|  |  |
| --- | --- |
|  |  |
| **学 号：** | 0121708910325 |

****

**《计算机视觉》**

**期末大作业**

|  |  |
| --- | --- |
|  |  |
| **学 院** | 计算机学院 |
| **专 业** | 计算机科学与技术 |
| **班 级** | 计算机 |
| **姓 名** |  |
| **指导教师** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2021 | 年 | 1 | 月 | 1 | 日 |

|  |  |
| --- | --- |
| **选**  **题说明** | **题目:** 定制化表格识别 |
| **作**  **业 内**  **容** | 1. 引言（简要叙述需要解决的问题，设计思想和技术路线，不少于300字，15分）。 2. 倾斜表格校正方法（可使用canny边缘检测、hough变换计算角度，仿射变换旋转表格，不少于500字，20分）。 3. 印章检测方法（可使用图像分割，hough圆变换，不少于500字，20分）。 4. 文字识别方法（详细描述tesseract 原理，训练与使用方法，不少于500字，20分）。 5. 实验（分别给出表格校正、印章检测、文字识别结果图像，以及对图像描述，不少于500字，20分）。 6. 总结（对采用方法正确性进行总结，不少于150字，5分）。 7. 检测表格结构，详述其实现逻辑，可额外加分。 8. 附录：附上全部代码，对算法关键部分进行注释。   **(关于格式：**答案不要写在题目页（本页）。整个作业必须用五号字体，不得改变封面及题目页的格式，不得没有封面及题目页，必须完整填写封面页信息。  答题时要写明标题，依次答题。**关于纪律**：不得抄袭，不得雷同。否则，按期末零分处理。  **作业上交时间**：16周周日--2020年12月27日**）**   1. 引言   定制化表格识别是通过对具有表格特征的表单通过扫描、图像处理、OCR（光学字符识别）技术，处理表格图像，将表格图像中的数据信息准确，快速、真实的提取并保存。  识别表格首先要对图像进行处理，如果表格是倾斜的，对于后面信息的提取等图像处理不是很方便，该问题可以通过先通过边缘检测，霍夫变换得到线条，计算角度然后进行，仿射变换进行矫正；表格中的印章可以通过hough圆进行检测，对于方形的印章，可以通过，图像分割中的GraphCut算法进行检测；表格中的文字通过tesseract-ocr进行识别，通过对图片集的识别训练，矫正得到数据集后可以和中文数据集一起合并使用，提高识别正确率。   1. 倾斜表格校正方法   1.倾斜表格的矫正主要流程就是读入图片，先对图片进行灰度化，然后进行腐蚀、膨胀，接着进行canny边缘检测，然后用hough变换得到线条，得到线条后画出线条并进行计算角度，最后进行仿射变换，将原图校正。  2.对图片进行灰度化是为了简化数组，方便处理；对图片进行腐蚀是为了去除图片中的一些毛刺，防止影响边缘检测和得到线条，腐蚀后再进行膨胀填补表格线之间的缺陷。    3.Canny边缘检测是一种非常流行的边缘检测算法，它是一个多步骤的算法：  首先应用高斯滤波来平滑图像去除噪声，接着计算图像梯度，得到可能边缘，因为梯度是灰度变化明显的地方，而边缘也是灰度变化明显的地方。然后进行非极大值抑制，这一步的目的是将模糊的边界变得清晰，实际操作就是在局部范围内的梯度方向上，灰度最大的保留下来，剔除其他点。经过非极大值抑制后图像中仍然有很多的噪声点，Canny算法通过双阈值技术，即设定一个阈值的上限和下限，灰度变化大于high的，设置为强边缘像素，低于low的，剔除。在low和high之间的设置为弱边缘。进一步判断，如果其领域内有强边缘像素，保留，如果没有，剔除。    4.霍夫变换通过一种投票算法检测直线或圆，主要原理是对于边缘的每一个像素点，把可能经过他的所有直线y=kx+b，映射到k-b空间，然后进行投票。但是对于与X轴垂直的直线斜率不存在，无法表示，所以用参数方程来表示，最后霍夫空间可以用r-theta表示。    5.得到并画出直线后，求出直线斜率，在进行计算旋转角度，计算角度,因为x轴向右，y轴向下，所有计算的斜率是常规下斜率的相反数，我们就用这个斜率（旋转角度）进行旋转。    6．Python中可以使用warpAffine进行仿射变换旋转，首先通过getRotationMatrix2D得到变换矩阵，然后调用warpAffine即可。     1. 印章检测方法   检测印章的方法可以使用Hough圆变换，也可以使用图像分割将印章分割显示出来。  1.对于圆型的印章，检测方法比较简单，可以调用HoughCircles进行检测，其检测原理和检测直线的原理差别不大，只不过直线是在二维空间，因为y=kx+b，只有k和b两个自由度。而圆形的一般性方程表示为(x-a)²+(y-b)²=r²。那么就有三个自由度圆心坐标a,b,和半径r，HoughCircles可以设定半斤r的取值范围，能够减少计算量。具体步骤如下：  (1)对输入图像进行边缘检测，获取边界点，即前景点。  (2)如果图像中存在圆形，那么其轮廓必定属于前景点。  (3)同霍夫变换检测直线一样，将圆形的一般性方程换一种方式表示，进行坐标变换。由x-y坐标系转换到a-b坐标系。写成如下形式(a-x)²+(b-y)²=r²。那么x-y坐标系中圆形边界上的一点对应到a-b坐标系中即为一个圆。  (4)那x-y坐标系中一个圆形边界上有很多个点，对应到a-b坐标系中就会有很多个圆。由于原图像中这些点都在同一个圆形上，那么转换后a,b必定也满足a-b坐标系下的所有圆形的方程式。直观表现为这许多点对应的圆都会相交于一个点，那么这个交点就可能是圆心(a, b)。  (5)统计局部交点处圆的个数，取每一个局部最大值，就可以获得原图像中对应的圆形的圆心坐标(a,b)。一旦在某一个r下面检测到圆，那么r的值也就随之确定。    2. 图像分割指的是根据灰度、颜色、纹理和形状等特征把图像划分成若干不相交的区域，并使这些特征在同一区域内呈现出相似性，而在不同区域间呈现出明显的差异性。  GrabCut算法，是基于图论的分割方法，该算法主要基于K均值聚类、高斯混合模型建模等知识。该算法的实现步骤是：  1.在图片中定义(一个或者多个)包含物体的矩形，矩形外的区域被自动认为是背景。  2.对于用户定义的矩形区域，可用背景中的数据来区分它里面的前景和背景区域。  3.用高斯混合模型(GMM)来对背景和前景建模，并将未定义的像素标记为可能的前景或者背景。    4.图像中的每一个像素都被看做通过虚拟边与周围像素相连接，而每条边都有一个属于前景或者背景的概率，这是基于它与周边像素颜色上的相似性。  5.每一个像素(即算法中的节点)会与一个前景或背景节点连接。  6.在节点完成连接后(可能与背景或前景连接)，若节点之间的边属于不同终端(即一个节点属于前景，另一个节点属于背景)，则会切断他们之间的边，这就能将图像各部分分割出来。    7. 调用完grabCut函数之后，掩模图像mask元素值已经变成了0~3之间的值。值为0和2的将转为0，值为1和3的将转为1，然后保存在mask2中，这样就可以用mask2过滤出所有的0值像素(理论上会保存所有的前景像素)。     1. 文字识别方法   Tesseract是惠普布里斯托实验室在1985到1995年间开发的一一个开源的OCR引擎，OCR（optical character recognition，光学字符识别）是指直接将包含文本的图像识别为计算机文字（计算机黑白点阵）的技术。 Tesseract的识别步骤大致如下：  1. 连通区域分析,检测出字符区域区域（轮廓外形）,以及子轮廓。在此阶段轮廓线集成为块区域。  2. 由字符轮廓和块区域得出文本行，以及通过空格识别出单词。固定字宽文本通过字符单元分割出单个字符，而对百分号的文本（Proportional text）通过一定的间隔和模糊间隔就（fuzzy spaces）来分割；  3. 依次对每个单词进行分析，采用自适应分类器，分类器有学习能力，先分析的且满足条件的单词也作为训练样本，所以后面的字符（比如页尾）识别更准确；此时，页首的字符识别比较不准确，所以tesseract会再次对识别不太好的字符识别是其精度得到提高。  4.最后，识别含糊不清的空格，及用其他方法，如由笔画高度（x-height），识别小写字母（small-cap）的文本。  安装Tesseract，并配置好环境变量和下载chi\_sim.traindata字库后即可用来识别文字：  命令格式： tesseract 图片路径 生成结果文件名 字库    运行命令后可在result.txt文件查看文字识别结果。但一般识别效果不理想，所以我们要训练自己的字库。步骤如下：   1. 将图片转换程tif图片(如果多张图片，则需要合并tif)。   tif文件命名格式[lang].[fontname].exp[num].tif  tif命名规则：lang为语言名称，fontname为字体名称，num为图片序号；   1. 生成box文件      1. 下载JTessBoxEditor，解压，点击运行，打开要训练的tif文件，手动对识别的文字进行矫正，完成后保存。 2. 生成font\_properties。 3. 生成训练文件，字符集文件，聚集字符特征文件，字符正常化特征文件。 4. 重新命令上面生成的文件。 5. 合并训练文件，合并后就可以使用训练好的traindata文件(复制到tesseract安装目录下的tessdata目录下)去识别文字。   以上步骤4-7步，可以写成bat脚本，矫正完成后生成相关文件，简化操作流程。脚本内容如下：     1. 实验（分别给出表格校正、印章检测、文字识别结果图像，以及对图像描述，不少于500字，20分）。   表格校正：  下面给出了原图和经过校正过后的表格图像，这张原图是样本中30001.jpg。从截图中可以看到，原图像的最下方横线与系统栏呈一定的角度，而经过校正的图像最下凡的横线几乎与系统栏平行。    印章检测：   |  |  | | --- | --- | |  | 左图是通过霍夫圆检测得到的结果，python中通过调用 cv2.HoughCircles函数来找到图中的圆，其中HoughCircles函数7个参数中minRadius和maxRadius两个参数分别设置为30，150能较好的识别出表格中的圆形印章。 |  |  |  | | --- | --- | |  |  |   上面两个图是用GrabCut图像分割算法检测得到的印章，该方法也能很好的检测出印章，我将原图中印章分为五个区域，三个圆印章，两个方形印章，对这五个区域分别进行识别。结果中，颜色鲜艳的圆形印章检测的十分好，但是方形印章只检测出半个来，所以存在的问题是该方法能较好的检测出颜色鲜艳的印章，但是若是印章的颜色较浅，就难以检测。      经过一些测试，我尝试用一个大的矩形代替五个矩形识别，识别结果如下图，如图所示，方形印章也能被检测出，但是印章的提取结果不是很规范，所以较好的方式还是对颜色较深的印章图片进行分区域提取，或者应该也可以进行一些图片的处理后再进行提取印章。    文字识别方法：    上图是对要识别的图片转换成tif文件，生成box文件后，打开tif文件进行识别结果的矫正，通过JTessBoxEditor打开tif文件，我们在左侧对识别的文字进行手动纠正，我们可以通过合并多个图片文件生成tif进行矫正，保存后，生成自己的训练集，tesseract的识别率，通过样本数量提升，呈对数型增长。   |  |  | | --- | --- | |  |  |   以上两个图片，左侧是用chi\_sim官方简体中文训练集训练出的，右侧是用自己训练出的训练集（样本小，手动训练效率低）加chi\_sim训练集训练出的。我们可以看到，经过自己手动矫正训练过的训练集使右侧的一些识别情况比左侧稍好些，但仍然不是很好，应该需要训练大量的样本才能达到较好的文字识别结果。   1. 总结（对采用方法正确性进行总结，不少于150字，5分）。   定制化表格识别中，倾斜表格矫正采用Canny边缘检测、Hough直线检测和仿射变换的方法进行表格矫正，从原图和结果图的对比来看，能够达到矫正表格的效果。  印章检测中，Hough圆检测能够较好的检测圆形印章，但是不能够检测出方形印章，图像分割GrabCut算法如果每个印章对应一个矩形，那就能够对颜色鲜艳的印章进行很好的检测，但检测不出颜色较浅的印章；如果用一个大矩形去检测，颜色不很鲜艳的方形印章也能够检测出，但是结果图形不是很贴合。  Tesseract进行文字识别，如果用官方中文训练包chi\_sim识别出的结果不是很好，自己对一些图片进行训练后结合chi\_sim使用能够改进一些识别结果，但是若自己训练集样本不够大，训练的不够多，那么文字识别正确率的提升有限。   1. 附录：附上全部代码，对算法关键部分进行注释。  |  | | --- | | *倾斜表格矫正完整代码：*  *#仿射变换*  *def rotate(image, angle, center=None, scale=1.0):*  *(w, h) = image.shape[0:2]*  *if center is None:*  *center = (w // 2, h // 2)*  *wrapMat = cv2.getRotationMatrix2D(center, angle, scale)*  *return cv2.warpAffine(image, wrapMat, (h, w))*  *"""*  *通过霍夫变换、反三角函数获得斜率*  *1、输入图片*  *2、灰度化*  *3、腐蚀、膨胀*  *4、边缘检测*  *5、霍夫变换得到线条并画出*  *6、计算角度*  *7、仿射变换，将原图校正*  *"""*  *def showAndWaitKey(winName,img):*  *cv2.imshow(winName,img)*  *cv2.waitKey()*  *def getCorrect(src):*  *# 读取图片，灰度化*  *gray = cv2.cvtColor(src, cv2.COLOR\_BGR2GRAY)*  *# 腐蚀、膨胀*  *kernel = np.ones((5, 5), np.uint8)*  *erode\_Img = cv2.erode(gray, kernel)*  *eroDil = cv2.dilate(erode\_Img, kernel)*  *#showAndWaitKey("eroDil", eroDil)*  *# 边缘检测*  *canny = cv2.Canny(eroDil, 50, 150)*  *#showAndWaitKey("canny", canny)*  *# 霍夫变换得到线条*  *lines = cv2.HoughLinesP(canny, 0.8, np.pi / 180, 90, minLineLength=100, maxLineGap=10)*  *drawing = np.zeros(src.shape[:], dtype=np.uint8)*  *# 画出线条*  *for line in lines:*  *x1, y1, x2, y2 = line[0]*  *cv2.line(drawing, (x1, y1), (x2, y2), (0, 255, 0), 1, lineType=cv2.LINE\_AA)*  *#showAndWaitKey("houghP", drawing)*  *"""*  *计算角度,因为x轴向右，y轴向下，所有计算的斜率是常规下斜率的相反数，我们就用这个斜率（旋转角度）进行旋转*  *"""*  *k = float(y1 - y2) / (x1 - x2)*  *thera = np.degrees(math.atan(k))*  *print(thera)*  *if thera>0:*  *thera=thera-90*  *"""*  *旋转角度大于0，则逆时针旋转，否则顺时针旋转*  *"""*  *rotateImg = rotate(src, thera)*  *cv2.imshow("rotateImg", rotateImg)*  *cv2.waitKey()*  *def cv\_imread(file\_path=""):*  *file\_path\_gbk=file\_path.encode('gbk')# unicode转gbk，字符串变为字节数组*  *img=cv2.imread(file\_path\_gbk.decode())# 字节数组直接转字符串，不解码*  *return img*  *if \_\_name\_\_ == '\_\_main\_\_':*  *src = cv\_imread("D:/图片/samples/30001.jpg")*  *w, h = src.shape[0:2]*  *image = cv2.resize(src, (0, 0), fx=0.25, fy=0.25, interpolation=cv2.INTER\_NEAREST)*  *getCorrect(image)* |   *---------------------------------------------------------------------------------------------------------------------------*   |  | | --- | | *印章检测：*  *Hough圆检测圆形印章代码：*  *"""*  *霍夫圆检测*  *"""*  *def detect\_circle\_demo(image):*  *dst=cv2.pyrMeanShiftFiltering(image,10,100)*  *cimage=cv2.cvtColor(dst,cv2.COLOR\_BGR2GRAY)*  *#cv2.imshow("gray",cimage)*  *circle = cv2.HoughCircles(cimage, cv2.HOUGH\_GRADIENT,1,200,param1=50,param2=30,minRadius=30,maxRadius=150)*  *if not circle is None:*  *circle=np.uint16(np.around(circle))*  *print(circle)*  *for i in circle[0,:]:*  *cv2.circle(image,(i[0],i[1]),i[2],(0,255,0),2)*  *cv2.imshow("circle",image)*  *"""*  *cv::imread不能读带有中文的字符，该函数用来解决该问题*  *"""*  *def cv\_imread(file\_path=""):*  *file\_path\_gbk=file\_path.encode('gbk')# unicode转gbk，字符串变为字节数组*  *img=cv2.imread(file\_path\_gbk.decode())# 字节数组直接转字符串，不解码*  *return img*  *if \_\_name\_\_ == '\_\_main\_\_':*  *src=cv\_imread("D:/图片/samples/30032.jpg")*  *w,h=src.shape[0:2]*  *image=cv2.resize(src,(0,0),fx=0.25,fy=0.25,interpolation=cv2.INTER\_NEAREST)*  *cv2.imshow("original\_image",image)*  *detect\_circle\_demo(image)*  *cv2.waitKey(0)*  *cv2.destroyAllWindows()* |   *图像分割检测印章代码（多个矩形）：*  *# Step1. 加载图像*  *img = cv2.imread('D:/3.jpg')*  *# Step2. 创建掩模、背景图和前景图*  *mask = np.zeros(img.shape[:2], np.uint8)# 创建大小相同的掩模*  *mask2 = np.zeros(img.shape[:2], np.uint8)#*  *mask3 = np.zeros(img.shape[:2], np.uint8)#*  *mask4 = np.zeros(img.shape[:2], np.uint8)#*  *mask5 = np.zeros(img.shape[:2], np.uint8)#创建大小相同的掩模*  *bgdModel = np.zeros((1,65), np.float64)# 创建背景图像*  *fgdModel = np.zeros((1,65), np.float64)# 创建前景图像*  *# Step3. 初始化矩形区域*  *# 这个矩形必须完全包含前景*  *rect1 = (120,160,240,240) #格式为（x, y, w, h）*  *rect2 = (420,320,240,240)*  *rect3 = (550,380,200,200)*  *rect4 = (380,550,240,240)*  *rect5 = (520,700,180,180)*  *# Step4. GrubCut算法，迭代5次*  *# mask的取值为0,1,2,3*  *#第一次识别*  *cv2.grabCut(img, mask, rect1, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT) # 迭代5次*  *#第二次识别*  *cv2.grabCut(img, mask2, rect2, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT) # 迭代5次*  *#第三次识别*  *cv2.grabCut(img, mask3, rect3, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT) # 迭代5次*  *#第四次识别*  *cv2.grabCut(img, mask4, rect4, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT) # 迭代5次*  *#第五次识别*  *cv2.grabCut(img, mask5, rect5, bgdModel, fgdModel, 5, cv2.GC\_INIT\_WITH\_RECT) # 迭代5次*  *# Step5. mask中，值为2和0的统一转化为0, 1和3转化为1*  *mask6 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')*  *mask7 = np.where((mask2 == 2) | (mask2 == 0), 0, 1).astype('uint8')*  *mask8 = np.where((mask3 == 2) | (mask3 == 0), 0, 1).astype('uint8')*  *mask9 = np.where((mask4 == 2) | (mask4 == 0), 0, 1).astype('uint8')*  *mask10 = np.where((mask5 == 2) | (mask5 == 0), 0, 1).astype('uint8')*  *mask11=np.add(mask6,mask7)*  *mask12=np.add(mask8,mask9)*  *mask13=np.add(mask11,mask12)*  *mask14=np.add(mask13,mask10)*  *img = img \* mask14[:,:,np.newaxis] # np.newaxis 插入一个新维度，相当于将二维矩阵扩充为三维*  *cv2.imshow("dst", img)*  *cv2.imwrite("3.jpg",img)*  *cv2.waitKey(0)*  *图像分割检测印章代码（单个矩形）：*  *#1. 读取图片*  *img = cv.imread('D:/3.jpg')*  *#2. 掩码图像*  *mask = np.zeros(img.shape[:2],np.uint8)*  *#3.矩形窗口(x,y,w,h)；*  *rect = [110,160,800,800]*  *#4.物体分割*  *cv.grabCut(img,mask,tuple(rect),None,None,5,cv.GC\_INIT\_WITH\_RECT)*  *#5.抠取图像*  *mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')*  *img\_show = img\*mask2[:,:,np.newaxis]*  *cv.imshow("test",img\_show)*  *cv.waitKey(0)*  *cv.imwrite("fullSeal.jpg",img\_show)*  *-------------------------------------------------------------------------------------------------------------*  *Tesseract文字识别脚本：*  *set filename=table.test.exp50*  *:: 生成font\_properties*  *echo Generate font\_properties...*  *echo test 0 0 0 0 0 >font\_properties*  *:: 生成训练文件*  *echo Run Tesseract for Training...*  *tesseract %filename%.tif %filename% nobatch -l chi\_sim nobatch box.train*  *:: 生成字符集文件*  *echo Generate unicharset...*  *unicharset\_extractor %filename%.box*  *:: 生成聚集字符特征文件*  *echo Generate Character...*  *mftraining -F font\_properties -U unicharset -O unicharset %filename%.tr*  *:: 生成字符正常化特征文件*  *echo Clustering...*  *cntraining %filename%.tr*    *:: 重新命名*  *echo Rename Files...*  *rename normproto table.normproto*  *rename inttemp table.inttemp*  *rename pffmtable table.pffmtable*  *rename shapetable table.shapetable*  *rename unicharset table.unicharset*    *:: 合并训练文件*  *echo Create Tessdata...*  *combine\_tessdata table.*  *pause* |
| **成绩评定** | **教师签名：** |