TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

CS231. Nhập môn Thị giác máy tính

Image segmentation

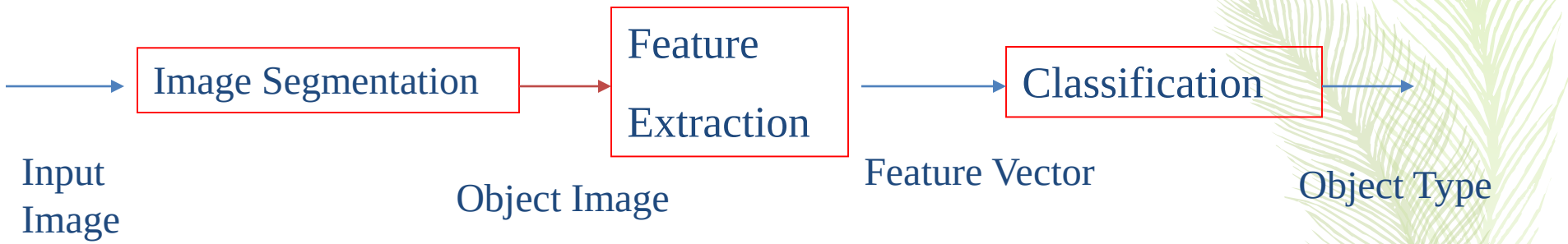
Mai Tiến Dũng

Image Segmentation

- Image segmentation is the operation of partitioning an image into a collection of connected sets of pixels.
- Segmentation in easy words is assigning labels to pixels. All picture elements or pixels belonging to the same category have a common label assigned to them.

Image Segmentation

- a



Approaches in Image Segmentation

1. Similarity approach
2. Discontinuity approach

Approaches in Image Segmentation

1. Similarity approach:

- Based on detecting similarity between image pixels to form a segment, based on a threshold.
- ML algorithms like clustering are based on this type of approach to segment an image.

Approaches in Image Segmentation

2. Discontinuity approach:

- This approach relies on the discontinuity of pixel intensity values of the image.
- Line, Point, and Edge Detection techniques use this type of approach for obtaining intermediate segmentation results which can be later processed to obtain the final segmented image.

Groups of image segmentation

- Semantic segmentation
- Instance segmentation

Groups of image segmentation

- Semantic segmentation is an approach detecting, for every pixel, belonging class of the object.
- For example, when all people in a figure are segmented as one object and background as one object.

Groups of image segmentation

- Instance segmentation is an approach that identifies, for every pixel, a belonging instance of the object. It detects each distinct object of interest in the image. For example, when each person in a figure is segmented as an individual object.

Groups of image segmentation

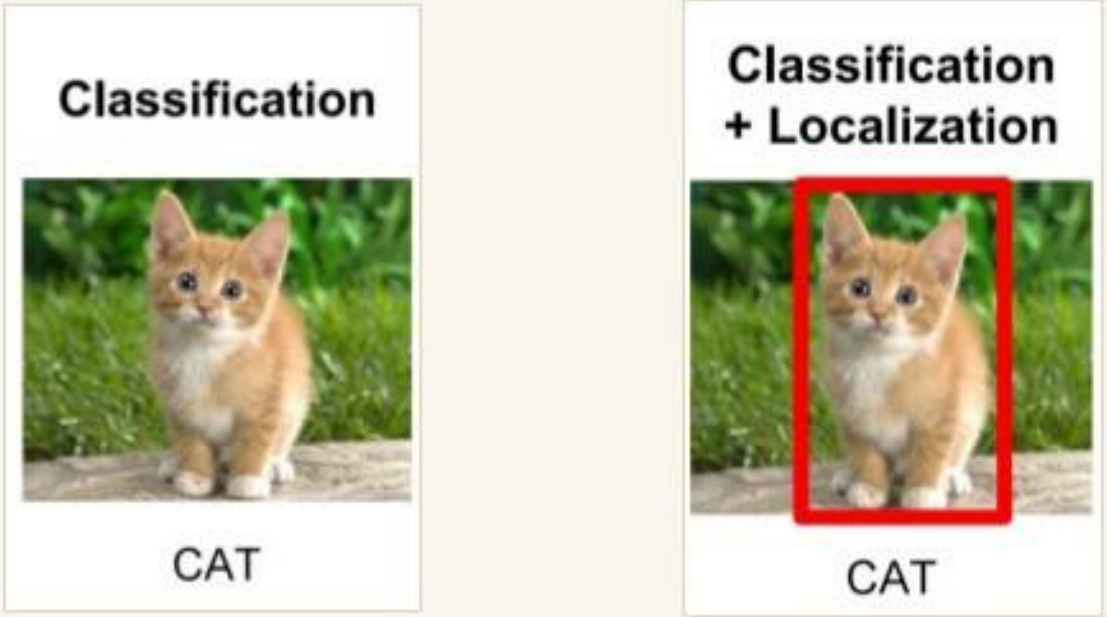
Image Classification: A core task in Computer Vision

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

cat

Groups of image segmentation

Classification + Localization

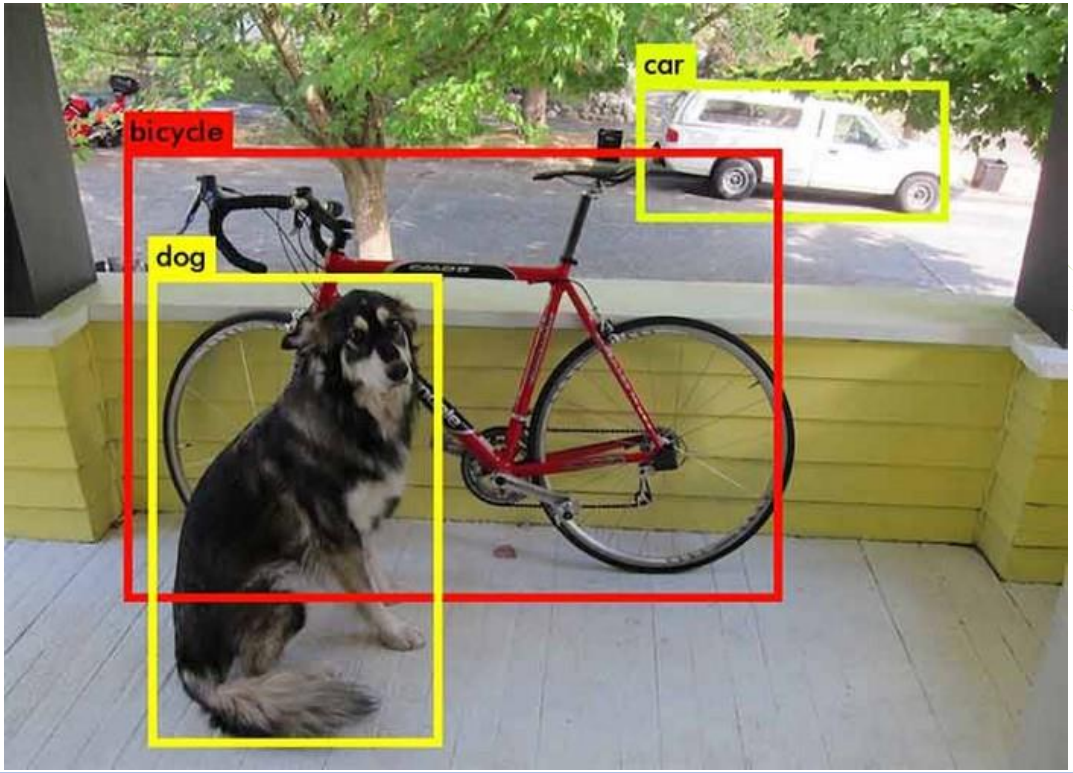


http://cs231n.stanford.edu/syllabus.html

Groups of image segmentation

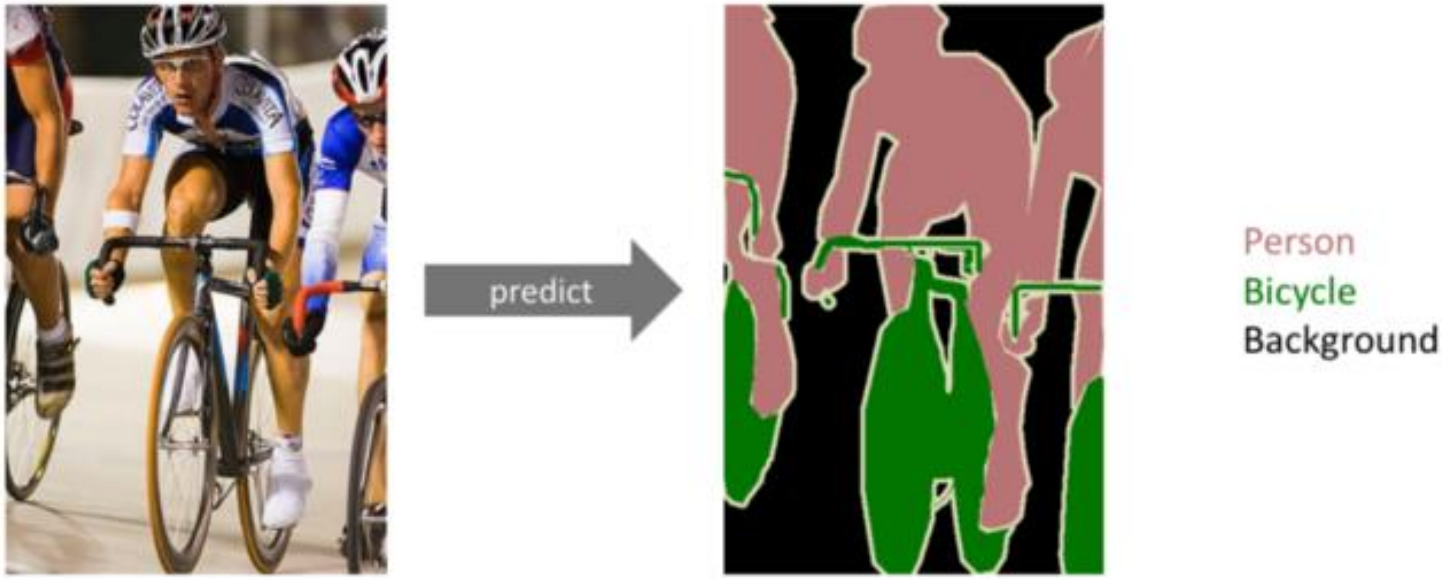
- Object Detection

→ to classify and localize all the objects in the image



Groups of image segmentation

- Semantic Segmentation



- The goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented. Because we're predicting for every pixel in the image, this task is commonly referred to as dense prediction.

Groups of image segmentation

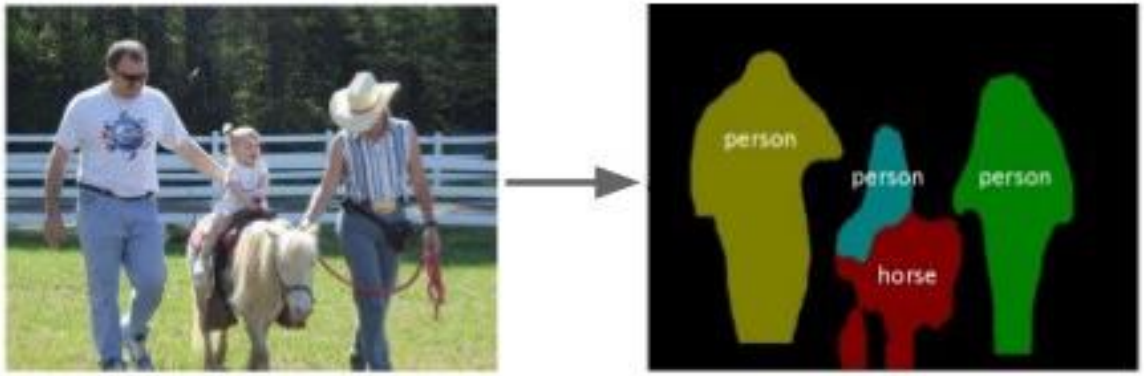
- Instance segmentation

→ to classify each instance of a class separately

Detect instances, give category, label pixels

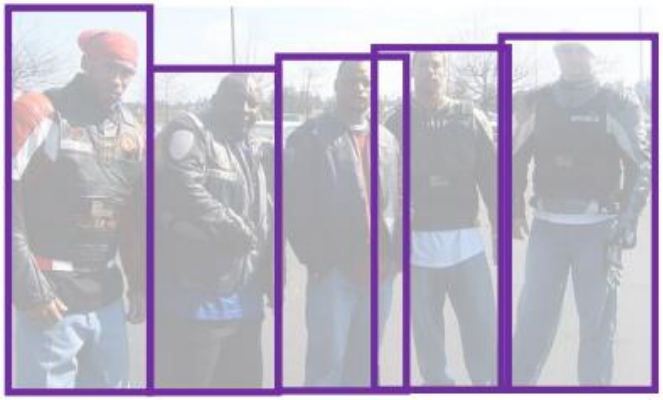
"simultaneous detection and segmentation" (SDS)

Label are class-aware and instance-aware

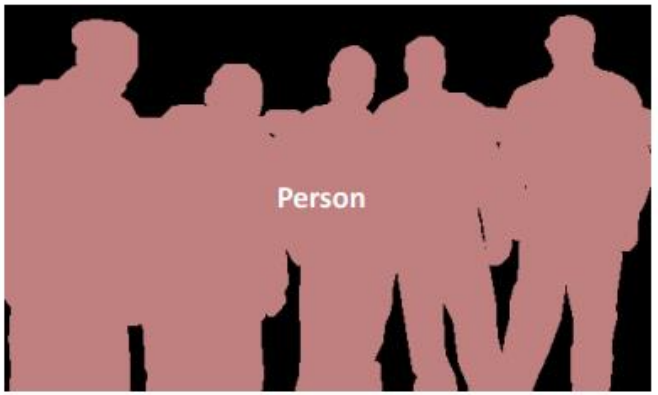


Slide Credit: CS231n

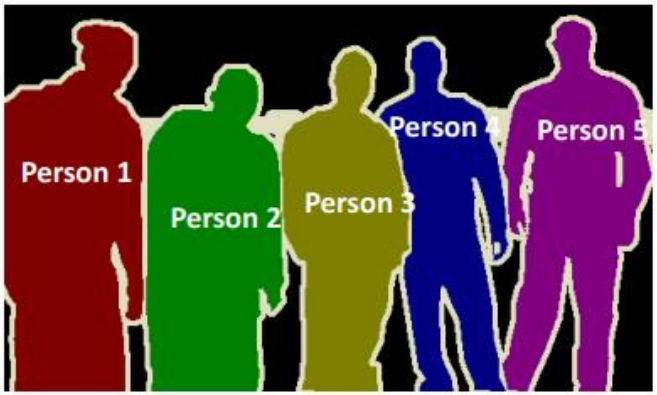
Groups of image segmentation



Object Detection



Semantic Segmentation



Instance Segmentation

Image Segmentation Techniques

1. Threshold Based Segmentation
2. Edge Based Segmentation
3. Region-Based Segmentation
4. Clustering Based Segmentation
5. Artificial Neural Network Based Segmentation

1. THRESHOLD BASED SEGMENTATION

Threshold Based Segmentation

- To create a binary or multi-color image based on setting a threshold value on the pixel intensity of the original image.

- How to find an automatic threshold ?
  → considering variations in illumination & surface
- One popular approach : histogramming

Threshold Based Segmentation

- Consider the intensity histogram of all the pixels in the image → then we will set a threshold to divide the image into sections.

Thresholding techniques

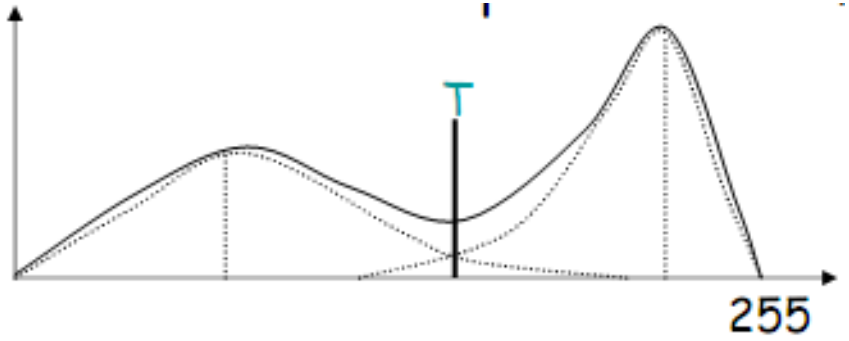
1. Global thresholding
2. Manual thresholding
3. Adaptive Thresholding
4. Optimal Thresholding
5. Local Adaptive Thresholding

Thresholding techniques: Global

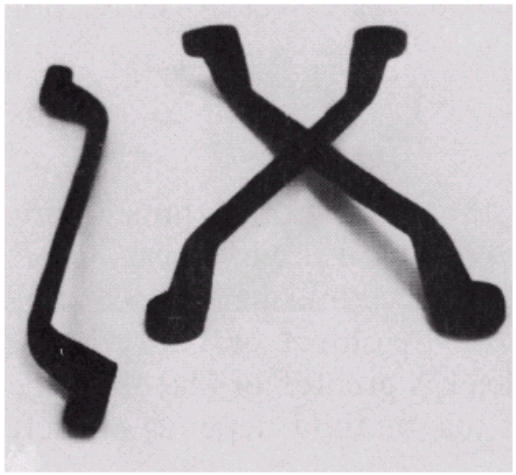
- Use a bimodal image → with 2 peaks of intensities in the intensity distribution plot.
  - One for the object and
  - One for the background.
- Deduce the global threshold value for the whole image.
- A disadvantage of this type of threshold is that it performs really poorly during poor illumination in the image.

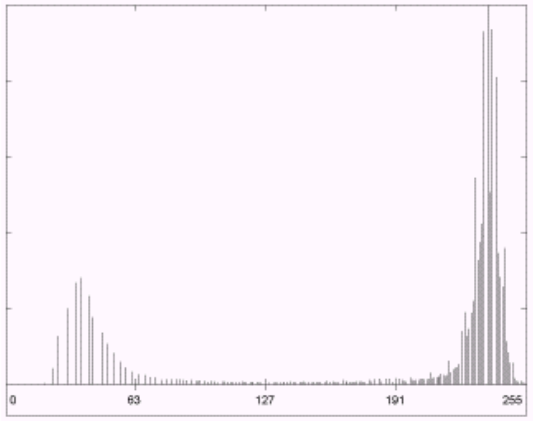
Thresholding techniques: Global

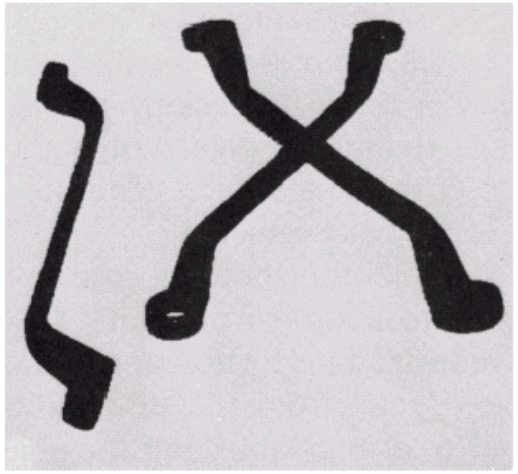
- Find the “peaks” and “valleys” of the histogram
- Set threshold to the pixel value of the “valley”



- Non-trivial to find peaks/valleys :
  - ignore local peaks; choose peaks at a distance find the valley between those peaks
  - Maximize “peakiness” (difference btw peaks & valleys) to find the threshold as valley







a
b c

FIGURE 10.28
(a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

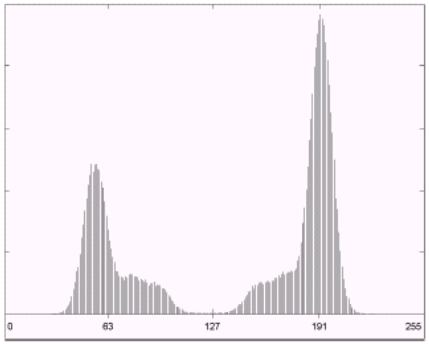
Thresholding techniques: Manual

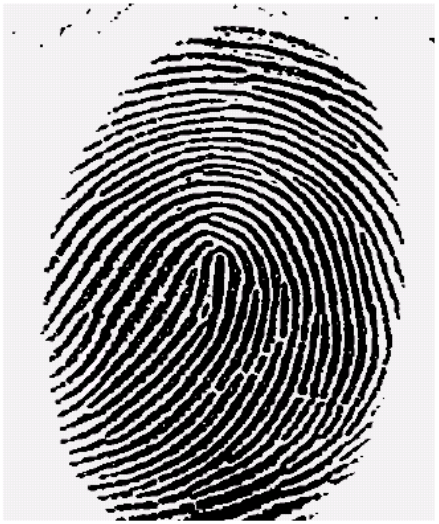
- Iterative threshold selection
- The following process goes as follows

a. Choose the initial threshold value.
b. Segment the image into 2 regions G1 and G2 using the threshold value.
c. Find the mean of pixels of region of region G1 and G2 (μ_1 and μ_2 respectively).
d. T = (μ_1 + μ_2) / 2 and then go back to step 2.

Continue the process the $| T_i - T_{i+1} | <= T'$, Where T' is some value defined by the user.

Thresholding techniques: Manual





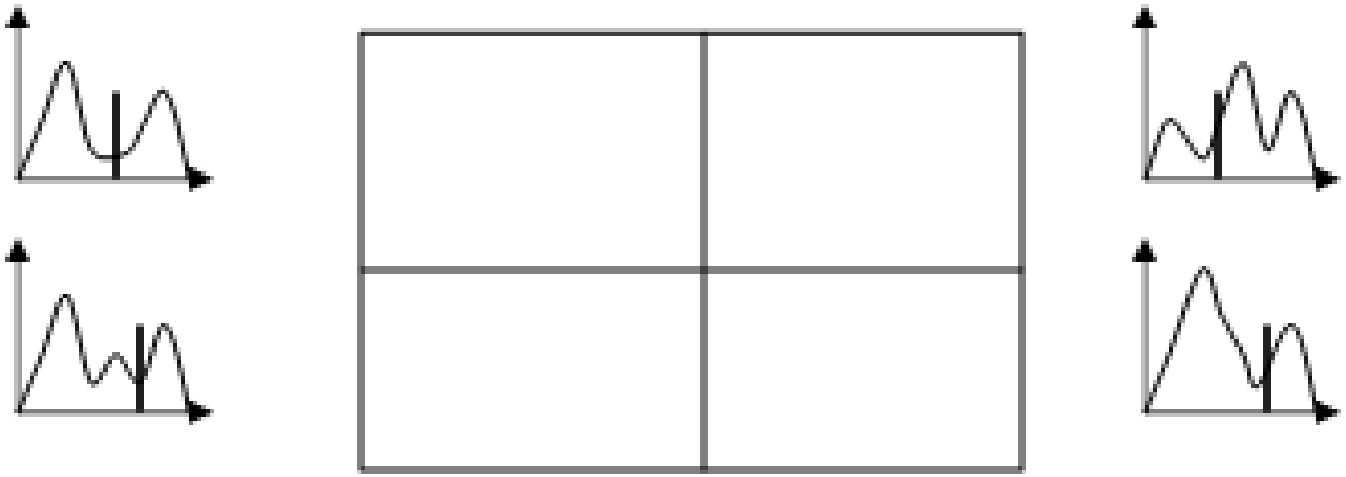
a b
c

FIGURE 10.29
(a) Original image. (b) Image histogram. (c) Result of segmentation with the threshold estimated by iteration. (Original courtesy of the National Institute of Standards and Technology.)

Thresholding techniques: Adaptive

- To overcome the effect of illumination, the image is divided into various subregions, and all these regions are segmented using the threshold value calculated for all these regions. Then these subregions are combined to image the complete segmented image.
- This helps in reducing the effect of illumination to a certain extent.

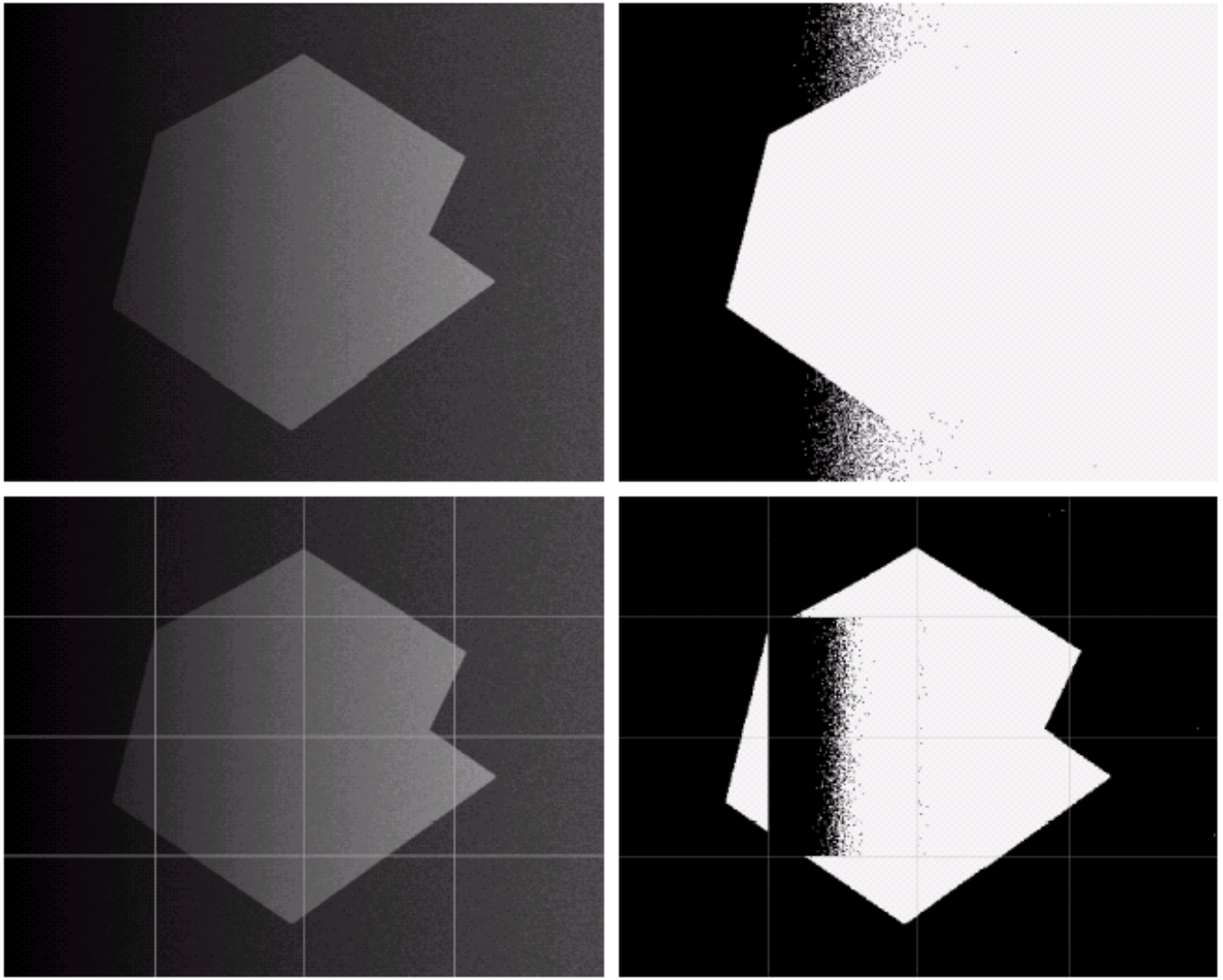
Thresholding techniques: Adaptive



Thresholding techniques: Adaptive

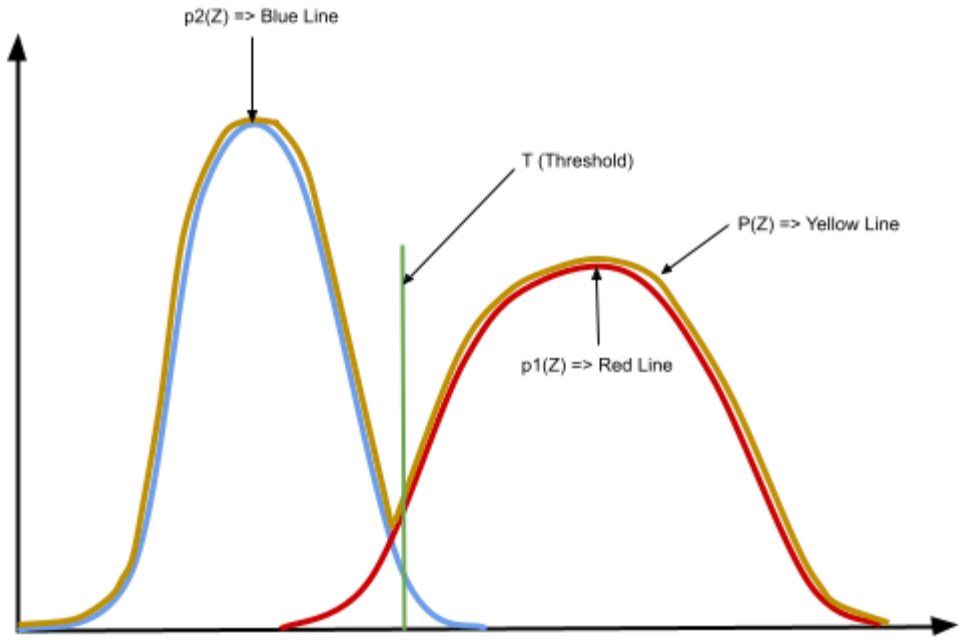
a b
c d

FIGURE 10.30
(a) Original image. (b) Result of global thresholding. (c) Image subdivided into individual subimages. (d) Result of adaptive thresholding.



Thresholding techniques: Optimal

- Optimal thresholding technique can be used to minimize the misclassification of pixels performed by segmentation.



Thresholding techniques: Optimal

- The probability of a pixel value is given by the following formulae :

$$P(Z) = P_1 \cdot p_1(Z) + P_2 \cdot p_2(z) \Rightarrow P1 + P2 = 1$$

$$p_1(Z) = PDF\ of\ background\ pixels$$

$$p_2(Z) = PDF\ of\ object\ pixels$$

Thresholding techniques: Local Adaptive

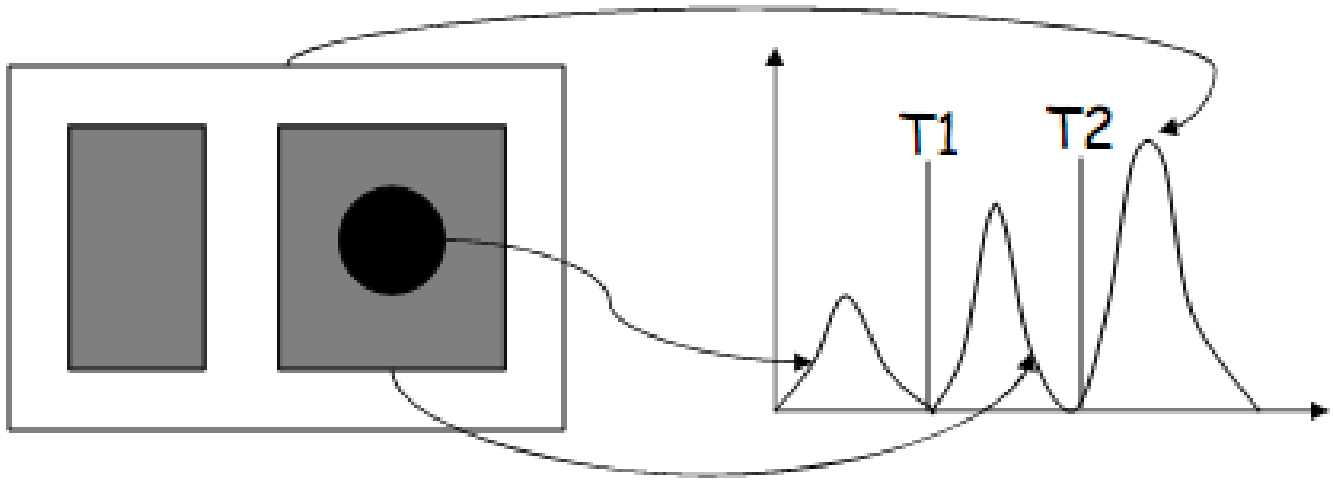
- Due to variation in the illumination of pixels in the image, global thresholding might have difficulty in segmenting the image.
- Hense the image is divided into smaller subgroups and then adaptive thresholding of those individual groups is done.

Thresholding techniques: Local Adaptive

- After individual segmentation of these subgroups, all of them are combined to form the completed segmented image of the original image.
- Hence, the histogram of subgroups helps in providing better segmentation of the image.

Thresholding techniques: Double

- Starting from a conservative initial threshold T1, determine the “core” parts of the object
- Continuing from this core part, grow this object by including neighboring pixels which are between T1 and T2



Thực hành 1 – Baitap5

- Áp dụng phân đoạn ảnh cho các ảnh:

1. Chọn ngưỡng: global
   - particles.bmp
   - Phandoan01.jpg
   - wdg2.jpg
   - Rice.png
2. Chọn ngưỡng: Adaptive
   - Ảnh wdg3.jpg

```
def drawHist(x):
  hist = cv.calcHist([x], [0], None, [256], [0, 256])
  plt.plot(hist)
  plt.xlim([0, 256])
  plt.legend(('histogram'), loc = 'upper left')
  plt.show()
```

```
def globalThresholding(img, thres=127):
  img_rst = img.copy()
  for i in range(img.shape[0]):
    for j in range(img.shape[1]):
      if img_rst[i][j] < thres:
        img_rst[i][j] = 255
      else:
        img_rst[i][j] = 0
  return img_rst
```

```
def adaptiveThresholding(f, nrow, ncol):
  g = f.copy()
  r = int(f.shape[0] / nrow)
  c = int(f.shape[1] / ncol)
  for i in range(int(nrow)):
    for j in range(int(ncol)):
      x = f[i * r : (i + 1) * r, j * c : (j + 1) * c]
      cv2_imshow(x)
      drawHist(x)
      #t = np.average(x)
       t = int(input())
      g[i * r : (i + 1) * r, j * c : (j + 1) * c] = globalThres
holding(x, t)
  return g
```

Edge Based Segmentation

Edge Based Segmentation

- Rely on edges found in an image using various edge detection operators.
- These edges mark image locations of discontinuity in gray levels, color, texture, etc. When we move from one region to another, the gray level may change. So if we can find that discontinuity, we can find that edge.

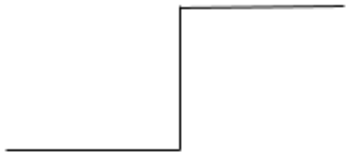
Edge Based Segmentation

- A variety of edge detection operators are available but the resulting image is an intermediate segmentation result and should not be confused with the final segmented image.
- We have to perform further processing on the image to the segment it.

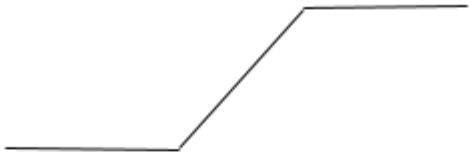
Edge Based Segmentation

- Additional steps include combining edges segments obtained into one segment in order to reduce the number of segments rather than chunks of small borders which might hinder the process of region filling. This is done to obtain a seamless border of the object.
- The goal of edge segmentation is to get an intermediate segmentation result to which we can apply region-based or any other type of segmentation to get the final segmented image.

Types of edges



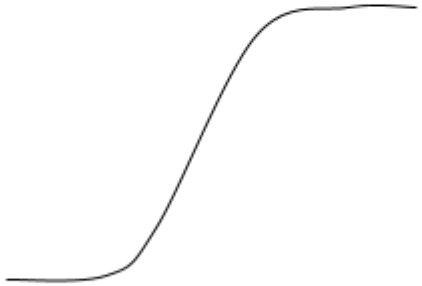
Step Edge



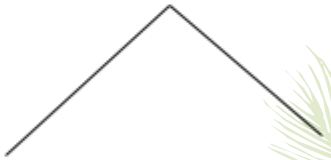
Step Ramp Edge

Continuous Ramp

Impulse Edge



Smooth Ramp Edge



Ridge Edge

- Tham khảo bài học về thông tin cạnh - edge

Clustering-Based Segmentation

Clustering-Based Segmentation

- Clustering is a type of unsupervised machine learning algorithm. It is highly used for the segmentation of images.
- One of the most dominant clustering-based algorithms used for segmentation is K-means Clustering. This type of clustering can be used to make segments in a colored image.

Partitioning Clustering

- Partitioning Clustering methods subdivides the data into k groups, where k is a number predefined by the user. For K-means Clustering which is the most popular Partitioning Cluster method

K-means

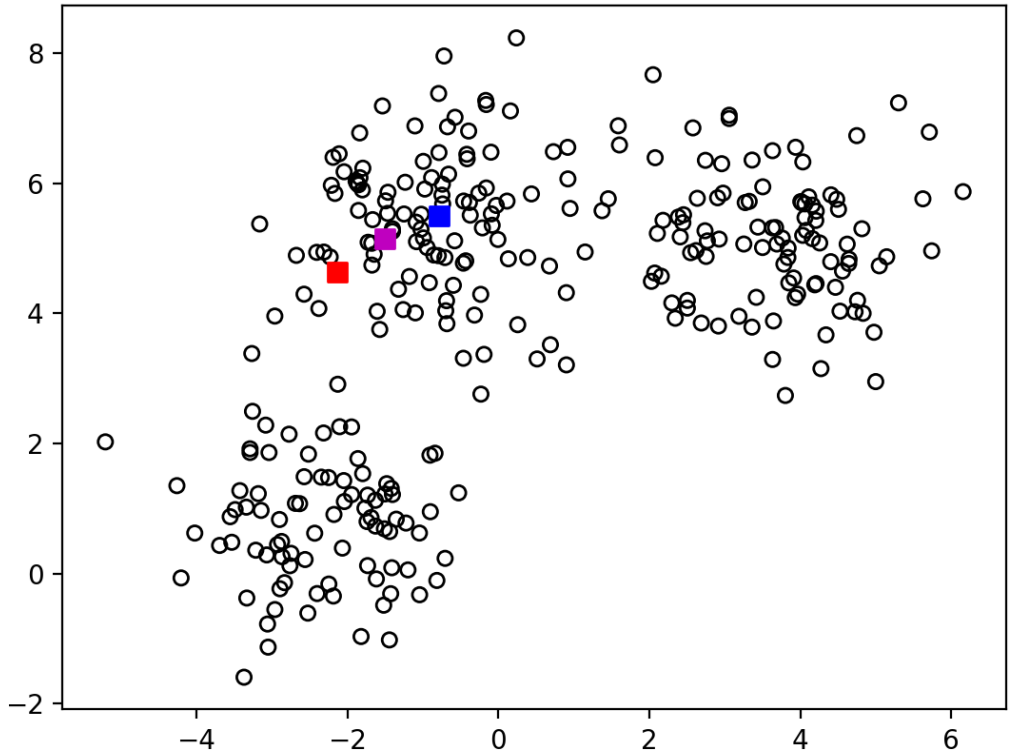
Algorithm:

1. We choose **k random points** in the data as the **center of clusters** and assign each point to **the nearest cluster by looking at the L2 distance between the point and the center**.
2. Compute the mean of each cluster, assign that mean value as the new center of the cluster.
3. Reassign each data point to its closest cluster center. Repeat step 2.

The process keeps going until **no new assignment is performed** (so the model is converged, there is nothing to go furthermore) or for a **given number of iteration**. Therefore, K-means Clustering is an iterative method where we can determine the iteration number too.

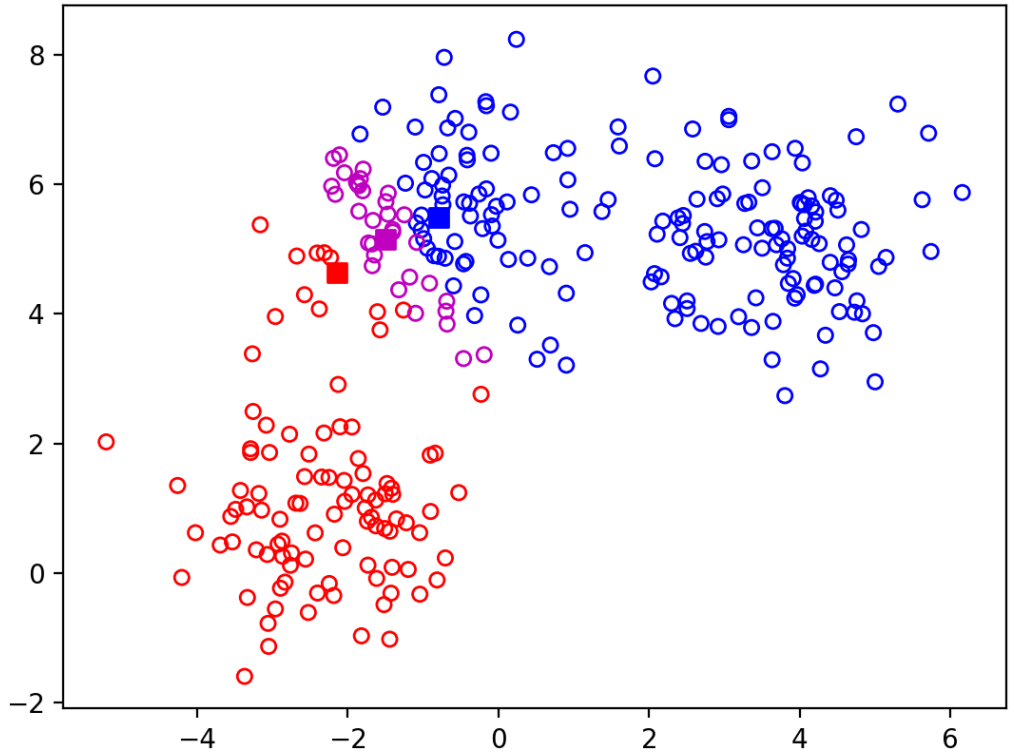
K-means

- Step-1 : randomly pick k centers



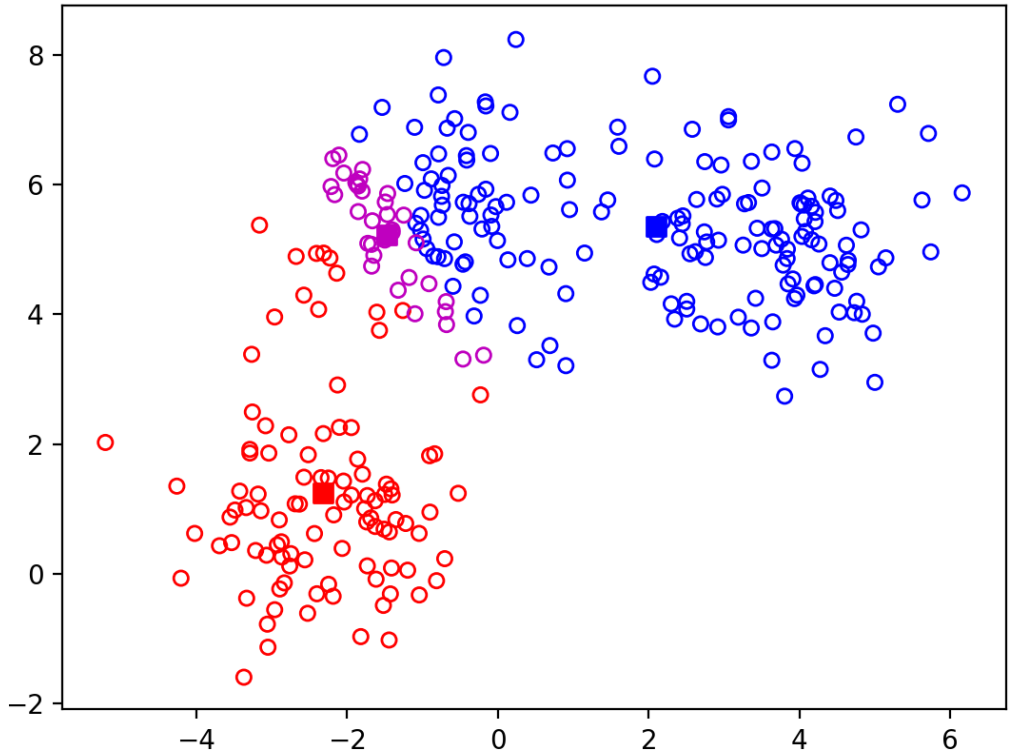
K-means

- Step 2: Assign each point to nearest center



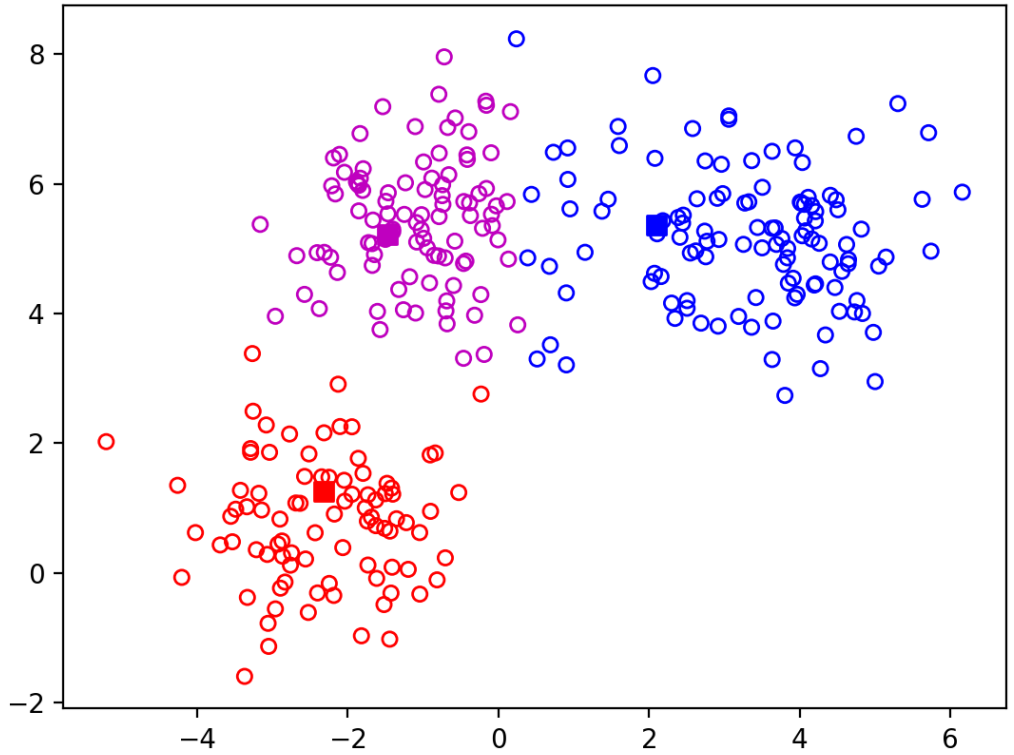
K-means

- Step 3: re-estimate centers



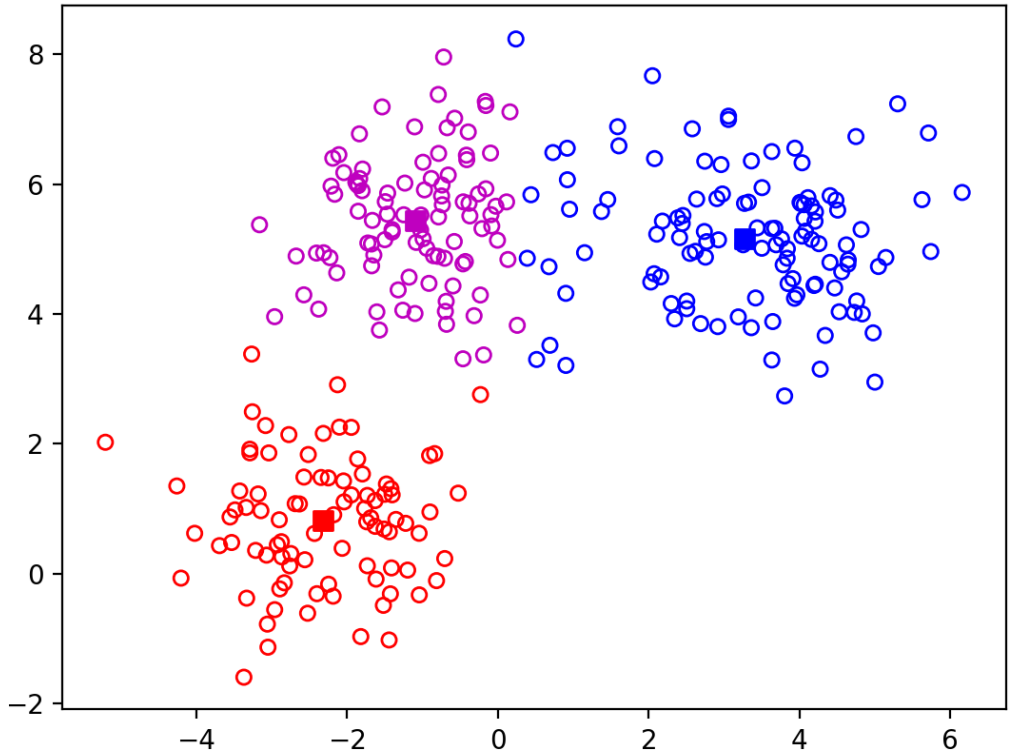
K-means

- Step 4: Repeat



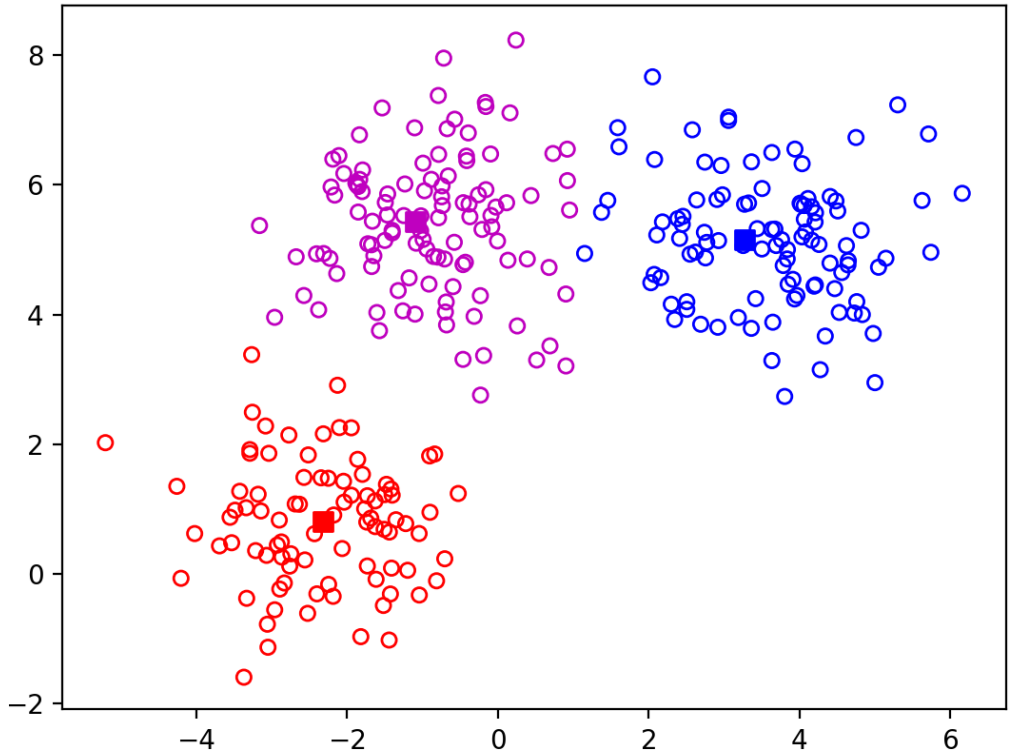
K-means

- Step 4: Repeat



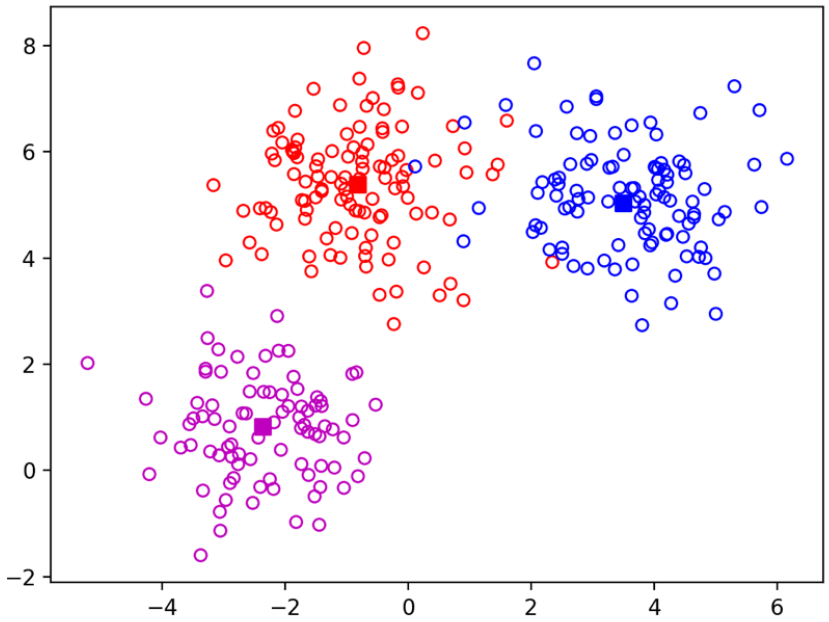
K-means

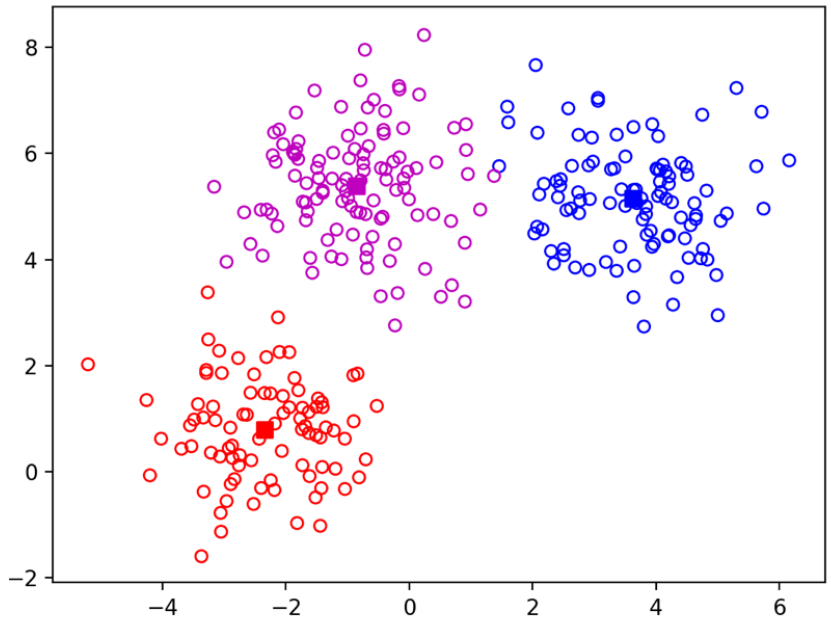
- Step 4: Repeat



K-means

- Ground-truth vs k-means



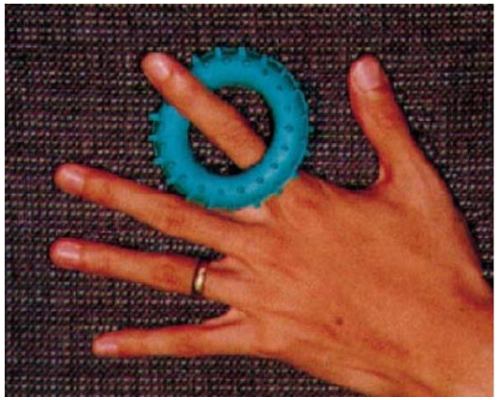


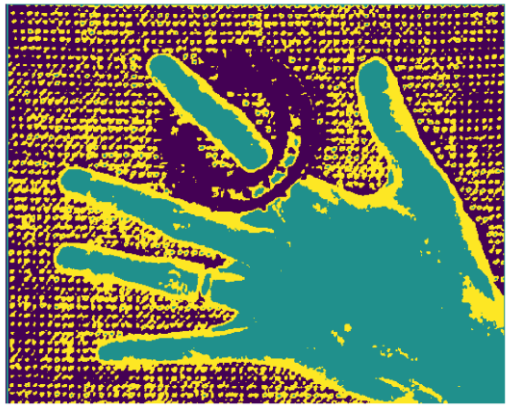
K-means on image pixels

```
path_filename = os.path.join(dir_baitap, 'vegetables.jpg')

img = cv.imread(path_filename)
cv2_imshow(img)
from sklearn.cluster import KMeans
def Kmeans(img, n_clusters = 6):
 nrow, ncol,nchl = img.shape
 g = img.reshape(nrow*ncol,nchl)
 k_means = KMeans(n_clusters = n_clusters,
                                 random_state = 0).fit(g)
 t = k_means.cluster_centers_[k_means.labels_]
 img_res = t.reshape(nrow, ncol, nchl)
 return img_res
```

K-means on image pixels



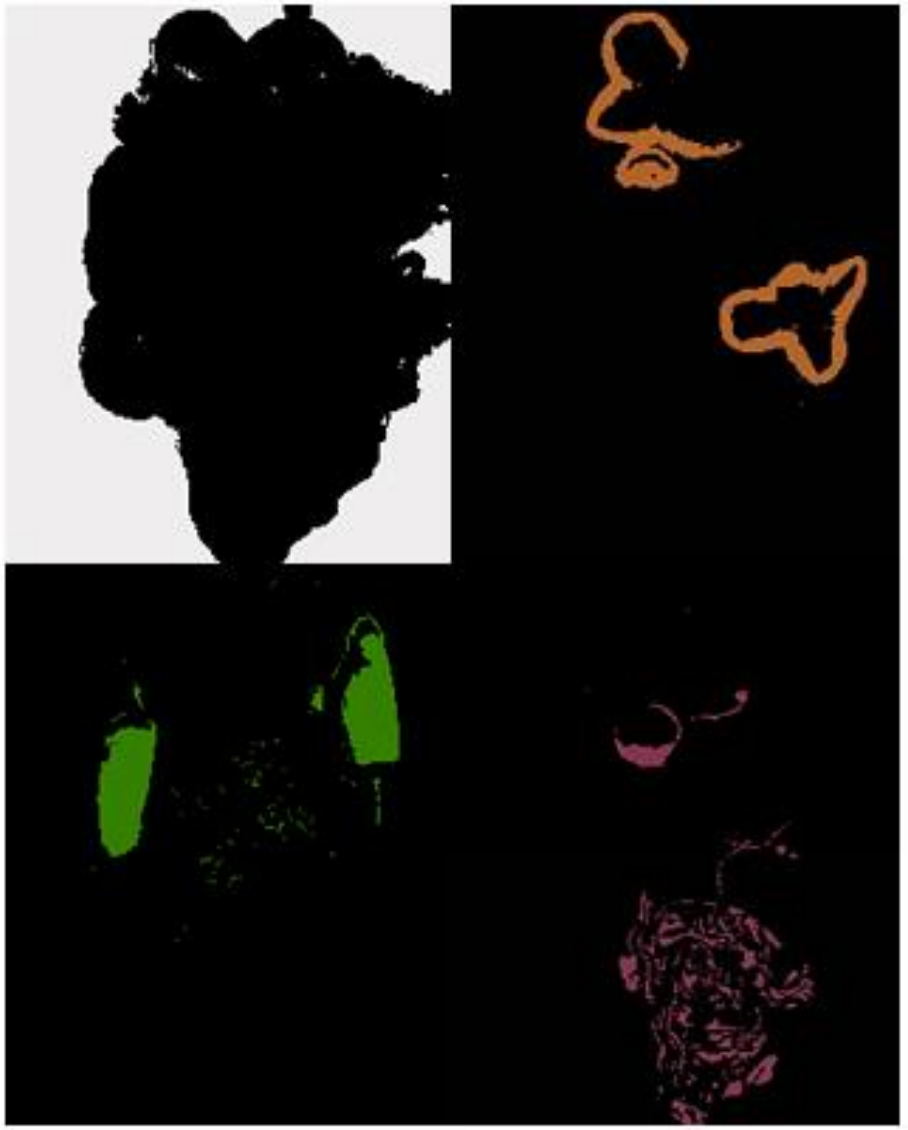


K-means on image pixels

Original Image

Clusters on color

K-means by using color (11 segments)



K-means on image pixels

Thực hành

- Representing each pixel as (r,g,b)
  - Áp dụng cho ảnh: vegetables.jpg, hand.jpg, thuoc.jpg
- Represent each pixel as (r,g,b,x,y)
  - Áp dụng cho ảnh: vegetables.jpg và thuoc.jpg

Áp dụng thuật toán k-means

```
def Kmeans2(img, n_clusters = 6):
  img_tmp = img.copy()
  nrow, ncol,nchl = img.shape

  g = []
  for y in range(nrow):
    for x in range(ncol):
      tmp = [img_tmp[y,x][0], img_tmp[y,x][1], img_tmp[y,x][2], x, y]
      g.append(tmp)

  k_means = KMeans(n_clusters=n_clusters, random_state=0).fit(g)
  # t = k_means.cluster_centers_[k_means.labels_]

  arrcolor = np.random.rand(20,3) * 255
  t = arrcolor[k_means.labels_]

  img_res = img_tmp
  i = 0
  for y in range(nrow):
    for x in range(ncol):
      img_res[y, x] = t[i][0:3]
      i += 1

  return img_res
```