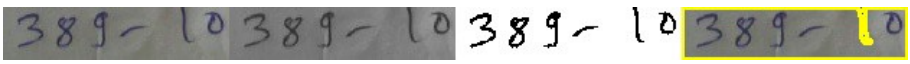
OpenCV (findContours) Detailed Guide

Raqueeb Shaikh Follow
Jun 22, 2020 · 4 min read

OpenCV has many Image Processing features that are helpful in detecting edges, removing Noise,Threshold Image etc. One such feature that often confuses a lot of Beginners is (**findContours**). In this article I will try to go to all of the topics covered in the findContours and how it can be helpful to you.

To put in simple words findContours detects change in the image color and marks it as contour. As an example, the image of number written on paper the number would be detected as contour.

1.Original Image — 2.Gray Scale — 3. Binary Threshold Inverted — 4. findContours



1.Original Image — 2.Gray Scale — 3. Binary Threshold — 4. findContours

From the above two images we can see the findContours detects differently on the same image because of how we calculated the binary image. The above one is Binary Threshold Inverted **(cv2.THRESH_BINARY_INV)** which shows the numbers highlighted in White and the Below image is just Binary Threshold **(cv2.THRESH_BINARY)**

The part that you want to detect should be white like above numbers in 1st image.

```
import cv2

def show_image(image):
    cv2.imshow('image',image)
```

```
    c = cv2.waitKey()
    if c >= 0 : return -1
    return 0

image = cv2.imread('./Photos/num.png')
img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
ret, im = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY_INV)
contours, hierarchy  = cv2.findContours(im, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
img = cv2.drawContours(res_img, contours, -1, (0,255,75), 2)
show_image(res_img)
```

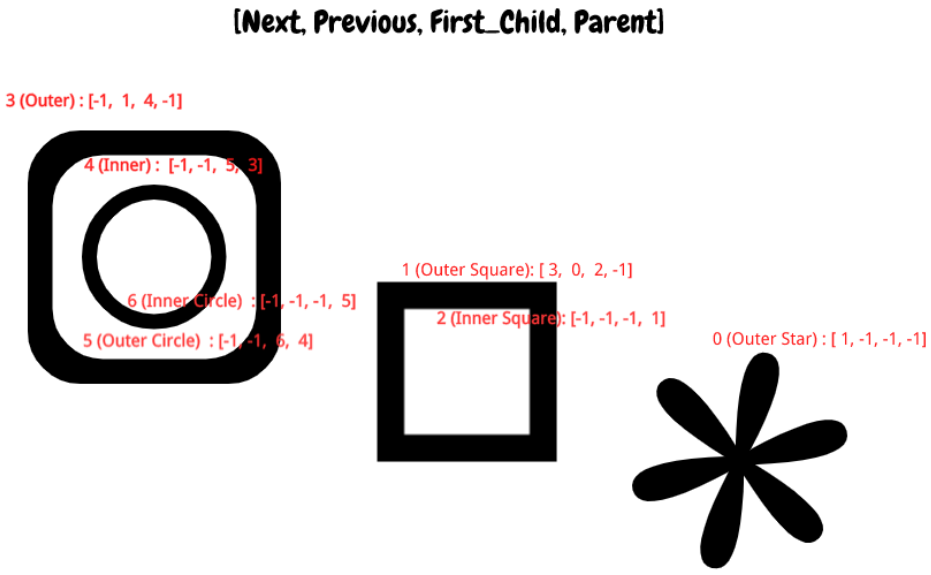
Parameters

image	source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary.
contours	detected contours.
hierarchy	containing information about the image topology. It has as many elements as the number of contours.
mode	contour retrieval mode(see cv.RetrievalModes).
method	contour approximation method(see cv.ContourApproximationModes).
offset	optional offset by which every contour point is shifted. This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context.

Parameter — Mode (RetrievalModes)

findContours function retrieve all the contours in the image that it can find . There can be various ways in which contours can be present in image. Some might me nested inother contours etc. To make it easy for to find the Contours which we are interested in and also to know the hierarchy in which the contours are nested **RetrievalModes** are very important . *source*

The output of RetrievalModes is array **Hierarchy** which shows how various contours are linked to each other, their relation with other contours, Parent Child relation



As you can see from the image OpenCV labels the hierarchy from bottom right .

> ***0 : Star [1,-1,-1,-1]***
> *… Next — Outer Square (1)*
> *… Previous — No Contour(-1)*
> *… Child — No Child (-1)*
> *… Parent — No Parent (-1)*
> *In the same manner all other contours are labeled . The above method is RETR_TREE where all contours as well as their occurrence Hierarchy is maintained . We will look about it in more details below*

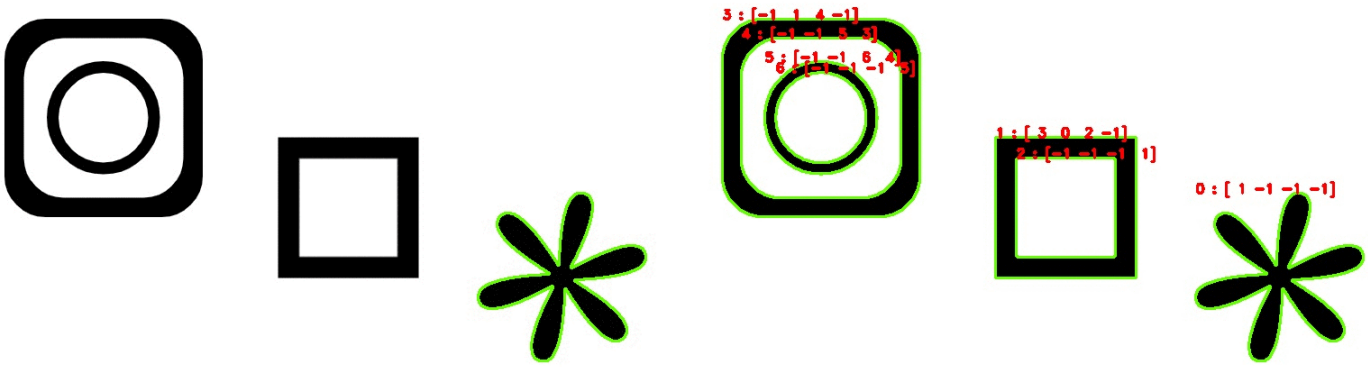
RETR_TREE — Python: cv.RETR_TREE

```
hierarchy
```

```
array([[[ 1, -1, -1, -1],
        [ 3,  0,  2, -1],
        [-1, -1, -1,  1],
        [-1,  1,  4, -1],
        [-1, -1,  5,  3],
        [-1, -1,  6,  4],
        [-1, -1, -1,  5]]], dtype=int32)
```

Retrieves all of the contours and reconstructs a full hierarchy of nested contours. *source*
Below is have shown how contours are created and its Hierarchy

Parent and Child Relationship are created as well in TREE



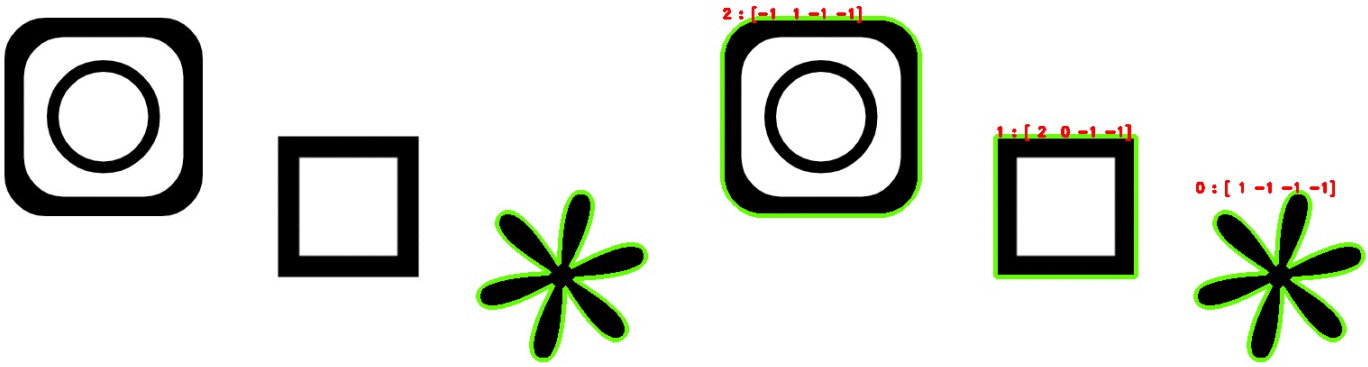
RETR_EXTERNAL — Python: cv.RETR_EXTERNAL

```
hierarchy
```

```
array([[[ 1, -1, -1, -1],
        [ 2,  0, -1, -1],
        [-1,  1, -1, -1]]], dtype=int32)
```

Retrieves only the extreme outer contours. As you can see only three contours are created i.e the external one .

No Parent ,Child relation ship is given. All External contours are at same hierarchy (Level)



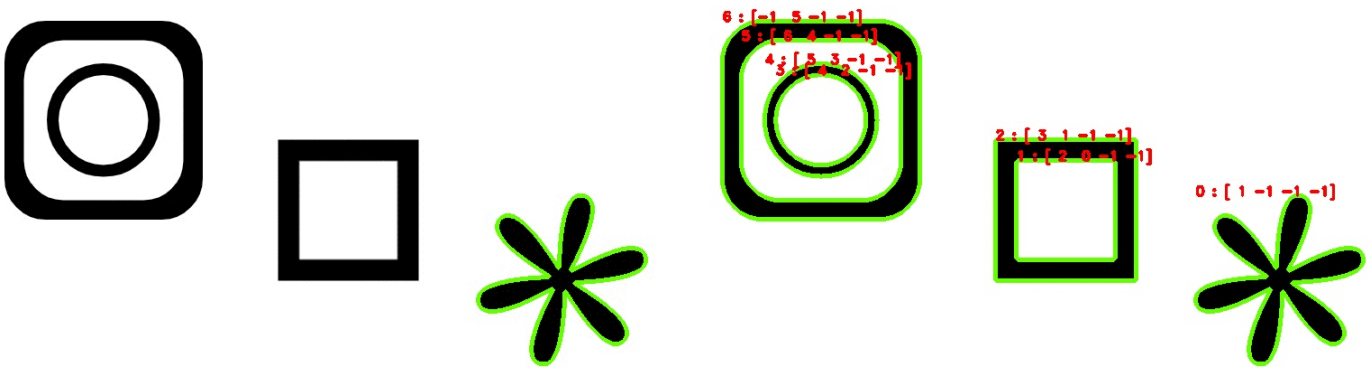
RETR_LIST — Python: cv.RETR_LIST

```
hierarchy
```

```
array([[[ 1, -1, -1, -1],
        [ 2,  0, -1, -1],
        [ 3,  1, -1, -1],
        [ 4,  2, -1, -1],
        [ 5,  3, -1, -1],
        [ 6,  4, -1, -1],
        [-1,  5, -1, -1]]], dtype=int32)
```

It is similar to Tree but it does not establish any parent child relationship . Retrieves all of the contours without establishing any hierarchical relationships.

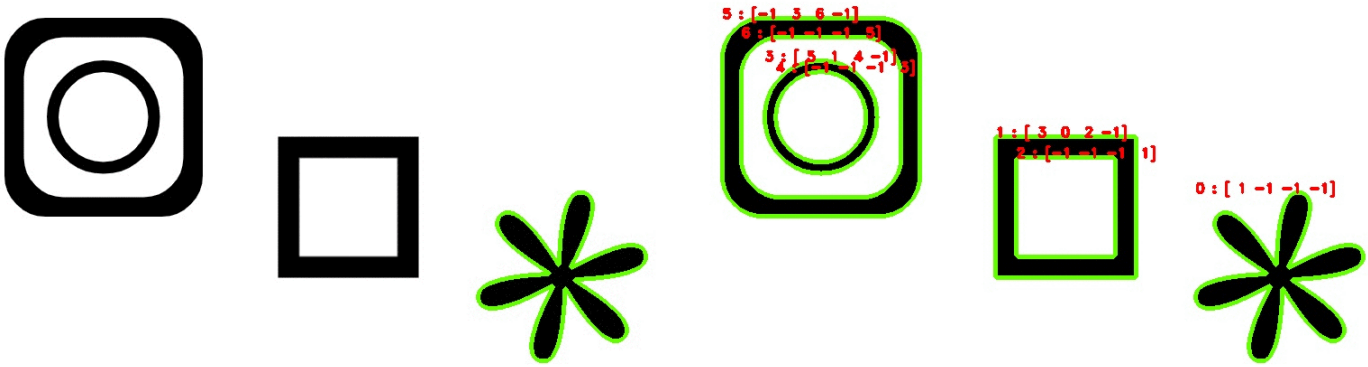
Because no relationship is established all Child,Parent are (-1)



RETR_CCOMP — Python: cv.RETR_CCOMP

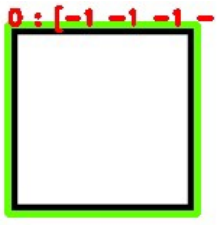
```
: hierarchy

: array([[[ 1, -1, -1, -1],
         [ 3,  0,  2, -1],
         [-1, -1, -1,  1],
         [ 5,  1,  4, -1],
         [-1, -1, -1,  3],
         [-1,  3,  6, -1],
         [-1, -1, -1,  5]]], dtype=int32)
```

Retrieves all of the contours and organizes them into a two-level hierarchy. At the top level, there are external boundaries of the components. At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.



Parameter — Method (Contour approximation method)

It is the method in which you want to store the Contours. If set to **CHAIN_APPROX_NONE** stores absolutely all the contour points. If set to **CHAIN_APPROX_SIMPLE** compresses horizontal, vertical, and diagonal segments and leaves only their end points. *source*



with CHAIN_APPROX_NONE length of contours stored is 524 i.e all points are stored in array

```
[82]: contours, hierarchy  = cv2.findContours(im, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

[83]: len(contours[0])

[83]: 524

[84]: contours
```

```
[84]: [array([[[16, 16]],

              [[16, 17]],

              [[16, 18]],

              ...,

              [[19, 16]],

              [[18, 16]],

              [[17, 16]]], dtype=int32)]
```

With CHAIN_APPROX_SIMPLE length of contours stored is 4 i.e only corner four points are stored in array

```
[85]: contours, hierarchy  = cv2.findContours(im, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

[86]: len(contours[0])

[86]: 4

[87]: contours

[87]: [array([[[ 16,  16]],

              [[ 16, 147]],

              [[147, 147]],

              [[147,  16]]], dtype=int32)]
```

Reference: https://docs.opencv.org/master/d9/d8b/tutorial_py_contours_hierarchy.html