

第 13 课 图像处理-轮廓介绍及特征

1.轮廓介绍

轮廓是连接具有相同颜色或强度的所有连续点（沿边界）的曲线，轮廓是用于形状分析以及对象检测 and 识别的有用工具。

为了获取更高的准确性，会先进行二值化处理，在得到二进制图像后，寻找轮廓就是从黑色背景中找到白色物体，因此我们要找的对象应是白色，背景应该是黑色。

2.寻找并绘制轮廓

通过分析出物体对象后，找出物体的轮廓点位并绘制出来。



2.1 实验步骤

注意：

1) 需要先将目录“第 4 章 OpenCV 计算机视觉学习->图像处理进阶篇->第 13 课 图像处理——轮廓介绍及特征->例程源码”下的例程“contours_demo.py”和示例图片“test.jpg”复制到共享文件夹。

2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

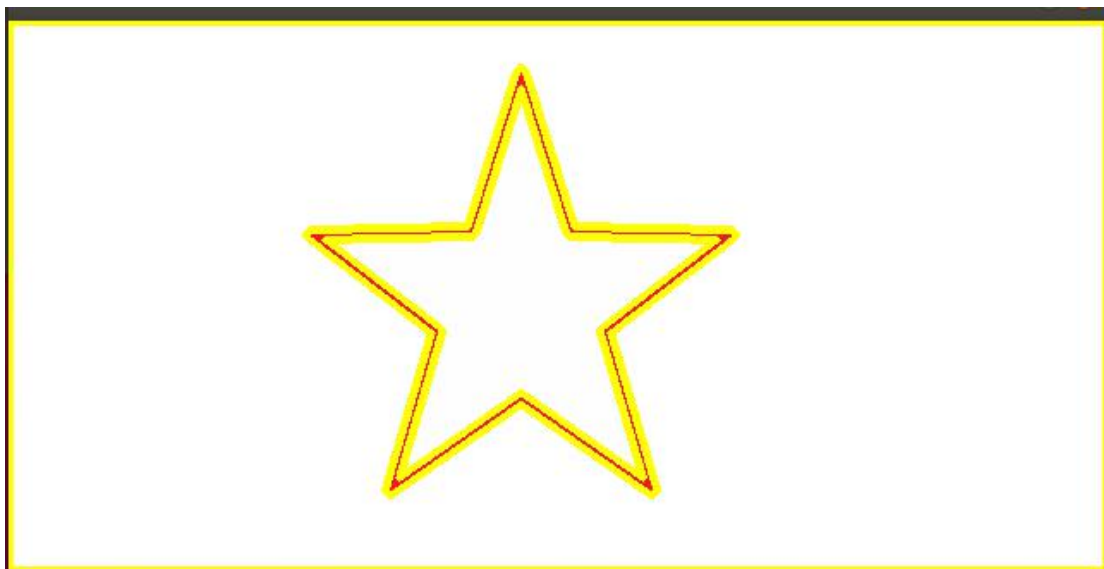
2) 输入指令“cd /mnt/hgfs/share/”，并按下回车，进入共享文件夹。

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

3) 输入指令“python3 contours_demo.py”，并按下回车，运行例程。

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

2.2 实现效果



执行后会找到的所有轮廓点位都绘制出来，如上图所示。

2.3 代码分析

可在目录“第 4 章 OpenCV 计算机视觉学习->图像处理进阶篇->第 13 课 图像处理——轮廓介绍及特征->例程源码”下查看例程“contours_demo.py”。

```
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()
```

导入模块：导入 cv2 模块。

读取图片：imread 函数读取图片，参数为图片名称。

颜色空间转换：cvtColor 函数转换成 GRAY 颜色空间，参数为转换的图片和转换模式。

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

阈值处理：使用函数 threshold，具体格式和参数如下：

threshold(src, thresh, maxval, type)

- 1) 第一个参数“**src**”，是要阈值处理的图像。
- 2) 第二个参数“**thresh**”，是设定的阈值。
- 3) 第三个参数“**maxval**”，当 type 指定为 THRESH_BINARY 或 THRESH_BINARY_INV 时，才需要设置该值，指的是高于（低于）阈值时赋予的新值。
- 4) 第四个参数“**type**”，阈值的处理方法，cv2.THRESH_BINARY 是超过阈值部分取 maxval，否则取 0。

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

寻找轮廓：使用函数 **findContours**，具体格式和参数如下：

findContours(img, mode, method)

- 1) 第一个参数“**img**”，代表需要处理的图像。
- 2) 第二个参数“**mode**”，代表轮廓的检测模式，具体模式如下：
 - ◆ RETR_EXTERNAL = 0，只检测最外面的轮廓；
 - ◆ RETR_LIST = 1，检测所有轮廓，不建立等级关系，所有轮廓放在一个列表中；
 - ◆ RETR_CCOMP = 2，检索所有的轮廓，并将它们组织为两层；
 - ◆ RETR_TREE = 3，按照树形存储轮廓，从右到左一层一层检测。
- 3) 第三个参数“**method**”，代表近似查找轮廓的方法，具体方法如下：
 - ◆ CHAIN_APPROX_NONE，保存所有轮廓上的点；
 - ◆ CHAIN_APPROX_SIMPLE：压缩水平的、垂直的、斜的部分，即只保留他们的角点坐标，例如一个矩形轮廓只需 4 个点来保存轮廓信息。

注意：在 OpenCV4.2 以上版本，该函数只返回两个值，“**contours**”和“**hierarchy**”，代表轮廓和层级关系，不需要“**binary**”。

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

轮廓绘制：使用函数 **drawContours**，具体格式和参数如下：

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

- 1) 第一个参数 “**image**”，代表要绘制轮廓的图像。
- 2) 第二个参数 “**contours**”，代表找到的所有轮廓坐标点。
- 3) 第三个参数 “**contourIdx**”，代表绘制轮廓的编号，-1 为绘制全部轮廓。
- 4) 第四个参数 “**color**”，代表轮廓的颜色。
- 5) 第五个参数 “**thickness**”，代表绘制轮廓的宽度，-1 为填充轮廓。

3.轮廓特征矩

特征矩代表了一个轮廓、一幅图像的全局特征，矩信息包含了对应对象不同类型的几何特征，特征矩分为三种：空间矩、中心矩和归一化中心矩。

1) 空间矩：也叫几何矩，是图像的大小面积及周长等。包括零阶矩： m_{00} ，一阶矩： m_{10}, m_{01} ，二阶矩： m_{20}, m_{11}, m_{02} ，三阶矩： $m_{30}, m_{21}, m_{12}, m_{03}$ 。

2) 中心矩：对于高阶图像，特征矩会随着位置的变化而变化，为了解决这个问题中心矩就应运而生，它通过减去均值而获取平移的不变性，因而能比较不同位置的两个对象是否一致，即中心矩具有平移不变性特征。

包括二阶中心矩： $\mu_{20}, \mu_{11}, \mu_{02}$ ，三阶中心矩： $\mu_{30}, \mu_{21}, \mu_{12}, \mu_{03}$ 。

3) 归一化中心矩：除平移之外，有些图像我们还会碰到缩放的情况，即在缩放后也能判断其特征，归一化中心矩通过除以物体总尺寸而获得缩放不变性。

包括二阶 Hu 矩： $\eta_{20}, \eta_{11}, \eta_{02}$ ，三阶 Hu 矩： $\eta_{30}, \eta_{21}, \eta_{12}, \eta_{03}$ 。接下来通过分析出的轮廓，计算出轮廓的特征矩，面积和周长。



3.1 实验步骤

注意：

1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 13 课 图像处理——轮廓介绍及特征->例程源码”下的例程“moments_demo.py”和示例图片“test.jpg”复制到共享文件夹。

2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

2) 输入指令“cd /mnt/hgfs/share/”，并按下回车，进入共享文件夹。

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

3) 输入指令“python3 moments_demo.py”，并按下回车，运行例程。

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

3.2 实现效果

```
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
```

面积和周长的单位为像素，这里计算的是图像最外围的轮廓。

3.3 代码分析

可在目录“第 4 章 OpenCV 计算机视觉学习->图像处理进阶篇->第 13 课 图像处理——轮廓介绍及特征->例程源码”下查看例程“moments_demo.py”。

```
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)
```

取出最外围的轮廓：在轮廓列表中，取出下标为 0 的第一个轮廓。

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

特征矩计算：使用函数 **moments** 计算特征矩，具体格式和参数如下：

moments(array,binaryImage)

- 1) 第一个参数“**array**”为轮廓点位。
- 2) 第二个参数“**binaryImage**”默认值是 False，如果为 True，则所有非零的像素都会按值 1 对待，也就是说相当于对图像进行了二值化处理。

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

面积计算：使用函数 **countourArea** 计算面积，具体格式和参数如下：

countourArea(contour)，“**contour**”为轮廓列表中的一个轮廓。

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

周长计算：使用函数 **arcLength** 计算周长，具体格式和参数如下：

arcLength(curve,closed)

- 1) 第一个参数“**curve**”为轮廓。
- 2) 第二个参数“**closed**”为轮廓是否闭合，若闭合设为 True，否则设为 False。

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

4.多边形逼近

寻找后的轮廓信息“contours”可能过于复杂不平滑，可以用 `approxPolyDP` 函数对该多边形曲线做适当近似，这就是轮廓的多边形逼近。

该函数是以多边形去逼近轮廓，采用的是 Douglas-Peucker 算法（方法名中的 DP），DP 算法原理比较简单，核心就是不断找多边形最远的点加入形成新的多边形，直到最短距离小于指定的精度。

接下来通过多边形逼近来分析出物体的轮廓。



4.1 实验步骤

注意：

1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 13 课 图像处理——轮廓介绍及特征->例程源码”下的例程“`approx_demo.py`”和示例图片“`test.jpg`”复制到共享文件夹。

2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“**Ctrl+Alt+T**”，打开命令行终端。

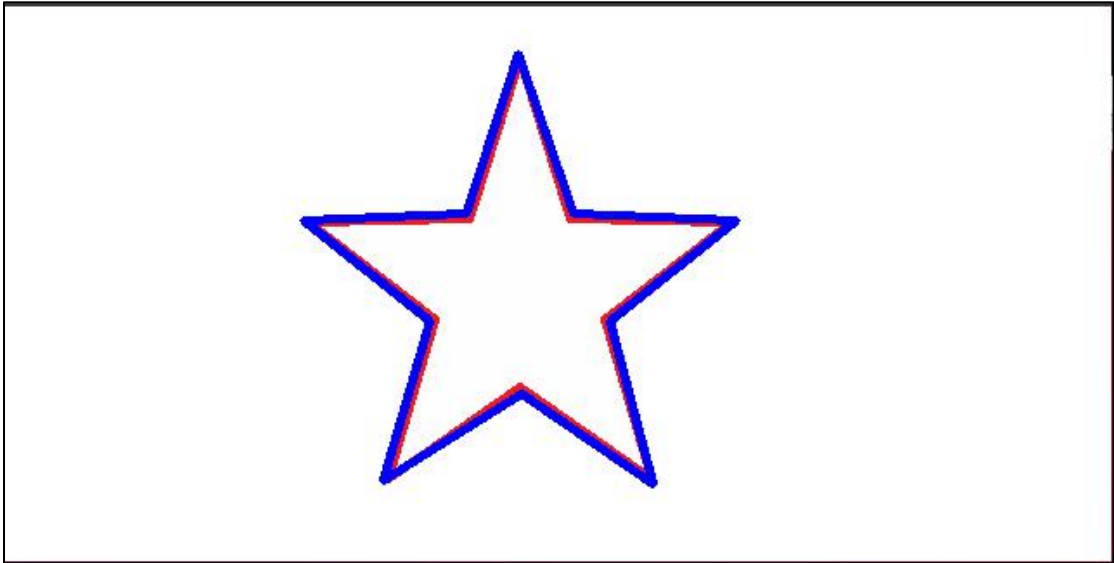
2) 输入指令“`cd /mnt/hgfs/share/`”，并按下回车，进入共享文件夹。

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

3) 输入指令“`python3 approx_demo.py`”，并按下回车，运行例程。

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


4.2 实现效果



多边形逼近后的轮廓会尽量贴合图形。

4.3 代码分析

可在目录“第4章 OpenCV 计算机视觉学习->图像处理进阶篇->第13课 图像处理——轮廓介绍及特征->例程源码”下查看例程“approx_demo.py”。

```
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()
```

取出轮廓：在轮廓列表中，取出下标为1的第二个轮廓。

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

多边形逼近：使用函数 `approxPolyDP`，具体格式和参数如下：

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

- 1) 第一个参数“**curve**”，是查找的轮廓；
- 2) 第二个参数“**epsilon**”，是表示精度，数字越小代表精度越低，表示和图形轮廓越

吻合;

- 3) 第三个参数“closed”，是轮廓是否闭合，若闭合设为 True，否则设为 False。

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

5.轮廓凸包

凸包跟逼近多边形很像，只不过它是物体最外层的凸多边形。凸包指的是完全包含原有轮廓，并且仅由轮廓上的点所构成的多边形。凸包的每一处都是凸的，即在凸包内连接任意两点的直线都在凸包的内部。在凸包内，任意连续三个点的内角小于 180° 。

接下来通过轮廓凸包来分析出物体的轮廓。



5.1 实验步骤

注意:

- 1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 13 课 图像处理——轮廓介绍及特征->例程源码”下的例程“hull_demo.py”和示例图片“test.jpg”复制到共享文件夹。

- 2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

- 3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。
-

- 1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

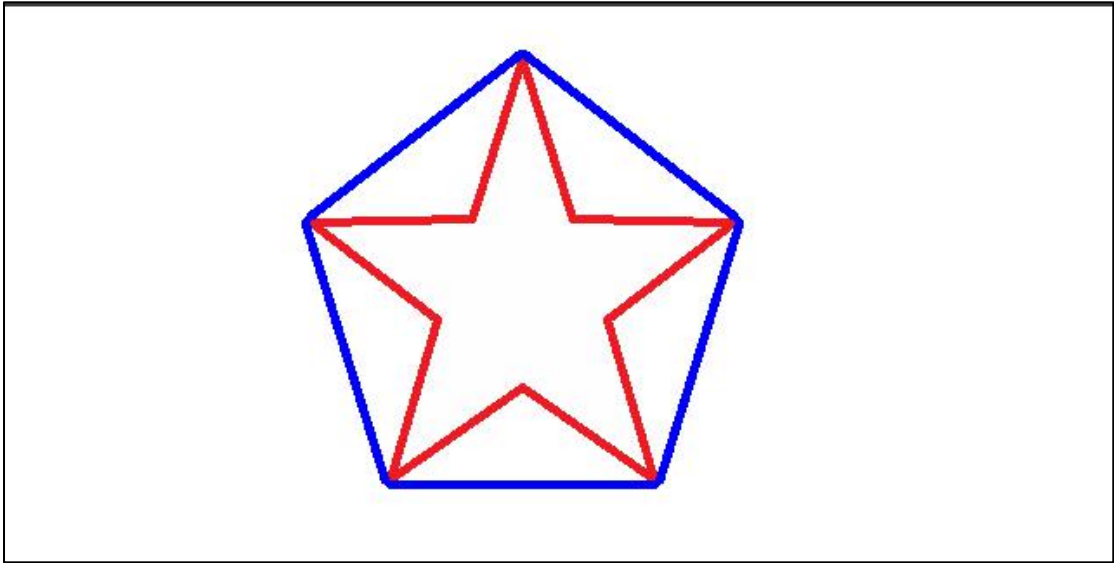
- 2) 输入指令“cd /mnt/hgfs/share/”，并按下回车，进入共享文件夹。

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

- 3) 输入指令“python3 hull_demo.py”，并按下回车，运行例程。

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

5.2 实现效果



轮廓凸包会连接轮廓的顶点部分。

5.3 代码分析

可在目录“第4章 OpenCV 计算机视觉学习->图像处理进阶篇->第13课 图像处理——轮廓介绍及特征->例程源码”下查看例程“hull_demo.py”。

```
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()
```

轮廓凸包：使用函数 `convexHull`，具体格式和参数如下：

`convexHull(points, clockwise,)`

- 1) 第一个参数“**points**”，是查找的轮廓。
- 2) 第二个参数“**clockwise**”，True 为顺时针绘制，False 为逆时针绘制。

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

6.外接矩形

外接矩形分为带旋转角度的最小外接矩形和常规外接矩形。

最小外接矩形：按照物体的角度以最小的面积进行衔接，可以知道物体有没有旋转。

常规外接矩形：把物体扩在里面但是面积最大。

接下来会分布绘制物体的常规和最小外接矩形。



6.1 实验步骤

注意：

1) 需要先将目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第 13 课 图像处理——轮廓介绍及特征->例程源码”下的例程“rect_demo.py”和示例图片“test.jpg”复制到共享文件夹。

2) 共享文件夹的配置方法可查看目录“第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法”下的文档。

3) 输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

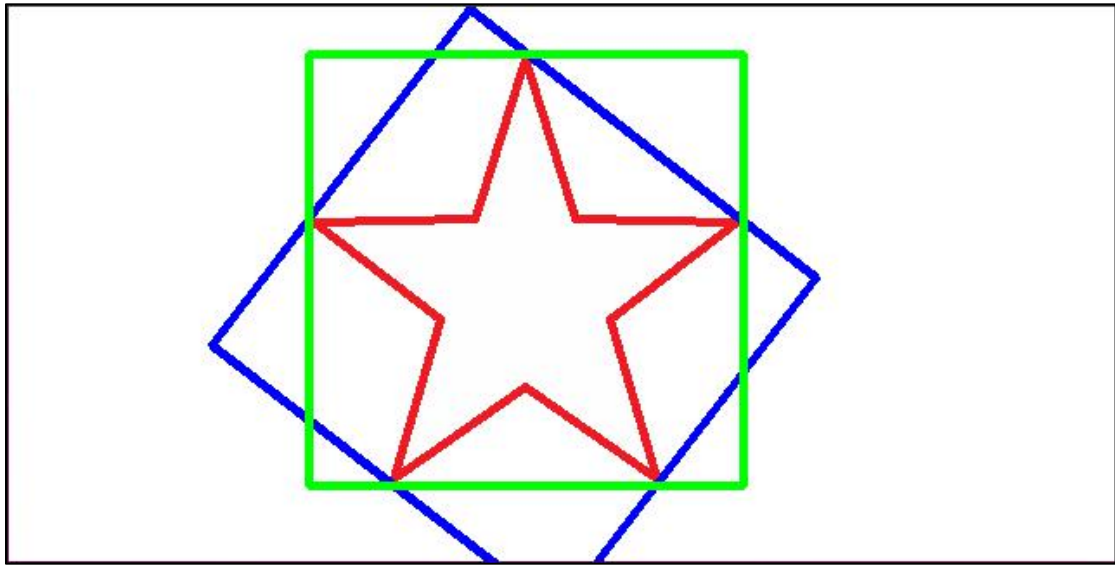
2) 输入指令“cd /mnt/hgfs/share/”，并按下回车，进入共享文件夹。

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

3) 输入指令“python3 rect_demo.py”，并按下回车，运行例程。

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

6.2 实现效果



绿色为常规的外接矩形，蓝色为最小的外接矩形。

6.3 代码分析

可在目录“第四章 OpenCV 计算机视觉学习->图像处理进阶篇->第13课 图像处理——轮廓介绍及特征->例程源码”下查看例程“rect_demo.py”。

```
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()
```

获取最小的外接矩形：使用函数 **minAreaRect**，具体格式和参数如下：

minAreaRect(points)，“points ”为轮廓，返回值“**RotatedRect**”包含起始点坐标，宽度，高度和角度。

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

获取常规的外接矩形：使用函数 **boundingRect**，具体格式和参数如下：

boundingRect (array) , “array” 为轮廓, 返回值 “Rect ” 包含起始点坐标, 宽度和高度。

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

获取最小外接矩形的顶点坐标: 使用函数 **boxPoints**, 具体格式和参数如下:

boxPoints(rect), “rect” 为需要获取顶点的矩形,数据类型为浮点型。

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

数值取整: 使用函数 **int0**, 具体格式和参数如下:

int0(date), “date” 为需要取整的数据。

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

绘制矩形: 使用函数 **rectangle**, 具体格式和参数如下:

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

- 1) 第一个参数 “**src**”, 代表要绘制轮廓的图像。
- 2) 第二个参数 “**pt1**”, 代表矩形其中一个顶点。
- 3) 第三个参数 “**pt2**”, 代表 pt1 的对角顶点。
- 4) 第四个参数 “**color**”, 代表矩形的颜色。
- 5) 第五个参数 “**thickness**”, 代表绘制矩形的宽度, -1 为填充矩形。

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