positive angle threshold

**Pixel with coord (s,t) in Sx,y is lined to the pixel at (x,y) if both condition are satisfied**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Local Processing

  1) Compute the gradient magnitude (M(x,y)) and angle (a(x,y)) of the input image (x,y)

  2) Form a binary image, whose value at any pair of coordinates (x,y) is given by

Band of acceptable direction about a

$$g(x,y) = \begin{cases} 1 & if\ M(x,y) > T_M\ AND\ \alpha(x,y) = A \pm T_A \\ 0 & otherwise \end{cases}$$

  3) Scan the rows of g and fill(set to 1) all gaps(set to 0) in each row that
  do not exceed a specific length K

  4) To detect gaps in any other direction theta, rotate g by this angle and
  apply the horizontal scanning procedure in step 3, and then rotate the result back by -theta

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Local Processing

  1) Compute the gradient magnitude (M(x,y)) and angle (a(x,y)) of the input image (x,y)

  2) Form a binary image, whose value at any pair of coordinates (x,y) is given by

**Band of acceptable direction about a**

$$g(x,y) = \begin{cases} 1 & if\ M(x,y) > T_M\ AND\ \alpha(x,y) = A \pm T_A \\ 0 & otherwise \end{cases}$$

  3) Scan the rows of g and fill(set to 1) all gaps(set to 0) in each row that do not exceed a specific length K

  4) To detect gaps in any other direction theta, **rotate g by this angle** and apply the horizontal scanning procedure in step 3, and then **rotate the result back by -theta**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

-> Loc

1) Com

2) Form

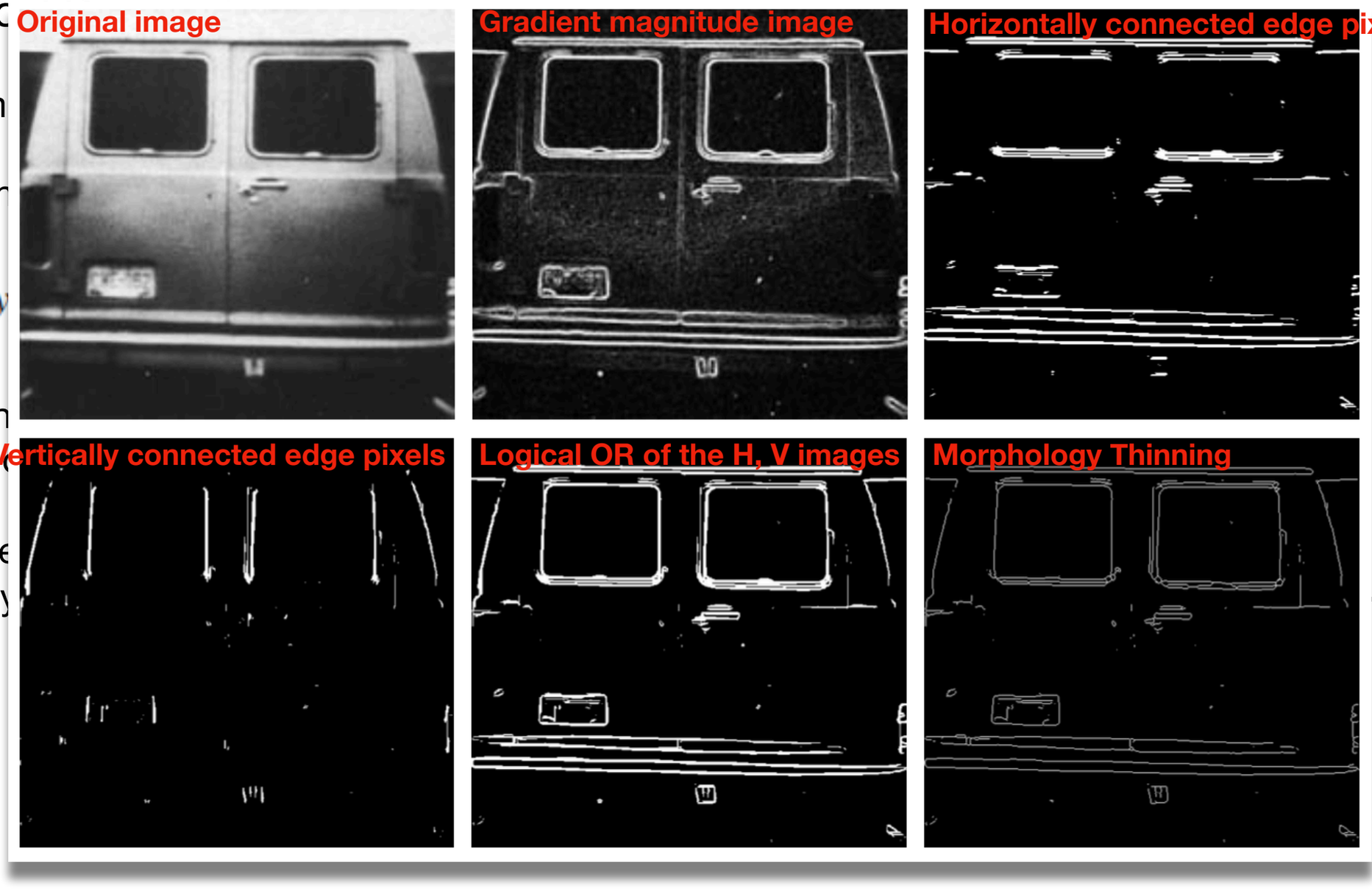
$g(x,y$

3) Scan
do n

4) To de
apply

bout a

theta



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing

  In the case of location of regions of interest in an image are known,

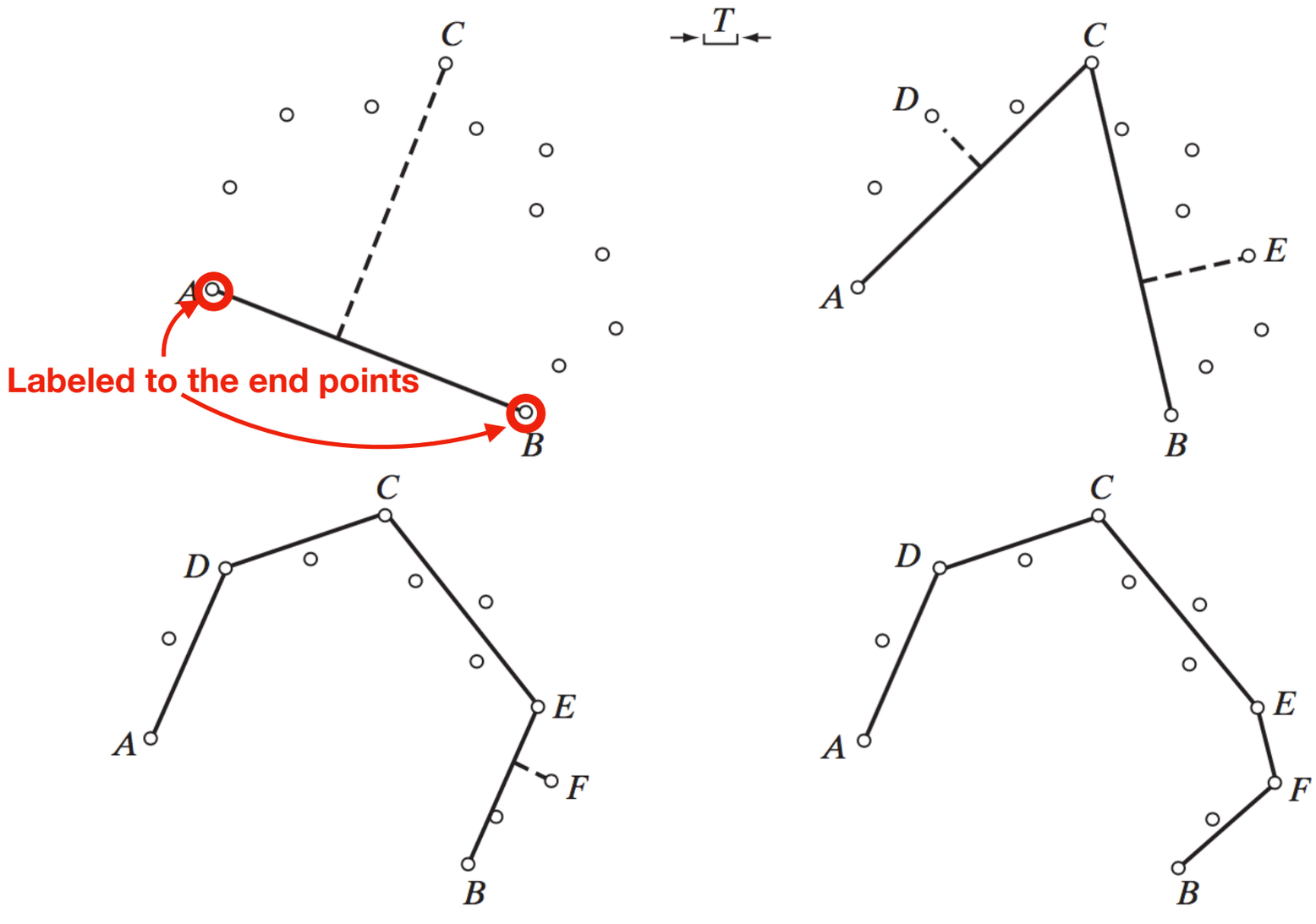
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing

  In the case of location of regions of interest in an image are known,
  => Use techniques which linking pixels on a regional basis, with the desired result being approximation to the boundary of the region

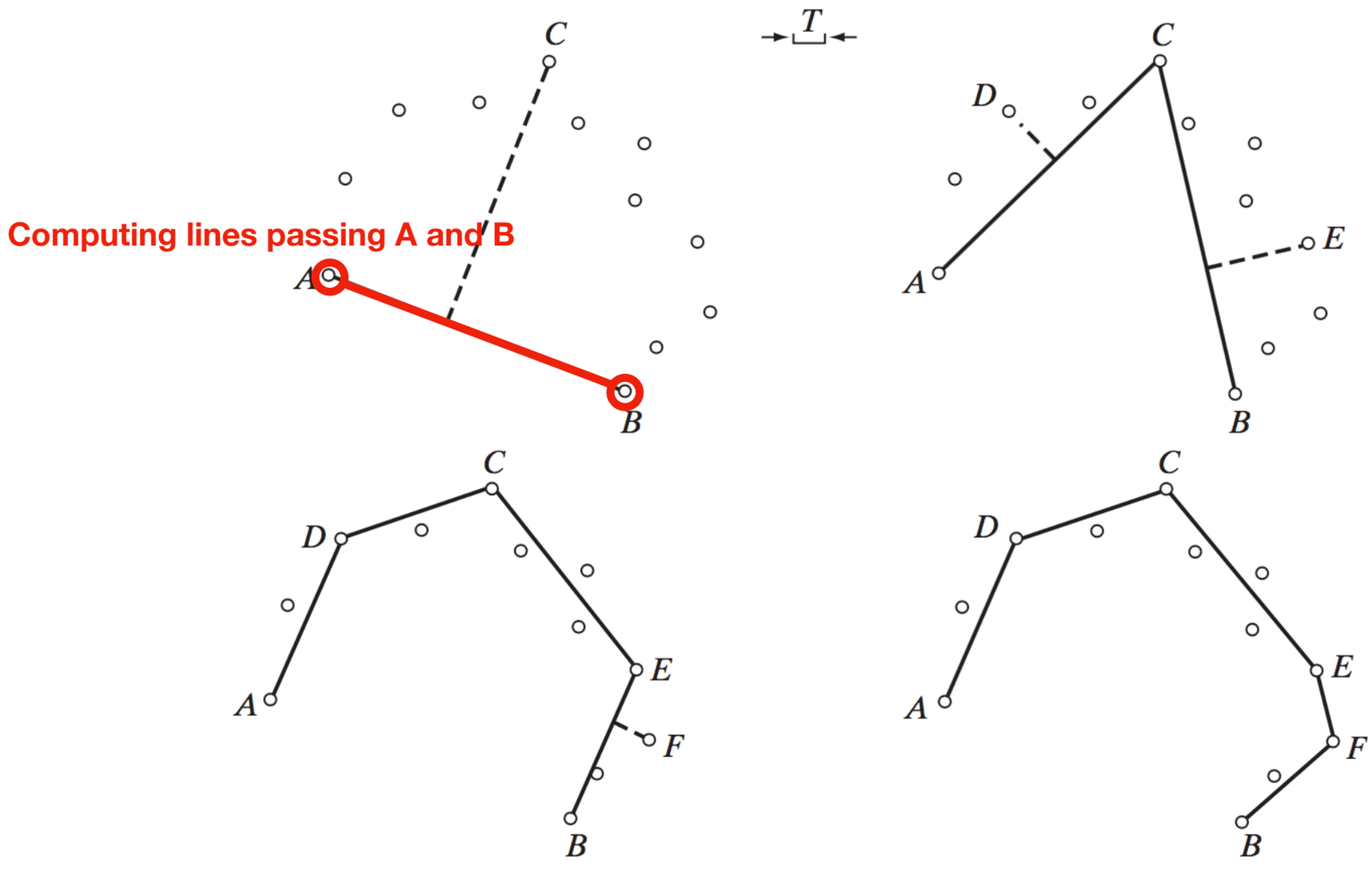
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)



# Point, Line, Edge Detection

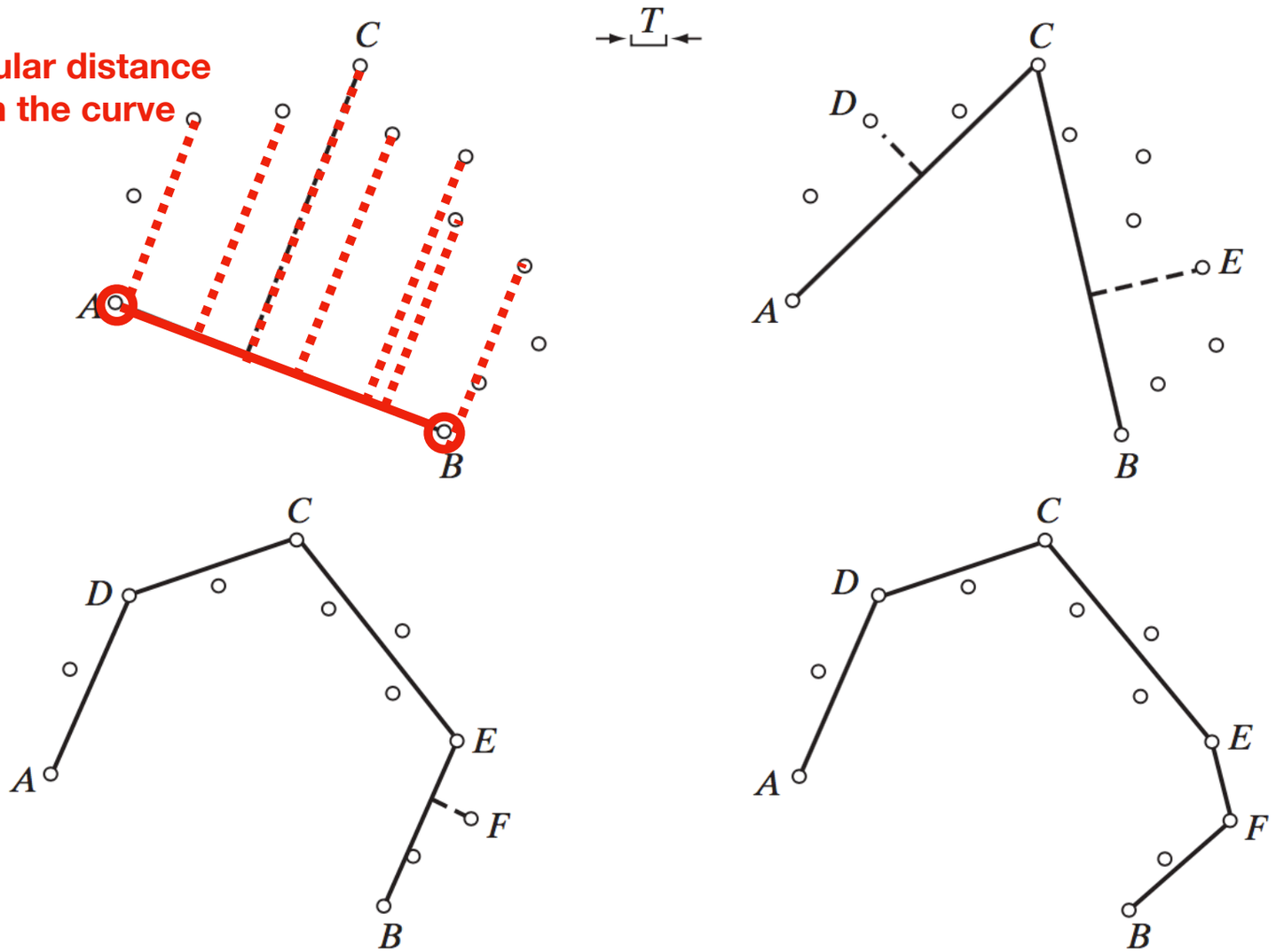
- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)

**Compute the perpendicular distance from all other points in the curve**

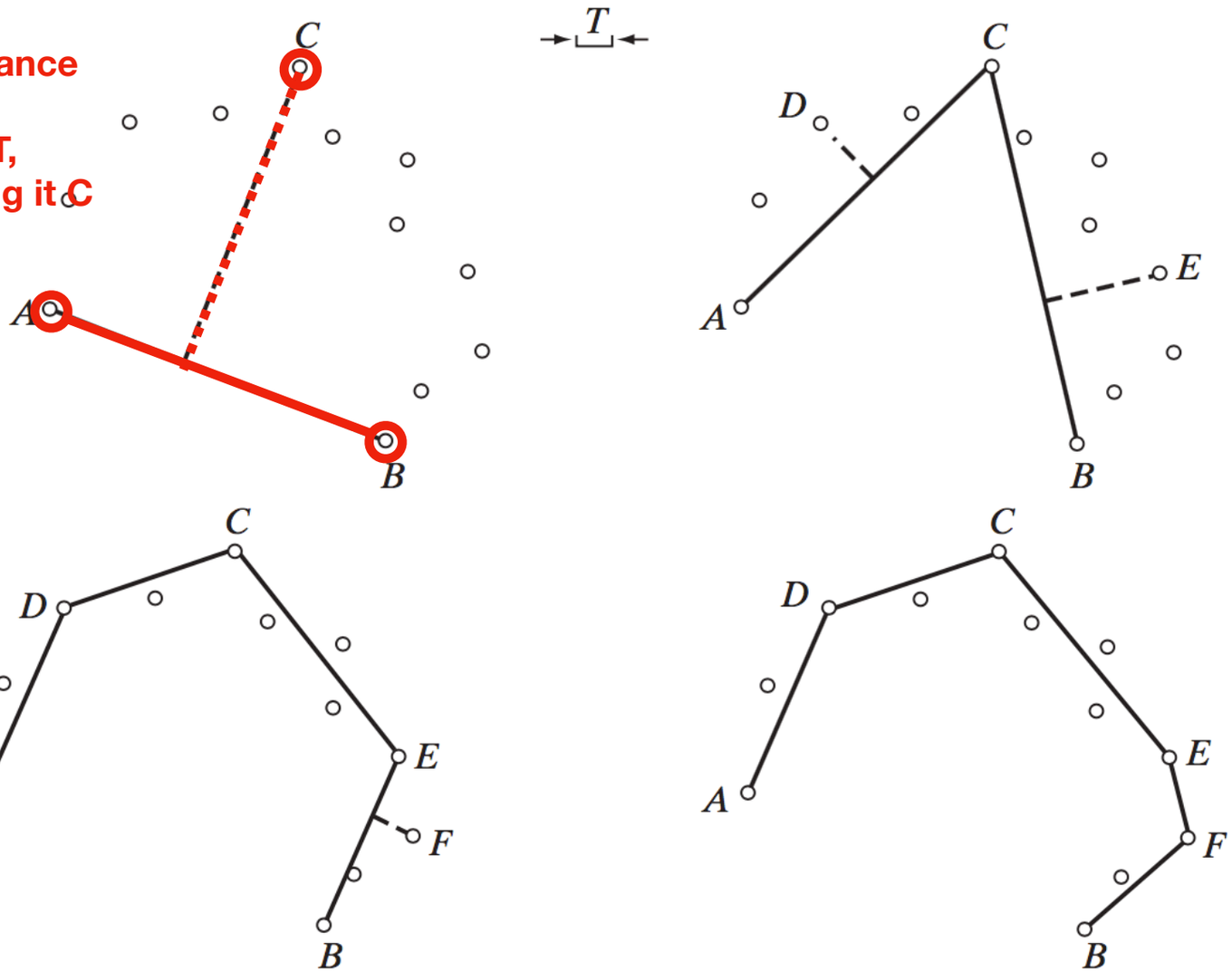


# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)

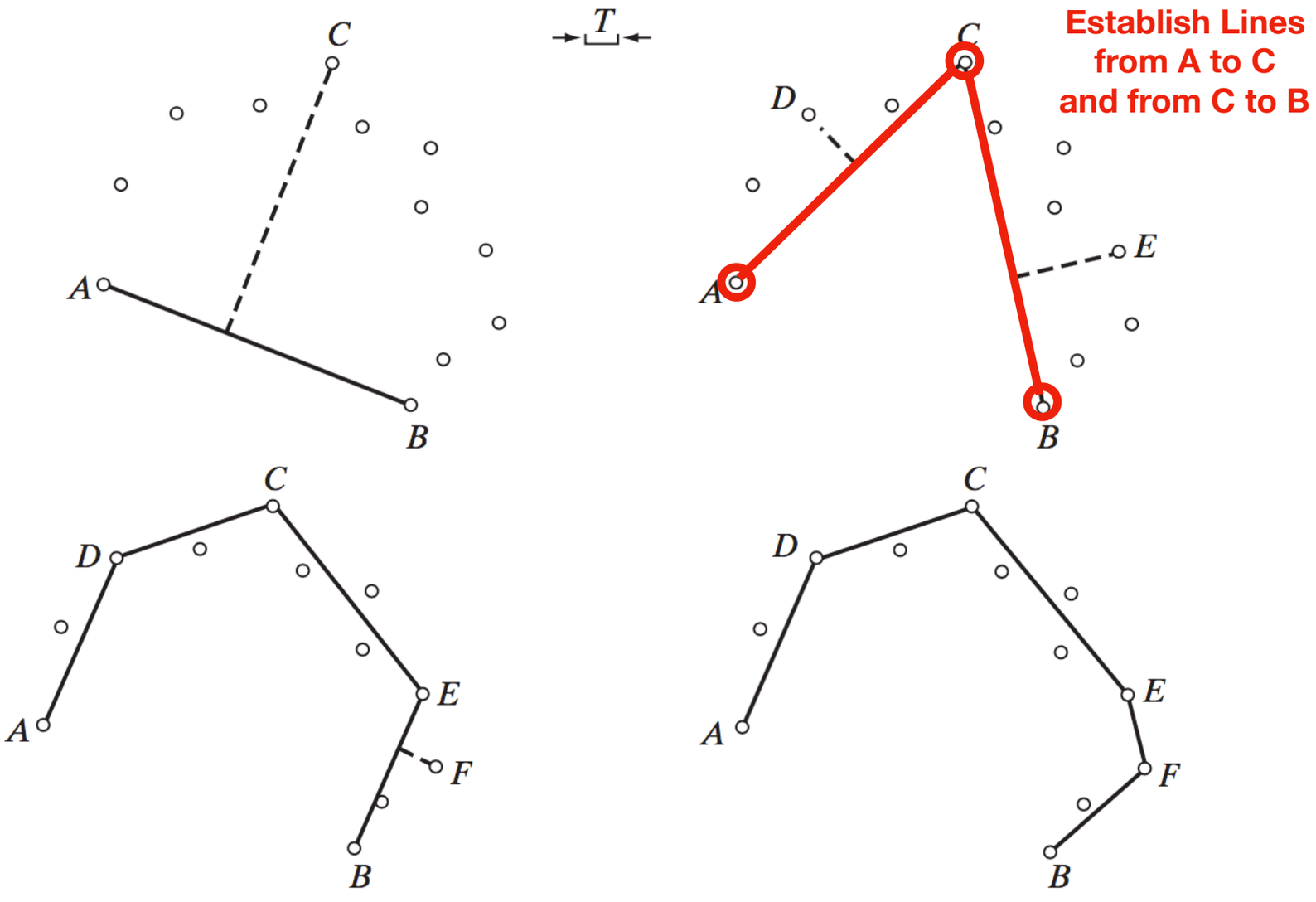
**Select the point that yielded the maximum distance**

**if exceeds thresholds T, declare the point by labeling it *C***



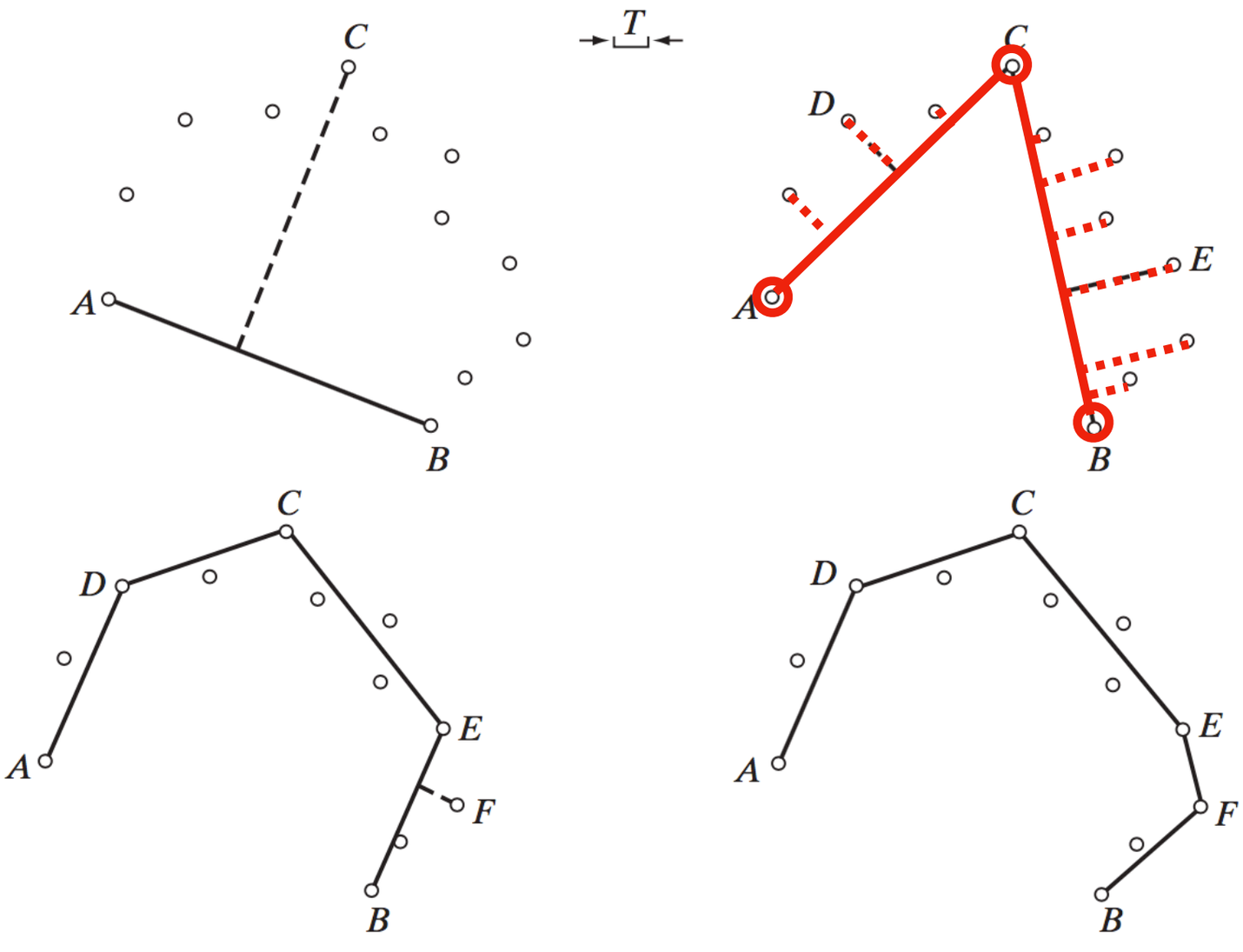
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)



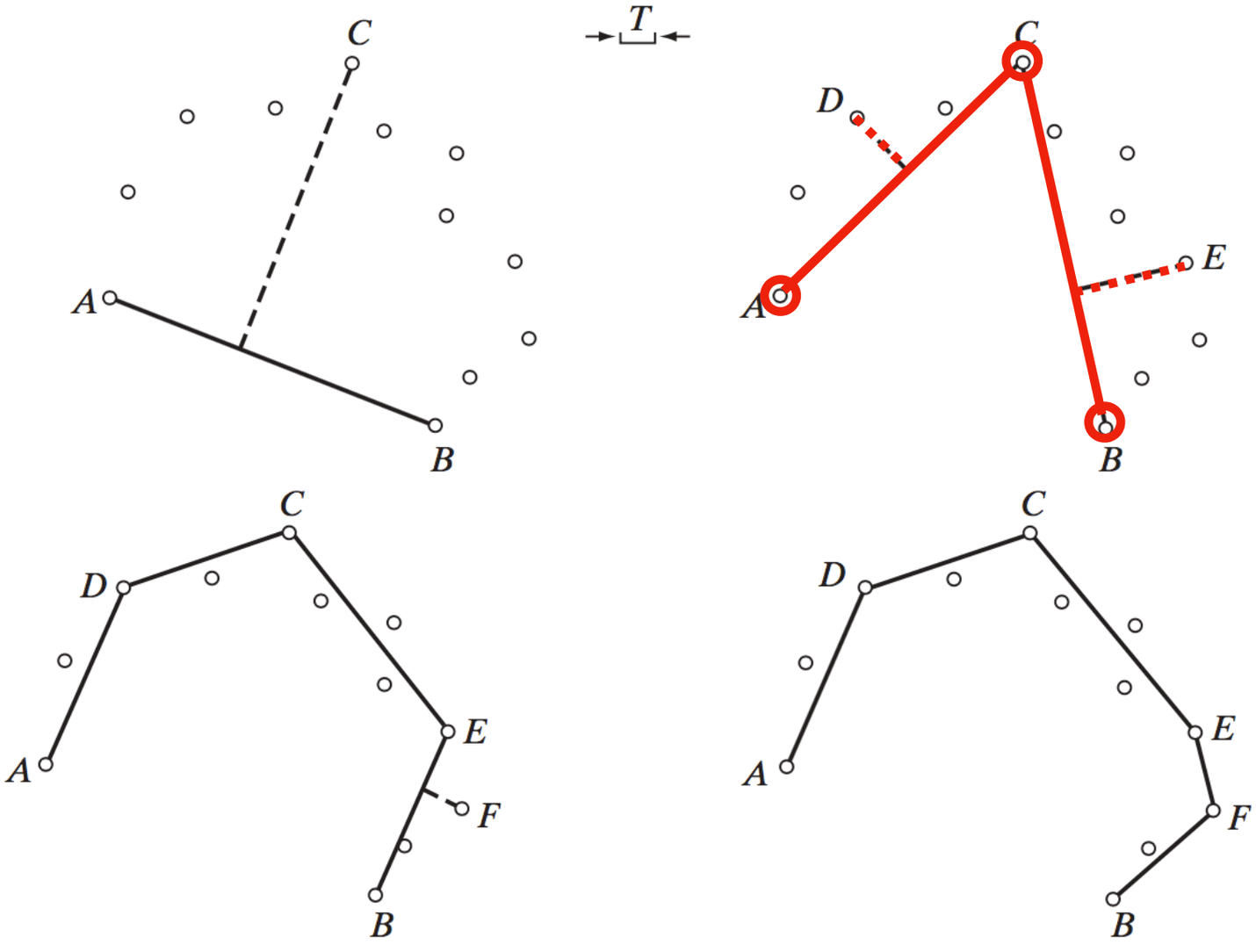
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)



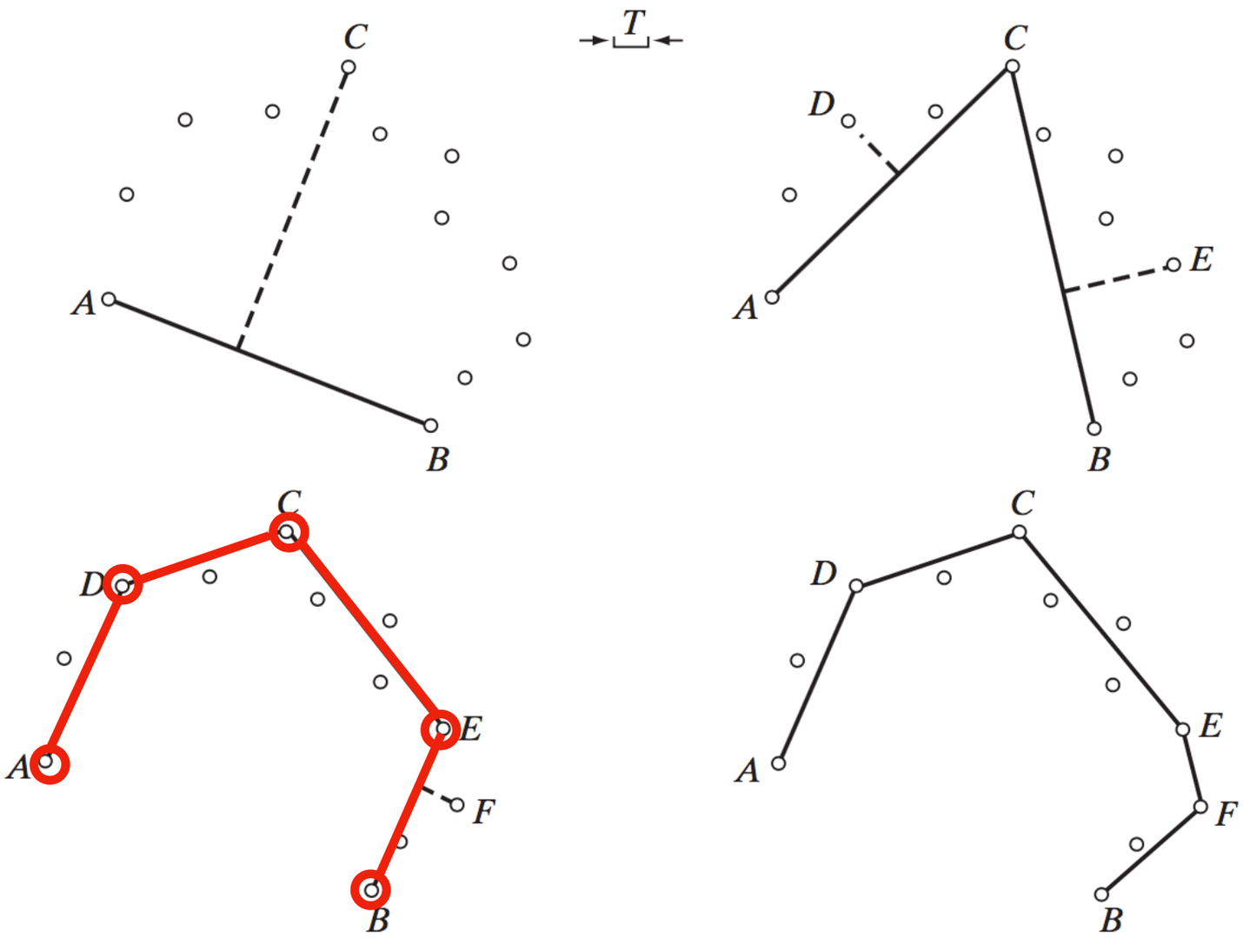
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)



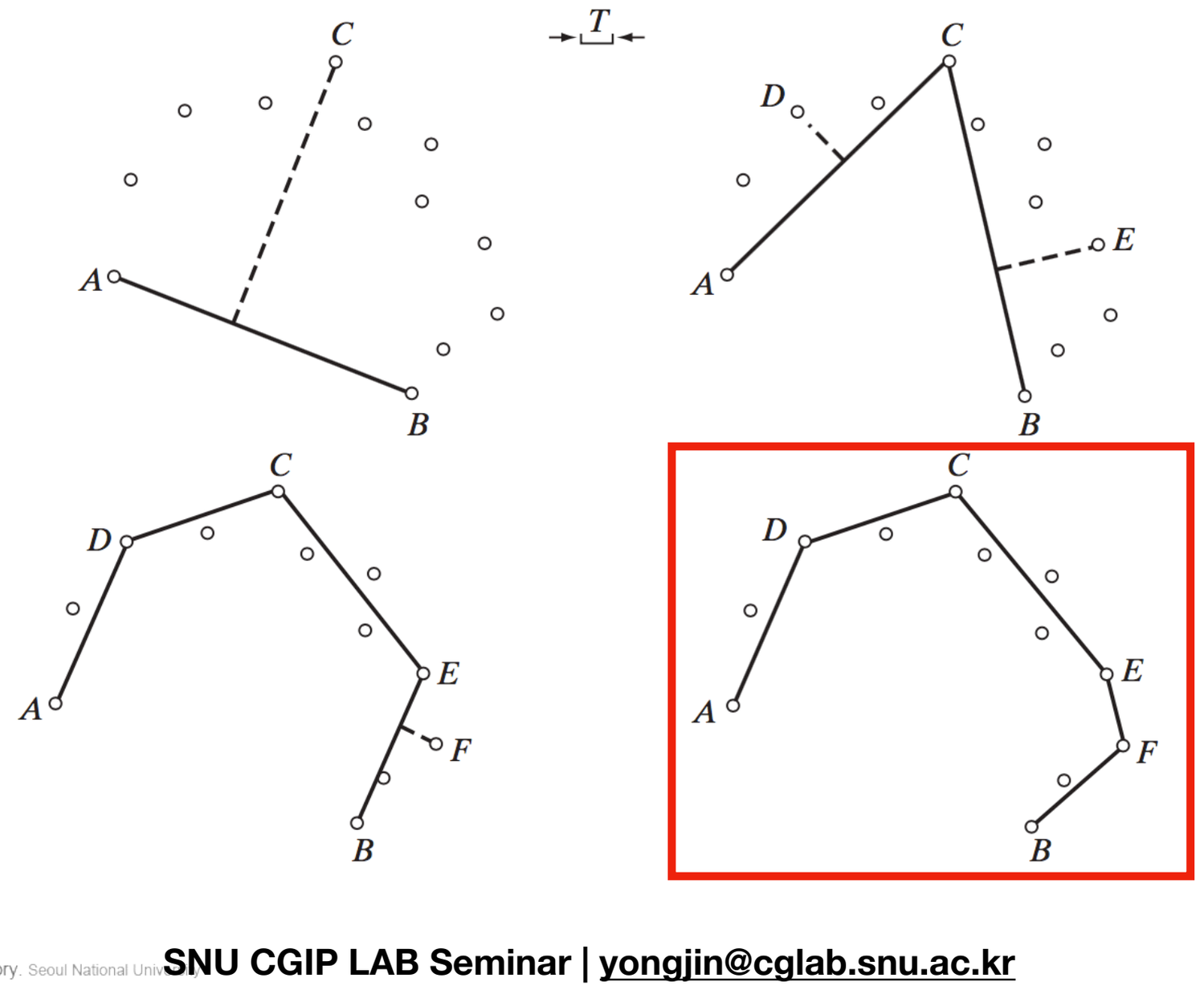
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Regional Processing
    => Polygonal Approximation (Polygonal fit algorithm)



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

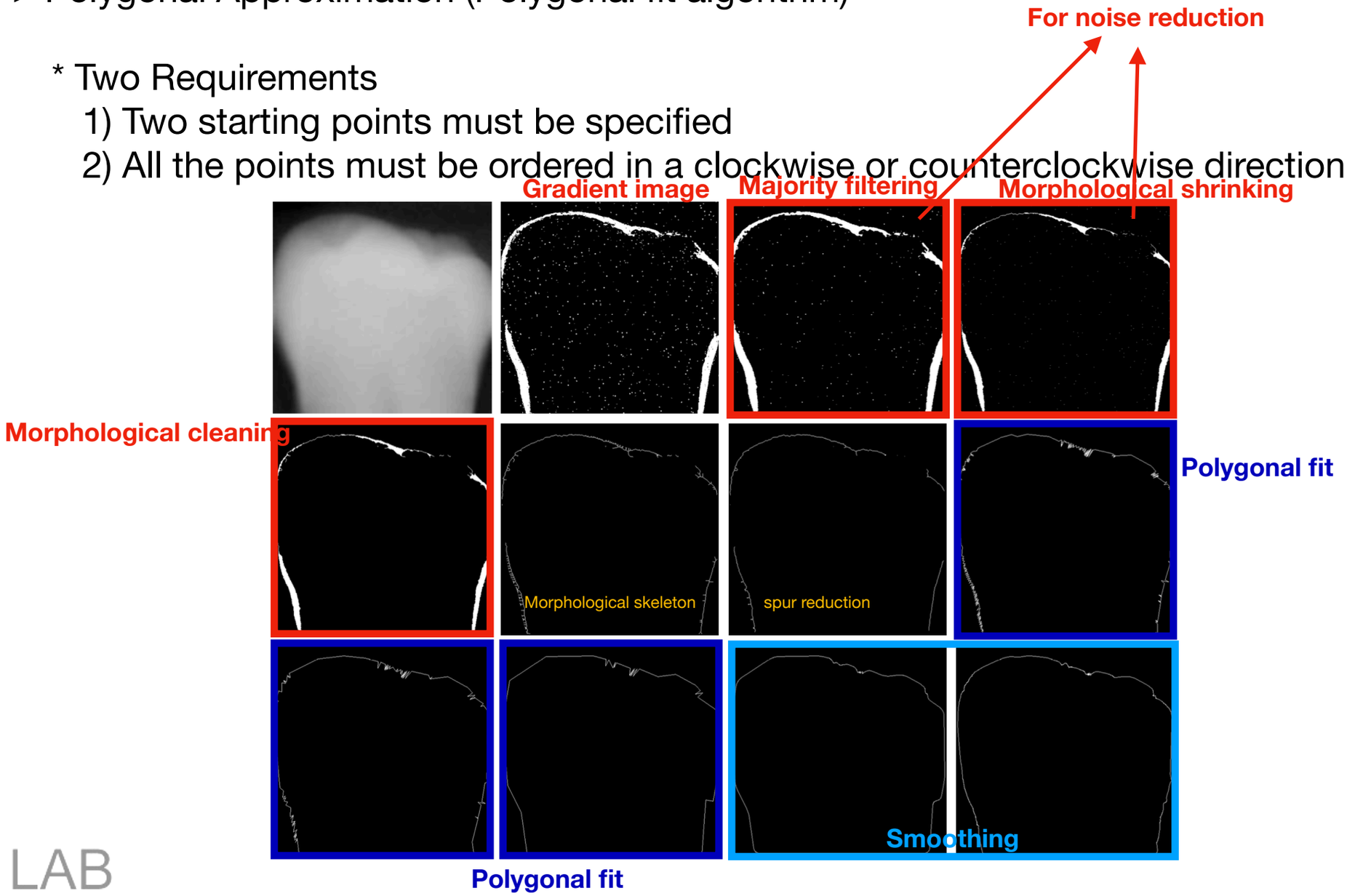
  -> Regional Processing

  => Polygonal Approximation (Polygonal fit algorithm)

  * Two Requirements

    1) Two starting points must be specified

    2) All the points must be ordered in a clockwise or counterclockwise direction



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Global processing using the Hough transform

  In the case of work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be
  => Need to develop the approach based on whether sets of pixels lie on curves of specified shape

  * with Given n points in an image,
  1) Find first all lines determined by every pair of points, and then find all subsets of points that are close to particular lines

$$O(n) = \frac{n(n-1)}{2} \sim n^2 \longrightarrow \textbf{finding lines}$$

$$O(n) = n \cdot \frac{n(n-1)}{2} \sim n^3 \longrightarrow \textbf{performing comparisons of every points}$$

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Global processing using the Hough transform

In the case of work with unstructured environments in which all we have is an edge image and no knowledge about where objects of interest might be

=> Need to develop the approach based on whether sets of pixels lie on curves of specified shape

* with Given n points in a **Computationally Expensive!**

1) Find first all lines determined by every pair of points, and then find all subsets of points that are close to particular lines

$$O(n) = \frac{n(n-1)}{2} \sim n^2 \longrightarrow \text{finding lines}$$

$$O(n) = n \cdot \frac{n(n-1)}{2} \sim n^3 \longrightarrow \text{performing comparisons of every points}$$

**Use Hough Transform!**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Global processing using the Hough transform

  => consider a point (xi, yi) in the xy-plane

  general equation form of straight line in slope-intercept form  $y_i = ax_i + b$

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Global processing using the Hough transform

  => consider a point (xi, yi) in the xy-plane

  general equation form of straight line in slope-intercept form  yi = axi + b

**Infinitely many lines pass through (xi, yi)**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Global processing using the Hough transform

  => consider a point (xi, yi) in the xy-plane

  general equation form of straight line in slope-intercept form  yi = axi + b

**Infinitely many lines pass through (xi, yi)**

**Convert the plane from xy into ab**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Global processing using the Hough transform

  => consider a point (xi, yi) in the xy-plane

  general equation form of straight line in slope-intercept form  yi = axi + b

**Infinitely many lines pass through (xi, yi)**

**Convert the plane from xy into ab**

**Converting equation to a single line for a fixed pair (xi, yi)**

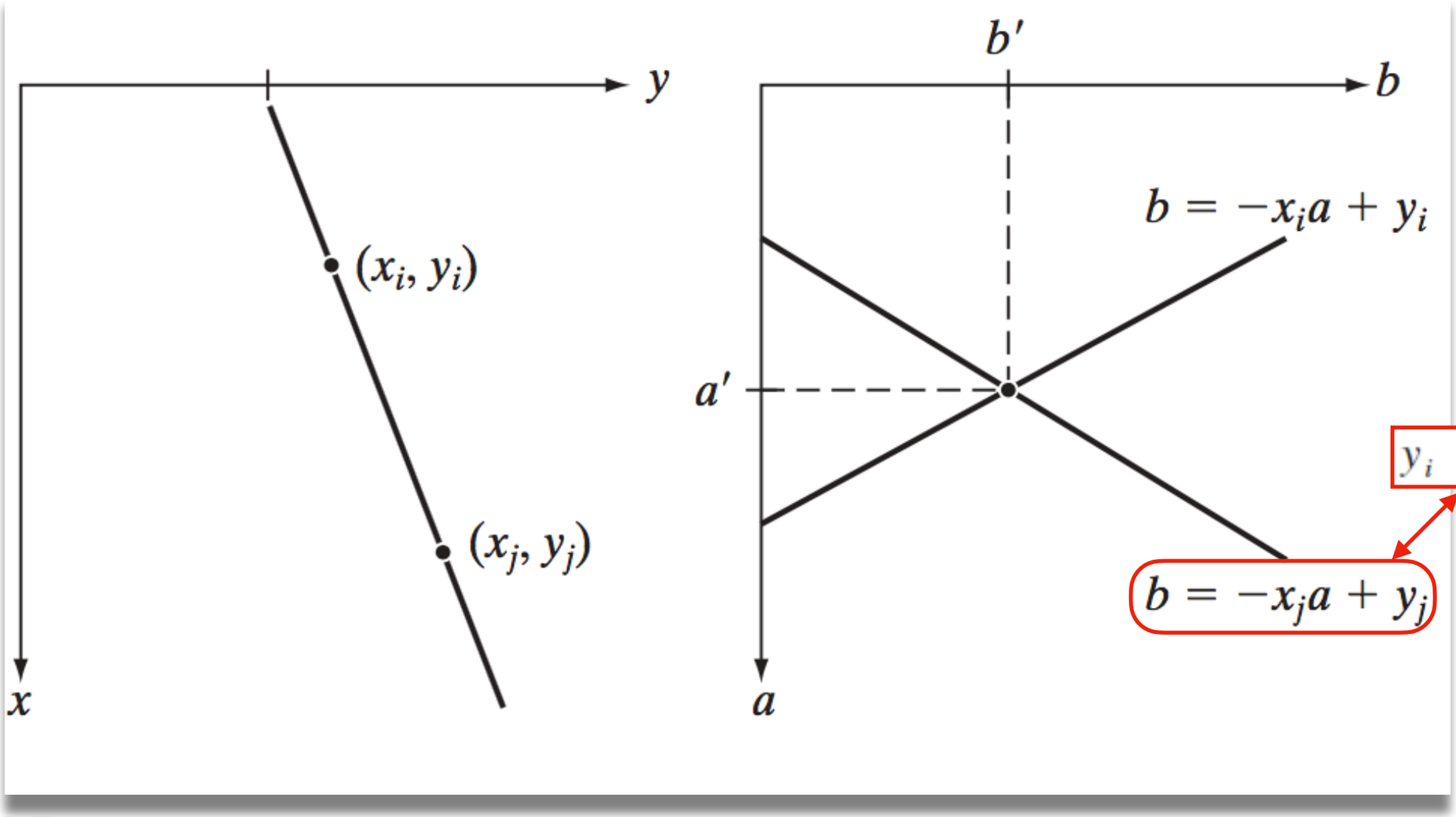
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Global processing using the Hough transform

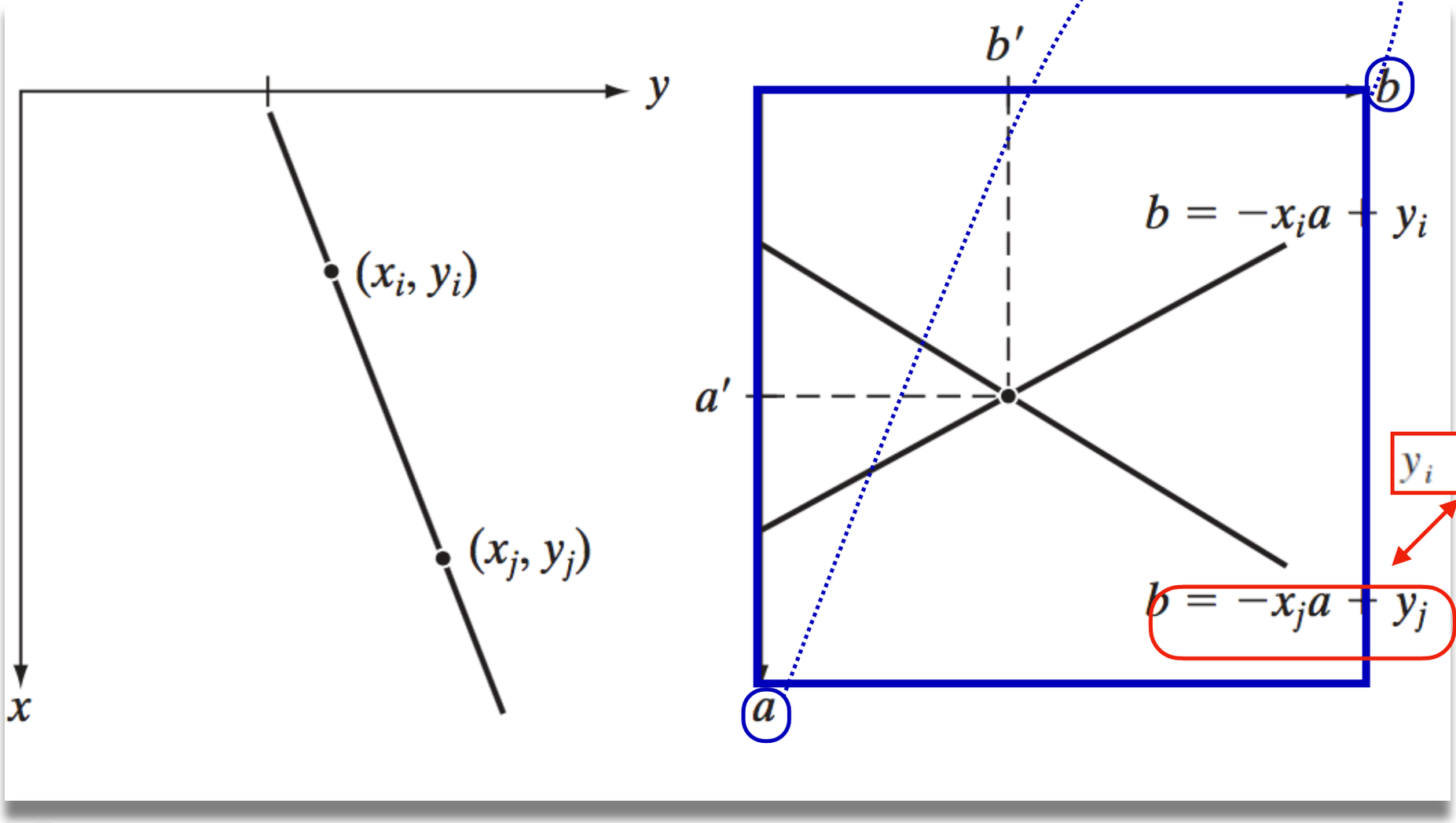
  => consider a point (xi, yi) in the xy-plane

  general equation form of straight line in slope-intercept form  yi = axi + b



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Global processing using the Hough transform
    => consider a point (xi, yi) in the xy-plane
      general equation form of straight line in slope-intercept form  yi = axi + b



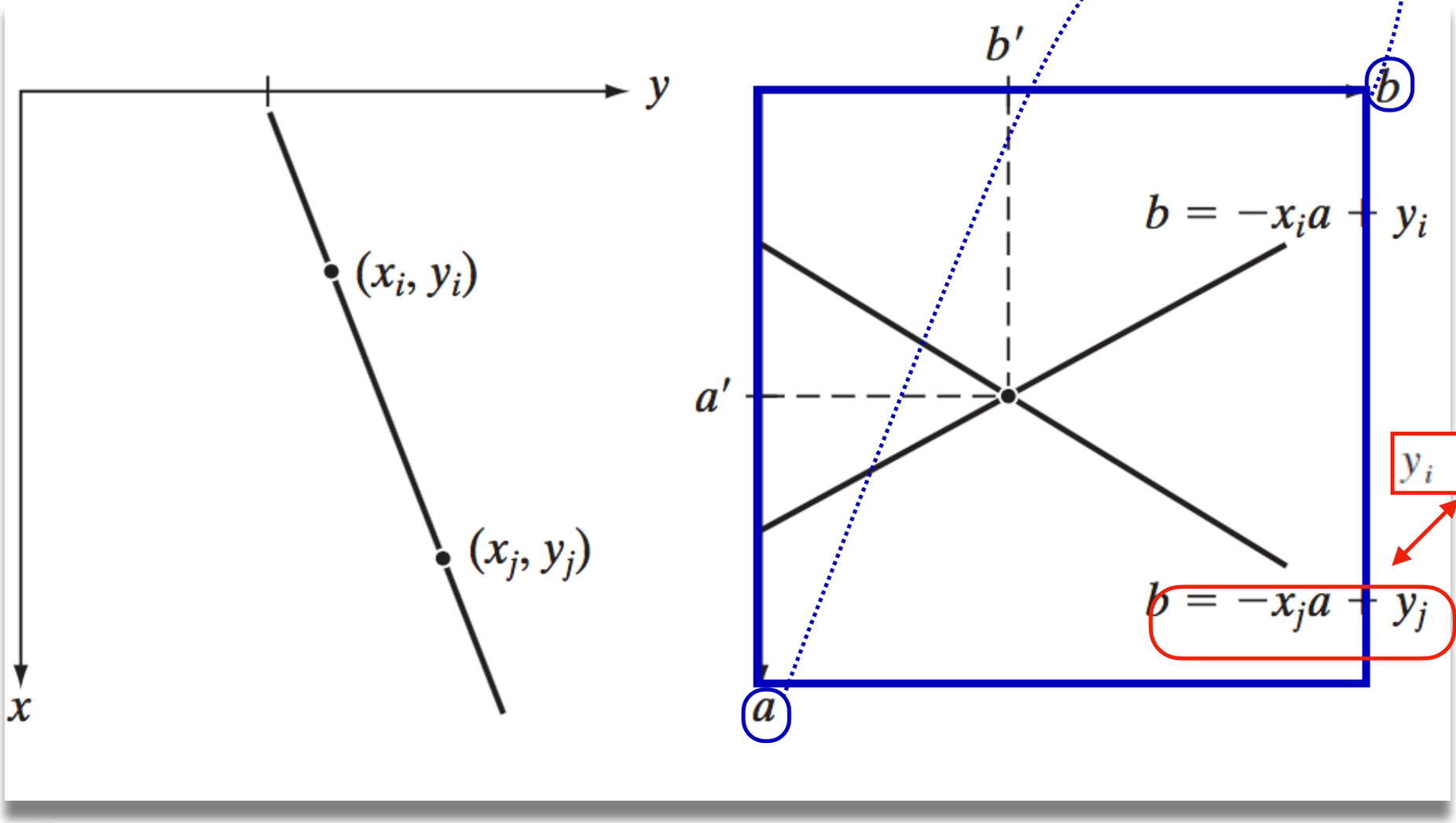
# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

  -> Global processing using the Hough transform

  => consider a point (xi, yi) in the xy-plane

  general equation form of straight line in slope-intercept form  yi = axi + b



# Point, Line, Edge Detection

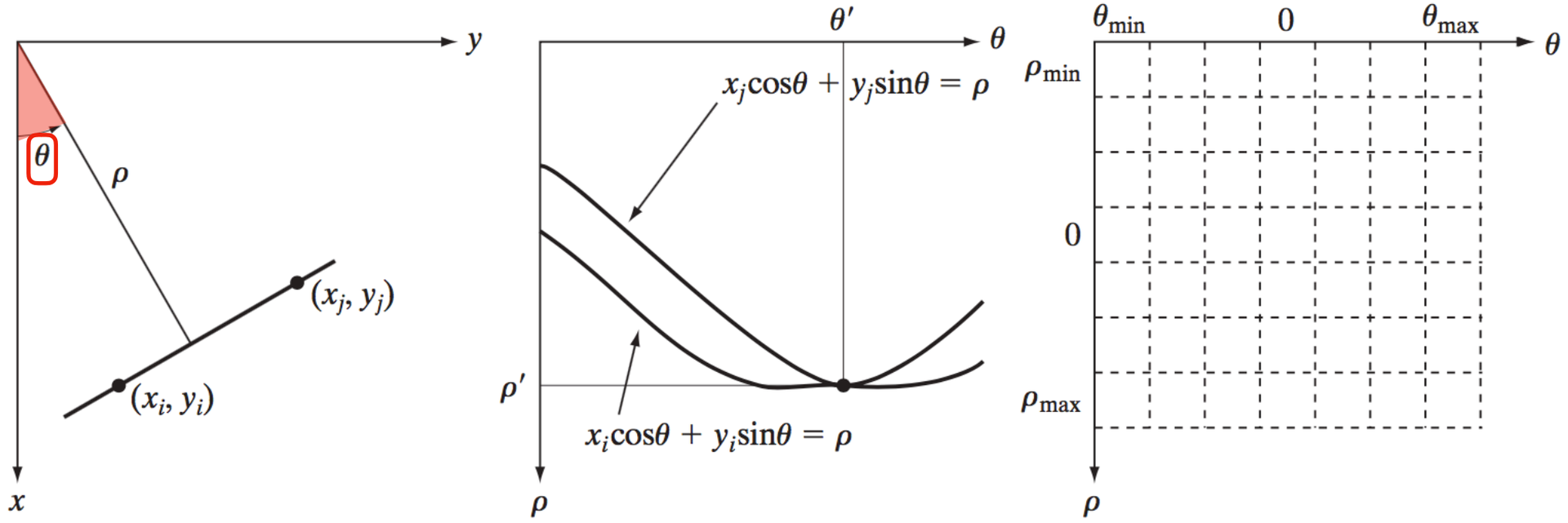
- Edge Linking and Boundary Detection

  -> Global processing using the Hough transform

  => Principal lines in the xy plane could be found by identifying points in parameter space where large numbers of **<u>parameter space lines intersect</u>**

  * Address by using normal representation of a line

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



# Point, Line, Edge Detection

- Edge Linking and Boundary Detection

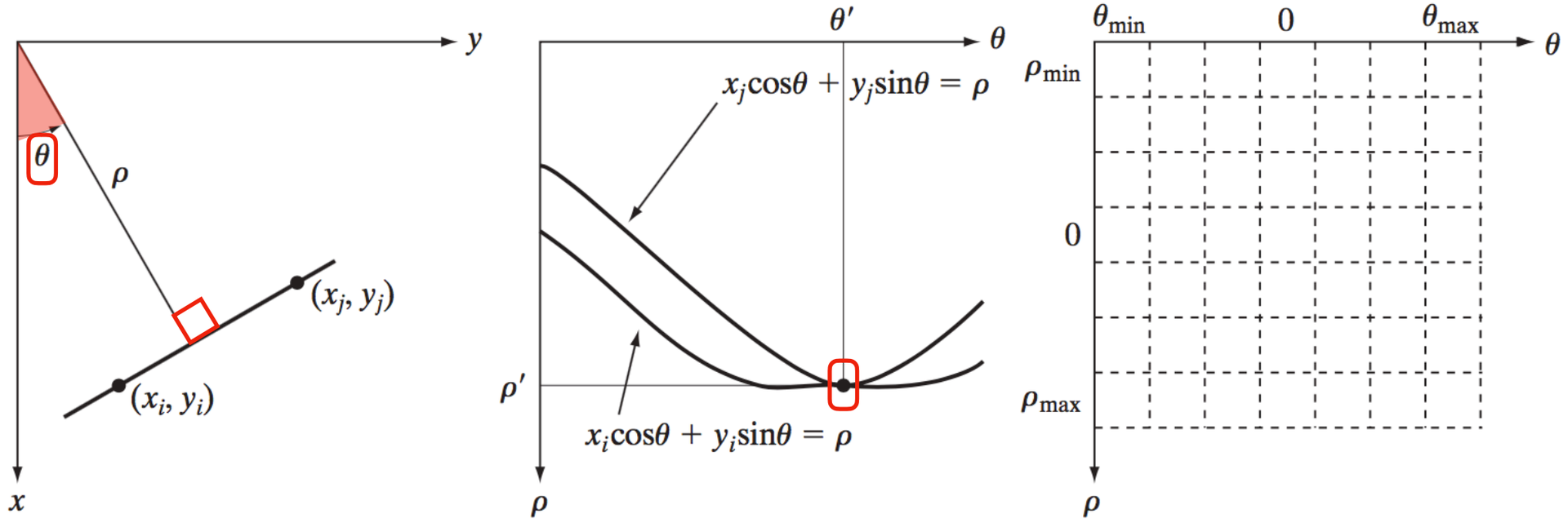
  -> Global processing using the Hough transform

  => Principal lines in the xy plane could be found by identifying points in parameter space where large numbers of **<u>parameter space lines intersect</u>**

  * Address by using normal representation of a line

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

**sinusoidal curves in rotheta-plane**




**geometric interpretation of the parameters ro and theta**

# Point, Line, Edge Detection

- Edge Linking

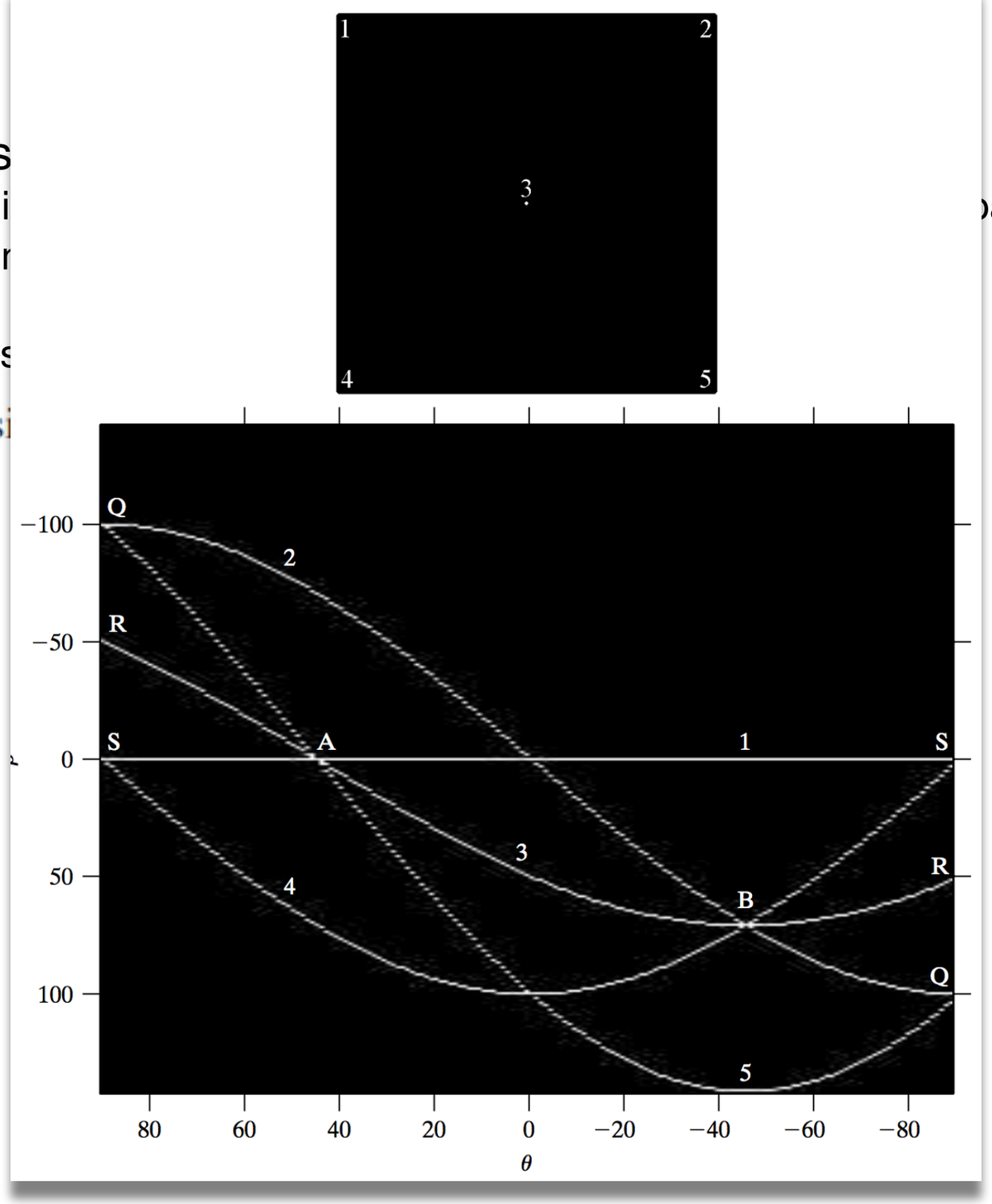
  -> Global process

  => Principal lines i

  where large nur

  * Address by us

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



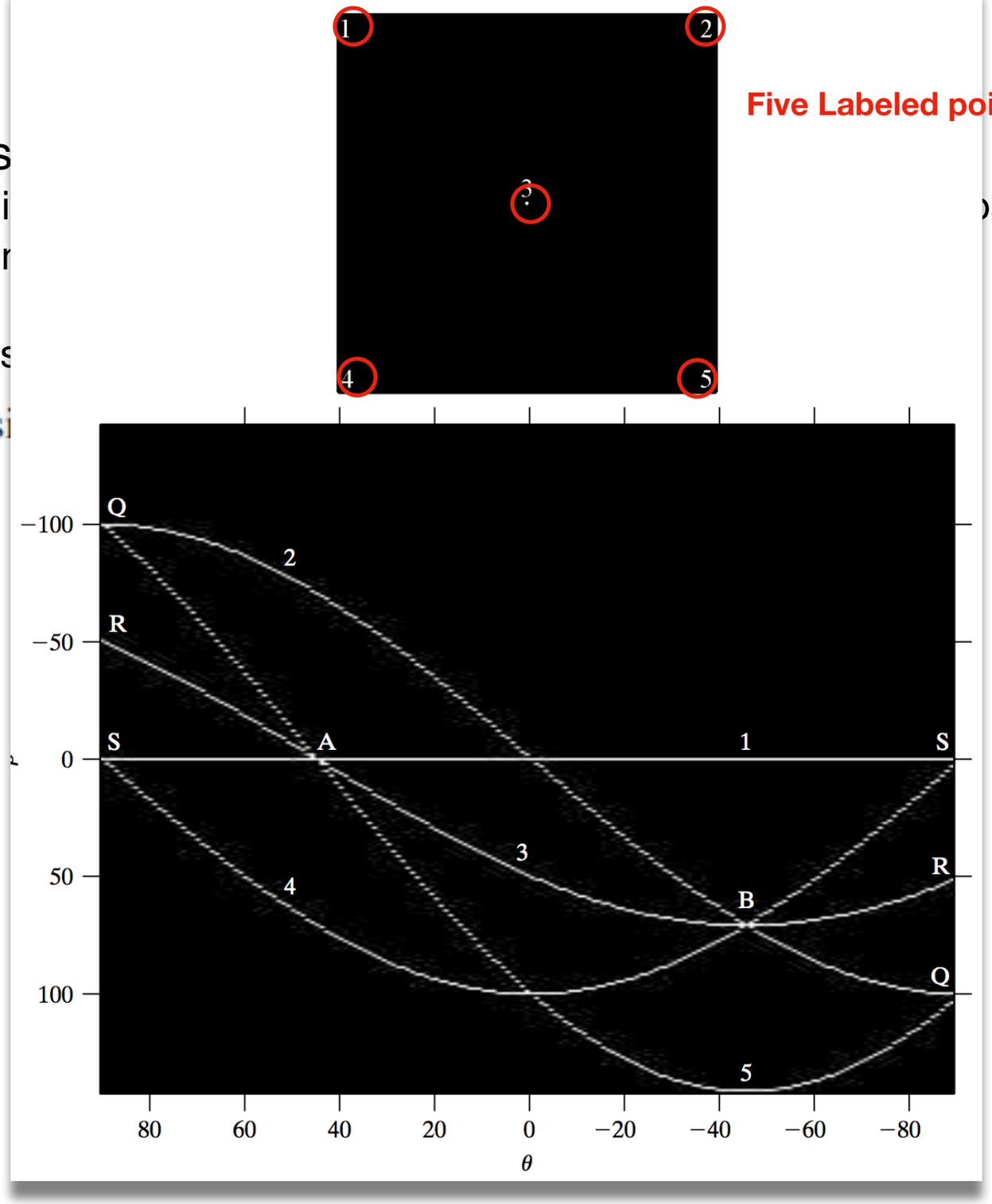

parameter space

# Point, Line, Edge Detection

- Edge Linking
  -> Global process
    => Principal lines i
    where large nur

    * Address by us

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




Five Labeled points

arameter space

# Point, Line, Edge Detection

- Edge Linking

  -> Global process

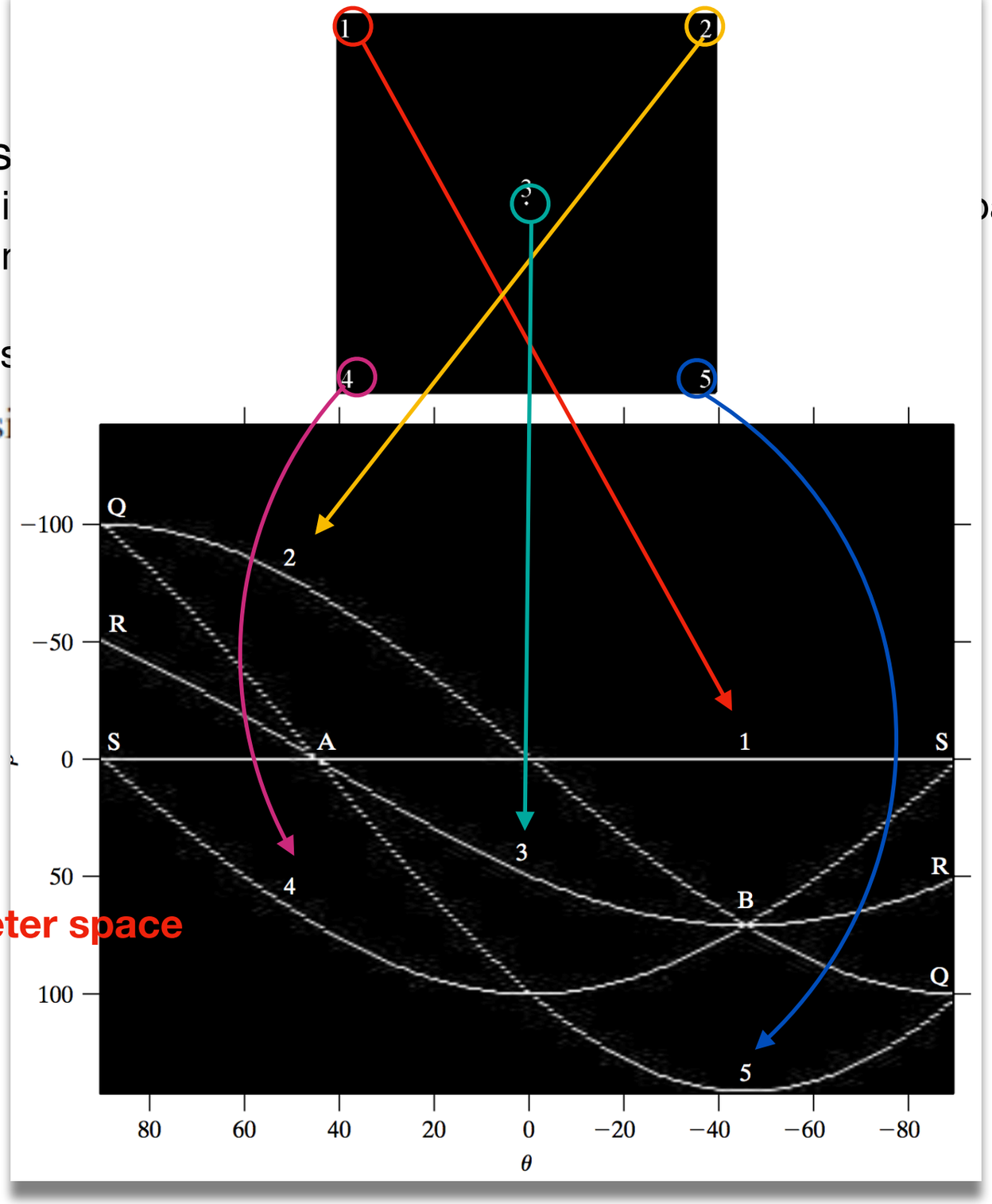
  => Principal lines i

  where large nur

  * Address by us

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

parameter space




**Corresponding parameter space**

# Point, Line, Edge Detection

- Edge Linking and Boundary Detection
  -> Hough transform Process

  1) Obtain a binary image using any of the techniques from the previous contents

  2) Specify subdivisions in the rotheta-plane

  3) Examine the counts of the accumulator cells for high pixel concentrations

  4) Examine the relationship between pixels in chosen cell

# Point, Line, Edge Detection

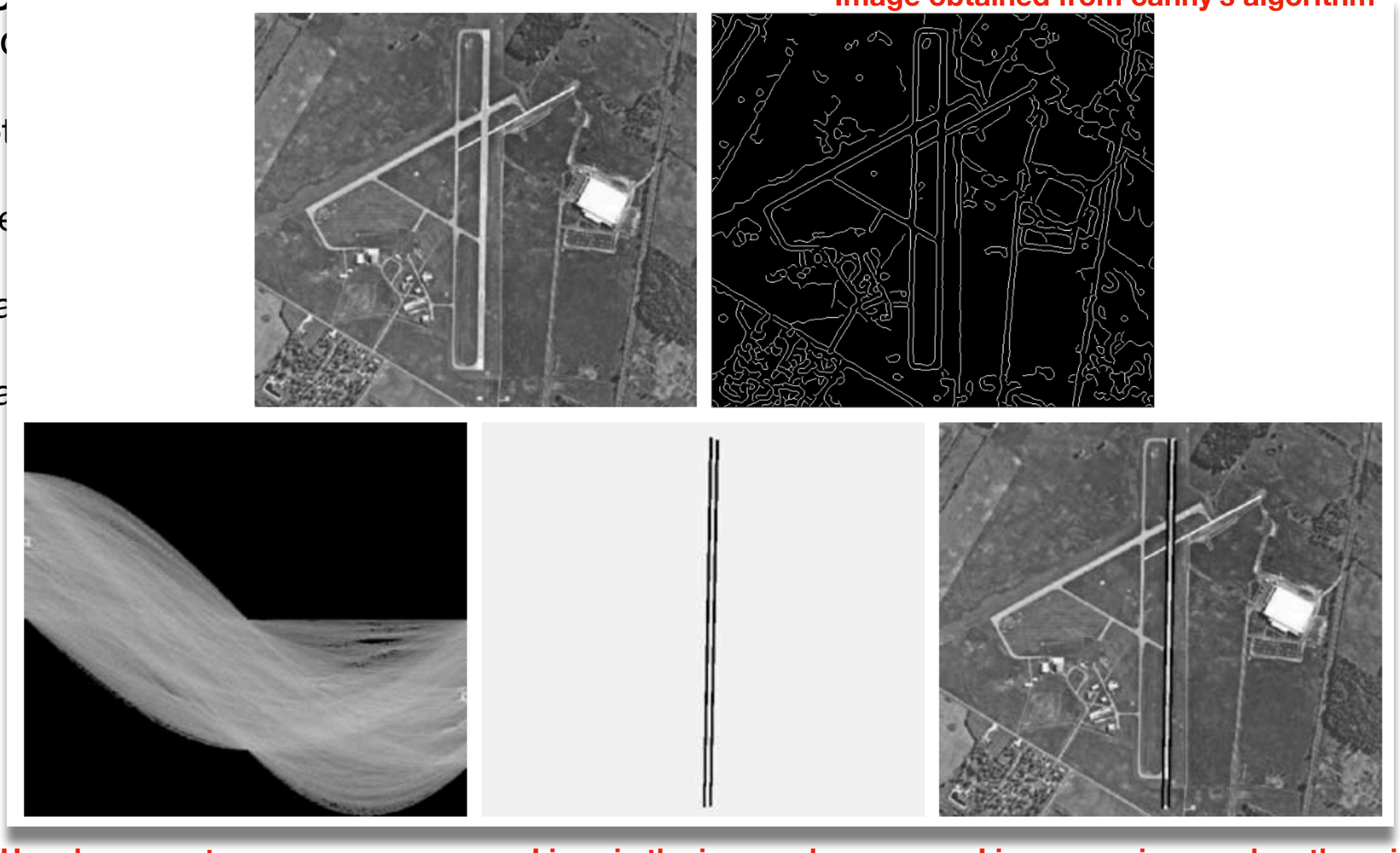
- Edge Linking and Boundary Detection
  -> He

  1) Obt

  2) Spe

  3) Exa

  4) Exa



# Thank you

## - Any Questions? -