**实验报告**

专业：计算机科学与技术

姓名：贺情怡

学号：3180105438

日期：2020/12/18

课程名称： 计算机视觉 指导老师： 宋明黎 成绩：

实验名称： HW#2:椭圆拟合

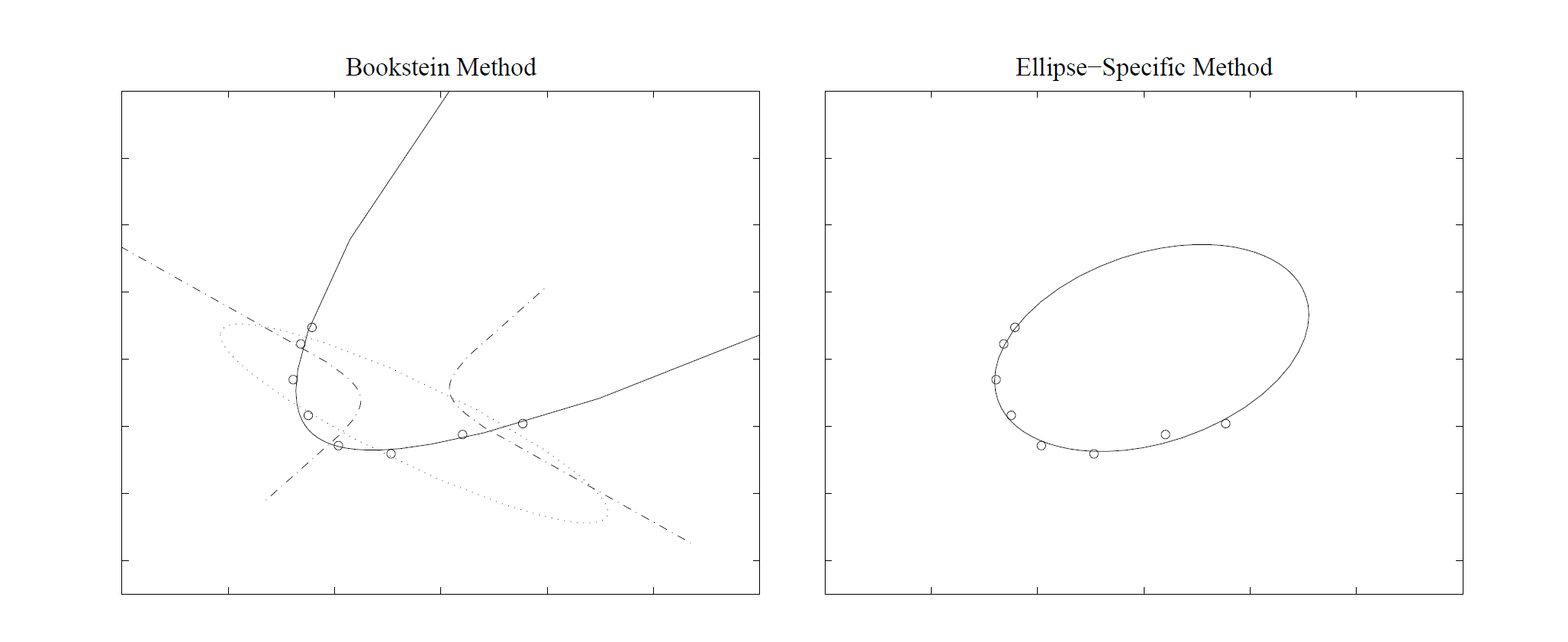
1. **实验目的和要求**

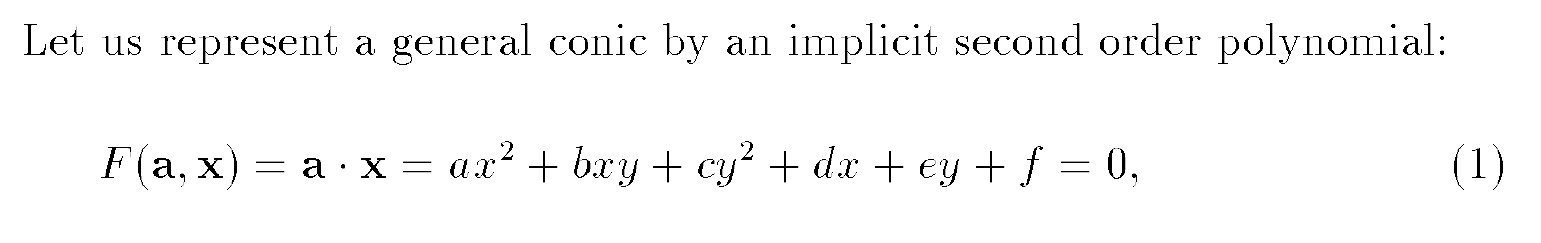
尝试使用cv.fitEllipse()函数，对图像进行椭圆拟合

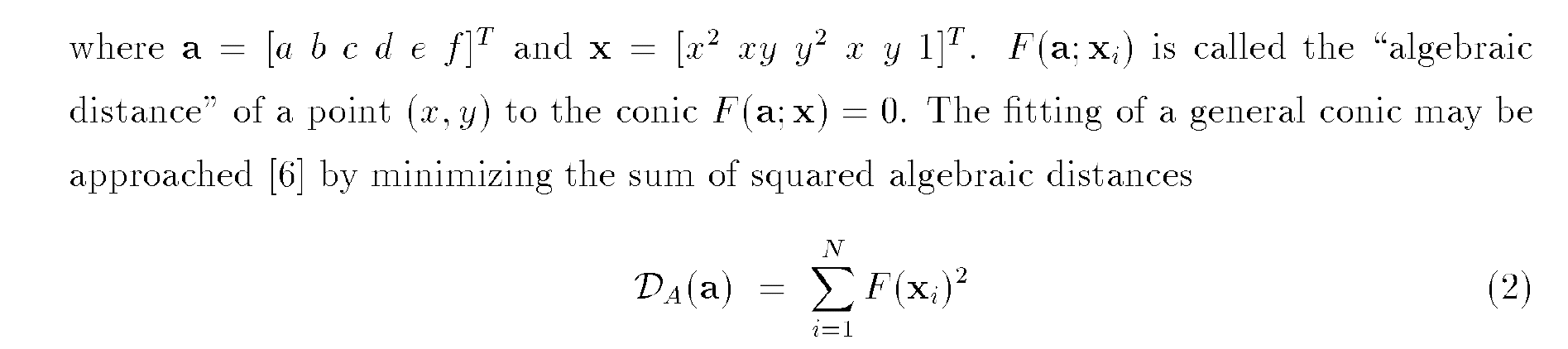
**二、实验内容和原理**

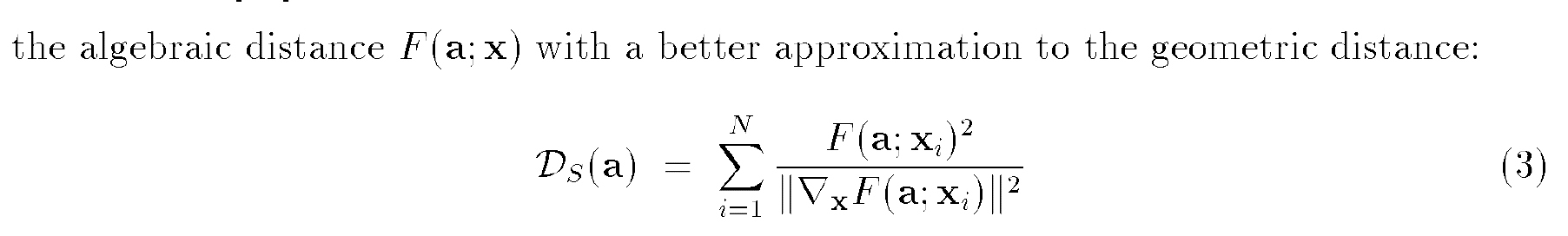
**椭圆拟合**

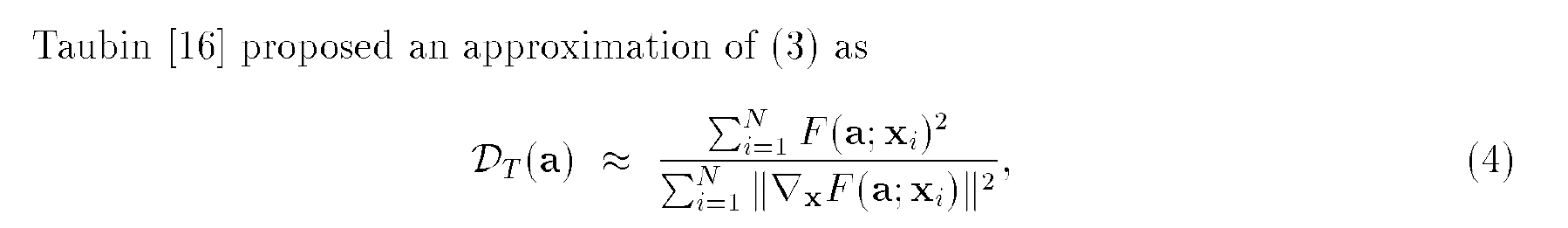
该函数使用的是最小二乘法拟合，要求输入的点至少有6个。

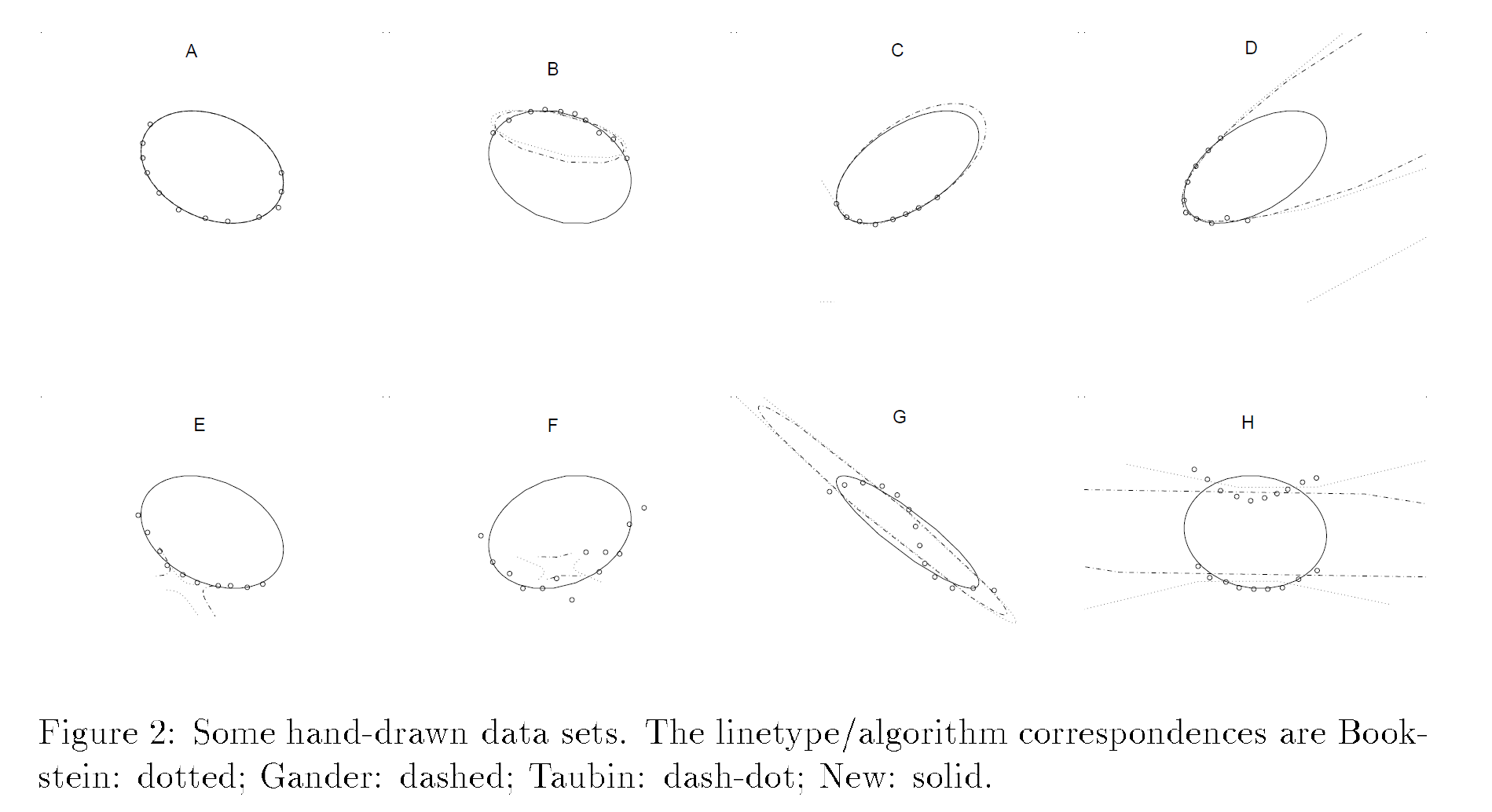




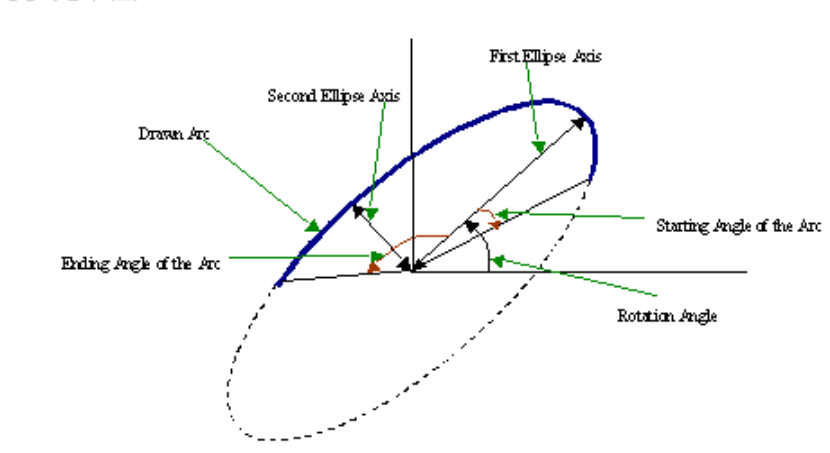








函数中对应的参数如下：

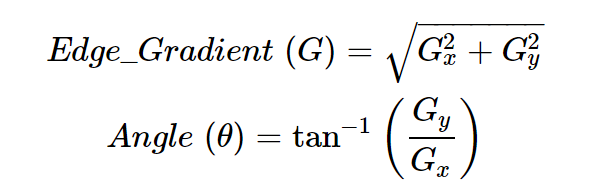


**对输入图像的预处理**

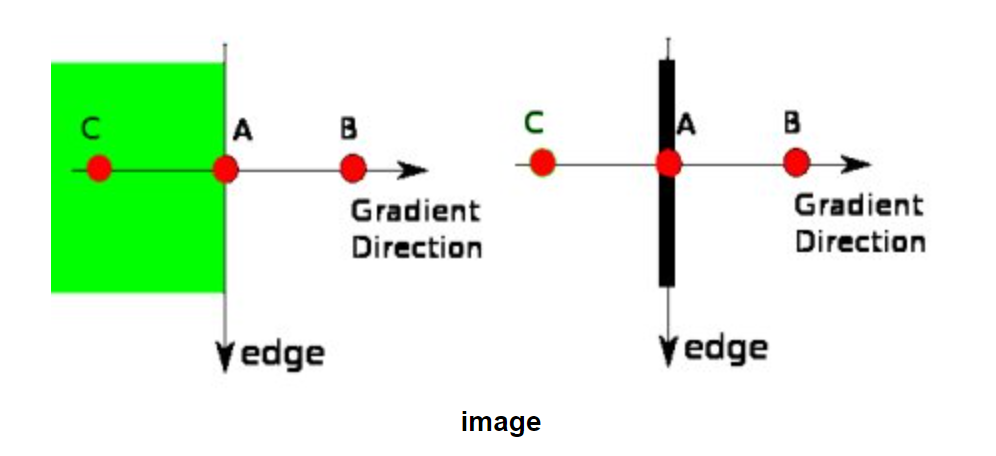
输入一张RGB图片，先转换为灰度图，本来打算先转换为二值图像再进行边缘检测的，但是发现二值化容易使阴影成为新的边缘，并丢失原有边缘信息，于是直接对灰度图进行了边缘检测。在边缘检测前还进行了降噪

**边缘检测**

用Sobel Kernel 进行滤波，来计算x和y方向的导数。



对于每个像素，检测它是否为局部最大值，如果不是就设置为0



输入的两个参数为min和max，用来限定寻找的边缘范围（大于max的肯定是边缘，小于min的肯定不是，在min和max之间的则通过判断连通性等方式来判断。

**三、实验步骤与分析**

**读入图片**

读入后先去掉高频的信息（降噪）。然后保存一个RGB图像信息，存储一个gray灰度图信息。边缘信息edge直接从gray中提取，min取到100，max取到200。

|  |
| --- |
| FILENAME = "dota2items4.bmp"  img = cv.imread(FILENAME)  img = cv.GaussianBlur(img, (5, 5), 0)  rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)  # \_, thresh = cv.threshold(gray, 130, 255, cv.THRESH\_BINARY + cv.THRESH\_OTSU)  edge = cv.Canny(gray, 100, 200) |

**使用fitEllipse()函数**

Ells数组储存椭圆信息，recs储存可以旋转的最小矩阵，recs2储存正矩形信息。Cols用于储存随机生成的颜色信息，方便区分不同椭圆和矩阵之间的对应关系

|  |
| --- |
| img = rgb  edge\_ = cv.cvtColor(edge, cv.COLOR\_GRAY2RGB)  ells = []  recs = []  recs2 = []  cols = []  contours, nothing = cv.findContours(edge, cv.RETR\_TREE, cv.CHAIN\_APPROX\_SIMPLE)  for index, contour in enumerate(contours):  if contour.shape[0] > 50:  ells.append(cv.fitEllipse(contour))  recs2.append(cv.boundingRect(contour))  recs.append(cv.minAreaRect(contour))  random.seed(10)  for i in range(len(ells)):  cols.append([random.randint(0, 256) for j in range(3)]) |

**绘制椭圆与矩形**

同时在RGB图像和边缘图像中绘制，这是椭圆

|  |
| --- |
| for index, ell in enumerate(ells):  cv.ellipse(img, ell, cols[index], 2)  cv.ellipse(edge\_, ell, cols[index], 2) |

这是可旋转的矩形

|  |
| --- |
| for index, rec in enumerate(recs):  box = cv.boxPoints(rec) # cv2.boxPoints(rect) for OpenCV 3.x  box = np.int0(box)  cv.drawContours(img,[box],0,cols[index],1)  cv.drawContours(edge\_,[box],0,cols[index],1) |

绘制正矩形

|  |
| --- |
| for index, rec2 in enumerate(recs2):  x,y,w,h = rec2  cv.rectangle(img,(x,y),(x+w,y+h),cols[index],1)  cv.rectangle(edge\_,(x,y),(x+w,y+h),cols[index],1) |

**四、实验结果**

**输入图像**

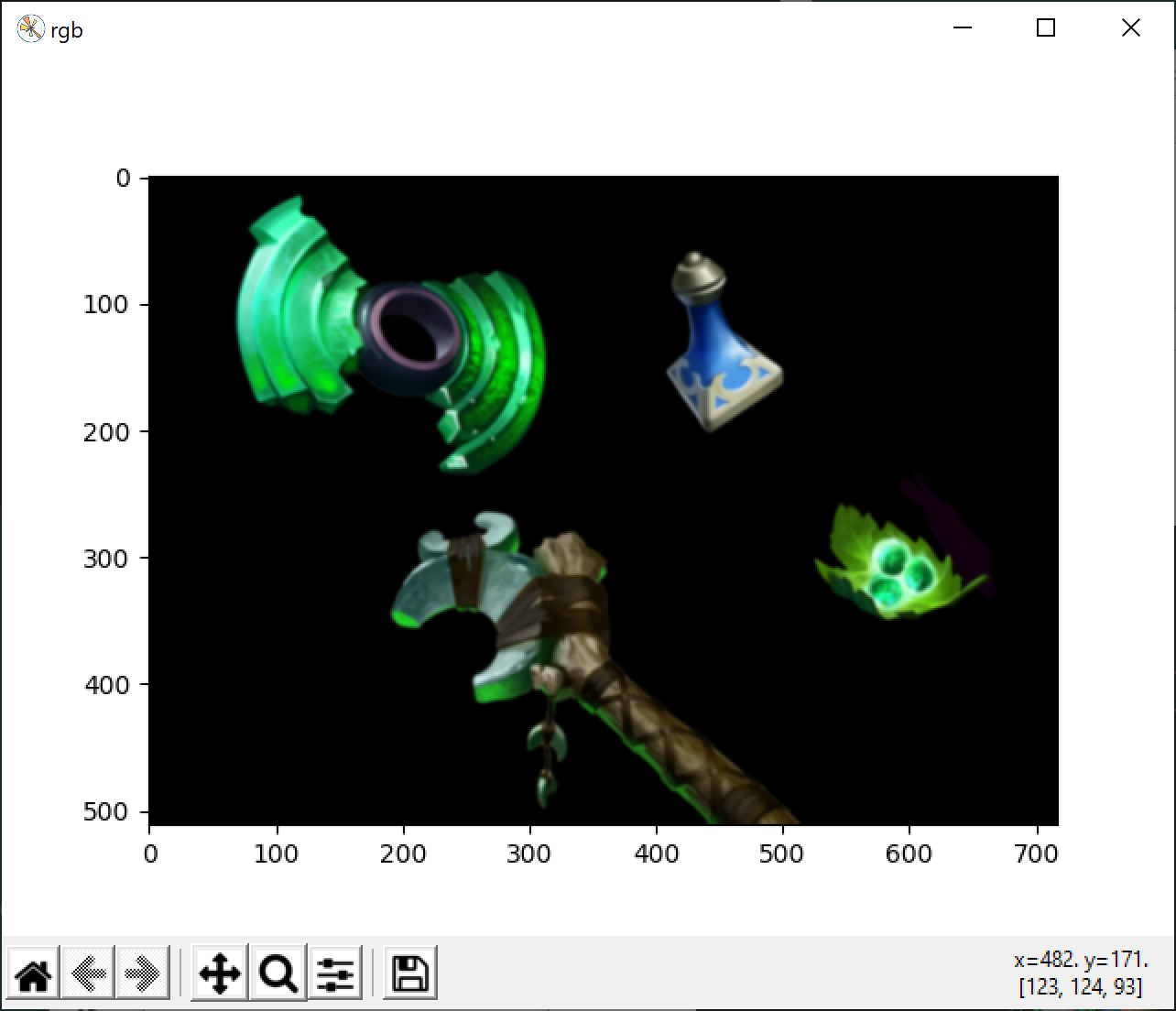
图里左上角的Hyper stone具有很多同心圆信息，右上的Clarity有不规则曲线和直线信息，右下的Tango有圆形信息且整体外形近似椭圆，正下方的Force Staff有近似的圆形信息和直线。

****

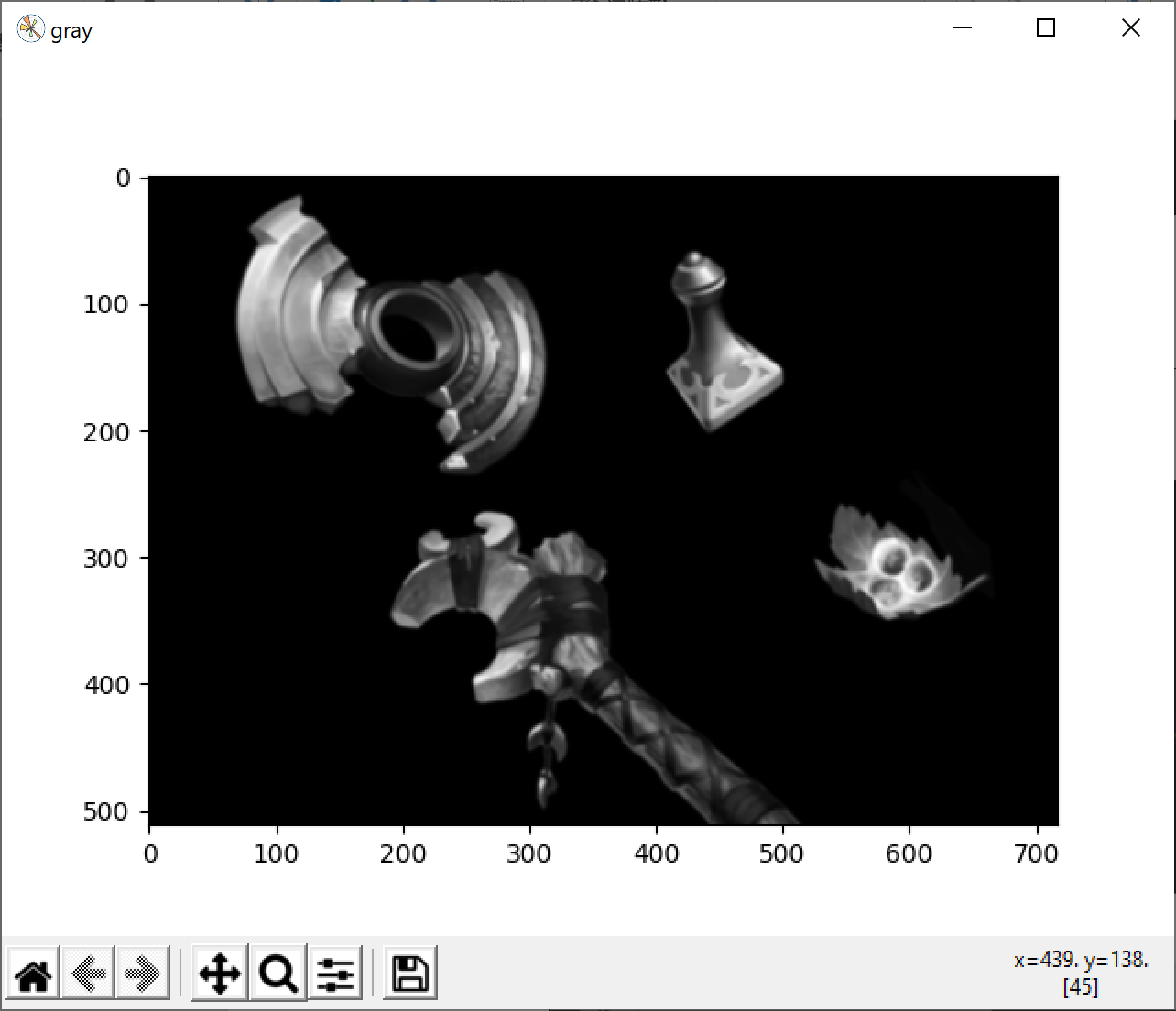
**输出：**

1. RGB

图像因为进行过高斯滤波而稍微模糊

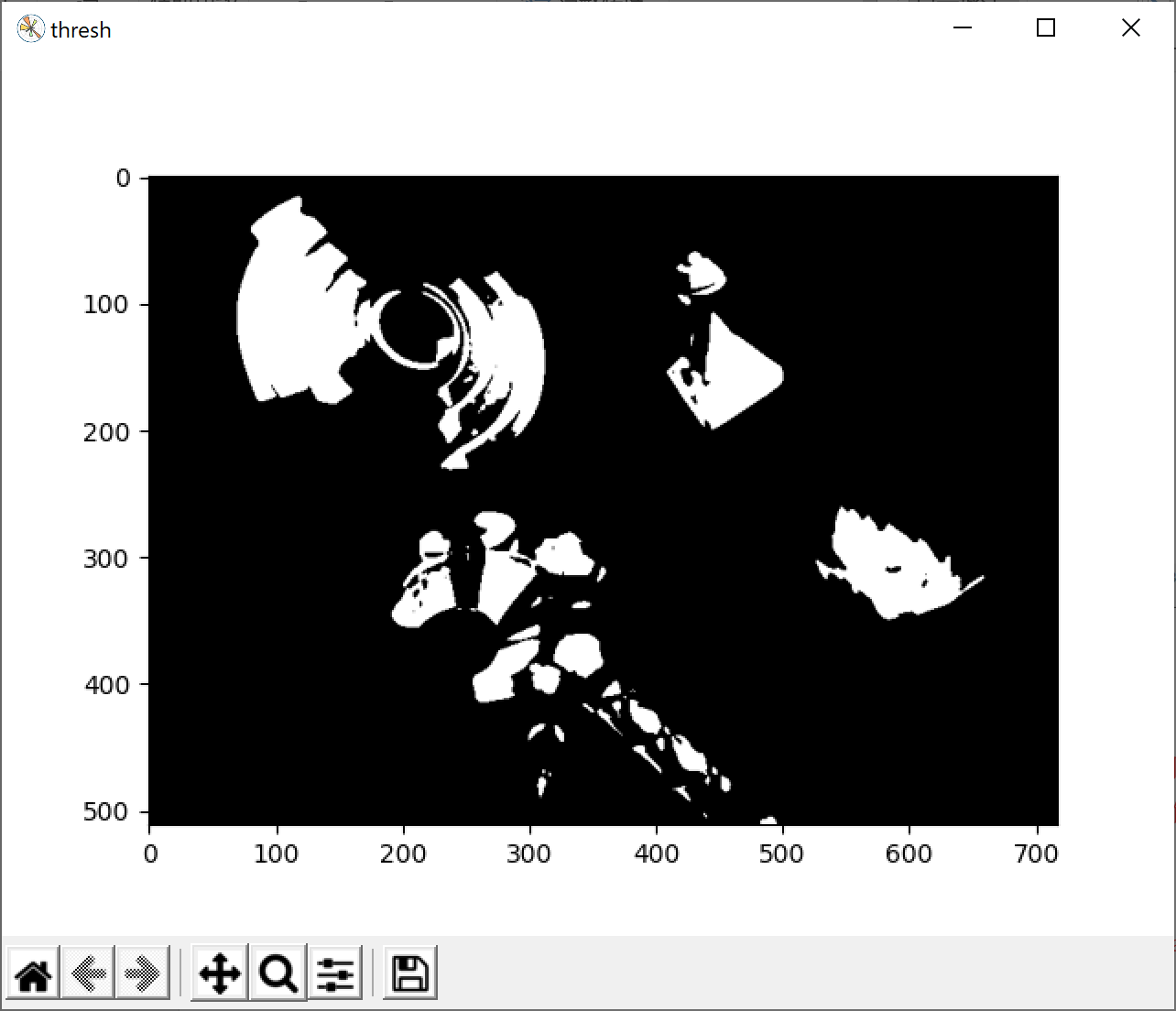


1. **GrayScale**



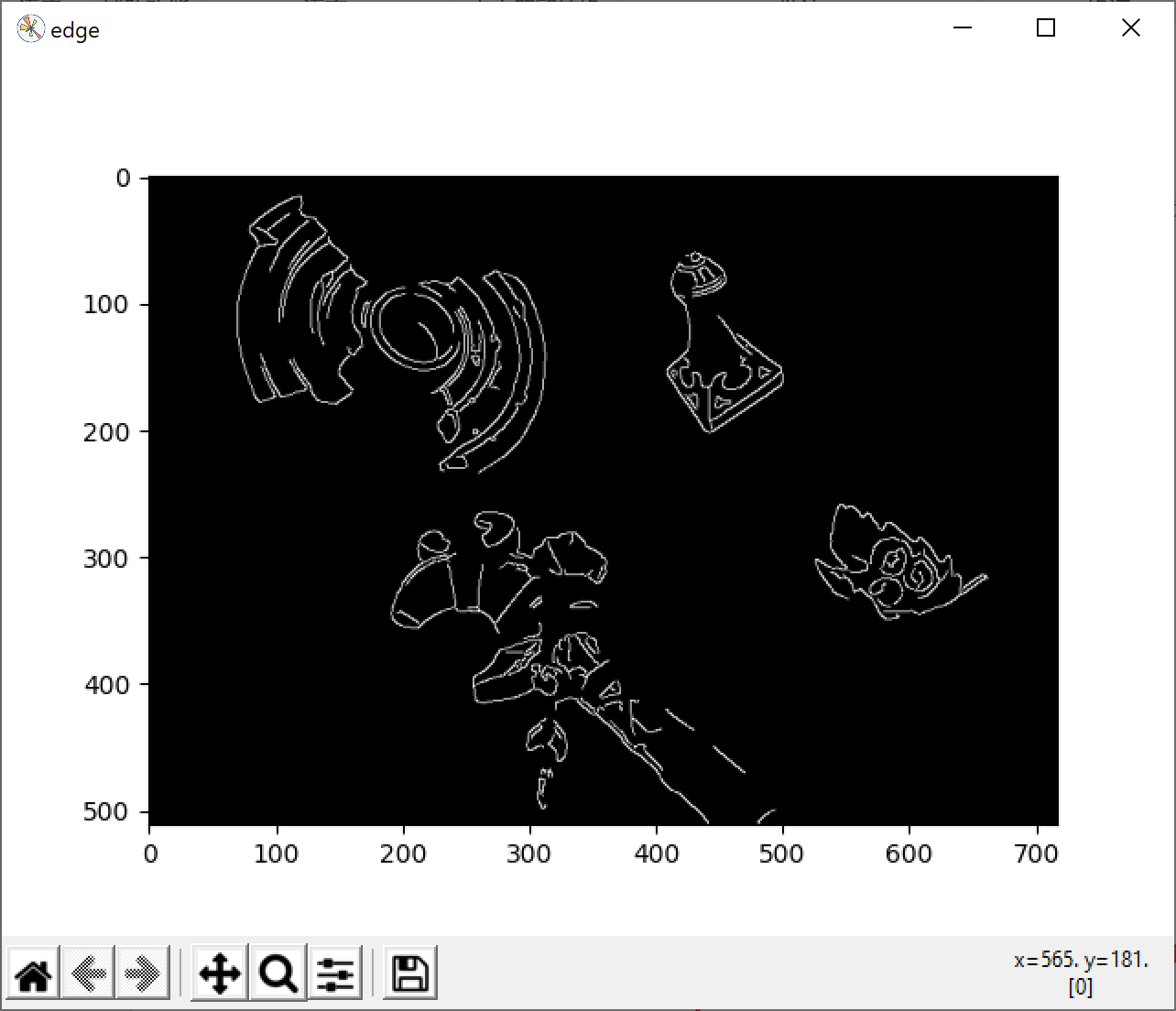
1. **Thresh**

可以看出，二值化后的图像失去了很多细节信息，同时引入了阴影的边缘信息，这是我们不太想要的



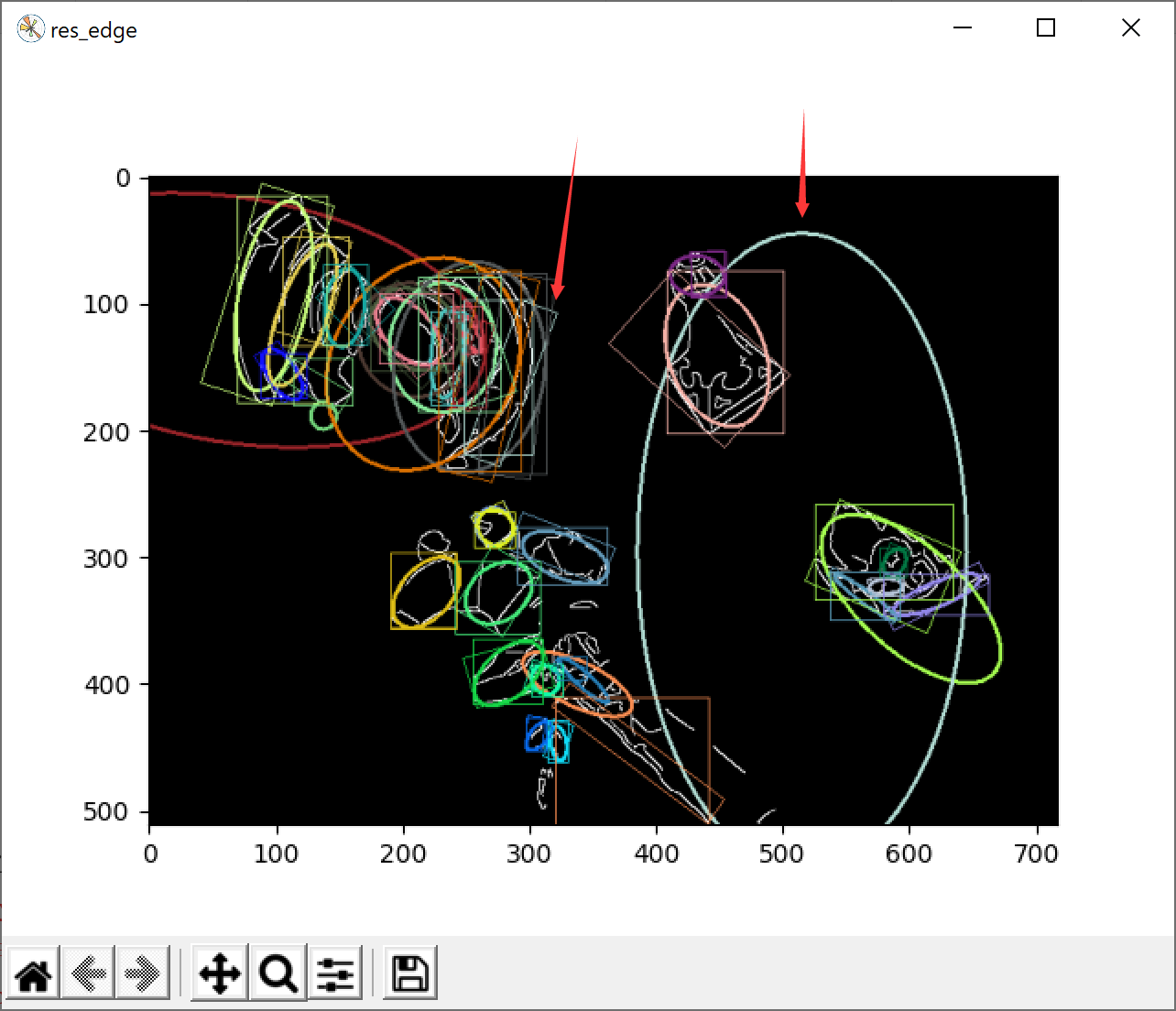
1. **Edge**

根据灰度图生成了图像的边缘信息

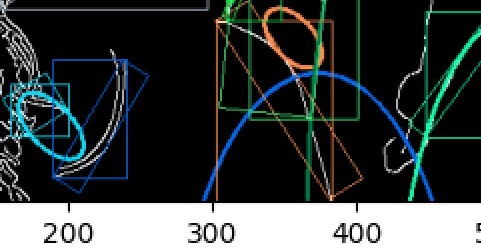


1. **Ellipse on Edge**

大部分边缘信息拟合良好，但是图中红色箭头标出的浅蓝色椭圆很奇怪，通过分析，它对应的contour是左边浅蓝色矩形内部的信息，并不知道为什么会发生这样的事，也许是bug？（这个现象同样在Lena图片中出现，根据搜索到的信息，可能是最小二乘法造成的问题）

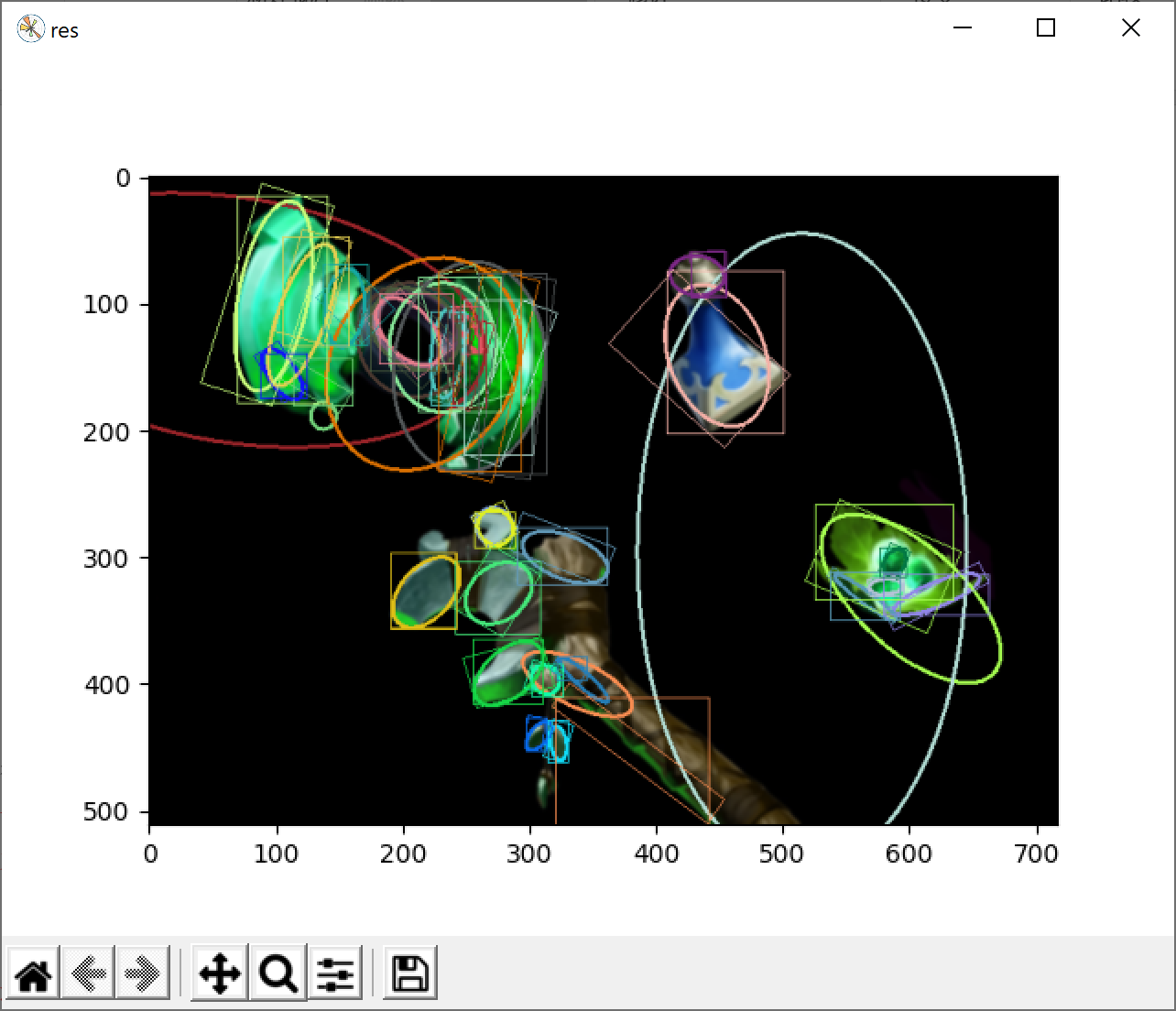


下图为Lena照片下半部分，看得出左边深蓝色矩阵和右边的椭圆对应，但并没有任何重合。

****

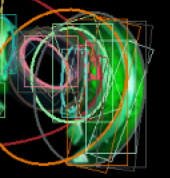
1. **Ellipse on RGB**

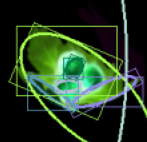
看得出，那个浅蓝色椭圆与整个图片格格不入

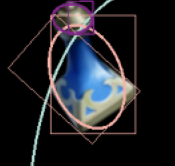


**五、心得体会**

本来想直接用那种波点图像就算了，看见同学做了对奇怪的游戏CG的拟合，于是考虑到继续使用Lena姐姐的照片，但是羽毛帽的细节信息太多，导致效果并不十分好。于是找了DOTA2的物品图标（虽然效果很好，但是分辨率实在太低，于是找了二次创作的高清图像，效果还不错，各种情况下的效果都能看见）

比如这里非常标准的椭圆

 这里对形状近似的物体的拟合

这个粉色应该是在拟合瓶子的左下边缘。

通过调大拟合时使用的点的数量阈值以及画的线的粗细，可以删除一些不靠谱的结果，防止出现一大团乱七八糟的线 的效果（如下）

（对应的其实是）