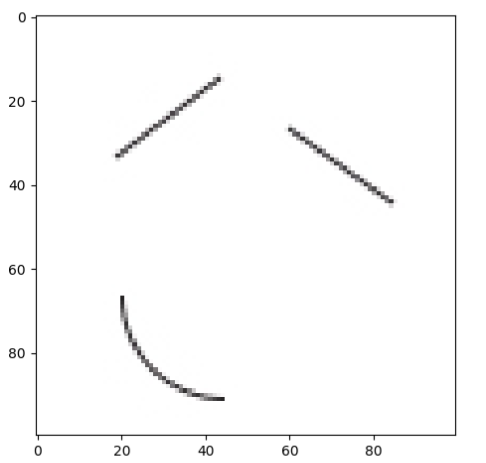
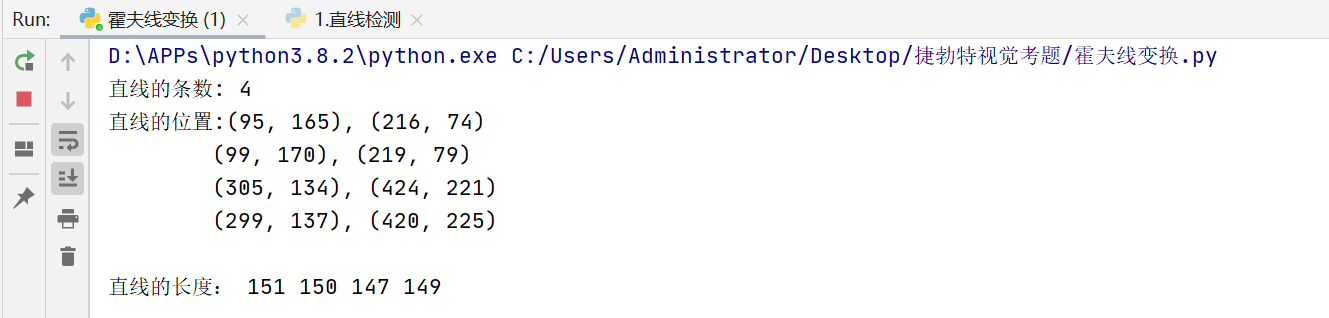
第三题:有1张图片，白色背景，上面有几条黑色的直线段和弯曲的线段。 现希望能找出这些直线段条数、 位置及长度。写一个小程序，实现此功能。不能使用OpenCV，Halcon等视觉库，可以使用一些科 学计算库； 提交方案简单说明，代码、测试结果（6′）。

|  |
| --- |
| 编程语言 python 环境numpy， matplotlib |
| image = matplotlib.Image.open(“img\_address.jpg”)  # 读取图片数据，得到一个numpy.ndarray数组 |
| Img\_gray\_image = image .convert(‘L’)  # 转换为灰度图 |
| threshold = 127  table = np.array([0 if pixel < threshold else 255 for pixel in range(256)])  binary\_image = gray\_image.point(table, '1')  # 设置阈值127，进行二值化，黑色像素的坐标信息储存在table中 |
| # 创建一个方法，输入点的坐标列表，输出直线的列表  def get\_line\_list(table):  k\_dist\_list 霍夫空间矩阵  for i in range(0, N):  p1 = point\_list[i]  for j in range(i + 1, N):  p2 = point\_list[j]  dist, k, b = get\_line\_coef\_length(p1, p2)  # 计算两个p1, p2所在直线的斜率k和截距b  if np.abs(b) <= 90:  k\_b\_matrix[k, int(b + 90)] += 1  # 遇到斜率k，截距为int(b+90),在矩阵相应位置加1  if dist > k\_dist\_list[k]:  k\_dist\_list[k] = dist |
| # 创建一个方法，输入两个点的坐标，输出两个点的斜率，角度，截距  def get\_line\_coef\_length(p1, p2):  # 计算两个点的斜率  import numpy as np  x1, y1 = p1  x2, y2 = p2  xe = x1 - x2  ye = y1 - y2  dist = np.sqrt(xe \* xe + ye \* ye)  k = float(y2) - float(y1)  if x2 - x1 == 0:  # 当斜率无穷大时，返回180。  return dist, 180, y1  k = k / (float(x2) - float(x1))  k\_theta = int(np.arctan(k) / 3.14 \* 180.0 + 90.0)  # 角度的取值范围[-90,90]，k\_b\_matrix下标从0开始，[-90,90]映射到了[0,180]  b = y1 - k \* x1  # 线段长度，角度  return dist, k\_theta, b |
| # 打印直线的条数，位置， 长度等  def dist(x1, y1, x2, y2):  mm = (x1 - x2) \*\* 2 + (y1 - y2) \*\* 2  return int(mm \*\* 0.5) print("直线的条数:", len(lines)) print("直线的位置", end=":") length = [] # 绘制直线 for line in lines:  x1, y1, x2, y2 = line[0]  print("(%d, %d), (%d, %d)" % (x1, y1, x2, y2), end="\n\t\t")  length.append(dist(x1, y1, x2, y2))  cv2.line(image, (x1, y1), (x2, y2), (0, 255, 0), 2) print("直线的长度：", \*length) |

测试结果





第四题：有1张图片，白色背景，有一段手绘的圆弧，大约1/4个圆大小，半径未知，有较明显边界，但是会存在其他物体干扰。请拟合这段圆弧。不能使用OpenCV，Halcon等视觉库，可以使用一些科学计 算库；提交方案简单说明，代码、测试结果（6′）。

|  |
| --- |
| 编程语言 python 环境numpy， matplotlib |
| # 定义圆心、半径和圆弧角度 |
| center = (0, 0)  radius = 1.0  angle = np.pi/2 # 圆弧角度为90度  # 定义圆弧的边界点  theta = np.linspace(0, angle, 100)  x = center[0] + radius \* np.cos(theta)  y = center[1] + radius \* np.sin(theta)  # 创建一个圆弧  arc = plt.figure(figsize=(5, 5))  plt.axes().set\_aspect('equal')  plt.axes().plot(x, y, color='blue')  plt.axes().scatter(center[0], center[1], color='black') # 圆心  plt.axes().set\_xlim(-1.5, 1.5)  plt.axes().set\_ylim(-1.5, 1.5)  # 检测点是否在圆弧内  def is\_point\_in\_arc(point, center, radius, start\_angle, end\_angle):  angle\_to\_point = np.arctan2(point[1] - center[1], point[0] - center[0])  if angle\_to\_point < 0:  angle\_to\_point += 2 \* np.pi  if start\_angle > end\_angle:  return (angle\_to\_point > start\_angle) or (angle\_to\_point < end\_angle)  else:  return (angle\_to\_point > start\_angle) and (angle\_to\_point < end\_angle)  # 测试点  test\_point = (np.cos(np.pi/8), np.sin(np.pi/8))  plt.axes().scatter(test\_point[0], test\_point[1], color='red')  # 检测结果  inside\_arc = is\_point\_in\_arc(test\_point, center, radius, 0, angle)  plt.axes().text(test\_point[0], test\_point[1], 'Inside' if inside\_arc else 'Outside', color='red')  plt.show() |

测试结果

