CAP 4453 Robot Vision

Dr. Gonzalo Vaca-Castaño

gonzalo.vacacastano@ucf.edu

Credits

- Some slides comes directly from:
    - Yogesh S Rawat (UCF)
    - Noah Snavely (Cornell)
    - Ioannis (Yannis) Gkioulekas (CMU)
    - Mubarak Shah (UCF)
    - S. Seitz
    - James Tompkin
    - Ulas Bagci
    - L. Lazebnik

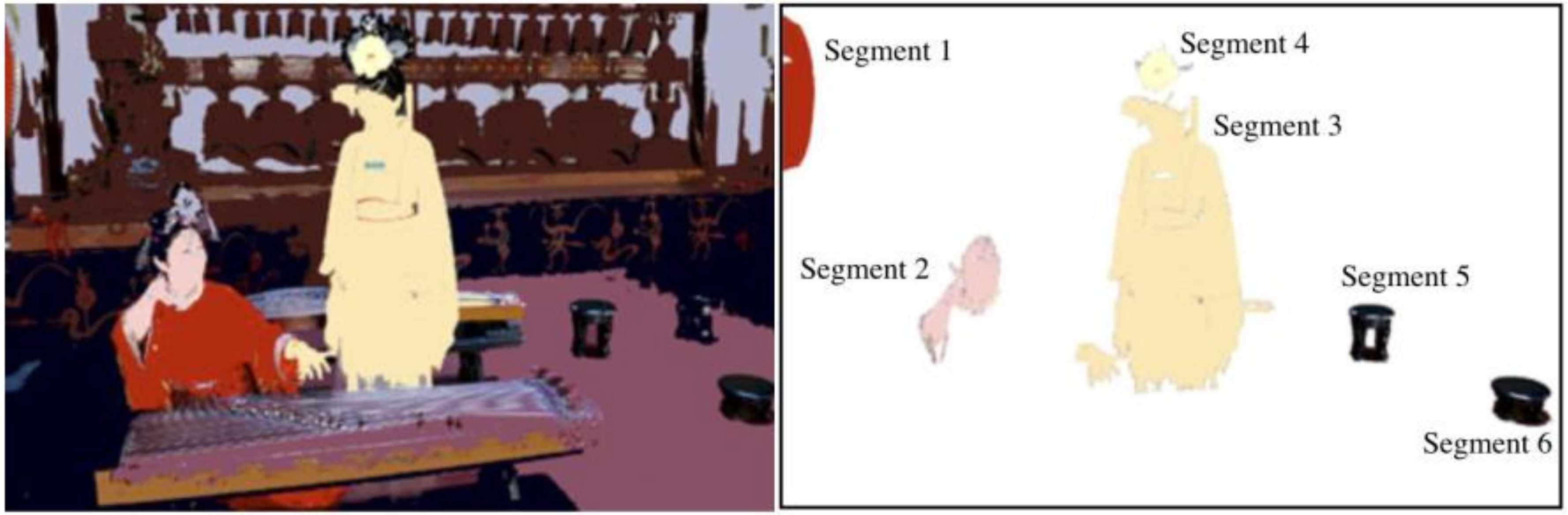
Short Review from last class

Outline

- ~~Image segmentation basics~~
- ~~Thresholding based~~
    - ~~Binarization~~
    - ~~Otsu~~
- ~~Region based~~
    - ~~Merging~~
    - ~~Splitting~~
- Clustering based
    - K-means (SLIC)

Image segmentation

- Image segmentation partitions an image into regions called segments.



- Image segmentation creates segments of connected pixels by analyzing some similarity criteria:
  - intensity, color, texture, histogram, features

Image segmentation methods

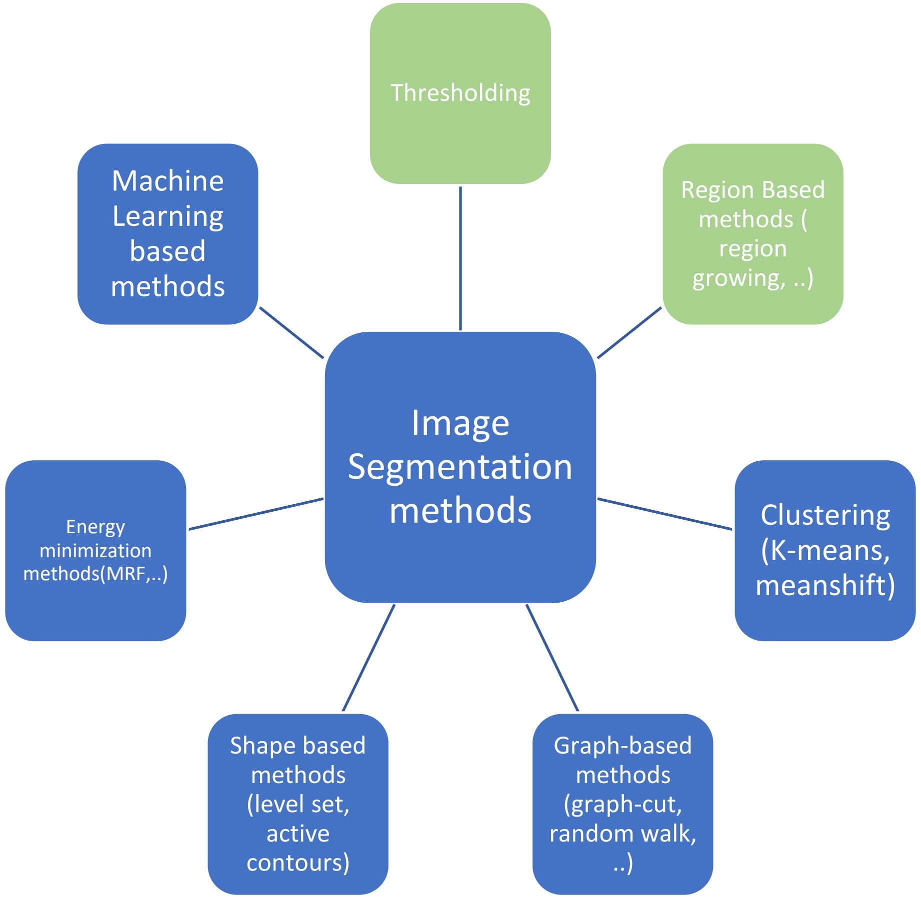


Image Segmentation methods

Thresholding

Region Based methods (region growing, ..)

Clustering (K-means, meanshift)

Graph-based methods (graph-cut, random walk, ..)

Shape based methods (level set, active contours)

Energy minimization methods(MRF,..)

Machine Learning based methods

Otsu thresholding

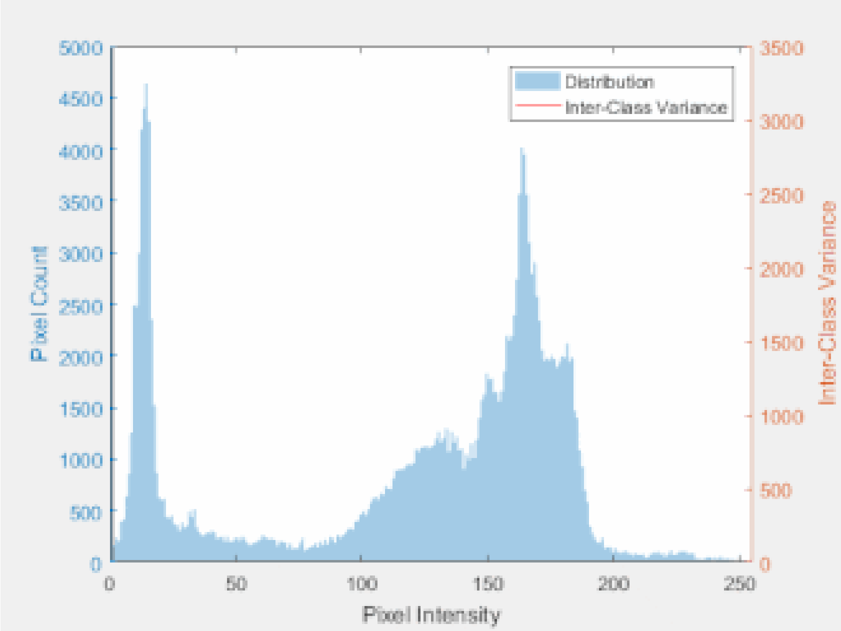
- Definition: The method uses grey-value histogram of the given image I as input and aims at providing the best threshold (foreground/background)
- Otsu’s algorithm selects a threshold that maximizes the between-class variance σ_b^2 or minimize within-class variance σ_w^2

Option 1: maximum of:

$$\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{w_1(t)} \qquad w_1(t) = \sum_{i=1}^{t} P(i)$$

$$\mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{w_2(t)} \qquad w_2(t) = \sum_{i=t+1}^{I} P(i)$$



Otsu thresholding

- Definition: The method uses grey-value histogram of the given image I as input and aims at providing the best threshold (foreground/background)
- Otsu’s algorithm selects a threshold that maximizes the between-class variance σ_b^2.

$$\sigma_{total}^2 = \underbrace{w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t)}_{\text{within-class variance}} + \underbrace{w_1(t)w_2(t)(\mu_2^2(t) - \mu_1^2(t))}_{\text{between-class variance}}$$

within-class variance $\downarrow$ Minimize

between-class variance $\downarrow$ Maximize

Option 2: minimum of:

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t)$$

$$w_1(t) = \sum_{i=1}^{t} P(i) \qquad P(i) = \frac{n_i}{n}$$

$$w_2(t) = \sum_{i=t+1}^{I} P(i)$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{w_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{w_2(t)}$$

Robot Vision

8. Segmentation II

Image segmentation methods

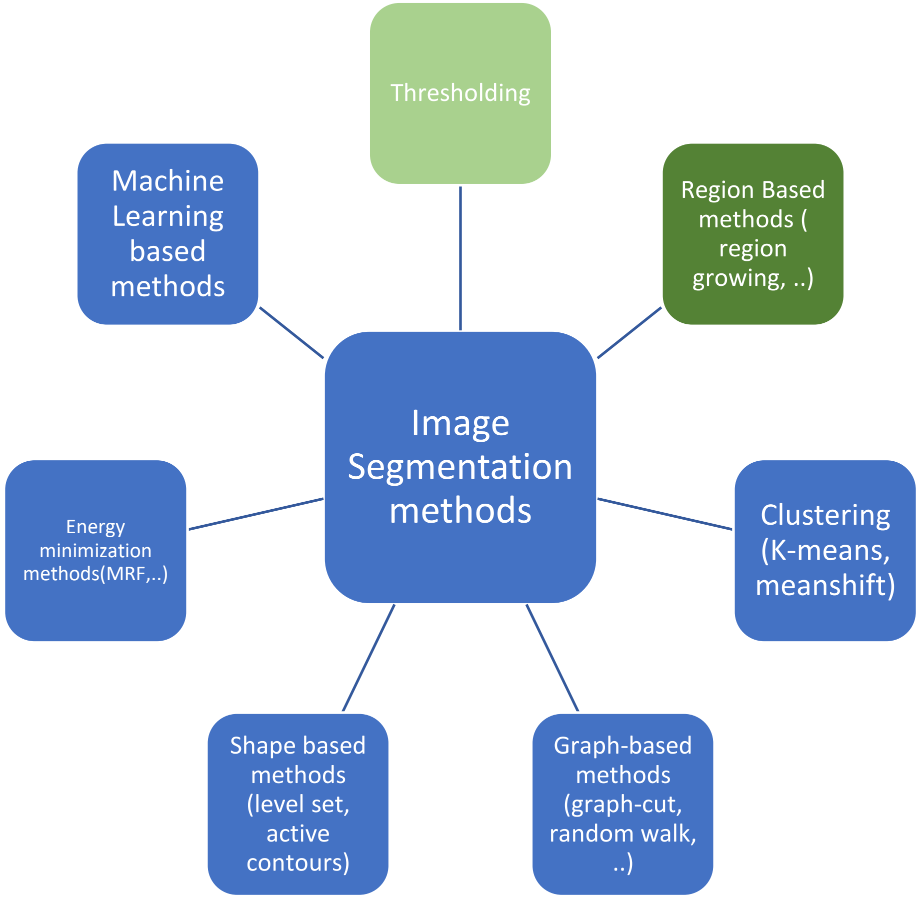


Image Segmentation methods

Thresholding

Region Based methods (region growing, ..)

Clustering (K-means, meanshift)

Graph-based methods (graph-cut, random walk, ..)

Shape based methods (level set, active contours)

Energy minimization methods(MRF,..)

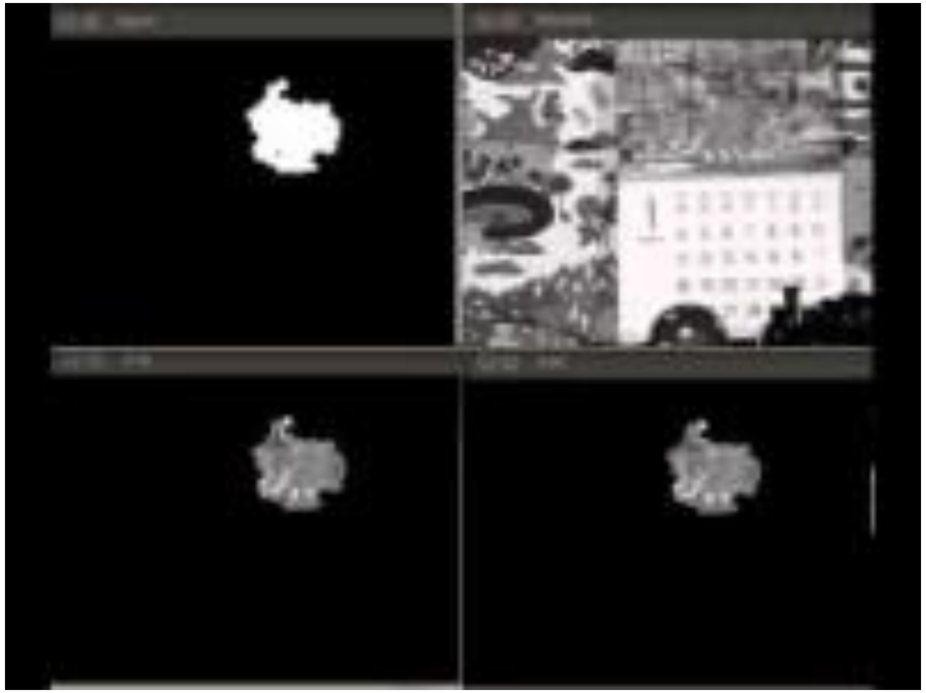
Machine Learning based methods

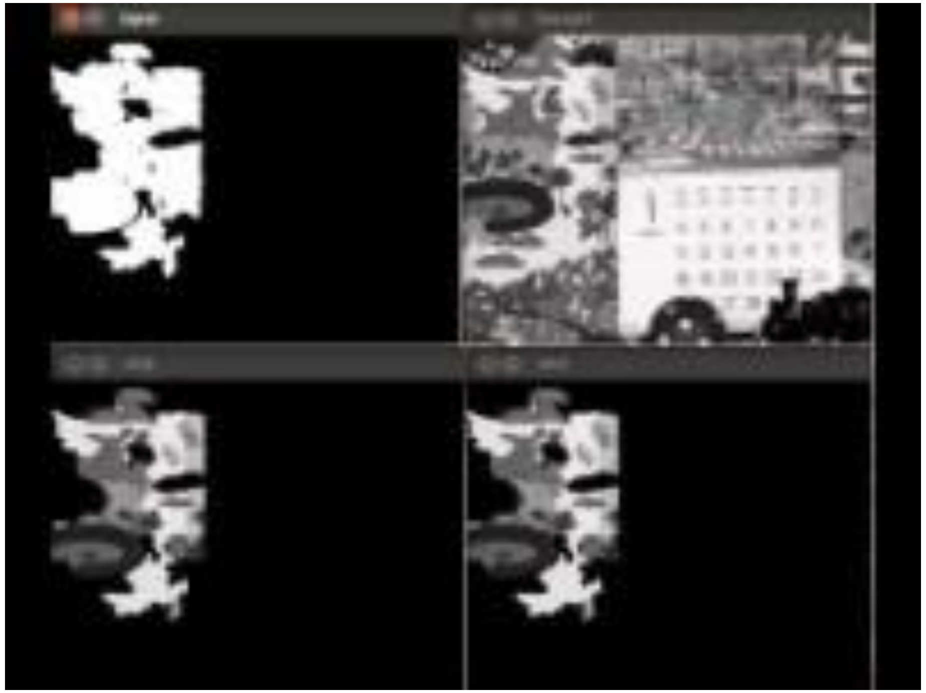
(a) (b)

(c) (d)

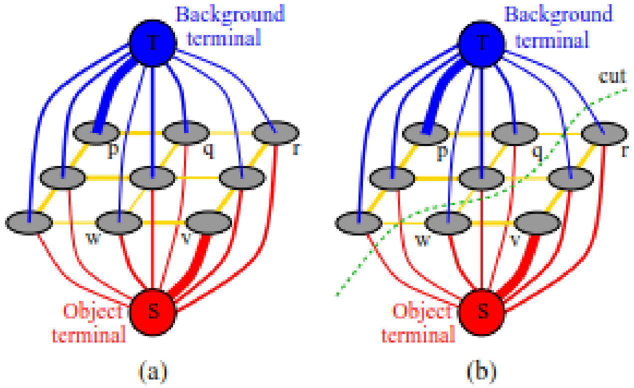
(e) (f)

Figure 5.1 Some popular image segmentation techniques: (a) active contours (Isard and Blake 1998) © 1998 Springer; (b) level sets (Cremers, Rousson, and Deriche 2007) © 2007 Springer; (c) graph-based merging (Felzenszwalb and Huttenlocher 2004b) © 2004 Springer; (d) mean shift (Comaniciu and Meer 2002) © 2002 IEEE; (e) texture and intervening contour-based normalized cuts (Malik, Belongie, Leung *et al.* 2001) © 2001 Springer; (f) binary MRF solved using graph cuts (Boykov and Funka-Lea 2006) © 2006 Springer.





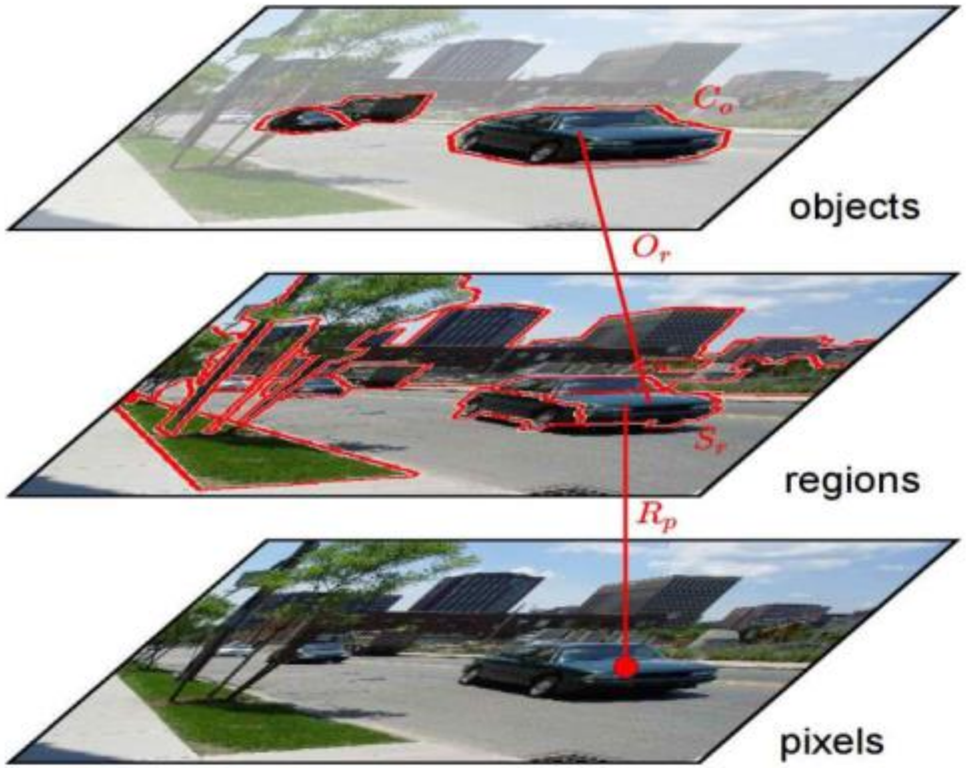
Energy-Based methods



Outline

- Image segmentation basics
- Thresholding based
  - Binarization
  - Otsu
- **Region based**
  - **Merging**
  - Splitting
- Clustering based
  - K-means (SLIC)

Region based segmentation



Region:
A group of connected pixels
with similar properties

Closed boundaries

Computation of regions is
based on similarity

Regions may correspond to Objects
in a scene or parts of objects

Spatial proximity + similarity

Region growing

- For segment generation in grey-level or color images, we may start at one seed pixel (x,y,I(x,y)) and add recursively adjacent pixels that satisfy a “similarity criterion” with pixels contained in the so-far grown region around the seed pixel.

- Defining similarity criteria alone is not an effective basis for segmentation
- It is necessary to consider the adjacency spatial relationship between pixels

Region growing

- Algorithm

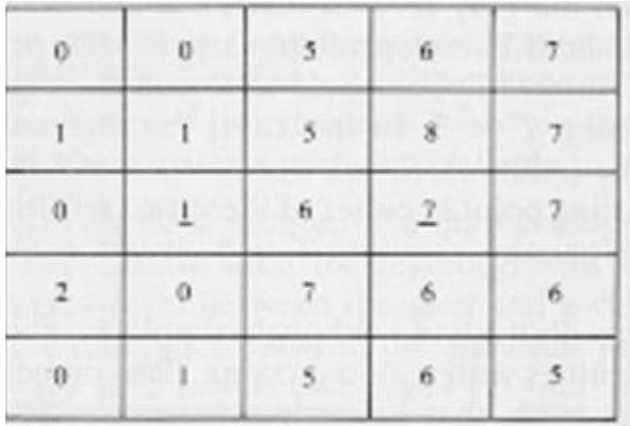
1. The absolute intensity difference between candidate pixel and the seed pixel must lie within a specified range

2. The absolute intensity difference between a candidate pixel and the running average intensity of the growing region must lie within a specified range;

3. The difference between the standard deviation in intensity over a specified local neighborhood of the candidate pixel and that over a local neighborhood of the candidate pixel must (or must not) exceed a certain threshold

Seeded segmentation. Region growing

1. Chose the seed pixel

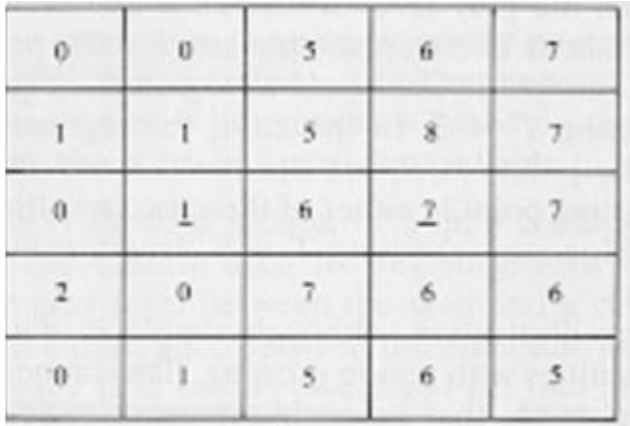


0	0	5	6	7
1	1	5	8	7
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

(a)

Seeded segmentation. Region growing

1. Chose the seed pixel
2. Check the neighboring pixels and add them to the region if they are similar to the seed

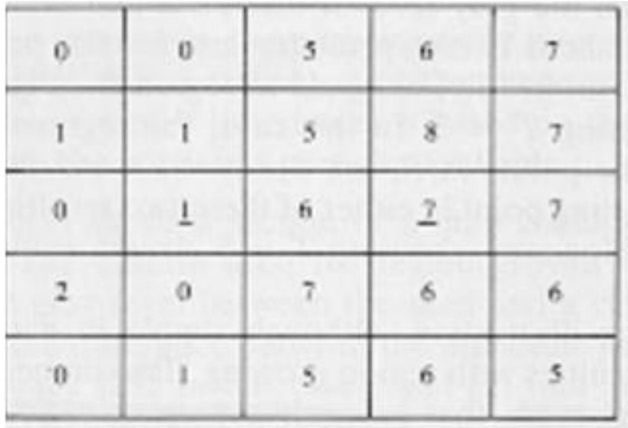


0	0	5	6	7
1	1	5	8	7
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

(a)

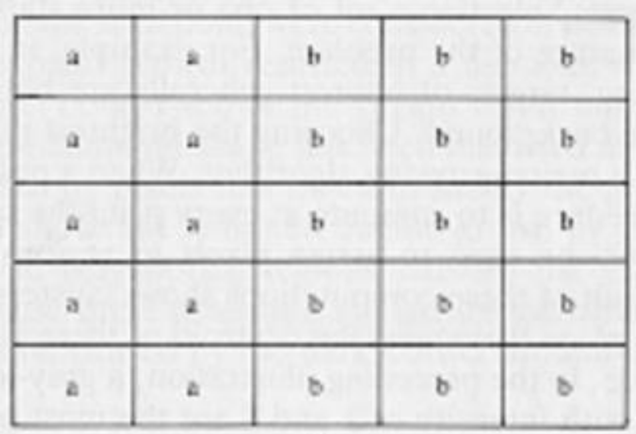
Seeded segmentation. Region growing

1. Chose the seed pixel
2. Check the neighboring pixels and add them to the region if they are similar to the seed



0	0	5	6	7
1	1	5	8	7
0	$\underline{1}$	6	$\underline{7}$	7
2	0	7	6	6
0	1	5	6	5

(a)

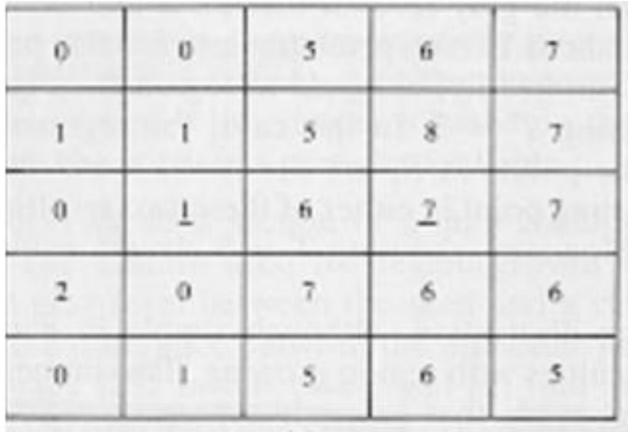


a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

(b)

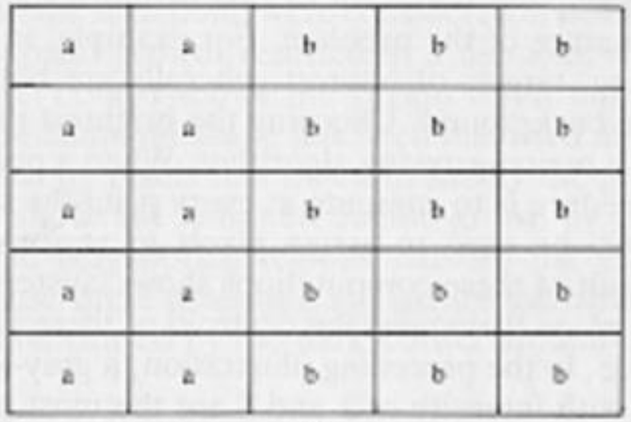
Seeded segmentation. Region growing

1. Chose the seed pixel
2. Check the neighboring pixels and add them to the region if they are similar to the seed
3. Repeat step 2 for each of the newly added pixels; stop if no more pixels can be added



0	0	5	6	7
1	1	5	8	7
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

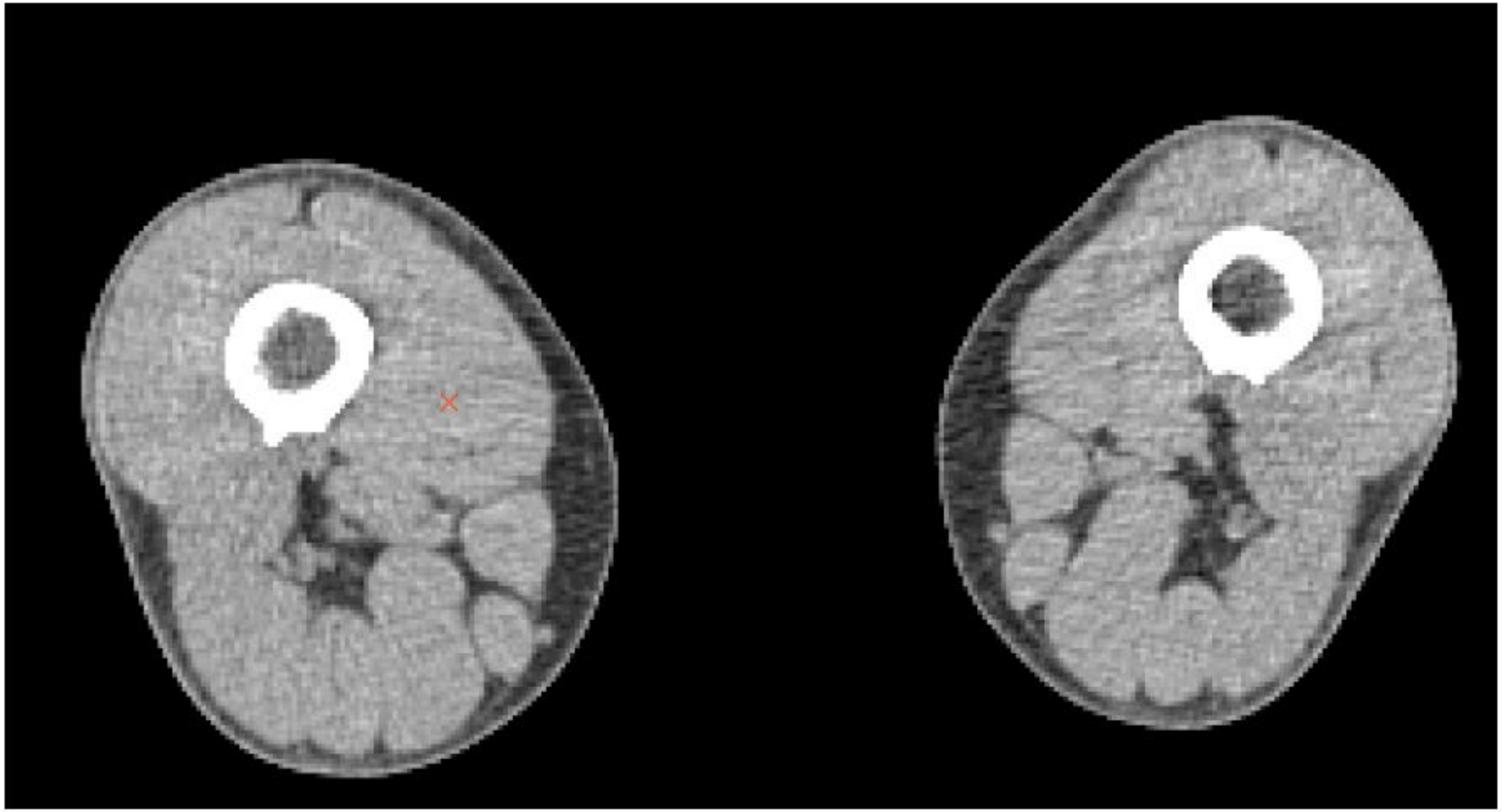
(a)



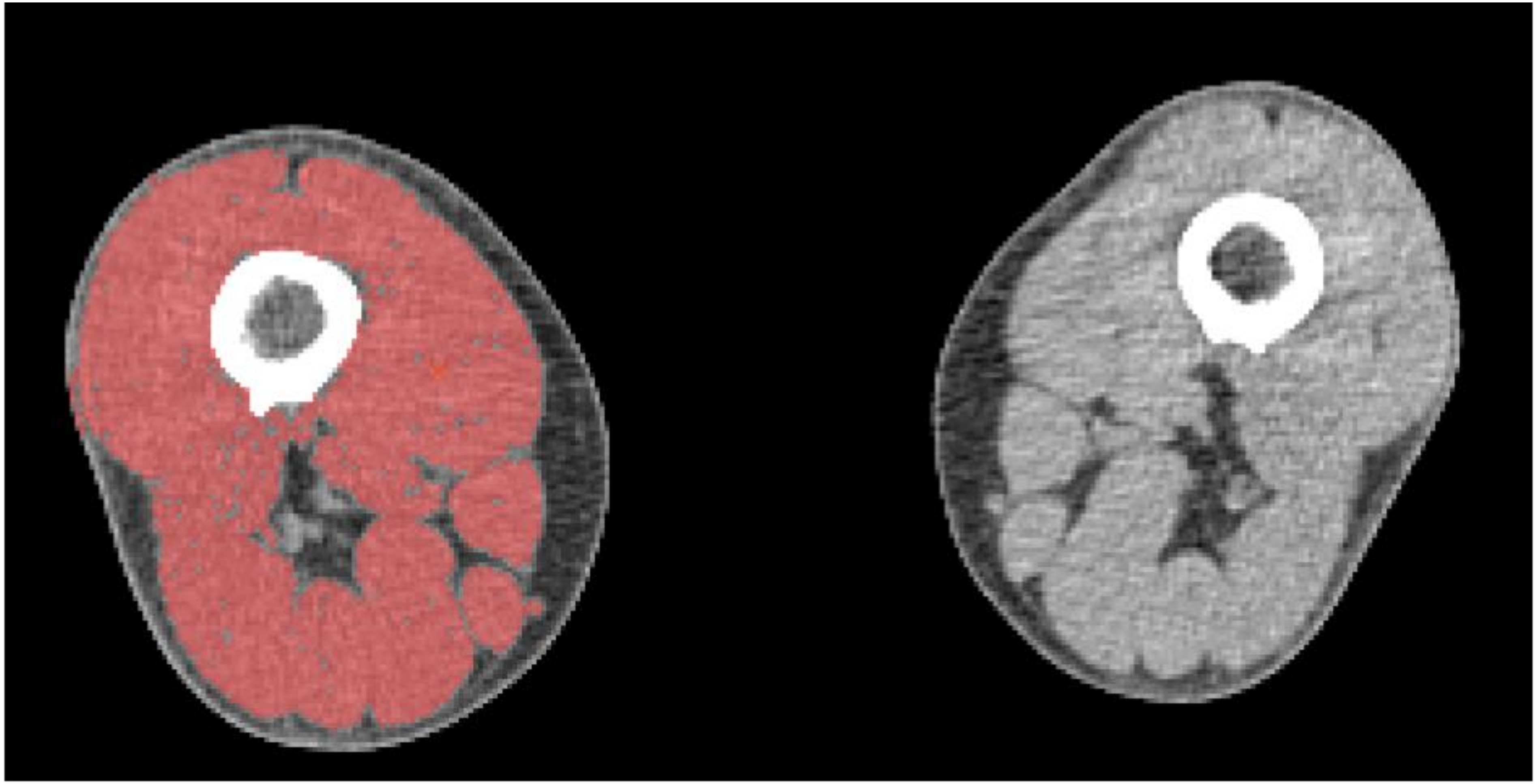
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

(b)

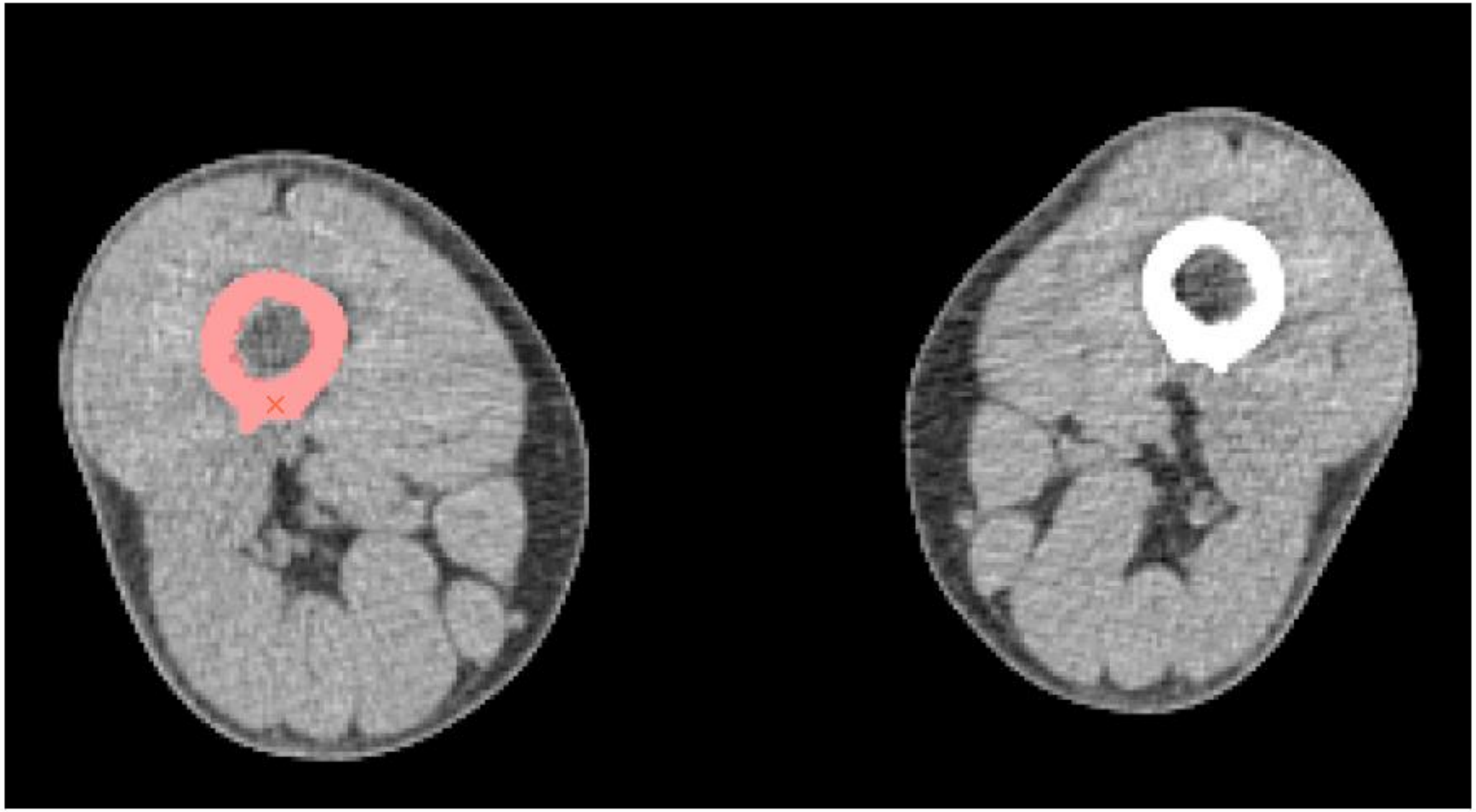
Ex: Muscle/Bone Segmentation in CT Scans



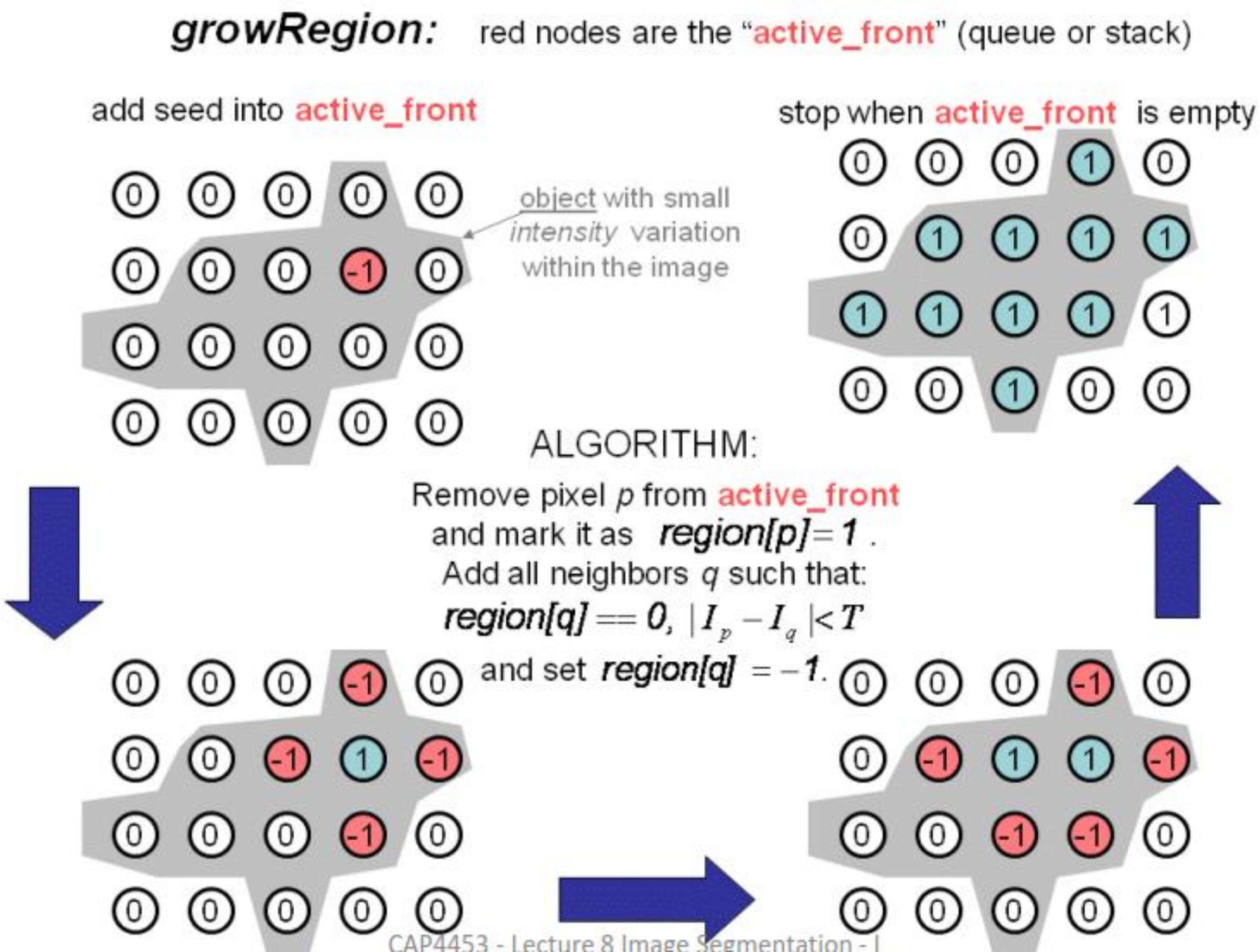
Ex: Muscle/Bone Segmentation in CT Scans



Ex: Muscle/Bone Segmentation in CT Scans



Region Growing Implementation



growRegion: red nodes are the "active_front" (queue or stack)

add seed into active_front

object with small *intensity* variation within the image

stop when active_front is empty

ALGORITHM:

Remove pixel *p* from active_front and mark it as ***region[p]*** $= 1$.
Add all neighbors *q* such that:
region[q] $== 0, \; | I_p - I_q | < T$
and set ***region[q]*** $= -1$.

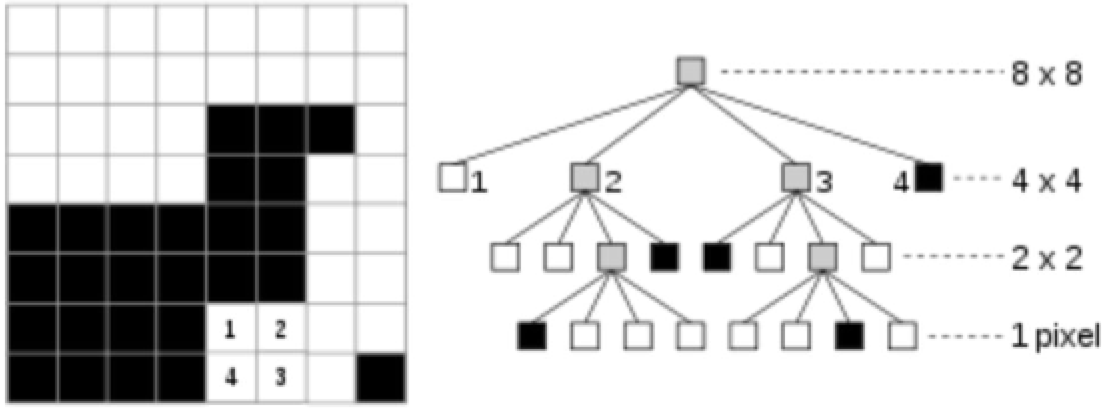
Outline

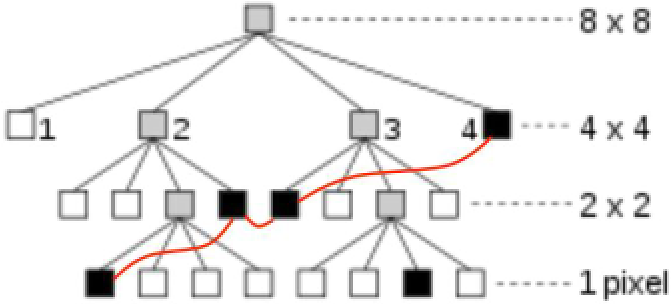
- Image segmentation basics
- Thresholding based
  - Binarization
  - Otsu
- **Region based**
  - Merging
  - **Splitting**
- Clustering based
  - K-means (SLIC)

Region splitting and Merging Segmentation

- Region splitting:
  - Unlike region growing, which starts from a set of seed points, region splitting starts with the whole image as a single region and subdivides it into subsidiary regions recursively while a condition of homogeneity is not satisfied.
- Region merging:
  - Region merging is the opposite of splitting, and works as a way of avoiding over-segmentation
  - Start with small regions (2x2 or 4x4 regions) and merge the regions that have similar characteristics (such as gray level, variance).

Region splitting and Merging Segmentation

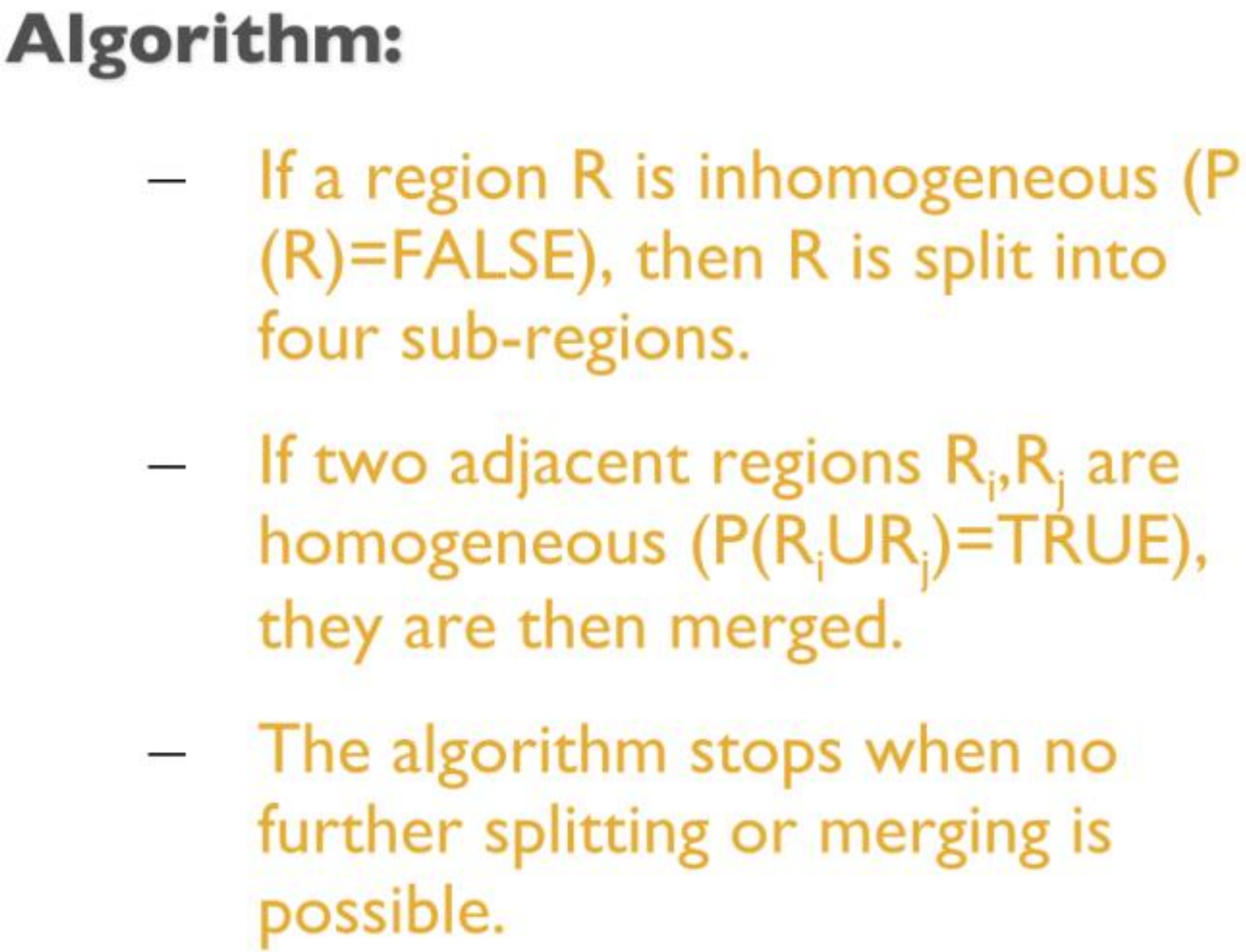




RAG with adjacency relations (in red) for big black region.

- RAG: region adjacency graph
- **Quadtree** for splitting (top-down) procedure

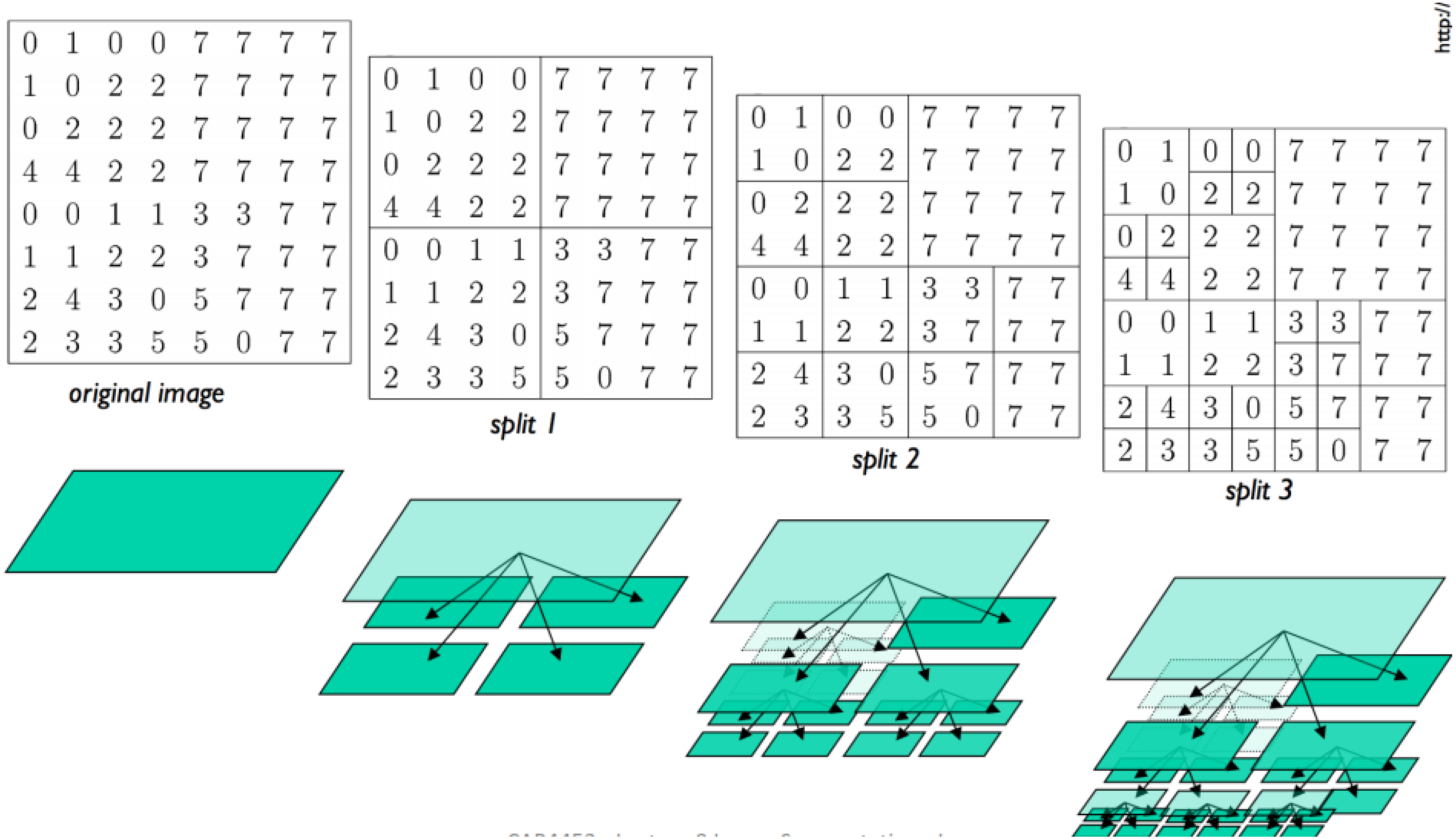
Region splitting and Merging Segmentation



Algorithm:

- If a region R is inhomogeneous (P(R)=FALSE), then R is split into four sub-regions.
- If two adjacent regions R_i,R_j are homogeneous ($P(R_i \cup R_j)$=TRUE), they are then merged.
- The algorithm stops when no further splitting or merging is possible.

Region splitting and Merging Segmentation



http://

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

original image

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 1

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 2

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 3

Region splitting and Merging Segmentation

- Region splitting:
  - Unlike region growing, which starts from a set of seed points, region splitting starts with the whole image as a single region and subdivides it into subsidiary regions recursively while a condition of homogeneity is not satisfied.
- **Region merging:**
  - **Region merging is the opposite of splitting, and works as a way of avoiding over-segmentation**
  - **Start with small regions (2x2 or 4x4 regions) and merge the regions that have similar characteristics (such as gray level, variance).**

Depth of RAG – how many levels?

2	2	2	2	1	1	1	1
2	2	2	2	1	1	1	1
2	2	3	3	2	2	2	2
2	2	3	3	3	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2

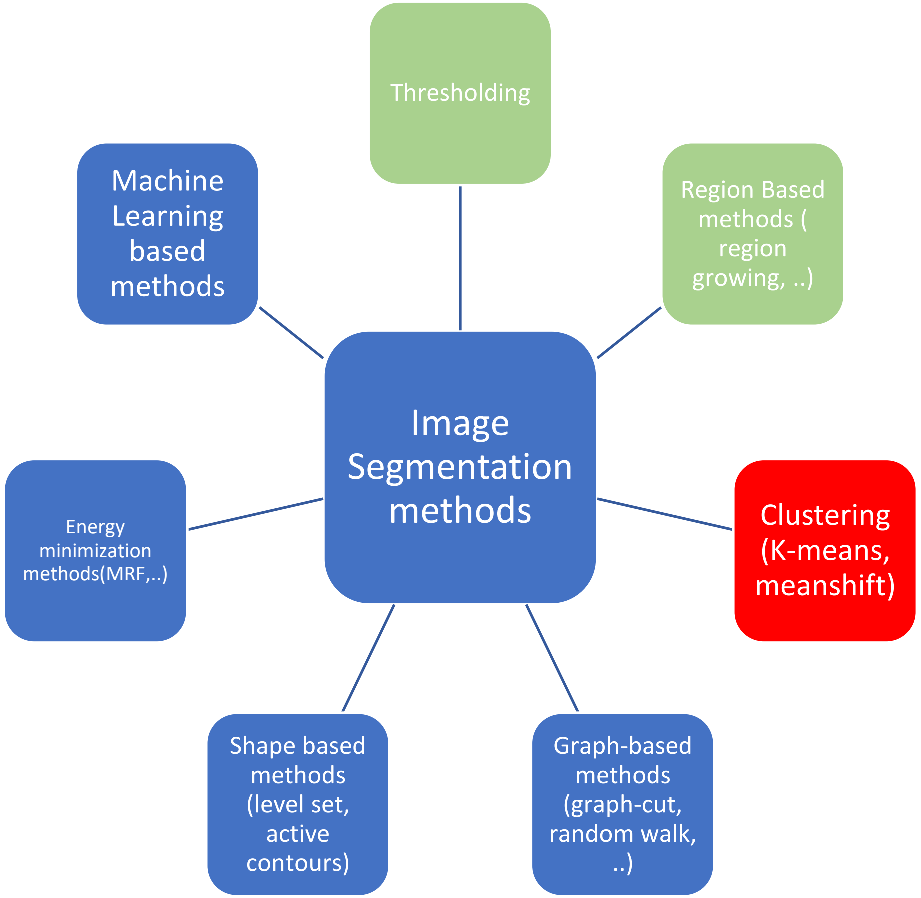
Depth of RAG – how many levels? 4

2	2	2	2	1	1	1	1
2	2	2	2	1	1	1	1
2	2	3	3	2	2	2	2
2	2	3	3	3	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2

Outline

- ~~Image segmentation basics~~
- ~~Thresholding based~~
    - ~~Binarization~~
    - ~~Otsu~~
- ~~Region based~~
    - ~~Merging~~
    - ~~Splitting~~
- **Clustering based**
    - K-means
    - Superpixels (SLIC)

Image segmentation methods



Thresholding

Machine Learning based methods

Region Based methods (region growing, ..)

Image Segmentation methods

Energy minimization methods(MRF,..)

Clustering (K-means, meanshift)

Shape based methods (level set, active contours)

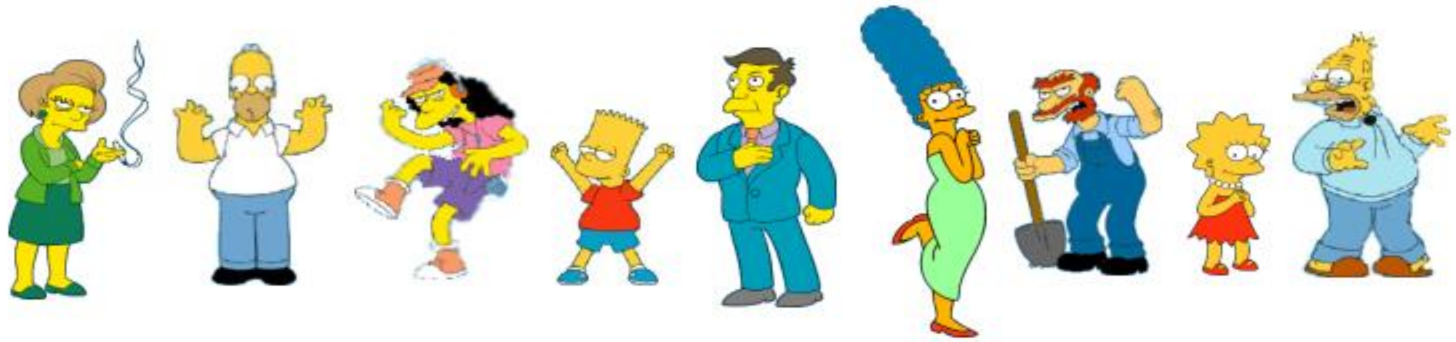
Graph-based methods (graph-cut, random walk, ..)

What is Clustering?

- Organizing data into classes such that:
  - High intra-class similarity
  - Low inter-class similarity

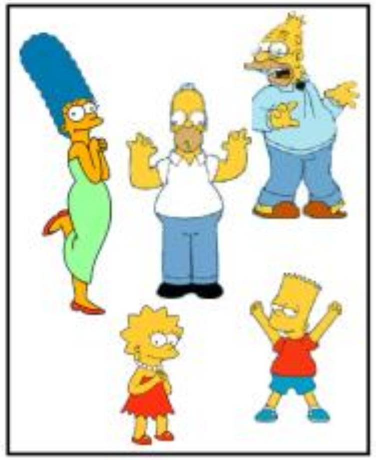
- Finding the class labels and the number of classes directly from the data (as opposed to classification tasks)

What is a natural grouping ?



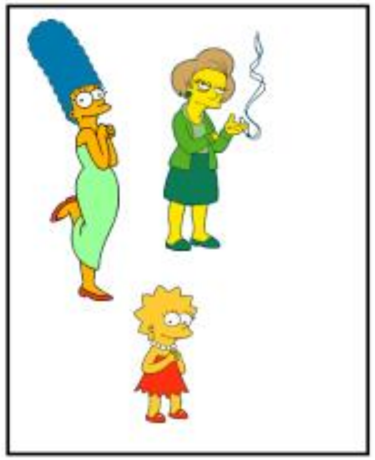
What is a natural grouping ?

Clustering is subjective



Simpson's family

School employees



Females

Males

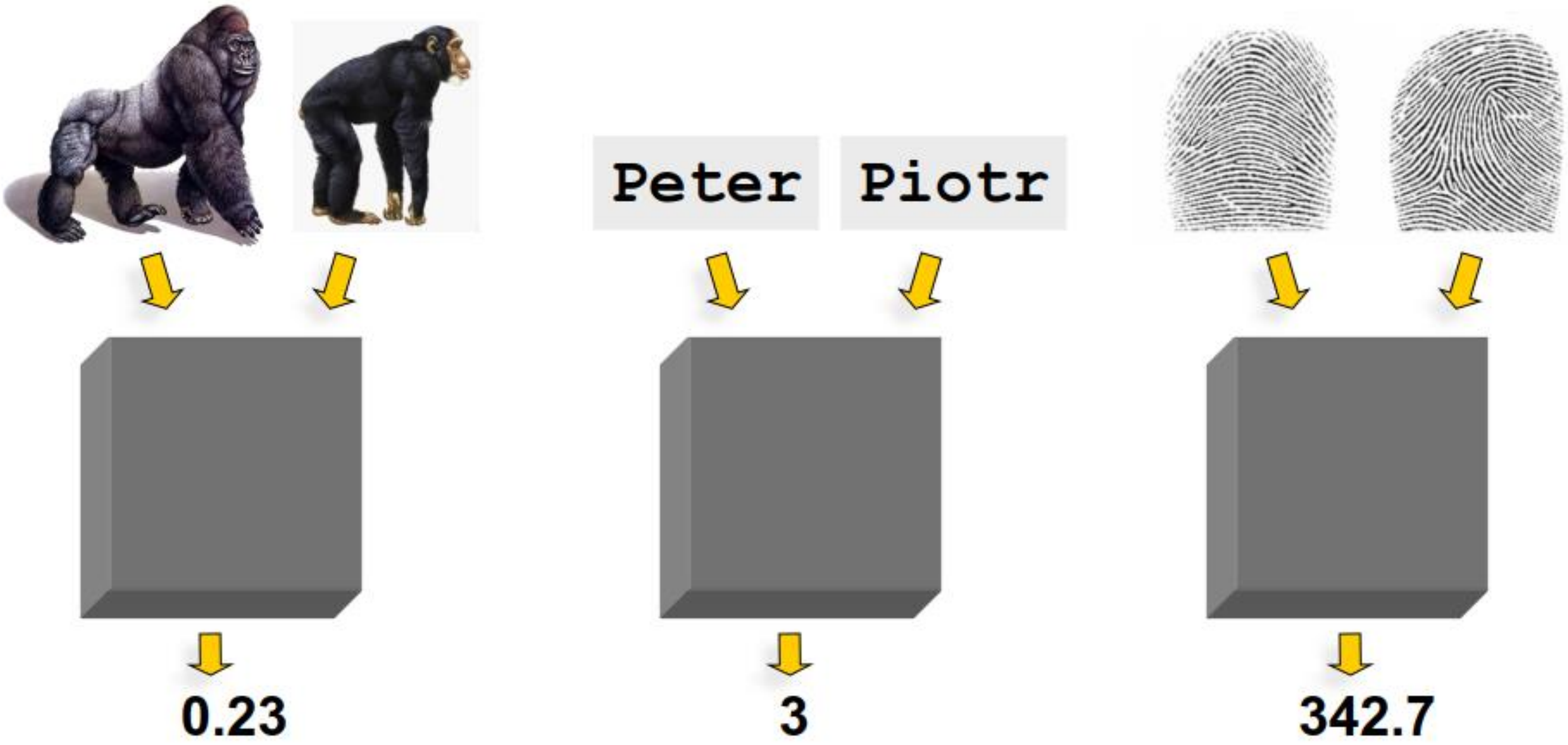
What is similarity ?

What is similarity ?

Cluster by features

- **Color**
- **Intensity**
- **Location**
- **Texture**
- **….**

Distance metrics



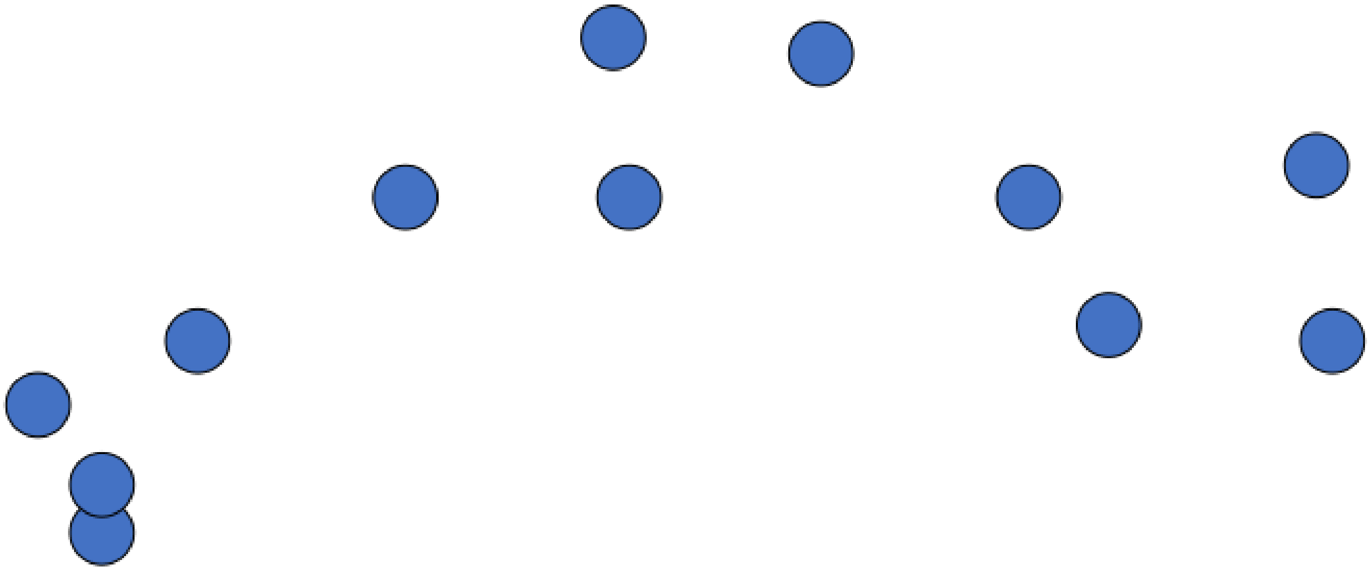
Outline

- ~~Image segmentation basics~~
- ~~Thresholding based~~
  - ~~Binarization~~
  - ~~Otsu~~
- ~~Region based~~
  - ~~Merging~~
  - ~~Splitting~~
- **Clustering based**
  - **K-means**
  - Superpixels (SLIC)

K-means

- Most well-known and popular clustering algorithm:
  - Start with some initial cluster centers
- Iterate:
  - Assign/cluster each example to closest center
  - Recalculate centers as the mean of the points in a cluster

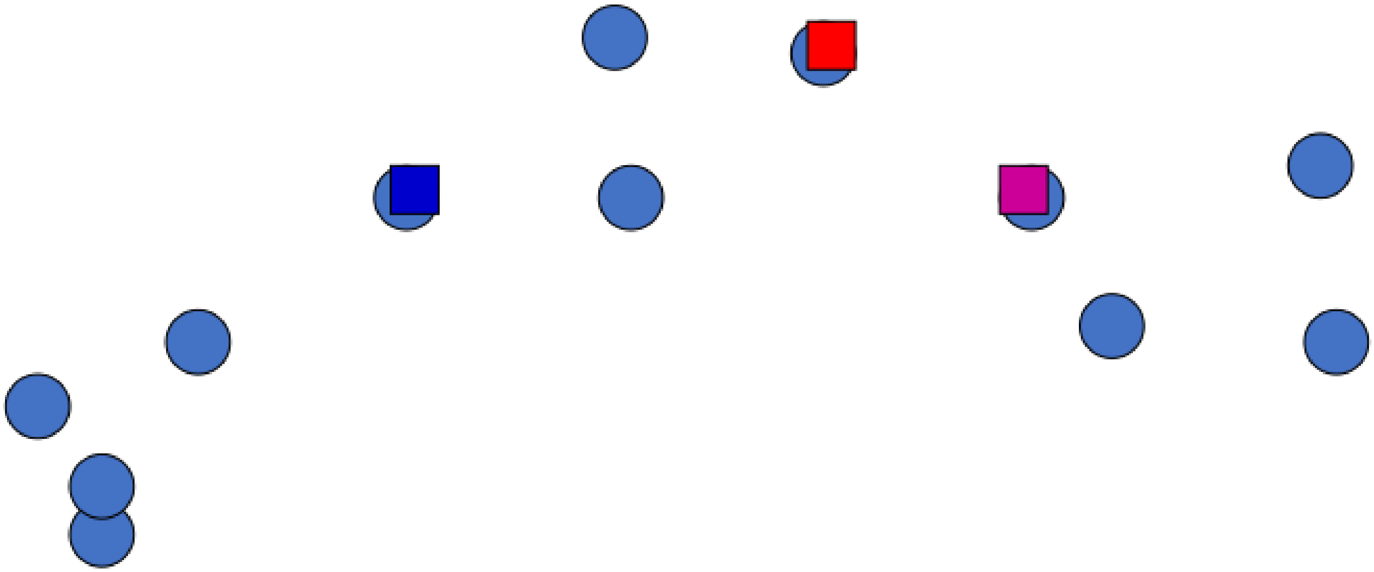
K-means



K-means

Step 0:

- Pick number of classes
- Pick seeds for those classes

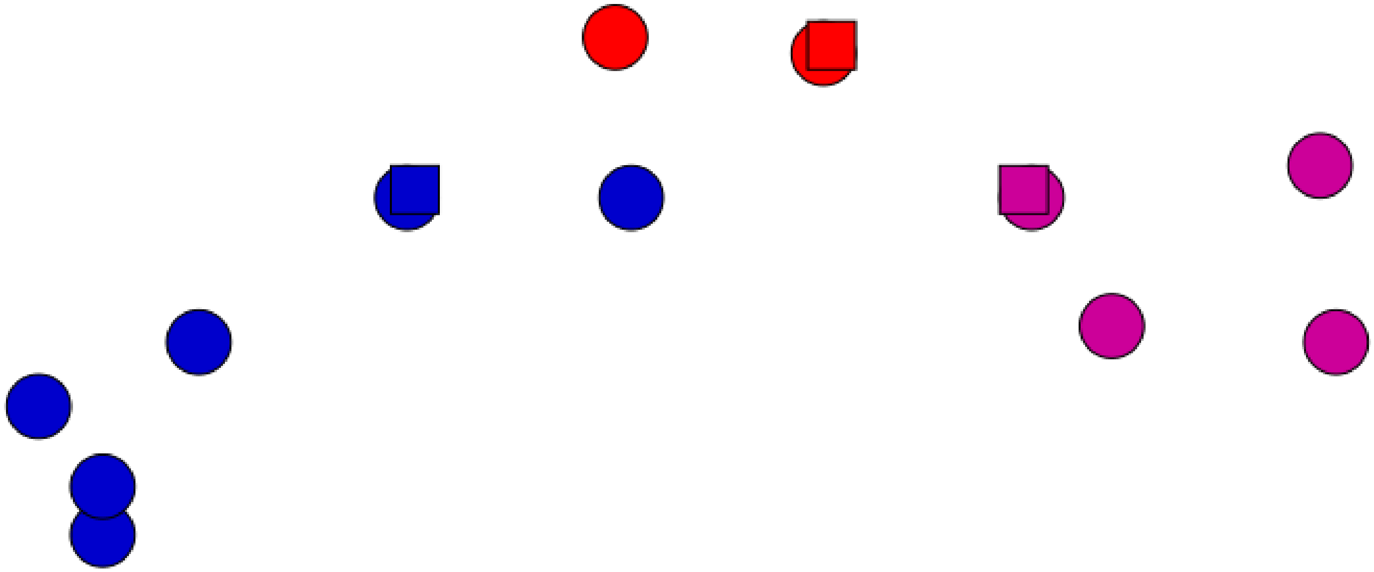


K-means

Iterate:

Assign/cluster each example to closest center

Recalculate centers as the mean of the points in a cluster

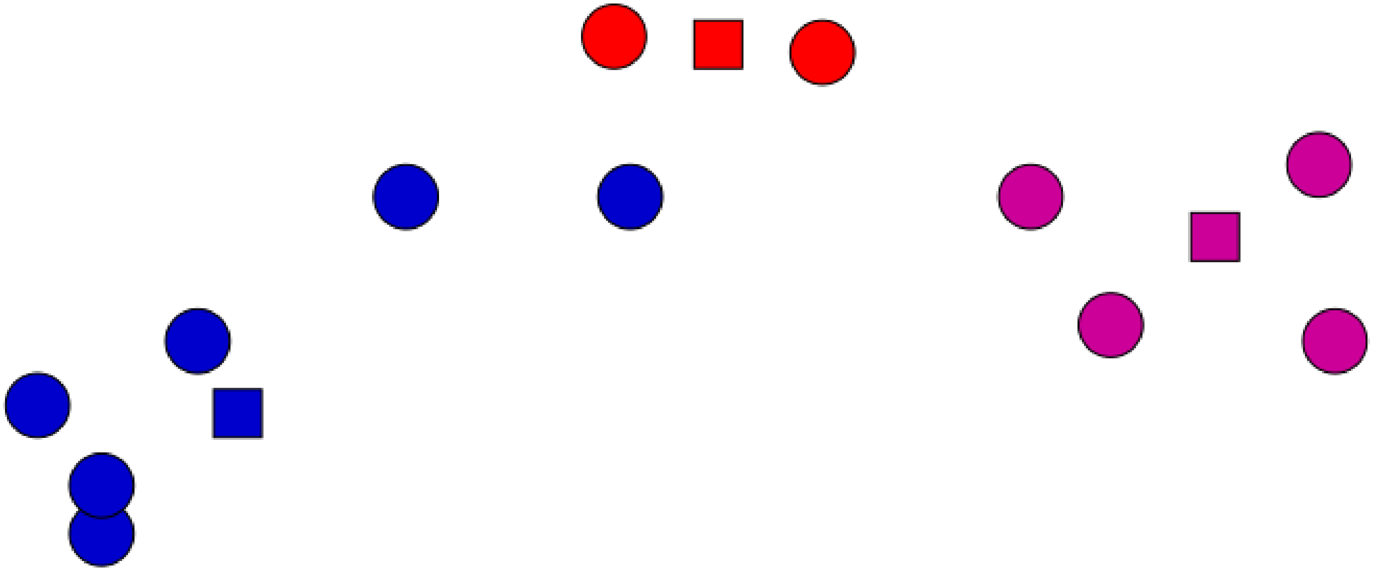


K-means

Iterate:

Assign/cluster each example to closest center

Recalculate centers as the mean of the points in a cluster

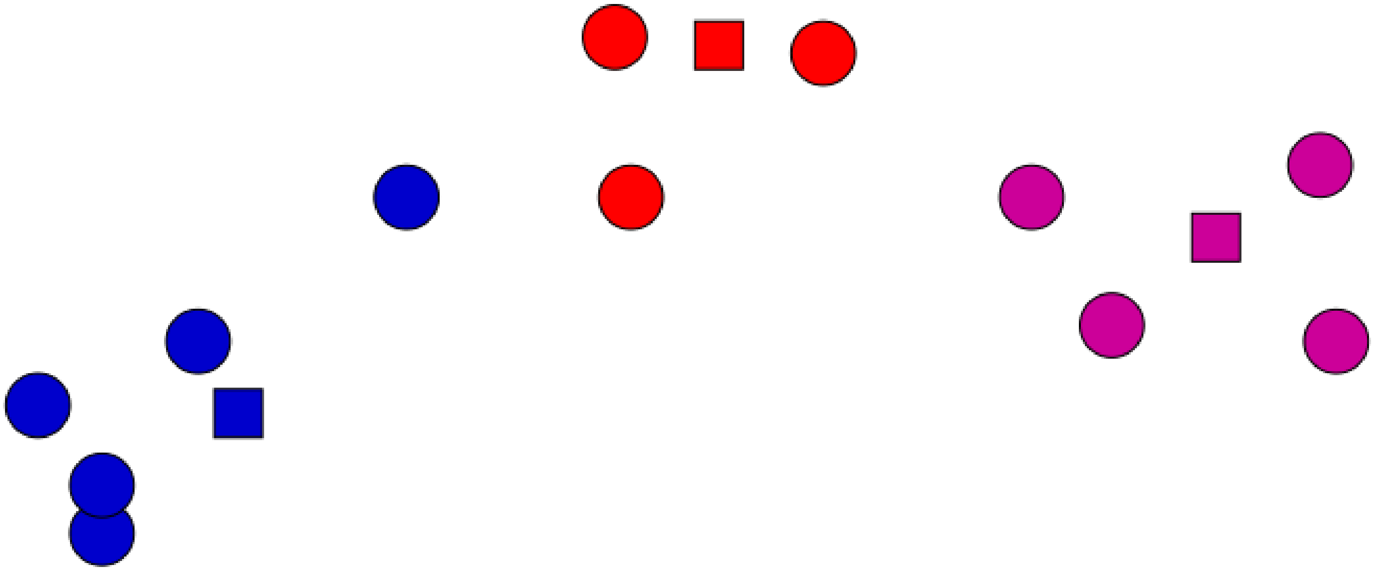


K-means

Iterate:

Assign/cluster each example to closest center

Recalculate centers as the mean of the points in a cluster

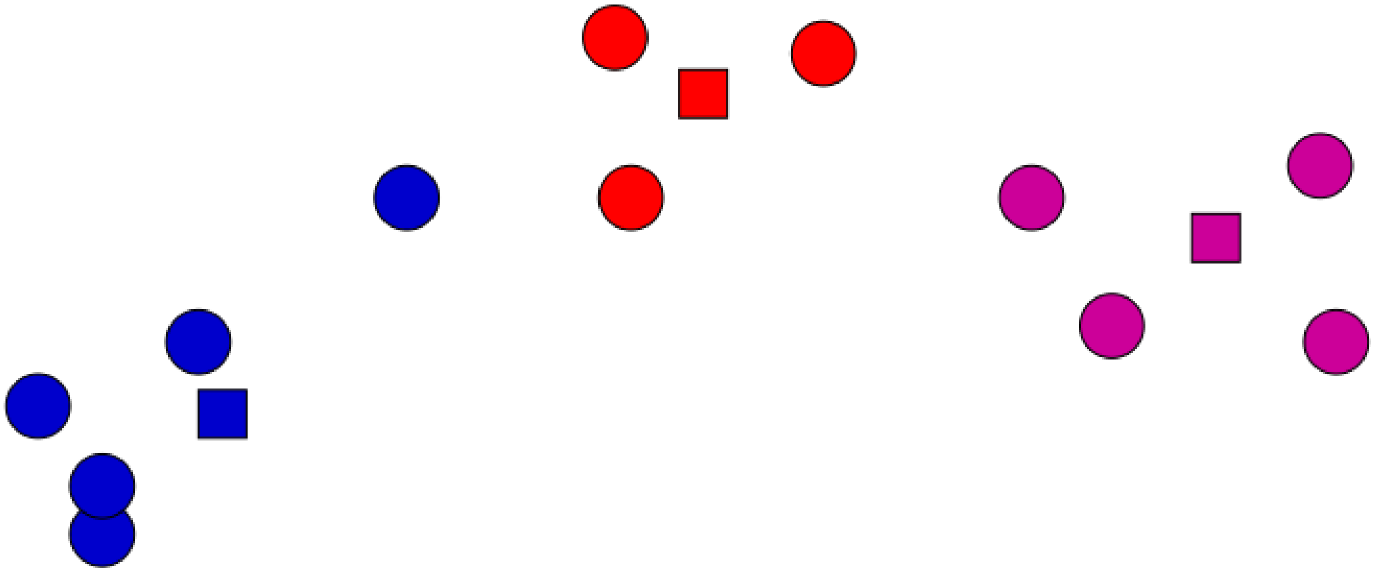


K-means

Iterate:

Assign/cluster each example to closest center

Recalculate centers as the mean of the points in a cluster

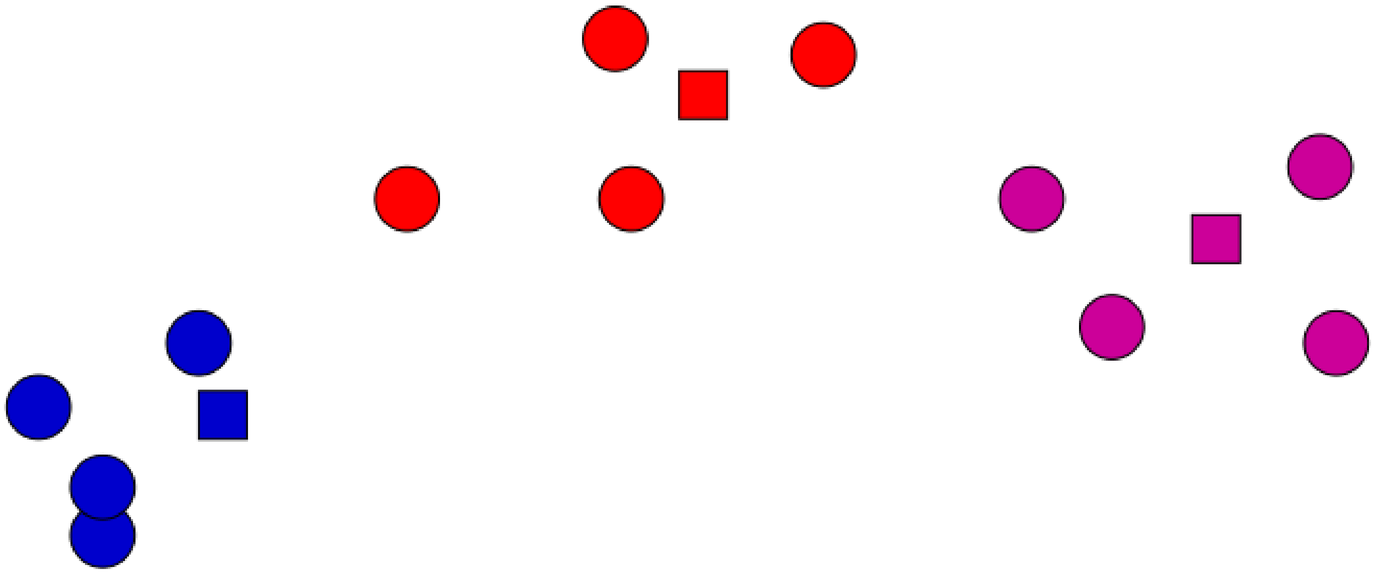


K-means

Irate:

Assign/cluster each example to closest center

Recalculate centers as the mean of the points in a cluster

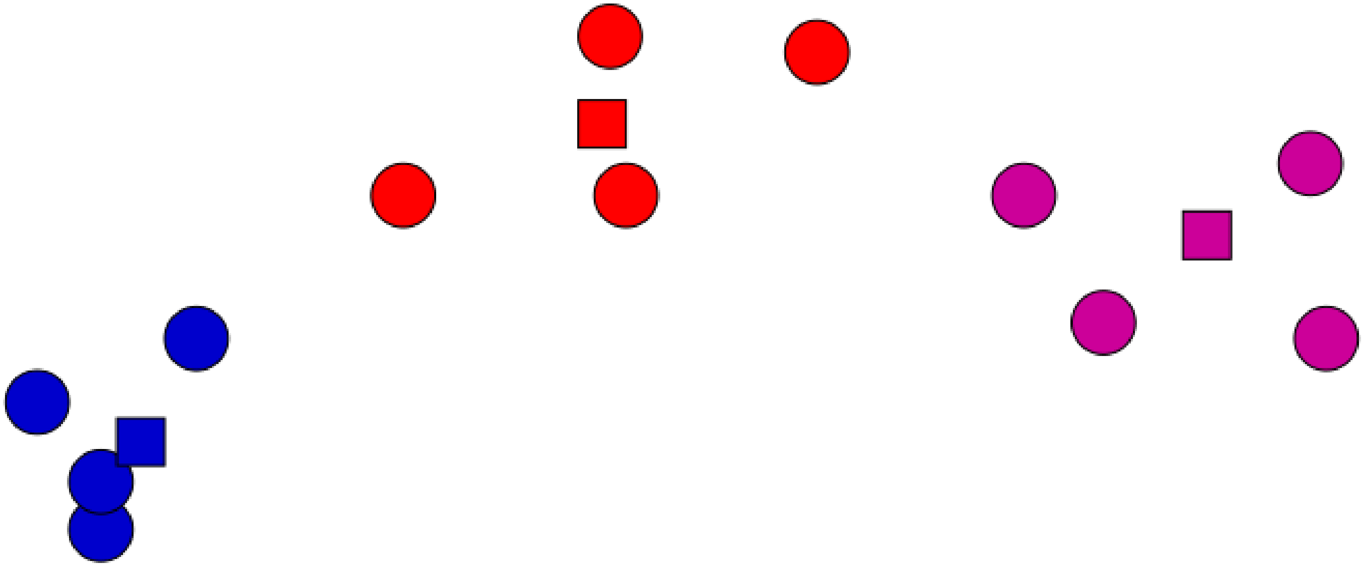


K-means

Iterate:

Assign/cluster each example to closest center

Recalculate centers as the mean of the points in a cluster

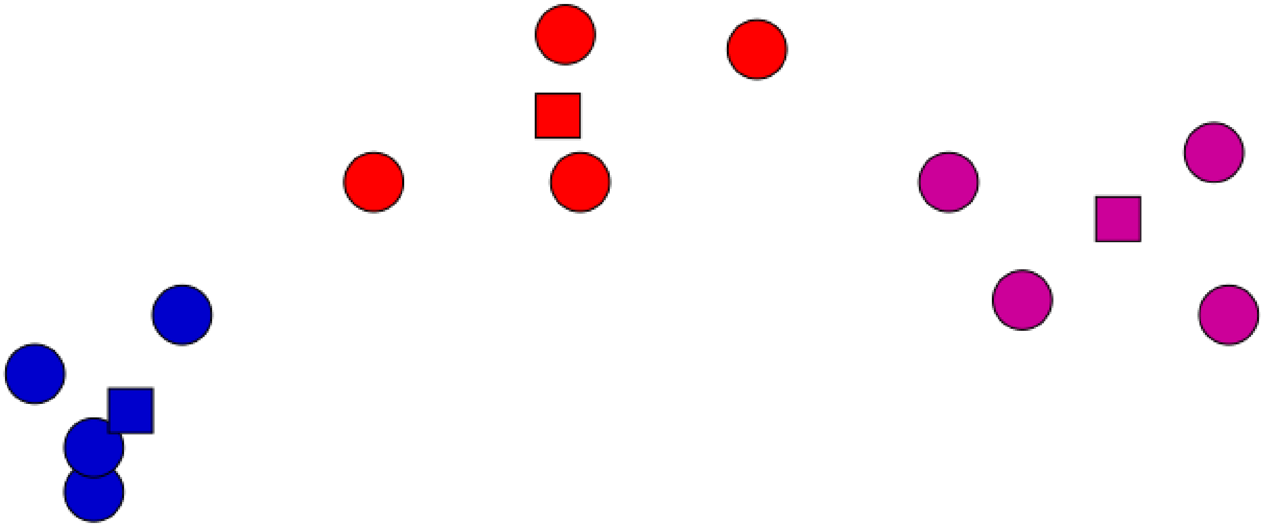


K-means

Iterate:

Assign/cluster each example to closest center

Recalculate centers as the mean of the points in a cluster

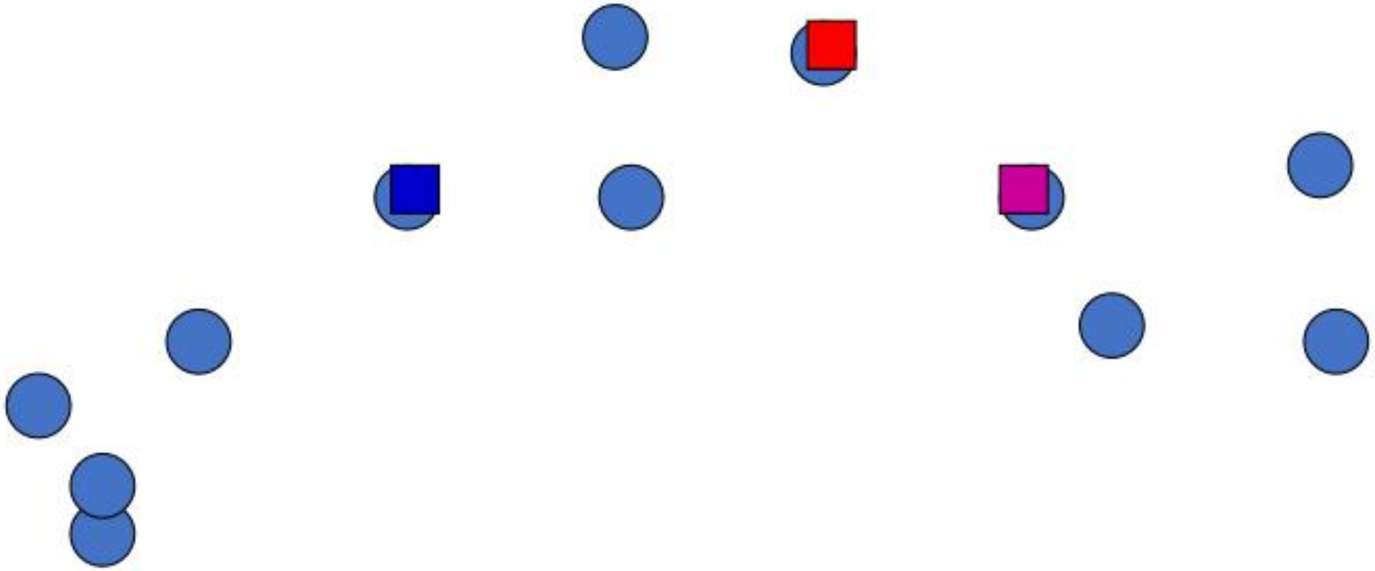


No changes: Done

K-means

Iterate:

- **Assign/cluster each example to closest center**
- Recalculate centers as the mean of the points in a cluster



How do we do this?

K-means

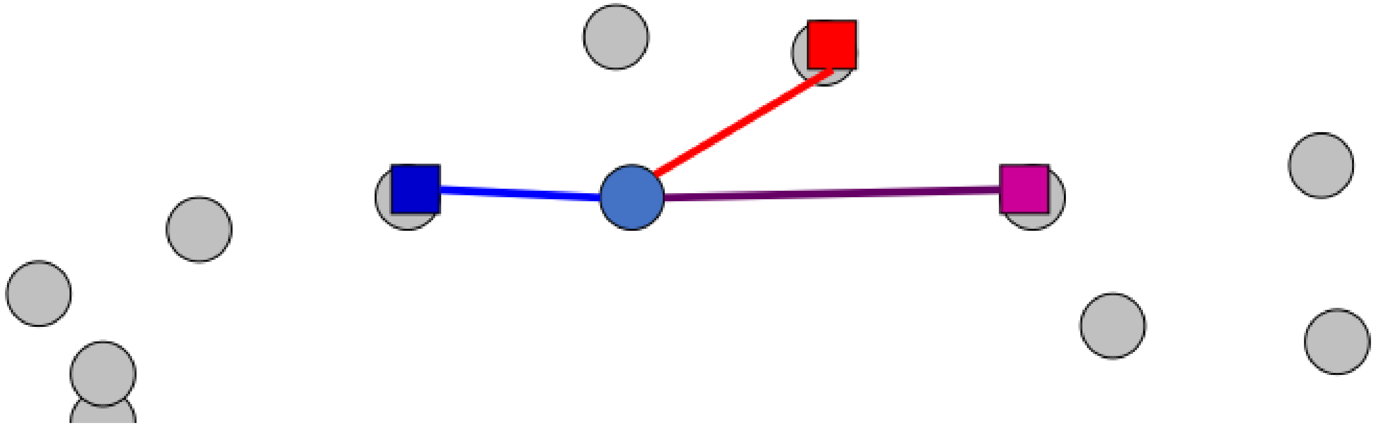
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

    - get distance to each cluster center
    - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



K-means

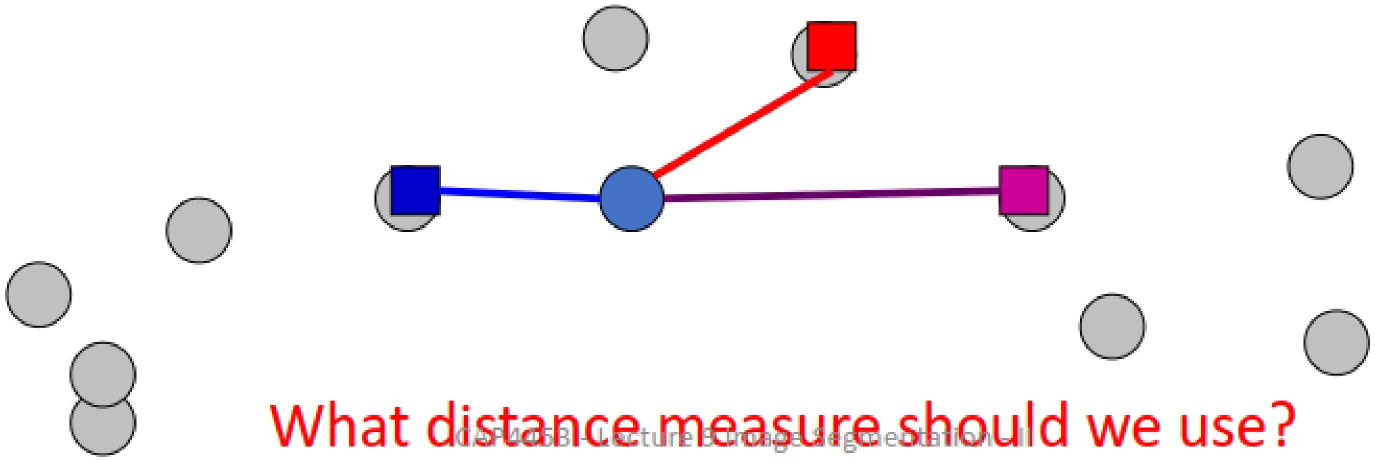
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

  - get **distance** to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



What distance measure should we use?

Distance measures

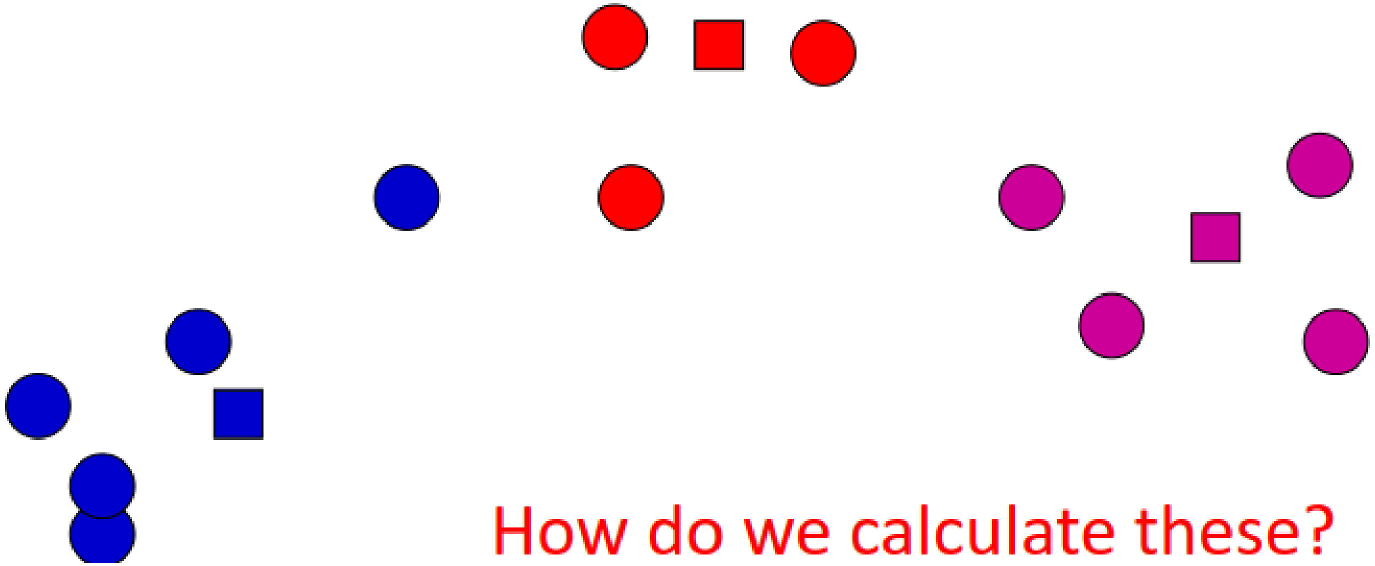
$$d = \sqrt{\sum_i (x_i^2 - y_i^2)}$$

- p-norm $$\|x\|_p = \left(\sum_i |a_i|^p \right)^{\frac{1}{p}} \qquad p \geq 1$$
- Euclidean norm $$\|x\|_2 = \left(\sum_i |a_i|^2 \right)^{1/2}$$
- L1-norm $$\|x\|_1 = \left(\sum_i |a_i| \right)$$

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster



How do we calculate these?

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

$$\mu_j = \frac{\sum_{i:y_i=j} x_i}{\sum_{i:y_i=j} 1}$$

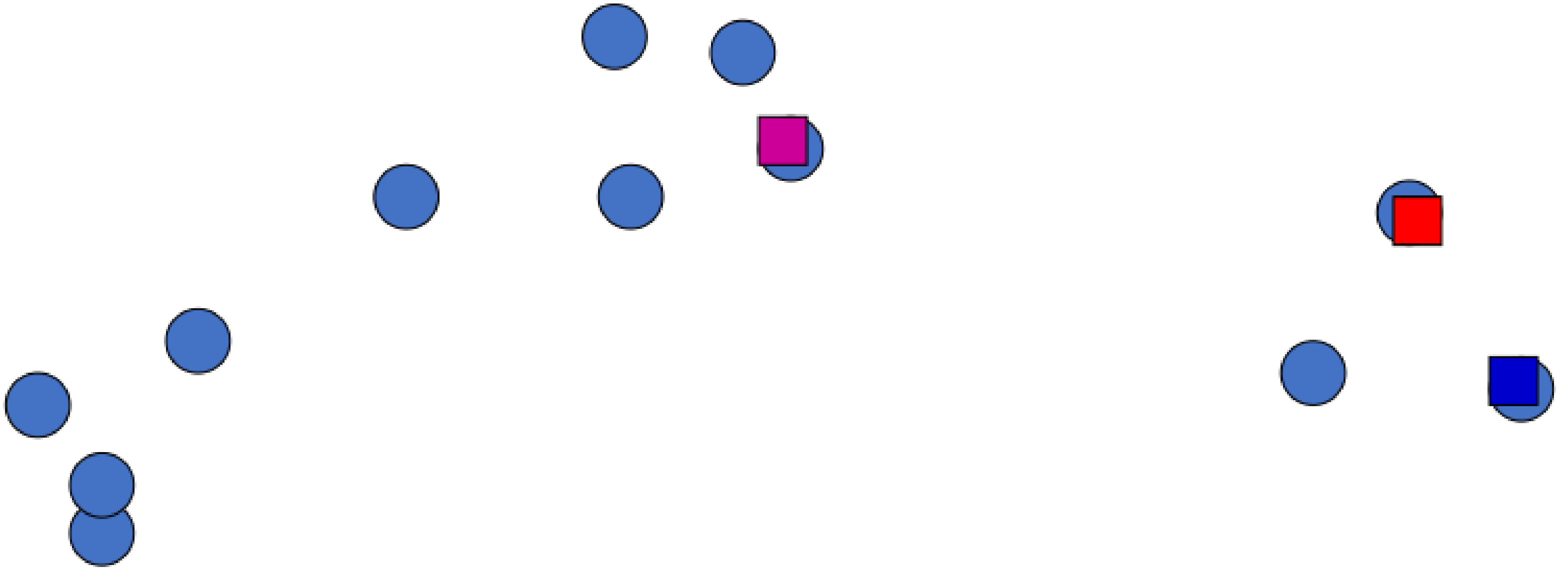
K-means loss function

K-means tries to minimize what is called the "k-means" loss function:

$$loss = \sum_{i=1}^{n} d^2(x_i, \mu_k), \qquad where\ \mu_k\ is\ the\ cluster\ center\ for\ x_i$$

that is, the sum of the squared distances from each point to the associated cluster center

K-means: initialization



What would happen here?

Seed selection ideas?

Seed choice

Results can vary drastically based on random seed selection

Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings

Common heuristics

- Random centers in the space
- Randomly pick examples
- Points least similar to any existing center (furthest centers heuristic)
- **Try out multiple starting points**
- Initialize with the results of another clustering method

Choosing the Appropriate Number of Clusters

1. The **elbow method**
2. The **silhouette coefficient**

K-Means Clustering in Python: A Practical Guide – Real Python

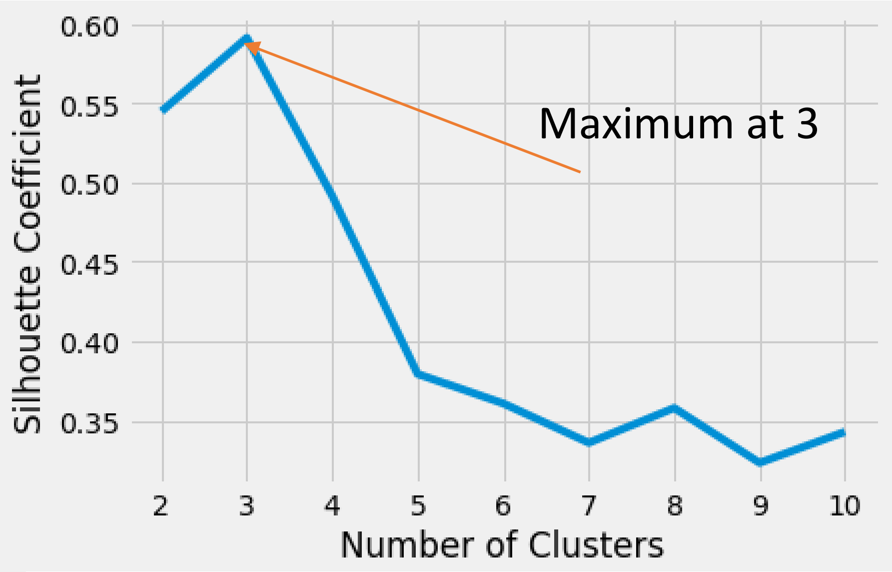
Choosing the Appropriate Number of Clusters

- run several k-means,
- increment k with each iteration
- record the **sum of the squared error (SSE)**
  - The SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid

1.The **elbow method**

Choosing the Appropriate Number of Clusters

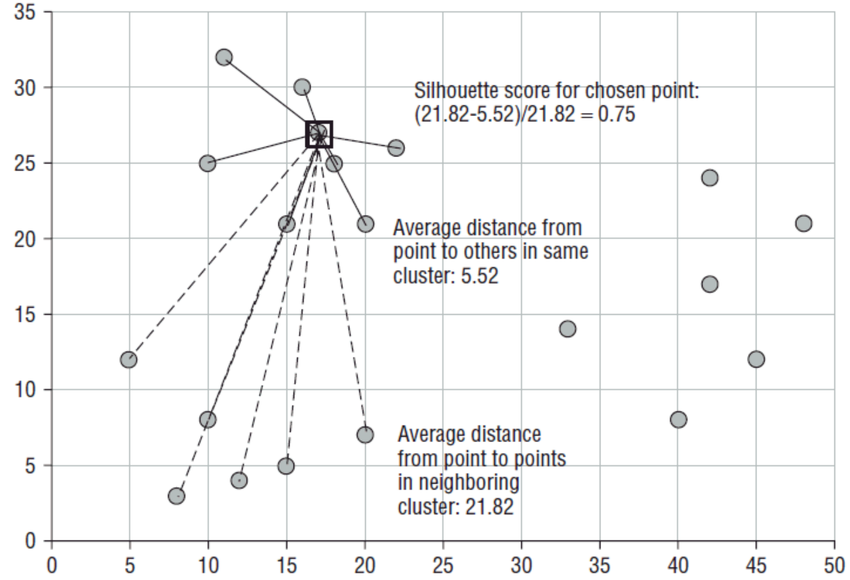
- run several k-means,
- increment k with each iteration
- Pick max silhouette coefficient
  1. How close the data point is to other points in the cluster
  2. How far away the data point is from points in other clusters



  - (b - a) / max(a, b). Where,

    a: intra-cluster distance

    b: distance between a sample and
    the nearest cluster that the sample is not a part of.

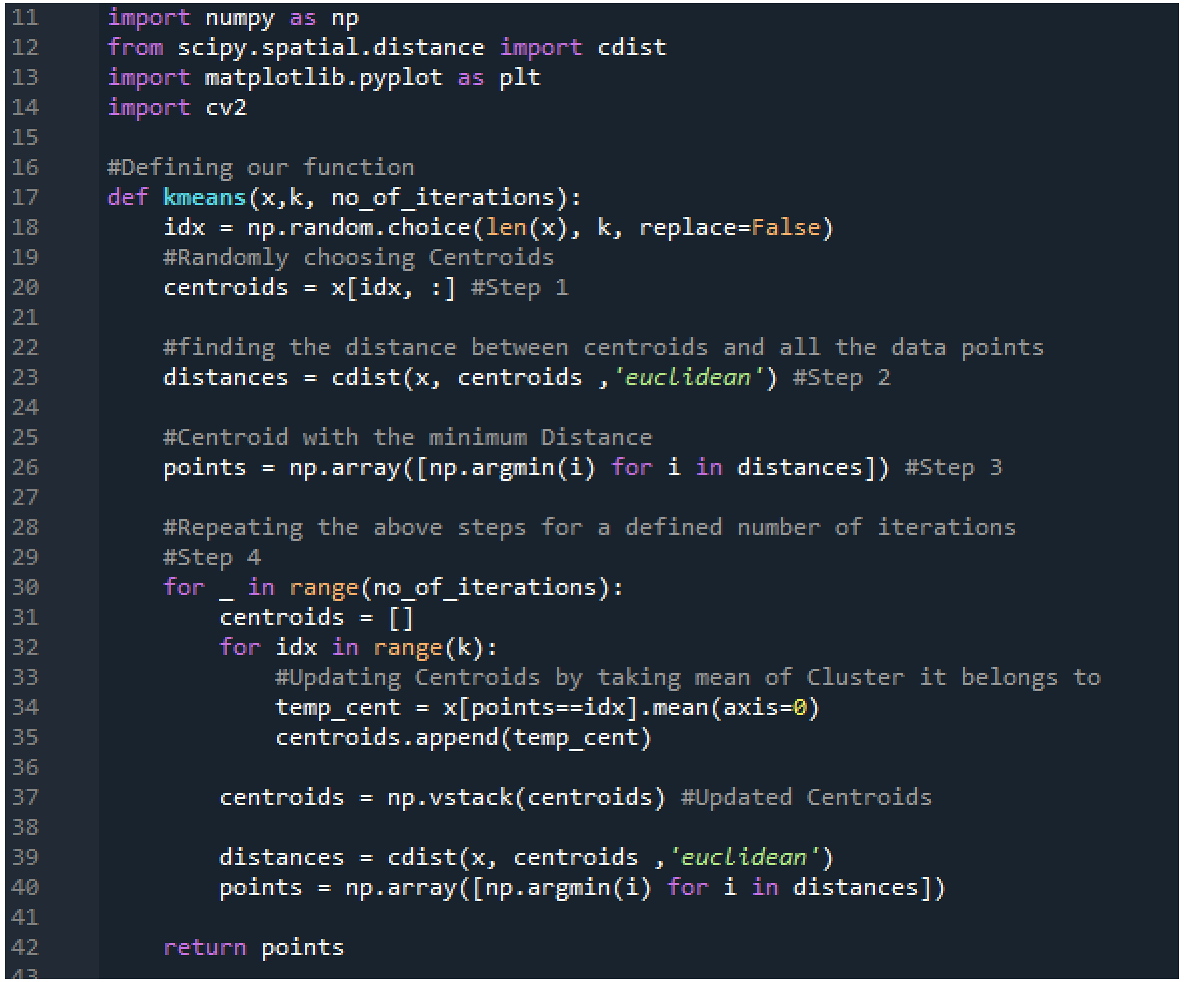


2. The **silhouette coefficient**

Segmenting an image with K-means

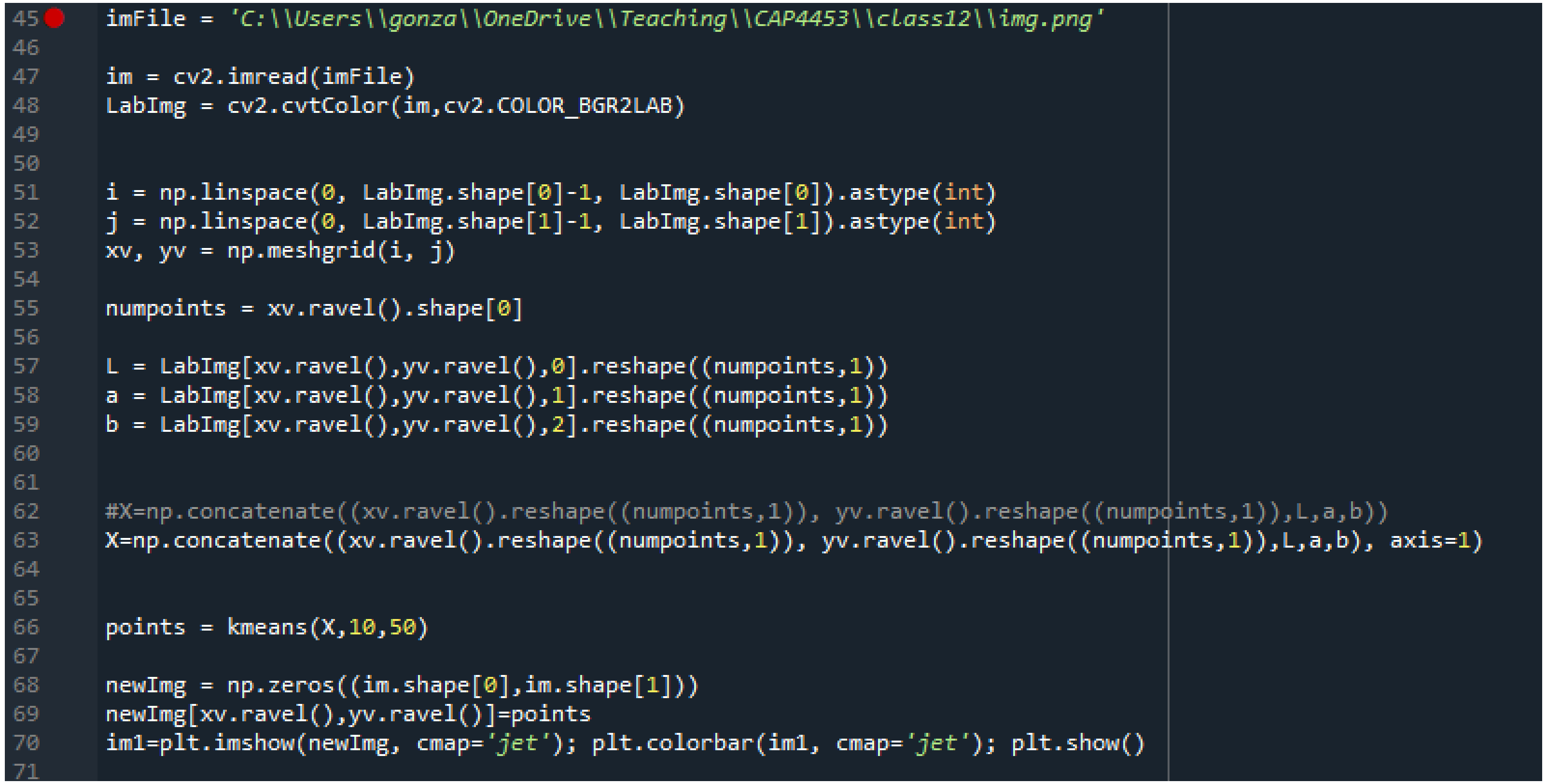
- Example:
  - Vector: (coordinates i, coordinate j , Color L, Color a, Color b) : 5 dims
  - Distance: Euclidean distance
  - Number of clusters: 10
  - Seeds selected randomly

Kmeans function from scratch



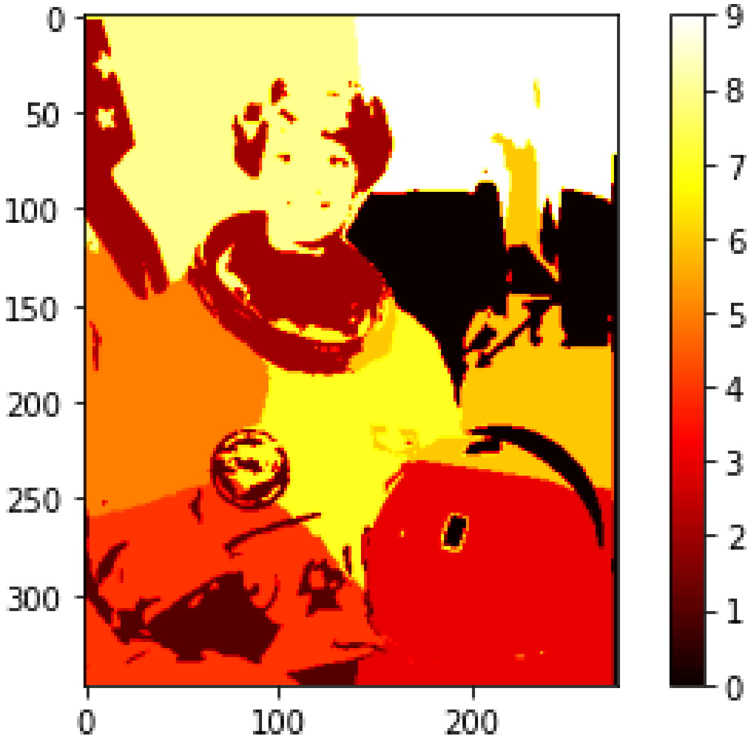
```python
import numpy as np
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
import cv2

#Defining our function
def kmeans(x,k, no_of_iterations):
    idx = np.random.choice(len(x), k, replace=False)
    #Randomly choosing Centroids
    centroids = x[idx, :] #Step 1

    #finding the distance between centroids and all the data points
    distances = cdist(x, centroids ,'euclidean') #Step 2

    #Centroid with the minimum Distance
    points = np.array([np.argmin(i) for i in distances]) #Step 3

    #Repeating the above steps for a defined number of iterations
    #Step 4
    for _ in range(no_of_iterations):
        centroids = []
        for idx in range(k):
            #Updating Centroids by taking mean of Cluster it belongs to
            temp_cent = x[points==idx].mean(axis=0)
            centroids.append(temp_cent)

        centroids = np.vstack(centroids) #Updated Centroids

        distances = cdist(x, centroids ,'euclidean')
        points = np.array([np.argmin(i) for i in distances])

    return points
```

Calling Kmeans



```
imFile = 'C:\\Users\\gonza\\OneDrive\\Teaching\\CAP4453\\class12\\img.png'

im = cv2.imread(imFile)
LabImg = cv2.cvtColor(im,cv2.COLOR_BGR2LAB)


i = np.linspace(0, LabImg.shape[0]-1, LabImg.shape[0]).astype(int)
j = np.linspace(0, LabImg.shape[1]-1, LabImg.shape[1]).astype(int)
xv, yv = np.meshgrid(i, j)

numpoints = xv.ravel().shape[0]

L = LabImg[xv.ravel(),yv.ravel(),0].reshape((numpoints,1))
a = LabImg[xv.ravel(),yv.ravel(),1].reshape((numpoints,1))
b = LabImg[xv.ravel(),yv.ravel(),2].reshape((numpoints,1))


#X=np.concatenate((xv.ravel().reshape((numpoints,1)), yv.ravel().reshape((numpoints,1)),L,a,b))
X=np.concatenate((xv.ravel().reshape((numpoints,1)), yv.ravel().reshape((numpoints,1)),L,a,b), axis=1)


points = kmeans(X,10,50)

newImg = np.zeros((im.shape[0],im.shape[1]))
newImg[xv.ravel(),yv.ravel()]=points
im1=plt.imshow(newImg, cmap='jet'); plt.colorbar(im1, cmap='jet'); plt.show()
```

Results

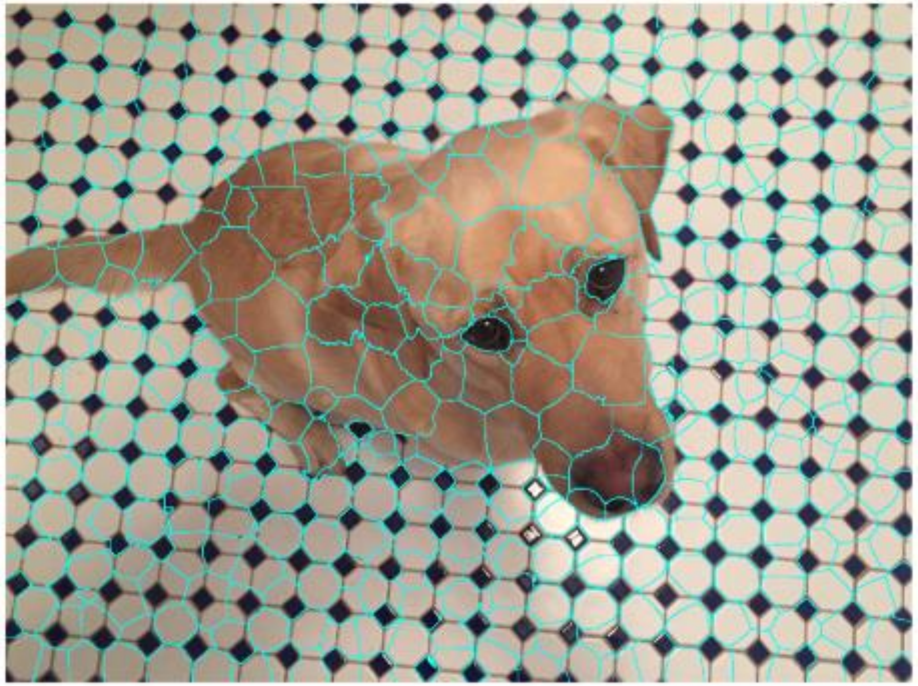


Outline

- ~~Image segmentation basics~~
- ~~Thresholding based~~
    - ~~Binarization~~
    - ~~Otsu~~
- ~~Region based~~
    - ~~Merging~~
    - ~~Splitting~~
- **Clustering based**
    - K-means
    - **Superpixels (SLIC)**

Superpixels

- They carry more information than pixels.
- Superpixels have a perceptual meaning since pixels belonging to a given superpixel share similar visual properties.
- They provide a convenient and compact representation of images that can be very useful for computationally demanding problems.

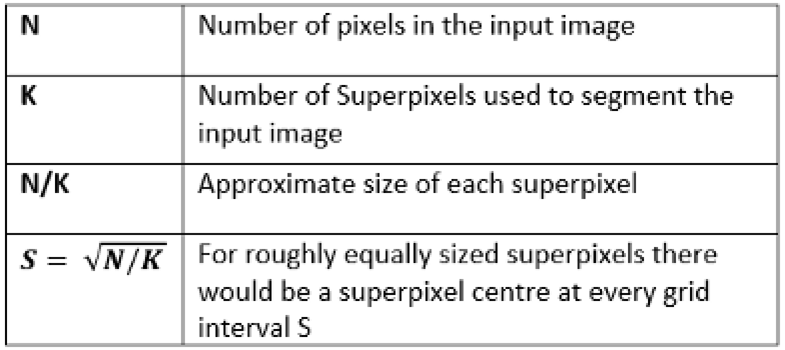


[Superpixels and SLIC. What is a Superpixel? | by Darshita Jain | Medium]

SLIC (Anchanta et. al. TPAMI 2012)

Input:

- a desired number of approximately equally-sized superpixels K



N	Number of pixels in the input image
K	Number of Superpixels used to segment the input image
N/K	Approximate size of each superpixel
$S = \sqrt{N/K}$	For roughly equally sized superpixels there would be a superpixel centre at every grid interval S

Features:

five-dimensional [labxy] space,

- [lab] is the pixel color vector in CIELAB color space
- xy is the pixel position.

Distances:

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

$$D_s = d_{lab} + \frac{m}{S} d_{xy} \, ,$$

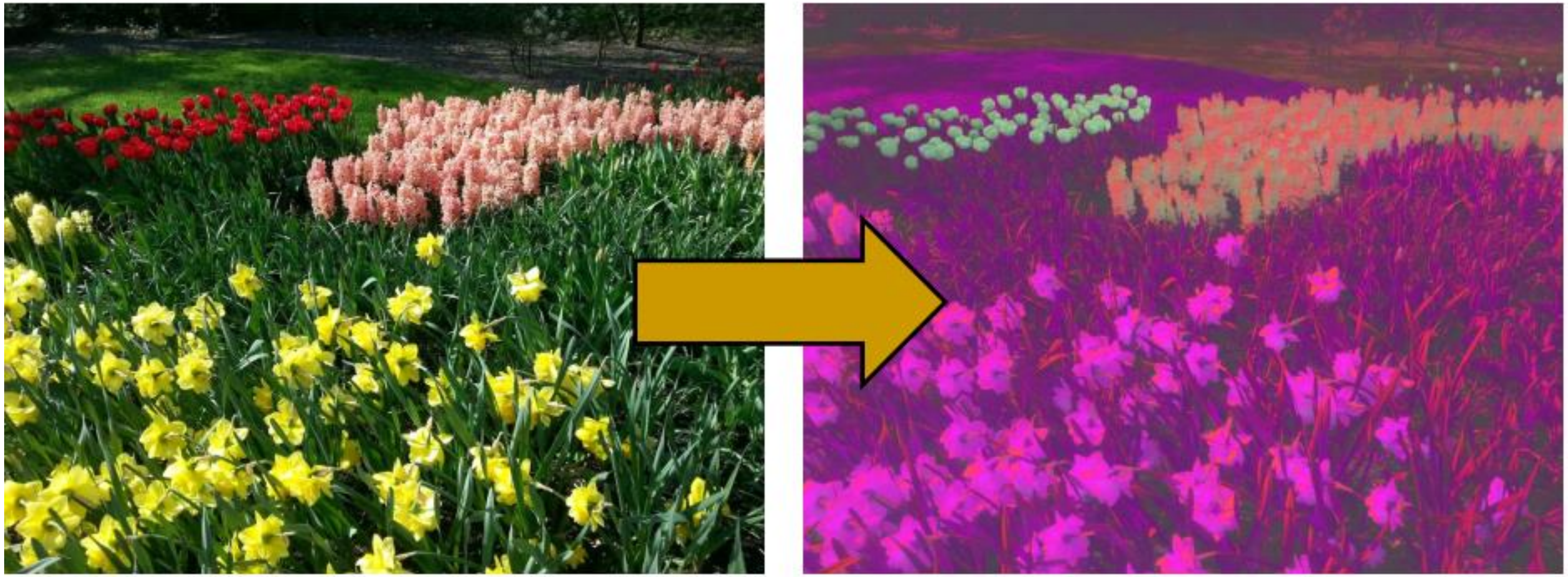
SLIC (Anchanta et. al. TPAMI 2012)

1. Get Features: Lab color, x-y position
2. Initialize cluster centers on pixel grid in steps S
3. Move centers to position in 3x3 window with smallest gradient
4. Compare each pixel to cluster center within 2S pixel distance and assign to nearest
5. Recompute cluster centers as mean color/position of pixels belonging to each cluster
6. Stop when residual error is small

+ Fast 0.36s for 320x240
+ Regular superpixels
+ Superpixels fit boundaries
- May miss thin objects
- Large number of superpixels

SLIC Example

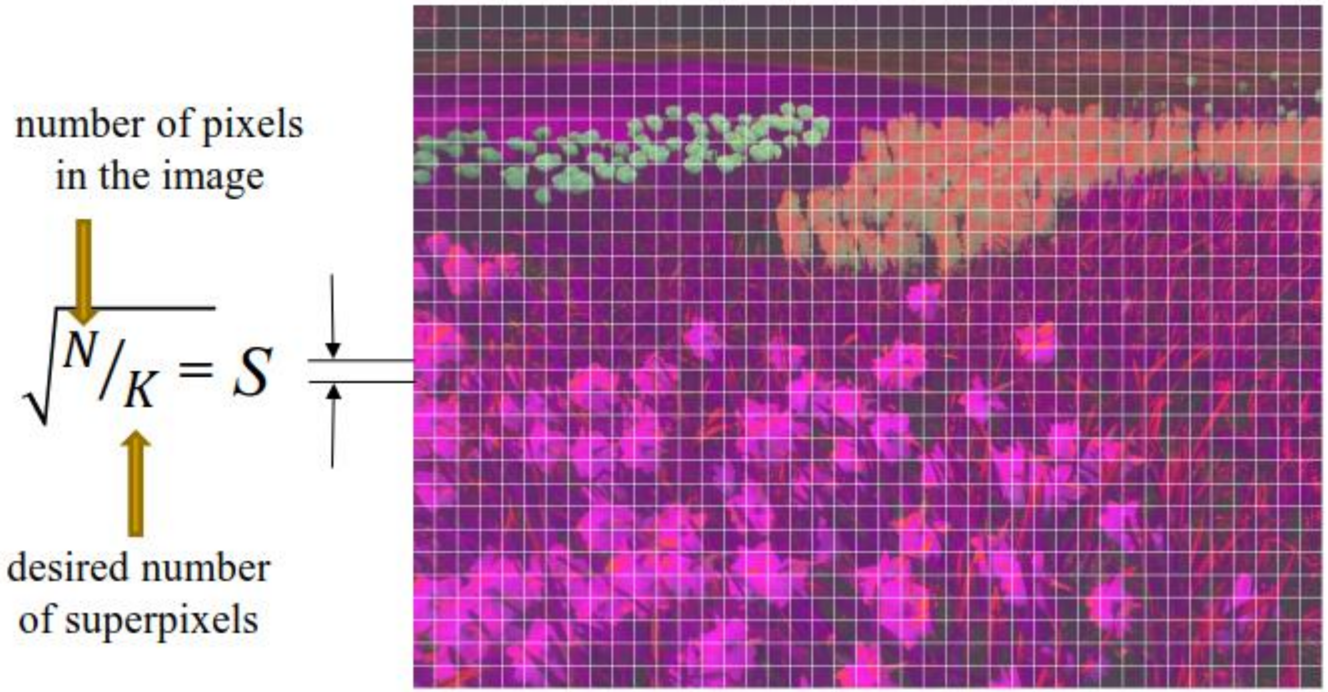
1. Convert the RGB image to CIELAB color space.



http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

SLIC Example

2. Initialize cluster centers $C_k = [l_k; a_k; b_k; x_k; y_k]^T$ by sampling pixels at regular grid steps S.



number of pixels in the image

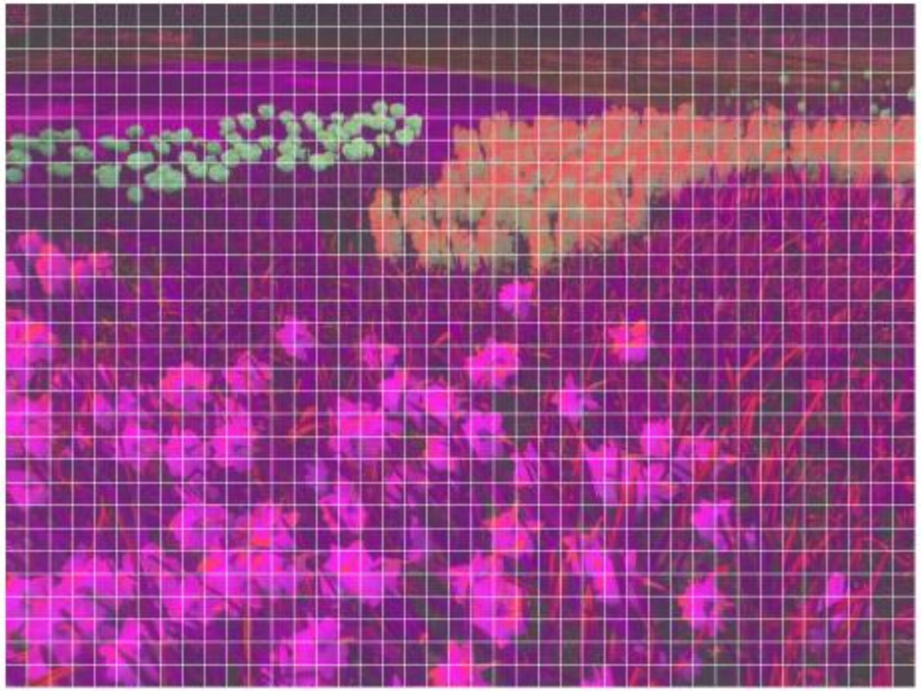
$$\sqrt{N/K} = S$$

desired number of superpixels

http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

SLIC Example

3. Move cluster centers to the lowest gradient position in a 3×3 neighborhood.

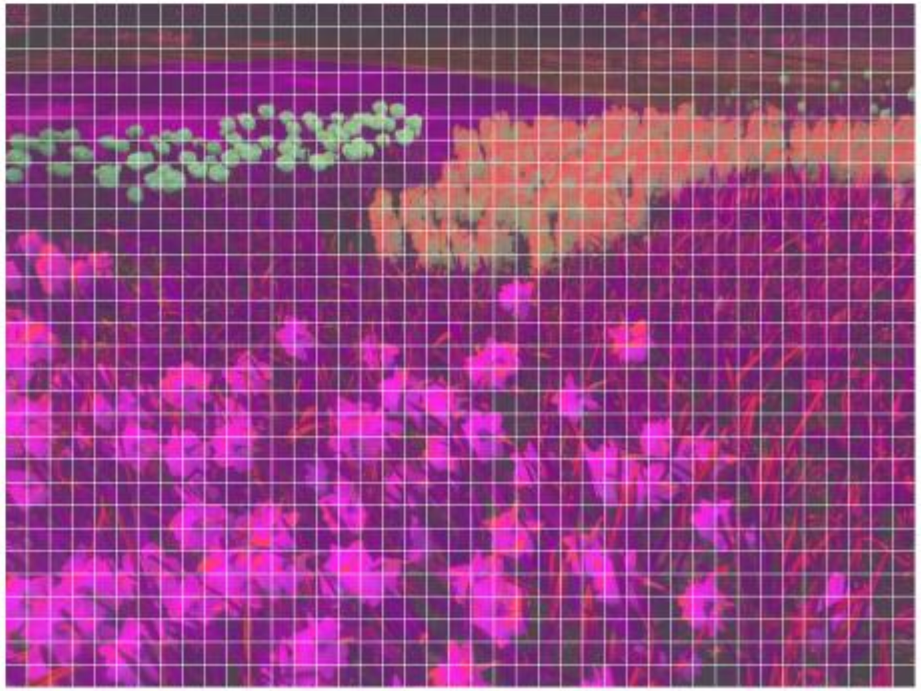


This is done to avoid placing them at an edge and to reduce the chances of choosing a noisy pixel

http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

SLIC Example

3. Move cluster centers to the lowest gradient position in a 3×3 neighborhood.



$$G(x,y) = \|I(x+1,y)-I(x-1,y)\|^2+\|I(x,y+1)-I(x,y-1)\|^2$$

- $I(x,y)$ is the lab vector corresponding to the pixel at position (x,y),
- $\|.\|$ is the L2 norm.

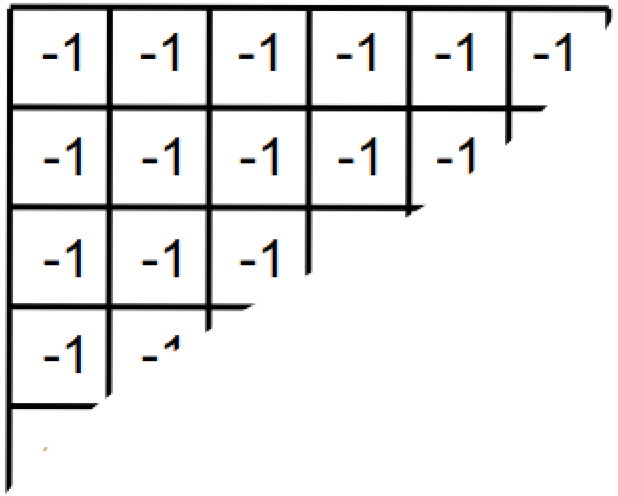
http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

SLIC Example

4. A 2D label matrix *L* as large as the input image will contain the superpixel each pixel belongs to. *L* is initialized with -1 for all pixels.

(meaning that each pixel belongs to no superpixel in the beginning)

$L=$



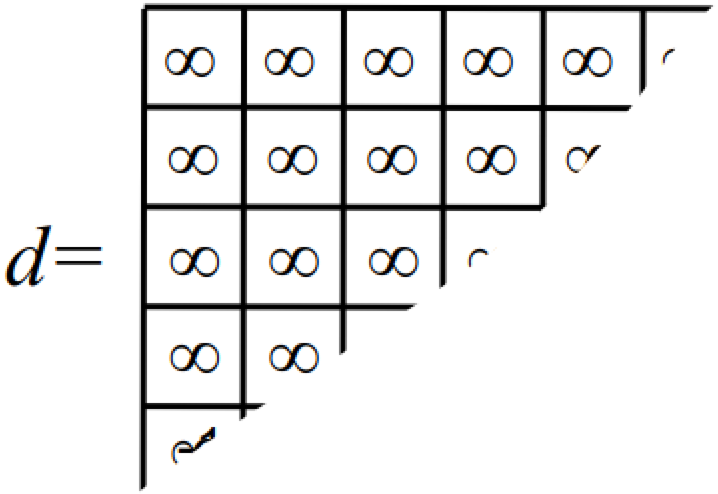
-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	
-1	-1	-1			
-1	-1				

http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

SLIC Example

5. A 2D distance matrix *d* as large as the input image will contain the distance of each pixel to the centroid of its superpixel. *d* is initialized with ∞ for all pixels.

(distance to superpixel centroid in the beginning)



$$d=\begin{array}{|c|c|c|c|c|} \hline \infty & \infty & \infty & \infty & \infty \\ \hline \infty & \infty & \infty & \infty & \\ \hline \infty & \infty & \infty & & \\ \hline \infty & \infty & & & \\ \hline \end{array}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$

$$D' = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2}$$

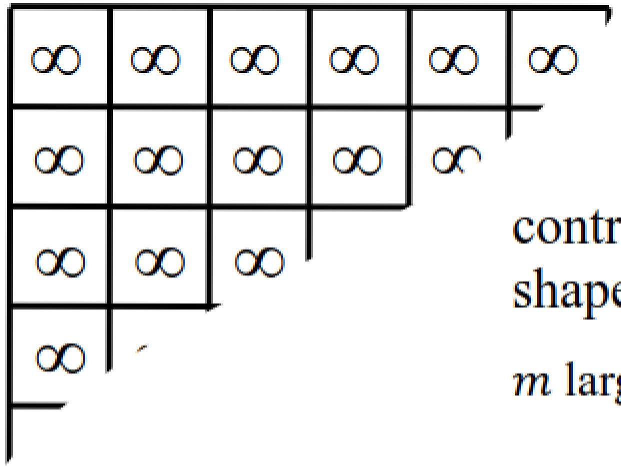
http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

SLIC Example

5. A 2D distance matrix *d* as large as the input image will contain the distance of each pixel to the centroid of its superpixel. *d* is initialized with ∞ for all pixels.

(distance to superpixel centroid in the beginning)

d=



∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	
∞	∞	∞			
∞					

$$D = \sqrt{(d_c)^2 + \left(\frac{d_s}{S}\right)^2 m^2}$$

controls the relative importance of shape and color

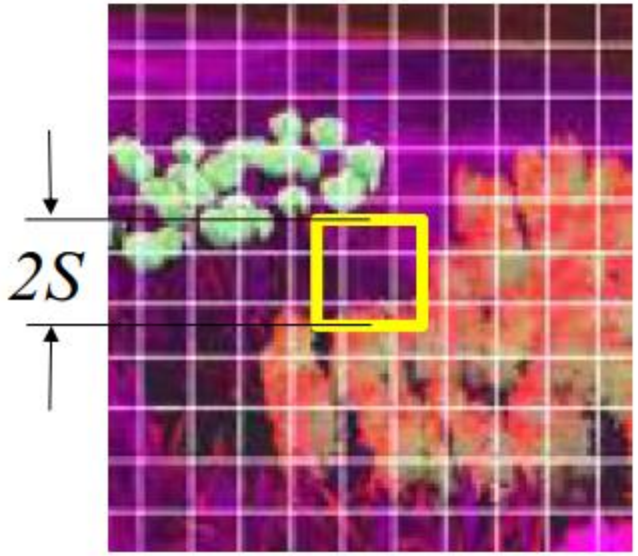
m large → favors more compact (lower area to perimeter ratio) superpixels.

m small → favors more adherence to edges.

http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

SLIC Example

```
6.  repeat
        for each cluster center $C_k$ do
            for each pixel $i$ in a $2S \times 2S$ region around $C_k$ do
                Compute the distance $D$ between $C_k$ and $i$.
                if $D < d(i)$ then
                    set $d(i) = D$
                    set $L(i) = k$
                end if
            end for
        end for
        compute new cluster centers.
        compute residual error $E$.
    until $E <$ threshold.
```



http://www.ele.puc-rio.br/~raul/ImageAnalysis/Superpixels.pdf

Example 1: image size = 735×980 pixels

$K = 1333$ superpixels; $m = 40$

More examples

Questions?