



IMAGE PROCESSING

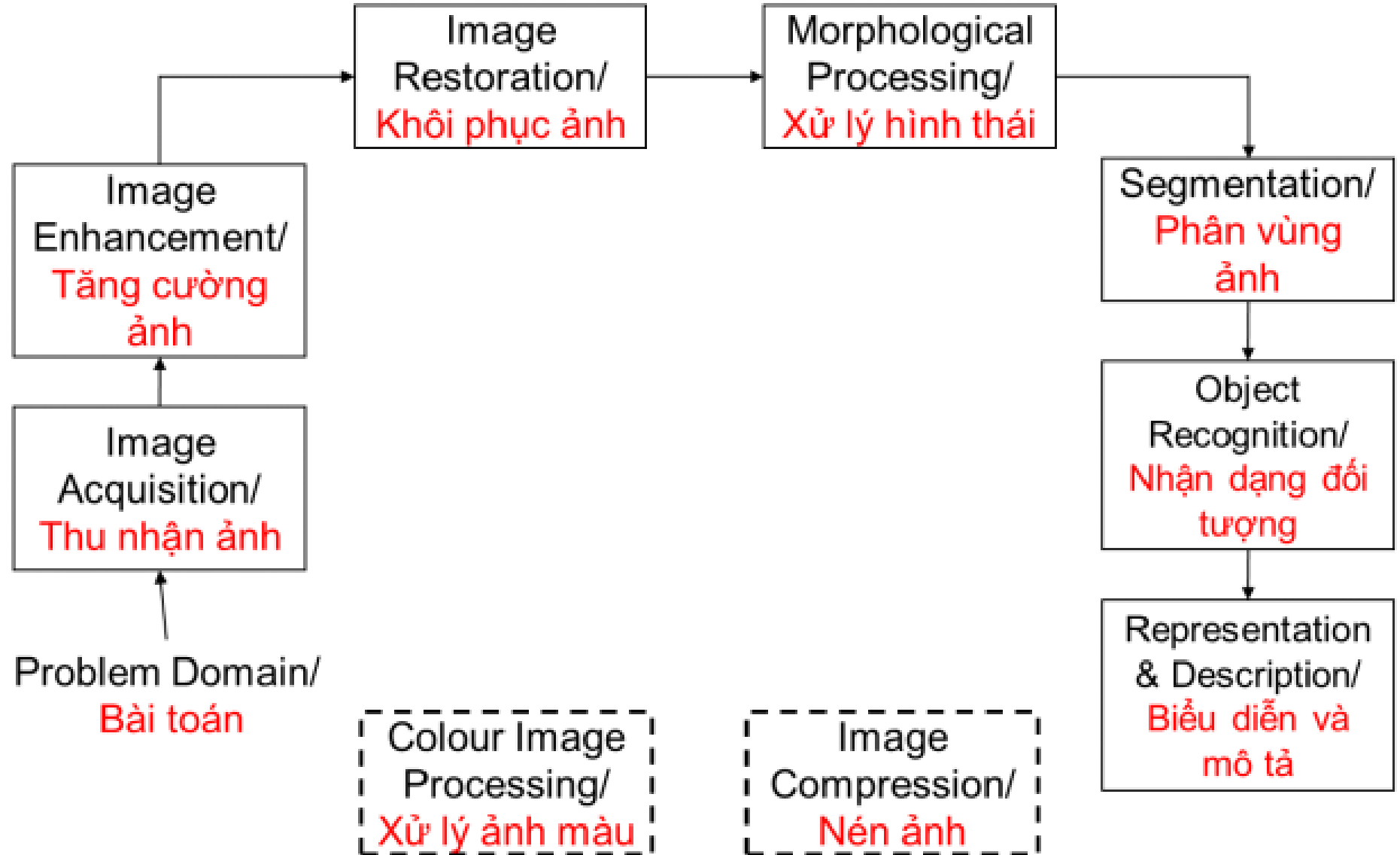
Segmentation

Dr. Cao Thị Luyen

Email: luyenct@utc.edu.vn

Tel: 0912403345

Segmentation position in image processing system





CONTENT

2

Basic Concepts

Edge, line, and point detection in images

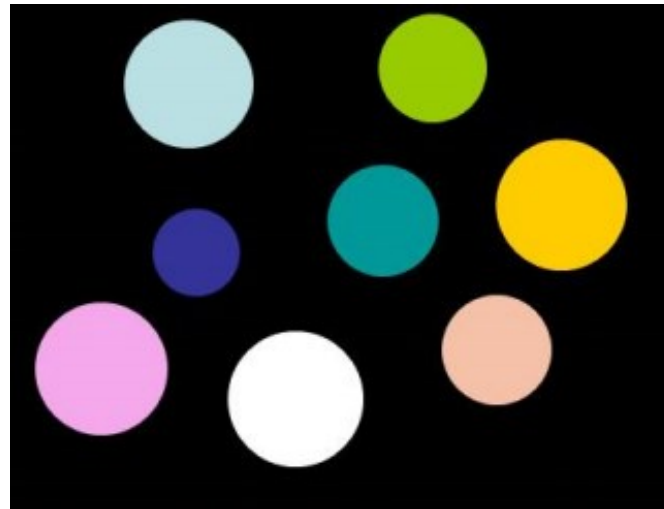
Thresholding

Segmentation by region growing, splitting, and merging



Basic Concepts

- Segmentation is the process of dividing the pixels of an image into groups that are closely related to the objects in the image.
- Segmentation is often the first step in any automatic computer vision application



Basic Concepts

- Let R represent the entire spatial region occupied by an image. We may view image segmentation as a process that partitions R into n subregions, R_1, R_2, \dots, R_n such that

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

(b) R_i is a connected set, for $i = 0, 1, 2, \dots, n.$

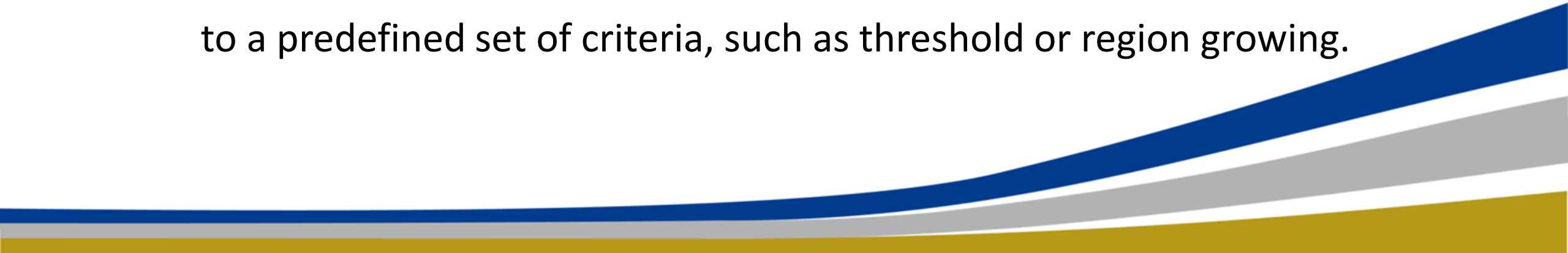
(c) $R_i \cap R_j = \emptyset$ for all i and j , $i \neq j.$

(d) $Q(R_i) = \text{TRUE}$ for $i = 0, 1, 2, \dots, n.$

(e) $Q(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and $R_j.$

Basic Concepts

- Image segmentation is based on one of two fundamental properties of intensity values:
 - Discontinuity: partitioning the image into regions based on sudden changes in intensity, such as edges.
 - Similarity: partitioning the image by dividing it into similar regions according to a predefined set of criteria, such as threshold or region growing.



Basic Concepts

a	b	c
d	e	f

FIGURE 10.1

(a) Image of a constant intensity region.

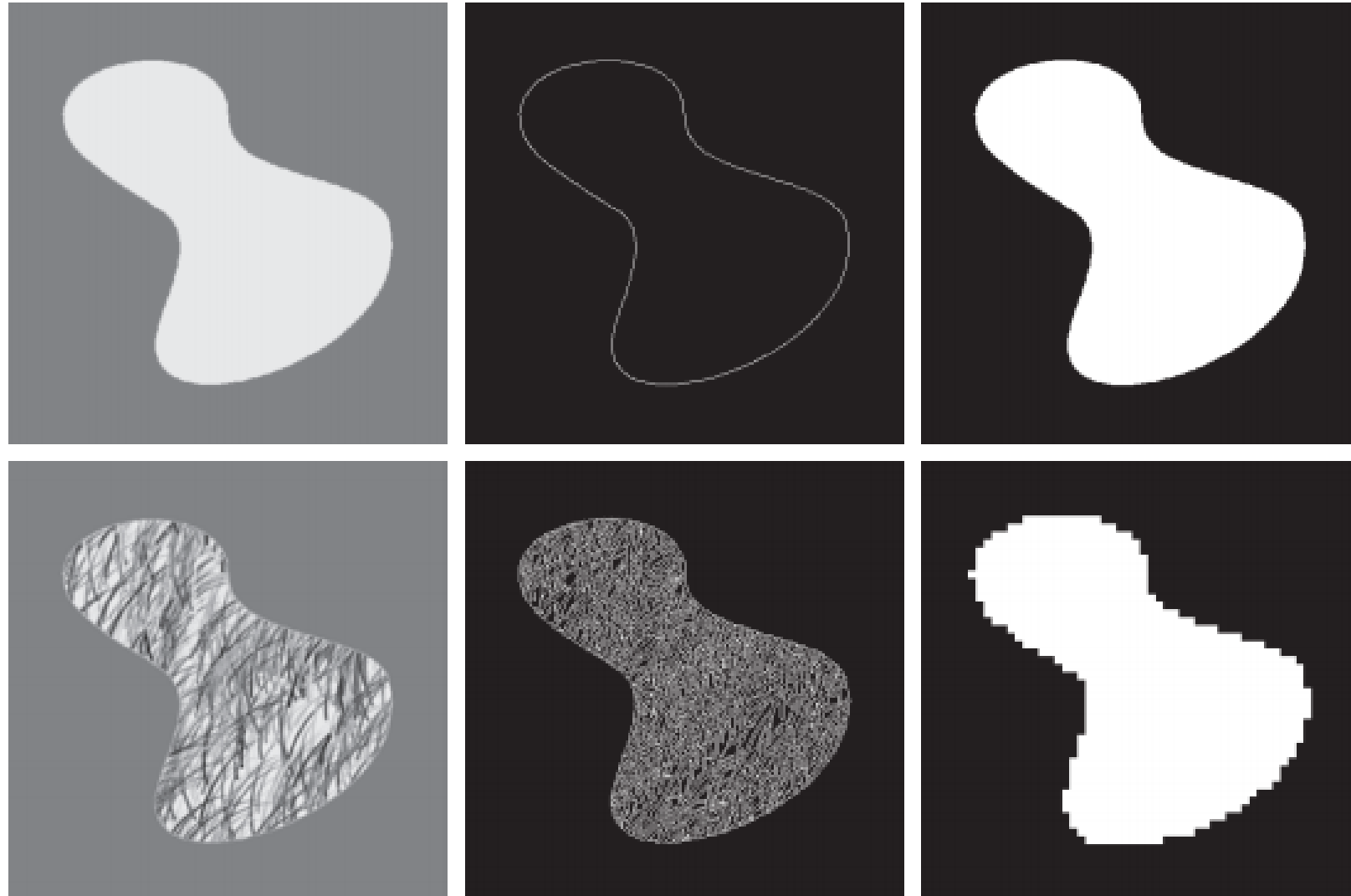
(b) Boundary based on intensity discontinuities.

(c) Result of segmentation.

(d) Image of a texture region.

(e) Result of intensity discontinuity computations (note the large number of small edges).

(f) Result of segmentation based on region properties.



Discontinuity detection

There are three types of gray level discontinuities

- Points
- Lines
- Edges

We often search for discontinuities using masks and correlations



Method

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

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

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

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

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

Point detection

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

A point at the position (x, y) is detected if the absolute value of the response at that location is greater than a certain threshold.

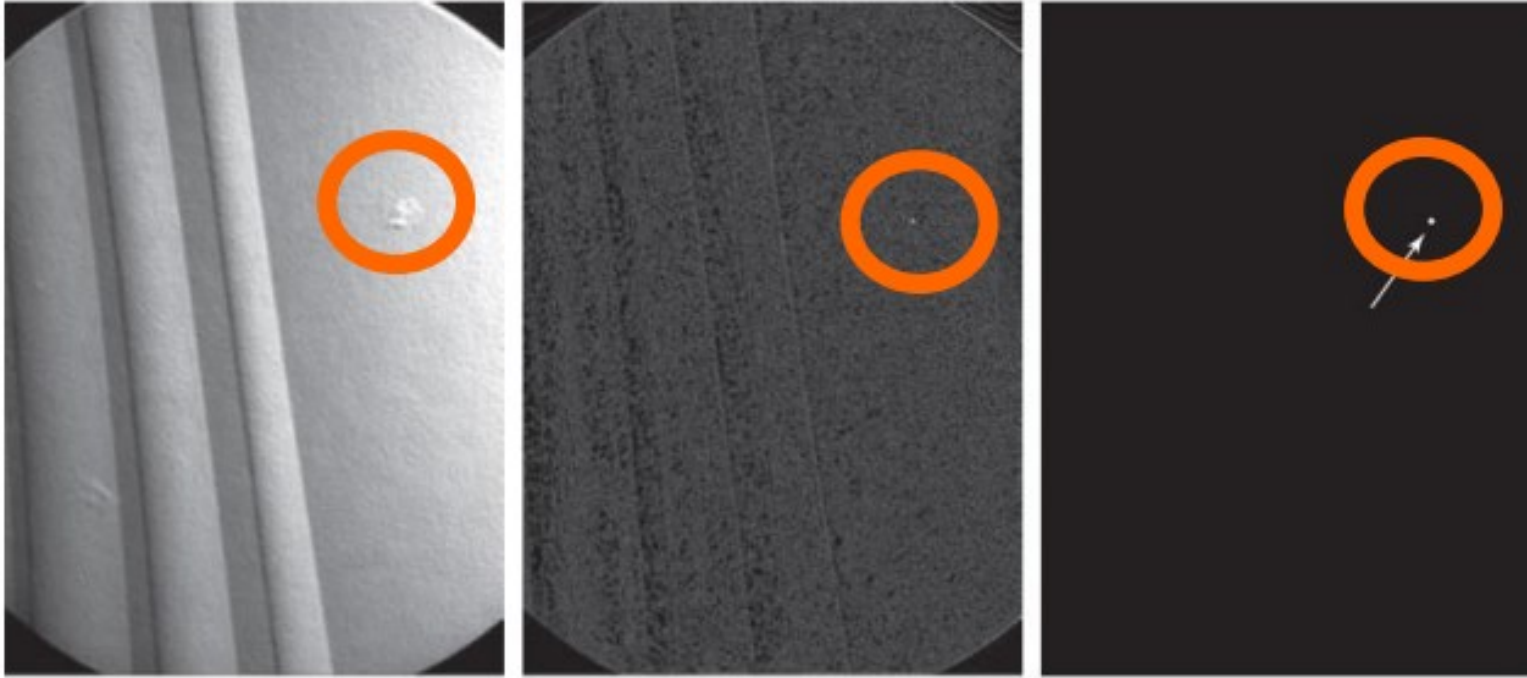
0	1	0	1	1	1	0	-1	0	-1	-1	-1
1	-4	1	1	-8	1	-1	4	-1	-1	8	-1
0	1	0	1	1	1	0	-1	0	-1	-1	-1

a b c d

FIGURE 3.45 (a) Laplacian kernel used to implement Eq. (3-53). (b) Kernel used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other Laplacian kernels.

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

Point detection

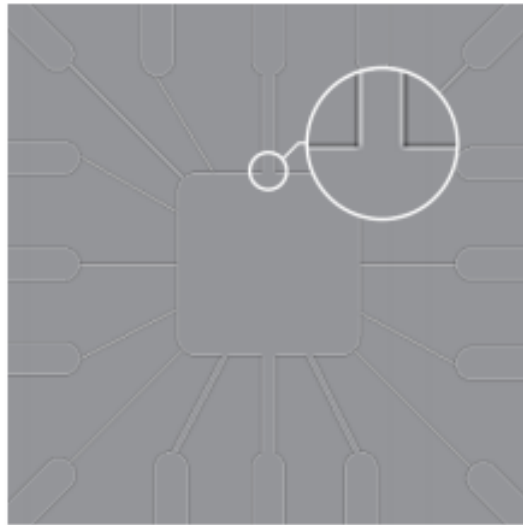


1	1	1
1	-8	1
1	1	1

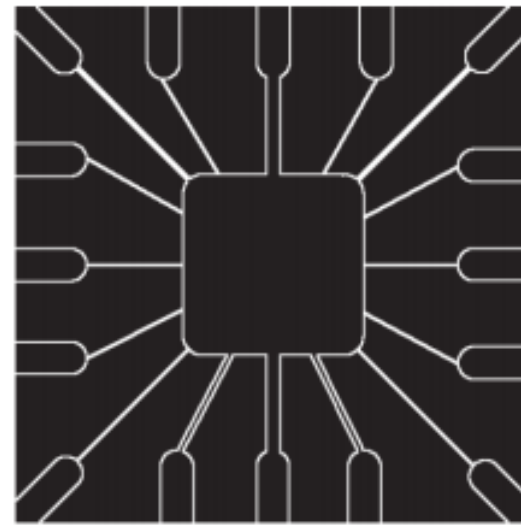
Line detection



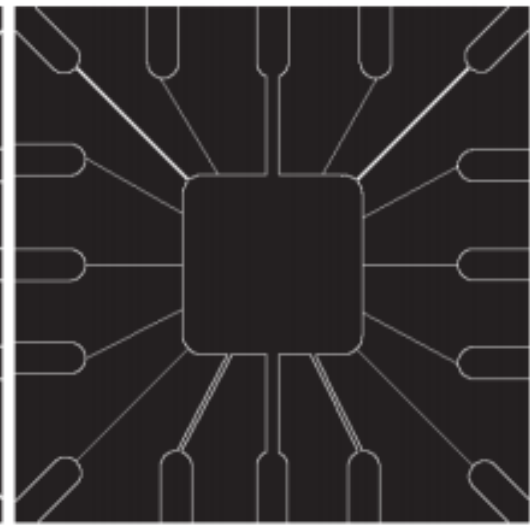
(a) Ảnh gốc



(b) Ảnh Laplacian;
phần phóng đại mô
tả hiệu ứng dòng kép
dương/âm



(c) Giá trị tuyệt
đối của Laplacian



(d) Giá trị dương
của Laplacian

Line detection

The filters below will extract lines that are 1 pixel thick and run in a specific direction

Using first and second derivatives for edge detection

Derivative-based edge detectors are always highly sensitive to noise

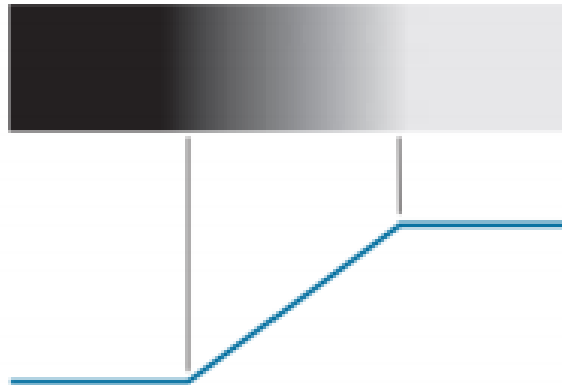
<table><tr><td>-1</td><td>-1</td><td>-1</td></tr><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>-1</td><td>-1</td><td>-1</td></tr></table>	-1	-1	-1	2	2	2	-1	-1	-1	<table><tr><td>2</td><td>-1</td><td>-1</td></tr><tr><td>-1</td><td>2</td><td>-1</td></tr><tr><td>-1</td><td>-1</td><td>2</td></tr></table>	2	-1	-1	-1	2	-1	-1	-1	2	<table><tr><td>-1</td><td>2</td><td>-1</td></tr><tr><td>-1</td><td>2</td><td>-1</td></tr><tr><td>-1</td><td>2</td><td>-1</td></tr></table>	-1	2	-1	-1	2	-1	-1	2	-1	<table><tr><td>-1</td><td>-1</td><td>2</td></tr><tr><td>-1</td><td>2</td><td>-1</td></tr><tr><td>2</td><td>-1</td><td>-1</td></tr></table>	-1	-1	2	-1	2	-1	2	-1	-1
-1	-1	-1																																					
2	2	2																																					
-1	-1	-1																																					
2	-1	-1																																					
-1	2	-1																																					
-1	-1	2																																					
-1	2	-1																																					
-1	2	-1																																					
-1	2	-1																																					
-1	-1	2																																					
-1	2	-1																																					
2	-1	-1																																					
Horizontal	+45°	Vertical	-45°																																				

Edge detection

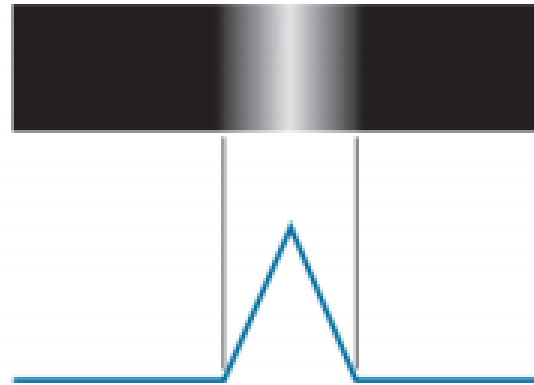
- An edge is a set of connected pixels located on the boundary between two regions.
- Some edge models:



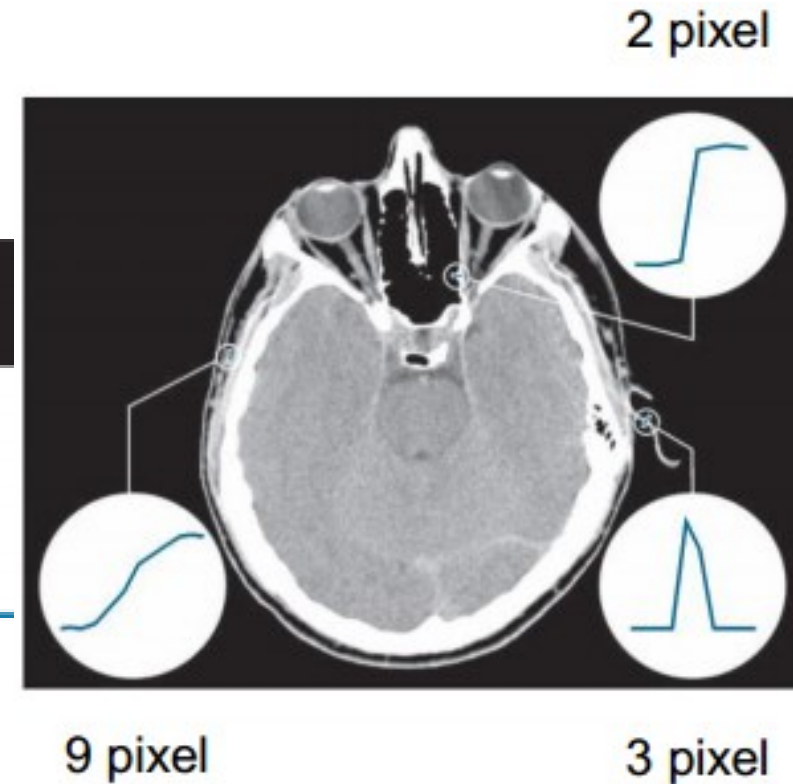
(a) Step edge



(b) Ramp edge



(c) roof edge



Edge detection steps

1. Smooth the image to reduce noise.
2. Detect edge points. As mentioned earlier, this is a local step to extract from an image all the points that are potential edge point candidates.
3. Localize edges. The goal of this step is to select from the candidates only the points that belong to the set of points forming an edge



Common edge detection filters

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	0	0	-1
0	1	1	0

Roberts

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

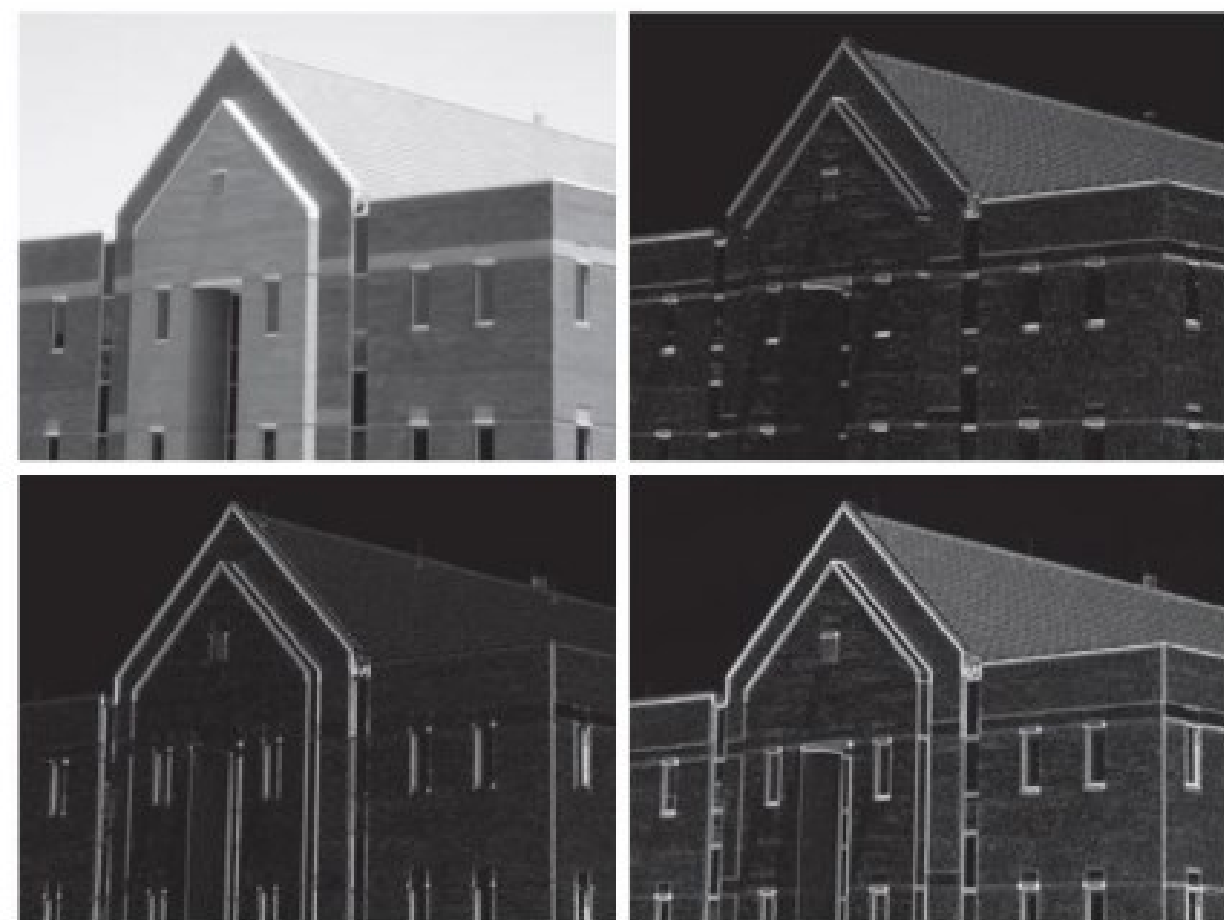
Prewitt

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

Sobel



Ảnh không làm mịn trước khi
phát hiện cạnh



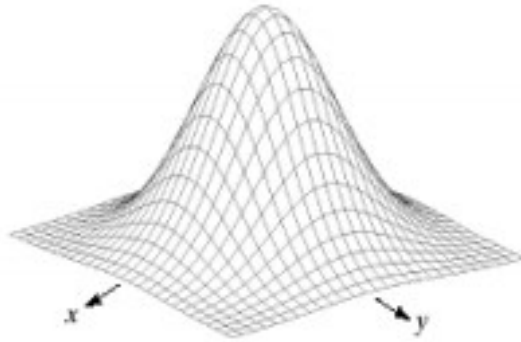
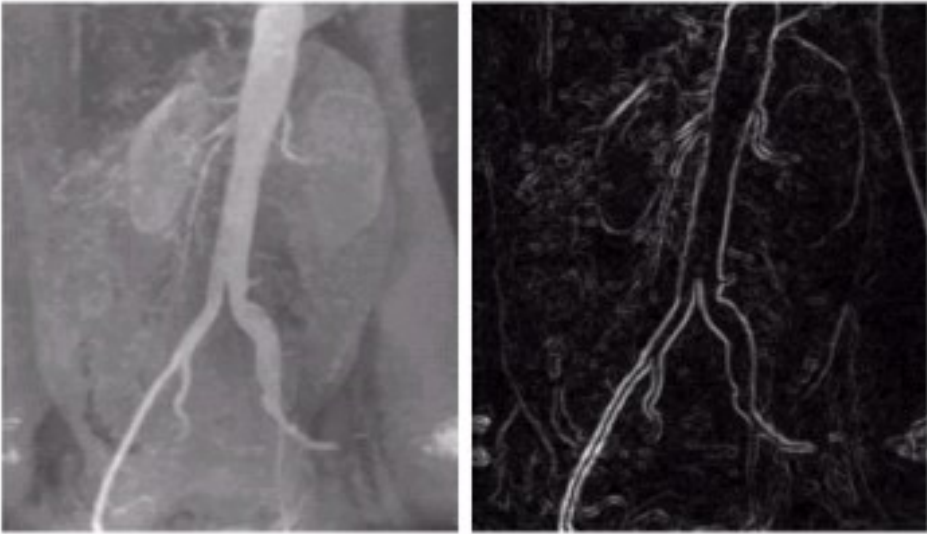
Ảnh có làm mịn trước khi
phát hiện cạnh

Laplacian for edge detection

- We have encountered the Laplacian filter based on the second derivative.
- Laplacian is generally not used alone because it is too sensitive to noise.
- Therefore, to use it for edge detection, the Laplacian is often combined with a Gaussian smoothing filter."

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

Laplacian for edge detection

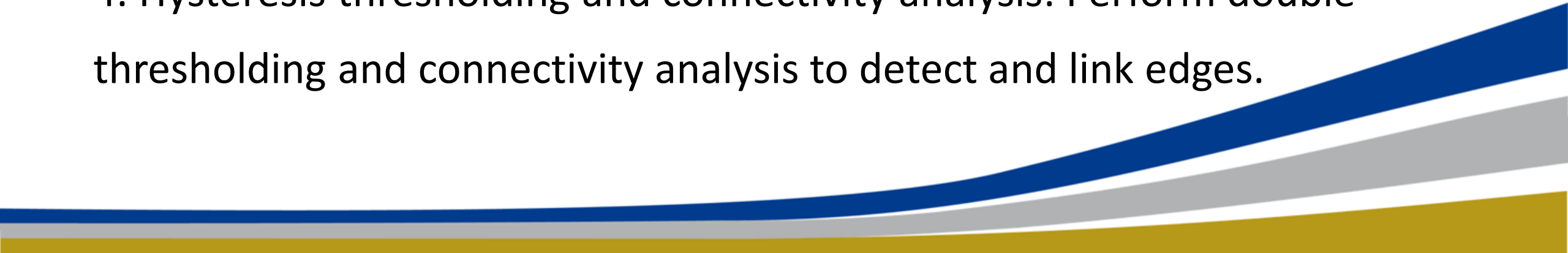


-1	-1	-1
-1	8	-1
-1	-1	-1



Canny for edge detection

- 1. Smoothing: Use a Gaussian filter to reduce noise in the image.
- 2. Gradient operator: Calculate the magnitude and direction of the gradient along the x and y axes using the Roberts or Sobel operator.
- 3. Non-maximum suppression: Eliminate points that do not reach the maximum value.
- 4. Hysteresis thresholding and connectivity analysis: Perform double thresholding and connectivity analysis to detect and link edges.



Step 1

- Since the Gaussian filter uses a simple filter window, it is used to reduce noise in the Canny algorithm.
- The larger the filter window size, the less sensitive the algorithm is to noise.
- For example, an approximation of the Gaussian filter

$$\mathbf{B} = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

Step 2

- The clarity of the detected edge is determined by calculating the gradient of the image.
- The gradient can be calculated using the Roberts operator or the Sobel operator.

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

G_x

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

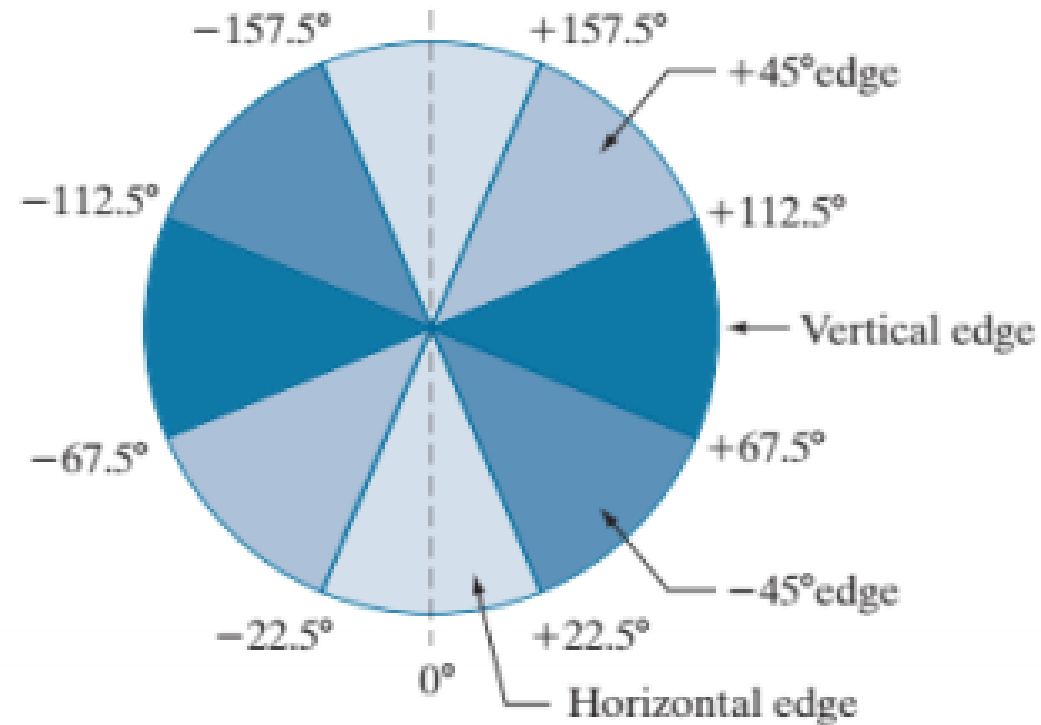
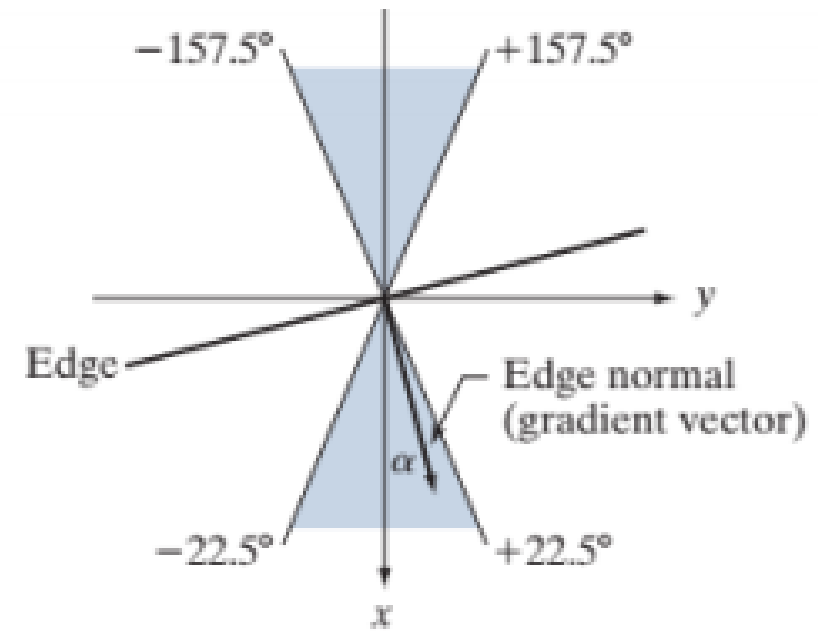
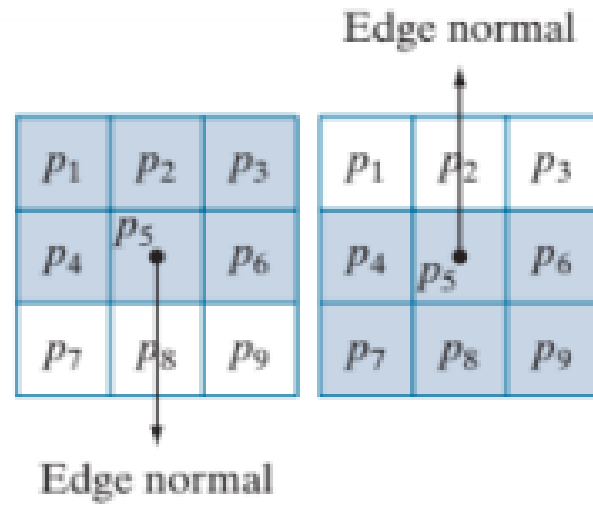
G_y

Orientation of Edge

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

$$|G| = |G_x| + |G_y|$$

Step 2



Step 3 Non-maximum suppression

Eliminate non-maximum values

- This is a technique aimed at thinning the edge line.
- A search is performed to determine whether the gradient magnitude is a local maximum along the gradient direction.
- At each pixel, set the pixel value to 0 if its magnitude is not greater than the magnitudes of its two neighbors in the gradient direction, and keep the pixel with the maximum magnitude



Step 4: Double thresholding

- **Use threshold T to reduce the number of spurious edges.**
- All pixels with values less than T are set to 0.
- Choosing the appropriate value of T is very difficult.
- If T is set too low, many spurious edges will appear.
- If T is set too high, some true edges will be lost.
→ **Solution:** Use two thresholds, T_1 (high) and T_2 (low)





(a) Original



(b) Smoothed



(c) Gradient magnitudes



(d) Edges after non-maximum suppression



(e) Double thresholding



(f) Edge tracking by hysteresis



(g) Final output

Phân vùng dựa vào ngưỡng

- Phân ngưỡng là gì?
 - Phân ngưỡng đơn giản
 - Phân ngưỡng thích ứng

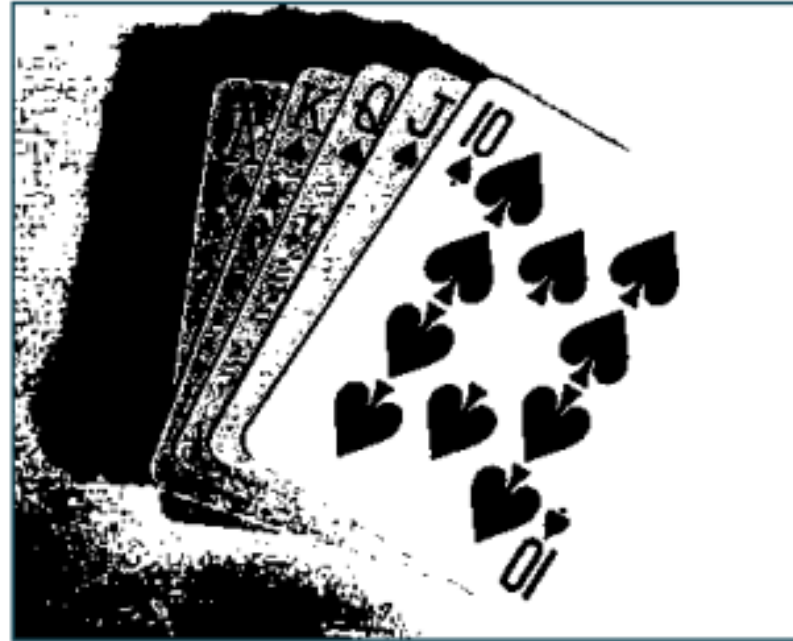


Phân ngưỡng cường độ

- Giả sử ảnh $f(x)$



Ngưỡng quá thấp



Ngưỡng quá cao



Phân loại ngưỡng

- Khi T là một hằng số có thể áp dụng cho toàn bộ hình ảnh, ngưỡng này được gọi là ngưỡng toàn cục (global thresholding).
 - Khi giá trị của T thay đổi trên một hình ảnh, chúng ta có ngưỡng thay đổi (variable thresholding). Các thuật ngữ phân ngưỡng cục bộ hoặc địa phương đôi khi được sử dụng để biểu thị ngưỡng thay đổi trong đó giá trị của T tại bất kỳ điểm (x,y) nào trong một hình ảnh phụ thuộc vào thuộc tính của vùng lân cận của (x,y) ví dụ: cường độ trung bình của các pixel trong vùng lân cận).
 - Nếu T phụ thuộc vào chính các tọa độ không gian (x,y) thì ngưỡng thay đổi thường được gọi là ngưỡng động hoặc ngưỡng thích ứng.



Các yếu tố ảnh hưởng tới ngưỡng

- 1) Khoảng cách giữa các đỉnh (các đỉnh càng xa nhau thì càng có cơ hội tốt hơn để tách các chế độ);
(2) Nhiễu trong ảnh;
(3) Kích thước tương đối của các đối tượng và nền;
(4) Tính đồng nhất của nguồn chiếu sáng;
(5) Tính đồng nhất của thuộc tính phản xạ của ảnh.



Sử dụng ngưỡng toàn cục

- - Khi phân bố cường độ của các pixel đối tượng và pixel nền đủ khác biệt, có thể sử dụng một ngưỡng duy nhất (toàn cục) áp dụng cho toàn bộ hình ảnh.
 - Trong hầu hết các ứng dụng, khi sử dụng ngưỡng toàn cục thì vẫn cần một thuật toán có khả năng ước tính giá trị ngưỡng cho mỗi hình ảnh.



Thuật toán đẳng liệu (lấp) - Ridler and Calvard

Ngưỡng toàn cục cơ bản T được tính như sau:

1. Chọn giá trị ước lượng ban đầu cho $T=t_0$ (thông thường là mức xám trung bình trong bức ảnh).
2. Phân vùng bức ảnh sử dụng t_k để tạo ra 2 nhóm pixel: G_1 gồm các pixel với mức xám $\leq t_k$ và G_2 gồm các pixel có mức xám $> t_k$
3. Tính mức xám trung bình của các pixel trong G_1 là $\mu_1(t_k)$ và trong G_2 là $\mu_2(t_k)$

Thuật toán đẳng liêu (lặp)

4. Tính giá trị ngưỡng mới:

$$t_{k+1} = \frac{\mu_1(t_k) + \mu_2(t_k)}{2} = \frac{1}{2} \left(\frac{\sum_{z=0}^{t_k} z \cdot p(z)}{\sum_{z=0}^{t_k} p(z)} + \frac{\sum_{z=t_k+1}^{L-1} z \cdot p(z)}{\sum_{z=t_k+1}^{L-1} p(z)} \right)$$

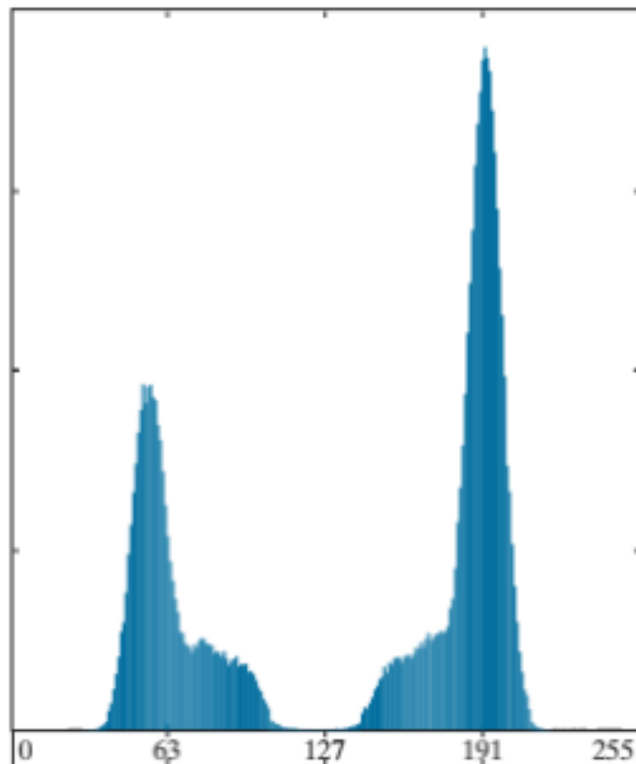
5. Lặp lại các bước 2 – 4 cho đến khi sự khác nhau giữa các T trong các vòng lặp liên tiếp nhỏ hơn một giá trị giới hạn định trước ΔT hay $|t_{k+1} - t_k| < \Delta T$

Thuật toán này làm việc hiệu quả trong việc tìm giá trị ngưỡng T khi ảnh có lược đồ xám thích hợp.

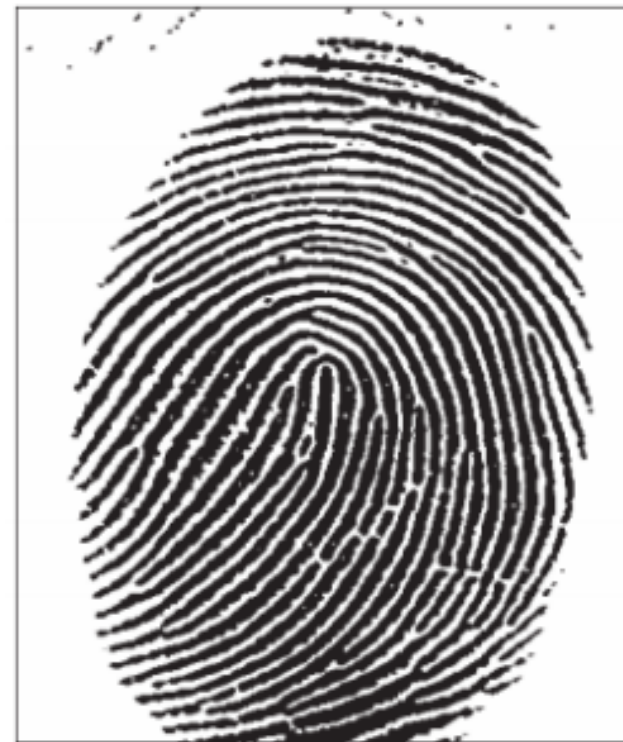
Thuật toán đẳng liệu (lập)



Vân tay bị nhiễu



Biểu đồ xám



ra các
hạn

Sử dụng ngưỡng từ
thuật toán lập ($T=125,4$ *trị*
sau 3 lần lập, $\Delta T=0$)

Ví dụ

Tìm ngưỡng toàn cục cho ảnh có phân bố xác suất tương ứng với mức xám như sau:

r	0	1	2	3	4	5	6	7	8	9
$p(r)$	0,09	0,2	0,1	0,06	0,06	0,08	0,35	0,04	0,01	0,01



Bài tập

r_k	0	1	2	3	4	5	6	7	8	9
$p(r_k)$	$\frac{9}{100}$	$\frac{3}{20}$	$\frac{11}{100}$	$\frac{11}{150}$	$\frac{11}{150}$	$\frac{3}{25}$	$\frac{3}{20}$	$\frac{17}{150}$	$\frac{23}{300}$	$\frac{13}{300}$

Bài tập

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Phân ngưỡng toàn cục tối ưu sử dụng phương pháp OTSU

- Phân ngưỡng là bài toán giảm thiểu lỗi trung bình phát sinh khi gán pixel vào một trong các lớp.
- Phương pháp OTSU tối ưu theo nghĩa là nó tối đa hóa phương sai giữa các lớp và các phép tính được thực hiện hoàn toàn trên biểu đồ của ảnh.
- Ý tưởng cơ bản là các lớp được phân ngưỡng thích hợp phải phân biệt tương ứng với các giá trị cường độ của các pixel của chúng và ngược lại, nghĩa là **ngưỡng mang lại sự tách biệt tốt nhất giữa các lớp về giá trị cường độ sẽ là ngưỡng tốt nhất (tối ưu).**



Phân ngưỡng toàn cục tối ưu sử dụng phương pháp OTSU


- 1. Tính toán biểu đồ chuẩn hóa của hình ảnh đầu vào. Biểu thị các thành phần của biểu đồ bởi $p_i, i = 0, 1, \dots, L - 1$
- 2. Tính tổng tích lũy $P_1(k), k = 0, 1, \dots, L - 1$, sử dụng phương trình

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

- 3. Tính giá trị trung bình tích lũy, $m(k), k = 0, 1, \dots, L - 1$ sử dụng:

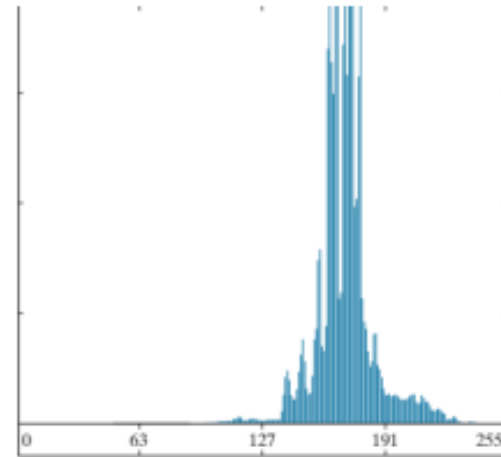
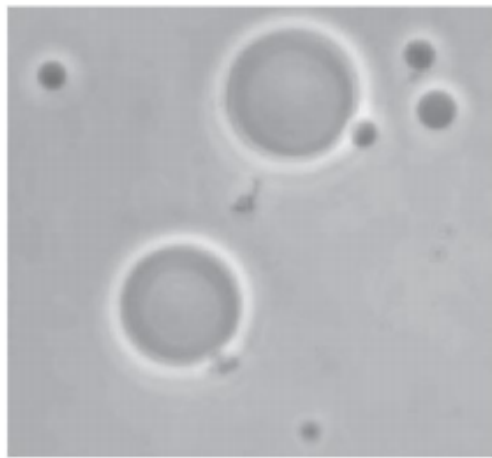
$$m(k) = \sum_{i=0}^k i p_i$$

- 4. Tính giá trị trung bình toàn cục, $m(G)$, sử dụng:

$$m(G) = \sum_{i=0}^{L-1} i p_i$$


Phân ngưỡng toàn cục tối ưu sử dụng phương pháp OTSU

5.
ph



6. L
khé
vớ

7.
tính

Ảnh gốc

Biểu đồ xám

Kết quả phân đoạn
sử dụng ngưỡng
bằng thuật toán lậ

Kết quả phân đoạn
sử dụng ngưỡng
bằng thuật toán Otsu

Ví dụ

I =

i	p_i	$P_i(k)$	$m(k)$	$m(G)$	$\sigma_B^2(k)$
0	15/30	0.5000	0.0000	1.1667	1.36
1	5/30	0.6667	0.1667	1.1667	1.68
2	4/30	0.8000	0.4333	1.1667	1.56
3	3/30	0.9000	0.7333	1.1667	1.11
4	2/30	0.9667	1.0000	1.1667	0.51
5	1/30	1.0000	1.1667	1.1667	0.00

■ Như vậy ngưỡng $T = 1$ với $\sigma_B^2(k) = 1.68$

Trong hình trước, các pixel đối tượng nằm ở bên trái của đỉnh nền (đỉnh nền nằm ở mức sáng $\max(p) = 183$).

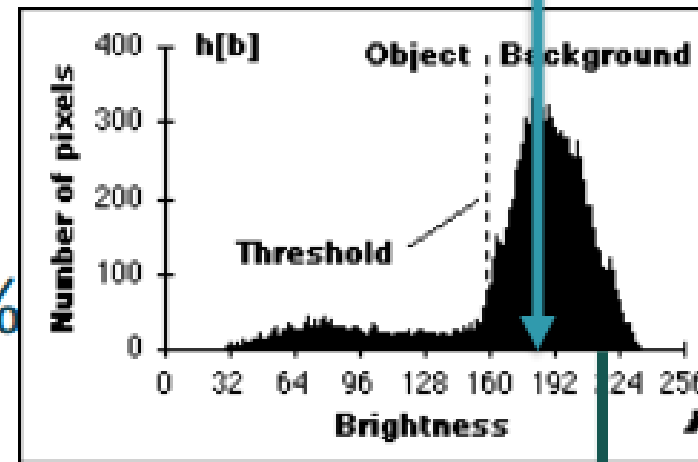
Do đó việc tìm kiếm thực hiện ở bên phải của đỉnh.

Ví dụ chọn $p=95\%$, điểm bên phải tương ứng với 95% mức sáng cực đại có độ sáng 216 ($p\%$)

Do giả định về tính đối xứng, ta tính được ngưỡng nằm ở bên trái của giá trị cực đại theo công thức:

$$\theta = \max p - (p\% - \max p)$$

Ngưỡng: $\theta = 183 - (216 - 183) = 150$



PHÂN ĐOẠN BẰNG PHÁT TRIỂN VÙNG VÀ PHÂN ĐOẠN BẰNG TÁCH & HỢP VÙNG



Phát triển vùng

- (1) Chọn pixel hạt giống
- (2) Kiểm tra các pixel lân cận và thêm vào vùng nếu nó tương tự pixel hạt giống.
- (3) Lặp lại bước 2 cho mỗi pixel mới thêm vào; dừng lại khi không có pixel mới nào được thêm vào.



Chia tách và hợp vùng

