实验 1 绘制图像掩模

实验时长：4 小时

实验难度：一般

实验摘要：首先读取 npy 的掩模图像，进行处理后和原图合成，完成在原图上增加红色掩模的可视化效果。

实验建议：了解目标检测基本概念。

实验目标：能够使用 opencv 绘制图像掩模。

实验 1 绘制图像掩模

实验时长：4 小时

实验难度：一般

实验摘要：首先读取 npy 的掩模图像，进行处理后和原图合成，完成在原图上增加红色掩模的可视化效果。

实验建议：了解目标检测基本概念。

实验目标：能够使用 opencv 绘制图像掩模。

1、绘制图像掩模

-

1.1、读取掩模图片和原图

1.2、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
```

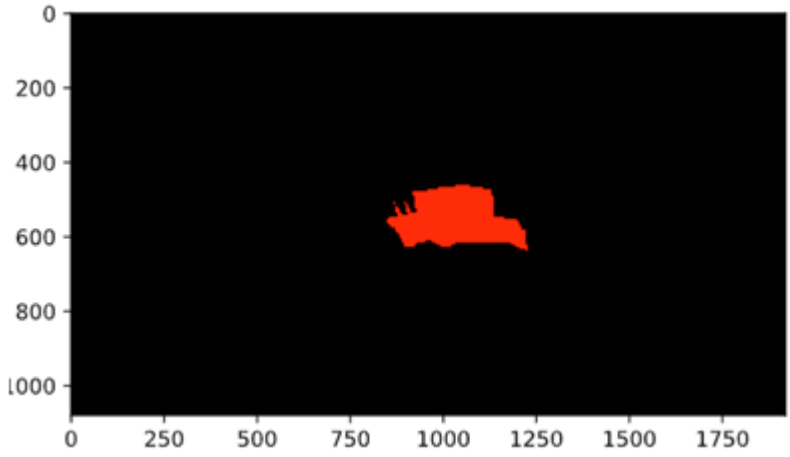
```
mask = mask.astype(np.uint8)
```

```
#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)
```



下一步

-

- 1.3、和原图合成

- 合成掩模图像和原图

-

-

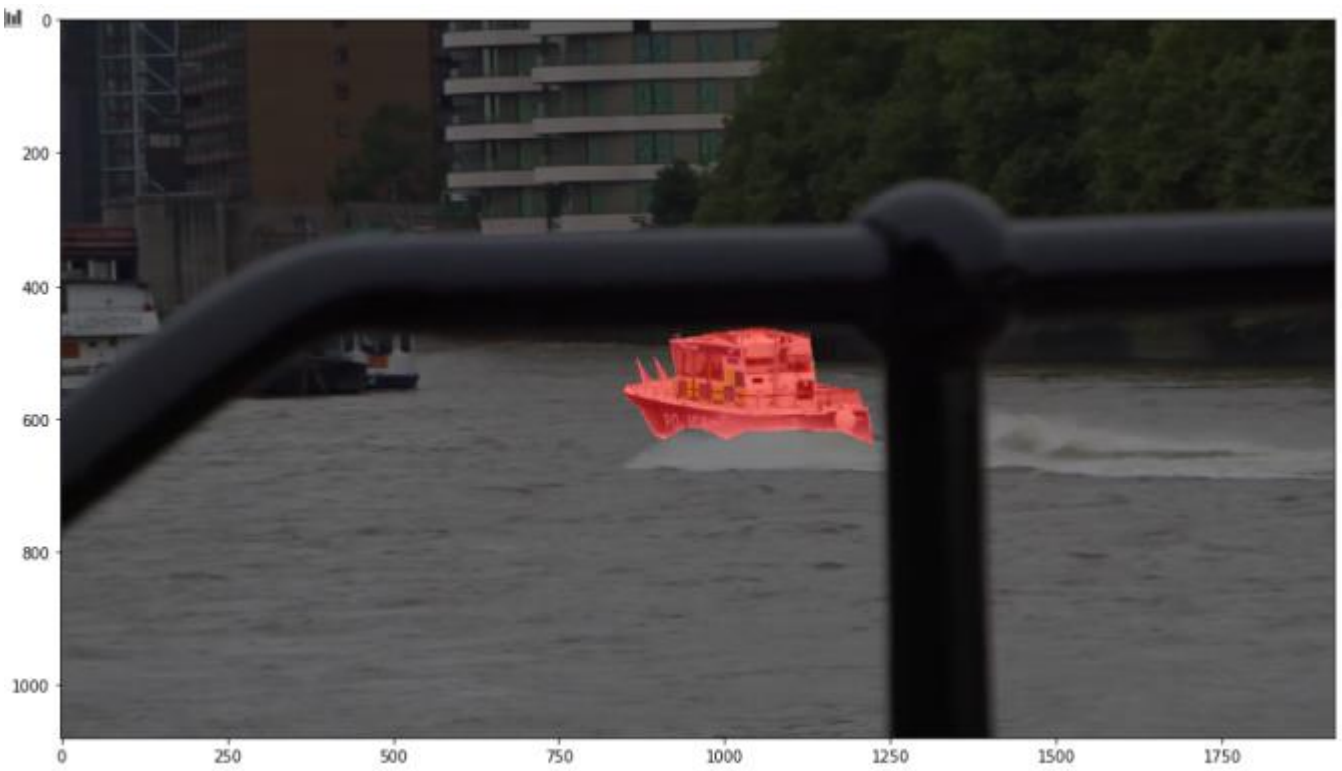
```
masked = cv2.addWeighted(img,0.5,red_mask,0.5,0)
```

- 显示查看结果

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

- `result = cv2.cvtColor(masked,cv2.COLOR_BGR2RGB)`

- `plt.imshow(result)`

- 

-
-

-

下一步