Lecture 5 – Image Segmentation （图像分割）

This lecture will cover:

– Morphological Image Processing （形态学图像处理）

- Morphological operation
- Morphological algorithm

– Image Segmentation（图像分割）

- **Point, Line and Edge Detection（点、线和边缘检测）**
- Thresholding（阈值处理）
- Segmentation using Morphological Watersheds（形态学分水岭分割）

上海科技大学 ShanghaiTech University

Segmentation

- **Subdivide an image into its constituent regions or object**
  - Determine the level of subdivision details by the problem to be solved
  - Stop the subdivision when the regions or objects have been detected
- **Improve segmentation accuracy during acquisition**
  - Environment control
  - Sensor selection
  - Imaging modality

Definition

Image segmentation can be considered as a process that partitions region R into n subregions $R_1, R_2, \cdots, R_n$, such

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

b) R_i is a connected set, $i = 1,2, \cdots, n$

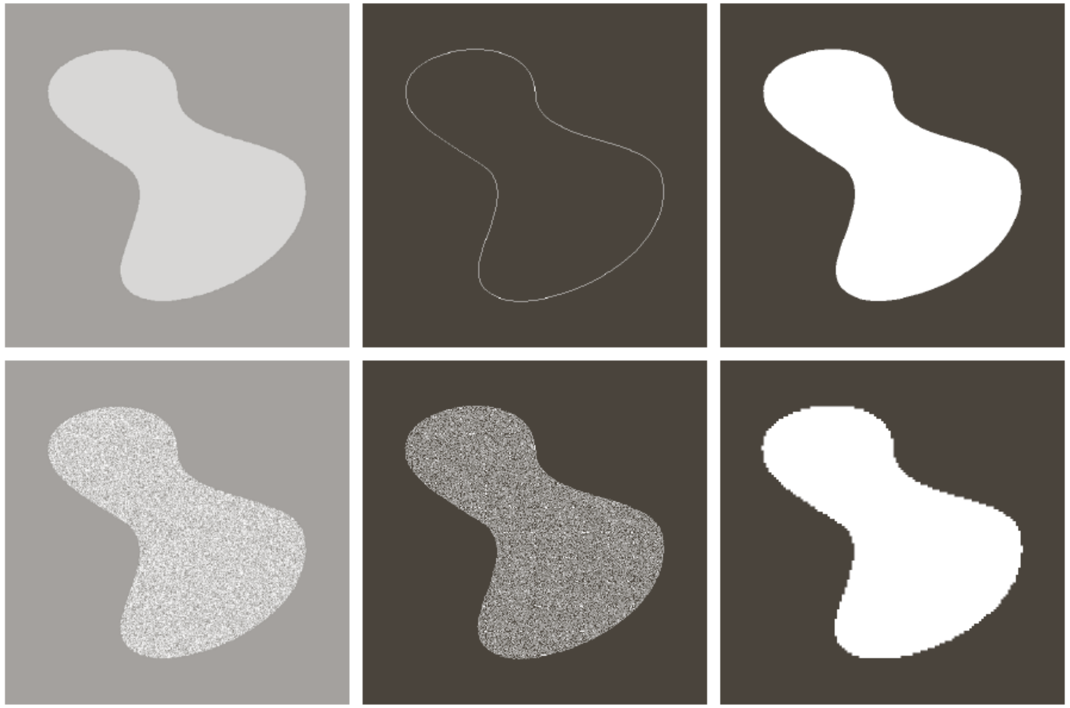
c) $R_i \cap R_j = \emptyset$ for all i and j, $i \neq j$

d) $Q(R_i) = True$ for $i = 1,2, \cdots, n$

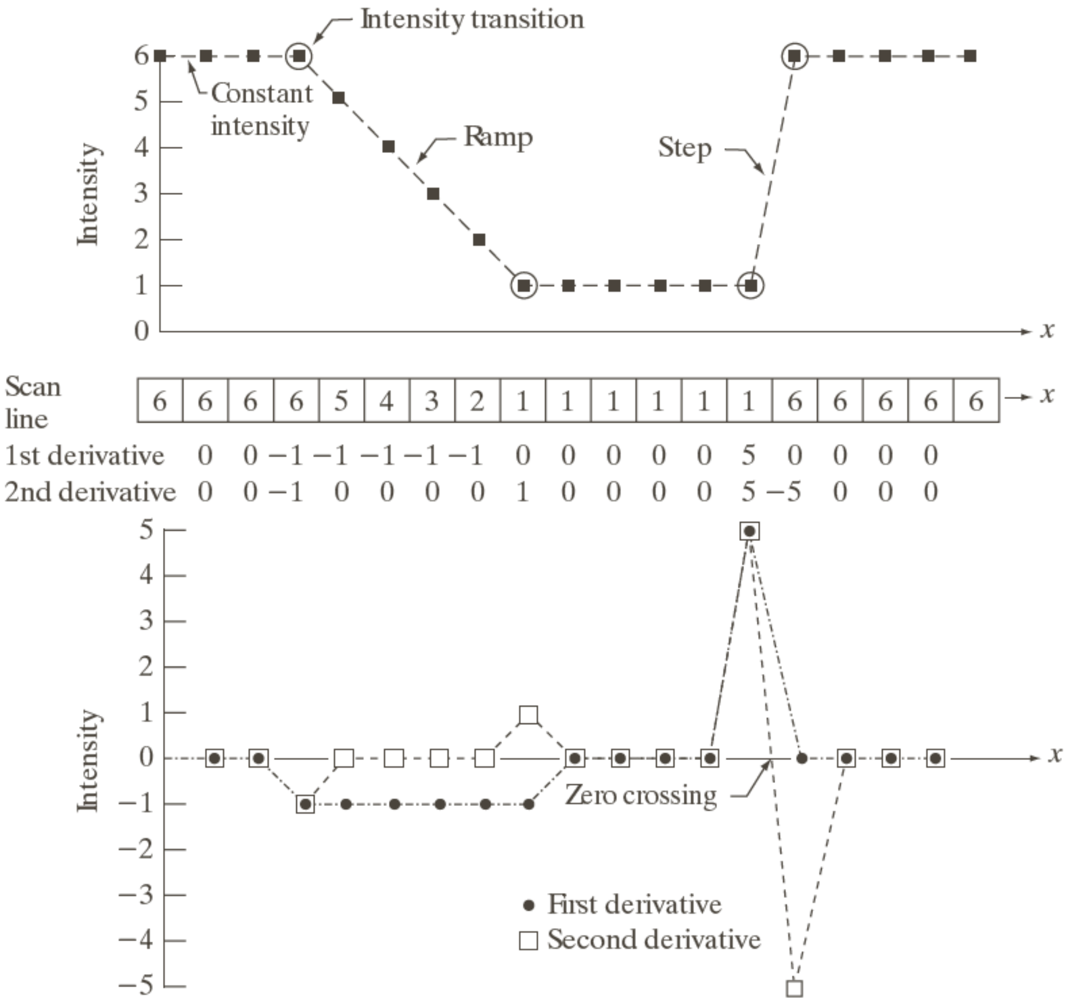
e) $Q(R_i \cup R_j) = False$ for any adjacent region R_i and R_j

Properties of Intensity values

- **Discontinuity** – Edge-based segmentation
- **Similarity** – Region-based segmentation



Derivatives



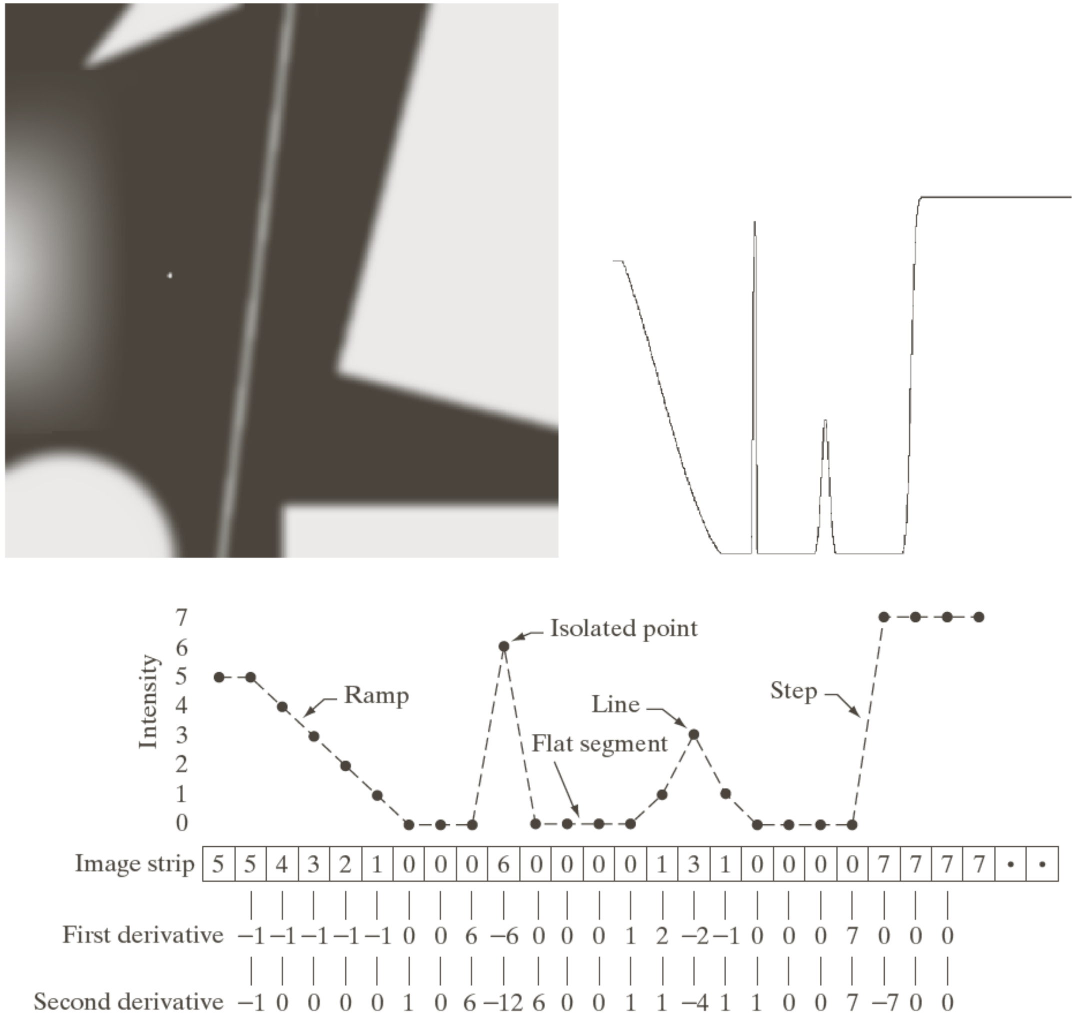
1. Zero in area of constant intensity

2. Nonzero at the onset of intensity step or ramp

3. (1) Nonzero along intensity ramp – 1^{st} order derivative

   (2) Zero along intensity ramp with constant slope – 2^{nd} order derivative

Edge Detection (边缘检测)



Spatial Filters

Vector Operation

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_{mn} z_{mn} = \sum_{k=1}^{mn} w_k z_k = w^T z$$

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Point Detection (点检测)

- **Output expression:**

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

Where

g: output image

T: nonnegative threshold

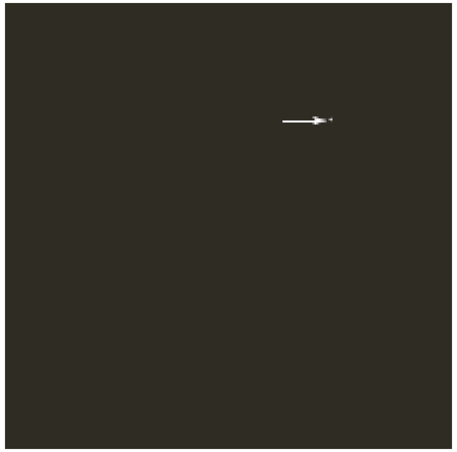
- **Matlab function:**

g = abs(imfilter(double(f)), w))>=T

g = ordfilt2(f,m*n,ones(m,n))-ordfilt2(f,1,ones(m,n)); g=g>=T

-1	-1	-1
-1	8	-1
-1	-1	-1

1	1	1
1	-8	1
1	1	1



Line Detection (线检测)

- **Output expression:**

$$|R_i| > |R_j|, \qquad j \neq i$$

Where R_i is the response to the masks

- **Matlab function:** g = abs(imfilter(double(f)), w))

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

2	−1	−1
−1	2	−1
−1	−1	2

+45°

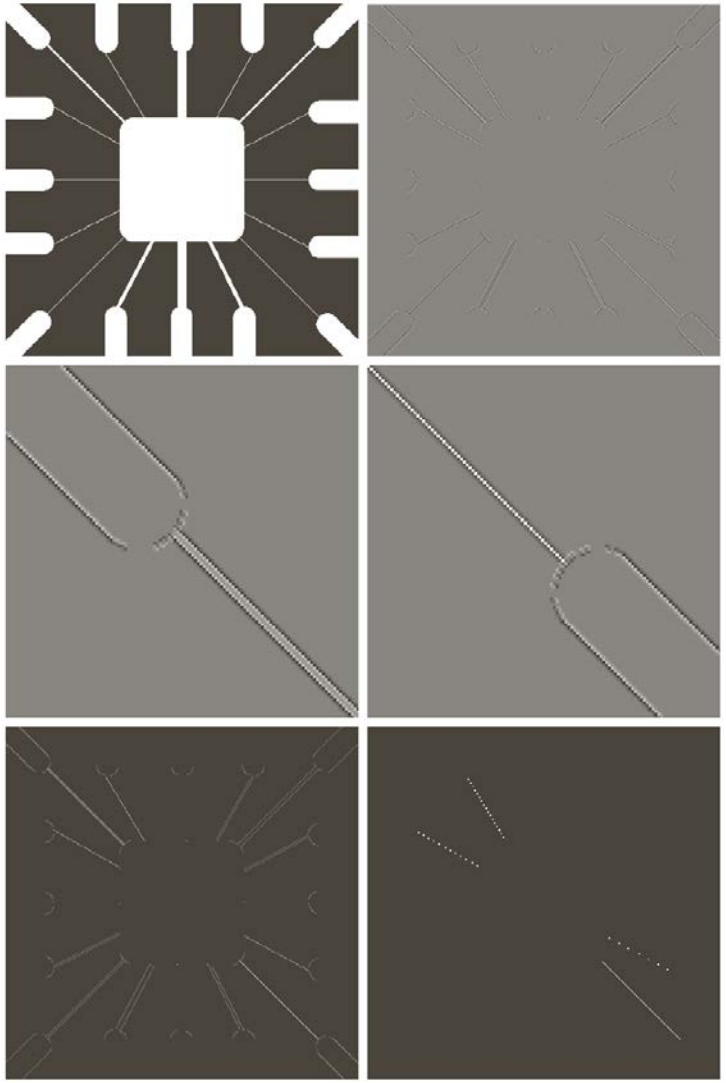
−1	2	−1
−1	2	−1
−1	2	−1

Vertical

−1	−1	2
−1	2	−1
2	−1	−1

−45°

Line Detection (线检测)



Edge Detection (边缘检测)

- **Rules:**
  - Seek the 1^{st} derivative greater than a threshold
  - Seek the zero-crossing point on the 2^{nd} derivative
- **Steps:**
  1. Image smoothing for noise reduction
  2. Detection of edge points
  3. Edge localization

Edge Detection (边缘检测)

Matlab function: [g, t] = edge(f, 'method', parameters)

Edge Detector	Description
Sobel	Finds edges using the Sobel approximation to the derivatives in Fig. 10.5(b)
Prewitt	Finds edges using the Prewitt approximation to the derivatives in Fig. 10.5(c).
Roberts	Finds edges using the Roberts approximation to the derivatives in Fig. 10.5(d).
Laplacian of a Gaussian (LoG)	Finds edges by looking for zero crossings after filtering $f(x, y)$ with a Laplacian of a Gaussian filter.
Zero crossings	Finds edges by looking for zero crossings after filtering $f(x, y)$ with a specified filter.
Canny	Finds edges by looking for local maxima of the gradient of $f(x, y)$. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. Therefore, this method is more likely to detect true weak edges.

Edge Detectors (边缘检测器)

Sobel

−1	−2	−1
0	0	0
1	2	1

$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$

−1	0	1
−2	0	2
−1	0	1

$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$

Prewitt

−1	−1	−1
0	0	0
1	1	1

$g_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$

−1	0	1
−1	0	1
−1	0	1

$g_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$

Roberts

−1	0
0	1

$g_x = z_9 - z_5$

0	−1
1	0

$g_y = z_8 - z_6$

Matlab function:

- Sobel Edge Detector

  [g, t] = edge(f, ‘sobel’, T, dir)

- Prewitt Edge Detector

  [g, t] = edge(f, ‘prewitt’, T, dir)

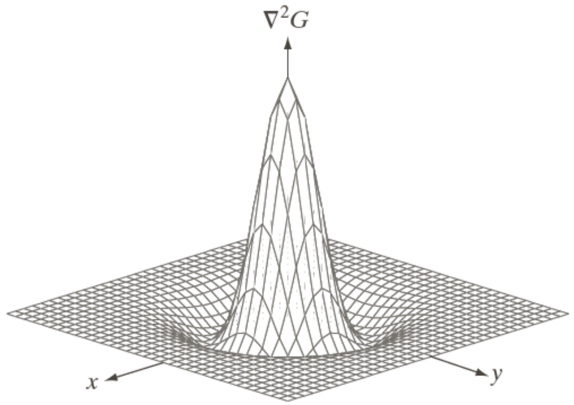
- Roberts Edge Detector

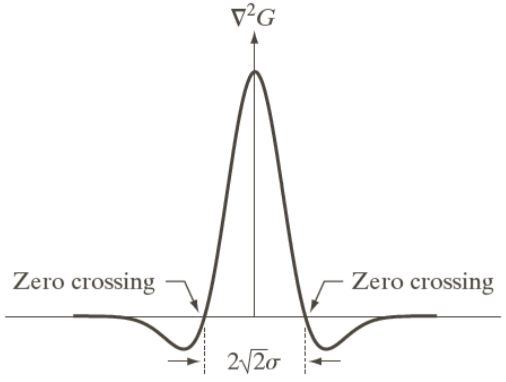
  [g, t] = edge(f, ‘Roberts’, T, dir)

Edge Detectors (边缘检测器)

- **LoG (Laplacian of a Gaussian, 高斯拉普拉斯算子):**

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





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

- **Matlab function:**
  - [g, t] = edge(f, ‘log’, T, sigma)
  - [g, t] = edge(f, ‘zerocross’, T, H)

Edge Detectors (边缘检测器)

➢ Canny Detector (坎尼边缘检测器):

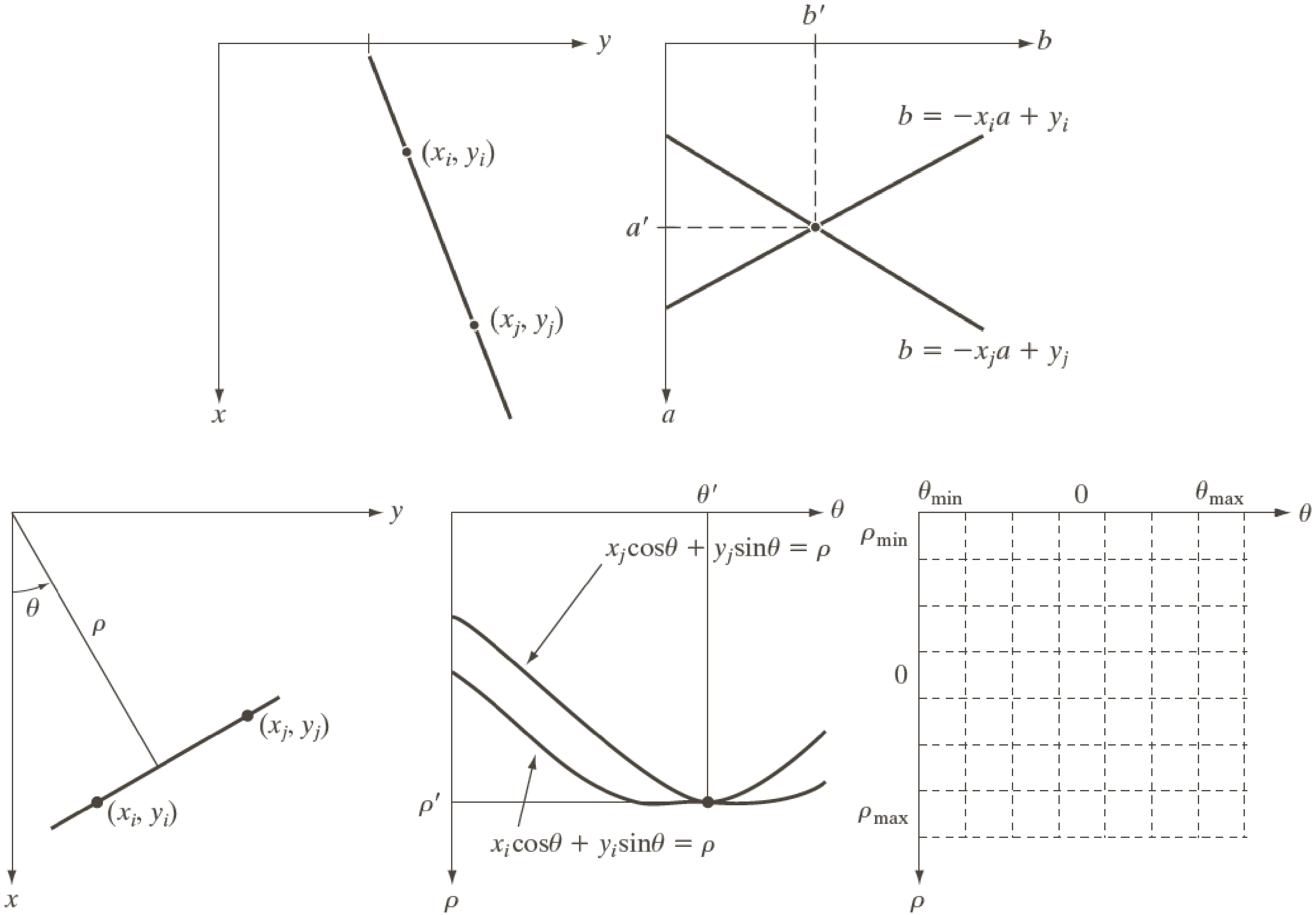
1. Smooth the input image with a Gaussian filter;
2. Compute the gradient magnitude and angle images;
3. Apply nonmaxima suppression (非最大值抑制) to the gradient magnitude image;
4. Use double thresholding and connectivity analysis to detect and link edge.

➢ Matlab function:

- [g, t] = edge(f, ‘canny’, T, sigma), where T=[T1, T2]

Edge Detectors (边缘检测器)

Hough Transform (霍夫变换)



Hough Transform (霍夫变换)

➢ An approach based on Hough Transform

1. Obtain a binary edge image using any edge detector;
2. Specify subdivisions in the $\rho\theta$-plane;
3. Examine the counts of the accumulator cells (累加器单元) for high pixel concentrations;
4. Examine the relationship between pixels in a chosen cell.

➢ Matlab function:

- [H, theta, rho] = hough(f);
- peaks = houghpeaks(H, NumPeaks)
- lines = houghlines(f, theta, rho, peaks)

Hough Transform (霍夫变换)