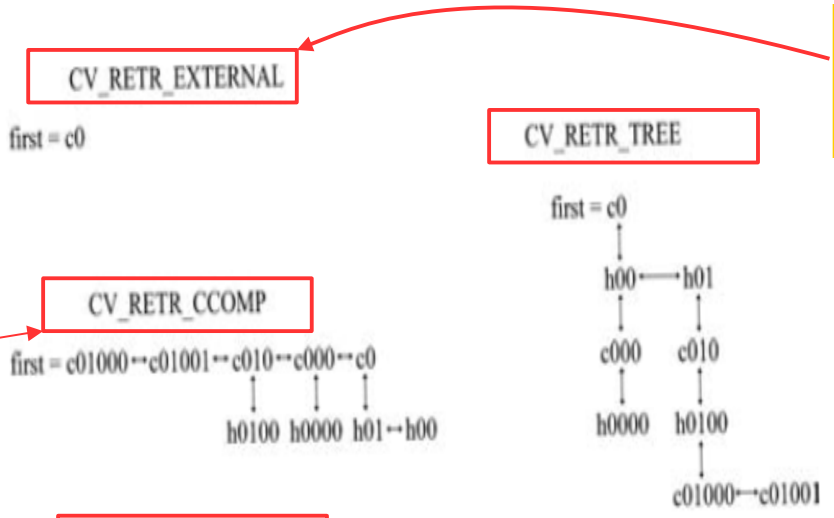
Contours

actly what a contour is. A contour is a list of points that represent, in one way or another, a curve in an image. This representation can be different depending on the circumstance at hand. There are many ways to represent a curve. Contours are represented in OpenCV by sequences in which every entry in the sequence encodes information about the location of the next point on the curve. We will dig into the details of such

Reference: Learning OpenCV, pp. 250

The function cvFindContours() computes contours from binary images. It can take images created by cvCanny(), which have edge pixels in them, or images created by functions like cvThreshold() or cvAdaptiveThreshold(), in which the edges are implicit as boundaries between positive and negative regions.*

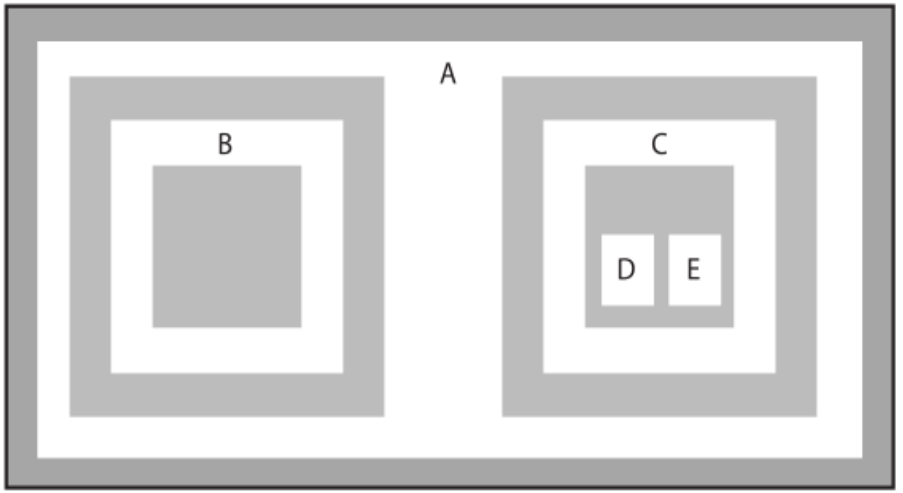
4 Contours Models

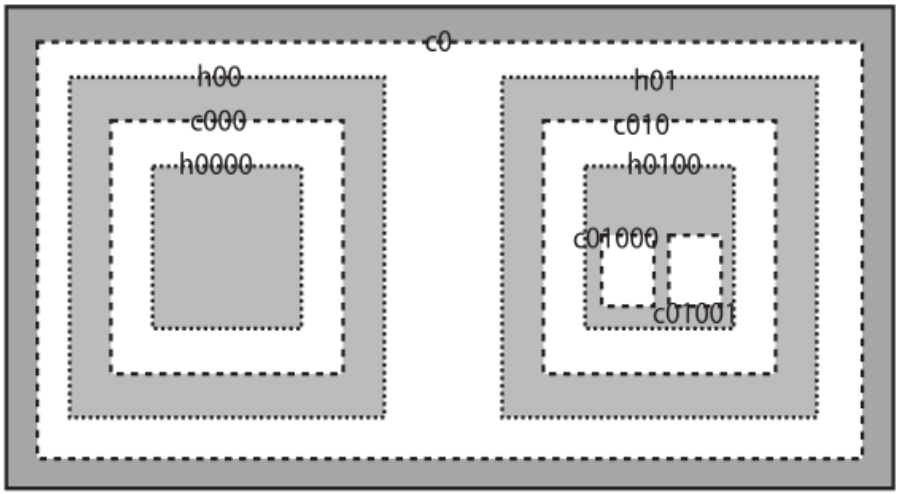


Retrieves only the external outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.

CV_RETR_CCOMP Retrieves all the contours into two-level hierarchy, top-level for external boundaries and the 2nd level for the holes.

retrieves all contours without any hierarchical relationships.





http://opencvexamples.blogspot.com/2013/09/find-contour.html

Contour Mask And Pixel Points

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html

```
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(imgray,mask = mask)
```

16. Mask and Pixel Points

All the points comprises that object (contour)

```
mask = np.zeros(imgray.shape,np.uint8)
cv2.drawContours(mask,[cnt],0,255,-1)
pixelpoints = np.transpose(np.nonzero(mask))
#pixelpoints = cv2.findNonZero(mask)
```

Above, “two methods, one using Numpy functions, next one using OpenCV function (last commented line) are given to do the same. Results are also same, but with a slight difference. Numpy gives coordinates in (row, column) format, while OpenCV gives coordinates in (x,y) format. So basically the answers will be interchanged. Note that, row = x and column = y.”

17. Maximum Value, Minimum Value and their locations

18. Mean Color or Mean Intensity

```
mean_val = cv2.mean(im,mask = mask)
```

19. Extreme Points

```
leftmost = tuple(cnt[cnt[:,:,0].argmin()][0])
rightmost = tuple(cnt[cnt[:,:,0].argmax()][0])
topmost = tuple(cnt[cnt[:,:,1].argmin()][0])
bottommost = tuple(cnt[cnt[:,:,1].argmax()][0])
```

Example Hierarchy Results on the Terminal

```
hierarchy_TREE:
countour 0 [ne, pr, ch, pa]: [-1, -1, 1, -1]
countour 1 [ne, pr, ch, pa]: [4, -1, 2, 0]
countour 2 [ne, pr, ch, pa]: [3, -1, -1, 1]
countour 3 [ne, pr, ch, pa]: [-1, 2, -1, 1]
countour 4 [ne, pr, ch, pa]: [-1, 1, 5, 0]
countour 5 [ne, pr, ch, pa]: [-1, -1, -1, 4]

hierarchy_CCOMP:
countour 0 [ne, pr, ch, pa]: [1, -1, -1, -1]
countour 1 [ne, pr, ch, pa]: [2, 0, -1, -1]
countour 2 [ne, pr, ch, pa]: [3, 1, -1, -1]
countour 3 [ne, pr, ch, pa]: [-1, 2, 4, -1]
countour 4 [ne, pr, ch, pa]: [5, -1, -1, 3]
countour 5 [ne, pr, ch, pa]: [-1, 4, -1, 3]
```

The contour hierarchies follow this format:

contour [index]: [next, previous, 1st child, parent]

'-1' means N/A.

For example:

```
countour 1 [ne, pr, ch, pa]: [4, -1, 2, 0]
```

The contour 1 has:

- Contour 4 is the next contour in the same level.
- No previous contour in the same level.
- First child is contour 2.
- Parent is contour 0

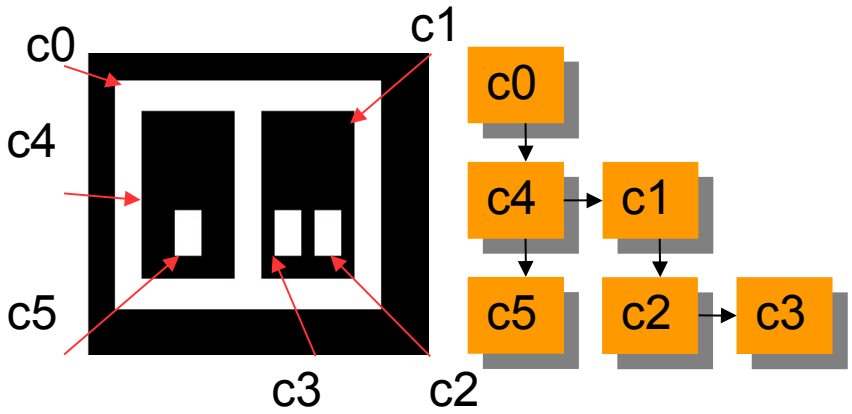


Figure 1. hierarchy_TREE

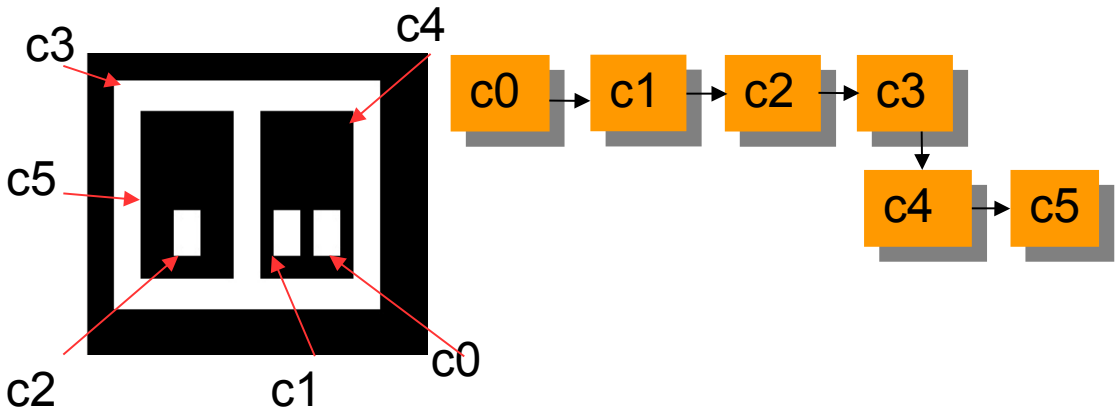
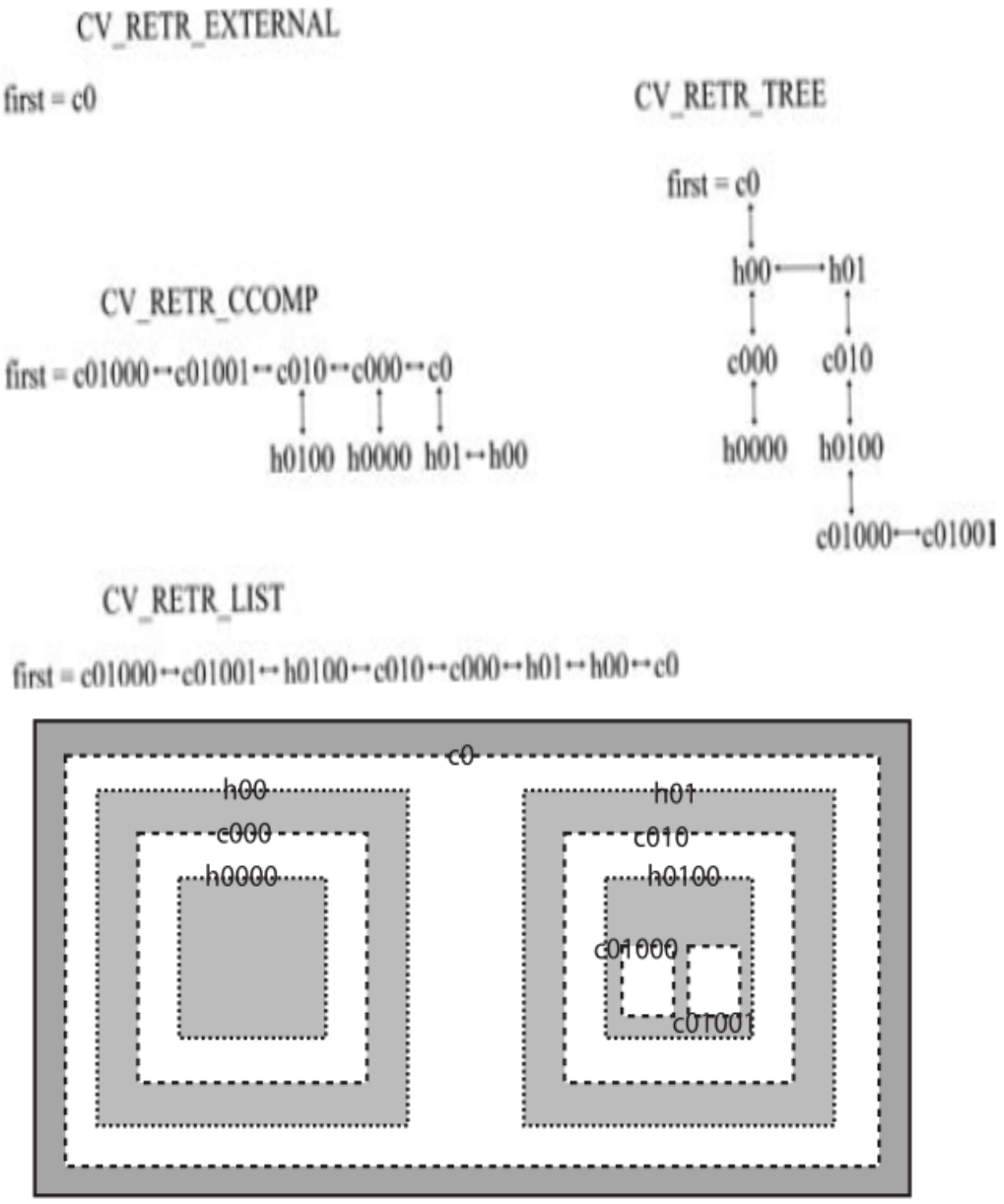


Figure 2. hierarchy_CCOMP

9-12-2018 Contour Hierarchy

The hierarchy form:
hierarchy[idx][{0,1,2,3}]={next contour (same level), previous contour (same level), child contour, parent contour}

CV_RETR_CCOMP, returns a hierarchy of outer contours and holes. This means elements 2 and 3 of hierarchy[idx] have at most one of these not equal to -1: that is, each element has either no parent or child, or a parent but no child, or a child but no parent.
An element with a parent but no child would be a boundary of a hole.
That means you basically go through hierarchy[idx] and draw anything with hierarchy[idx][3]>-1.
Something like (works in Python, but haven't tested the C++. Idea is fine though.):

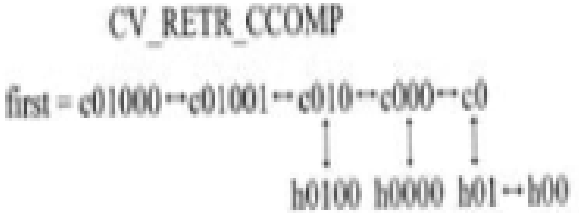


http://opencvexamples.blogspot.com/2013/09/find-contour.html

Contours Hierarchy

```
findContours( temp, contours, hierarchy,
              RETR_CCOMP,
              CHAIN_APPROX_SIMPLE );

for( ; idx >= 0; idx = hierarchy[idx][0] )
{
    const vector<Point>& c = contours[idx];
    double area = fabs(contourArea(Mat(c)));
    if( area > maxArea )
    {
        maxArea = area;
        largestComp = idx;
    }
}
Scalar color( 0, 0, 255 );
drawContours( dst, contours, largestComp, color,
FILLED, LINE_8, hierarchy );
```

Note: 1 hierarchy[idx][0]



From example figure 1, first element, e.g., “0” in hierarchy[idx][0] defines the level for c-type contours (no holes) when use RETR_CCOMP

Note 2: assign all points of a contour to a vector

```
const vector<Point>& c = contours[idx];
```

Note 3: find an area of a contour

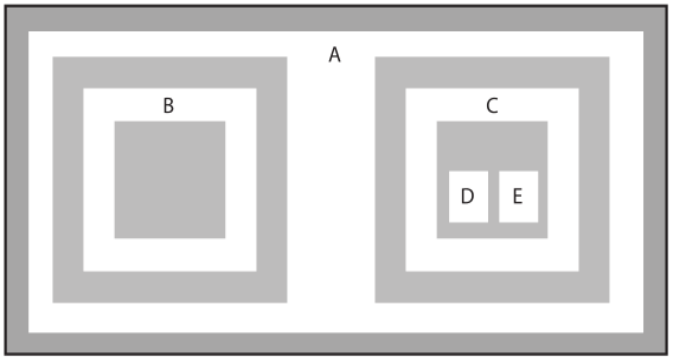
```
const vector<Point>& c = contours[idx];

Mat A = Mat(c)
double area = fabs(contourArea(Mat(c)));
```


http://opencvexamples.blogspot.com/2013/09/find-contour.html

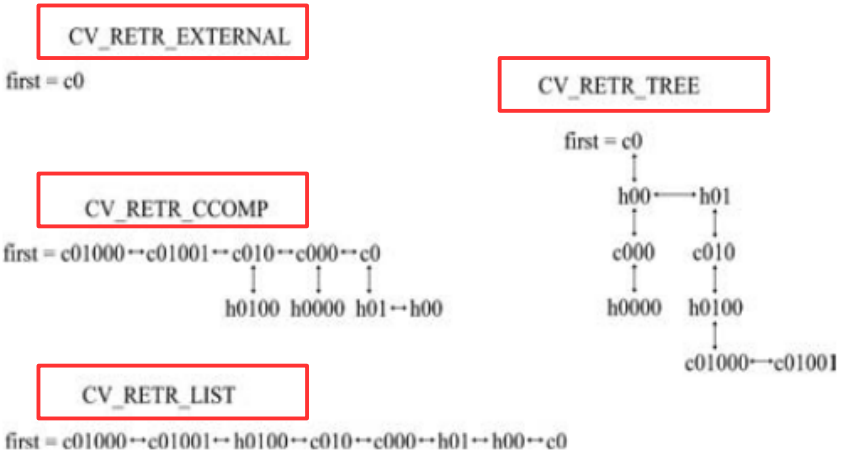
Contours Mode Variable

findContours(image_canny, contours, hierarchy,
RETR_CCOMP,
CHAIN_APPROX_SIMPLE)

The mode variable can be set to any of four options: CV_RETR_EXTERNAL, CV_RETR_LIST, CV_RETR_CCOMP, or CV_RETR_TREE. The value of mode indicates to cvFindContours() exactly what contours we would like found and how we would like the result presented to us. In particular, the manner in which the tree node variables (h_prev, h_next, v_prev, and v_next) are used to "hook up" the found contours is determined by the value of mode. In Figure 8-3, the resulting topologies are shown for all four possible values of mode. In every case, the structures can be thought of as "levels" which are related by the "horizontal" links (h_next and h_prev), and those levels are separated from one another by the "vertical" links (v_next and v_prev).

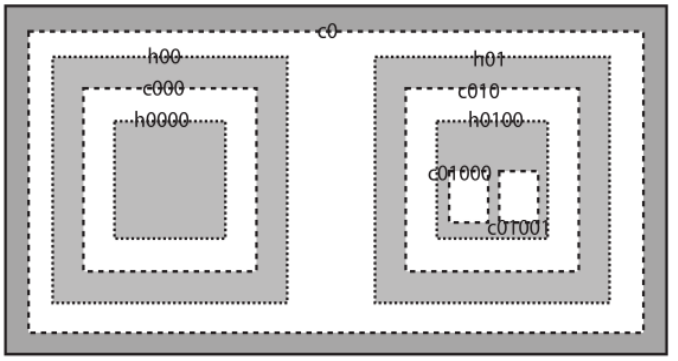


Retrieves only the extreme outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.



CV_RETR_CCOMP
Retrieves all the contours into two-level hierarchy, top-level for external boundaries and the 2nd level for the holes.

retrieves all contours without any hierarchical relationships.



http://opencvexamples.blogspot.com/2013/09/find-contour.html

Compute Contours Features

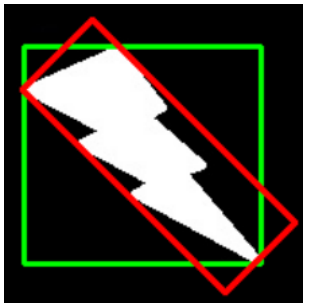
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

1. Moments

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread('star.jpg',0)
5 ret,thresh = cv2.threshold(img,127,255,0)
6 contours,hierarchy = cv2.findContours(thresh, 1, 2)
7
8 cnt = contours[0]
9 M = cv2.moments(cnt)
10 print M
```

2. Contour Area

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



3. Contour Perimeter

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

4. Contour Approximation

```
1 epsilon = 0.1*cv2.arcLength(cnt,True)
2 approx = cv2.approxPolyDP(cnt,epsilon,True)
```

5. Convex Hull Convexity defects

checks a curve for convexity defects and corrects it

```
hull = cv2.convexHull(cnt)
```

6. Checking Convexity

```
k = cv2.isContourConvex(cnt)
```

7.a. Straight Bounding Rectangle

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

7.b. Rotated Rectangle

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

Compute Contours Features

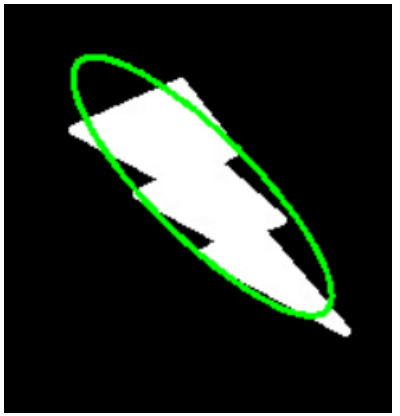
https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

8. Minimum Enclosing Circle

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

9. Fitting an Ellipse

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



http://nicky.vanforeest.com/misc/fitEllipse/fitEllipse.html

10. Fitting a Line

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

From Contour Find Shapes

https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

4. Contour Approximation

```
1 epsilon = 0.1*cv2.arcLength(cnt,True)
2 approx = cv2.approxPolyDP(cnt,epsilon,True)
```

5. Convex Hull Convexity defects

checks a curve for convexity defects and corrects it

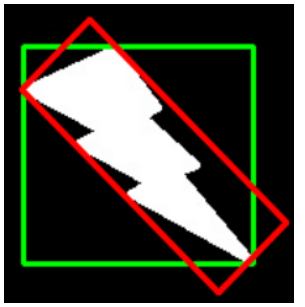
```
hull = cv2.convexHull(cnt)
```

7.a. Straight Bounding Rectangle

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

7.b. Rotated Rectangle

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

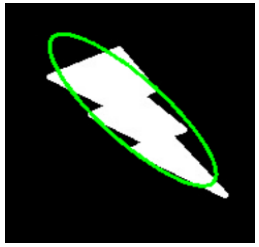


8. Minimum Enclosing Circle

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

9. Fitting an Ellipse

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



Contour-Shapes Properties

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_properties/py_contour_properties.html

11. Aspect Ratio

$$Aspect\ Ratio = \frac{Width}{Height}$$

```
x,y,w,h = cv2.boundingRect(cnt)
aspect_ratio = float(w)/h
```

12. Extent

$$Extent = \frac{Object\ Area}{Bounding\ Rectangle\ Area}$$

```
area = cv2.contourArea(cnt)
x,y,w,h = cv2.boundingRect(cnt)
rect_area = w*h
extent = float(area)/rect_area
```

13. Solidity

$$Solidity = \frac{Contour\ Area}{Convex\ Hull\ Area}$$

```
area = cv2.contourArea(cnt)
hull = cv2.convexHull(cnt)
hull_area = cv2.contourArea(hull)
solidity = float(area)/hull_area
```

14. Equivalent Diameter

$$Equivalent\ Diameter = \sqrt{\frac{4 \times Contour\ Area}{\pi}}$$

```
area = cv2.contourArea(cnt)
equi_diameter = np.sqrt(4*area/np.pi)
```

15. Orientation

Following method also gives the Major Axis and Minor Axis lengths.

```
(x,y),(MA,ma),angle = cv2.fitEllipse(cnt)
```