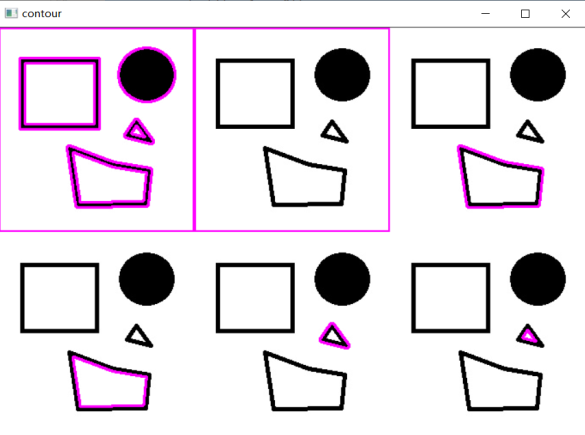
*'''  
2021.1.30FromIvicxDS:openCV;E5图像形态处理：图像金字塔  
'''***'''  
高斯金字塔:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
1.向下采样法【向下为图像缩小的方向(图中箭头)】>>>>>>>>>>>>>>>>>>>>  
 /\_\_\ ↑  
/\_\_\_\_\ ↑  
step1.将Gi与高斯内核卷积【高斯滤波】  
高斯内核:  
1 4 6 4 1  
4 16 24 16 4  
6 24 36 24 6 1/16  
4 16 24 16 4  
1 4 6 4 1  
step2.去除所有偶数行列  
2.向上采样法>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
 /\_\_\ ↓  
/\_\_\_\_\ ↓  
step1.图像每个方向上扩大为原来的两倍，新增的行列以0填充  
10 30  
55 23  
↓  
10 0 30 0  
0 0 0 0  
55 0 23 0  
0 0 0 0  
step2.使用与先前同样的内核(x4)与放大后的图像卷积。获得近似值  
拉普拉斯金字塔\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Li【处理后的结果】=Gi【原图】-pyrUp(pyrDown(Gi))  
step1.低通滤波  
step2.缩小尺寸  
step3.放大尺寸  
step4.图像相减【比较出差异】  
'''**import cv2*#读取格式为BGR*def showPicture(name,picture):  
 *#图像的显示,也可以显示多窗口* cv2.imshow(name,picture)  
 *#在键盘中按任意键退出显示并向后执行语句  
 #cv2.waitKey(1000)表示只显示1秒* cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
img=cv2.imread(**"image/x1.jpg"**)*#高斯金字塔>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*print(img.shape)  
downPicture=cv2.pyrDown(img)  
print(downPicture.shape)  
showPicture(**"pyrUp"**,downPicture)  
*#pyrUp用法相同*down=cv2.pyrDown(img)  
down\_up=cv2.pyrUp(down)  
lupls=img-down\_up  
showPicture(**"LplsPyr"**,lupls)



*'''  
2021.1.30FromIvicxDS:openCV;E5图像形态处理：轮廓检测  
'''***'''  
cv2.findContours(img,mode,method)  
mode:轮廓检测模式  
RETR\_TREE:检测所有轮廓,并重构嵌套轮廓的整个层次【保留所有轮廓信息,最常用】  
method:轮廓逼近方法  
CHAIN\_APPROX\_NONE:以Freeman链码方式输出轮廓【长方形保留所有边】  
CHAIN\_APPROX\_SIMPLE:压缩,结果更精简计算速度快【长方形保留四个顶点】  
  
使用二值图像  
'''**import cv2*#读取格式为BGR*import numpy  
  
def showPicture(name,picture):  
 *#图像的显示,也可以显示多窗口* cv2.imshow(name,picture)  
 *#在键盘中按任意键退出显示并向后执行语句  
 #cv2.waitKey(1000)表示只显示1秒* cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
img=cv2.imread(**"image/B1.jpg"**)*#轮廓检测*grayImage=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)*#转化为灰度图*ret,thresh=cv2.threshold(grayImage,200,255,cv2.THRESH\_BINARY)*#转化为二值图*showPicture(**"thresh"**,thresh)  
  
contours,hierachy=cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE)  
*#[做完二值的结果]【报错?】，轮廓信息，轮廓层级  
  
#绘制轮廓*draw\_img1=img.copy()*#防止后一步修改原图*draw\_img2=img.copy()*#防止后一步修改原图*draw\_img3=img.copy()*#防止后一步修改原图*draw\_img4=img.copy()*#防止后一步修改原图*draw\_img5=img.copy()*#防止后一步修改原图*draw\_img6=img.copy()*#防止后一步修改原图  
  
#↓参数说明:原图，轮廓信息，绘制轮廓的层级【-1表示描绘所有】，画的颜色BGR,画的粗细*r1=cv2.drawContours(draw\_img1,contours,-1,(255,0,255),2)  
r2=cv2.drawContours(draw\_img2,contours,0,(255,0,255),2)  
r3=cv2.drawContours(draw\_img3,contours,1,(255,0,255),2)  
r4=cv2.drawContours(draw\_img4,contours,2,(255,0,255),2)  
r5=cv2.drawContours(draw\_img5,contours,3,(255,0,255),2)  
r6=cv2.drawContours(draw\_img6,contours,4,(255,0,255),2)  
res1=numpy.hstack((r1,r2,r3))  
res2=numpy.hstack((r4,r5,r6))  
res=numpy.vstack((res1,res2))  
showPicture(**"contour"**,res)  
  
*#轮廓特征提取\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#计算图形面积*cnt=contours[2]  
print(cv2.contourArea(cnt))  
*#计算周长，True表示闭合的*print(cv2.arcLength(cnt,True))

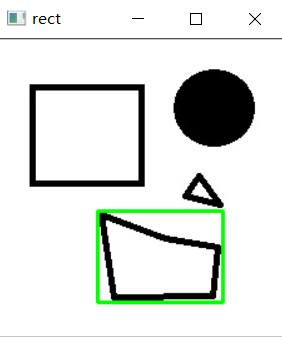


3451.0

265.7401144504547

Process finished with exit code 0

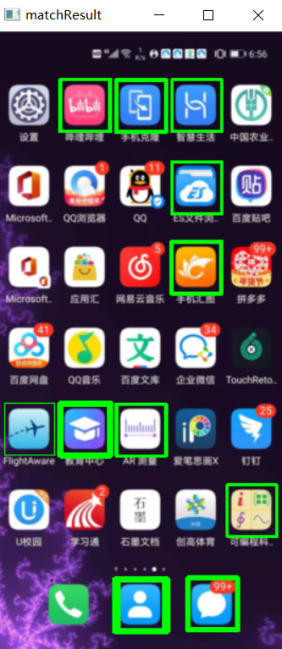
*'''  
2021.1.30FromIvicxDS:openCV;E5图像形态处理：轮廓近似  
'''***'''  
不断二分检测曲线上的点到两个端点连线段的距离是否小于阈值  
'''**import cv2*#读取格式为BGR*import numpy  
  
def showPicture(name,picture):  
 *#图像的显示,也可以显示多窗口* cv2.imshow(name,picture)  
 *#在键盘中按任意键退出显示并向后执行语句  
 #cv2.waitKey(1000)表示只显示1秒* cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
img=cv2.imread(**"image/B2.jpg"**)  
grayImage=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)*#转化为灰度图*ret,thresh=cv2.threshold(grayImage,200,255,cv2.THRESH\_BINARY)*#转化为二值图*contours,hierachy=cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE)  
*#[做完二值的结果]【报错?】，轮廓信息，轮廓层级*cnt=contours[1]  
  
draw\_img=img.copy()  
res=cv2.drawContours(draw\_img,[cnt],-1,(0,0,255),2)  
showPicture(**"res"**,res)  
  
*#轮廓近似>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*elsilon=0.05\*cv2.arcLength(cnt,True)*#周长的10%*approx=cv2.approxPolyDP(cnt,elsilon,True)*#设置轮廓近似的阈值*draw\_img=img.copy()  
res=cv2.drawContours(draw\_img,[approx],-1,(0,0,255),2)  
showPicture(**"approx"**,res)

*'''  
2021.1.30FromIvicxDS:openCV;E5图像形态处理：外接图形  
'''*import cv2*#读取格式为BGR*import numpy  
  
def showPicture(name,picture):  
 *#图像的显示,也可以显示多窗口* cv2.imshow(name,picture)  
 *#在键盘中按任意键退出显示并向后执行语句  
 #cv2.waitKey(1000)表示只显示1秒* cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
img=cv2.imread(**"image/B1.jpg"**)  
gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  
ret,thresh=cv2.threshold(gray,127,255,cv2.THRESH\_BINARY)  
*#contours,hierarchy=cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE)*contours,hierachy=cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE)  
cnt=contours[1]  
  
x,y,w,h=cv2.boundingRect(cnt)  
img=cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)  
  
area=cv2.contourArea(cnt)  
rectArea=w\*h  
extent=float(area)/rectArea  
print(**"轮廓与边界面积比"**,extent)  
  
showPicture(**"rect"**,img)

*'''  
2021.1.30FromIvicxDS:openCV;E5图像形态处理：模板匹配  
'''***'''  
原图：AxB,模板axb->结果矩阵:(A-a+1)x(B-b+1)  
匹配算法:  
TM\_SQDIFF:计算平方不同，值越小，越相关  
TM\_CCORR:计算相关性，值越大，越相关  
TM\_CCOEFF:计算相关系数，值越大，越相关  
TM\_SQDIFF\_NORMED:计算归一化平方不同，越接近0，越相关#也是最大最小值  
TM\_CCORR\_NORMED:计算归一化相关性，越接近1，越相关#也是最大最小值  
TM\_CCOEFF\_NORMED:计算归一化相关系数，越接近1，越相关#也是最大最小值  
'''**import cv2  
  
def showPicture(name,picture):  
 *#图像的显示,也可以显示多窗口* cv2.imshow(name,picture)  
 *#在键盘中按任意键退出显示并向后执行语句  
 #cv2.waitKey(1000)表示只显示1秒* cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
img=cv2.imread(**"image/C2\_1.jpg"**,0)*#转为灰度图*imgOriginal=cv2.imread(**"image/C2\_1.jpg"**)  
template=cv2.imread(**"image/CM\_1.jpg"**,0)  
h,w=template.shape[:2]  
  
showPicture(**"template"**,template)  
  
res=cv2.matchTemplate(img,template,cv2.TM\_CCOEFF\_NORMED)  
  
min\_val,max\_val,min\_loc,max\_loc=cv2.minMaxLoc(res)*#定位左上角坐标，宽高为templte*img1=cv2.rectangle(imgOriginal,max\_loc,(max\_loc[0]+w,max\_loc[1]+h),(0,255,0),2)  
  
showPicture(**"matchResult"**,img1)





*'''  
2021.1.30FromIvicxDS:openCV;E5图像形态处理：模板匹配  
'''*import cv2  
import numpy  
  
def showPicture(name,picture):  
 *#图像的显示,也可以显示多窗口* cv2.imshow(name,picture)  
 *#在键盘中按任意键退出显示并向后执行语句  
 #cv2.waitKey(1000)表示只显示1秒* cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
imgOriginal=cv2.imread(**"image/D1.jpg"**)  
imgOriginal=cv2.pyrDown(cv2.pyrDown(imgOriginal))  
img=cv2.imread(**"image/D1.jpg"**,0)*#转为灰度图*img=cv2.pyrDown(cv2.pyrDown(img))  
template=cv2.imread(**"image/DM.jpg"**,0)  
template=cv2.pyrDown(cv2.pyrDown(template))  
h,w=template.shape[:2]  
  
showPicture(**"template"**,template)  
  
res=cv2.matchTemplate(img,template,cv2.TM\_CCOEFF\_NORMED)  
threshold=0.81  
loc=numpy.where(res>=threshold)  
for i in range(len(loc[1])):  
 print(loc[0][i],loc[1][i])  
for pt in zip(\*loc[::-1]):  
 bottom\_right=(pt[0]+w,pt[1]+h)  
 cv2.rectangle(imgOriginal,pt,bottom\_right,(0,255,0),1)  
showPicture(**"matchResult"**,imgOriginal)