COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Lecture 9 **(4.7.08)**

Image Segmentation

Shahram Ebadollahi

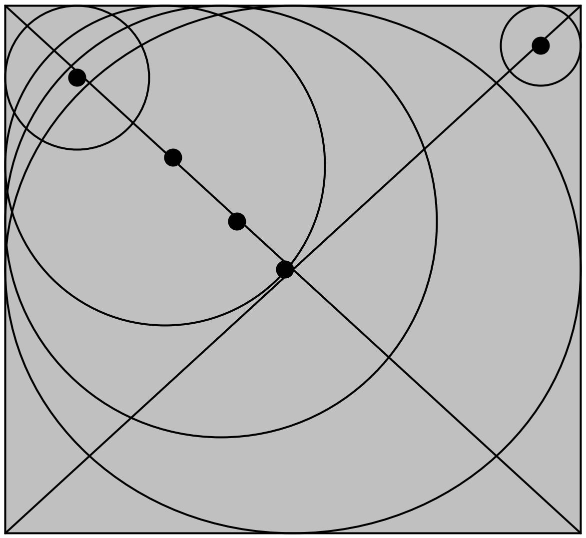
Lecture Outline

- Skeletonization
  - Extension to Gray-level images -- GSAT
- Image Segmentation – Introduction
- Thresholding
- Edge Segmentation and Linking
  - Hough Transform
- Region-based Approach
- Using Morphology for Segmentation
  - Watershed Algorithm

Skeletonization (Medial Axis Transform)

B is a “Maximal Disc” in set X if there are no other discs included in X and containing B

Notion of “Maximal Disc”



Skeleton is the loci of the centers of all “maximal discs”

$$S(X) = \bigcup_{k\geq 0}\left\{\varepsilon_{kB}(X) \setminus \gamma_B[\varepsilon_{kB}(X)]\right\}$$

Skeletonization

$$S(X) = \bigcup_{k=0}^{K} S_k(X)$$

$$S_k(X) = \varepsilon_{kB}(X) - \gamma_B(\varepsilon_{kB}(X))$$

$$\varepsilon_{kB}(X) = \varepsilon_B(\varepsilon_B(\cdots(\varepsilon_B(X)))$$

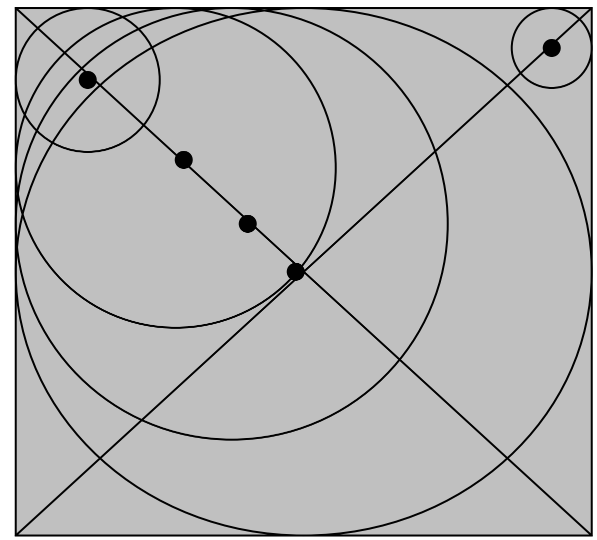
$$K = \max\{k \mid \varepsilon_{kB}(X) \neq \phi\}$$

Reconstruction

$$X = \bigcup_{k=0}^{K} \delta_{kB}(S_k(X))$$

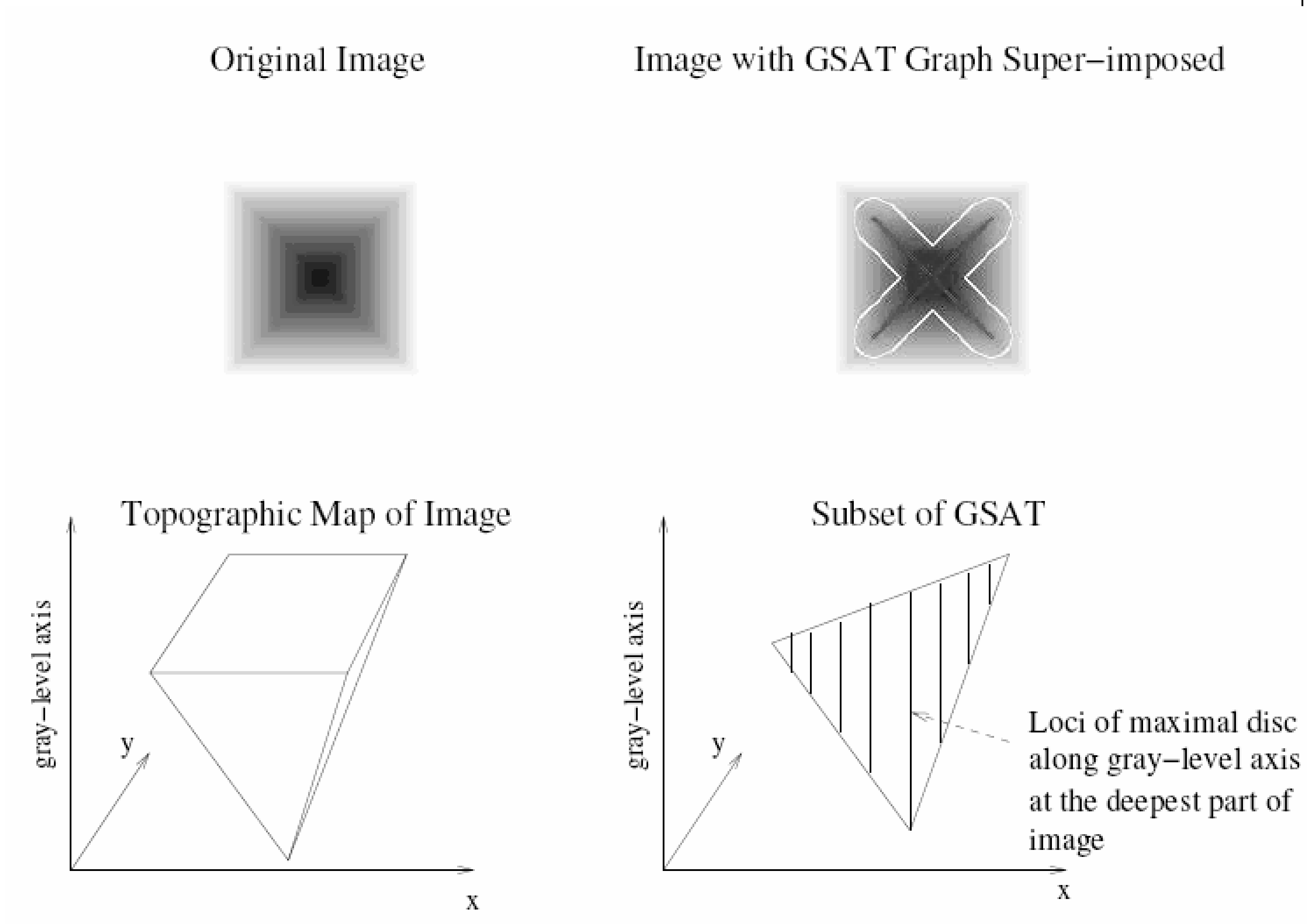
$$\delta_{kB}(X) = \delta_B(\delta_B(\cdots(\delta_B(S_k(X))))$$

Notion of "Maximal Disc"



Skeleton is the loci of the centers of all "maximal discs"

Gray-level SAT
(Skeletonization for Gray-level Images)



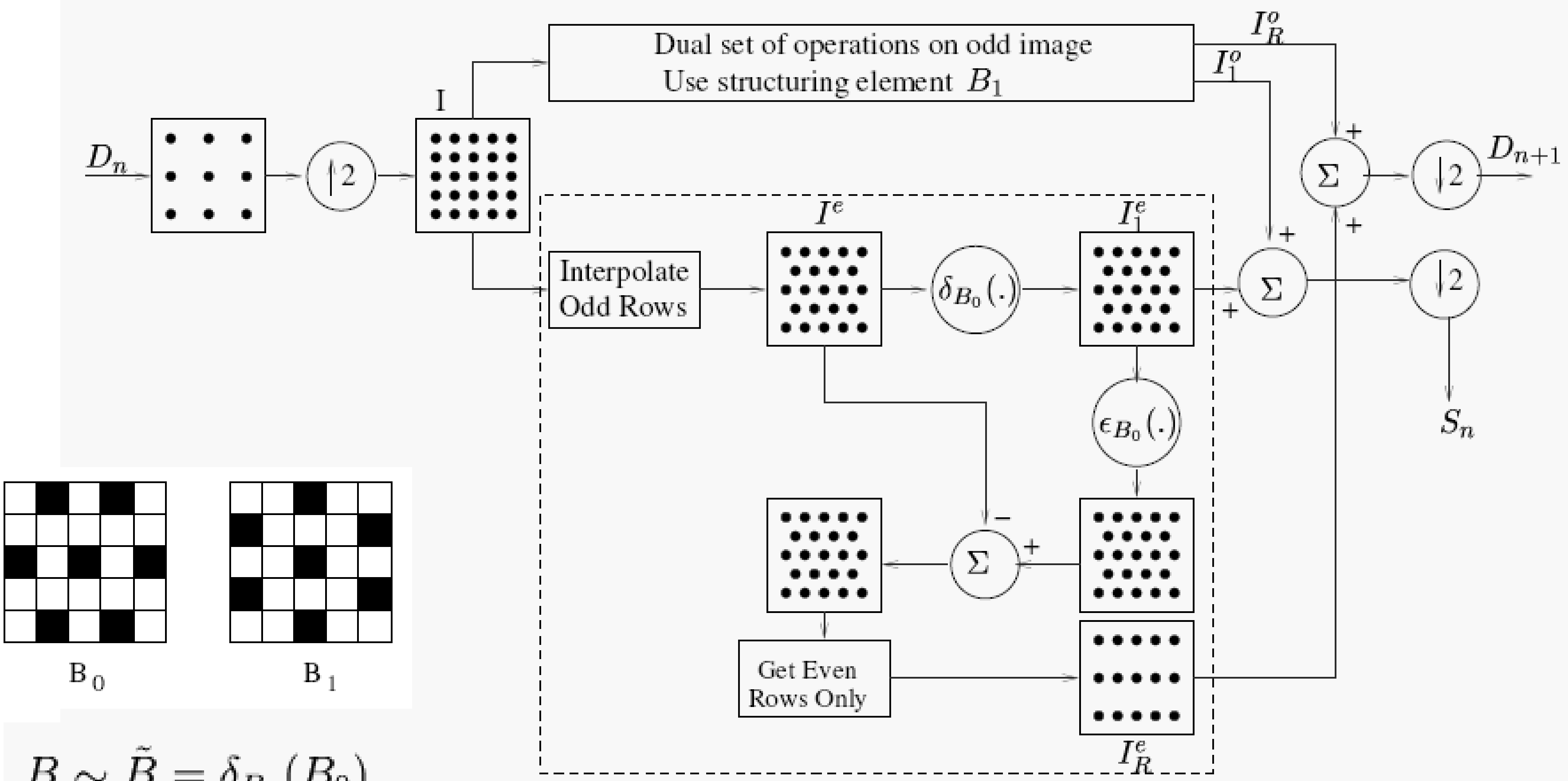
GSAT

- GSAT is the extension of skeletonization to gray-level images
- GSAT is the locus of the centers of maximal discs whose plane is perpendicular to the gray-level axis and which fit in the region above or below the topographic map of image
- Symmetry surface for gray-level == skeleton for bi-level
- Represent the collection of symmetry surfaces as a graph → GSAT graph
- nodes:
  - g_min: gray-level at bottom of surface path
  - delta_g: difference in gray-level between top and bottom
  - p_avg: average location of maximal discs on the path
  - n_avg: average size of maximal discs on the path

$$D_n(f) = \delta_{nB}(f)$$

$$S_n(f) = \phi_B(D_n(f)) - D_n(f) = \epsilon_B(D_{n+1}(f)) - D_n(f)$$

GSAT - Implementation



$$B \sim \tilde{B} = \delta_{B_1}(B_0)$$

$D_n(f)$

$S_n(f)$

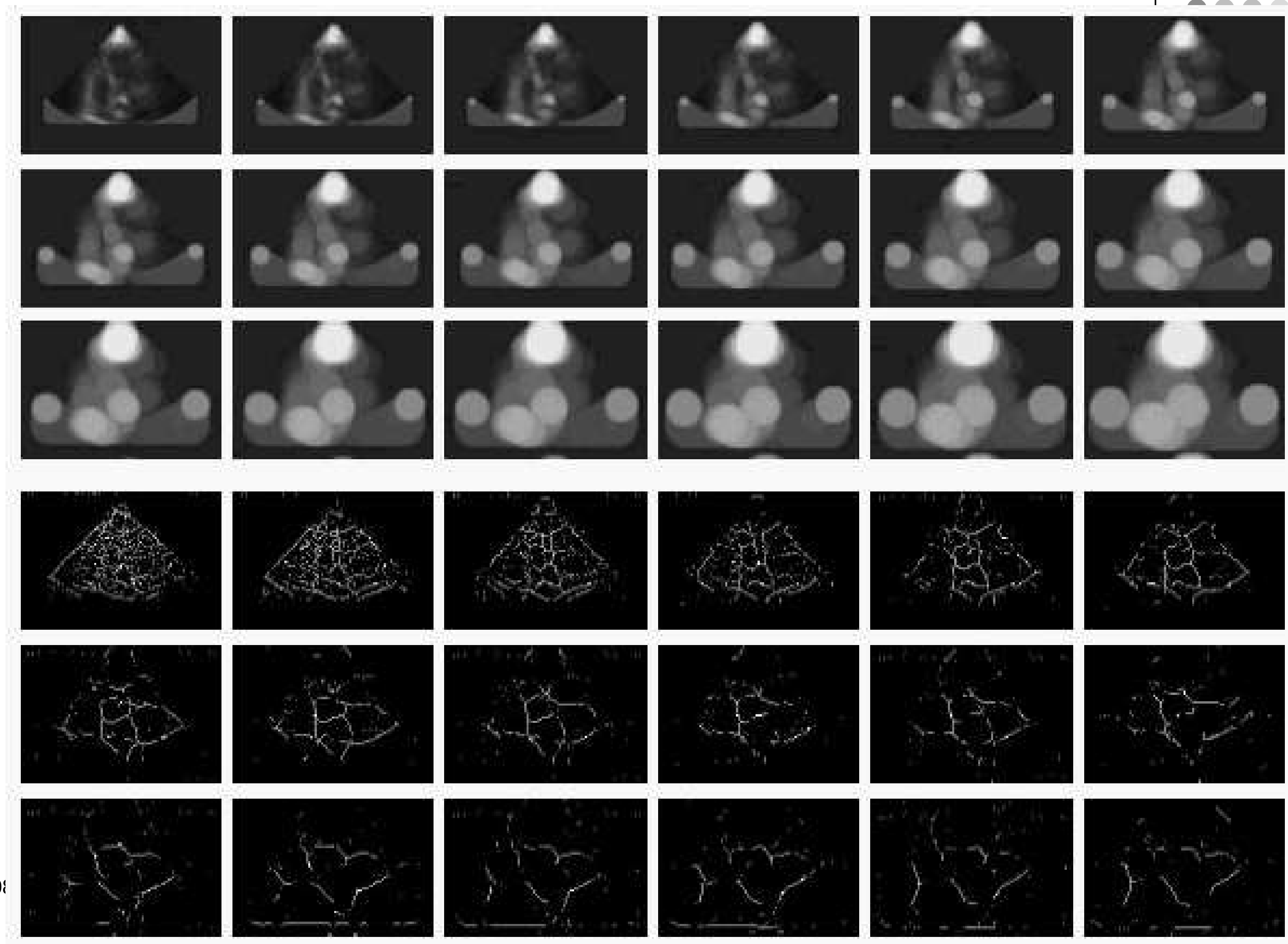
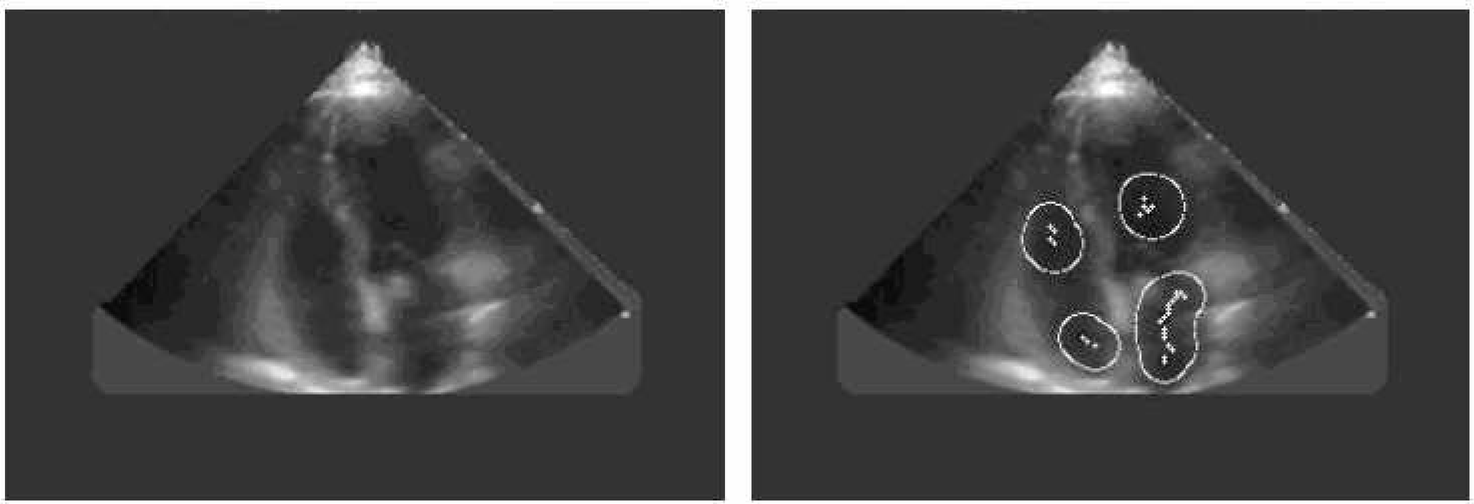
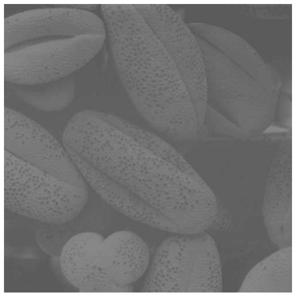


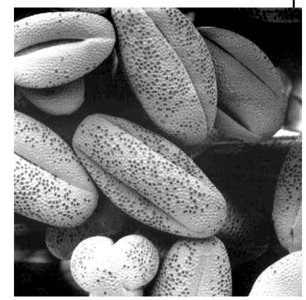
Image Segmentation by Region Growing on GSAT graph



Digital Image Processing



Low level



Mid level

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_P \end{bmatrix}$$

High level

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Image Segmentation - Intro

<u>Goal</u> decompose image into regions R_k such that:

$$f = \bigcup_{k=1}^{K} R_k$$

$$R_i \bigcap R_j = \phi \quad i \neq j$$

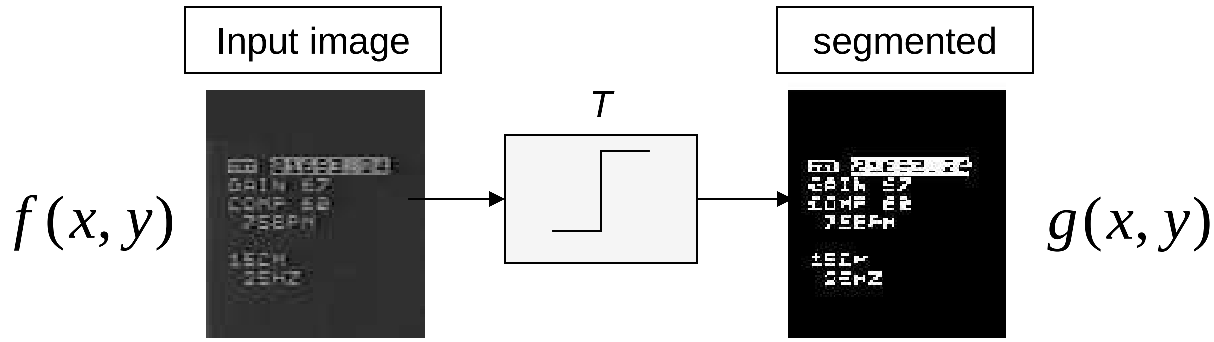
$$H(R_k) = true \quad \forall k \in \{1, \cdots, K\}$$

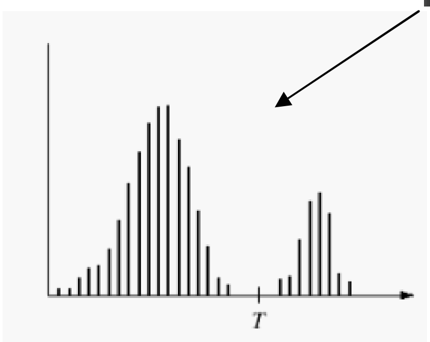
$$H(R_i \bigcup R_j) = false \quad i \neq j$$

There are various approaches:

- edge-based vs. region-based
- global vs. local
- feature – texture, motion, color

Thresholding



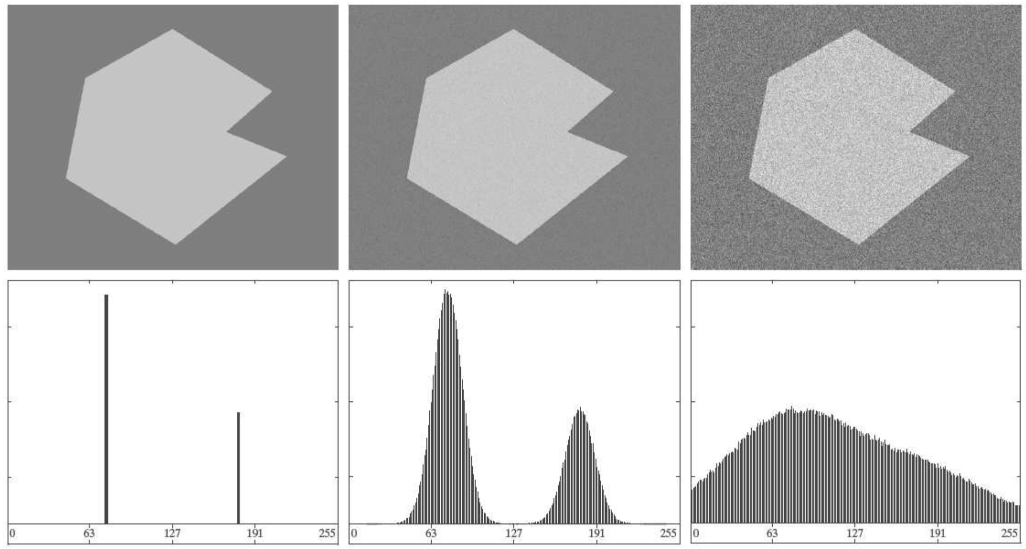


$$g(x,y)=\begin{cases}1 & if \quad f(x,y)>T\\ 0 & if \quad f(x,y)\le T\end{cases}$$

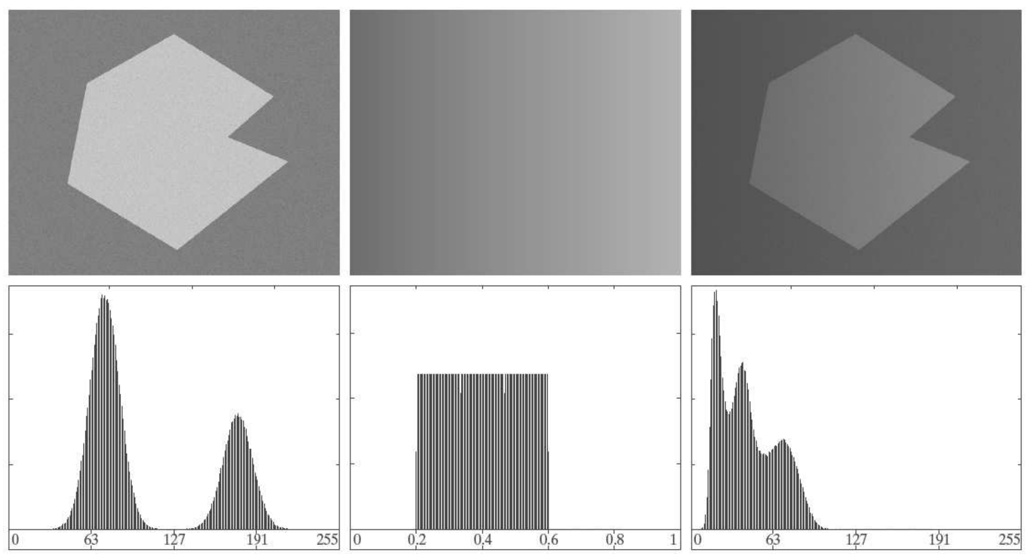
- Global Thresholding: T is constant everywhere in the image
- Variable Thresholding: T varies
  - Local Thresholding: T changes based on the neighborhood properties
  - Adaptive Thresholding: T changes based on the coordinates

Thresholding - pitfalls

Effect of Noise

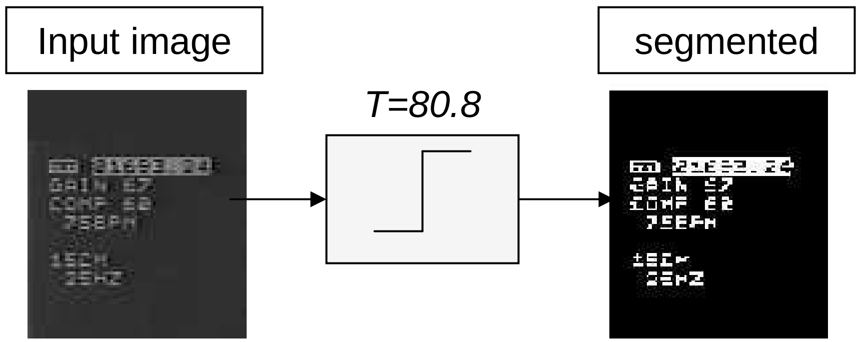


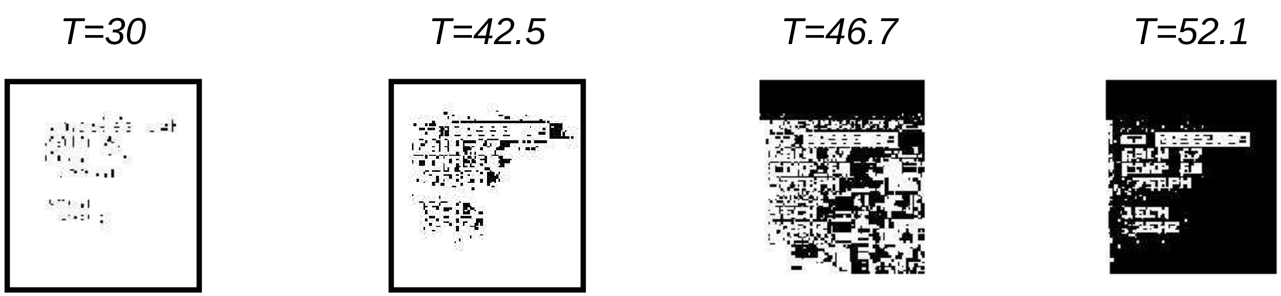
Effect of Illumination



Global Thresholding

1. $T=T_0$
2. Segment using T
3. Get average gray-level for region G_1 (f > T) and region G_2 (f <= T)
4. T_new = average of average of gray-levels
5. Repeat until convergence



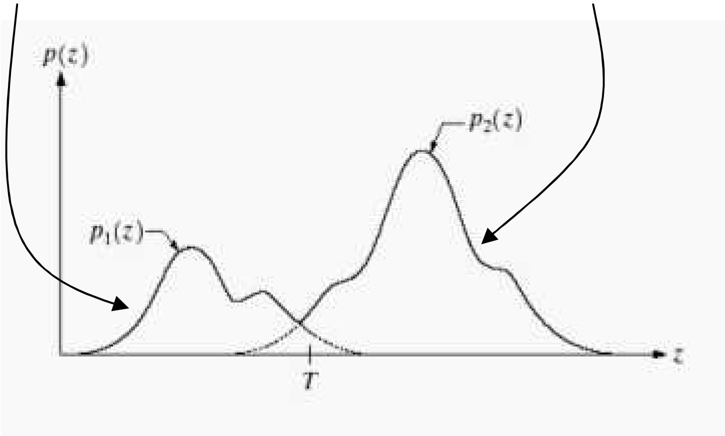


Optimal Threshold

- Pose problem as:

minimizing error of assigning pixels in image to two or more groups

$\Pr(f(x,y)=z \mid (x,y)\in FG)$ $\Pr(f(x,y)=z \mid (x,y)\in BG)$

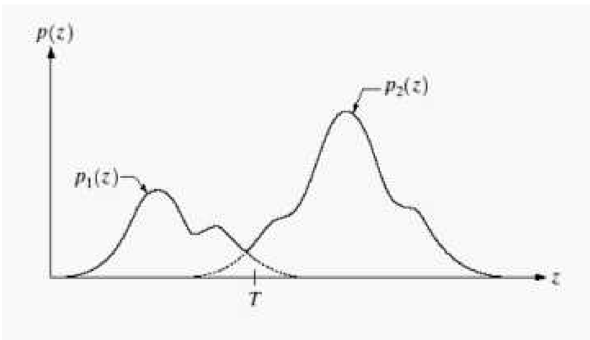


$$\Pr(f(x,y)=z) = \Pr([f(x,y)=z,(x,y)\in BG] \vee [f(x,y)=z,(x,y)\in FG])$$

$$\Pr(f(x,y)=z) = \Pr((x,y)\in FG)\Pr(f(x,y)=z \mid (x,y)\in FG) + \Pr((x,y)\in BG)\Pr(f(x,y)=z \mid (x,y)\in BG)$$

$$\Pr(f(x,y)=z) = P_{FG}\,p_1(z) + P_{BG}\,p_2(z) \qquad \text{where,} \qquad P_{FG}+P_{BG}=1$$

Optimal Threshold (cont.)



$$E_1(T) = \int_{-\infty}^{T} p_2(z)dz$$

Probability of classifying a BG pixel as FG by mistake

$$E_2(T) = \int_{T}^{\infty} p_1(z)dz$$

Probability of classifying a FG pixel as BG by mistake

Overall probability of error: $$E(T) = P_{BG}E_1(T) + P_{FG}E_2(T)$$

$$T_{opt} = \arg\min_T E(T)$$

Optimal threshold minimizes the probability of misclassification

Gaussian densities case:

$$p(z) = \frac{P_{FG}}{\sqrt{2\pi}\sigma_{FG}} e^{-\frac{(z-\mu_{FG})^2}{2\sigma_{FG}^2}} + \frac{P_{BG}}{\sqrt{2\pi}\sigma_{BG}} e^{-\frac{(z-\mu_{BG})^2}{2\sigma_{BG}^2}} \Rightarrow T_{opt} = \frac{\mu_{FG}+\mu_{BG}}{2} + \frac{\sigma^2}{\mu_{FG}-\mu_{BG}} \ln\left(\frac{P_{FG}}{P_{BG}}\right)$$

$$\sigma_{FG}^2 = \sigma_{BG}^2 = \sigma^2$$

Optimal Threshold – Otsu's method

- Otsu defines the measure of "goodness" of a threshold based on how well it can separate two (or more) classes (regions)

Measure of goodness

Goal

$$k^* = \underset{k\in\{0,\cdots,L-1\}}{\arg\max} \frac{\sigma_B^2}{\sigma_T^2}$$

Between-class Variance

$$\sigma_B^2 = \omega_1(\mu_1-\mu_T)^2 + \omega_2(\mu_2-\mu_T)^2 = \omega_1\omega_2(\mu_2-\mu_1)^2$$

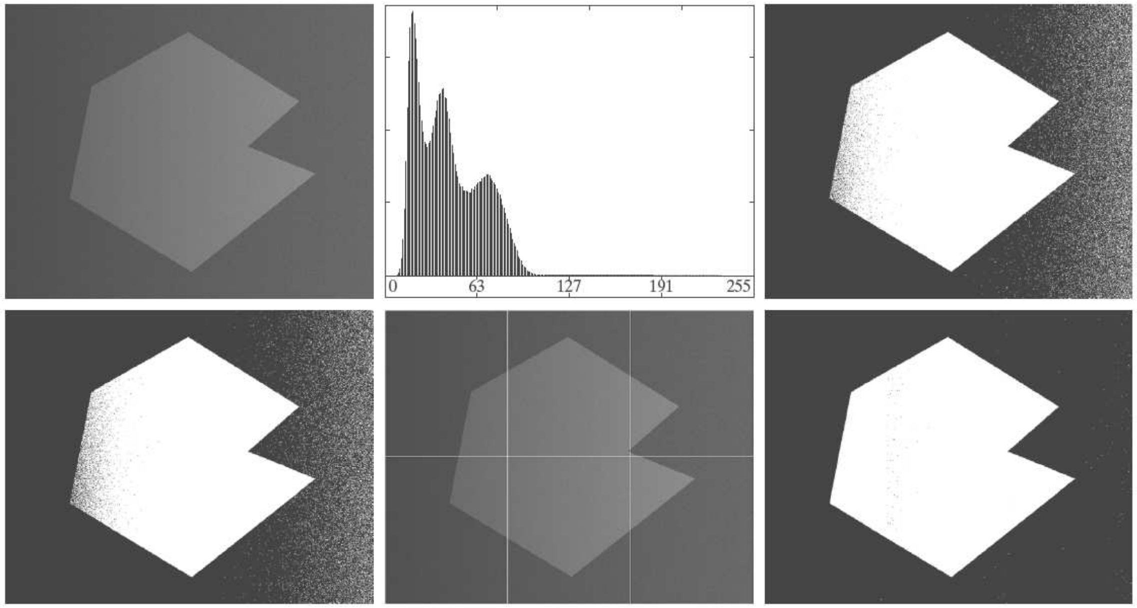
Total Variance

$$\sigma_T^2 = \sum_{i=0}^{L-1} p_i(i-\mu_T)^2$$

Thresholding in MATLAB using Otsu's method for determining the threshold:

```
I=imread('coins.png');
level=graythresh(I);
BW=im2bw(I,level);
imshow(BW)
```

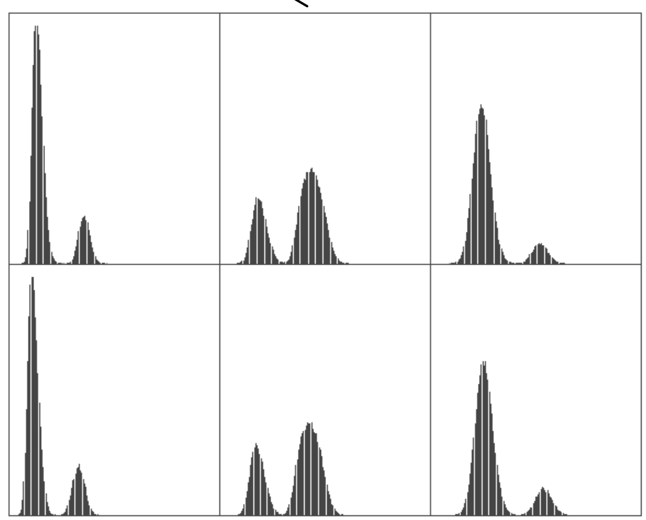
Adaptive Thresholding – Image Partitioning



Global threshold obtained using iterative algorithm

Global threshold obtained using Otsu's method

Under what condition this method fails?



Bimodal histograms

Adaptive Thresholding – Moving Average

$$m(k+1) = \frac{1}{n}\sum_{i=k+2-n}^{k+1} z_i = m(k) + \frac{1}{n}\left(z_{k+1} - z_{k-n}\right)$$

original | Otsu | Moving average

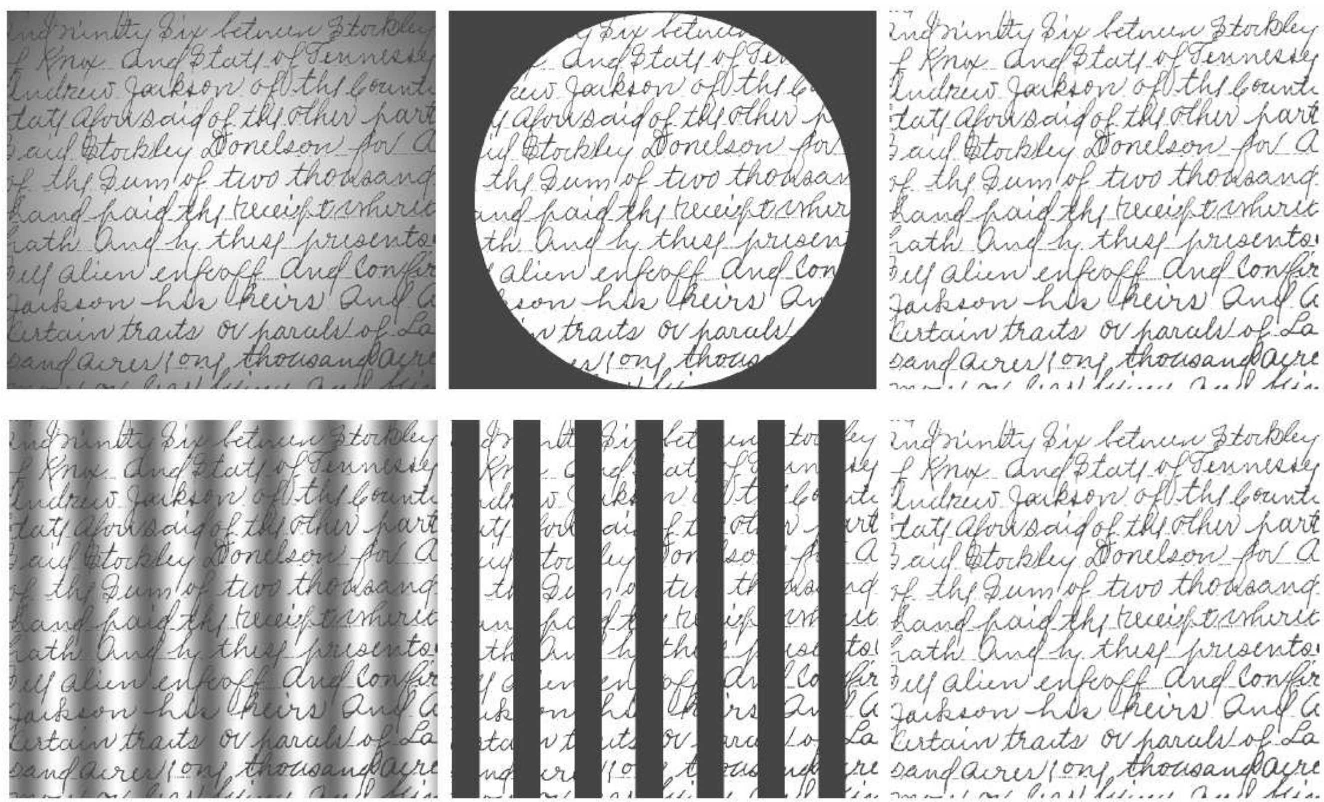
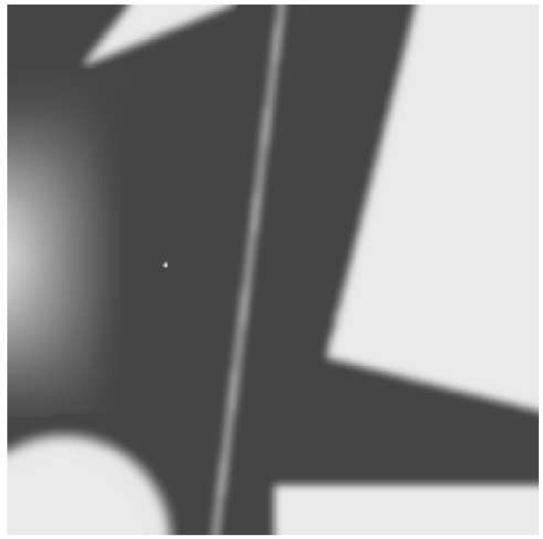
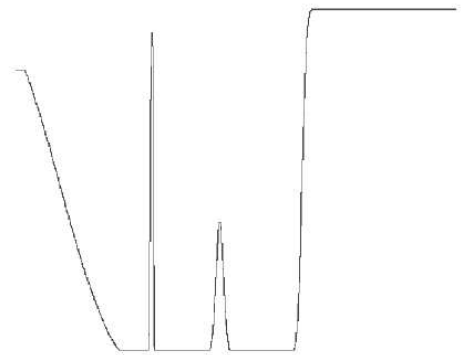


Image Derivatives

- 1^{st} order derivative produces "thick" edges
- 2^{nd} order derivative has stronger response to fine edges
- 2^{nd} order derivative produces double edge response at ramp and step transitions in intensity
- 2^{nd} order derivative's sign can be used to find out if going from dark to light or vice versa





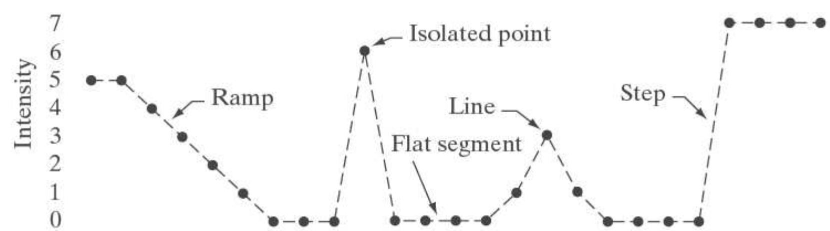


Image strip	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7	•	•
First derivative		−1	−1	−1	−1	−1	0	0	6	−6	0	0	0	1	2	−2	−1	0	0	0	7	0	0	0		
Second derivative		−1	0	0	0	0	1	0	6	−12	6	0	0	1	1	−4	1	1	0	0	7	−7	0	0		

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

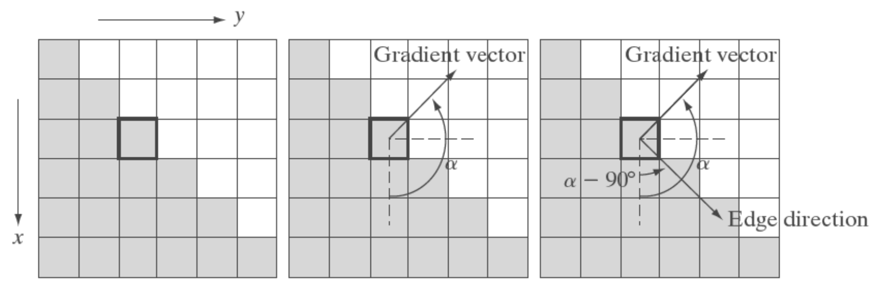
$$f''(x) = f(x+1) + f(x-1) - 2f(x)$$

Edge detection – gradient operator

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

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

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



<u>Gradient operator for discrete images</u>

−1
1

−1	1

$$g_x = f(x+1, y) - f(x,y)$$
$$g_y = f(x, y+1) - f(x,y)$$

2-D gradient operators

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$g_x = (z_9 - z_5)$$
$$g_y = (z_8 - z_6)$$

−1	0	0	−1
0	1	1	0

Roberts

$$g_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$
$$g_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

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

Prewitt

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

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

Sobel

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

Prewitt

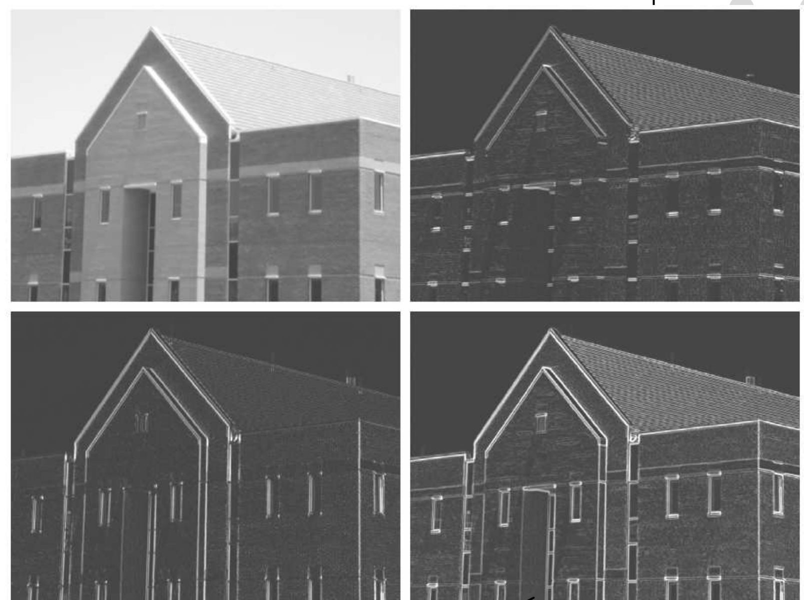
0	1	2	−2	−1	0
−1	0	1	−1	0	1
−2	−1	0	0	1	2

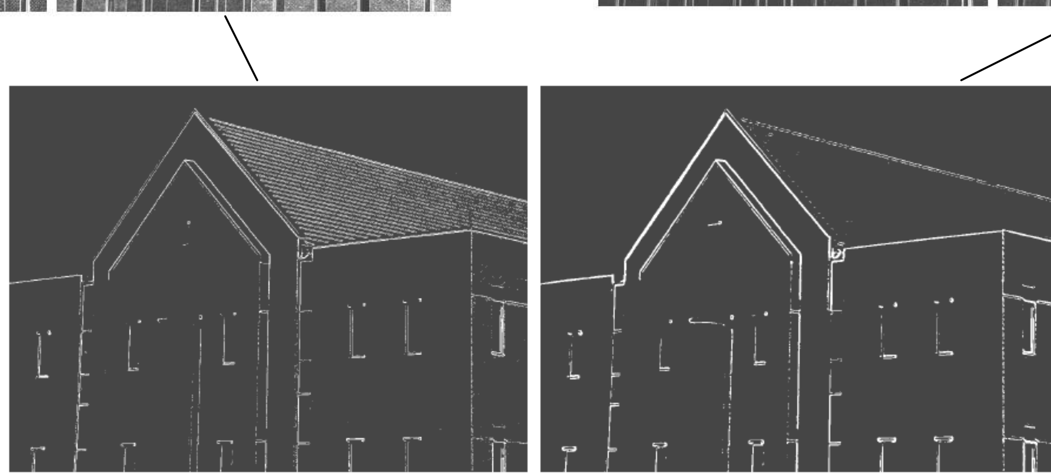
Sobel

<u>for diagonal edges</u>

Edge detection

Averaged prior to edge detection





After thresholding

Edge detection: Marr-Hildreth method

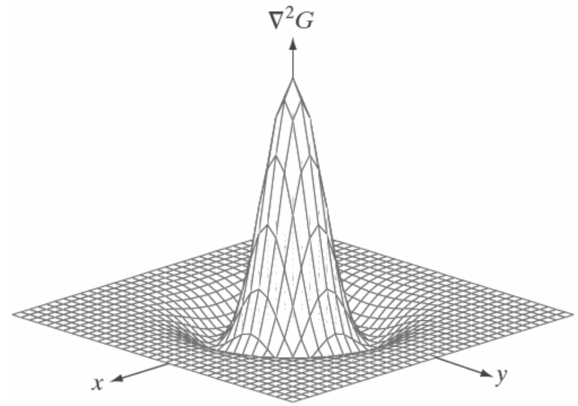
* Edge detection operator should be “tunable” to detect edges at different “scales”

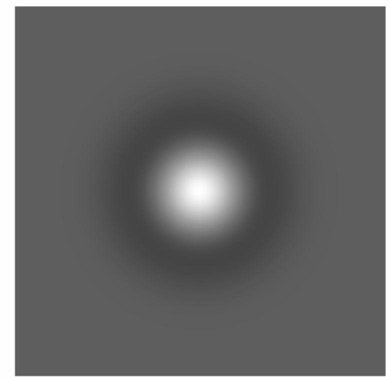
$\nabla^2 G$ LoG

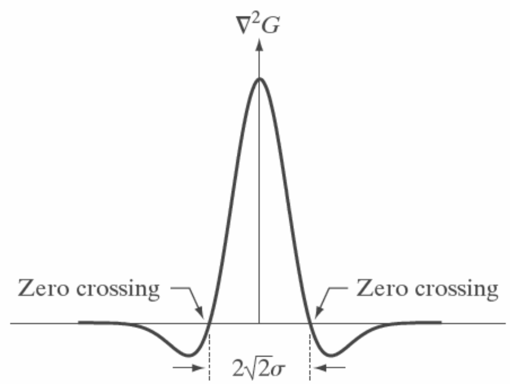
$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

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

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







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

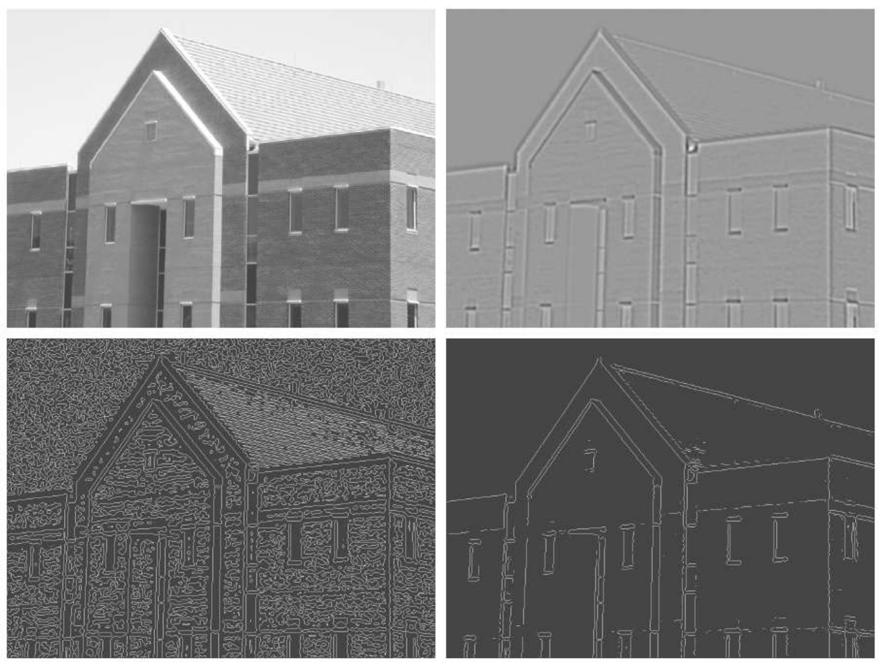
Edge detection: Marr-Hildreth method

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

<u>In practice</u>

- sample Gaussian function [nxn samples]
- convolve with image f(x,y): image smoothing
- convolve result with Laplacian mask
- find zero-crossings of g(x,y) ⟶ How?

1	1	1
1	−8	1
1	1	1



$g(x,y)$

Low zero-crossing threshold

Higher zero-crossing threshold

Edge detection: Canny method

* An optimal method for detecting step edges corrupted by white noise

•<u>Goal</u> Satisfy the following 3 criteria:

- Detection: should not miss important edges
- Localization: distance between the actual and located position of the edge should be minimal
- One response: only one response (edge) to a single actual edge

•<u>Algorithm</u>

- Step 1: Smooth input image with a Gaussian filter
- Step 2: Compute the gradient magnitude and angle images

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

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

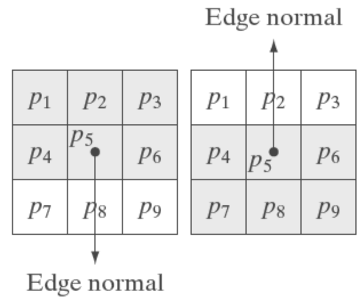
$$M(x,y) = \sqrt{g_x^2 + g_y^2}$$

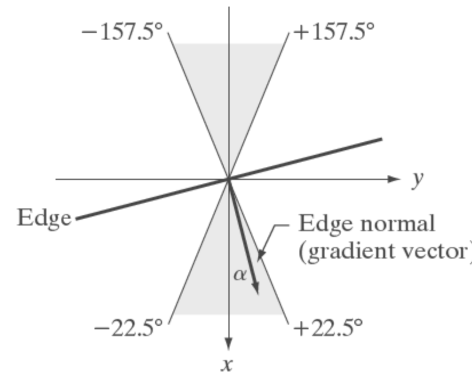
$$\alpha(x,y) = \tan^{-1}\left[\frac{g_x}{g_y}\right]$$

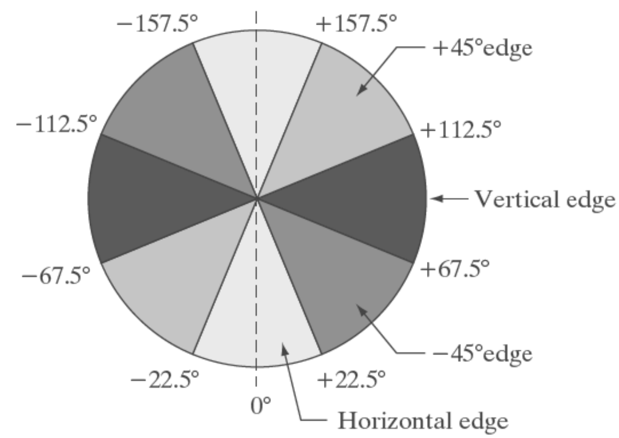
Edge detection –
Step 3: nonmaxima suppression

1. Find direction d_k closest to $\alpha(x, y)$
2. If value of M(x,y) is less than at least one of its neighbors along d_k, let $g_N(x, y) = 0$, otherwise

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







Edge detection –
Step 4: Reducing false edge points (hysteresis thresholding)

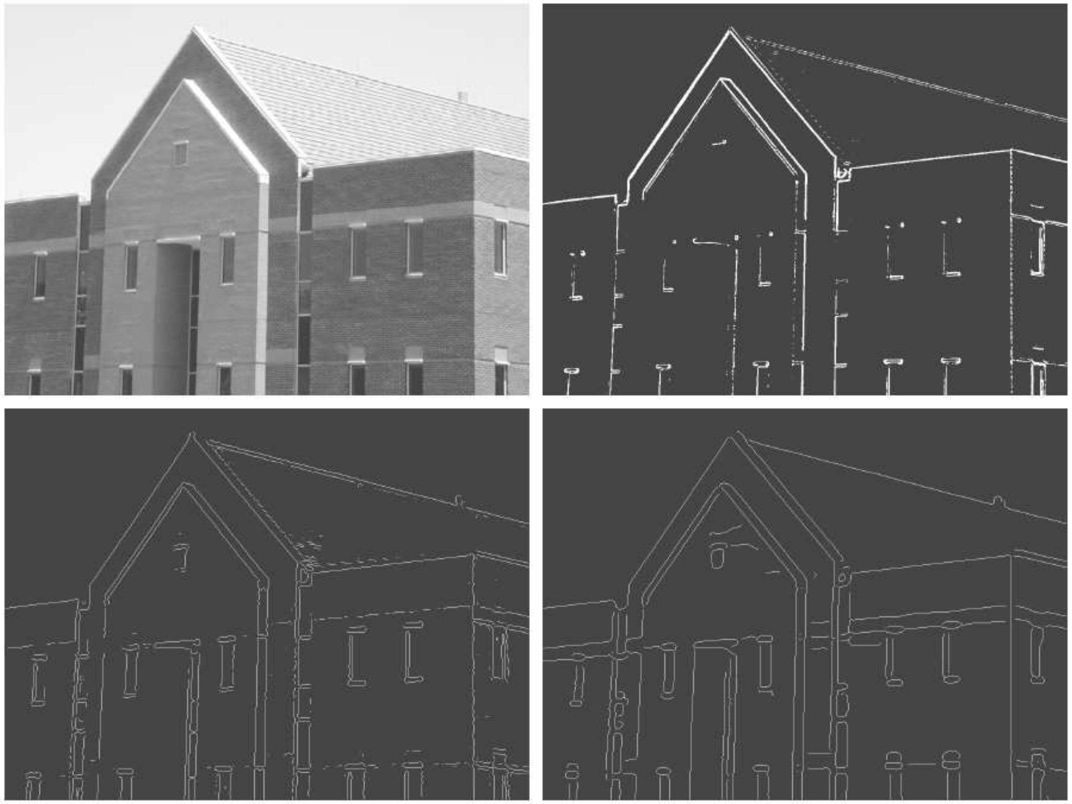
Strong edges: $$g_{NH}(x,y) = g_N(x,y) \geq T_H$$

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

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

1. Locate unvisited edge pixel in the strong edge image
2. Mark as valid edge pixels all the weak pixels that are connected to above pixel in the neighborhood
3. If all nonzero pixels in strong edge image havee been visited continue, otherwise go to (1)
4. Set to zero all pixels in weak edge image not already marked
5. Append all remaining non-zero pixels from weak edge image to strong edge image

Edge detection: Canny method

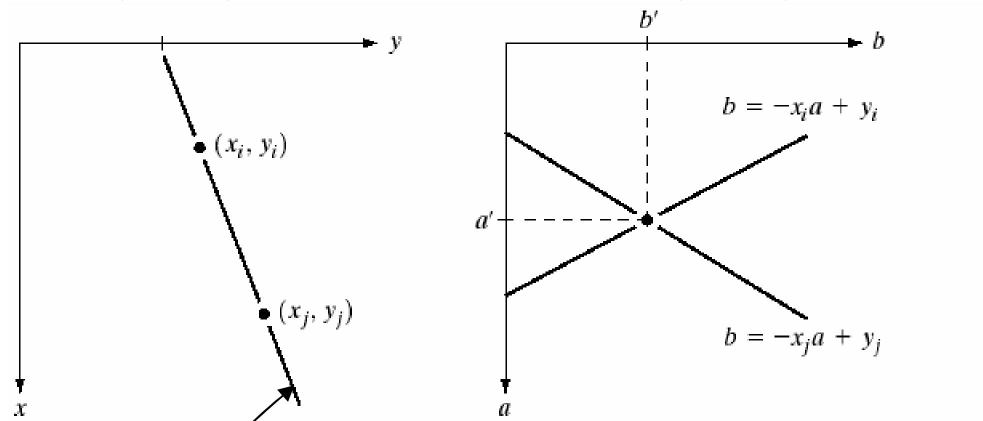


Hough Method for Curve Detection

Goal: find subset of points in the image that lie on a straight line

$$y_i = ax_i + b$$
$$y_j = ax_j + b$$

$$b = -x_i a + y_i$$
$$b = -x_j a + y_j$$



a b

FIGURE 10.17 (a) *xy*-plane. (b) Parameter space.

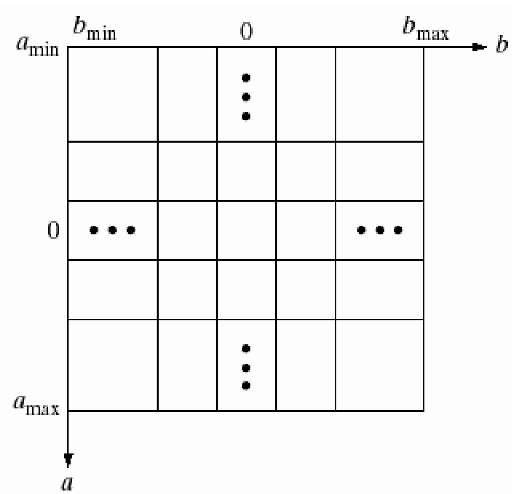
(a, b)

Accumulator matrix

$$A(p,q)$$

$$A(p,q) = M$$

$\Rightarrow$ M points in image with slope a_p and intercept b_q

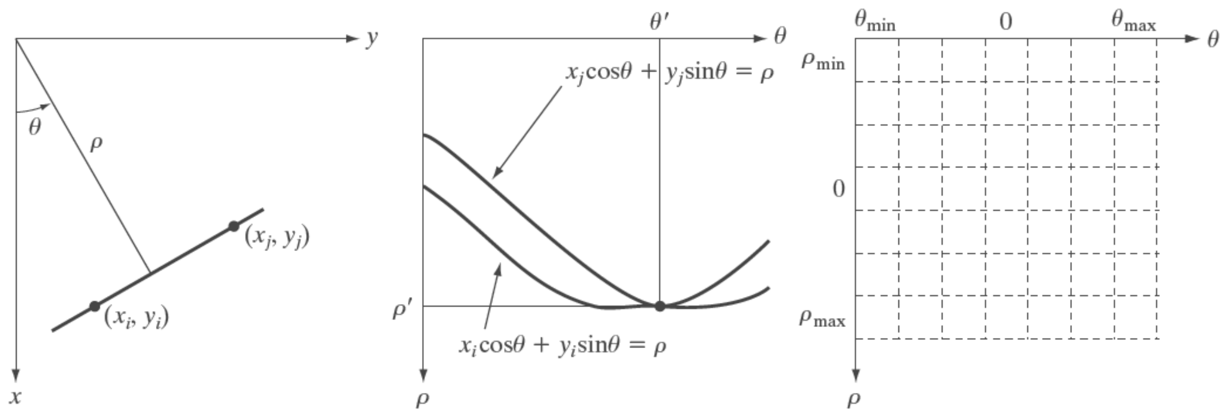


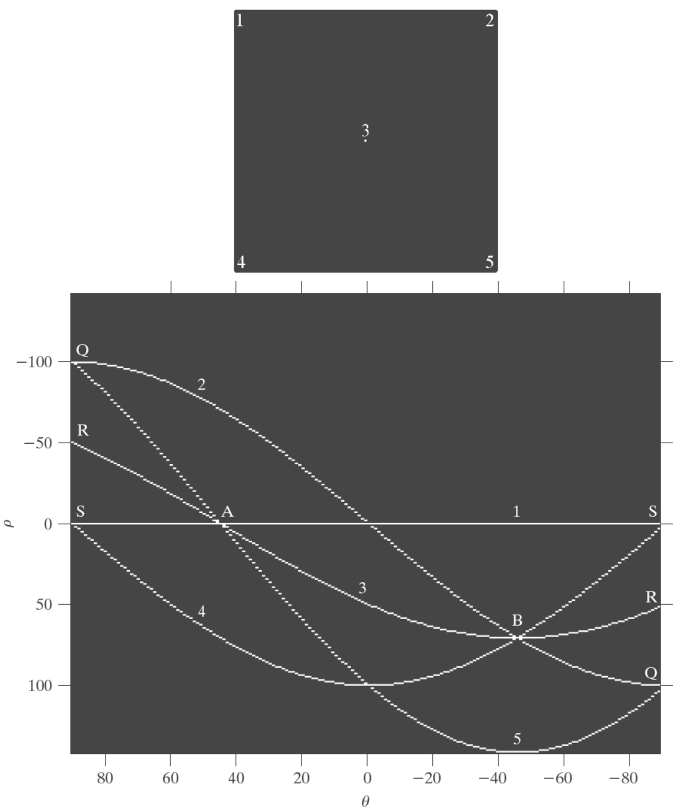
Algorithm:

1. For each point (x,y) in image determine a_p and b_q that satisfy the line equation
2. Increment A(p,q) by 1

Hough Method: Normal line representation

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





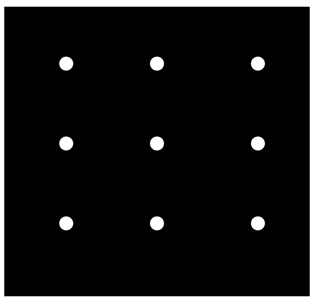
Algorithm:

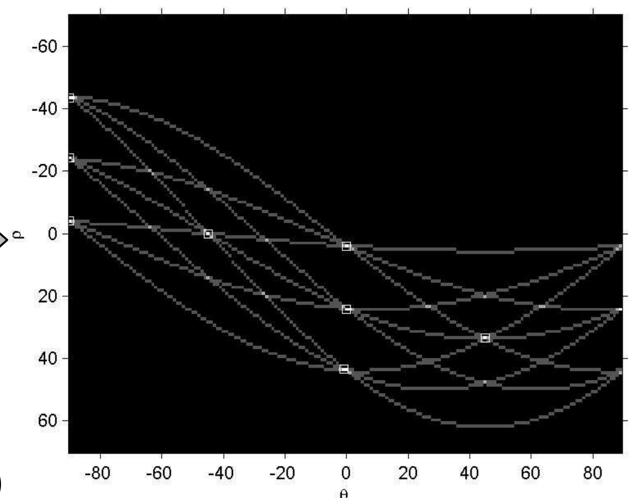
1. Quantize parameter space
2. Initialize the accumulator matrix
3. For each point in the edge image, iterate on choices of angle bins and find distance. Increase corresponding accumulator bins
4. Lines in image correspond to the local maxima in the accumulator array

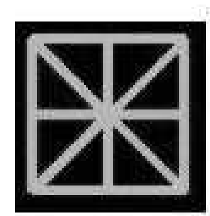
Hough Method: Matlab

image

accumulator



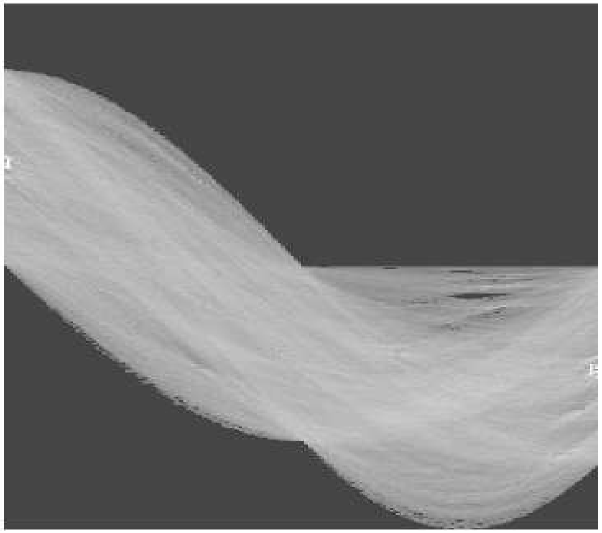




`hough()`

`houghpeaks()`

`houghlines()`

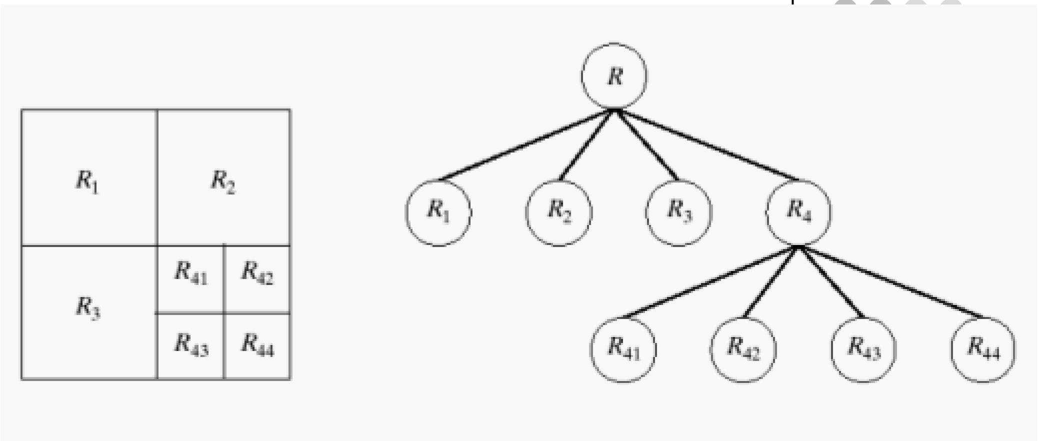


Edge-based segmentation: Problems

- Spurious edges due to noise and low quality image. Difficult to identify spurious edges.
- Dependent on local neighborhood information
- No notion of higher order organization of the image
- Gaps and discontinuities in the linked edges

Split & Merge Method

1. Pick a grid structure and homogeneity property H
2. If for any region R, H(R)=false, split region into 4
3. If for any neighboring regions $H(R_1 U R_2)$=true, merge regions into single region
4. Stop when no more split or merge

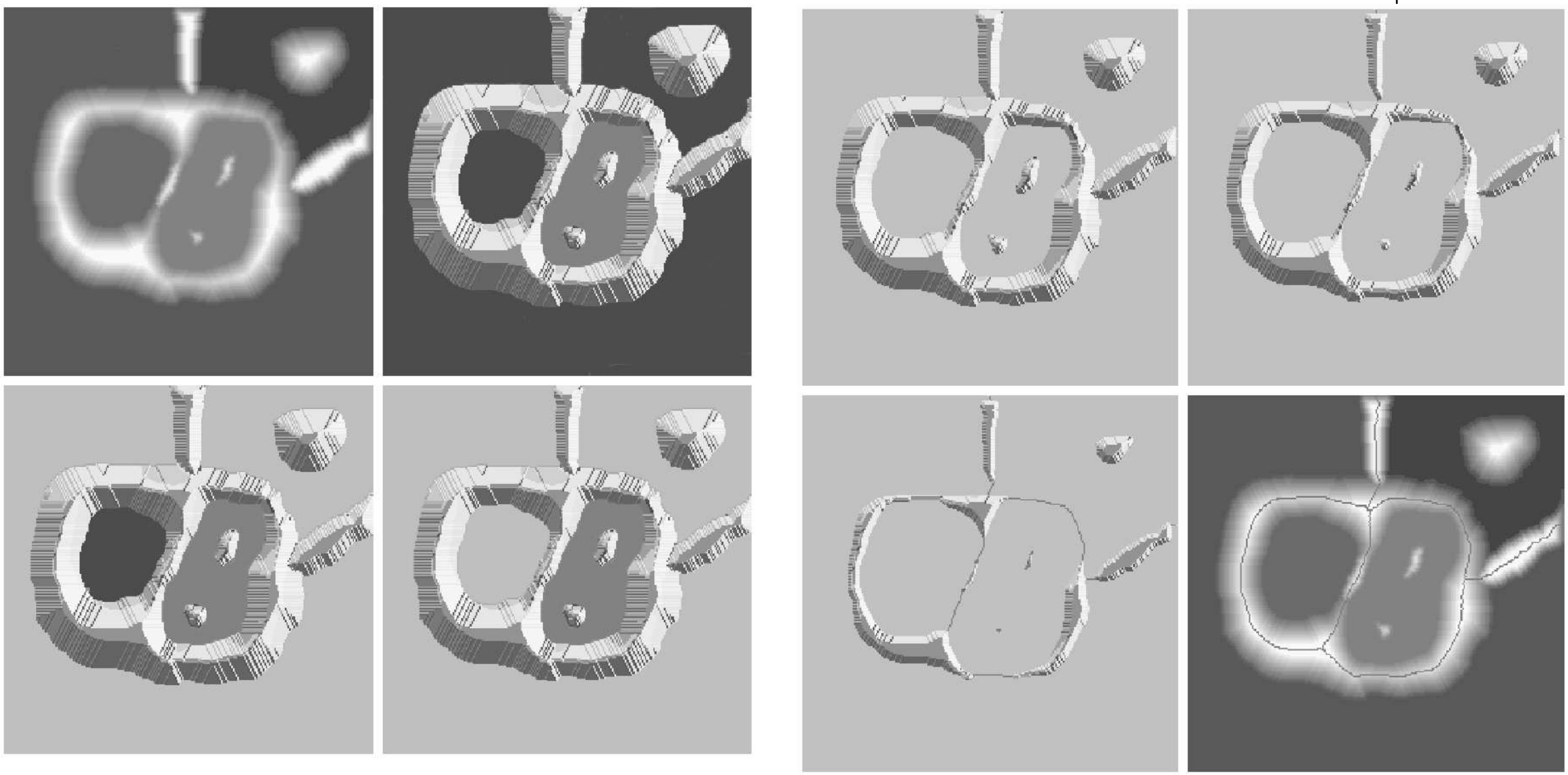


a b c
FIGURE 10.43 (a) Original image. (b) Result of split and merge procedure. (c) Result of thresholding (a).

$$H(R_k) = true \quad cnt(| z_j - m_k | \leq 2\sigma_k) \geq 0.8$$

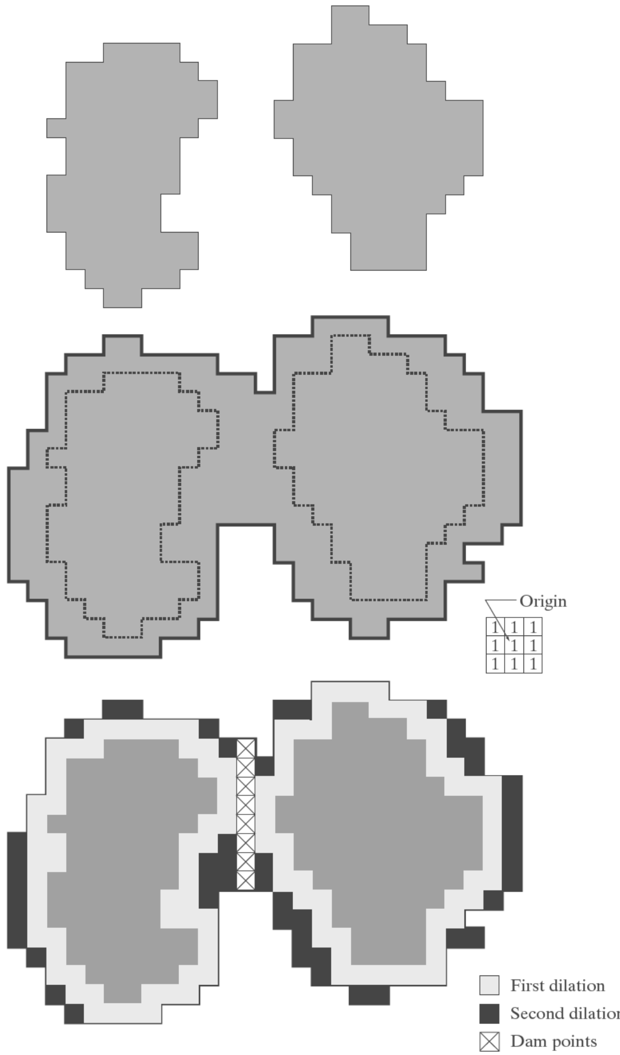
Watershed

Goal: Find watershed lines

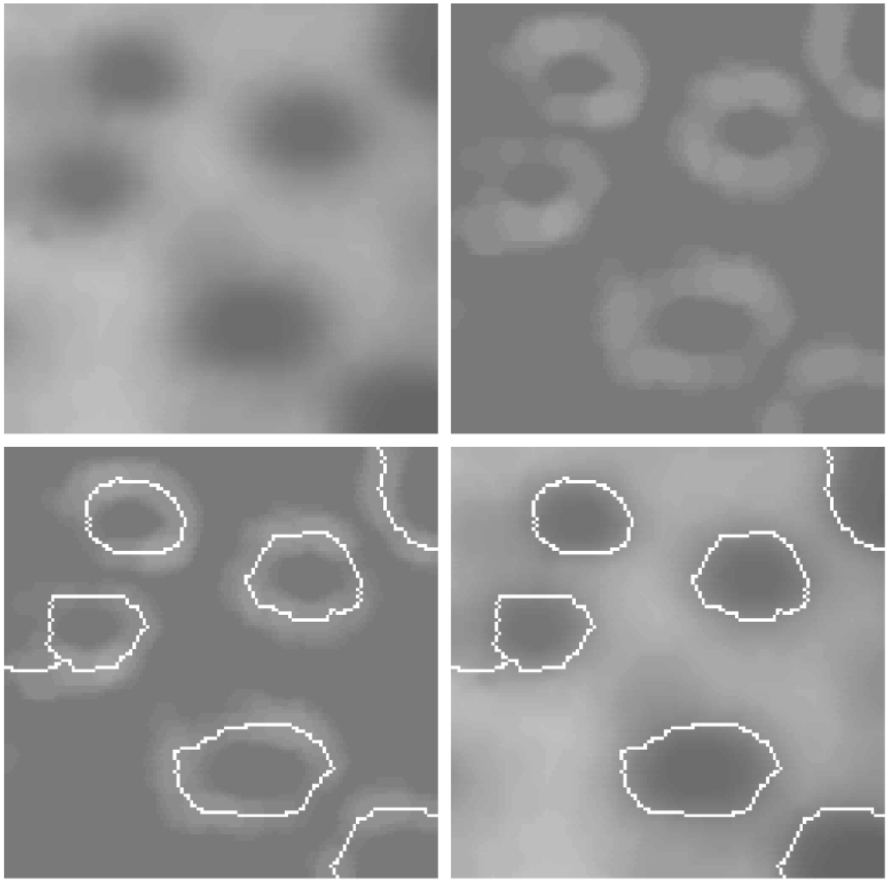


Build dams to prevent water flow from one catchment basin to other

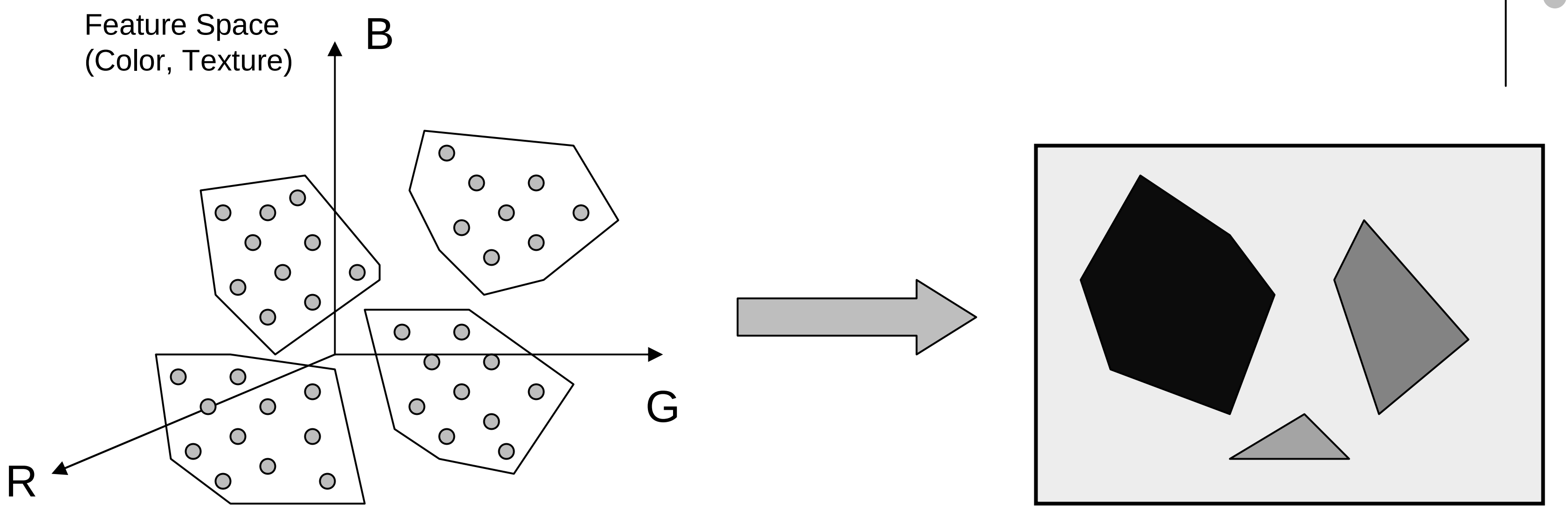
Boundaries between flooded catchment basins



Segmentation using Watershed



Segmentation as Clustering in Feature Space



Clustering algorithms:

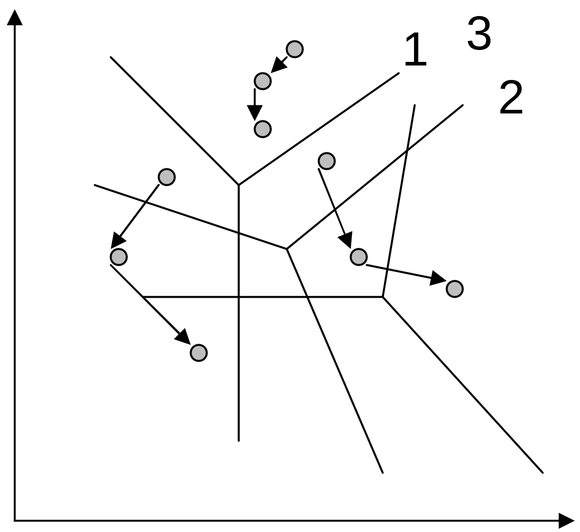
- K-means
- Gaussian Mixture Model
- Neural Network

Issues/Choices:

- Feature
- Distance
- Method

K-means Clustering Algorithm

- Select K initial cluster centers: $C_1, C_2, \cdots, C_K$
- Assign each pixel representation x_i to nearest cluster C_k
- Recomputer cluster centroids for all clusters
- Iterate until convergence



Change in clusters and migration of centroids in consecutive steps