图像轮廓

电子信息工程系

袁羽

目录

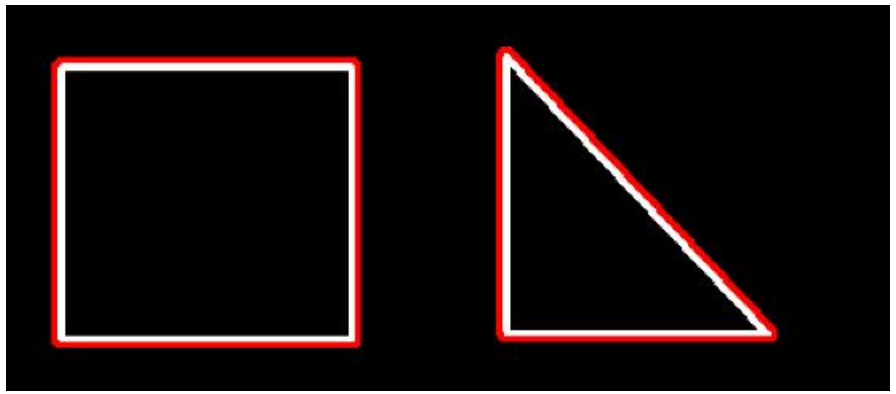
CONTENTS

一 图像轮廓

图像轮廓是具有相同颜色或灰度的连续点的曲线。轮廓在形状分析和物体的检测和识别中很有用。

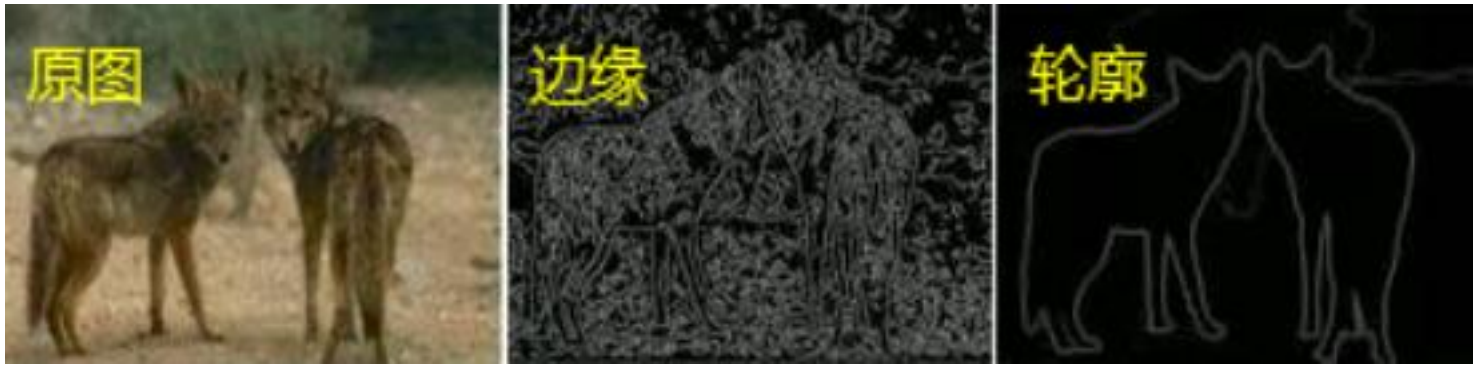
轮廓的作用:

- 用于图形分析
- 物体的识别和检测



一 图像轮廓

图像轮廓和图像边缘不是一回事，图像边缘不是图像轮廓。图像边缘是图像中的线条，这些线条是不连续的、零散的线段，把有梯度的像素点提取出来就可以了，这是边缘检测的操作手法。而轮廓是一系列相连的点组成的曲线，代表了物体的基本外形，相对于边缘，轮廓是连续的。图像轮廓是图像中非常重要的一个特征信息，通过对图像轮廓的操作，我们能够获取目标图像的大小、位置、方向等信息。



二 轮廓检测

OpenCV提供了 cv2.findContours()函数，实现图像轮廓的查找：

contours,hierarchy=cv2.findContours(img,mode,method)

输入参数：

img： 输入原始图。从黑色背景中查找白色对象。因此，对象必须是白色的，背景必须是黑色的

mode： 轮廓检索模式，决定了轮廓的提取方式。

method： 轮廓的保存方式。

返回值：

contours： 返回的轮廓，为列表的形式，例如contours[i]表示第i个轮廓。

hierarchy： 图像的轮廓层次。

二 轮廓检测

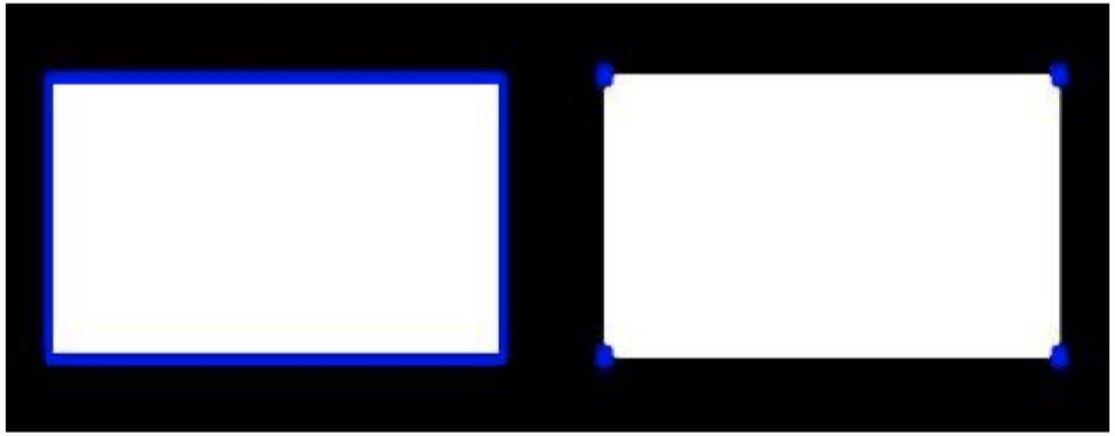
mode： 轮廓检索模式，决定了轮廓的提取方式。

检索模式	功能
cv2.RETR_EXTERNAL	只检测外轮廓
cv2.RETR_LIST	检测的轮廓不建立等级关系
cv2.RETR_CCOMP	建立两个等级的轮廓
cv2.RETR_TREE	建立一个等级树结构的轮廓

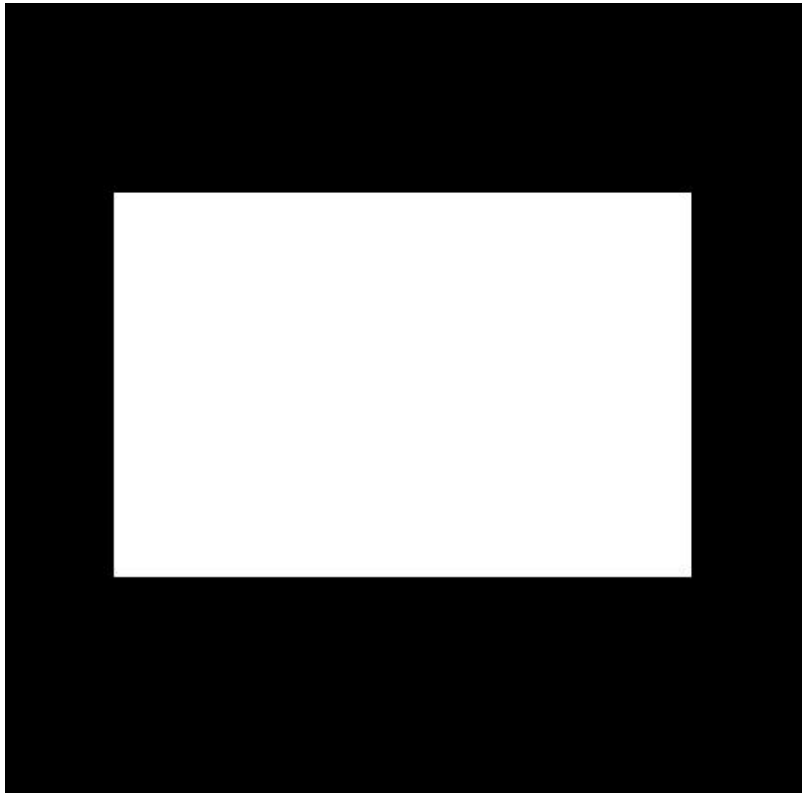
二 轮廓检测

method： 轮廓的保存方式。

轮廓保存方式	功能
cv2.CHAIN_APPROX_NONE	保存所有轮廓上的点
cv2.CHAIN_APPROX_SIMPLE	压缩水平方向，垂直方向，对角线方向的元素，只保留该方向的终点坐标



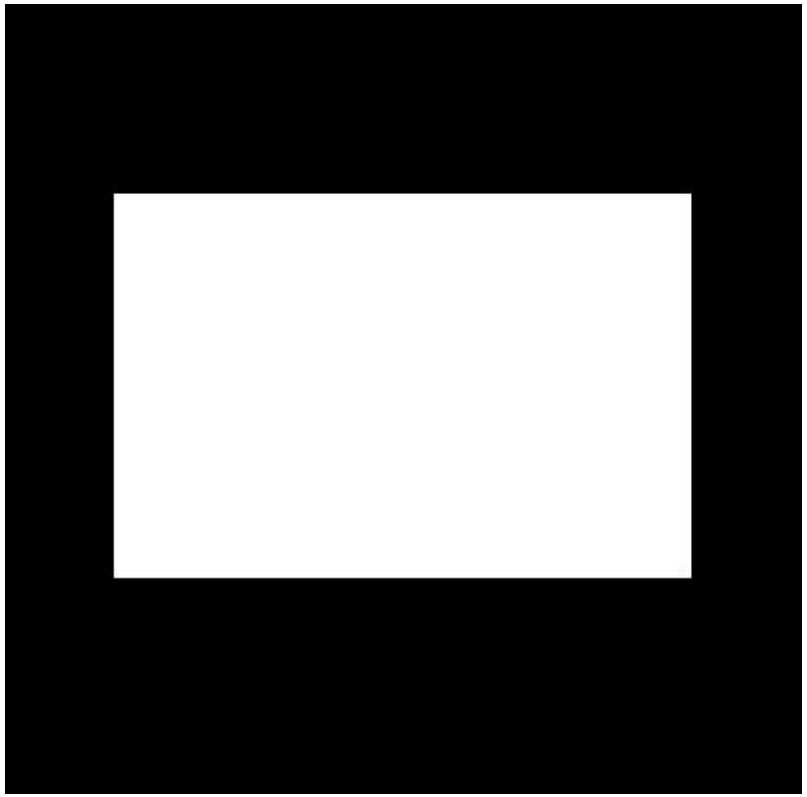
例 查找简单图形轮廓。



例 查找简单图形轮廓。

```
import cv2
import numpy as np
img = cv2.imread('rec.jpg')
# 变成单通道的黑白图片
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# 二值化
thresh, binary = cv2.threshold(img2, 150, 255, cv2.THRESH_BINARY)
# 查找轮廓，返回两个参数，分别是轮廓和层级
contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
# contours是list，里面放的是ndarray，每个ndarray表示一个contour
print(contours)
print(hierarchy)
```

例 查找简单图形轮廓。



运行结果：
(array([[[62, 108]],

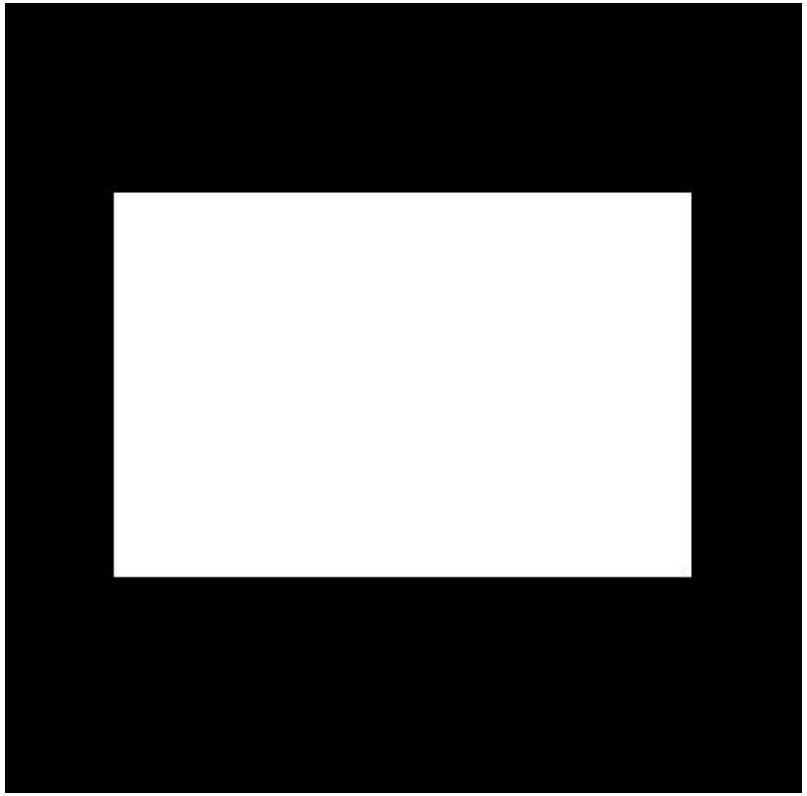
[[62, 326]],

[[390, 326]],

[[390, 108]]], dtype=int32),)
[[[-1 -1 -1 -1]]]

例 查找简单图形轮廓。

将图像轮廓保存方式改为：cv2.CHAIN_APPROX_NONE



运行结果：

```
(array([[[ 62, 108]],

       [[ 62, 109]],

       [[ 62, 110]],

       ...,

       [[ 65, 108]],

       [[ 64, 108]],

       [[ 63, 108]]], dtype=int32),)
[[[-1 -1 -1 -1]]]
```

二 轮廓检测

使用 cv.findContours() 函数来检测图像中的对象，有时对象位于不同的位置。在某些情况下，某些形状位于其他形状内，就像嵌套的数字一样。在这种情况下，我们将外部的一个称为父级 ，将内部的一个称为子级 。这样，图像中的轮廓彼此之间就具有某种关系，这种关系的表示称为轮廓层次结构 。

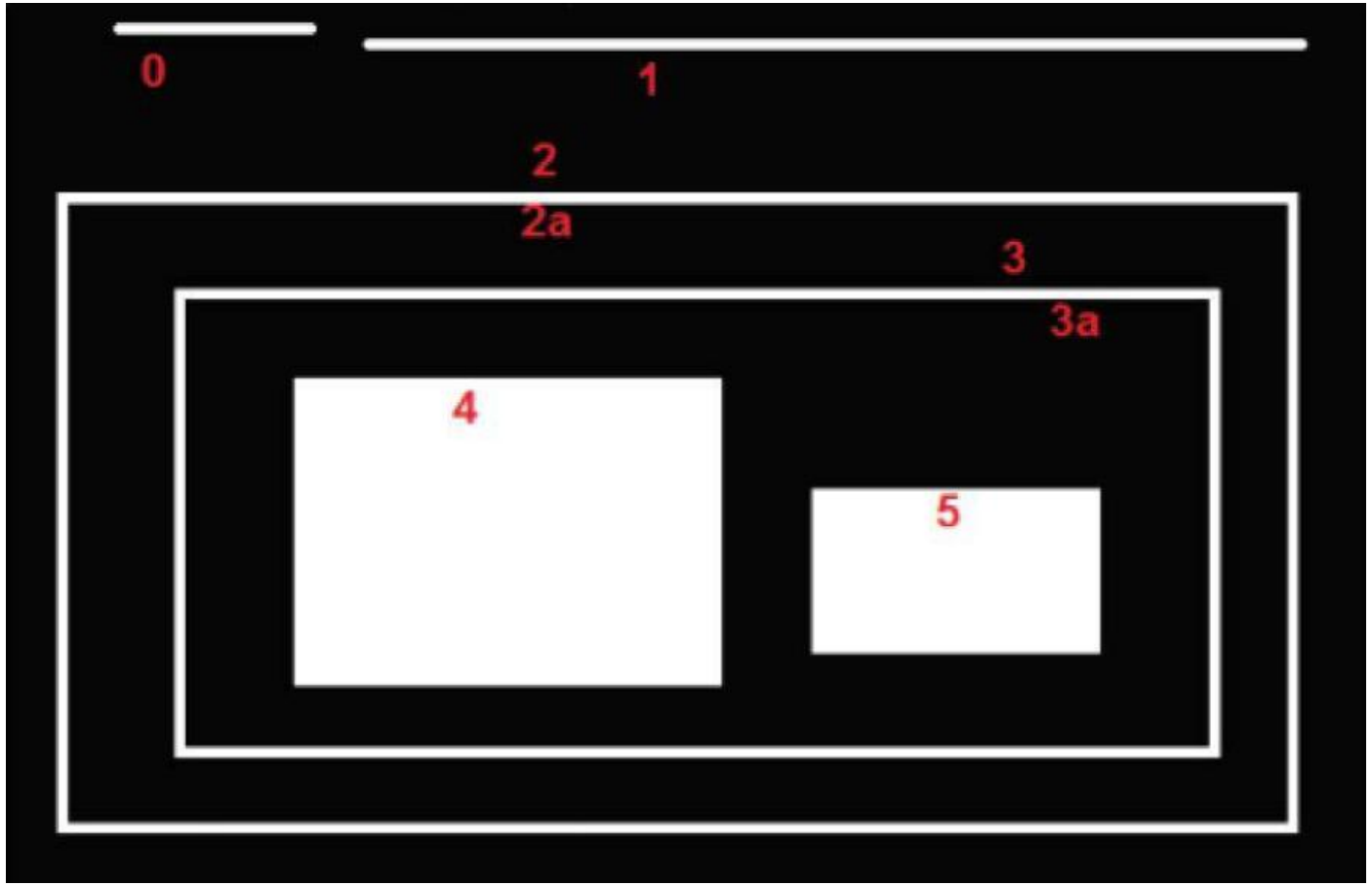
轮廓查找函数：

contours,hierarchy=cv2.findContours(img,mode,method)

返回值hierarchy表示图像的轮廓层次，OpenCV将hierarchy表示为四个值的数组：

[Next，Previous，First_Child，Parent]

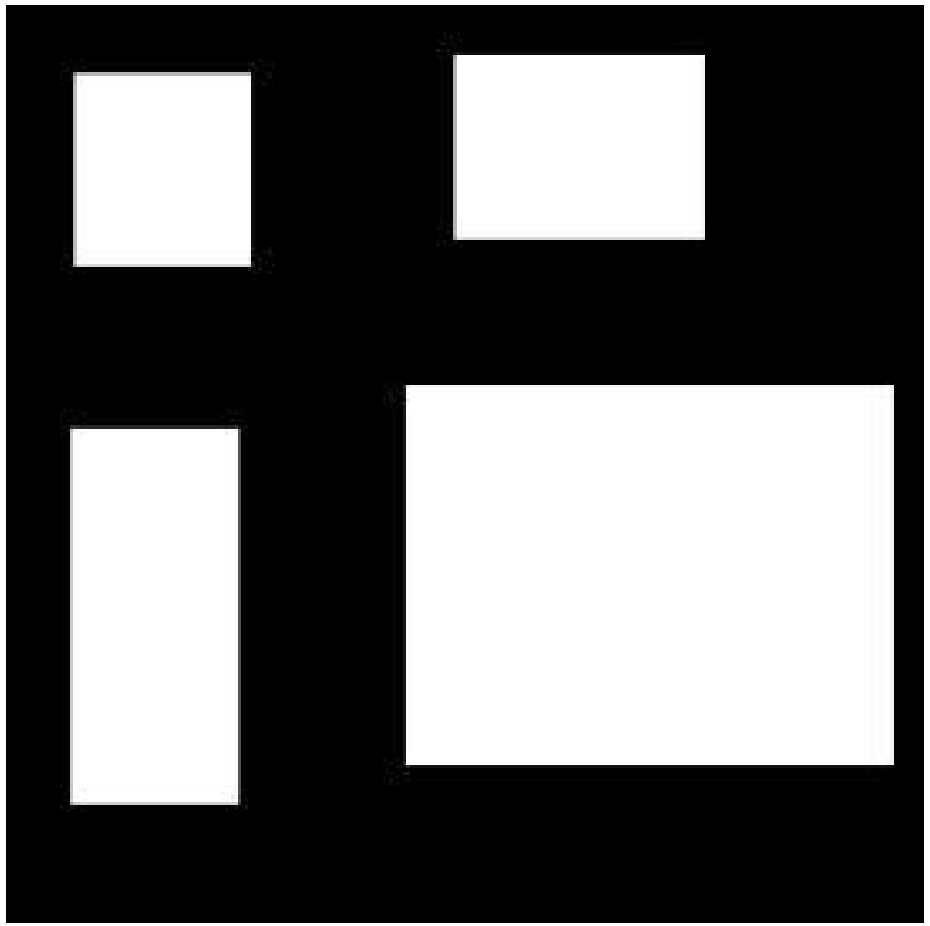
二 轮廓检测



轮廓0,1,2在最外部，它们处于第0级层次结构中，处于相同的层次结构级别中，属于兄弟关系。

轮廓2a 可以将其视为轮廓2的子级，为第1级。以此类推：3是2a的子级，为第2级。3a是3的子轮廓，为第3级，轮廓4,5是轮廓3a的子级，为第4级，它们位于最后的层次结构级别。从编号方式上来说，轮廓4是轮廓3a的第一个子元素。

例 查看以下图像轮廓的层次结构



```
[[[ 1 -1 -1 -1]
 [ 2  0 -1 -1]
 [ 3  1 -1 -1]
 [-1  2 -1 -1]]]
```

三 轮廓绘制

drawContours(img, contours, contoursIdx, color, thickness,...)

img：在哪张图片上进行展示

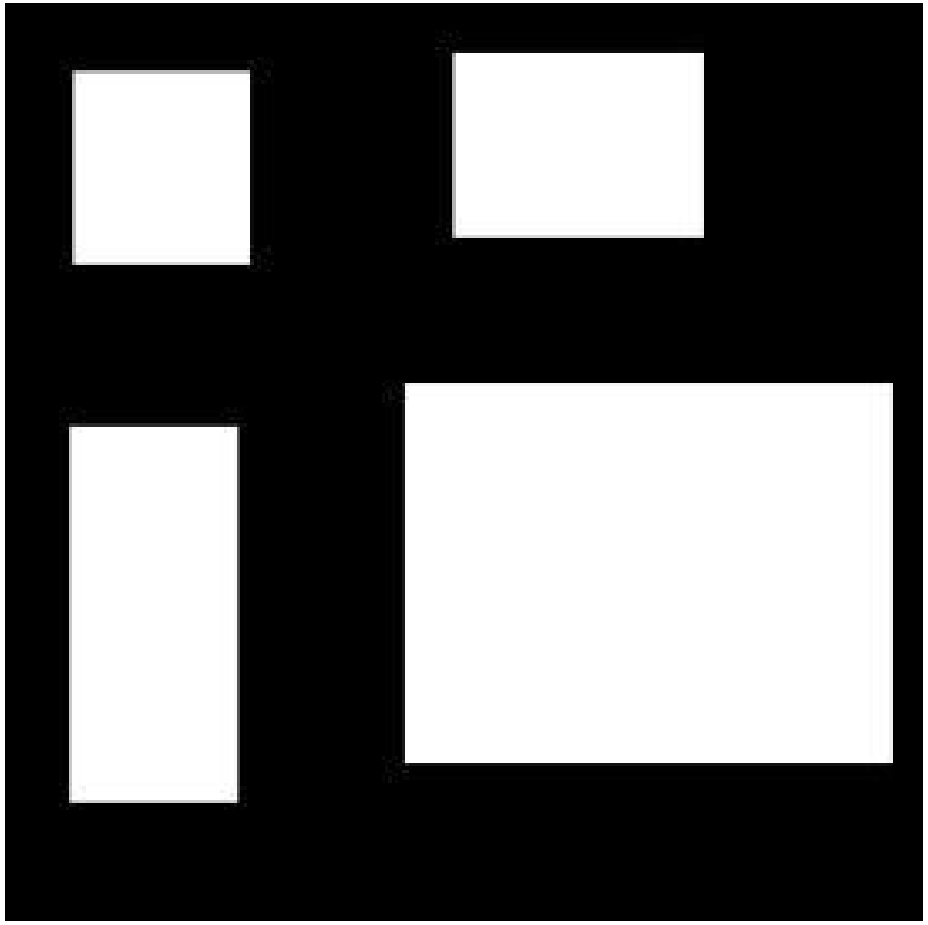
contours ： 轮廓坐标点，也就是findContours第一个返回值。

contoursIdx：轮廓顺序号，拿到的所有轮廓坐标点会按照储存方式有个顺序，从0开始【-1表示绘制所有轮廓】

color ： 轮廓的颜色，书写为（0，0，255）

thickness ： 显卡，1为细线，2为粗线，-1表示对整个轮廓进行填充

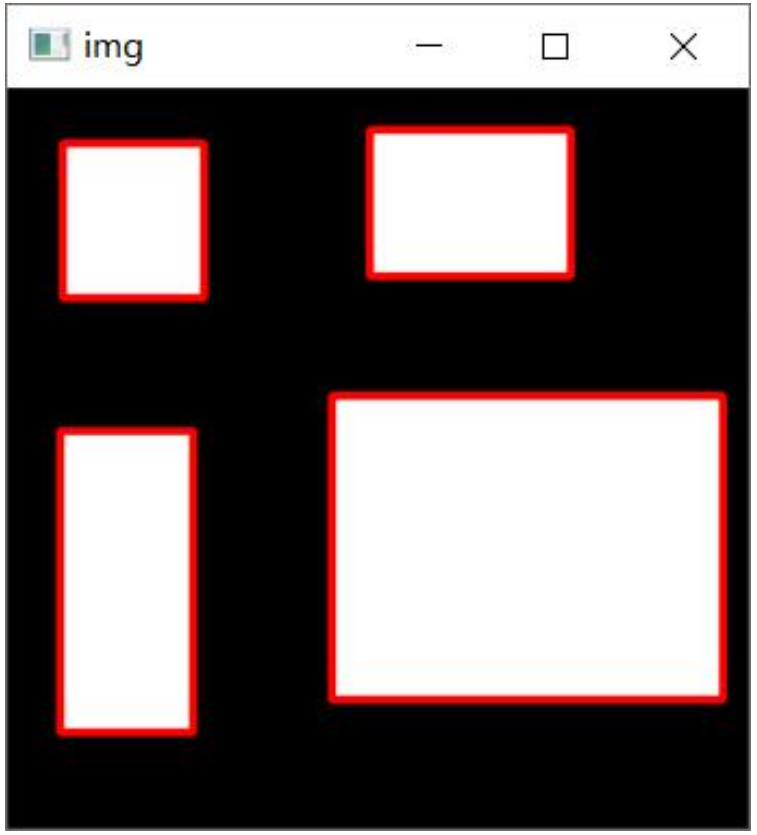
例 对简单图形进行轮廓寻找，并绘制轮廓。



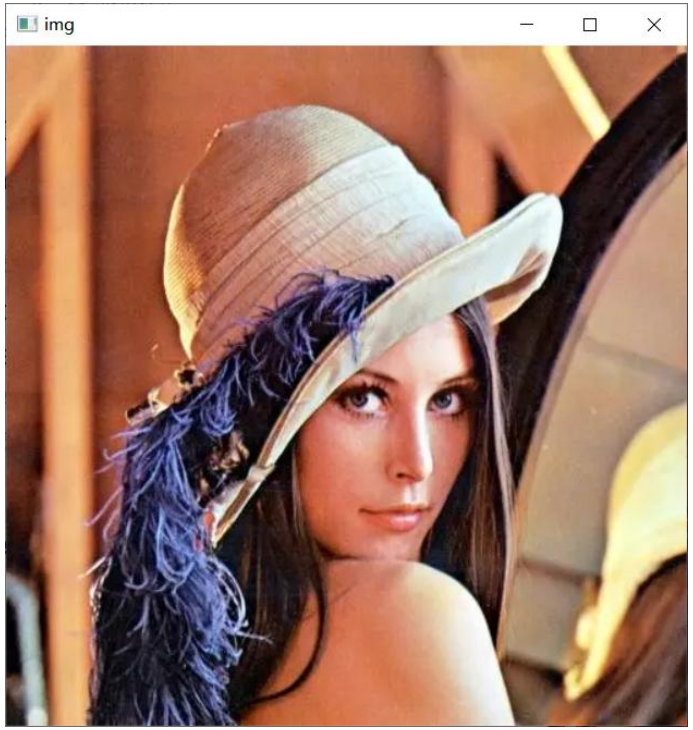
例 对简单图形进行轮廓寻找，并绘制轮廓。

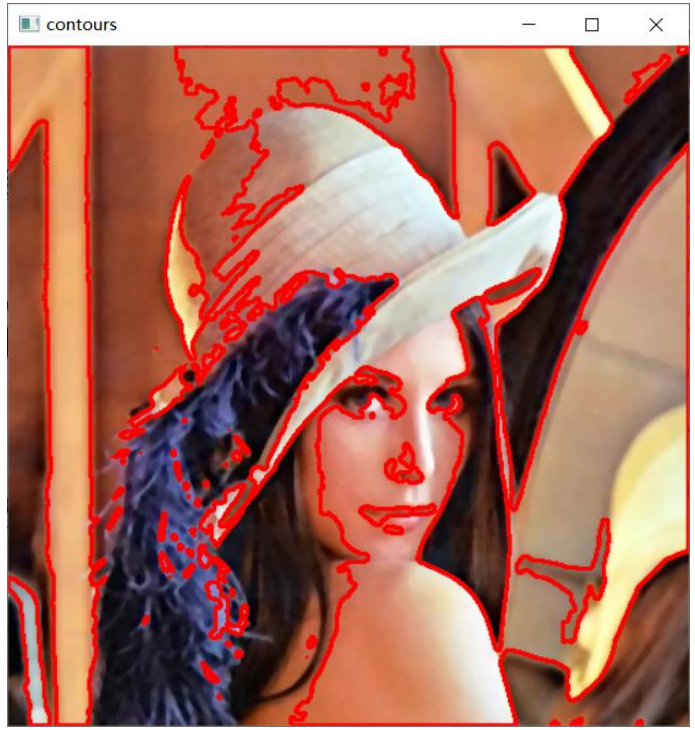
```
import cv2

import numpy as np

img = cv2.imread('cc.jpg')

img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 变成单通道的黑白图片

thresh, binary = cv2.threshold(img2, 150, 255, cv2.THRESH_BINARY) # 二值化

# 查找轮廓，新版本返回两个参数，分别是轮廓和层级

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

# contours是list，里面放的是ndarray，每个ndarray表示一个contour

cv2.drawContours(img, contours, -1, (0, 0, 255), 2) # 绘制轮廓会直接修改原图

cv2.imshow('img', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

例 对简单图形进行轮廓寻找，并绘制轮廓。



例 对复杂图像进行轮廓寻找，并绘制轮廓。





例 对复杂图像进行轮廓寻找，并绘制轮廓。

```
import cv2
img = cv2.imread("lenna.jpg")  # 读取原图
cv2.imshow("img", img)  # 显示原图
img = cv2.medianBlur(img, 5)  # 使用中值滤波去除噪点
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # 原图从彩图变成单通道灰度图像
t, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)  # 灰度图像转化为二值图像
cv2.imshow("binary", binary)  # 显示二值化图像
contours, hierarchy = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)# 获取轮廓
cv2.drawContours(img, contours, -1, (0, 0, 255), 2)  # 在原图中绘制轮廓
cv2.imshow("contours", img)  # 显示绘有轮廓的图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

四 轮廓特征

我们绘制图像中的物体轮廓的目的是什么？目的就是获取图像中目标的大小、位置、方向等信息。 但是怎么通过轮廓获得图像中目标的大小位置方向？通过计算轮廓的一些特征去判断。比如计算轮廓的面积、周长、质心、边界等信息去判断。

在计算轮廓时可能并不需要实际的轮廓，而仅需要一个接近轮廓的多边形，绘制这个多边形称为轮廓拟合。

四 轮廓特征

在OpenCV提取轮廓之后，还可以进行许多操作：

ContourArea()：计算轮廓区域的面积

ArcLength()：计算轮廓长度

BoundingRect()：轮廓的外包矩形

MinAreaRect()：轮廓的最小外包矩形

MinEnclosingCircle()：轮廓的最小外包圆

fitEllipse()：用椭圆拟合轮廓

approxPolyDP()：逼近多边形拟合轮廓

ConvexHull()：提取轮廓的凸包

四 轮廓特征

1、轮廓的面积和周长

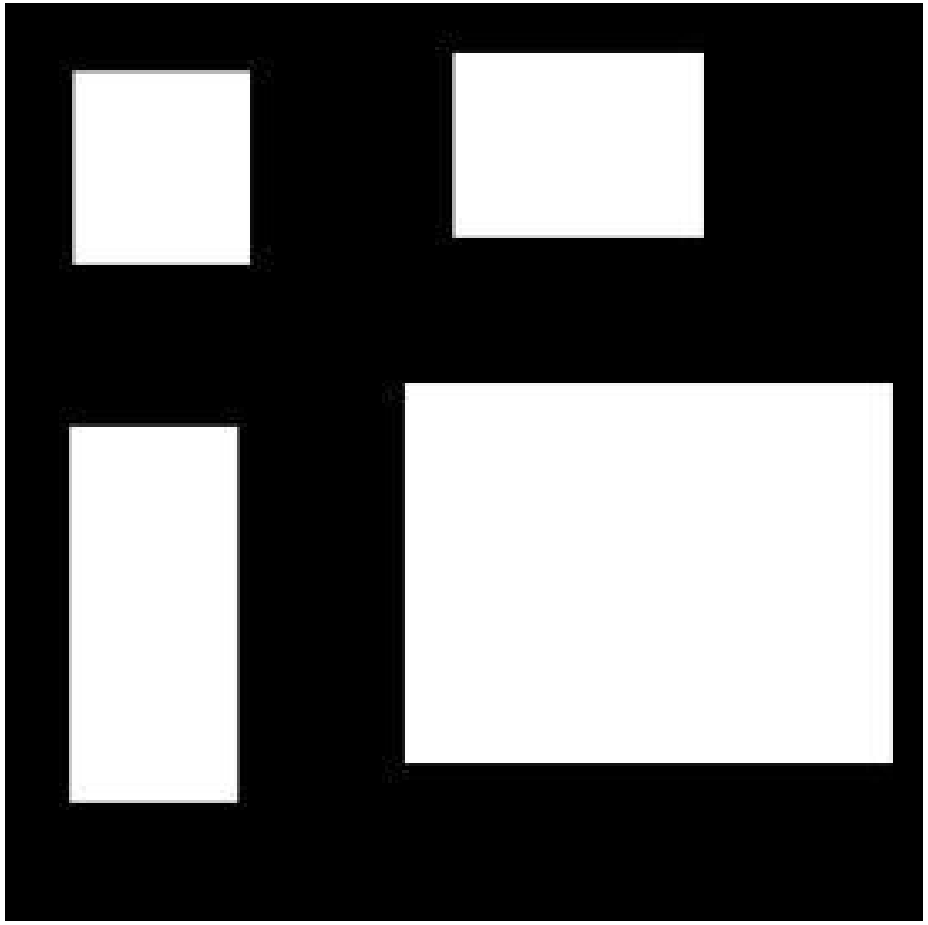
轮廓面积是指每个轮廓中所有像素点围成区域的面积，单位为像素。轮廓周长是指每个轮廓中所有像素点围成区域的周长，单位同样为像素。

通过分析轮廓面积和轮廓周长，我们可以区分物体的大小，识别物体的不同，同时还能分析出一些其他内容，例如，正方形区域的周长和面积是有固定关系的，圆形区域的周长和面积是有固定关系的。通过计算轮廓面积和周长，再结合这些固定关系，我们是可以得到一些结论的。

轮廓面积api：
countourArea（contour），contour指列表中具体的某一个轮廓，而且算出的面积是图像中的面积，不是实际的。

轮廓的周长api：
arcLength(curve , closed)
curve： 计算的轮廓
closed： 轮廓是否闭合，若闭合设为True，否则设为False

例 对简单图形进行轮廓寻找，获取轮廓的面积和周长。



例 对简单图形进行轮廓寻找，获取轮廓的面积和周长。

```
import cv2
import numpy as np
img = cv2.imread('cc.jpg')
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 变成单通道的黑白图片
thresh, binary = cv2.threshold(img2, 150, 255, cv2.THRESH_BINARY) # 二值化
contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
area = cv2.contourArea(contours[0]) #计算面积
len = cv2.arcLength(contours[0],True) #计算周长
print("len = ",len)
print("area=%d"%(area))
cv2.drawContours(img, contours, 0, (0, 0, 255), 2) # 绘制轮廓会直接修改原图
cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

四 轮廓特征

2、多边形逼近

approxPolyDP函数可以对多边形曲线做适当近似，这就是轮廓的多边形逼近，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。

approxCurve = cv2.approxPolyDP(curve, epsilon, closed)

参数说明：

curve 要近似逼近的轮廓

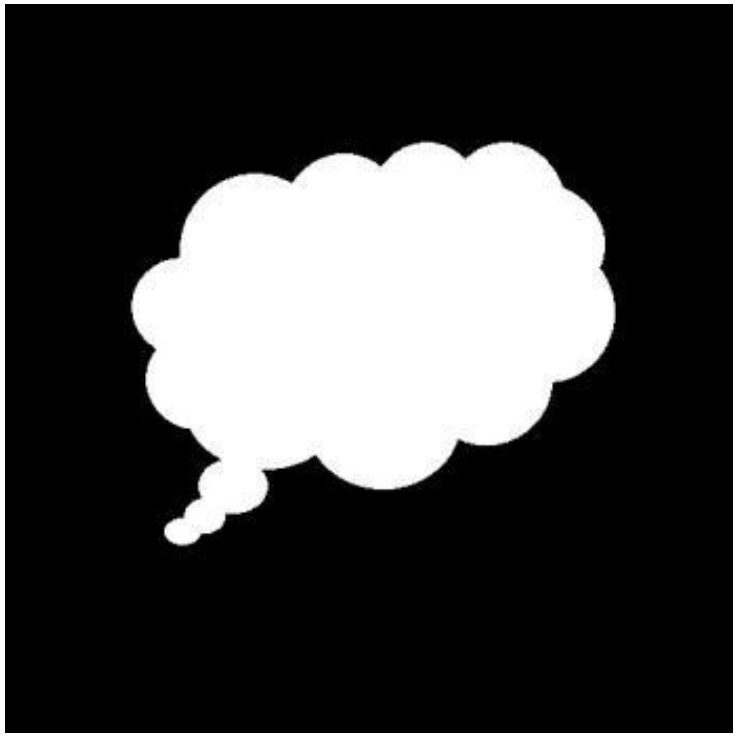
epsilon：指定的精度，也即是原始曲线与近似曲线之间的最大距离。该参数一般设为轮廓周长的百分比形式。

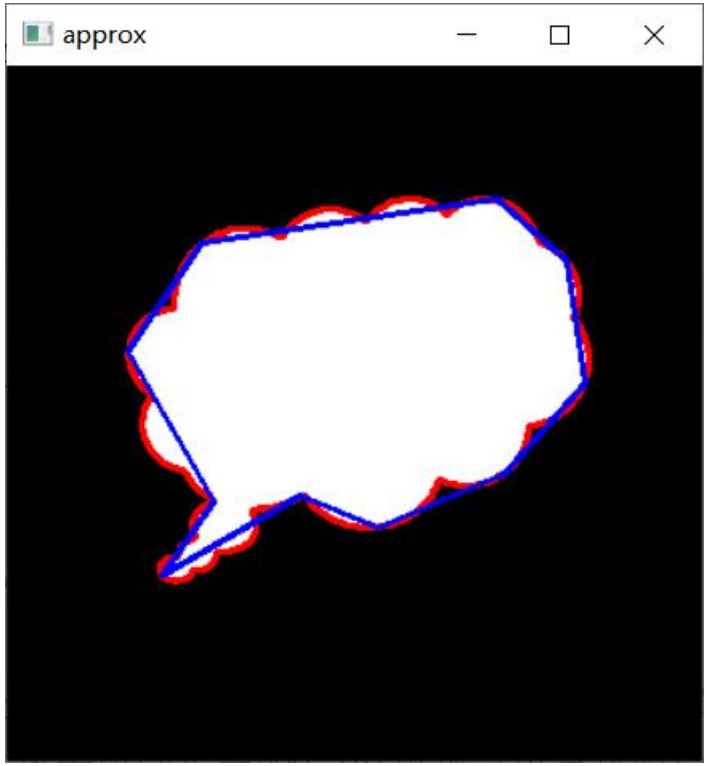
closed：轮廓是否闭合

返回值：

approxCurve:为逼近的多边形的点集，就是多边形逼近的数据，利用drawContours画出来就可以了。

例 对图像进行多边形逼近。





例 对图像轮廓进行多边形逼近。

```
import cv2
import numpy as np
img = cv2.imread('cloud.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, binary = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)# 二值化
contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
cv2.drawContours(img, contours, -1, (0, 0, 255), 2)# 绘制轮廓, 注意, 绘制轮廓会改变原图
# 进行多边形逼近, 返回的是多边形上一系列的点, 即多边形逼近之后的轮廓
epsilon=0.02*cv2.arcLength(contours[0],True) #精度，最大距离为轮廓长度的0.02倍也可以直接给定值
approx=cv2.approxPolyDP(contours[0],epsilon,True)
cv2.drawContours(img,[approx],0,(255,0,0),2)
cv2.imshow("approx",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

四 轮廓特征

3、凸包

多边形逼近是在图像轮廓的里面，而且多边形可能有超过180度的角。而凸包是包在轮廓最外面的凸多边形，是将轮廓上的像素点全部包住，凸包内任意两点的连线都在凸包内部，凸包内的角都是小于180度的。

hull = cv2.convexHull(points [, clockwise, returnPoints])

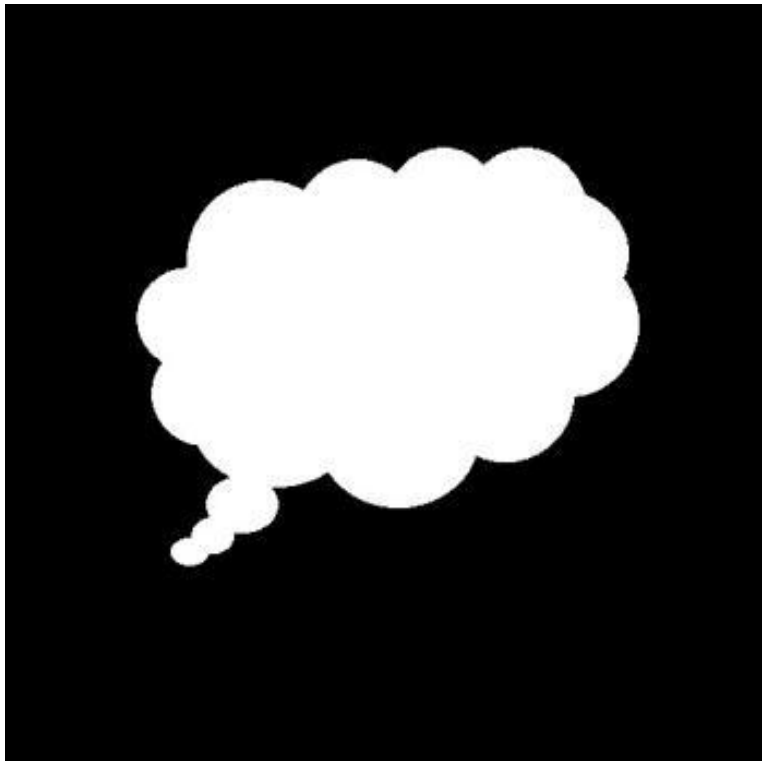
hull：返回的是凸包角点

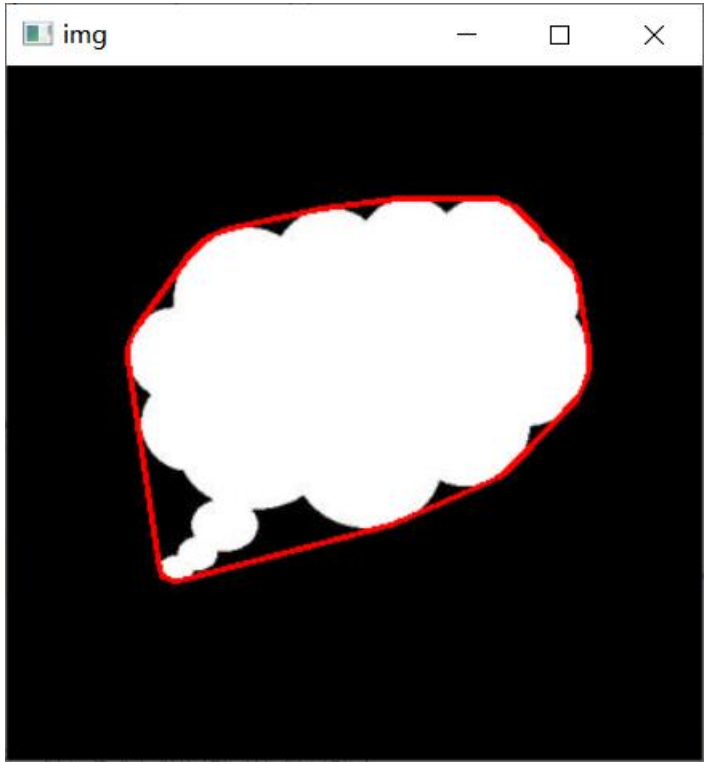
points：轮廓

clockwise：这个参数=True时，凸包角点按顺时针方向排列，反之按逆时针排列

returnPoints：默认值是True，返回凸包角点的xy轴坐标，否则返回轮廓中凸包角点的索引。

例 对图像轮廓绘制凸包。





例 对图像轮廓绘制凸包。

```
import cv2
img = cv2.imread("cloud.jpg")  # 读取原始图像
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # 转为灰度图像
ret, binary = cv2.threshold(gray, 127, 225, cv2.THRESH_BINARY)  # 二值化阈值处理
# 检测图像中出现的所有轮廓
contours, hierarchy = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
hull = cv2.convexHull(contours[0])  # 获取轮廓的凸包
#cv2.drawContours(img, [hull], 0, (0, 0, 255), 2) # 绘制凸包
cv2.polylines(img, [hull], True, (0, 0, 255), 2)  # 绘制凸包
cv2.imshow("img", img)  # 显示图像
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

作业：在OpenCV提取轮廓之后，还可以进行许多操作，请展示一下操作效果：

BoundingRect()：轮廓的外包矩形

MinAreaRect()：轮廓的最小外包矩形

MinEnclosingCircle()：轮廓的最小外包圆

fitEllipse()：用椭圆拟合轮廓

小结

图像轮廓指的是将图像的边缘连接起来形成的一个整体，它是图像的一个重要的特征信息，通过对图像的轮廓进行操作，能够得到这幅图像的大小、位置和方向等信息，用于后续的计算。为此，OpenCV提供了findContours()方法判断图像的轮廓。为了绘制图像的轮廓，OpenCV又提供了drawContours()方法。

THANKS