**实验3 数字图像分割与边缘检测实验**

姓名：张日恒 学号：201050076

本次作业需要提交以下内容：

|  |  |
| --- | --- |
| 提交内容 | 详细要求 |
| 作业文档 | 对算法原理进行简单说明；  展示实验效果（每个实验的测试图片不得少于5张，实验用图像自行选择。）；  对实验结果进行分析。 |
| 程序源代码 | 相关程序的全部源代码，要求能够正常编译和运行。 |

**作业一：使用一阶检测算子（导数）进行图像的边缘检测。**

问题：自行编写程序而非调用函数，实现基于一阶Sobel算子的canny算子，进行图像的边缘提取。

原理:el算子根据像素点上下、左右邻点灰度加权差，在边缘处达到极值这一现象检测边缘。对噪声具有平滑作用，提供较为精确的边缘方向信息，边缘定位精度不够高。当对精度要求不是很高时，是一种较为常用的边缘检测方法。

|  |  |
| --- | --- |
| 原图 | Sobel |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

作业二： Hough线检测

问题1：对作业一中边缘检测的结果，进行Hough线检测。

问题2：测试多组（不少于3组）参数，提取较长的边界，分析结果。

|  |  |
| --- | --- |
| Sobel | Hough线检测 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**作业三：采用多种图像分割算法**

问题1：参考相关文献，编写程序实现分水岭算法，输入彩色图像，输出二值分割结果图像；

原理: 逐渐增加一个灰度阈值，每当它大于一个局部极大值时，就把当时的二值图像（只区分陆地和水域，即大于灰度阈值和小于灰度阈值两部分）与前一个时刻（即灰度阈值上一个值的时刻）的二值图像进行逻辑异或（XOR）操作，从而确定出灰度局部极大值的位置。根据所有灰度局部极大值的位置集合就可确定分水岭

|  |  |
| --- | --- |
| 原图 | 结果图 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

问题2：实现直方图阈值法，具体方法为采用灰度直方图求双峰或多峰，基于两峰之间的谷底计算阈值，将图像分割为前景和背景部分，结果显示为二值图像。

原理:

图像在小波变换后可分解为一系列尺度不同的分量。图像直方图在小波变换后也可进行多分辨率分析首先利用在粗分辨率下的直方图细节信息确定分割区域的类数，即检测出真正的峰点和谷点，确定类数后，可利用多分辨率的层次结构在直方图的相邻峰之间确定最优阈值，即对峰点和谷点进行较精确的定位

def sobel():  
 outdir = './result3/'  
 for index in range(5):  
 index = index + 1  
 imgpath = 'img' + str(index) + '.jpg'  
 image = cv2.imread(imgpath)  
 image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
 height, width = image.shape[:2]  
 new\_image = np.zeros((height, width, 3))  
 new\_imageX = np.zeros(image.shape)  
 new\_imageY = np.zeros(image.shape)  
 s\_X = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]]) # X方向  
 s\_Y = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])  
 for k in range(3):  
 for i in range(height - 2):  
 for j in range(width - 2):  
 new\_imageX[i + 1, j + 1, k] = abs(np.sum(image[i:i + 3, j:j + 3, k] \* s\_X))  
 new\_imageY[i + 1, j + 1, k] = abs(np.sum(image[i:i + 3, j:j + 3, k] \* s\_Y))  
  
 new\_image[:, :, k] = (new\_imageX[:, :, k] \* new\_imageX[:, :, k]  
 + new\_imageY[:, :, k] \* new\_imageY[:, :,k]) \*\* 0.5  
  
 new\_image = np.uint8(new\_image)  
 outimg1 = outdir + 'sobel\_' + str(index) + '.jpg'  
 cv2.imwrite(outimg1, new\_image)  
  
def hough\_line():  
 outdir = './result3/'  
 for index in range(5):  
 index = index + 1  
 imgpath = outdir + 'sobel\_' + str(index) + '.jpg'  
 image = cv2.imread(imgpath)  
 image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
 gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)  
 edges = cv2.Canny(gray, 50, 150, apertureSize=3)  
 lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 100, minLineLength=50, maxLineGap=10)  
 for line in lines:  
 x1, y1, x2, y2 = line[0]  
 cv2.line(gray, (x1, y1), (x2, y2), (0, 0, 255), 2)  
  
 outimg1 = outdir + 'hough\_' + str(index) + '.jpg'  
 cv2.imwrite(outimg1, gray)  
  
def watershed():  
 outdir = './result3/'  
 for index in range(5):  
 index = index + 1  
 imgpath = 'img' + str(index) + '.jpg'  
 image = cv2.imread(imgpath)  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)  
 kernel = np.ones((3, 3), np.uint8)  
 opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations=2)  
 sure\_bg = cv2.dilate(opening, kernel, iterations=3)  
 dist\_transform = cv2.distanceTransform(opening, cv2.DIST\_L2, 5) # DIST\_L1 DIST\_C只能 对应掩膜为3 DIST\_L2 可以为3或者5  
 ret, sure\_fg = cv2.threshold(dist\_transform, 0.1 \* dist\_transform.max(), 255, 0)  
 sure\_fg = np.uint8(sure\_fg)  
 unknow = cv2.subtract(sure\_bg, sure\_fg)  
 ret, markers = cv2.connectedComponents(sure\_fg, connectivity=8) # 对连通区域进行标号 序号为 0 - N-1  
 markers = markers + 1  
 markers[unknow == 255] = 0  
 markers = cv2.watershed(image, markers) # 分水岭算法后，所有轮廓的像素点被标注为 -1  
 # print(markers)  
  
 image[markers == -1] = [0, 0, 255] # 标注为-1 的像素点标 红  
 outimg1 = outdir + 'watershed\_' + str(index) + '.jpg'  
 cv2.imwrite(outimg1, image)  
  
 # cv2.imshow("thresh", thresh)  
  
def calcGrayHist(image):  
 rows,cols = image.shape  
 grayHist = np.zeros([256],np.uint64)  
 for r in range(rows):  
 for c in range(cols):  
 grayHist[image[r][c]] +=1#把图像灰度值作为索引  
 return(grayHist)  
  
def histpart():  
 outdir = './result3/'  
 for index in range(5):  
 index = index + 1  
 imgpath = 'img' + str(index) + '.jpg'  
 image = cv2.imread(imgpath)  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 print(gray.shape)  
 g = gray.flatten()  
 outpath = outdir + 'grayhist' + str(index) + '.jpg'  
 plt.title(outpath)  
 plt.hist(g, bins=256, density=1, facecolor='blue')  
 plt.savefig(outpath)  
 plt.close()