Marker based localization

Contours and Hierarchy

Team members

Niharika Jayanthi Dheeraj Kamath

Under the guidance of
Sanam Shakya

Goal

In this chapter we will learn,
* Understand what contours and hierarchy mean.
* Learn about finding and drawing contours.
* We will see: **cv2.findContours()**, **cv2.drawContours()**

Theory

A contour is basically a line or a curve having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.
* For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.
* `cv2.findContours()` function modifies the source image. So if you want source image even after finding contours, store it to some other variables.
* In OpenCV, finding contours is like finding white object from black background. So remember, object to be found should be white and background should be black.

Function

`cv2.findContours(image, mode, method[, contours[, hierarchy[, offset]]])` → contours, hierarchy

Parameters

- **image** – 8 bit single channel source image(binary image)
- **mode** –
  1. `CV_RETR_EXTERNAL`
  2. `CV_RETR_LIST`
  3. `CV_RETR_CCOMP`
  4. `CV_RETR_TREE`

  Learn about the different modes in detail in hierarchy given in the document further below.
- **method** – Contour approximation method

1. `CV_CHAIN_APPROX_NONE` - stores absolutely all the contour points. That is, any 2 subsequent points (x1,y1) and (x2,y2) of the contour will be either horizontal, vertical or diagonal neighbors, that is, max(abs(x1-x2),abs(y2-y1)) == 1.
2. `CV_CHAIN_APPROX_SIMPLE` - compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points.
3. `CV_CHAIN_APPROX_TC89_L1`,`CV_CHAIN_APPROX_TC89_KCOS` - applies one of the flavors of the Teh-Chin chain approximation algorithm.

- **Contours** - Detected contours. Each contour is stored as a vector of points.
- **Hierarchy** - Optional output vector, containing information about the image topology

How to draw the 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.

Function

```
cv2.drawContours(image, contours, contourIdx, color[, thickness[,
lineType[, hierarchy[, maxLevel[, offset]]]]])
```

Parameters

- **Image** – Destination image
- **Contours** – All the input contours. Each contour is stored as a point vector.
- **ContourIdx** - Parameter indicating a contour to draw. If it is negative, all the contours are drawn.
- **Color** – Color of the contours.
- **Thickness** – Thickness of lines the contours are drawn with.
- **lineType** – Line connectivity
- **hierarchy** – Optional information about hierarchy
- **maxLevel** – Maximum level for drawn contours. If it is 0, only the specified contour is drawn. If it is 1, the function draws the contour(s) and all the nested contours. If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on. This parameter is only taken into account when there is hierarchy available.
- **offset** – Optional contour shift parameter. Shift all the drawn contours by the specified.

Hierarchy

When we deal with different images, not all have contours only on the outside. In some cases the shapes can have shapes inside them. In other words, they can have nested contours. Outer one is called the parent and the inner one is called the child. This way, contours in an image have some relationship to each other. Representation of this relationship is called as hierarchy.

In this image, there are a few shapes which we have numbered from 0 to 5. 2 and 2a denote the external and internal contours of the outermost box.
Here, contours 0,1,2 are external or outermost. We can say, they are in hierarchy-0 or simply they are in same hierarchy level.

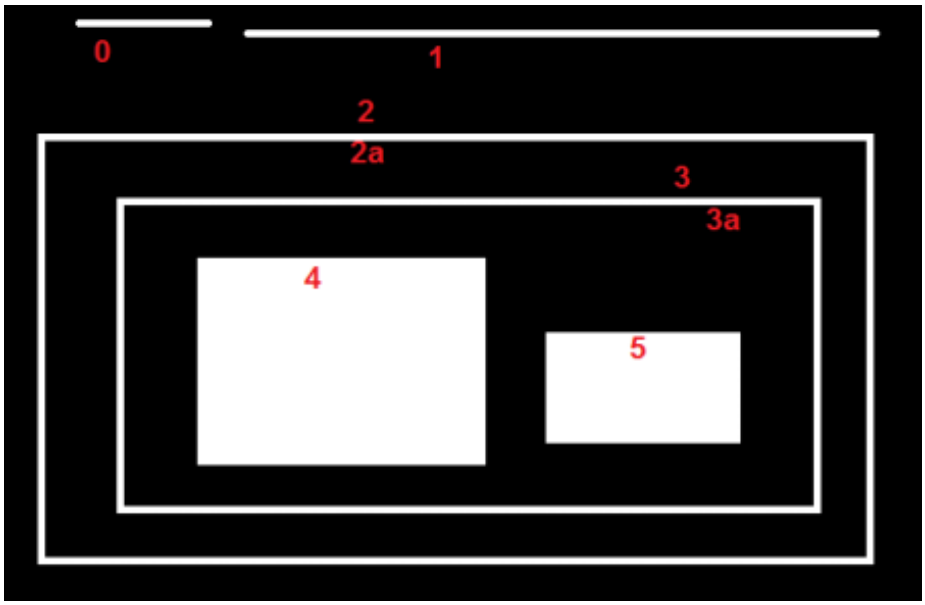

Figure 1: Hierarchy

Next comes contour-2a. It can be considered as a child of contour-2 (or in opposite way, contour-2 is parent of contour-2a). So let it be in hierarchy-1. Similarly contour-3 is child of contour-2 and it comes in next hierarchy. Finally contours 4,5 are the children of contour-3a, and they come in the last hierarchy level. From the way I numbered the boxes, I would say contour-4 is the first child of contour-3a (It can be contour-5 also).

Hierarchy representation

Each hierarchy of a contour can be represented by:
`[Next, Previous, First Child, Parent]`
Next denotes next contour at the same hierarchical level.
Previous denotes previous contour at the same hierarchical level
First_Child denotes its first child contour.
Parent denotes index of its parent contour.
For eg, take **contour-0** in the above picture. Who is next contour in its same level ? It is contour-1. So simply put Next = 1.
Similarly for **contour-1**, next is contour-2. So Next = 2.
What about **contour-2**? There is no next contour in the same level. So simply, put Next = -1.
What about **contour-4**? It is in same level with contour-5.So its next contour is contour-5, so Next = 5.

And so on for the others. . . ..

Contour retrieval mode

1. **RETR_LIST** - retrieves all of the contours without establishing any hierarchical relationships.

2. **RETR_EXTERNAL** retrieves only the extreme outer contours. It sets hierarchy[i][2] = hierarchy[i][3] = -1 for all the contours.

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.
   We can explain it with a simple image. Here we have labelled the order of contours in red color and the hierarchy they belongs to, in green color (either 1 or 2).

```
[[[ 1 -1 -1 -1]
  [ 2  0 -1 -1]
  [ 5  1  3 -1]
  [ 4 -1 -1  2]
  [-1  3 -1  2]
  [ 7  2  6 -1]
  [-1 -1 -1  5]
  [ 8  5 -1 -1]
  [-1  7 -1 -1]]]
```

Figure 2: RETR_CCOMP

4. **RETR_TREE** - It retrieves all the contours and creates a full family hierarchy list.

Code

```
#Importing modules
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

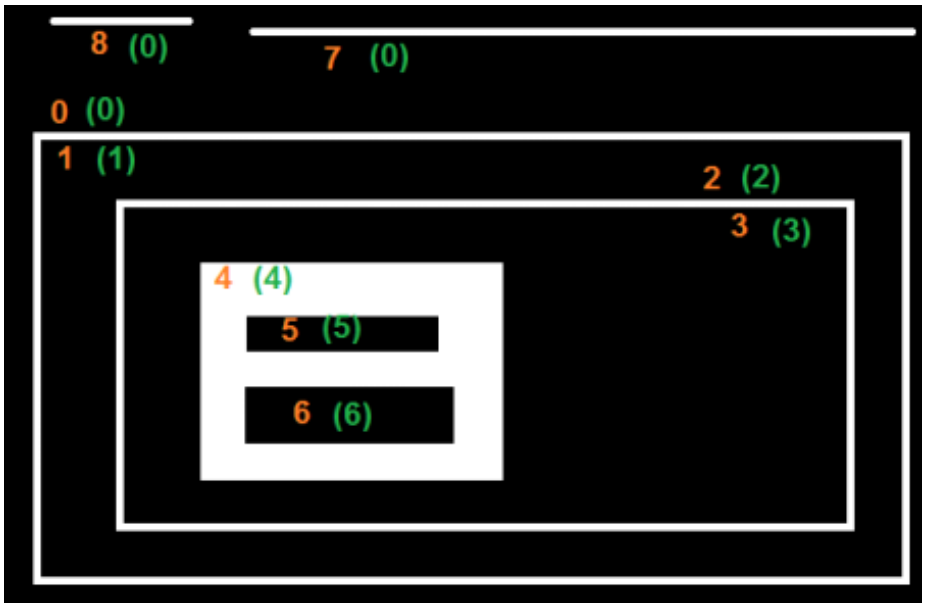

Figure 3: RETR_TREE

```
#Read the image from the camera
img1 = cv2.imread('heirarchy.png')
img2 =  img1.copy()   #Using the same image by copying
img3 =  img1.copy()
img4 =  img1.copy()

#Converting to grayscale
gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

#Thesholding
ret,thresh = cv2.threshold(gray,200,255,0)

#Finding contours for each of them
contours1,heirarchy1 = cv2.findContours(thresh,cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)
contours2, heirarchy2 = cv2.findContours(thresh, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
contours3,heirarchy3 = cv2.findContours(thresh,cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours4, heirarchy4 = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPL

#Drawing contours for all of them
cv2.drawContours(img1, contours1, -1, (0,255,0),1)
cv2.drawContours(img2, contours2, -1, (0,255,0),1)
cv2.drawContours(img3, contours3, -1, (0,255,0),1)
```

```
 cv2.drawContours(img4, contours4, -1, (0,255,0),1)

 #Printing their hierarchy/ order of relationship
 print heirarchy1
 print heirarchy2
 print heirarchy3
 print heirarchy4

#Plotting the image
plt.subplot(221),plt.imshow(img1),plt.title("RETR_CCOMP")
plt.xticks([]),plt.yticks([])
plt.subplot(222),plt.imshow(img2),plt.title("RETR_LIST")
plt.xticks([]),plt.yticks([])
plt.subplot(223),plt.imshow(img3),plt.title("RETR_TREE")
plt.xticks([]),plt.yticks([])
plt.subplot(224),plt.imshow(img4),plt.title("RETR_EXTERNAL")
plt.xticks([]),plt.yticks([])
plt.show()


#Escape sequence
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The output will be as follows:

We observe that LIST, TREE and CCOMP draw contours the same way. However the naming of contours differs as you see in the figure below:

Resources

1. Find contours - opencv docs
2. Contours and hierarchy - opencv docs
3. Contours - introduction

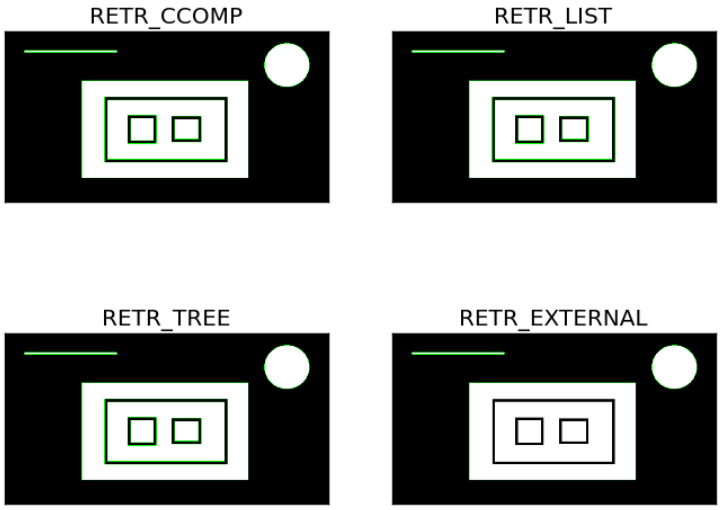

Figure 4: Output

RETR_CCOMP

```
[[[ 1 -1 -1 -1]
  [ 2  0 -1 -1]
  [ 5  1  3 -1]
  [ 4 -1 -1  2]
  [-1  3 -1  2]
  [ 7  2  6 -1]
  [-1 -1 -1  5]
  [ 8  5 -1 -1]
  [-1  7 -1 -1]]]
```

RETR_LIST

```
[[[ 1 -1 -1 -1]
  [ 2  0 -1 -1]
  [ 3  1 -1 -1]
  [ 4  2 -1 -1]
  [ 5  3 -1 -1]
  [ 6  4 -1 -1]
  [ 7  5 -1 -1]
  [ 8  6 -1 -1]
  [-1  7 -1 -1]]]
```

RETR_TREE

```
[[[ 7 -1  1 -1]
  [-1 -1  2  0]
  [-1 -1  3  1]
  [ 5 -1  4  2]
  [-1 -1 -1  3]
  [-1  3  6  2]
  [-1 -1 -1  5]
  [ 8  0 -1 -1]
  [-1  7 -1 -1]]]
```

RETR_EXTERNAL

```
[[[ 1 -1 -1 -1]
  [ 2  0 -1 -1]
  [-1  1 -1 -1]]]
```

Figure 5: Output