

Lesson 13 Image Processing---Contour

Introduction and Feature

1.Contour Introduction

Contour is defined as the line joining all the points along the boundary of an image that are having the same color or intensity. Contour is useful tool for shape analyzing as well as object detection and recognition.

For higher accuracy, binaryzation will be performed first. After the binary image is obtained, search the contour that is find the white object under the black background. Therefore, our target is the white object and the background is black.

2.Search and Draw Contour

After the object is found, search for the contour points and draw the contour.

2.1Operation Steps



Note:

1) **Before operation, please copy the routine “contours_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 13 Image Processing --- Contour Introduction and Feature->Routine Code” to the shared folder.**

2) **For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.**

3) **The input command should be case sensitive and the keywords**

can be complemented by “Tab” key.

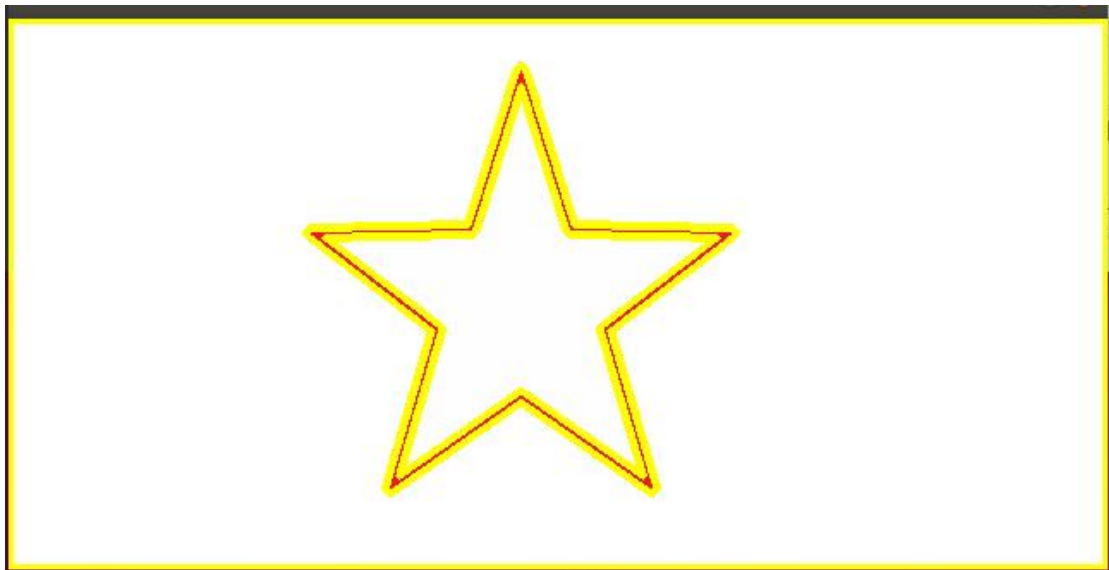
- 1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.
- 2) Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

- 3) Input command “**python3 contours_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 contours_demo.py
```

2.2 Program Outcome



The final output picture is as above.

2.3 Code Analysis

The routine “**contours_demo.py**” can be found in “**4.OpenCV->Lesson 13 Image Processing --- Contour Introduction and Feature->Routine Code**”.

```
import cv2
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
binary, contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
img3 = cv2.drawContours(img, contours, -1, (0,255,255), 3)
cv2.imshow("BINARY", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Import Module: Import cv2 module

Read Picture: Call imread function to read picture and the parameter stands for the name of the picture.

Color Space Conversion: call cvtColor function to convert the image into GRAY color space, and the parameter indicates the picture and conversion mode.

```
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
```

Thresholding: use threshold function to execute. The specific format and parameters are as follow.

threshold(src, thresh, maxval, type)

- 1) The first parameter “**src**” is the pending image.
- 2) The second parameter “**thresh**” is the set threshold.
- 3) The third parameter “**maxval**” will be set only when the type is “**THRESH_BINARY**” or “**THRESH_BINARY_INV**”. It refers to the new value assigned when the gray values of the picture pixels are greater (smaller) than the threshold
- 4) The fourth parameter “**type**” represents the type of thresholding. cv2.THRESH_BINARY indicates the part greater than threshold is set as maxval, otherwise 0

```
binary, contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Search for contour: findContours function is employed to search the

contour, and the specific format and parameters are as follow.

findContours(img,mode,method)

- 1) The first parameter “**img**” represents the pending picture.
- 2) The second parameter “**mode**” represents the detection mode of the contour.
 - ◆ RETR_EXTERNAL = 0: only the outermost contour will be detected
 - ◆ RETR_LIST = 1: detect all the contours without building hierarchy and all the contours will be put into one list
 - ◆ RETR_CCOMP = 2, detect all the contour and divide them into two layers
 - ◆ RETR_TREE = 3, Detect layer by layer from right to left according to the tree-shaped storage contour
- 3) The third parameter “**method**” stands for the method used to search the contour. The specific method is as follow.
 - ◆ CHAIN_APPROX_NONE, Preserve all the dots on the contour
 - ◆ CHAIN_APPROX_SIMPLE: Compress the horizontal, vertical and oblique part that is only keep their corner coordinate. For example, 4 dots are enough to save the information about the rectangle contour.

Note: In OpenCV4.2 or above, this function will only return two values, including “contours” and “hierarchy”. And “binary” will not be returned.

```
img3 = cv2.drawContours(img, contours, -1, (0,255,255), 3)
```

Contour drawing: adopt drawContours function to draw the contour, and

its specific format and parameters are as follow.

drawContours(image, contours, contourIdx, color, thickness)
--

- 1) The first parameter “**image**” represents the image whose contour will be drawn.
- 2) The second parameter “**contours**” represents the coordinate of all contours are found.
- 3) The third parameter “**contourIdx**” indicates the serial number of the contour drawing. -1 means that all the contours will be drawn.
- 4) The fourth parameter “**color**” refers to the color of the contour.
- 5) The fifth parameter “**thickness**” represents the width of the contour. -1 means that the contour will be padded.

3. Contour Feature Moment

Feature moment is global feature of a contour and a picture. The moment contains the geometric features in different types of the corresponding objects. There are three types of moments, including spatial moment, central moment and normalized central moment

1) Spatial moment: it is also called geometric moment about the area and perimeter of the image, including zero-order moment: m_{00} , first-order moment: m_{10} , m_{01} , second-order moment: m_{20} , m_{11} , m_{02} , third-order moment: m_{30} , m_{21} , m_{12} , m_{03}

2) Central moment: For the higher--order image, the moment will vary with the position, which can be fixed by central moment. The invariance of translation can be obtained by subtracting the mean value, so we can compare whether two objects at different positions are consistent, that is, the central

moment features translation invariance.

It includes two-order central moments: μ_{20} , μ_{11} and μ_{02} , as well as three-order central moment: μ_{30} , μ_{21} , μ_{12} and μ_{03} .

3) Normalized central moment: Apart from translation, some pictures will be scaled. Even though the image is scaled, its features can be detected. Normalized central moment obtain the scale invariance by dividing by the dimension of the object.

It includes two-order Hu moment: ν_{20} , ν_{11} and ν_{02} , and three-order Hu moment: ν_{30} , ν_{21} , ν_{12} and ν_{03} . Following, based on the obtained contour, calculate the feature moment, area and perimeter of the contour.



3.1 Operation Steps

Note:

1) Before operation, please copy the routine “moments_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 13 Image Processing --- Contour Introduction and Feature->Routine Code” to the shared folder.

2) For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3) The input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click , and then  or press “Ctrl+Alt+T” to open command line terminal.

2) Input command “cd /mnt/hgfs/share/” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) Input command “**python3 moments_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 moments_demo.py
```

3.2 Program Outcome

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 moments_demo.py
特征矩: {'m00': 203841.0, 'm10': 65127199.5, 'm01': 32512639.5, 'm20': 27744186
987.0, 'm11': 10387788320.25, 'm02': 6914354667.0, 'm30': 13296401613519.75, 'm2
1': 4425197824426.5, 'm12': 2209136316106.5, 'm03': 1654259354079.75, 'mu20': 69
36046746.75, 'mu11': 0.0, 'mu02': 1728588666.75, 'mu30': 0.0, 'mu21': 0.0, 'mu12
': 0.0, 'mu03': 0.0, 'nu20': 0.1669278996865204, 'nu11': 0.0, 'nu02': 0.04160146
061554513, 'nu30': 0.0, 'nu21': 0.0, 'nu12': 0.0, 'nu03': 0.0}
面积: 203841.0
周长: 1916.0
```

The unit of the area and perimeter is pixel. The outermost contour of the image will be calculated

3.3 Code Analysis

The routine “**moments_demo.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 13 Image Processing --- Contour Introduction and Feature->Routine Code**”.

```
import cv2
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
binary, contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnt=contours[0]
m=cv2.moments(cnt)
area=cv2.contourArea(cnt)
perimeter=cv2.arcLength(cnt,True)
print("特征矩: ",m)
print("面积: ",area)
print("周长: ",perimeter)
```

Take out the outermost contour: take the first contour with index 0 in the contour list

```
cnt=contours[0]
```

Feature moment calculation: use **moments** function to calculate the

feature moment, and the function format and parameters are as follow.

moments(array,binaryImage)

- 1) The first parameter “**array**” is the contour point.
- 2) The second parameter “**binaryImage**” is set as False by default. If it is True, all non-zero pixels will be treated as 1, which is equivalent to image binaryzation.

```
m=cv2.moments(cnt)
```

Area calculation: adopt **countourArea** function to calculate the area, and its format and parameters are as follow.

countourArea(contour): “**contour**” is a contour in the contour list.

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

Perimeter calculation: employ **arcLength** function to calculate the perimeter, and its specific format and parameter is as follow.

arcLength(curve,closed)

- 1) The first parameter “**curve**” represents the contour.
- 2) The second parameter “**closed**” decide whether the contour is closed or not. If it is closed, set is as **True**, otherwise False.

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

4.Polygon Approximation

The searched “contours” maybe too complex and not smooth, approxPolyDP function can be adopted to appropriately approximate the polygon curve, which is polygon approximation.

This function uses polygons to approximate the contour, utilizing the

Douglas-Peucker algorithm (DP). The principle of the DP algorithm is simple. Its core is to continuously find the farthest point of the polygon to form a new polygon until the shortest distance is less than the specified accuracy.

Next, analyze the object contour with polygon approximation.


4.1 Operation Steps

Note:

1) Before operation, please copy the routine “approx_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 13 Image Processing --- Contour Introduction and Feature->Routine Code” to the shared folder.

2) For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3) The input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “

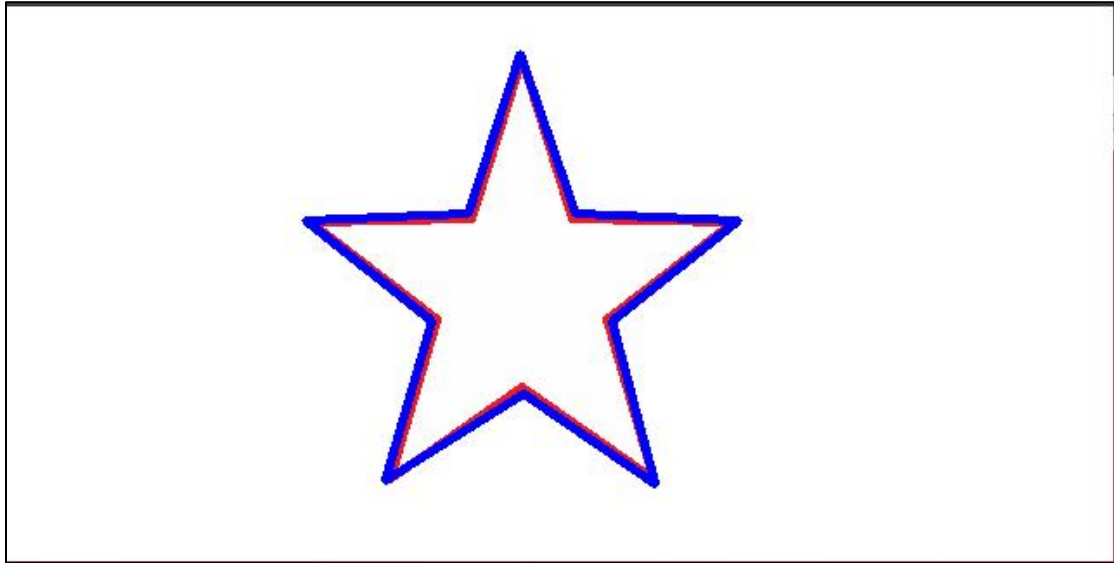
2) Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) Input command “**python3 approx_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 approx_demo.py
```

4.2 Program Outcome



The contour after polygon approximation will try to fit the figure as much as possible.

4.3 Code Analysis

The routine “`approx_demo.py`” can be found in “4. OpenCV Computer Vision Lesson->Lesson 13 Image Processing---Contour Introduction and Feature->Routine Code”.

```
import cv2
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
binary, contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnt=contours[1]
approx1=cv2.approxPolyDP(cnt,20,True)
img3 = cv2.drawContours(img, [approx1], -1, (255,0,0), 3)
cv2.imshow("BINARY", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Take the contour: take the second contour with index 1 in the contour list

```
cnt=contours[1]
```

Polygon approximation: `approxPolyDP` function is adopted. The specific format is as follow.

```
approxPolyDP(curve, epsilon, closed)
```

- 1) The first parameter “**curve**” is the contour to be searched.
- 2) The second parameter “**epsilon**” indicates accuracy. The smaller the number, the lower the accuracy and the more consistent with the pattern contour.
- 3) The third parameter “**closed**” decide whether the contour is closed or not. If it is closed, set is as **True**, otherwise False.

```
approx1=cv2.approxPolyDP(cnt,20,True)
```

5. Contour Convex Hull

Convex Hull will look similar to contour approximation, but it is the convex polygon in the outermost of the object. Convex hull refers to a polygon that completely contains the original contour and consists only of points on the contour. Every part of the convex hull is convex, that is, the line connecting any two points in the convex hull is inside the convex hull. In the convex hull, the interior angle of any three consecutive points is less than 180° .

Next, analyze the object contour through contour convex hull.



5.1 Operation Steps

Note:

1) Before operation, please copy the routine “hull_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 13 Image Processing --- Contour Introduction and Feature->Routine Code” to the shared folder.

2) For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3) The input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

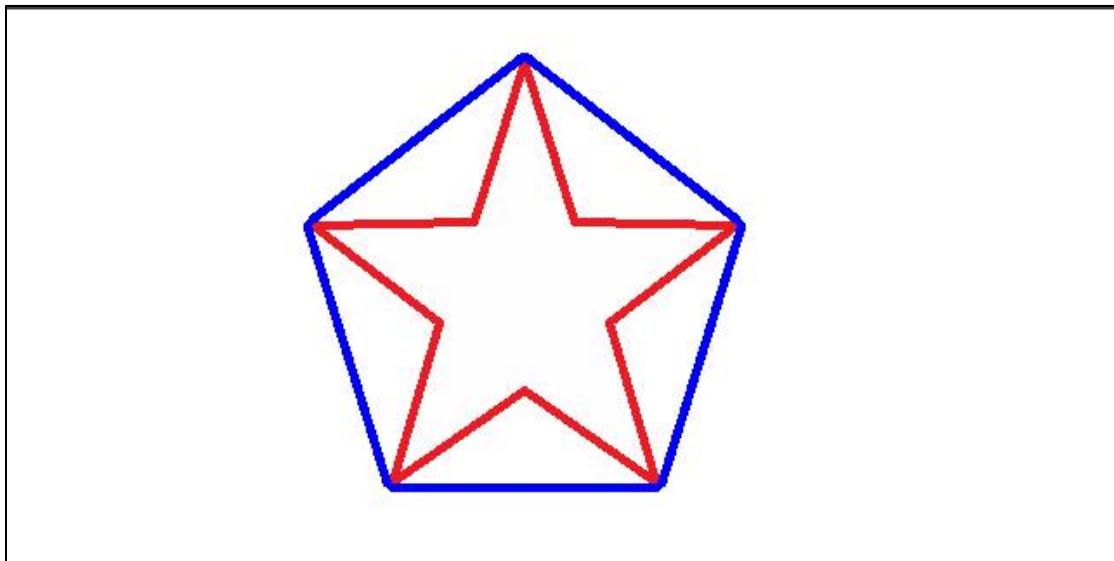
2) Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) Input command “**python3 hull_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 hull_demo.py
```

5.2 Program Outcome



The convex hull will connect the vertices of the contour.

5.3 Code Analysis

The routine “**hull_demo.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 13 Image Processing---Contour Introduction and**

Feature->Routine Code”

```
import cv2
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
binary, contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnt=contours[1]
hull=cv2.convexHull(cnt,True)
img3 = cv2.drawContours(img, [hull], -1, (255,0,0), 3)
cv2.imshow("BINARY", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Convex hull: use **convexHull** function to draw the convex hull. The function format and parameters are as follow.

convexHull(points, clockwise,)

- 1) The first parameter “**points**” is the contour to be searched.
- 2) The second parameter “**clockwise**” refers to the drawing direction.

When it is True, the convex hull will be draw clockwise. When it is False, the convex hull will be drawn counterclockwise.

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

6. Circumscribed Rectangle

The bounding rectangle is divided into the minimum bounding rectangle with rotation angle and the regular circumscribed rectangle.

Minimum bounding rectangle: it is drawn with minimum area, so it considers the rotation.

Regular bounding rectangle: frame the object with the rectangle.

Next, draw the regular bounding rectangle and minimum bounding rectangle.


6.1 Operation Steps

Note:

1) Before operation, please copy the routine “rect_demo.py” and sample picture “test.jpg” in “4.OpenCV->Lesson 13 Image Processing --- Contour Introduction and Feature->Routine Code” to the shared folder.

2) For how to configure the shared folder, please refer to the file in “2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement”.

3) The input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “

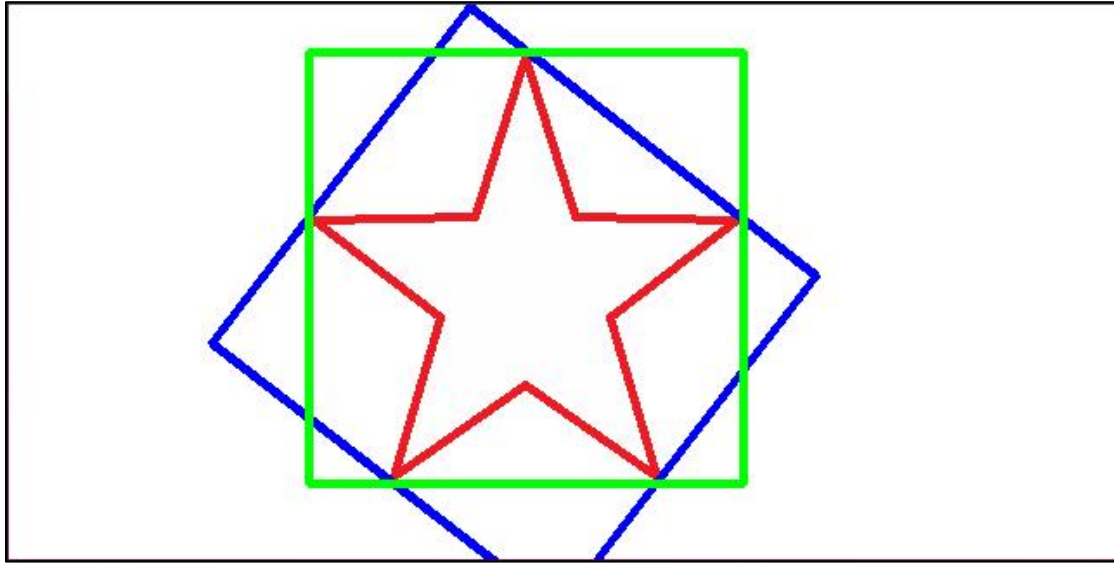
2) Input command “**cd /mnt/hgfs/share/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/share/
```

3) Input command “**python3 rect_demo.py**” and press Enter to run the routine.

```
ubuntu@ubuntu-virtual-machine:/mnt/hgfs/share$ python3 rect_demo.py
```

6.2 Program Outcome



The green one is the regular bounding rectangle, and the blue one is the minimum bounding rectangle.

6.3 Code Analysis

The routine “rect_demo.py” can be found in “4. OpenCV Computer Vision Lesson->Lesson 13 Image Processing---Contour Introduction and Feature->Routine Code”

```
import cv2
import numpy as np
img=cv2.imread('test.jpg')
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, img2 = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
binary, contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnt=contours[1]
RotatedRect=cv2.minAreaRect(cnt)
x,y,w,h=cv2.boundingRect(cnt)
box=cv2.boxPoints(RotatedRect)
box=np.int0(box)
img3 = cv2.drawContours(img, [box], -1, (255,0,0), 3)
img4=cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 3)
cv2.imshow("BINARY", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Acquire minimum bounding rectangle: adopt **minAreaRect** function to get the minimum bounding rectangle. And the function format and parameters are as follow.

minAreaRect(points): “points” refers to contour, and the returned value

contains the starting coordinate, width, height and angle.

```
RotatedRect=cv2.minAreaRect (cnt)
```

Obtain the regular bounding rectangle: use “**boundingRect**” function to realize. The function format and parameters are as follow.

boundingRect (array): “**array**” refers to contour and the returned value “**Rect**” contains the starting coordinate, width and height.

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

Get the vertex coordinate of the minimum bounding rectangle:

boxPoints function will be employed. The function format and parameters are as follow.

boxPoints(rect): refers to the rectangle whose vertex coordinate needs to be obtained. Its data type belongs to floating point number.

```
box=cv2.boxPoints (RotatedRect)
```

Number rounding: use **int0** function to execute. The function format and parameters are as follow.

int0(date): “date” is the data to be rounded.

```
box=np.int0 (box)
```

Draw rectangle: rectangle function will be adopted to draw the rectangle. The function format and parameters are as follow.

```
rectangle(src,pt1,pt2,color,thickness)
```

- 1) The first parameter “**src**” refers to the image to draw the contour.
- 2) The second parameter “**pt1**” represents one of the vertices of the rectangle.

- 3) The third parameter “**pt2**” refers to the diagonal vertices of pt1
- 4) The fourth parameter “**color**” represents the color of the rectangle.
- 5) The fifth parameter “**thickness**” represents the width of the drawn rectangle. “-1” indicates padding rectangle.

```
img4=cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
```