

EE 4211 Computer Vision

Lecture 9: Image segmentation

Semester A, 2020-2021

Schedules

Week	Date	Topics
1	Sep. 4	Introduction/Imaging
2	Sep. 11	Image enhancement in spatial domain
3	Sep. 18	Image enhancement in frequency domain (HW1 out)
4	Sep. 25	Morphological processing
5	Oct. 2	Image restoration(HW1 due)
6	Oct. 9	Image restoration
7	Oct. 16	Midterm (no tutorials this week)
8	Oct. 23	Edge detection (HW2 out, illustrate the project)
9	Oct. 30	Image segmentation (HW2 due)
10	Nov. 6	Face recognition with PCA, LDA (tutorial on deep learning framework)
11	Nov. 13	Face recognition based on deep learning Image segmentation based on deep learning (tutorial on coding)
12	Nov. 20	Object detection with traditional methods (Quiz) Object detection based on deep learning
13	Nov. 27	Project presentation and summary
14	Dec. 4	Review and Summary

Review: Segmentation – Categories

- Edge-based Segmentation
 - Finding boundary between adjacent regions
- Threshold-based Segmentation
 - Finding regions by grouping pixels of similar gray values
- Region-based Segmentation
 - Finding regions directly using growing or splitting
- Motion-based Segmentation
 - Finding regions by comparing successive frames of a video sequence to identify regions that correspond to moving objects

Review: Edge-based Segmentation

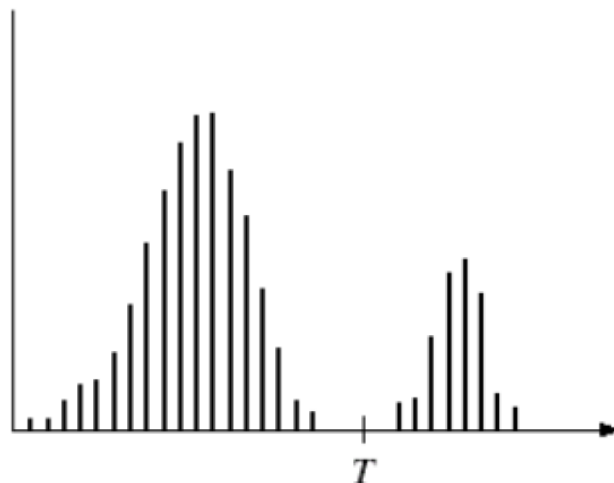
- Finding discontinuities (sharp, local changes in intensity) as boundary of regions
- Discontinuities in digital images (**Spatial filters**)
 - Point (**Laplacian**)
 - Line (**horizontal, -45 degree, vertical, 45 degree**)
 - Edges (**Roberts, Prewitt, Sobel, LOG, Canny**)
- Techniques
 - Point detection
 - Edge (pixel) detection
 - Edge formation from edge pixels – **Edge linking, Hough Transform**

Outline

- Thresholding
 - Overview
 - Basic Global Thresholding
 - Optimum Global Thresholding (OSTU)
 - Multiple Thresholding
 - Variable Thresholding
- Region based segmentation
 - Region grow
 - K-means method

Overview

- Suppose the intensity histogram of an image, $f(x,y)$ composed of light objects on a dark background. Assuming that the intensity values are grouped into two dominant modes in the distribution.
- Obvious method – extract the objects (light) from the background (dark) by selecting a threshold, T , that separates these modes



Overview

- Any point (x,y) in the image at which $f(x,y) > T$ is called an object point, otherwise, it is called a background point
- The segmented (thresholded) image, $g(x,y)$ is:

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

- T is a constant applicable over an entire image – Global Thresholding

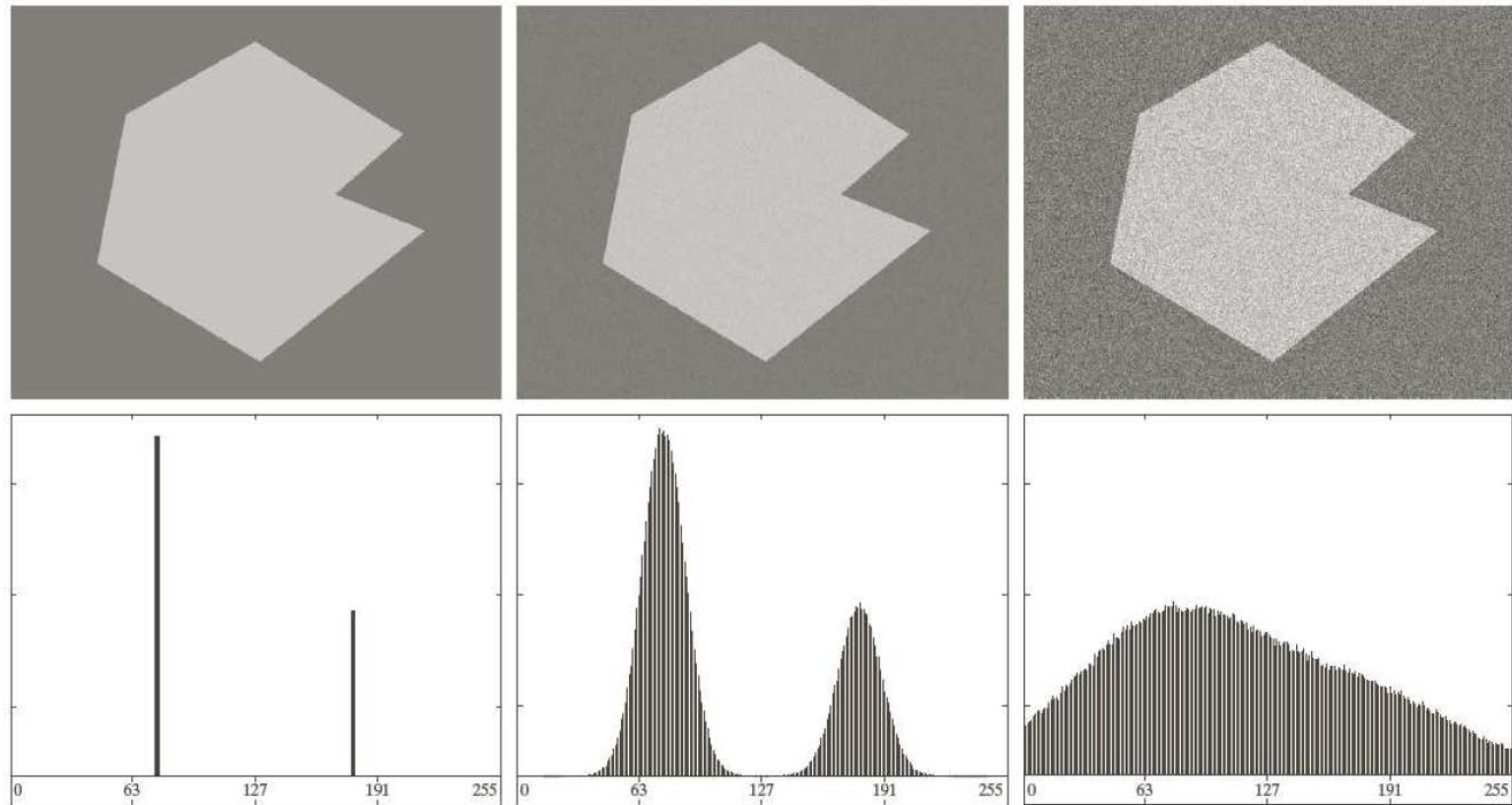
Basics of Intensity Thresholding

- A histogram with three dominant modes (two light objects on a dark background) – Multiple Thresholding classifies a point (x,y) as belonging to the background if $f(x,y) \leq T_1$, to one object class if $T_1 < f(x,y) \leq T_2$, and to the other object class if $f(x,y) > T_2$
- The segmented image is given by:

$$g(x, y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \leq T_2 \\ c & \text{if } f(x,y) \leq T_1 \end{cases}$$

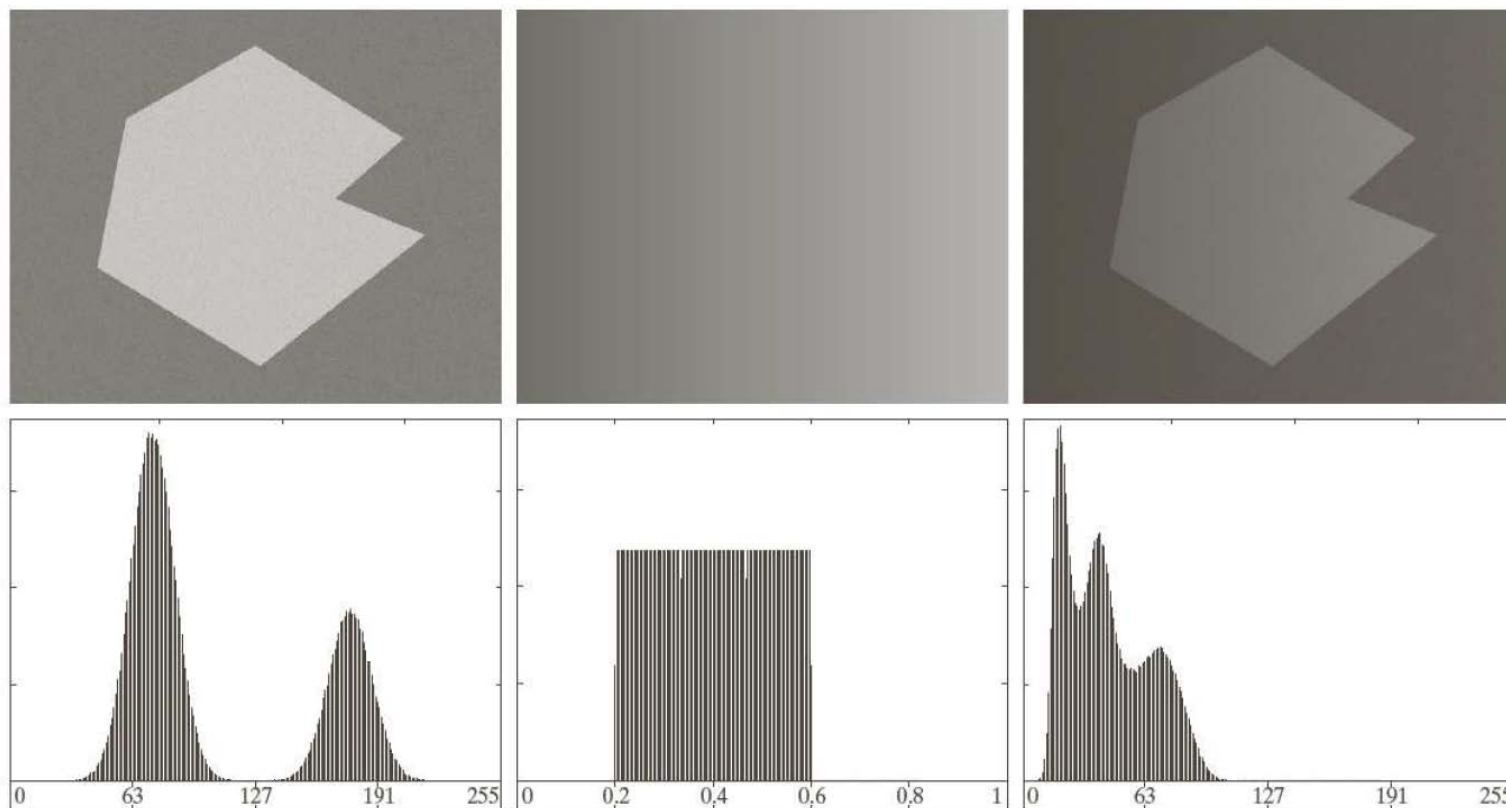
Effect of Noise

- With the added Gaussian noise, there is no way to differentiate between the two modes!



Effect of Illumination & Reflectance

- Nonuniform illumination causes the deep valley between peaks to be corrupted to the point where separation of the modes is no longer possible



Key Factor – The Valleys

- Intuitively, we can say that the success of intensity thresholding is directly related to the **width and depth** of the valley(s) separating the histogram modes
- Key factors affecting the properties of the valley(s):
 - Separation between peaks
 - Noise content
 - Relative size of objects and background
 - Uniformity of illumination source

Outline

- Thresholding
 - Overview
 - **Basic Global Thresholding**
 - Optimum Global Thresholding (OSTU)
 - Multiple Thresholding
 - Variable Thresholding
- Region based segmentation
 - Region grow
 - K-means method

Basic Global Thresholding

- Global thresholding may be sufficient when there is enough variability between images (distribution of object pixels and background pixels are sufficiently distinct)
- Even if global thresholding is a suitable approach, we need an algorithm to automatically estimate the threshold value for each image

Algorithm: Basic Global Thresholding

[1] Select an initial estimate for the global threshold T



[2] Segment image using $T \Rightarrow$ Two groups of pixels

$$G_1 \Leftarrow f(x, y) > T, \quad G_2 \Leftarrow f(x, y) \leq T$$



[3] Calculate mean of intensity values m_1, m_2 of G_1, G_2



[4] Compute new threshold value $T = \frac{1}{2}(m_1 + m_2)$



[5] Go to [2] until the difference in values of T is small enough

Algorithm: Basic Global Thresholding

- Select an initial estimate for the global threshold, T
- Threshold the image using T . This will produce two groups of pixels: G_1 consisting of all pixels with intensity values $> T$, and G_2 consisting of pixels with values $\leq T$.
- Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2 , respectively.
- Compute a new threshold value: $T = \frac{1}{2}(m_1 + m_2)$
- Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined parameter ΔT .

Basic Global Thresholding

- This simple algorithm works well in situations where there is a reasonably clear valley between modes of the histogram
- **Parameter ΔT is used to control the number of iterations** when speed is an important issue
 - Larger ΔT , fewer iterations, faster
 - Smaller ΔT , more iterations, slower
- The average intensity of the image is a good initial choice for T .

Basic Global Thresholding



Basic Global Thresholding

```
%% gloabl threshold
f = imread('7.tif');
T=0.5*(double(min(f(:)))+double(max(f(:))));
done=false;
while ~done
    g=f>=T;
    Tnext=0.5*(mean(f(g))+mean(f(~g)));
    done=abs(T-Tnext)<0.5;
    T=Tnext;
end
figure;
subplot(1,2,1);imshow(f);title('original image')
subplot(1,2,2);imshow(g);title('Global threshold')
```

Disadvantages

- Basic Global Thresholding requires **successive calculation** of means (for both groups of pixels) at each step
- It is possible to develop a more efficient and accurate procedure by expressing all computations in terms of the image histogram, which has to be computed only once

Outline

- Thresholding
 - Overview
 - Basic Global Thresholding
 - Optimum Global Thresholding (OSTU)
 - Multiple Thresholding
 - Variable Thresholding
- Region based segmentation
 - Region grow
 - K-means method

Optimum Global Thresholding

- View thresholding as a statistical-decision theory problem
- **Objective:** Minimize the average error incurred in assigning pixels to two or more groups (or classes)
- Solution – based on only two parameters
 - The probability density function (PDF) of intensity levels of each class
 - Probability of each class in the application
- Estimating PDFs is a common problem
- Alternative: **Otsu's Method**

Otsu's Method

- Maximizes the between-class variance
- Basic idea: Well-thresholded classes should be distinct with respect to the intensity values of pixels
- A threshold giving the best separation between classes in terms of intensity values will be the best (optimum) threshold
- Histogram of image – does create a PDF of intensity values
- To measure the ratio of between-class variance to global variance

Otsu's Method

Otsu's Method $\Rightarrow$ Maximize the between-class variance

p_i : Normalized histogram of the i^{th} intensity levels $\Rightarrow \sum_{i=0}^{L-1} p_i = 1$

Threshold Level $T = k$

$f(x, y) \in [0, k] \Rightarrow \text{Class } C_1$

Probability that $f(x, y)$ is in class C_1
when the threshold level is k

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

$f(x, y) \in [k + 1, L - 1] \Rightarrow \text{Class } C_2$

Probability that $f(x, y)$ is in class C_2
when the threshold level is k

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

Otsu's Method

Otsu's Method

Bayes' Formular

Mean intensity of the pixels in class C_1 }
$$m_1(k) = \sum_{i=0}^k iP(i|C_1) = \sum_{i=0}^k i \frac{P(C_1|i)P(i)}{P(C_1)} = \frac{1}{P_1(k)} \sum_{i=0}^k ip_i$$

Mean intensity of the pixels in class C_2 }
$$m_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^L ip_i$$

Global mean
$$m_G = \sum_{i=0}^{L-1} ip_i \Rightarrow P_1m_1 + P_2m_2 = m_G$$

Cumulative mean
$$m(k) = \sum_{i=0}^k ip_i$$

Otsu's Method

Otsu's Method

Separability Measure $\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$

Global Variance $\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$

Between-class variance
$$\begin{aligned}\sigma_B^2(k) &= P_1(k)(m_1(k) - m_G)^2 + P_2(k)(m_2(k) - m_G)^2 \\ &= \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))}\end{aligned}$$

The farther two means $\Rightarrow$ the larger σ_B^2

Find k to maximize η

Otsu's Method

[1] Compute the normalize histogram $p_i, i = 1, 2, \dots, L-1$



[2] Compute $P_1(k) = \sum_{i=0}^k p_i, k = 1, 2, \dots, L-1$



[3] Compute $m(k) = \sum_{i=0}^k ip_i, k = 1, 2, \dots, L-1$



[4] Compute $m_G = \sum_{i=0}^{L-1} ip_i$



[5] Compute $\sigma_B^2(k) = \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))}, k = 1, 2, \dots, L-1$



[6] Obtain Otsu threshold $k^* = \arg \max_k \sigma_B^2(k)$



[7] Compute $\eta(k^*) = \frac{\sigma_B^2(k^*)}{\sigma_G^2}$

Otsu's Method

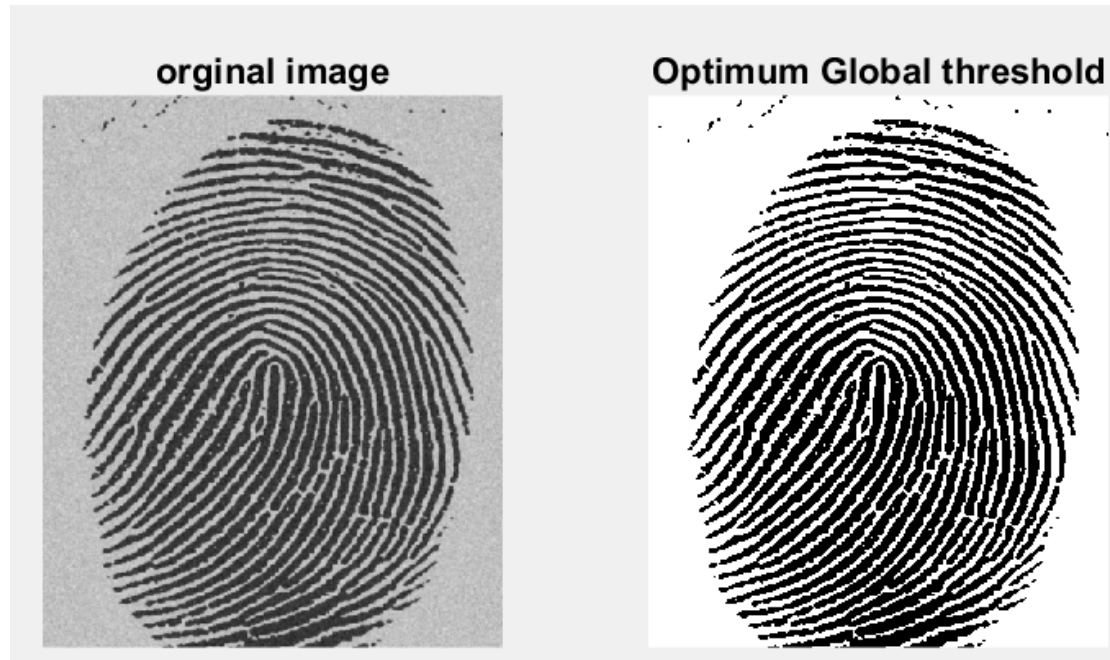
- By a bit of manipulation, you can calculate what is called the between class variance, which is far quicker to calculate. Luckily, the threshold with the maximum between class variance also has the minimum within class variance. So it can also be used for finding the best threshold and therefore due to being simpler is a much better approach to use.

■

Within Class Variance $\sigma_W^2 = W_b \sigma_b^2 + W_f \sigma_f^2$ (as seen above)

Between Class Variance $\sigma_B^2 = \sigma^2 - \sigma_W^2$
 $= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2$ (where $\mu = W_b \mu_b + W_f \mu_f$)
 $= W_b W_f (\mu_b - \mu_f)^2$

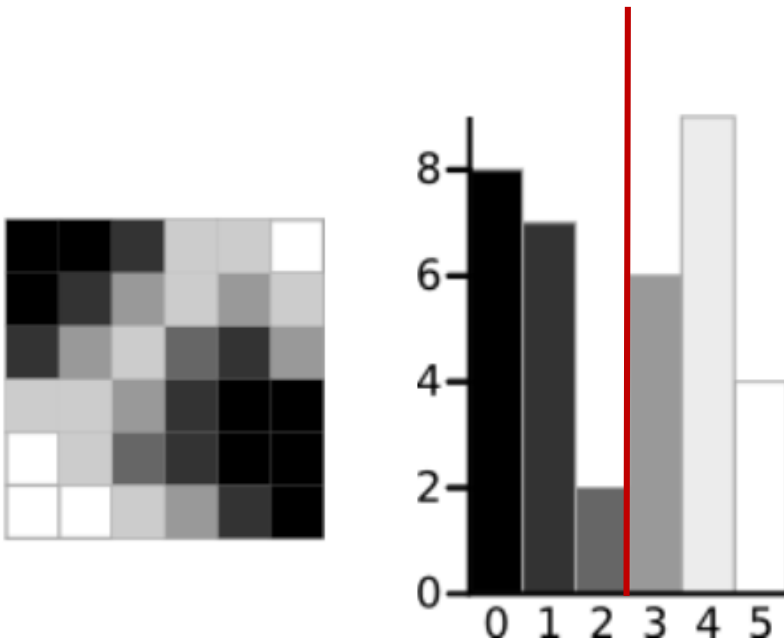
Example (1)



```
%% optimum global thresholdostu method
T=graythresh(f);
T=T*255;
g=f>=T;
figure;
subplot(1,2,1);imshow(f);title('original image')
subplot(1,2,2);imshow(g);title('Optimum Global threshold')
```

Example (2)

- The simple 6x6 image and its histogram is shown below. In this image, only 6 greyscale levels are used.



When the threshold is 3, the calculation of background

$$\text{Weight } W_b = \frac{8 + 7 + 2}{36} = 0.4722$$

$$\text{Mean } \mu_b = \frac{(0 \times 8) + (1 \times 7) + (2 \times 2)}{17} = 0.6471$$

$$\begin{aligned} \text{Variance } \sigma_b^2 &= \frac{((0 - 0.6471)^2 \times 8) + ((1 - 0.6471)^2 \times 7) + ((2 - 0.6471)^2 \times 2)}{17} \\ &= \frac{(0.4187 \times 8) + (0.1246 \times 7) + (1.8304 \times 2)}{17} \\ &= 0.4637 \end{aligned}$$

When the threshold is 3, the calculation of foreground

$$\text{Weight } W_f = \frac{6 + 9 + 4}{36} = 0.5278$$

$$\text{Mean } \mu_f = \frac{(3 \times 6) + (4 \times 9) + (5 \times 4)}{19} = 3.8947$$

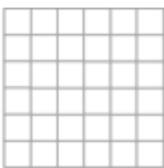
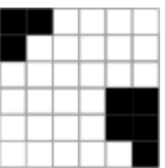
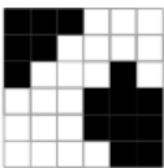
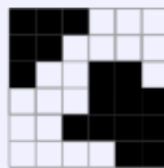
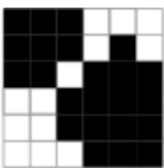
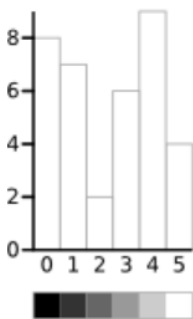
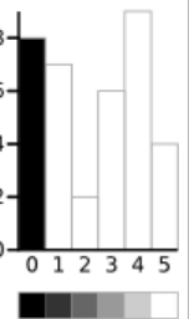
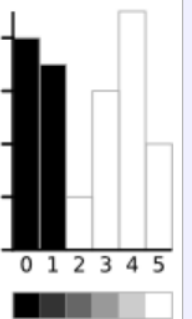
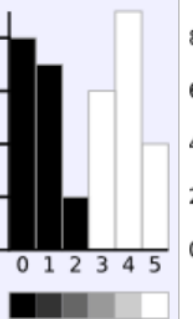
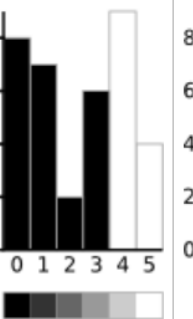
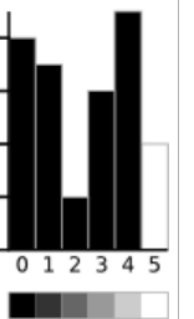
$$\begin{aligned} \text{Variance } \sigma_f^2 &= \frac{((3 - 3.8947)^2 \times 6) + ((4 - 3.8947)^2 \times 9) + ((5 - 3.8947)^2 \times 4)}{19} \\ &= \frac{(4.8033 \times 6) + (0.0997 \times 9) + (4.8864 \times 4)}{19} \\ &= 0.5152 \end{aligned}$$

Within-Class Variance

$$\sigma_W^2 = W_b \sigma_b^2 + W_f \sigma_f^2 = 0.4722 \times 0.4637 + 0.5278 \times 0.5152 = 0.4909$$

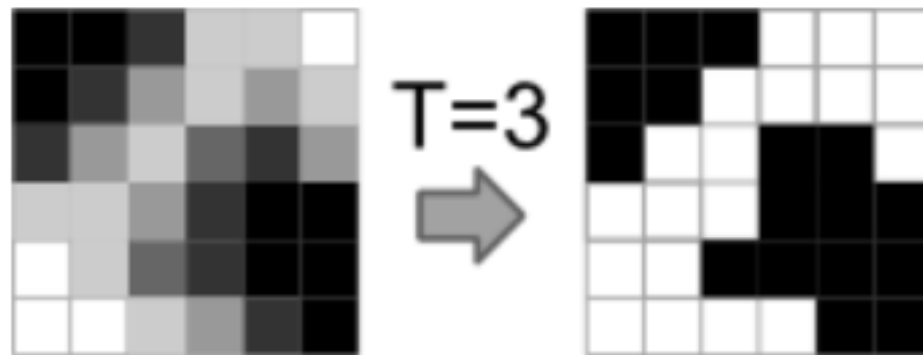
Example (2)

- This same calculation needs to be performed for all the possible threshold values 0 to 5.

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
						
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$	$W_b = 0.4722$	$W_b = 0.6389$	$W_b = 0.8889$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$	$M_b = 0.6471$	$M_b = 1.2609$	$M_b = 2.0313$
Variance, Background	$\sigma_b^2 = 0$	$\sigma_b^2 = 0$	$\sigma_b^2 = 0.2489$	$\sigma_b^2 = 0.4637$	$\sigma_b^2 = 1.4102$	$\sigma_b^2 = 2.5303$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$	$W_f = 0.5278$	$W_f = 0.3611$	$W_f = 0.1111$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$	$M_f = 3.8947$	$M_f = 4.3077$	$M_f = 5.0000$
Variance, Foreground	$\sigma_f^2 = 3.1196$	$\sigma_f^2 = 1.9639$	$\sigma_f^2 = 0.7755$	$\sigma_f^2 = 0.5152$	$\sigma_f^2 = 0.2130$	$\sigma_f^2 = 0$
Within Class Variance	$\sigma_W^2 = 3.1196$	$\sigma_W^2 = 1.5268$	$\sigma_W^2 = 0.5561$	$\sigma_W^2 = 0.4909$	$\sigma_W^2 = 0.9779$	$\sigma_W^2 = 2.2491$

Example (2)

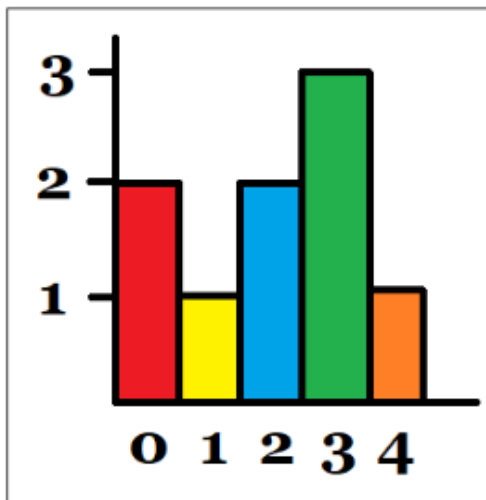
- The threshold equal to 3 has the lowest sum of weighted variances. Therefore, this is the final selected threshold. All pixels with a level less than 3 are background, all those with a level equal to or greater than 3 are foreground.



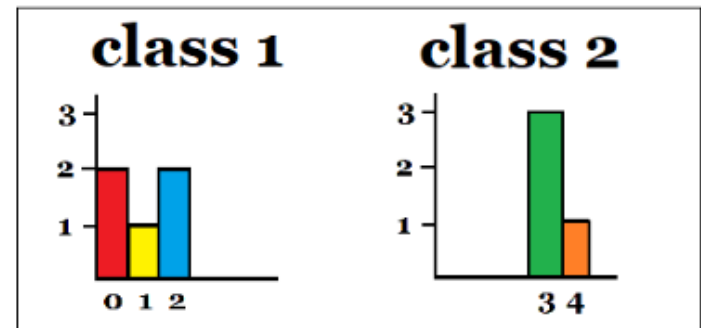
Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$	$\sigma^2_W = \mathbf{0.4909}$	$\sigma^2_W = 0.9779$	$\sigma^2_W = 2.2491$
Between Class Variance	$\sigma^2_B = 0$	$\sigma^2_B = 1.5928$	$\sigma^2_B = 2.5635$	$\sigma^2_B = \mathbf{2.6287}$	$\sigma^2_B = 2.1417$	$\sigma^2_B = 0.8705$

Example (3): Within class variance

if pixels are classified into **N classes** (categories),
then the **within class variance** (V_w) = $\sum_{i=0}^N (W_i * \sigma_i^2)$,
where W_i is (# of pixels in class i)/(total pixel)



divide into
→
two classes



$$W_1 = 5/9$$

$$W_2 = 4/9$$

$$\sigma_1^2 = 4/5$$

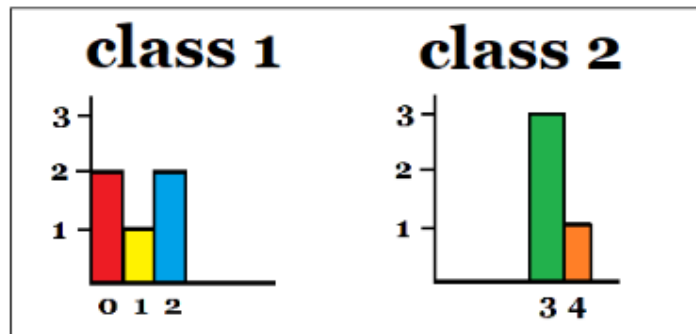
$$\sigma_2^2 = 3/16$$

$$V_w = W_1 * \sigma_1^2 + W_2 * \sigma_2^2 = 0.52777$$

Example (3): Between Class Variance

if pixels are classified into **N classes (categories)**,
then the **between class variance (V_b)** = $V_T - V_w$,
where V_T is the total variance

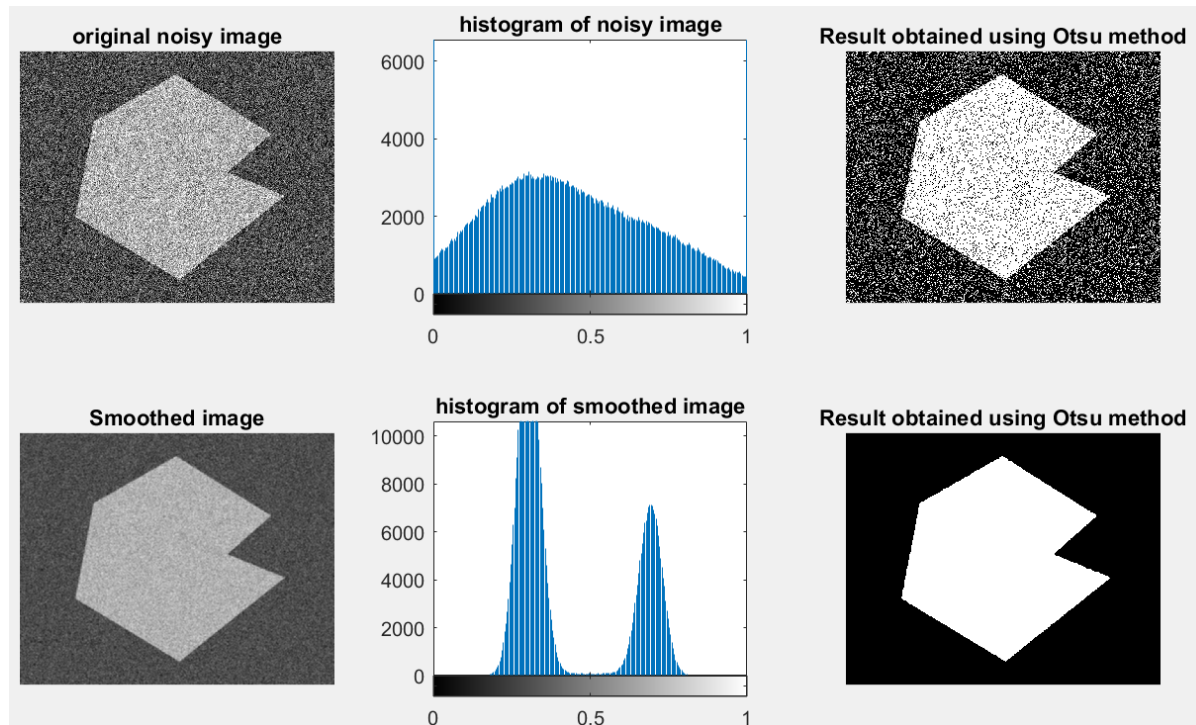
if pixels are classified into **2 classes**,
then the between class variance **(V_b)** = $W_1 W_2 (\mu_1 - \mu_2)^2$



$$W_1 = 5/9 \quad W_2 = 4/9$$
$$\mu_1 = 1 \quad \mu_2 = 13/4$$

$$V_b = \frac{5}{9} * \frac{4}{9} * (1 - \frac{13}{4})^2 = 1.25$$

Using smoothing to Improve Global Thresholding

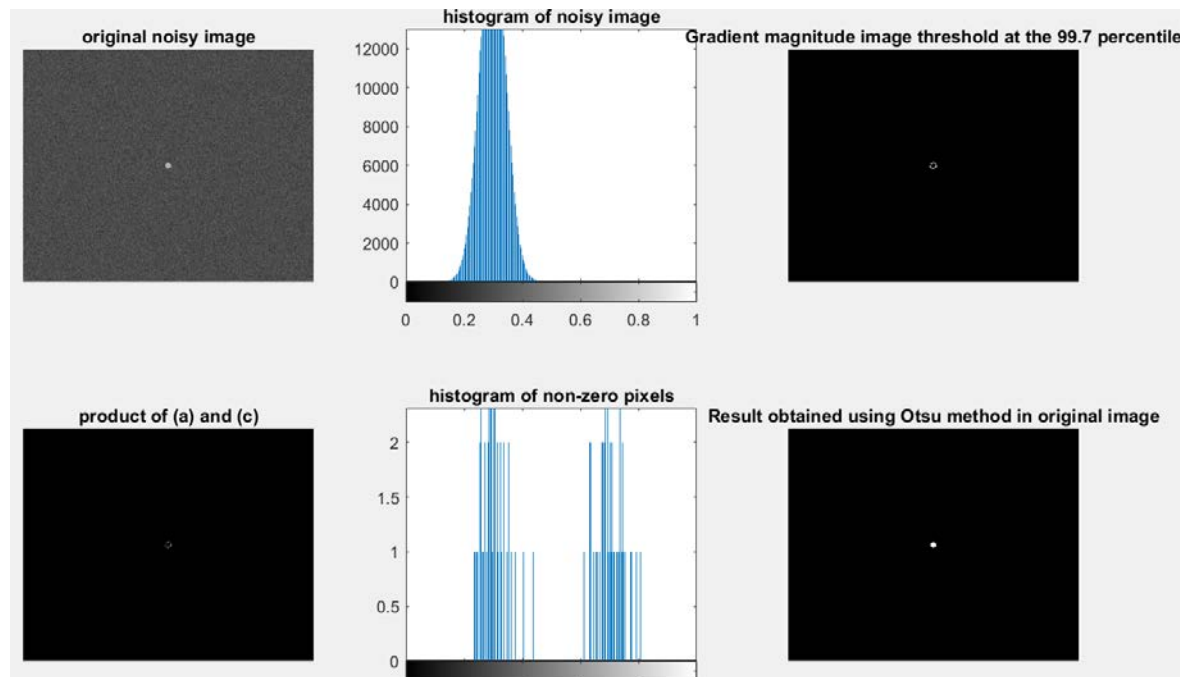


```
%% Reduce Noise By Smoothing
f = im2double(imread('3.tif'));
T1 = graythresh(f);
g1 = (f>=T1);

average_filter = 1/ (5.^2)*ones (5);
J = imfilter(f, average_filter,'conv','replicate','same');
T2 = graythresh(J);
g2 = (J>=T2);

figure;
subplot(231); imshow(f); title('original noisy image');
subplot(232); imhist(f); title('histogram of noisy image');
subplot(233); imshow(g1); title('Result obtained using Otsu method');
subplot(234); imshow(J); title('Smoothed image');
subplot(235); imhist(J); title('histogram of smoothed image');
subplot(236); imshow(g2); title('Result obtained using Otsu method');
```

Using Edges to Improve Global Thresholding



```
% Reduce Noise By gradient information
f = im2double(imread('4.tif'));
[Gmag, Gdir] = imgradient(f);
g1 = Gmag>=0.997;

image_d = f.*g1;
Nonzero_idx = (image_d>0.0);
T = graythresh(image_d(Nonzero_idx));
g2 = (f>=T);

figure;
subplot(231); imshow(f); title('original noisy image');
subplot(232); imhist(f); title('histogram of noisy image');
subplot(233); imshow(g1); title('Gradient magnitude image threshold at the 99.7 percentile');
subplot(234); imshow(image_d); title('product of (a) and (c)');
subplot(235); imhist(image_d(Nonzero_idx)); title('histogram of non-zero pixels');
subplot(236); imshow(g2); title('Result obtained using Otsu method in original image');
```

Outline

- Thresholding
 - Overview
 - Basic Global Thresholding
 - Optimum Global Thresholding (OSTU)
 - Multiple Thresholding
 - Variable Thresholding
- Region based segmentation
 - Region grow
 - K-means method

Multiple Thresholds

$$\sigma_B^2 = \sum_{j=1}^J P_j (m_j - m_G)^2$$

$$P_j = \sum_{i \in C_j} p_i$$

$$m_j = \frac{1}{P_j} \sum_{i \in C_j} ip_i$$



Double thresholding (k_1, k_2) for $0 < k_1 < k_2 < L-1$

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2$$

$$P_1 = \sum_{i=0}^{k_1} p_i$$

$$P_2 = \sum_{i=k_1+1}^{k_2} p_i$$

$$P_3 = \sum_{i=k_2+1}^{L-1} p_i$$

$$m_1 = \frac{1}{P_1} \sum_{i=0}^{k_1} ip_i$$

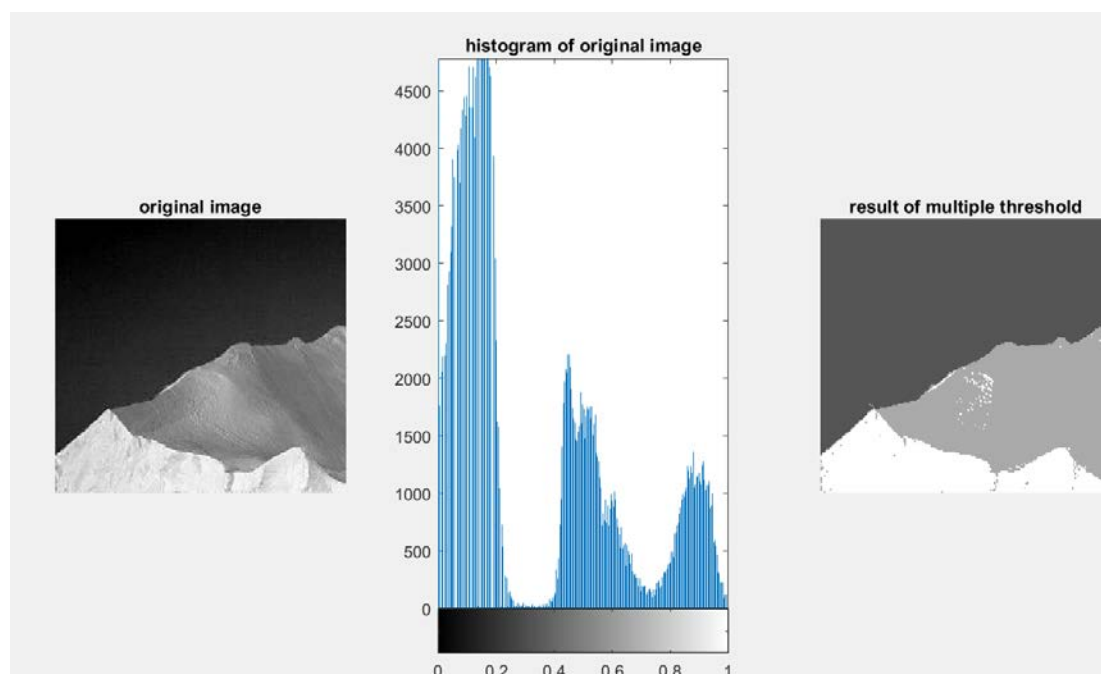
$$m_2 = \frac{1}{P_2} \sum_{i=k_1+1}^{k_2} ip_i$$

$$m_3 = \frac{1}{P_3} \sum_{i=k_2+1}^{L-1} ip_i$$

Find (k_1, k_2) to maximize $\sigma_B^2(k_1, k_2)$

Multiple Thresholds

- Objective: Segment image into three regions – the dark background, illuminated area of iceberg, area in shadows



```
% multiple thresholds
f = im2double(imread('5.tif'));
T = multithresh(f,2);
J = imquantize(f, T)/(length(T)+1);
figure;
subplot(131); imshow(f); title('original image');
subplot(132); imhist(f); title('histogram of original image');
subplot(133); imshow(J); title('result of multiple threshold');
```

Outline

- Thresholding
 - Overview
 - Basic Global Thresholding
 - Optimum Global Thresholding (OSTU)
 - Multiple Thresholding
 - **Variable Thresholding**
- Region based segmentation

Variable Thresholding

- In certain cases, preprocessing methods can be either impractical or simply still ineffective in improving the accuracy of thresholding.
- Variable thresholding (or adaptive thresholding) techniques can be used
- Threshold values can be made to change or vary according to local image regions
- One of the simplest approaches – **Image Partitioning**

Image Partitioning

- Subdivide an image into non-overlapping rectangles (areas)
 - Compensate for non-uniformities in illumination
 - Rectangles are to be chosen small enough so that illumination is approximately uniform

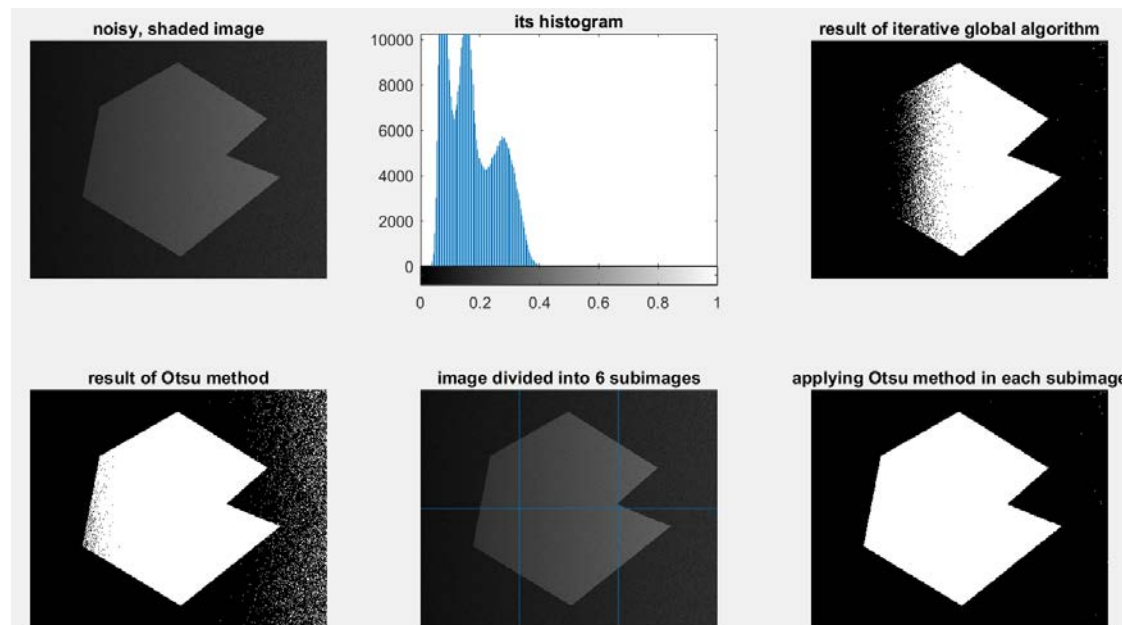


Image Partitioning

```
% image partitioning
f = im2double(imread('6.tif'));
[m n] = size(f);
%iterative global algorithm
T1=0.5*(double(min(f(:)))+double(max(f(:))));
done=false;
while ~done
    g1 = f>=T1;
    Tnext=0.5*(mean(f(g1))+mean(f(~g1)));
    done=abs(T1-Tnext)<0.5;
    T1=Tnext;
end
%Otsu's method
T2 = graythresh(f);
g2 = f>=T2;
% image partitioning
block_funtion = @(block_struct) (block_struct.data >= graythresh(block_struct.data));
g3 = blockproc(f,[m/2 n/3],block_funtion);

figure;
subplot(231); imshow(f); title('noisy, shaded image');
subplot(232); imhist(f); title('its histogram');
subplot(233); imshow(g1); title('result of iterative global algorithm');
subplot(234); imshow(g2); title('result of Otsu method');
subplot(235); imshow(f);line([0 n],[m/2 m/2]); line([n/3, n/3],[0, m]); line([2*n/3, 2*n/3],[0, m]);title('image divided into 6 subimages');
subplot(236); imshow(g3); title('applying Otsu method in each subimage');
```

Outline

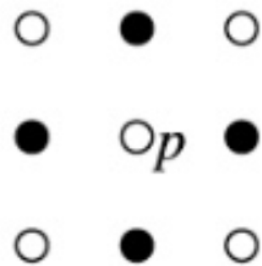
- Thresholding
 - Overview
 - Basic Global Thresholding
 - Optimum Global Thresholding (OSTU)
 - Multiple Thresholding
 - Variable Thresholding
- Region based segmentation
 - Region grow
 - K-means method

Region Growing

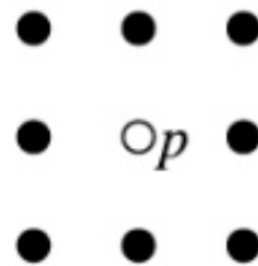
- Procedure that groups pixels or subregions into larger regions based on predefined criteria for growth
- **Basic approach:**
 - Start with a set of “seed” points
 - Grow regions by appending to each seed those neighbouring pixels that have predefined properties similar to the seed (intensity, color, etc.)
- Region growing based techniques are better than the edge-based techniques in noisy images where edges are difficult to detect.

Region grow: steps

- Select a seed points
- Define a growth criteria
- Joint all pixels connected to the seed that follow the growth criteria
- Stop when no adjacent pixel agree with the growth criteria



4-connectivity



8-connectivity

Example (1)

- Region Grown with 8-Connectivity
 - Seed: (8)
 - Criteria: the difference between two pixels is small than 1

$$\begin{bmatrix} 4 & 3 & 7 & 3 & 3 \\ 1 & 7 & (8) & 7 & 5 \\ 0 & 5 & 6 & 1 & 3 \\ 2 & 2 & 6 & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

(a) Original image with seed

$$\begin{bmatrix} 4 & 3 & (7) & 3 & 3 \\ 1 & (7) & (8) & (7) & 5 \\ 0 & 5 & 6 & 1 & 3 \\ 2 & 2 & 6 & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

(b) First result after region grow

$$\begin{bmatrix} 4 & 3 & (7) & 3 & 3 \\ 1 & (7) & (8) & (7) & 5 \\ 0 & 5 & (6) & 1 & 3 \\ 2 & 2 & 6 & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

(c) Second result after region grow

$$\begin{bmatrix} 4 & 3 & (7) & 3 & 3 \\ 1 & (7) & (8) & (7) & 5 \\ 0 & (5) & (6) & 1 & 3 \\ 2 & 2 & (6) & 0 & 4 \\ 1 & 2 & 1 & 3 & 1 \end{bmatrix}$$

(d) Third result after region grow

Example (2)

Suppose that we have the image given below.

(a) Use the region growing idea to segment the object. The seed for the object is the center of the image. Region is grown in horizontal and vertical directions, and when the difference between two pixel values is less than or equal to 5.

Table 1: Show the result of Part (a) on this figure.

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	<u>60</u>	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

Example (2)

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

Example (2)

■ Region Grown with 4-Connectivity

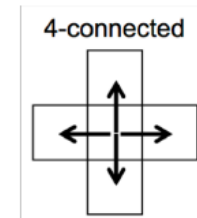
Suppose that we have the image given below.

(a) Use the region growing idea to segment the object. The seed for the object is the center of the image. Region is grown in horizontal and vertical directions, and when the difference between two pixel values is less than or equal to 5.

Table 1: Show the result of Part (a) on this figure.

10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	60	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

4-connectivity



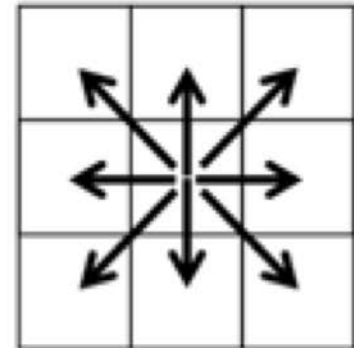
Example (2)

- Region Grown with 8-Connectivity

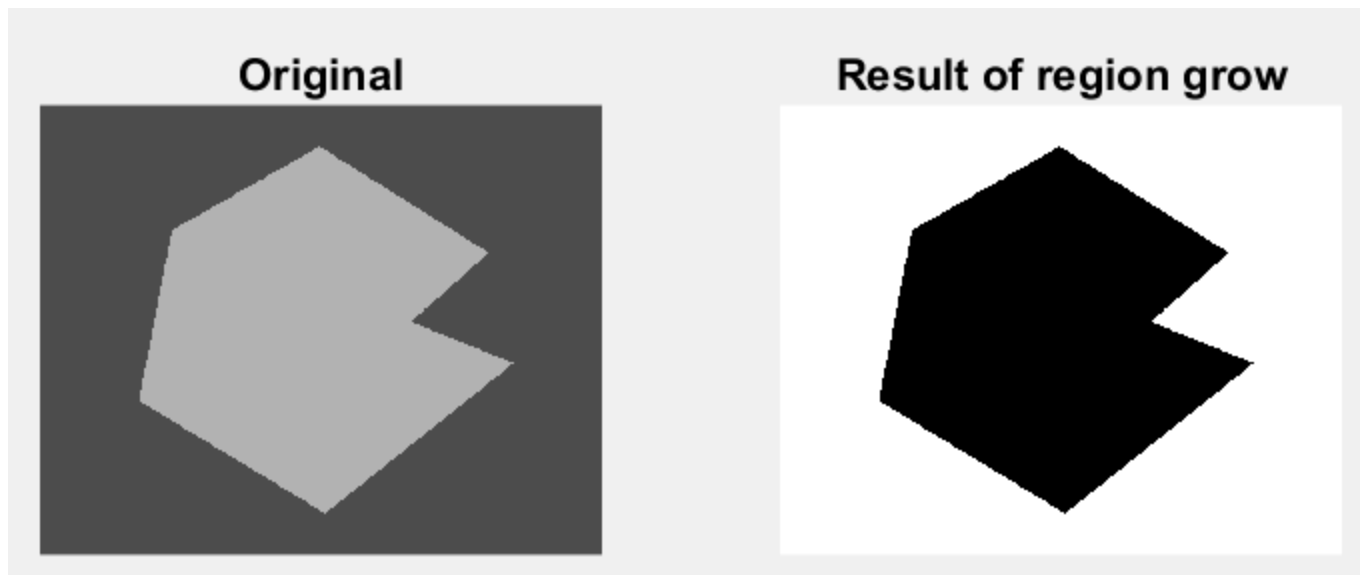
10	10	10	10	10	10	10
10	10	10	69	70	10	10
59	10	60	64	59	56	60
10	59	10	<u>60</u>	70	10	62
10	60	59	65	67	10	65
10	10	10	10	10	10	10
10	10	10	10	10	10	10

8-connectivity

8-connected



Example (3)



```
%% region grow
I= (im2double(imread('8.tif')));
figure,
x=138; y=249;
J = regiongrowing(I,x,y,0.05);
subplot(1,2,1);imshow(I);title('Original');
subplot(1,2,2);imshow(J);title('Result of region grow');
```

Example (3)

```
function J=regiongrowing(I,x,y,reg_maxdist)
% This function performs "region growing" in an image from a specified
% seedpoint (x,y)
%
% J = regiongrowing(I,x,y,t)
%
% I : input image
% J : logical output image of region
% x,y : the position of the seedpoint (if not given uses function getpts)
% t : maximum intensity distance (defaults to 0.2)
%
% The region is iteratively grown by comparing all unallocated neighbouring pixels to the
% The difference between a pixel's intensity value and the region's mean,
% is used as a measure of similarity. The pixel with the smallest difference
% measured this way is allocated to the respective region.
% This process stops when the intensity difference between region mean and
% new pixel become larger than a certain threshold (t)
%
% Example:
%
% I = imread('medtest.png');
% x=198; y=359;
% J = regiongrowing(I,x,y,0.2);
% figure, imshow(I+J);
%

if(exist('reg_maxdist','var')==0), reg_maxdist=0.2; end
if(exist('y','var')==0), figure, imshow(I,[]); [y,x]=getpts; y=round(y(1)); x=round(x(1))
J = zeros(size(I)); % Output
Isizes = size(I); % Dimensions of input image
reg_mean = I(x,y); % The mean of the segmented region
reg_size = 1; % Number of pixels in region
% Free memory to store neighbours of the (segmented) region
neg_free = 10000; neg_pos=0;
neg_list = zeros(neg_free,3);

while (pixdist<reg_maxdist&&reg_size<numel(I))
    % Add new neighbors pixels
    for j=1:4,
        % Calculate the neighbour coordinate
        xn = x +neighb(j,1); yn = y +neighb(j,2);

        % Check if neighbour is inside or outside the image
        ins=(xn>=1)&&(yn>=1)&&(xn<=Isizes(1))&&(yn<=Isizes(2));

        % Add neighbor if inside and not already part of the segmented area
        if(ins&&(J(xn,yn)==0))
            neg_pos = neg_pos+1;
            neg_list(neg_pos,:) = [xn yn I(xn,yn)]; J(xn,yn)=1;
        end
    end
    % Add a new block of free memory
    if(neg_pos+10>neg_free), neg_free=neg_free+10000; neg_list((neg_pos+1):neg_free,:)=0;

    % Add pixel with intensity nearest to the mean of the region, to the region
    dist = abs(neg_list(1:neg_pos,3)-reg_mean);
    [pixdist, index] = min(dist);
    J(x,y)=2; reg_size=reg_size+1;

    % Calculate the new mean of the region
    reg_mean= (reg_mean*reg_size + neg_list(index,3))/(reg_size+1);

    % Save the x and y coordinates of the pixel (for the neighbour add process)
    x = neg_list(index,1); y = neg_list(index,2);

    % Remove the pixel from the neighbour (check) list
    neg_list(index,:)=neg_list(neg_pos,:); neg_pos=neg_pos-1;
end
% Return the segmented area as logical matrix
J=J>1;
```

Outline

- Thresholding
 - Overview
 - Basic Global Thresholding
 - Optimum Global Thresholding (OSTU)
 - Multiple Thresholding
 - Variable Thresholding
- Region based segmentation
 - Region grow
 - K-means method

K-means Clustering

- One of the most commonly-used clustering algorithms, because it is easy to implement and quick to run.
- Clustering means grouping similar data together
- From a set of data points, K-means attempts to classify all data points into K clusters
- K clusters may represent K segments of an image
- The algorithm is iterative in nature and not confined to image segmentation

K-means: Steps

- Partition the data points into K clusters randomly. Find the centroids of each cluster.
- For each data point:
 - Calculate the distance from the data point to each cluster.
 - Assign the data point to the closest cluster.
- Recompute the centroid of each cluster.
- Repeat steps 2 and 3 until there is no further change in the assignment of data points (or in the centroids).

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of } i\text{'th cluster}} \|x_j - \mu_i\|^2 \right\}$$

Kmeans Convergence

Objective

$$\min_{\mu} \min_C \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix μ , optimize C :

$$\min_C \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2 = \min_C \sum_i^n |x_i - \mu_{x_i}|^2$$

Step 1 of kmeans

2. Fix C , optimize μ :

$$\min_{\mu} \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

- Take partial derivative of μ_i and set to zero, we have

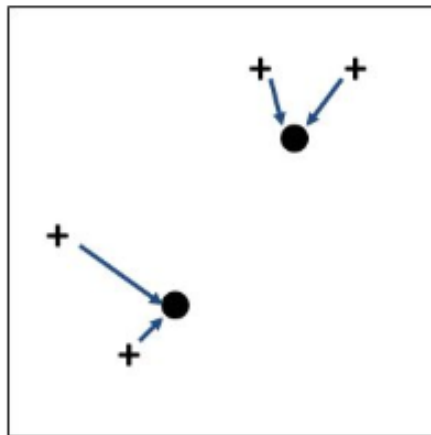
$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Step 2 of kmeans

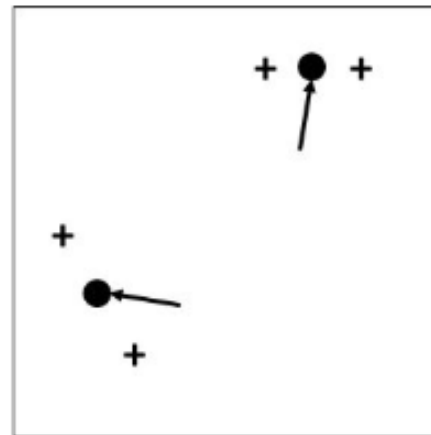
Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

Kmeans Convergence

- Each iteration includes two steps
- Assign points to clusters
- Recompute the cluster centers



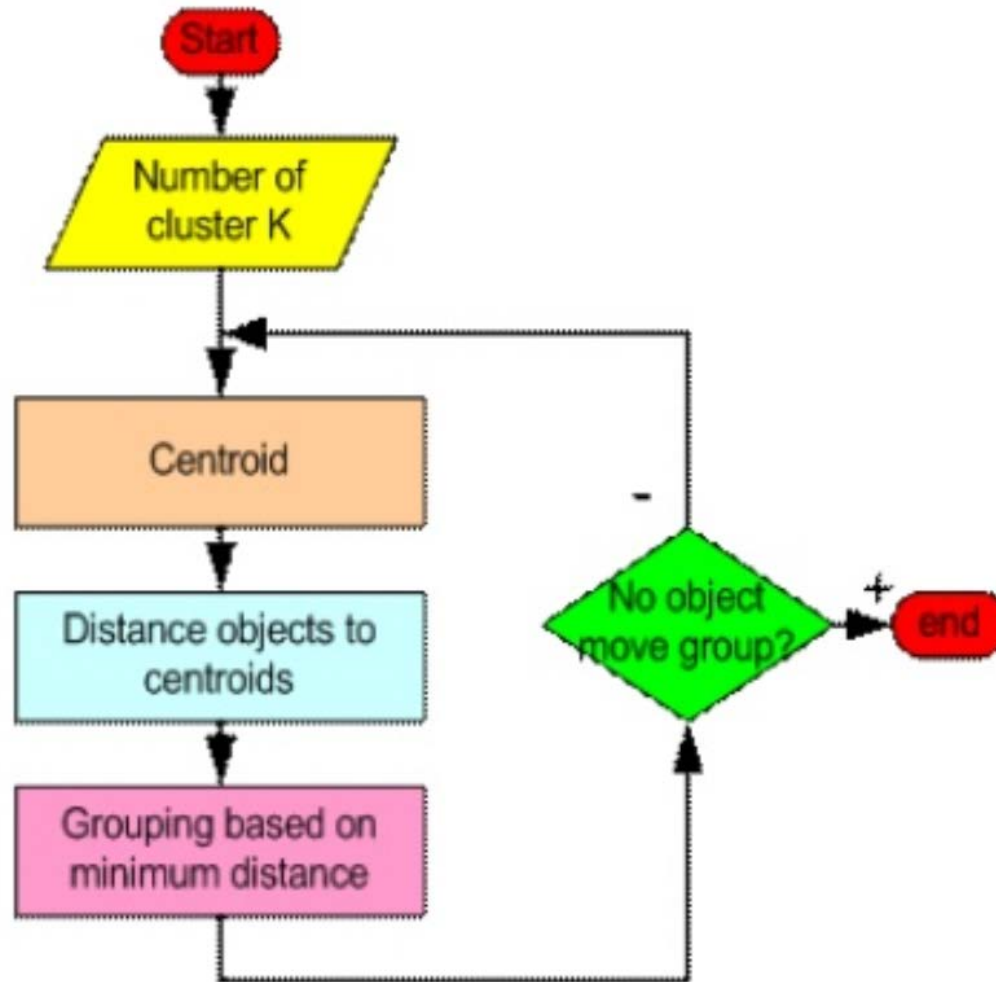
assignment



re-computation of cluster centers

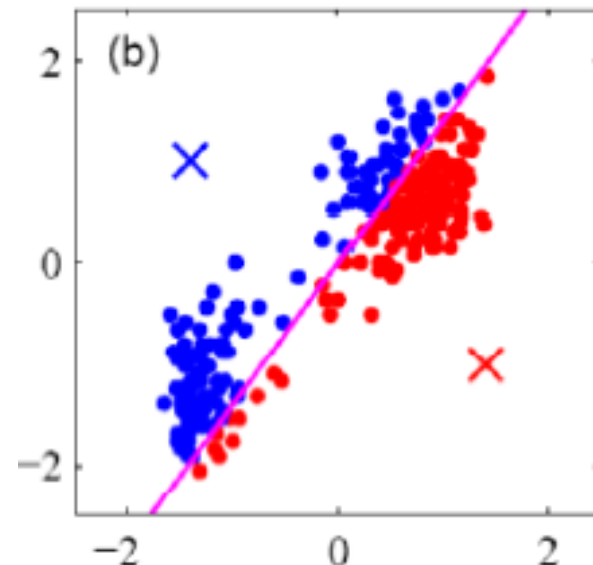
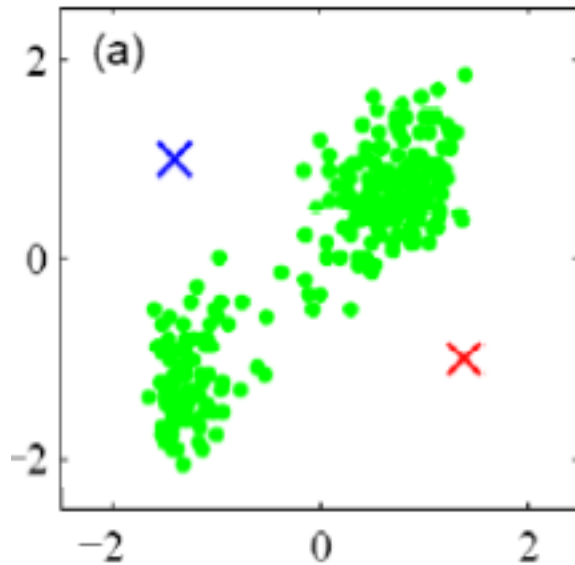
kmeans

- Workflow



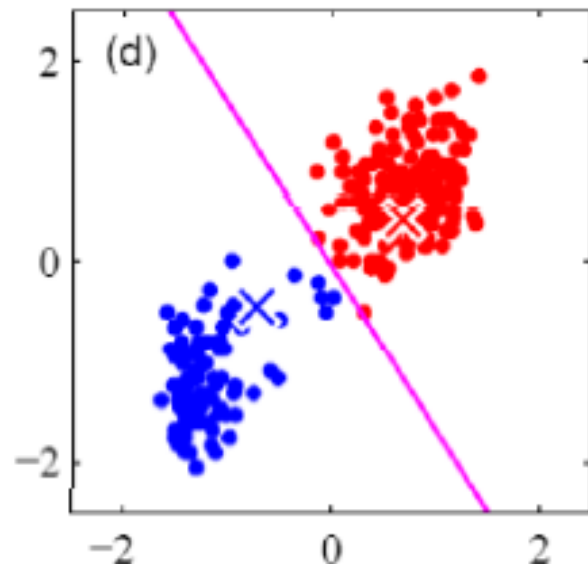
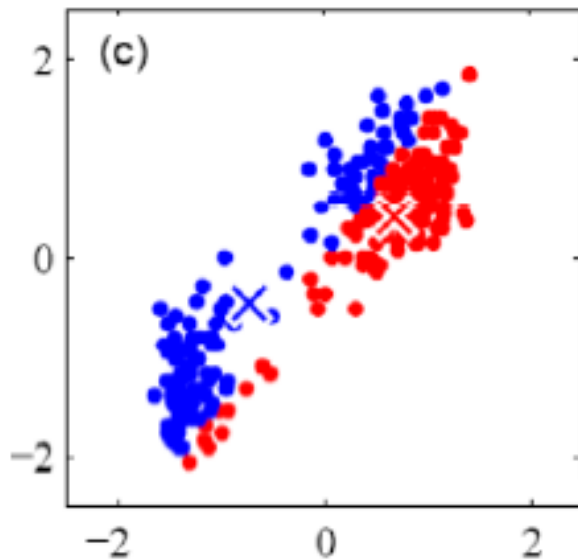
Example (1)

- Pick k random points as cluster centers (means), Here $K=2$
- Iterative step 1: assign data points to closest cluster center



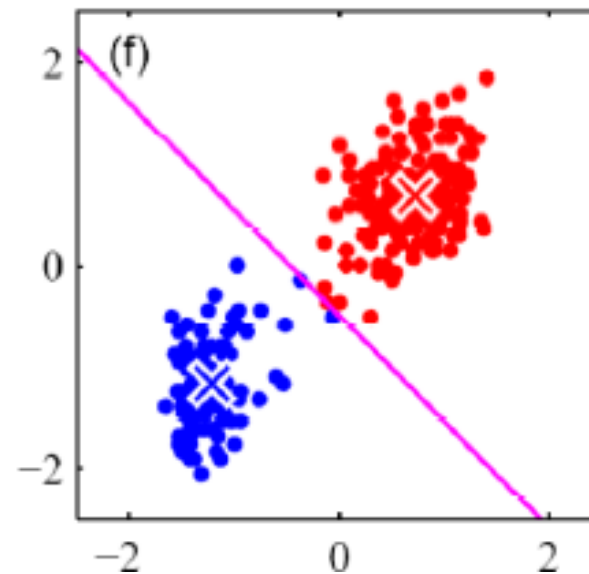
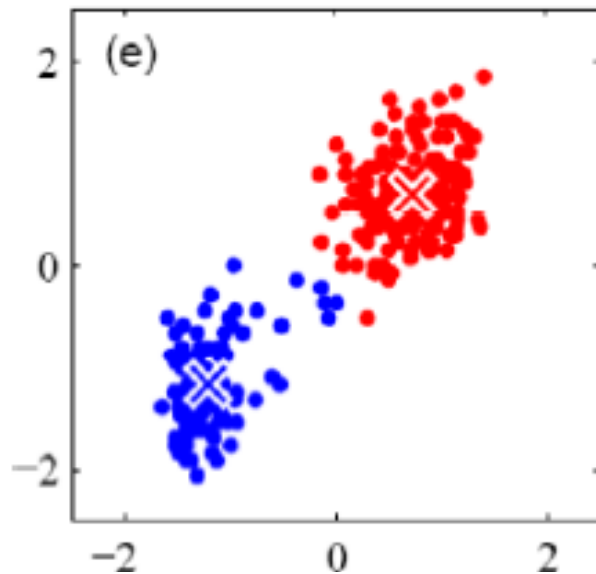
Example (1)

- Iterative step 2: change the cluster center to the average of the assigned points
- Assign data points to closest cluster center



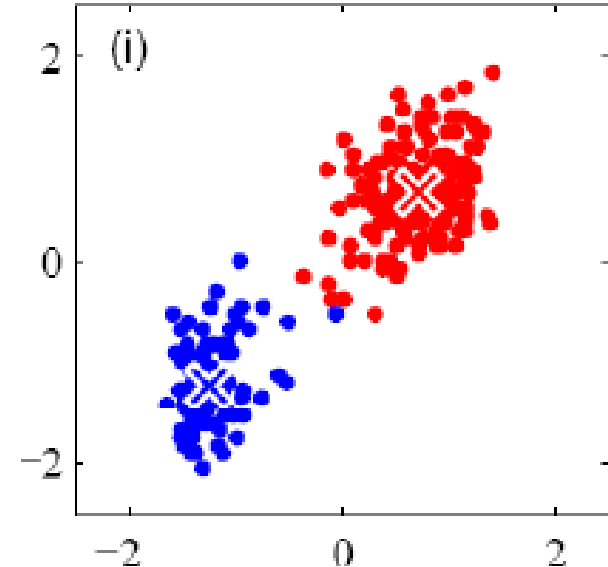
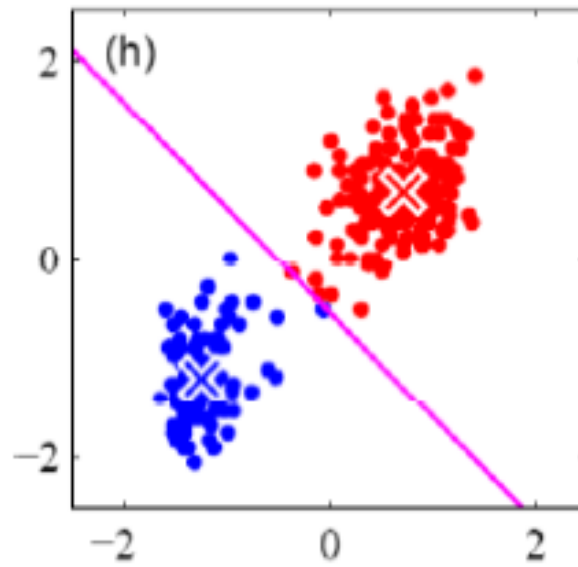
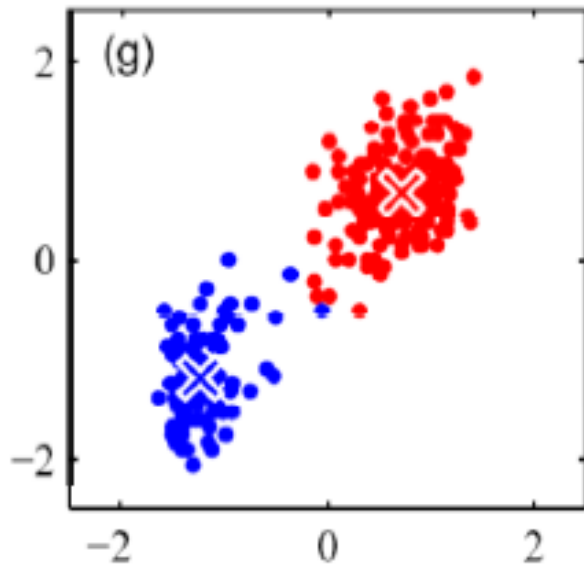
Example (1)

- Iterative step 3: change the cluster center to the average of the assigned points
- Assign data points to closest cluster center

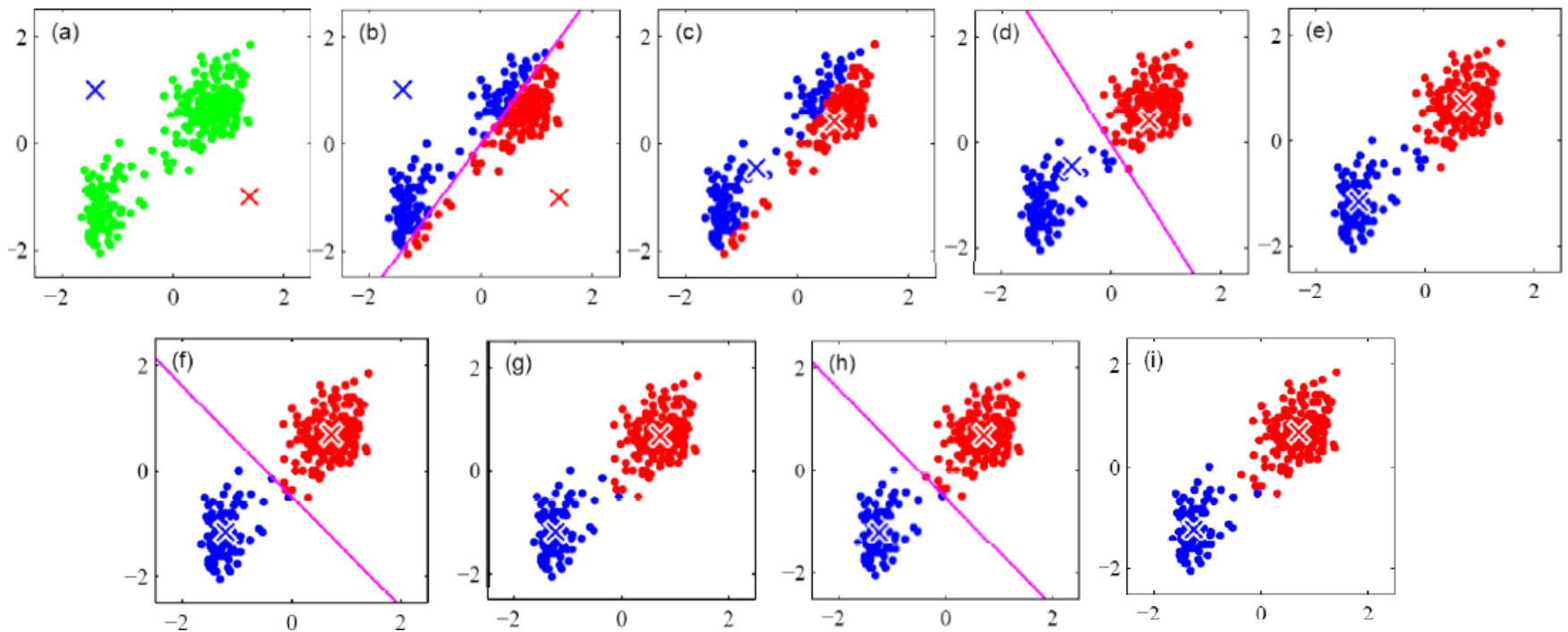


Example (1)

- Repeat until convergence



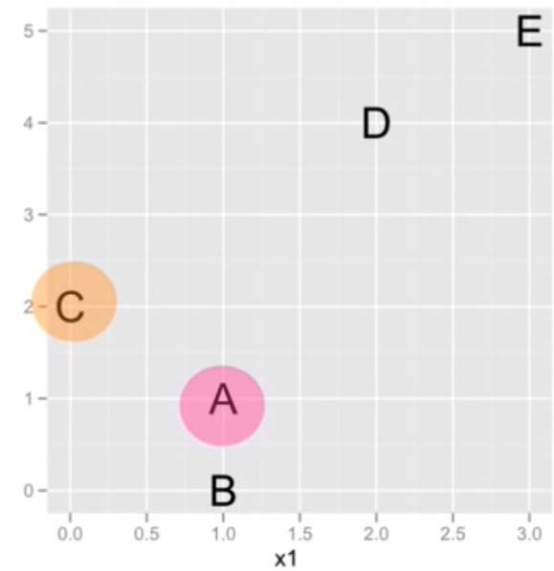
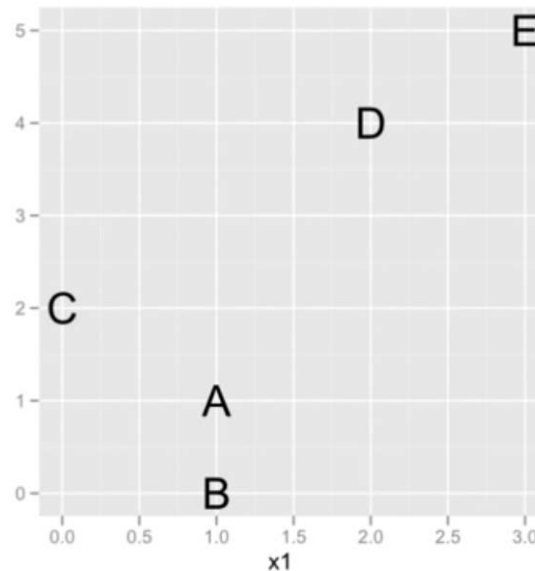
Example (1)



Example (2)

- Use $K=2$, suppose A and C are randomly selected as the initial means

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5



Example (2)

- Compute distances between each of the cluster means and all other points

$\bar{X}_1^0$ 

$\bar{X}_2^0$ 

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

i	1	2
A	0	1.4
B	1	2.2
C	1.4	0
D	3.2	2.8
E	4.5	4.2

Example (2)

- Assign each case to the cluster having the closest mean.
Recalculate the cluster means.

i	1	2	Cluster
A	0	1.4	1
B	1	2.2	1
C	1.4	0	2
D	3.2	2.8	2
E	4.5	4.2	2

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

● $\bar{X}_1^1 = (1, 0.5)$

● $\bar{X}_2^1 = (1.7, 3.7)$

Example (2)

- Compute distances between each of the cluster means and all other points

i	X ₁	X ₂
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

● $\bar{X}_1^1 = (1, 0.5)$

● $\bar{X}_2^1 = (1.7, 3.7)$

i	1	2
A	0.5	2.7
B	0.5	3.7
C	1.8	2.4
D	3.6	0.5
E	4.9	1.9

Example (2)

- Assign each case to the cluster having the closest mean. Recalculate the cluster means.
- Algorithm has converged-recalculating distances, reassigning cases to cluster results in no change. This is the final solution

i	1	2	Cluster
A	0.5	2.7	1
B	0.5	3.7	1
C	1.8	2.4	1
D	3.6	0.5	2
E	4.9	1.9	2

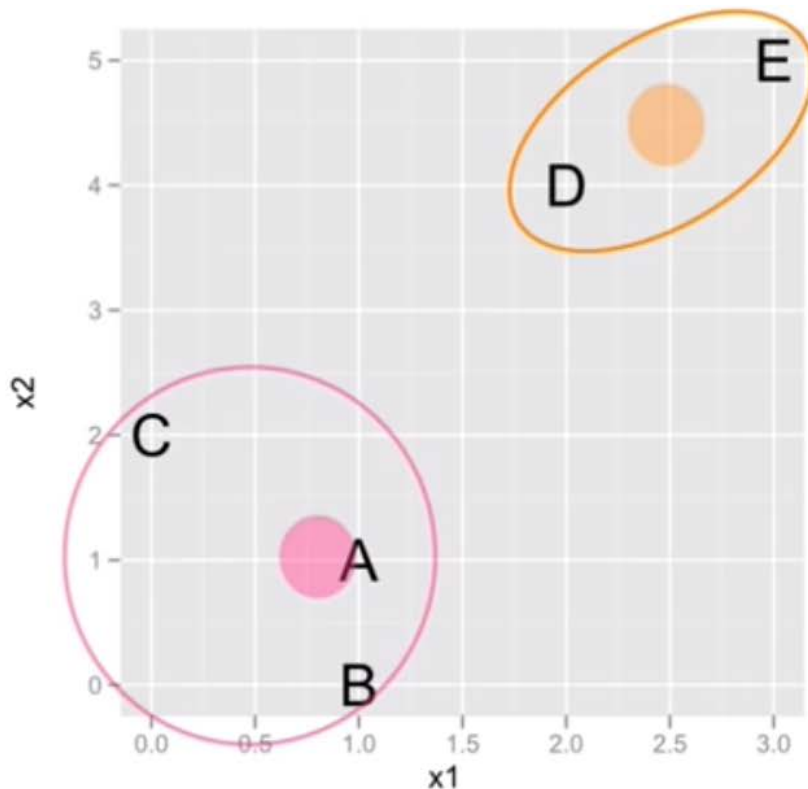
i	X ₁	X ₂
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

● $\bar{X}_1^2 = (0.7, 1)$

● $\bar{X}_2^2 = (2.5, 4.5)$

Example (2)

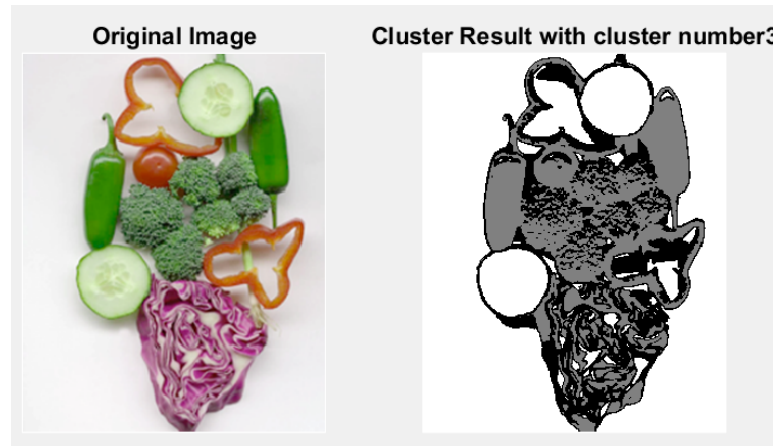
- Algorithm has converged-recalculating distances, reassigning cases to clusters results in no change, this is the final solution.



● $\bar{X}_1^2 = (0.7, 1)$

● $\bar{X}_2^2 = (2.5, 4.5)$

Example (3)



Example (3)

```
close all
clear
clc

%%initialize
cluster_num =3; %choose how many colors to segment;

I = imread('prob1.png');
I2 = rgb2ycbcr(I);
ycbcr = double(I2( : , : , 1:3));
nrows = size(I2,1);
ncols = size(I2,2);
ab = reshape(ycbcr,nrows*ncols,3);
ab=ab';

cluster_image=zeros(3,nrows*ncols);
segmented_images = cell(1,cluster_num);
%%%% Using Kmeans homework to get the cluster result
[cluster_idx cluster_center] = kmeans_function(ab,cluster_num,0);
pixel_labels = reshape(cluster_idx,nrows,ncols); %reshape the result matrix in order to make it be
rgb_label = repmat(pixel_labels,[1 1 3]); %reconstruct RGB space;
```

Example (3)

```
function [class_idx,center_next]=kmeans_function(X,cluster_num,weight)

%X is the data samples
%cluster_num is the number of the clusters
%weight is used as the combination of the color information and space information

[row,N]=size(X);
if weight~=0
    X_first=weight*X(1:3,:);
    X_second=(1-weight)*X(3+1:row,:);
    X=[X_first;X_second];
end

%initialize
center_pre=ones(row,cluster_num);
center_next=ones(row,cluster_num);
d=ones(1,cluster_num);
Class=cell(cluster_num,1);
cluster_info=ones(cluster_num,N+1);

%select the original center randomly
cluster_info(:,2)= randsrc(cluster_num,1,[1:N]);
center_pre=X(:,cluster_info(:,2));

while 1
    clear cluster_info Class;
    for k=1:cluster_num
        cluster_info(k,1)=1;
    end

    %for all points, assign it to the centers
    for t=1:N
        for k=1:cluster_num
            d(k)=(X(:,t)-center_pre(:,k))'*(X(:,t)-center_pre(:,k));
        end
        j=find(d==min(d));
        cluster_info(j,1)=cluster_info(j,1)+1;
        cluster_info(j,cluster_info(j,1))=t;
    end

    %compute the points in the cluster
    for k=1:cluster_num
        Class{k}=ones(row,cluster_info(k,1)-1);
        for j=2:1:cluster_info(k,1)
            Class{k}(:,j-1)=X(:,cluster_info(k,j));
        end
    end

    %recompute the center of the cluster
    for k=1:cluster_num
        center_next(:,k)=mean(Class{k},'');
    end

    %if the new cluster and the cluster of the last time is similar,
    %then the result is satisfied, and we can get the cluster result
```

Example (4)

Suppose A, D are randomly selected as initial means. Illustrate first two iterations of Kmeans method by the following data input with $K=2$.

i	X1	X2
A	1	1
B	2	0
C	0	1
D	2	2
E	1	5
G	3	4

Example (4)

A	0	$\sqrt{2}$	1(A)
B	$\sqrt{2}$	2	1(A)
C	1	$\sqrt{5}$	1(A)
D	$\sqrt{2}$	0	2(D)
E	4	$\sqrt{10}$	2(D)
G	$\sqrt{13}$	$\sqrt{5}$	2(D)

New data center

1 (1, 2/3)

2 (2, 11/3)

Iteration 2

i	D1	D2	Cluster
A	1/3	$\sqrt{73}/3$	1
B	$\sqrt{13}/3$	11/3	1
C	$\sqrt{10}/3$	10/3	1
D	5/3	5/3	1 or 2
E	13/3	5/3	2
G	$\sqrt{136}/3$	$\sqrt{10}/3$	2

New data center

1 (5/4, 1)

2 (2, 9/2)

OR

1 (1, 2/3)

2 (2, 11/3)

K-means

■ Advantages

- Easy for implementation and high-speed performance
- Measurable and efficient in large data collection

■ Disadvantages

- Selection of optimal number of clusters is difficult
- Selection of the initial centroids is random, and different initializations lead to different results.

K-means

- The solution found is locally optimal, but not globally optimal
- Because the solution depends on the initial assignment of instances to clusters, random restarts will give different solutions

