智能医学数字图像处理实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 19084129 | 班级 | 190841班 |
| 姓名 | 李奕澄 | 专业 | 生物医学工程 |
| 实验  名称 | 实验6：医学图像分割 | | |
| 实验  目的  和  内容 | **实验目的和要求：**  掌握传统基于边缘检测的图像分割⽅法  掌握传统基于区域检测的图像分割⽅法  **实验内容：**  python实现基于边缘检测的图像分割  python实现基于区域检测的图像分割 | | |
| 实验  结果  与  分析 | 基于边缘检测的图像分割  基于区域的图像分割  （1）区域生长算法：    （2）区域分裂合并算法  （3）分水岭算法： | | |
| 实验  代码 | #基于边缘检测图像分割  import cv2  import matplotlib.pyplot as plt  #import numpy as np  img = cv2.imread("test01.png")  gray\_img=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  # 高斯平滑  gaussian\_img = cv2.GaussianBlur(gray\_img,(3, 3),0)  # Laplacian算子  laplacian\_edge\_img = cv2.Laplacian(gaussian\_img, -1, ksize =3, delta = 5)  # Sobel提取边缘  grad\_x = cv2.Sobel(gaussian\_img, cv2.CV\_64F, 1, 0 )  abs\_grad\_x = cv2.convertScaleAbs( grad\_x )  grad\_y = cv2.Sobel( gaussian\_img, cv2.CV\_64F, 0, 1 )  abs\_grad\_y = cv2.convertScaleAbs(grad\_y)  sobel\_img = cv2.addWeighted(abs\_grad\_x, 0.5, abs\_grad\_y, 0.5, 0)  # Canny算子  canny\_edges\_img = cv2.Canny(gaussian\_img, 100, 200, 5)  plt.subplot(221), plt.imshow(img, 'gray'), plt.title('Source')  plt.axis('off')  plt.subplot(222), plt.imshow(laplacian\_edge\_img, 'gray'), plt.title('laplacian\_edge\_img')  plt.axis('off')  plt.subplot(223), plt.imshow(sobel\_img, 'gray'), plt.title('sobel\_edge\_img')  plt.axis('off')  plt.subplot(224), plt.imshow(canny\_edges\_img, 'gray'), plt.title('canny\_edges\_img')  plt.axis('off')  #基于区域检测图像分割  '''区域生长'''  import numpy as np  import cv2  import matplotlib.pyplot as plt    class Point(object):  def \_\_init\_\_(self,x,y):  self.x = x  self.y = y    def getX(self):  return self.x  def getY(self):  return self.y    def getGrayDiff(img,currentPoint,tmpPoint):  return abs(int(img[currentPoint.x,currentPoint.y]) - int(img[tmpPoint.x,tmpPoint.y]))    def selectConnects(p):  if p != 0:  connects = [Point(-1, -1), Point(0, -1), Point(1, -1), Point(1, 0), Point(1, 1), \  Point(0, 1), Point(-1, 1), Point(-1, 0)]  else:  connects = [ Point(0, -1), Point(1, 0),Point(0, 1), Point(-1, 0)]  return connects    def regionGrow(img,seeds,thresh,p = 1):  height, weight = img.shape  seedMark = np.zeros(img.shape)  seedList = []  for seed in seeds:  seedList.append(seed)  label = 1  connects = selectConnects(p)  while(len(seedList)>0):  currentPoint = seedList.pop(0)    seedMark[currentPoint.x,currentPoint.y] = label  for i in range(8):  tmpX = currentPoint.x + connects[i].x  tmpY = currentPoint.y + connects[i].y  if tmpX < 0 or tmpY < 0 or tmpX >= height or tmpY >= weight:  continue  grayDiff = getGrayDiff(img,currentPoint,Point(tmpX,tmpY))  if grayDiff < thresh and seedMark[tmpX,tmpY] == 0:  seedMark[tmpX,tmpY] = label  seedList.append(Point(tmpX,tmpY))  return seedMark      img = cv2.imread('test01.png',0)  seeds = [Point(10,10),Point(82,150),Point(20,300)]  binaryImg = regionGrow(img,seeds,10)  plt.subplot(121), plt.imshow(img, 'gray'), plt.title('input'),plt.axis('off')  plt.subplot(122), plt.imshow(binaryImg, 'gray'), plt.title('output'),plt.axis('off')  '''区域分裂合并算法'''  import numpy as np  import cv2  import matplotlib.pyplot as plt  #判断方框是否需要再次拆分为四个  def judge(w0, h0, w, h):  a = img[h0: h0 + h, w0: w0 + w]  ave = np.mean(a)  std = np.std(a, ddof=1)  count = 0  total = 0  for i in range(w0, w0 + w):  for j in range(h0, h0 + h):  #注意！我输入的图片数灰度图，所以直接用的img[j,i]，RGB图像的话每个img像素是一个三维向量，不能直接与avg进行比较大小。  if abs(img[j, i] - ave) < 1 \* std:  count += 1  total += 1  if (count / total) < 0.95:#合适的点还是比较少，接着拆  return True  else:  return False  ##将图像将根据阈值二值化处理，在此默认125  def draw(w0, h0, w, h):  for i in range(w0, w0 + w):  for j in range(h0, h0 + h):  if img[j, i] > 125:  img[j, i] = 255  else:  img[j, i] = 0  def function(w0, h0, w, h):  if judge(w0, h0, w, h) and (min(w, h) > 5):  function(w0, h0, int(w / 2), int(h / 2))  function(w0 + int(w / 2), h0, int(w / 2), int(h / 2))  function(w0, h0 + int(h / 2), int(w / 2), int(h / 2))  function(w0 + int(w / 2), h0 + int(h / 2), int(w / 2), int(h / 2))  else:  draw(w0, h0, w, h)  img = cv2.imread('test01.png', 0)  img\_input = cv2.imread('test01.png', 0)#备份  height, width = img.shape  function(0, 0, width, height)  plt.subplot(121), plt.imshow(img\_input, 'gray'), plt.title('input'),plt.axis('off')  plt.subplot(122), plt.imshow(img, 'gray'), plt.title('output'),plt.axis('off')  '''分水岭算法'''  import numpy as np  import cv2  import matplotlib.pyplot as plt  def 分水岭算法(img):  #转化成灰度图，方便处理  gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)  #二值化  ret,thresh=cv2.threshold(gray,0,255,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)  #去除噪音(要不然最终成像会导致过度分割)  kernel=np.ones((3,3),np.uint8)  opening=cv2.morphologyEx(thresh,cv2.MORPH\_OPEN,kernel,iterations=2)  #确定非对象区域  sure\_bg=cv2.dilate(opening,kernel,iterations=3)#进行膨胀操作  #确定对象区域  dist\_transform=cv2.distanceTransform(opening,1,5)  ret,sure\_fg=cv2.threshold(dist\_transform,0.7\*dist\_transform.max(),255,0)  #寻找未知的区域  sure\_fg=np.uint8(sure\_fg)  unknown=cv2.subtract(sure\_bg,sure\_fg)#非对象区域减去对象区域就是不确定区域  # 为对象区域类别标记  ret, markers = cv2.connectedComponents(sure\_fg)  # 为所有的标记加1，保证非对象是0而不是1  markers = markers+1  # 现在让所有的未知区域为0  markers[unknown==255] = 0  #执行分水岭算法  markers = cv2.watershed(img,markers)  img[markers == -1] = [255,0,0]  #解决中文显示问题  return gray,opening,sure\_bg,dist\_transform,unknown,img  if \_\_name\_\_ == '\_\_main\_\_':  img3 = cv2.imread("test01.png")#读图  gray,opening,sure\_bg,dist\_transform,unknown,img = 分水岭算法(img3)    plt.rcParams['font.sans-serif']=['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  plt.subplot(231), plt.imshow(gray, 'gray'), plt.title('输入图片'),plt.axis('off')  plt.subplot(232), plt.imshow(opening,'gray'), plt.title('二值化去噪之后'),plt.axis('off')  plt.subplot(233), plt.imshow(sure\_bg,'gray'), plt.title('确定非对象区域'),plt.axis('off')  plt.subplot(234), plt.imshow(dist\_transform,'gray'), plt.title('确定对象区域'),plt.axis('off')  plt.subplot(235), plt.imshow(unknown,'gray'), plt.title('未知区域'),plt.axis('off')  plt.subplot(236), plt.imshow(img,'gray'), plt.title('分水岭算法'),plt.axis('off')  plt.show() | | |
| 成绩  评定 | 教师签名：  年 月 日 | | |