第 6 讲　可视化图像检测

绘制图形（直线、圆矩形、椭圆、多边形、文本）、目标检测框

6.1 绘制图形

6.1.1 绘制直线

cv2.line(img, pt1, pt2, color[, thickness[, lineType]]) → img

img，背景图
pt1，直线起点坐标
pt2，直线终点坐标
color，当前绘画的颜色。如在 BGR 模式下，传递(255,0,0)表示蓝色画笔。灰度图下，只需要传递亮度值即可。
thickness，画笔的粗细，线宽。若是-1 表示画封闭图像，如填充的圆。默认值是 1.
lineType，线条的类型，如 8-connected 类型、anti-aliased 线条（反锯齿），默认情况下是 8，cv2.LINE_AA 表示反锯齿线条，在曲线的时候视觉效果更佳。

```
#绘制直线，蓝颜色，粗细 5
import numpy as np
import cv2

img=np.zeros((512,512,3),np.uint8) # 返回来一个给定形状和类型的用 0 填充的数组
cv2.line(img,(0,0),(511,511),(255,0,0),5)
cv2.imshow('line',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.1.2 绘制圆形

cv2.circle(img, center, radius, color[, thickness[, lineType]])
作用：根据给定的圆心和半径等画圆
参数说明
img：输入的图片 data
center：圆心位置
radius：圆的半径
color：圆的颜色
thickness：默认是 1，圆形轮廓的粗细（如果为正）。负厚度表示要绘制实心圆。
lineType： 圆边界的类型。

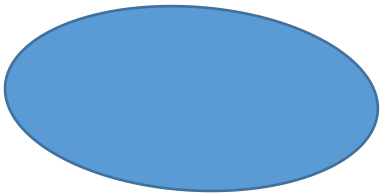
```
#绘制红色实心圆
img=np.zeros((512,512,3),np.uint8) # 返回来一个给定形状和类型的用 0 填充的数组
cv2.circle(img,(256,256),60,(0,0,255),-1)
cv2.imshow('circle',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.1.3 绘制矩形

cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType]]) → img

img，背景图
pt1，左上角坐标
pt2，右下角坐标
color，当前绘画的颜色。如在 BGR 模式下，传递(255,0,0)表示蓝色画笔。灰度图下，只需要传递亮度值即可。
thickness，画笔的粗细，线宽。若是-1 表示画封闭图像，默认值是 1.
lineType，线条的类型，如 8-connected 类型、anti-aliased 线条（反锯齿），默认情况下是 8，cv2.LINE_AA 表示反锯齿线条，在曲线的时候视觉效果更佳。

```
#绘制图像右上角绘制绿色矩形
img=np.zeros((512,512,3),np.uint8) # 返回来一个给定形状和类型的用 0 填充的数组
cv2.rectangle(img,(384,0),(512,128),(0,255,0),3)
cv2.imshow('rectangle',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



6.1.4 绘制椭圆

cv2.ellipse(image, center, axesLength, angle, startAngle, endAngle, color [, thickness [, lineType]]])
参数：
image：要在其上绘制椭圆的图像。

centerCoordinates：椭圆的中心坐标。坐标表示为两个值的元组，即(X 坐标值, Y 坐标值)。

axesLength：它包含两个变量的元组，分别包含椭圆的长轴和短轴(长轴长度，短轴长度)。
angle：椭圆旋转角度，以度为单位。
startAngle：椭圆弧的起始角度，以度为单位。
endAngle：椭圆弧的终止角度，以度为单位。
color：是要绘制的形状的边界线的颜色。
对于 BGR，我们传递一个元组。例如：(255, 0, 0)为蓝色。
粗细：是形状边界线的粗细(以像素为单位)。 -1 px 的厚度将用指定的颜色填充形状。
lineType：这是一个可选参数，它给出了椭圆边界的类型。

shift：这是一个可选参数。它表示中心坐标和轴值中的小数位数。
返回值：返回图像。

```
#绘制蓝色的椭圆
img=np.zeros((512,512,3),np.uint8) # 返回来一个给定形状和类型的用 0 填充的数组
cv2.ellipse(img,(256,256),(100,50),0,45,300,(255,0,0),-1)
cv2.imshow('ellipse',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.1.5 绘制多边形

cv2.polylines(img, pts, isClosed, color, thickness, lineType)

参数：
img（array）：为 ndarray 类型，原始图像
pts（array）：为所画多边形的顶点坐标，举个简单的例子：当一张图片需要有多个四边形时，该数组 ndarray 的 shape 应该为（N，4，2），三维数组
isClosed（bool）：所画四边形是否闭合，通常为 True
color（tuple）：RGB 三个通道的值
thickness（int）：画线的粗细

```
#绘制一个闭合多边形,
img=np.zeros((512,512,3),np.uint8) # 返回来一个给定形状和类型的用 0 填充的数组
pts=np.array([[100,5],[500,100],[512,200],[200,300]],np.int32)
print(pts)
pts.reshape((-1,1,2)) # 等于-1，那么 Numpy 会根据剩下的维度计算出数组的另外一个 shape 属性值
cv2.polylines(img,[pts],True,(255,0,0),3)
cv2.imshow('polylines',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.2 绘制文本

cv2.putText(img,text, pt1, font, size,color, thinkness)
各参数依次是：图片，添加的文字，左上角坐标，字体，字体大小，颜色，字体粗细。

其中字体可以选择

cv2.FONT_HERSHEY_SIMPLEX
cv2.FONT_HERSHEY_PLAIN
cv2.FONT_HERSHEY_DUPLEX
cv2.FONT_HERSHEY_COMPLEX

```
#绘制文字
img=np.zeros((512,512,3),np.uint8) # 返回来一个给定形状和类型的用 0 填充的数组
font=cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'openCV',(10,500),font,4,(255,255,255),2)
cv2.imshow('putText',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.3 提取感兴趣的区域

1.感兴趣区域（Region of Interest， ROI）


```
img=cv2.imread('face.jpg')
face=img[200:400,200:400]
gray_face=cv2.cvtColor(face,cv2.COLOR_BGR2GRAY)
back_face=cv2.cvtColor(gray_face,cv2.COLOR_GRAY2BGR)
img[200:400,200:400]=back_face
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.用 selectROI（）还书通过交互的方式选取 ROI 区域。

selectROI(windowName, img, showCrosshair=None, fromCenter=None):

- 参数 windowName：选择的区域被显示在的窗口的名字
- 参数 img：要在什么图片上选择 ROI
- 参数 showCrosshair：是否在矩形框里画十字线.
- 参数 fromCenter：是否是从矩形框的中心开始画

```
import numpy as np
import cv2
img=cv2.imread('face.jpg')
r = cv2.selectROI(img)#返回的是一个元组[min_x,min_y,w,h]：
```

```
print(r)
roi=img[int(r[1]):int(r[1])+int(r[3]),int(r[0]):int(r[0])+int(r[2])]
cv2.imshow('ROI',roi)
cv2.imwrite('roi.jpg',roi)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.4 图像的按位运算

6.4.1 按位与运算

对图像进行按位与运算的函数是 cv2. bitwise_and(),其规则如下：

1&1=1，1&0=0，0&1=0，0&0=0

函数	说明
cv2. bitwise_and(	对图像进行按位与运算。
src1,	**src1**：图像 1
src2,	**src2**：图像 2
dst=None,	**dst**：默认选项
mask=None)	**mask**：图像掩模

对一张圆形和方形，完成按位与运算

```
import numpy as np
import cv2
rectangle = np.zeros((300,300),dtype="uint8")
cv2.rectangle(rectangle,(25,25),(275,275),255,-1)
cv2.imshow("Rectangle",rectangle)
 circle = np.zeros((300,300),dtype="uint8")
cv2.circle(circle,(150,150),150,255,-1)
cv2.imshow("Circle",circle)
bitwiseAnd = cv2.bitwise_and(rectangle,circle)
cv2.imshow("And",bitwiseAnd)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.4.2 按位或运算

对图像进行按位或运算的函数是 cv2. bitwise_or(),其规则如下：

1|1=1，1|0=1，0|1=1，0|0=0

函数	说明
cv2. bitwise_or(	对图像进行按位或运算。
src1,	**src1**：图像 1
src2,	**src2**：图像 2
dst=None,	**dst**：默认选项
mask=None)	**mask**：图像掩模

对一张圆形和方形图像，完成按位或运算。

```
bitwiseOr = cv2.bitwise_or(rectangle,circle)
cv2.imshow("OR",bitwiseOr)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.4.3 按位非运算

对图像进行按位非运算的函数是 cv2. bitwise_not(),其规则如下：~0=1，~1=0

函数	说明
cv2. bitwise_not(	对图像进行按位非运算。
src1,	**src1**：图像 1
dst=None,	**dst**：默认选项
mask=None)	**mask**：图像掩模

对圆形完成非运算，示例如下

```
bitwiseNot = cv2.bitwise_not(circle)
cv2.imshow("Not",bitwiseNot)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.4.4 异或运算

可以使用函数 cv2.bitwise_xor 来实现按位异或运算，其规则如下：1 ^ 0 =1，1 ^ 1 =0，0 ^ 1 =1，0 ^ 0 =0

函数	说明
cv2. bitwise_xor(	对图像进行按位与运算。
src1,	**src1**：图像 1
src2,	**src2**：图像 2
dst=None,	**dst**：默认选项
mask=None)	**mask**：图像掩模

对一张圆形和方形，完成按位异或运算。

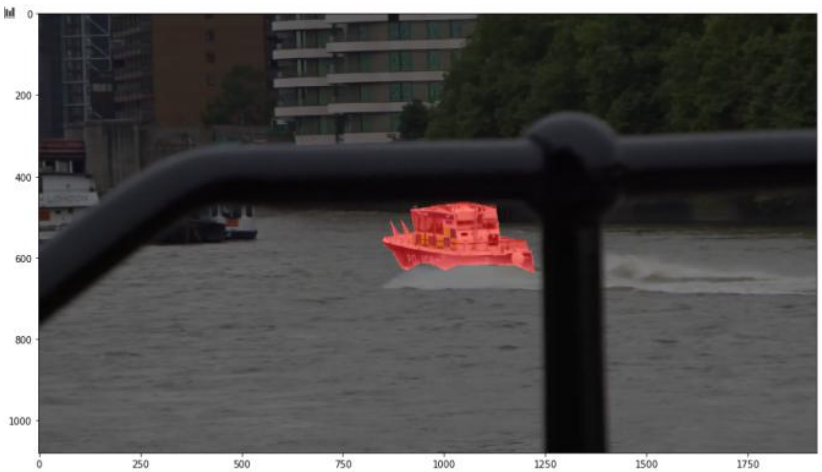
```
bitwiseXor = cv2.bitwise_xor(rectangle,circle)
cv2.imshow("XOR",bitwiseXor)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.4.5 加密和解密

对图像进行异或操作，完成加密和解密的工作。对图像进行按位运算，并输出运算结果。示例代码如下：

```
import cv2
import numpy as np
image = cv2.imread("cat.jpg",0)
w,h= image.shape
key = np.random.randint(0,256,size=[w,h],dtype=np.uint8)
encryption = cv2.bitwise_xor(image,key)
decryption = cv2.bitwise_xor(encryption,key)
cv2.imshow("encryption",encryption)
cv2.imshow("decryption",decryption)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.5 实验：绘制图像掩模



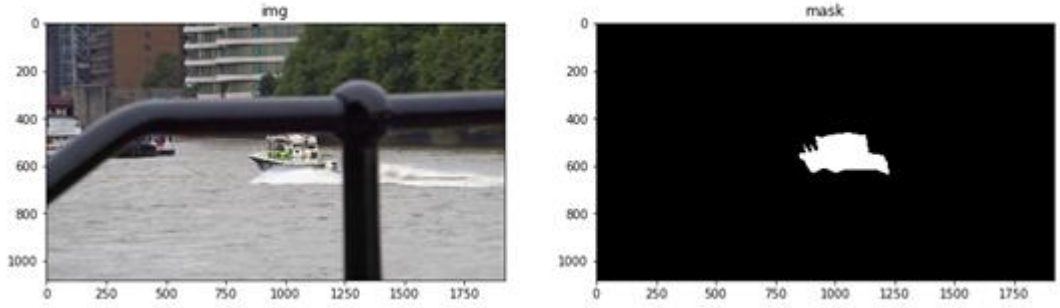
6.5.1 读取掩模图片和原图

引入相关的包，会用到 opencv 库，numpy 库，Matplotlib 库。

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

使用 numpy 包，和 opencv 包读取 npy 格式的掩模文件和原图。使用 np.load 函数可以提取 npy 文件。cv2.imread 函数读取 jpg 图片。

```
#读取 npy 掩模文件和原图
mask = np.load('./00000000.npy')
img = cv2.imread('./00000000.jpg')
通过 matplotlib 显示一下效果，注意原图需要转换成 RGB 颜色通道。
plt.figure(figsize=(15,15))
#显示图像
img_rgb = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.subplot(121),plt.imshow(img_rgb),plt.title("img")
plt.subplot(122),plt.imshow(mask, cmap="gray"),plt.title("mask")
```



6.5.2 准备红色掩模文件

要在原图上实现红色掩模效果，要准备一个红色的掩模文件。

首先，创建一个和原图同样大小的红色图片。通过 np.zeros 可以创建一个和原图一样的全 0 矩阵，注意数据类型是 uint8.

```
#创建和原图相同大小的红色图片
red = np.zeros(img.shape,np.uint8)
```

因为 BGR 图像通道的顺序，设置每个像素点的值为[0,0,255]，则把图像改成了红色。

```
red[:,:,2] = 255
```

通过 matplotlib 可以显示一下图像效果

```
#显示效果
red_rgb = cv2.cvtColor(red,cv2.COLOR_BGR2RGB)
plt.imshow(red_rgb)
```

查看 npy 读入的 mask，可以查看一下其形状，

```
print(mask.shape,mask.dtype)
print(img.shape)
print(mask.min(),mask.max())
```

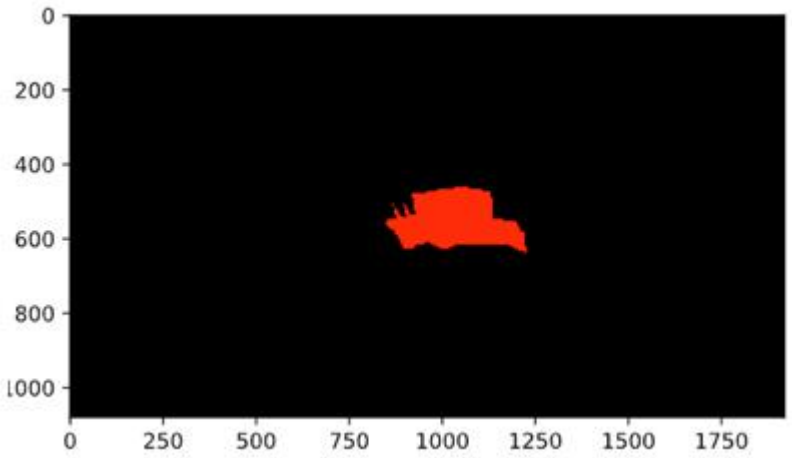
```
(1080, 1920) int32
(1080, 1920, 3)
```

0 1

可以看到 mask 是和原图一下大小，只包括 0 和 1 整型数的矩阵。这里 1 代表检测的目标，0 代表背景。

因此可以直接使用 mask 作为掩模，在红色图像中提取目标的部分。使用 cv2.bitwise_and 操作，保留 mask 中数值为 1 的像素点为红色，其他部分是黑色背景。并显示查看效果。

```
#修改 mask 的数据类型
mask = mask.astype(np.uint8)
#获取红色掩模
red_mask = cv2.bitwise_and(red,red,mask=mask)
red_mask_rgb = cv2.cvtColor(red_mask,cv2.COLOR_BGR2RGB)
plt.imshow(red_mask_rgb)
```



6.5.3 和原图合成

合成掩模图像和原图

```
masked = cv2.addWeighted(img,0.5,red_mask,0.5,0)
```

显示查看结果

```
plt.figure(figsize=(15,15))
result = cv2.cvtColor(masked,cv2.COLOR_BGR2RGB)
plt.imshow(result)
```

6.6 实验:绘制目标检测框

6.6.1 读取目标框文件

目标框文件是 txt 格式,可以先查看一下,其中一行表示一个检测目标,分为对应 label,x 坐标, y 坐标, 宽度和高度。中间用逗号分隔。首先读取文件。

```
f = open('./00000000.txt','r')
objs = f.readlines()
print(len(objs))
print(objs)
```

得到结果:['boat, 847,463,379,171']

可以看到当前文件只有一项,说明只有一个检测目标。

6.6.2 绘制框和文字

解析每一行的结果,可以通过 split 方法,把每一行转换成数组,获取标签名称,坐标点转换成 int 类型,宽高也需要转换成 int 类型。并在上一任务产生的图像上绘制目标矩形核标签文字。

```
#针对每一行
for obj in objs:
   print(obj)
     #用逗号分隔的每个结果
     items = obj.split(',')
     #第一项是 label
     label = items[0]
     #第二、三项是 x, y 坐标点
     x = int(items[1])
     y = int(items[2])
     #第四、五项是宽度和高度
     w = int(items[3])
     h = int(items[4])
     print(x,y,w,h)
```

得到 847 463 379 171

然后通过 opencv 的方法 rectangle 在图像对应的位置上画红色矩形框。

```
#绘制红色的目标框
cv2.rectangle(masked, (x,y),(x+w, y+h),(0,0,255),3)
```

通过 opencv 的 putText 方法, 在矩形框的上方写上标签名称。

```
#在左上角绘制标签
cv2.putText(masked,label,(x-5,y-5),cv2.FONT_HERSHEY_PLAIN,3,(0,0,255),3)
```

通过 matplotlib 显示一下最终结果

```
plt.figure(figsize=(15,15))
```

```
result2 = cv2.cvtColor(masked,cv2.COLOR_BGR2RGB)
plt.imshow(result2)
```

6.2.3 保存结果

把绘制的结果通过 cv2.imwrite 保存成文件。

```
cv2.imwrite("result.png",masked)
```

可以看到产生了 result.png 的图像文件，是包括的绘制结果的图像。

作业：

1.在图片 face.jpg 上画一个 200*300（脸的位置）的矩形框，在框左上角标记文字“脸”，红色。

2.完成实验 6.5,、6.6。