A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light greenish-blue. They are positioned diagonally, with the blue one in front of the green one.

# Automatic Number Plate Recognition (Python)

Deepak Goyal



# Contours

- Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity
- In other words, A contour is a closed curve joining all the continuous points having some color or intensity, they represent the shapes of objects found in an image



# Difference between Edge Detection and Contours Detection

The aim of Contour detection is surrounded by the determination of shapes of closed objects particularly because for the continual points having the same color intensity the method to hunt out the contours is to ascertain whereas edge detection is carried by detecting the change within the color intensity.

Edge detection is administered for the whole image whereas contour detection is administered just for the objects within the image.



# Contours Detection

To do contours detection, OpenCV provides a function called *FindContours* which intent to find contours in the image. Of course to some treatment should be applied to the picture to get a good contours detection.

```
cv2.findContours(.....)
```



# Arguments in this function

- 1) **Image** : The input image for which we want to find contours.
- 2) **Contour Retrieval Mode**: It can take 4 type of values.
  - a) RETR\_LIST
  - b) RETR\_EXTERNAL
  - c) RETR\_CCOMP
  - d) RETR\_TREE
- 3) **Contour Approximation Method**: It can take 2 type of values.
  - a) CHAIN\_APPROX\_NONE
  - b) CHAIN\_APPROX\_SIMPLE

Before going in details, Let's understand what is hierarchy



# Hierarchy

- Sometimes objects are in different locations. But in some cases, some shapes are inside other shapes.
- In the nested figures, we call outer one as **parent** and inner one as **child**. This way, contours in an image has some relationship to each other.
- And we can specify how one contour is connected to each other, like, is it child of some other contour, or is it a parent etc.
- Representation of this relationship is called the **Hierarchy**.



# Hierarchy Array

So each contour has its own information regarding what hierarchy it is, who is its child, who is its parent etc.

OpenCV represents it as an array of four values :

**[Next, Previous, First\_Child, Parent]**

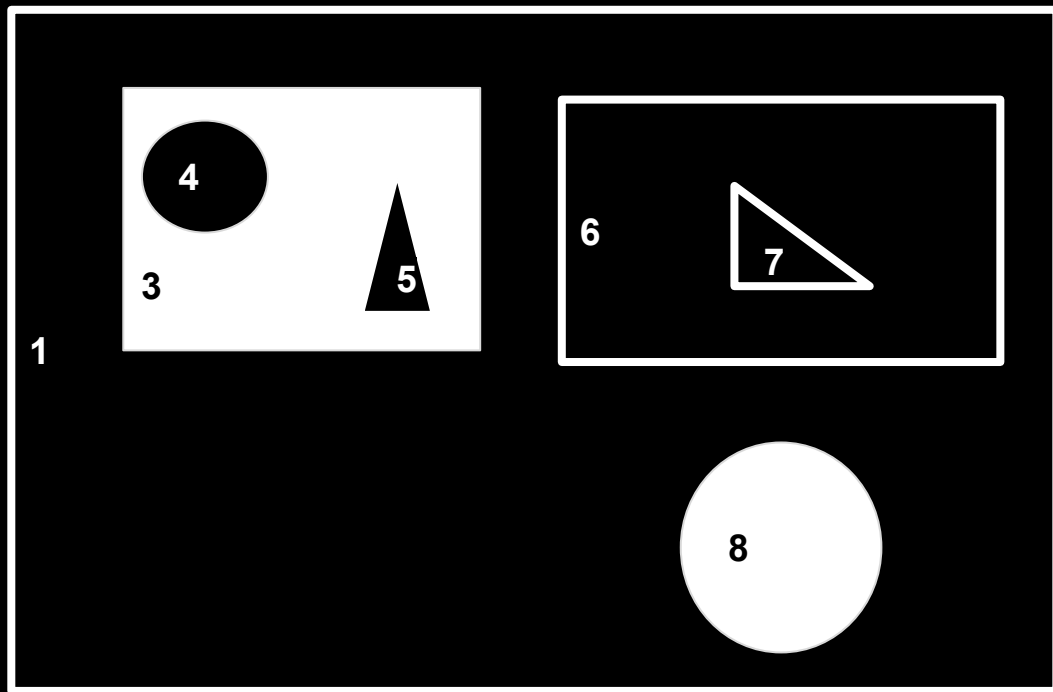
**Next:** Next element in hierarchy

**Previous:** Previous Element in hierarchy

**First\_Child:** The First immediate child in the hierarchy.

**Parent:** The immediate Parent in hierarchy

If any value is not present , then it return -1







# 1. RETR\_LIST

This is the simplest of the four flags (from explanation point of view). It simply retrieves all the contours, but doesn't create any parent-child relationship. **Parents and kids are equal under this rule, and they are just contours.** ie they all belongs to same hierarchy level.

Therefore all contours will be there in the list.

In above diagram, 1,2,3,4,5,6,7, and 8 all the contours will be in the list.



## 2. RETR\_EXTERNAL

If you use this flag, it returns only extreme outer flags. All child contours are left behind. **We can say, under this law, Only the eldest in every family is taken care of. It doesn't care about other members of the family**

Therefore contours 1 and 2 will be there.



### 3. RETR\_CCOMP

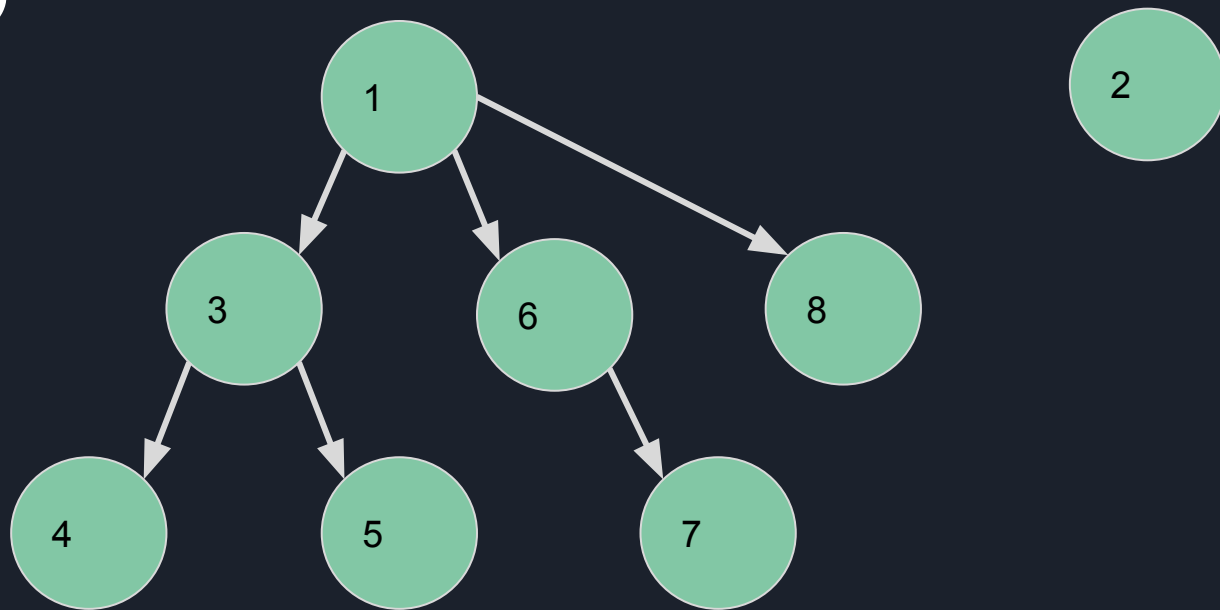
This flag retrieves all the contours and arranges them to a 2-level hierarchy. ie external contours of the object (ie its boundary) are placed in hierarchy-1. And the contours of holes inside object (if any) is placed in hierarchy-2. If any object inside it, its contour is placed again in hierarchy-1 only. And its hole in hierarchy-2 and so on.

Hierarchy 1: 1, 2, 3, 6

Hierarchy 2: 4, 5, 7, 8

## 4. RETR\_TREE

It retrieves all the contours and creates a full family hierarchy list. It even tells, who is the grandpa, father, son, grandson and even beyond... :)





# Argument 3: Contour Approximation Method


Contours are the boundaries of a shape with same intensity. It stores the (x,y) coordinates of the boundary of a shape. But does it store all the coordinates ? That is specified by this contour approximation method.

1. `cv2.CHAIN_APPROX_NONE`
2. `cv2.CHAIN_APPROX_SIMPLE`

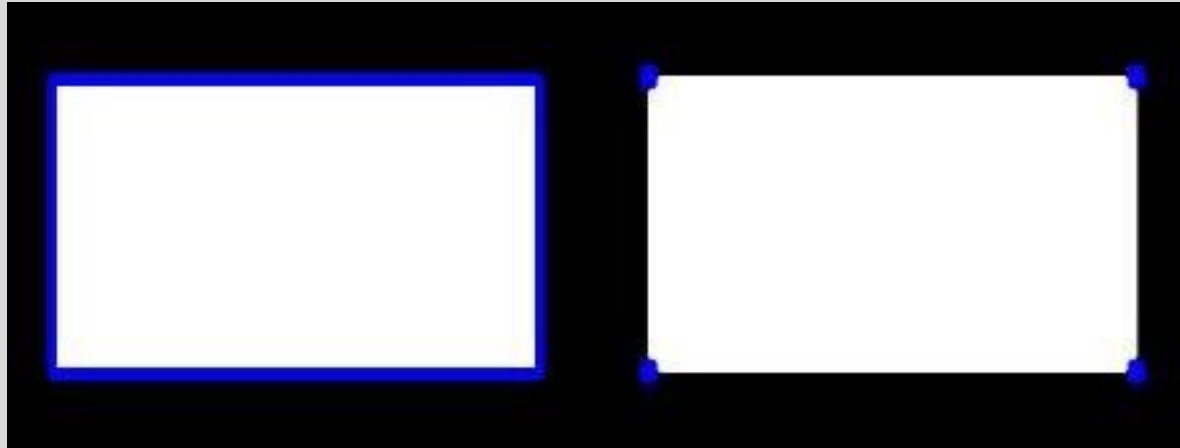


## `cv2.CHAIN_APPROX_NONE` and `cv2.CHAIN_APPROX_SIMPLE`

If we pass `cv2.CHAIN_APPROX_NONE`, all the boundary points are stored. But actually, do we need all the points? For eg, if we have to find the contour of a straight line. We need just two endpoints of that line. This is what `cv2.CHAIN_APPROX_SIMPLE` does. It removes all redundant points and compresses the contour, thereby saving memory.



Below image of a rectangle demonstrate this technique. Just draw a circle on all the coordinates in the contour array (drawn in blue color). First image shows points I got with `cv2.CHAIN_APPROX_NONE` (734 points) and second image shows the one with `cv2.CHAIN_APPROX_SIMPLE` (only 4 points). See, how much memory it saves!!!






# Draw Contours

To draw the contours, `cv2.drawContours` function is used. It can also be used to draw any shape provided you have its boundary points.

- 1st argument is the source image . Notice that in this we need to create a copy of image because this function will change the image content so to preserve the original image we make a copy of this.
- 2nd argument is the list of contours.



- 
- 3rd argument is value that contour we have to draw. If we pass -1 in this then this will draw all contours
  - 4th argument is color in rgb format.
  - 5th argument is the thickness of drawn contours.



# Haar cascades

Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

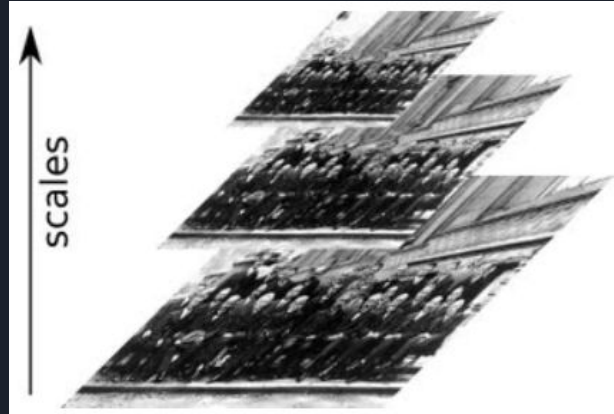
- **Positive images** – These images contain the images which we want our classifier to identify.
- **Negative Images** – Images of everything else, which do not contain the object we want to detect.
- 


It can be downloaded from this link->

<https://github.com/opencv/opencv/tree/master/data/haarcascades>

# detectMultiScale method

1. Image: First argument is our image
2. scaleFactor: Parameter specifying how much the image size is reduced at each image scale.





This scale factor is used to create scale pyramid as shown in the picture. Suppose, the scale factor is 1.03, it means we're using a small step for resizing, i.e. reduce size by 3 %, we increase the chance of a matching size with the model for detection is found, while it's expensive.

3. **minNeighbors** : Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality

End