

Contours : Getting Started

Goal

- Understand what contours are.
- Learn to find contours, draw contours etc
- You will see these functions : `cv.findContours()`, `cv.drawContours()`

What are contours?

Contours can be explained simply as a curve joining all the continuous points (along the boundary), 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.
- Since OpenCV 3.2, `findContours()` no longer modifies the source image.
- 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.

Let's see how to find contours of a binary image:

```
import numpy as np
import cv2 as cv

im = cv.imread('test.jpg')
imgray = cv.cvtColor(im, cv.COLOR_BGR2GRAY)
ret, thresh = cv.threshold(imgray, 127, 255, 0)
contours, hierarchy = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
```

See, there are three arguments in `cv.findContours()` function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy. Contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.

Note

We will discuss second and third arguments and about hierarchy in details later. Until then, the values given to them in code sample will work fine for all images.

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. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.

- 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)
```

Note

Last two methods are same, but when you go forward, you will see last one is more useful.

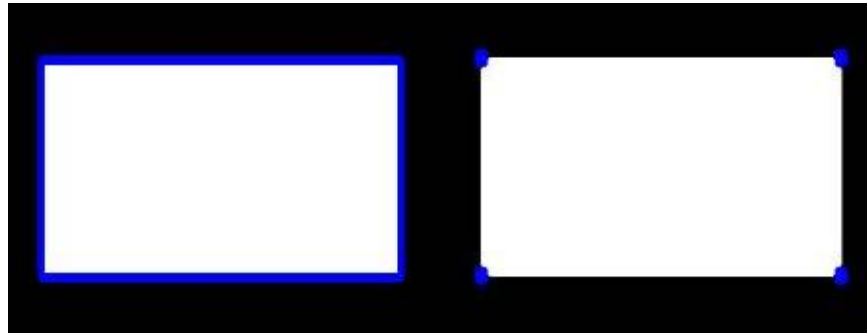
Contour Approximation Method

This is the third argument in `cv.findContours` function. What does it denote actually?

Above, we told that 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.

If you pass `cv.CHAIN_APPROX_NONE`, all the boundary points are stored. But actually do we need all the points? For eg, you found the contour of a straight line. Do you need all the points on the line to represent that line? No, we need just two end points of that line. This is what `cv.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 `cv.CHAIN_APPROX_NONE` (734 points) and second image shows the one with `cv.CHAIN_APPROX_SIMPLE` (only 4 points). See, how much memory it saves!!!



image

Additional Resources

Exercises