KMITL
BEYOND THE LIMIT

Computer Visions

01416500
Course Code

Image Segmentation

Overview

1. Basic Image segmentation
2. Distance Transforms
3. Watershed Algorithm
4. Fundamentals of image contours
5. Workshop#1

Asst. Prof. Dr. Anakkapon Saenthon
King Mongkut's Institute of Technology Ladkrabang

1. Basic Image segmentation

- **Image segmentation** is the process of partitioning a digital image into multiple segments.
- The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image.
- Each of the pixels in a region are similar with respect to some characteristic or computed property, such as **color**, **intensity**, **shape**, **or texture**.

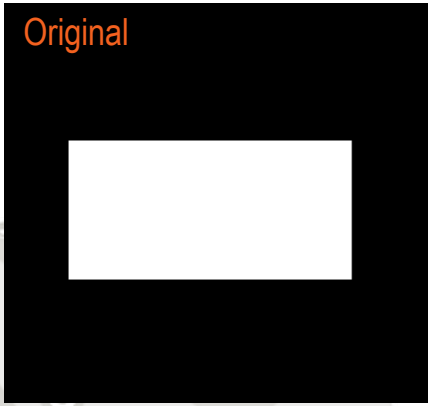
2. Distance Transform

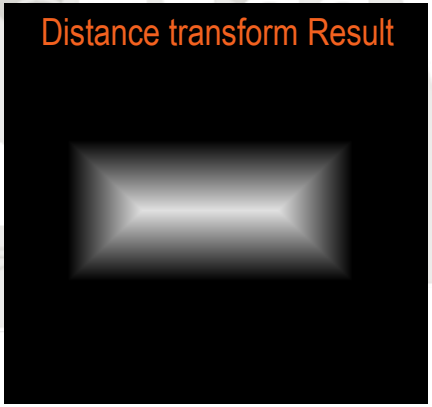
The **distance transform** is an operator normally **only applied to binary images**.

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

→

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0





The map labels each pixel of the image with the distance to the nearest obstacle pixel.

2. Distance Transform

How to calculate pixel's value of distance transforms ?

Matrix of original image

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

Matrix of the result image

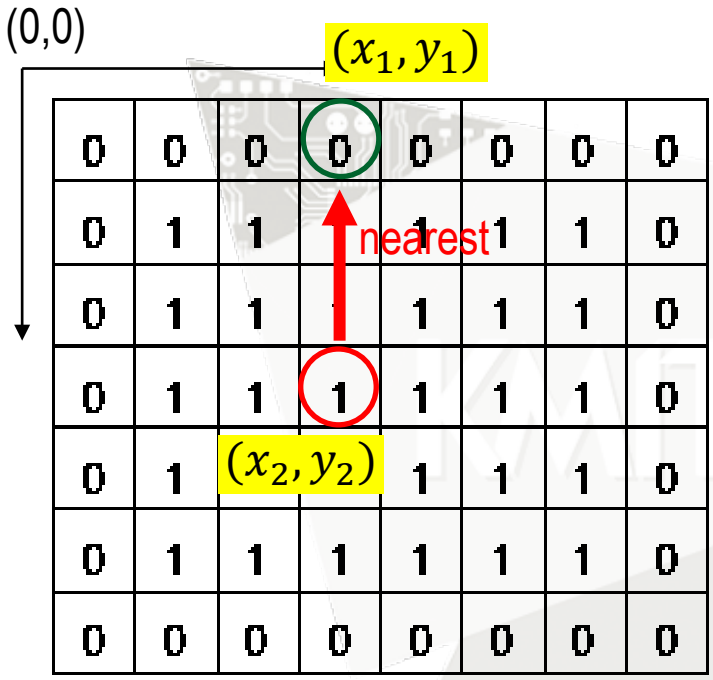
0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

This example change pixel value 1 to 3

Chessboard Distance

2. Distance Transform

How to calculate pixel's value of distance transforms ?



0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

$(x_2, y_2) = (3,3)$ $(x_1, y_1) = (3,0)$

Euclidean Distance

$$D_{Euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$D_{Euclid} = \sqrt{(3-3)^2 + (3-0)^2}$$

$$D_{Euclid} = \sqrt{(3)^2}$$

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

This example change pixel value 1 to 3

2. Distance Transform

There are several different sorts of distance transform, depending upon which distance metric is being used to determine the distance between pixels.

1. Euclidean Distance
(DIST_L2)

$$D_{Euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2. City Block Distance
(DIST_L1)

$$D_{City} = |x_2 - x_1| + |y_2 - y_1|$$

3. Chessboard Distance
(DIST_C)

$$D_{Chess} = \max(|x_2 - x_1|, |y_2 - y_1|)$$

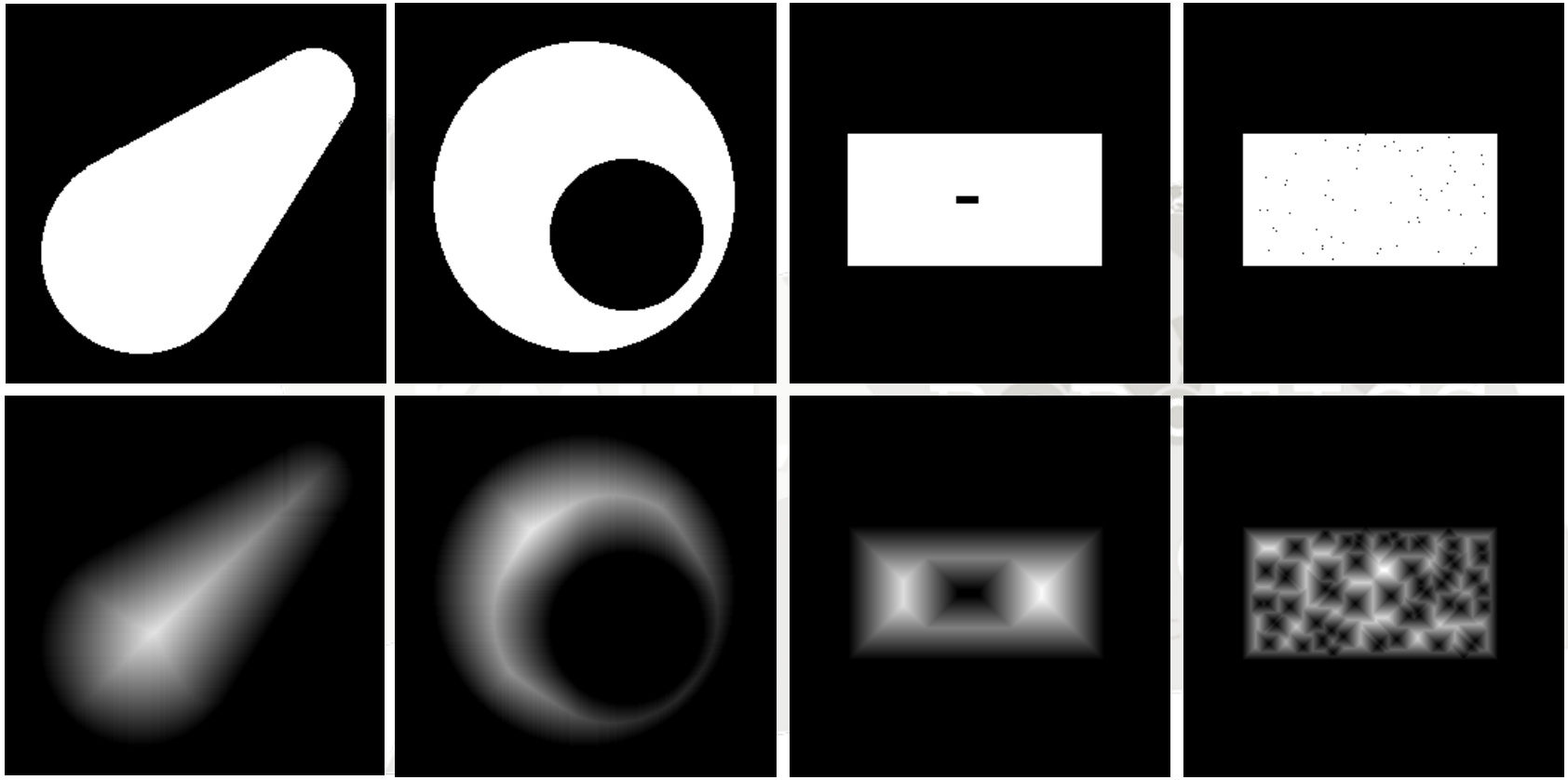
2. Distance Transforms

Distance types for Distance Transform and M-estimators

Enumerator	
DIST_L1 Python: cv.**DIST_L1**	distance = |x1-x2| + |y1-y2|
DIST_L2 Python: cv.**DIST_L2**	the simple euclidean distance
DIST_C Python: cv.**DIST_C**	distance = max(|x1-x2|,|y1-y2|)
DIST_L12 Python: cv.**DIST_L12**	L1-L2 metric: distance = 2(sqrt(1+x*x/2) - 1))
DIST_FAIR Python: cv.**DIST_FAIR**	distance = c^2(|x|/c-log(1+|x|/c)), c = 1.3998
DIST_WELSCH Python: cv.**DIST_WELSCH**	distance = c^2/2(1-exp(-(x/c)^2)), c = 2.9846
DIST_HUBER Python: cv.**DIST_HUBER**	distance = |x|<c ? x^2/2 : c(|x|-c/2), c=1.345

2. Distance Transform

This is applied the ***Euclidean distance*** with some examples.



ref : https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm#:~:text=The%20distance%20transform%20is%20an,closest%20boundary%20from%20each%20point.

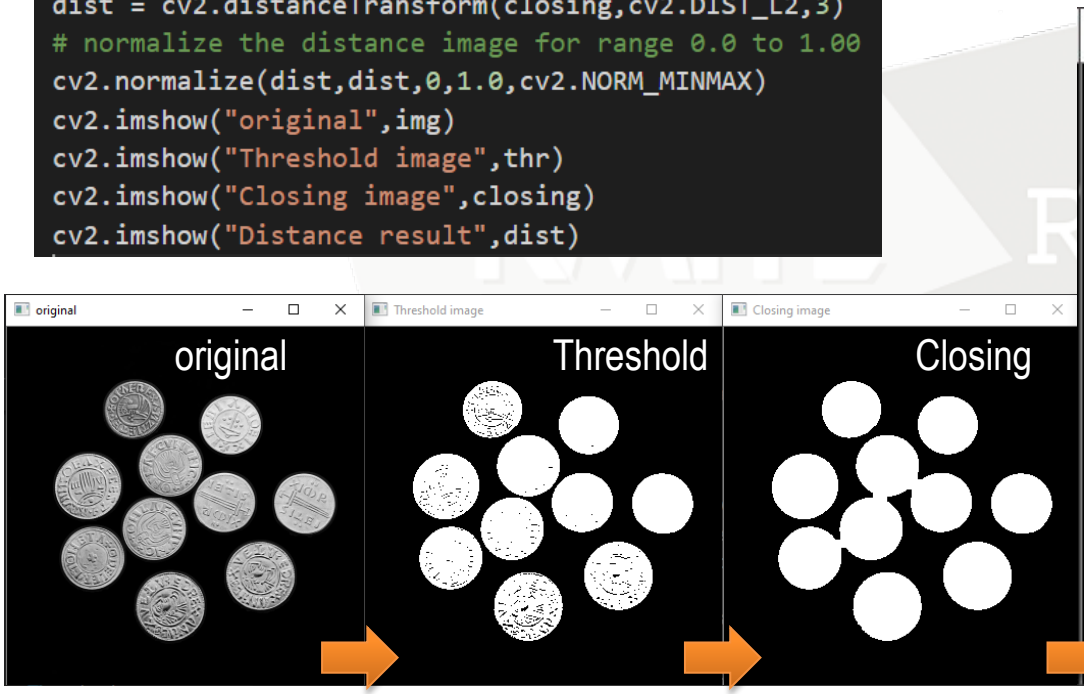
2. Distance Transforms

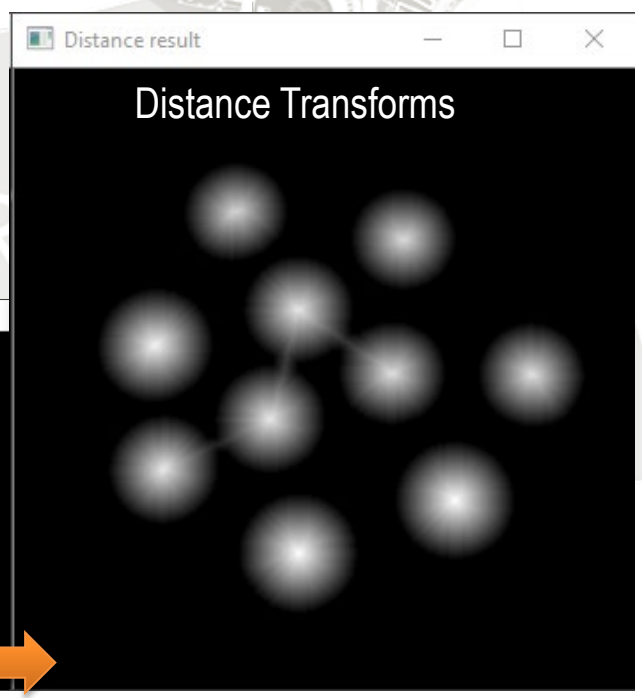
Example of Distance Transforms

```
_,thr = cv2.threshold(img,80,255,cv2.THRESH_BINARY)
# closing
closing = cv2.morphologyEx(thr,cv2.MORPH_CLOSE,
                           np.ones((3,3),np.uint8),
                           iterations=2)
# applied distance transforms
dist = cv2.distanceTransform(closing,cv2.DIST_L2,3)
# normalize the distance image for range 0.0 to 1.00
cv2.normalize(dist,dist,0,1.0,cv2.NORM_MINMAX)
cv2.imshow("original",img)
cv2.imshow("Threshold image",thr)
cv2.imshow("Closing image",closing)
cv2.imshow("Distance result",dist)
```

OpenCV function :

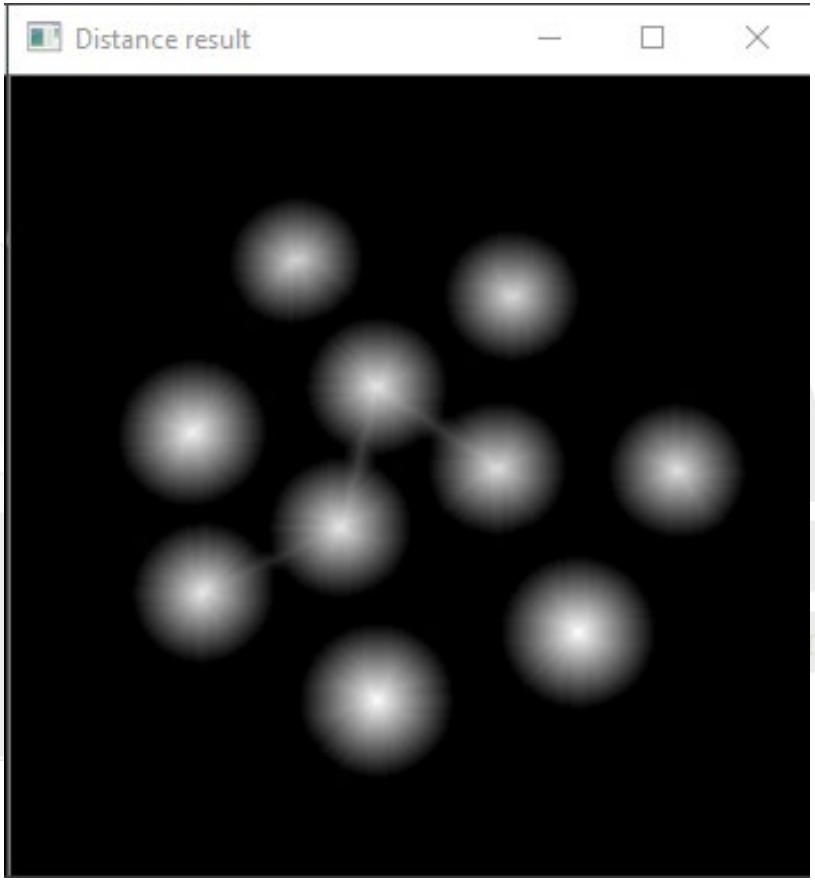
cv2.distanceTransform()





2. Distance Transforms

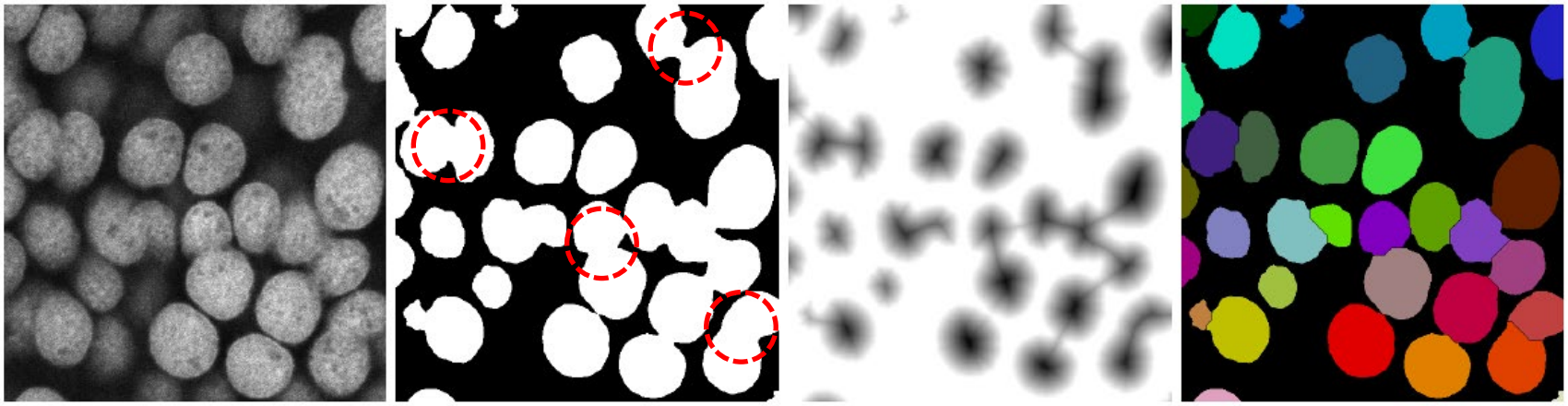
Quiz#2 Let's show the distance transforms result. "coins.jpg"



3.Watershed Algorithm

a **watershed** is a transformation defined on a grayscale image that considered as **a topographic surface**. .

Example watershed

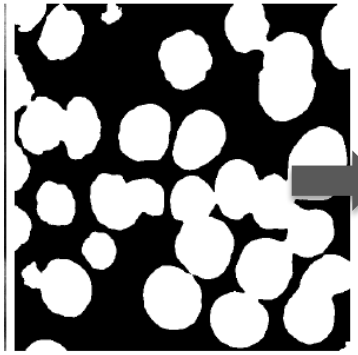


A classic way of **separating touching objects** in binary images makes use of the *distance transform* and the *watershed method*

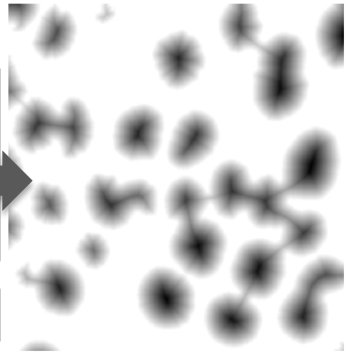
3.Watershed Algorithm

Concept of Watershed Algorithms

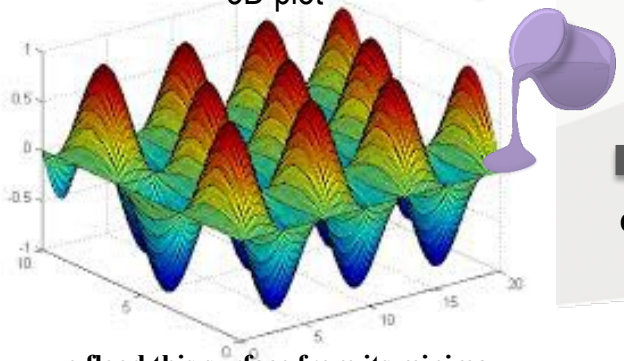
Distance Binary image



Distance transforms

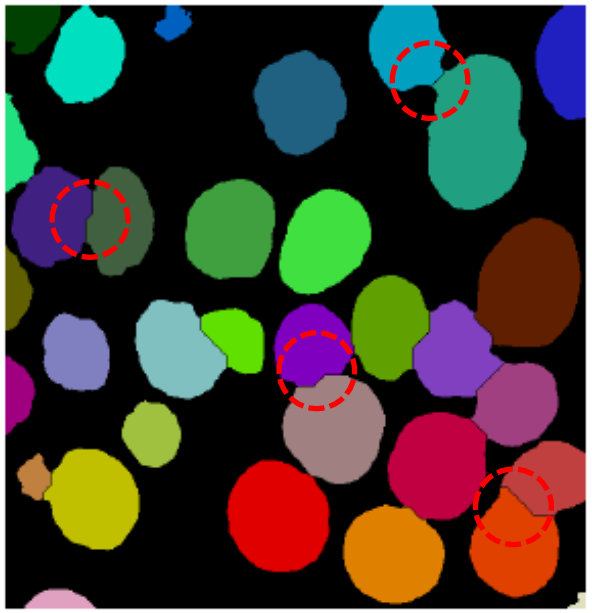


3D plot

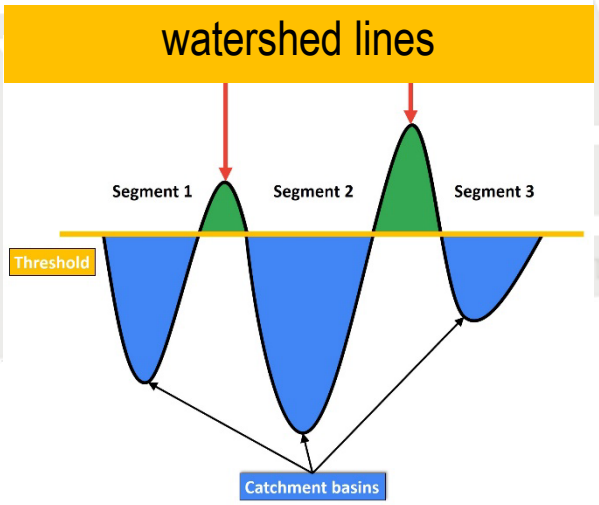


we flood this surface from its minima.

defined set of markers

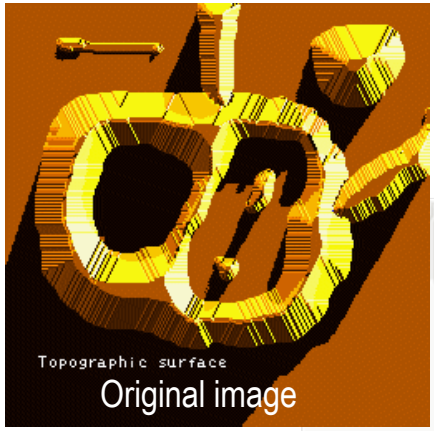


Final watersheds

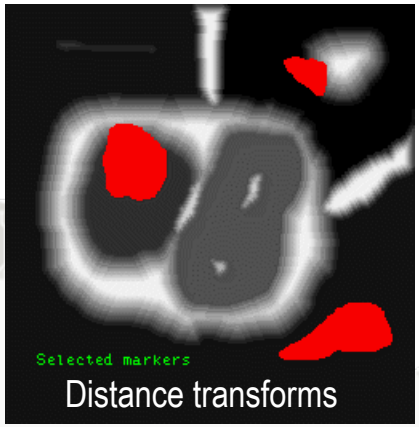


3. Watershed Algorithm

Example of Watershed Algorithms

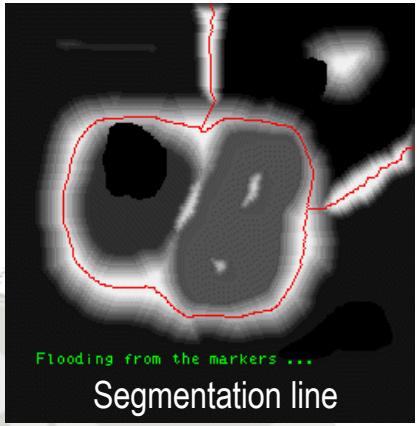


Original image



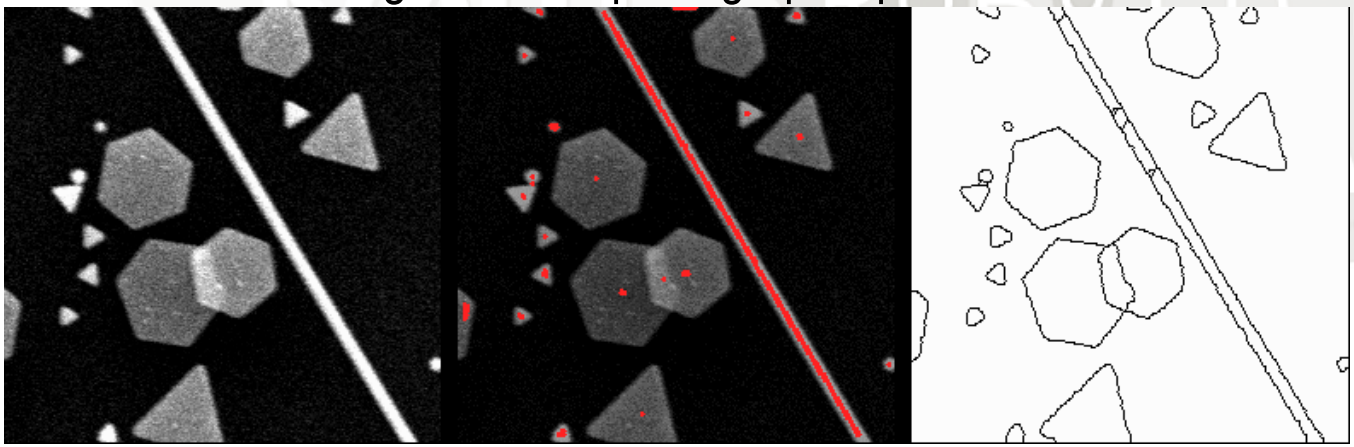
Distance transforms

Flooding masker



Segmentation line

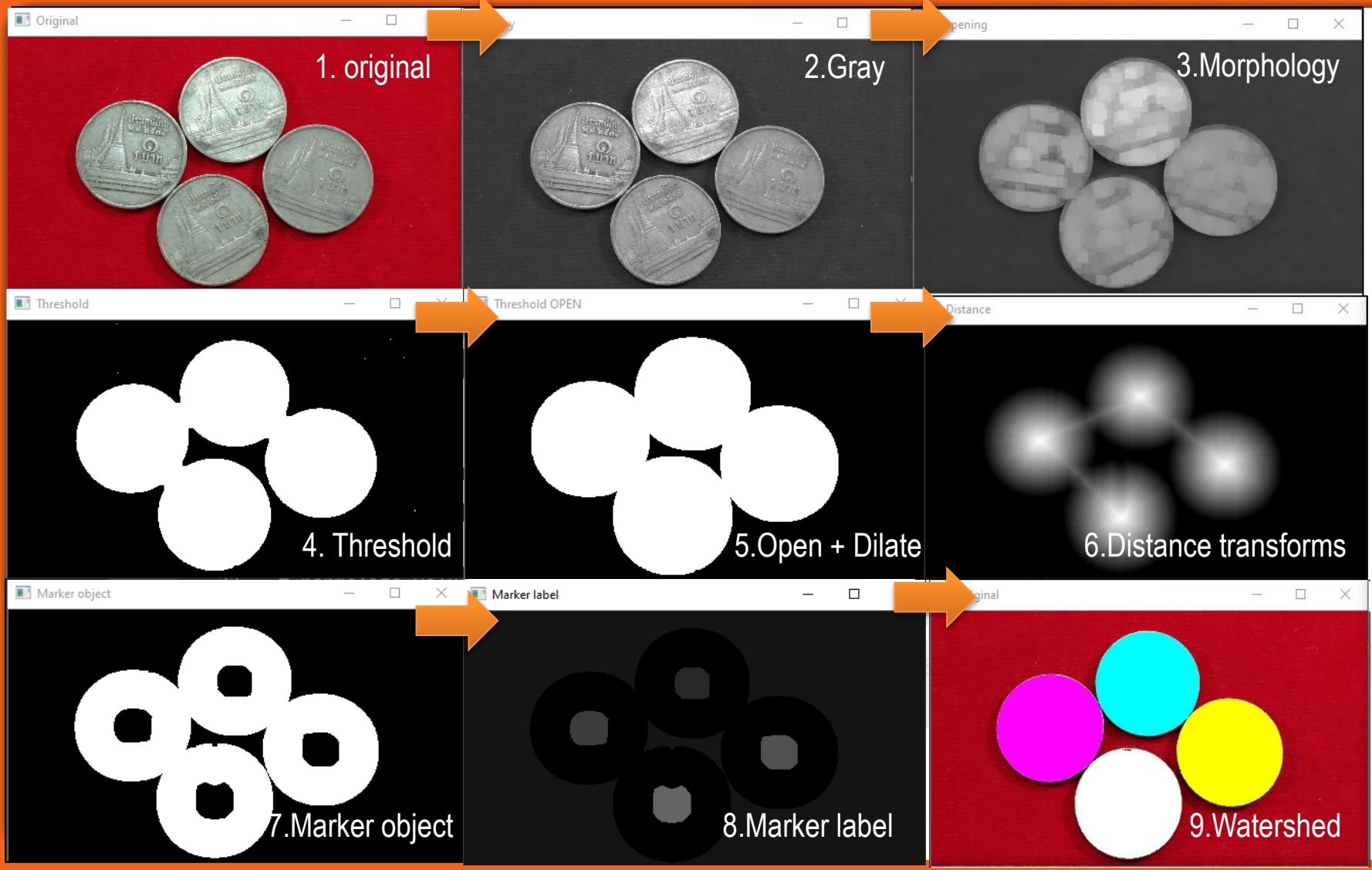
Silver grains on a photographic plate



3. Watershed Algorithm

OpenCV function :

cv2.watershed()



3. Watershed Algorithm

connectedComponents functions

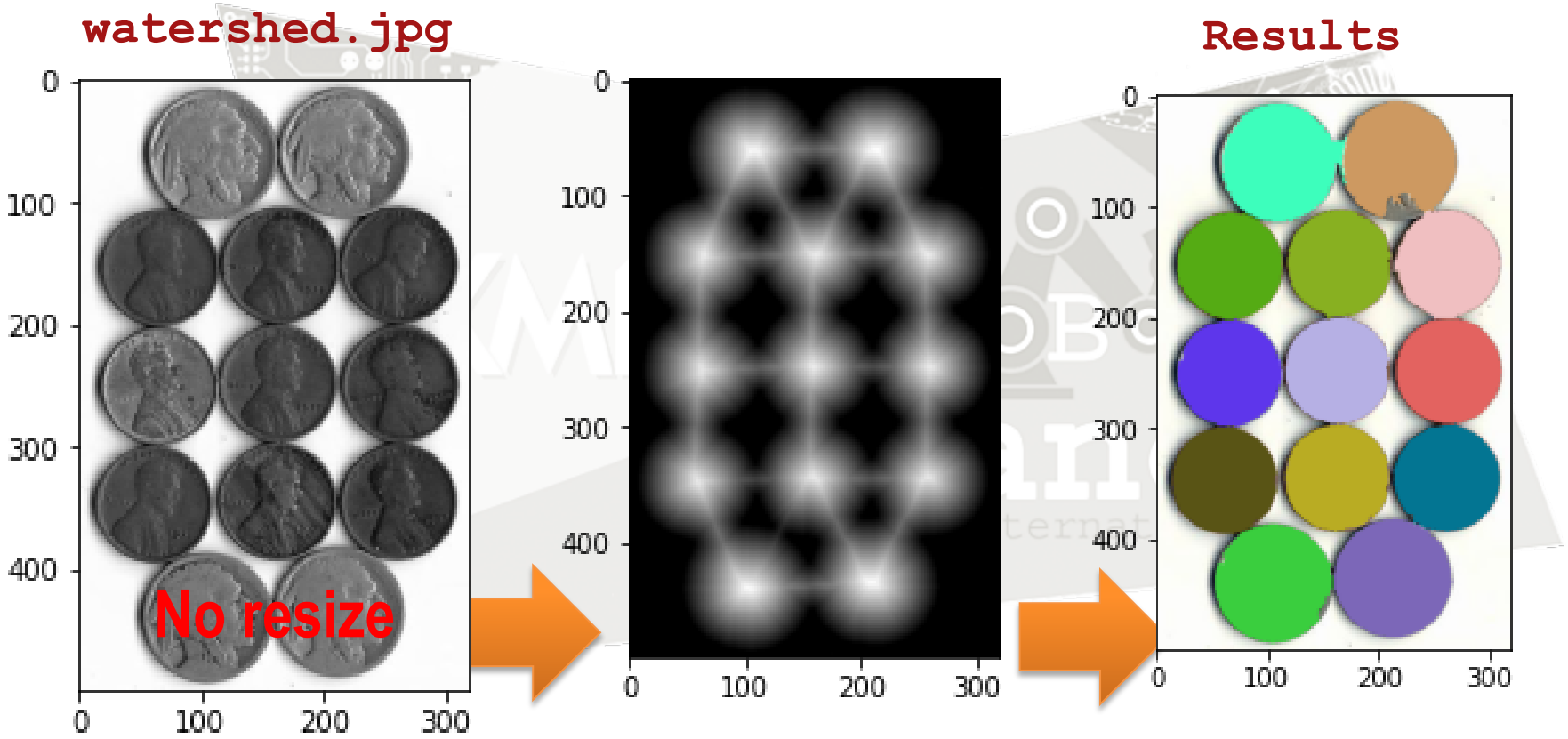
num_labels, labels_im = cv2.connectedComponents(img)

It is used to labeling object in binary image

Quiz 20 min

Let's show how to use the watershed technique to segmentation ?

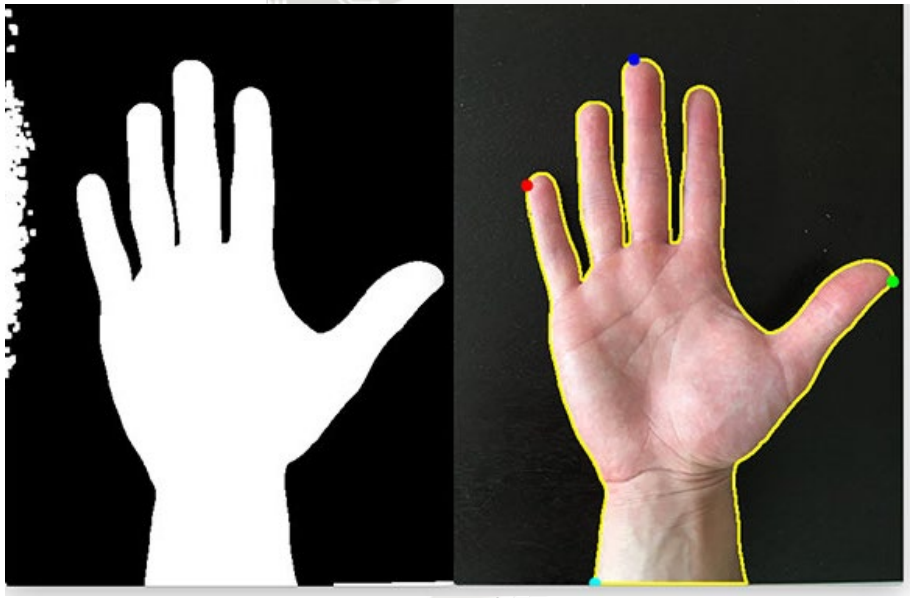
- Find the correct parameters

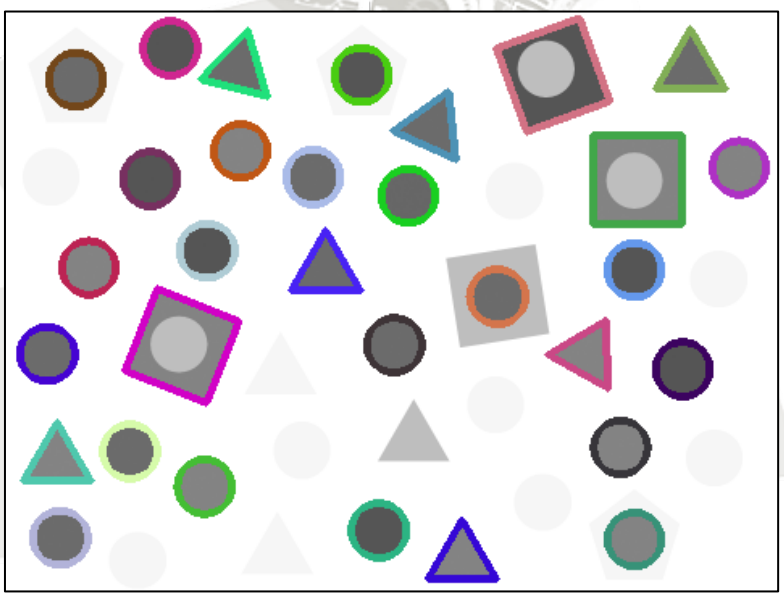


4. Fundamentals of image contours

What are contours?

Contours can be explained simply as **a curve joining all the continuous points** (along the boundary), having same color or intensity.





4. Fundamentals of image contours

OpenCV function :

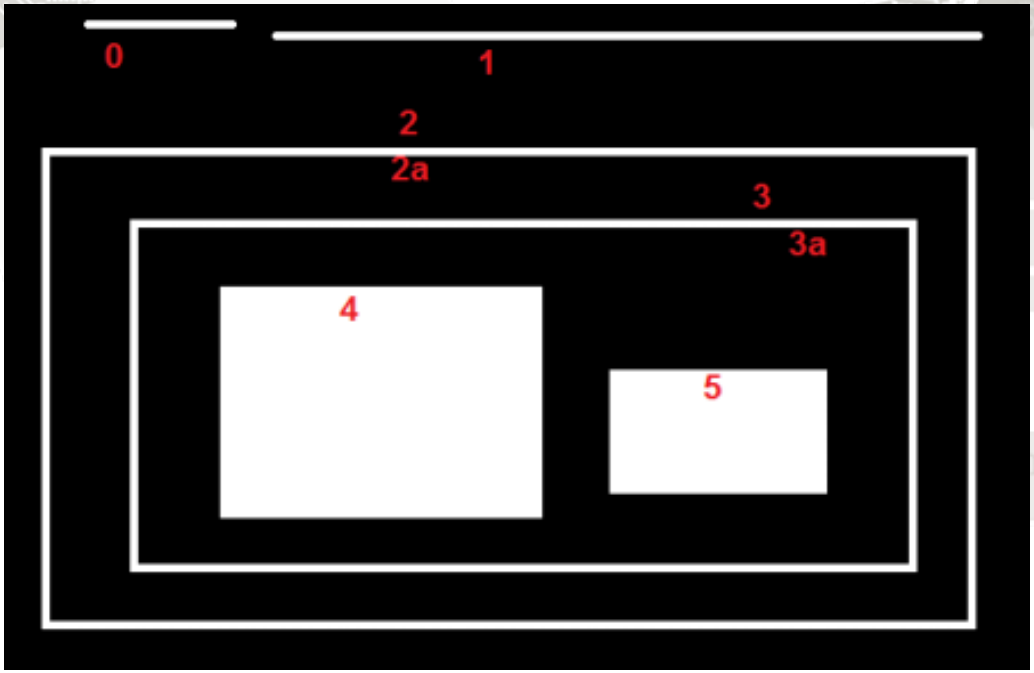
cv2.findContours (image, **hierarchy**, method)

image	Source, an 8-bit single-channel image.
hierarchy	Optional output vector have 4 types **RETR_LIST, RETR_EXTERNAL,** **RETR_CCOMP, RETR_TREE**
method	Contour approximation method **CHAIN_APPROX_NONE ,CHAIN_APPROX_SIMPLE**

4. Fundamentals of image contours

What is Hierarchy?

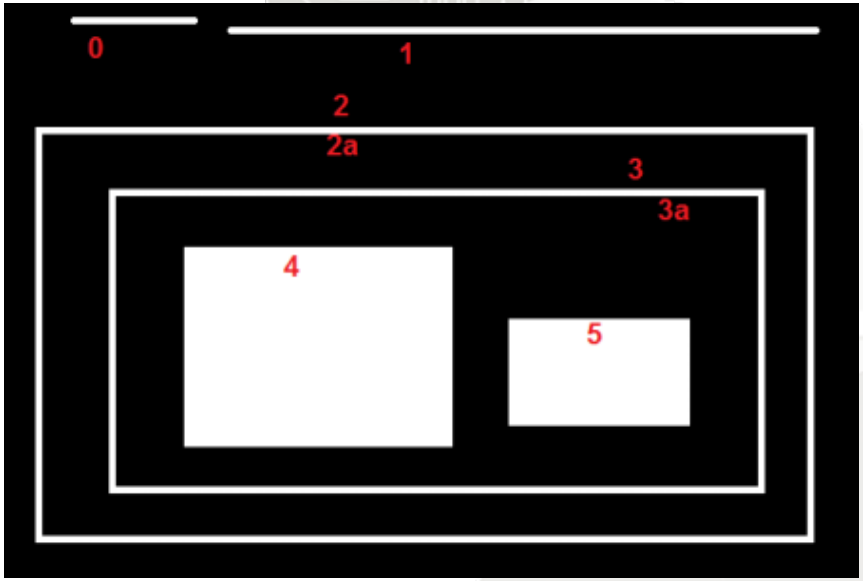
Normally, **some shapes are inside other shapes**. Just like nested figures. In this case, we call outer one as parent and inner one as child.



4. Fundamentals of image contours

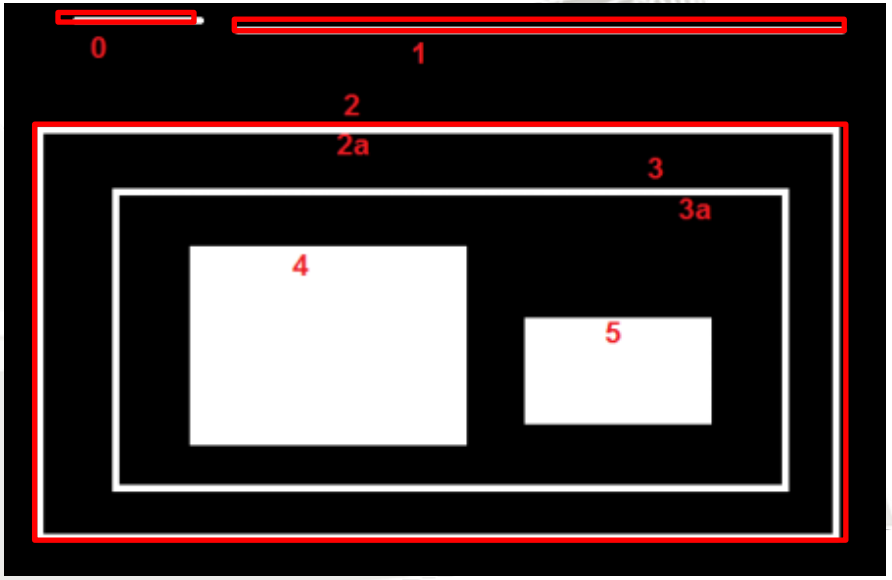
What is 4th Contour Retrieval Mode in OpenCV ?

1. RETR_LIST [0:5]



All object : 6 contours

2. RETR_EXTERNAL [0,1,2]

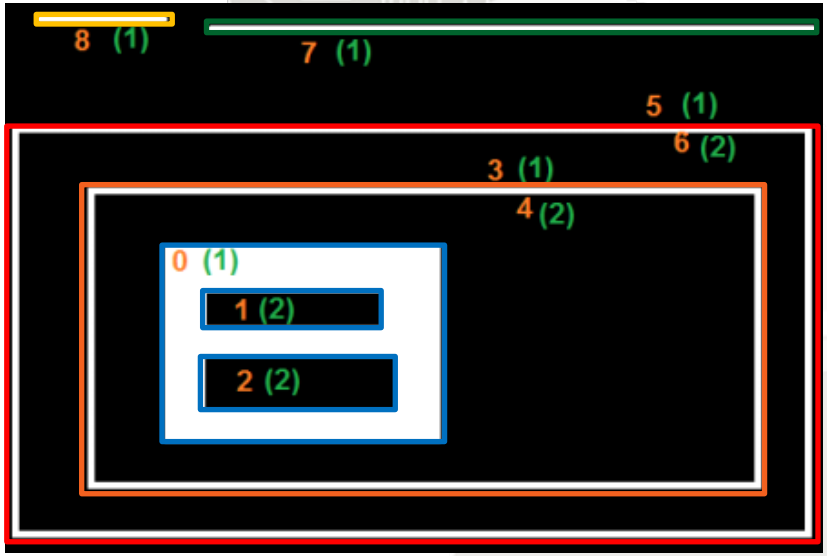


Only outer :3 contours

4. Fundamentals of image contours

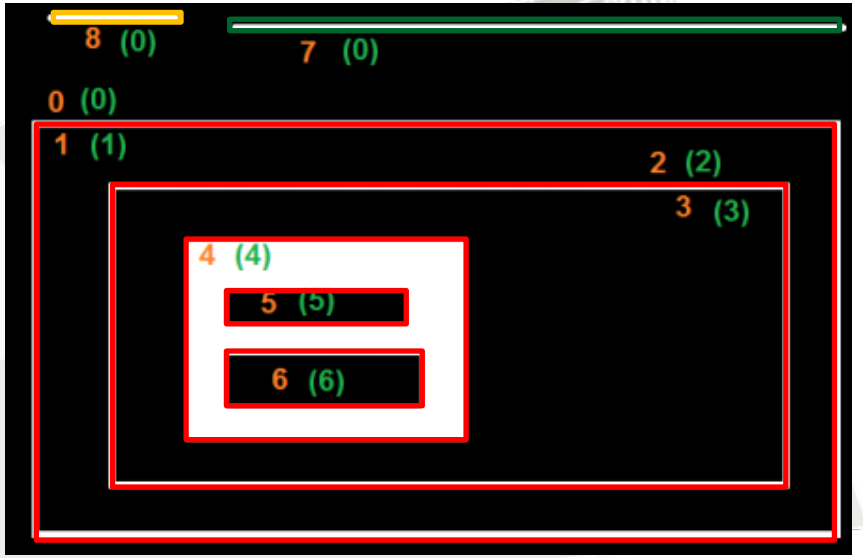
What is 4th Contour Retrieval Mode in OpenCV ?

3. RETR_CCOMP (2-level)



5 contours

4. RETR_TREE



3 contours

4. Fundamentals of image contours

OpenCV function :

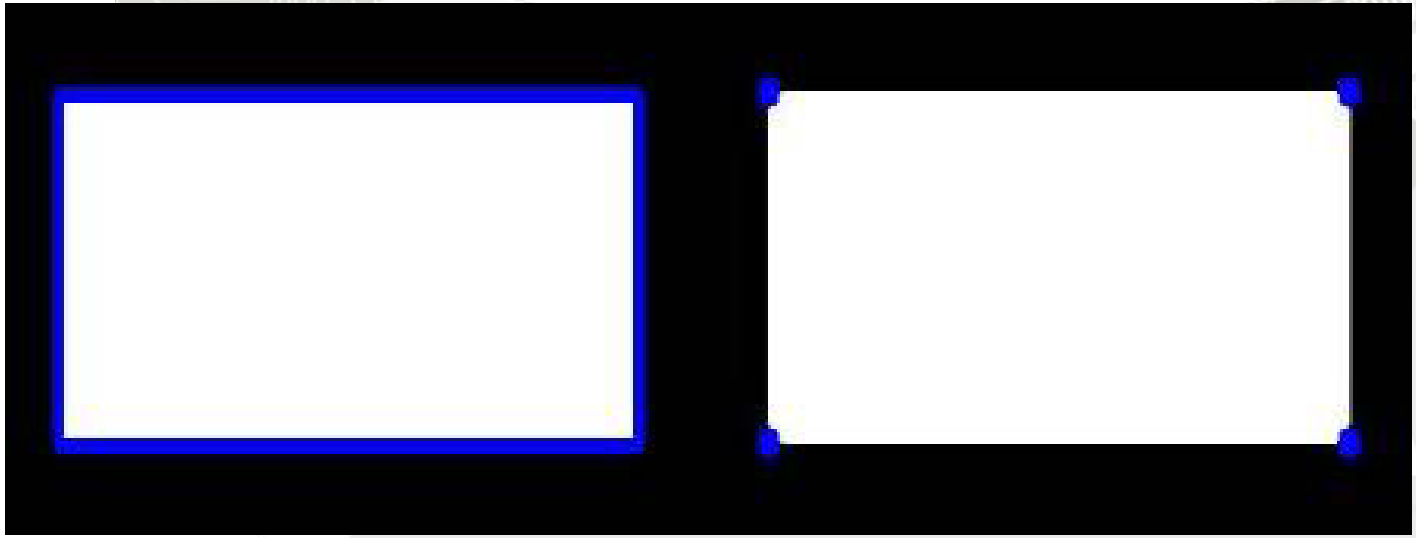
cv2.findContours (image, hierarchy, **method**)

image	Source, an 8-bit single-channel image.
hierarchy	Optional output vector
	RETR_LIST, RETR_EXTERNAL,
	RETR_CCOMP, RETR_TREE
method	Contour approximation method
	CHAIN_APPROX_NONE ,CHAIN_APPROX_SIMPLE

4. Fundamentals of image contours

Contour Approximation Method

- First image shows points I got with cv2.CHAIN_APPROX_NONE (734 points)
  Second image shows the one with cv2.CHAIN_APPROX_SIMPLE (only 4 points).



OpenCV function :
cv2.CHAIN_APPROX_NONE

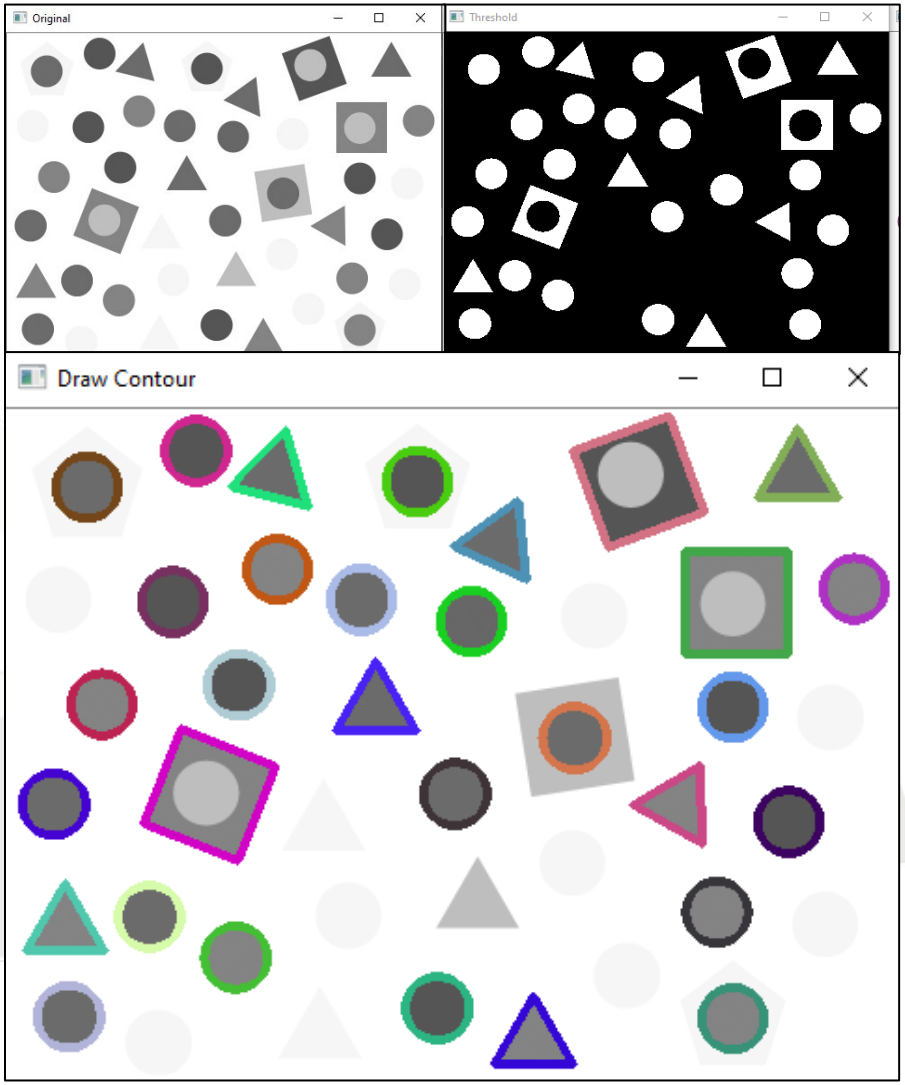
OpenCV function :
cv2.CHAIN_APPROX_SIMPLE

4. Fundamentals of image contours

Example
Find Contour

```
img2 = img.copy()
contour,hierarchy= cv2.findContours(
                        bw,
                        cv2.RETR_LIST,
                        cv2.CHAIN_APPROX_NONE)

print(len(contour))
i = 0
cv2.drawContours(img2,contour,i,(255,0,0),thickness=3)
plt.imshow(img2)
```



4. Fundamentals of image contours

How to draw the contours?

To draw the contours, **cv.drawContours** function is used. It can also be used to draw any shape provided you have its boundary points.

- To draw all the contours in an image:

```
cv.drawContours(img, contours, -1, (0,255,0), 3)
```

- To draw an individual contour, say 4th contour:

```
cv.drawContours(img, contours, 3, (0,255,0), 3)
```

- But most of the time, below method will be useful:

```
cnt = contours[4]
cv.drawContours(img, [cnt], 0, (0,255,0), 3)
```

Contour Area

Contour area is given by the function **cv.contourArea()** or from moments

```
area = cv2.contourArea(cnt)
```

Contour Perimeter

It is also called arc length. It can be found out using **cv.arcLength()** function. Second argument specify whether shape is a closed contour (if passed True), or just a curve.

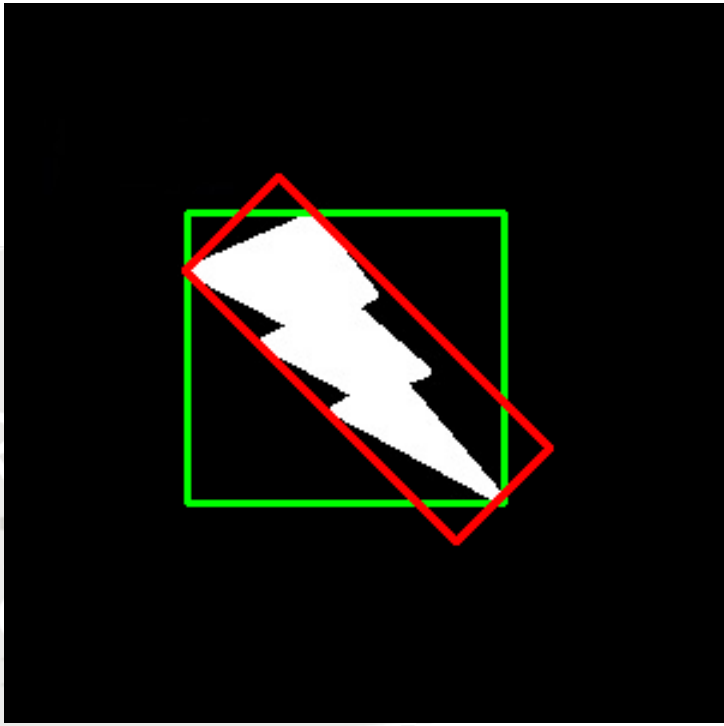
```
perimeter = cv2.arcLength(cnt,True)
```

4. Fundamentals of image contours

Bounding Rectangle

Straight Bounding Rectangle

```
x,y,w,h = cv2.boundingRect(cnt)
cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```



Rotated Rectangle

```
rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.int0(box)
cv2.drawContours(img,[box],0,(0,0,255),2)
```

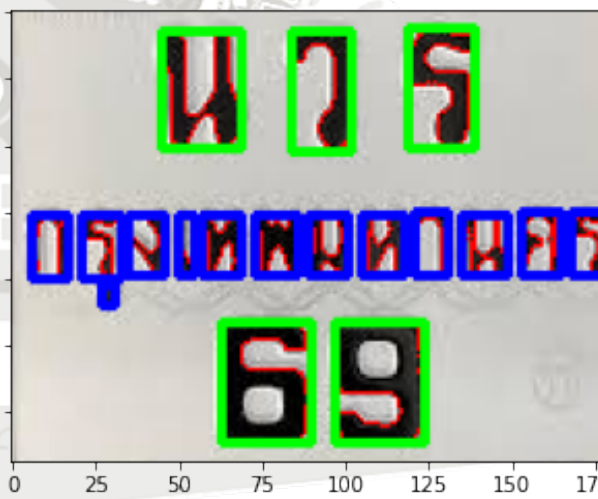
Quiz (15 min)

Let's you show the rectangle of text objects with green color and thickness = 2 pixels .

"69.jpg"


```
img = img[50: 185,20:205]
# crop
```

Output

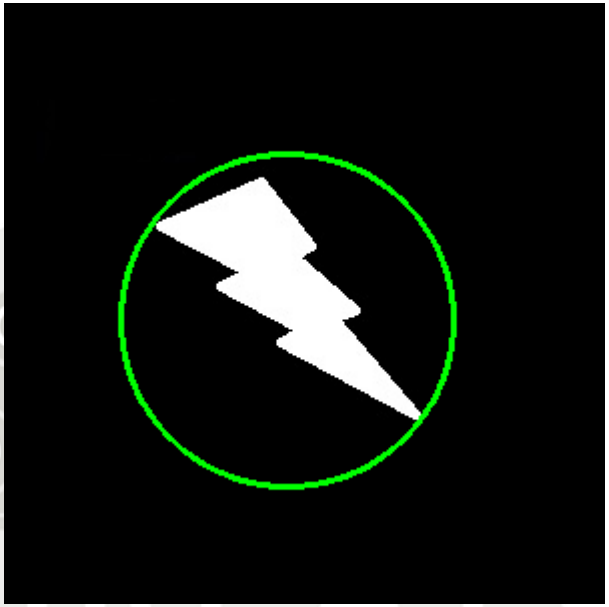


4. Fundamentals of image contours

Minimum Enclosing Circle

Next we find the circumcircle of an object using the function **cv2.minEnclosingCircle()**.

It is a circle which completely covers the object with ***minimum area***.

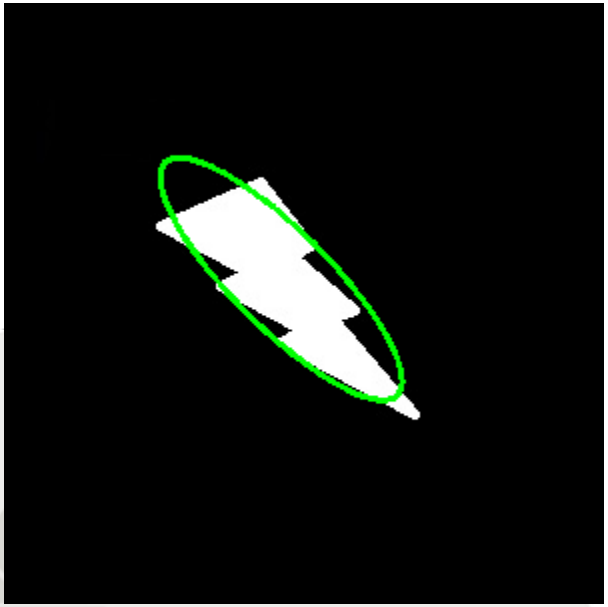


```
(x,y),radius = cv2.minEnclosingCircle(cnt)
center = (int(x),int(y))
radius = int(radius)
cv2.circle(img,center,radius,(0,255,0),2)
```

4. Fundamentals of image contours

Fitting an Ellipse

Next one is to fit an ellipse to an object. It returns the rotated rectangle in which the ellipse is inscribed.

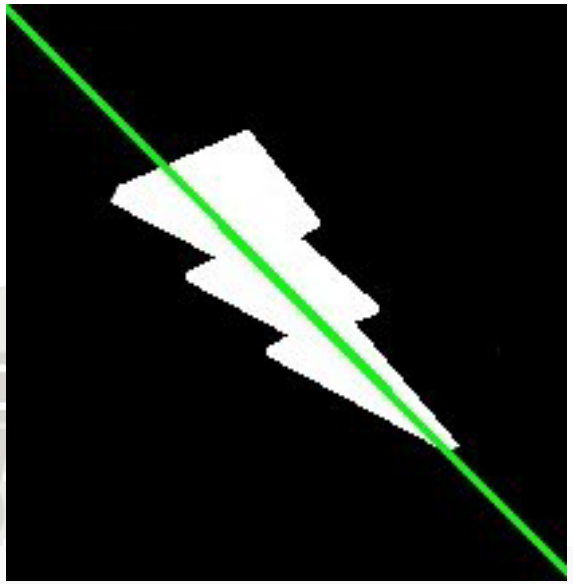


```
ellipse = cv2.fitEllipse(cnt)
cv2.ellipse(img,ellipse,(0,255,0),2)
```

4. Fundamentals of image contours

Fitting a Line

we can fit a line to a set of points. Below image contains a set of white points. We can approximate a straight line to it.



```
rows,cols = img.shape[:2]
[vx,vy,x,y] = cv2.fitLine(cnt, cv2.DIST_L2,0,0.01,0.01)
lefty       = ((-x*vy/vx) + y)
righty      = (((cols-x)*vy/vx)+y)
cv2.line(img,(cols-1,righty),(0,lefty),(0,255,0),2)
```