华南师范大学

软件 学院 2023 —2024 学年第一学期期末考试试卷

《 数字图像处理基础 》试卷（ 作品）

专业 软件工程 年级 21 班级 7 姓名 曾荣豪

学号 20212005343

题号	一	二	三	四	五	六	七	八	九	十	总分
得分											

一、概述

使用Python的OpenCV，pyplot，numpy实现的停车场车位使用情况分析，包括车牌定位，车牌字符分割，车牌字符识别，停车场车位编号识别。

二、整体设计

首先对图像进行增强，然后使用findContours函数找到图像中的轮廓，对轮廓进行筛选，找到可能属于车牌和标号的轮廓，然后对此轮廓进行处理，使用findContours分割一个字符，最后使用matchTemplate方法识别车牌字符和停车场编号字符，将停车场编号与车牌编号进行匹配，得到车牌对应的车位编号。

三、具体实现

1. 车牌识别

1. 读取需要进行车牌识别的图片；

```
# 读取待检测图片
origin_image = cv2.imread('./image/car.jpg')
```

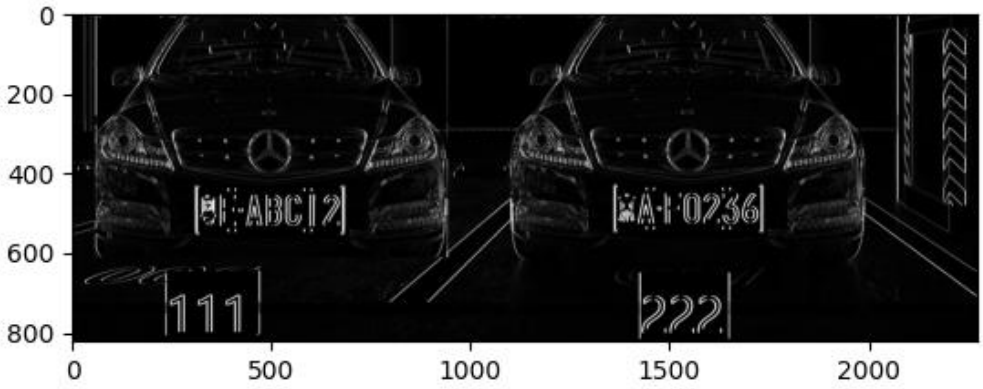
```
# 复制一张图片，在复制图上进行图像操作，保留原图
image = origin_image.copy()
```

2. 图像去噪灰度处理;

```
gray_image = gray_guss(image)
```

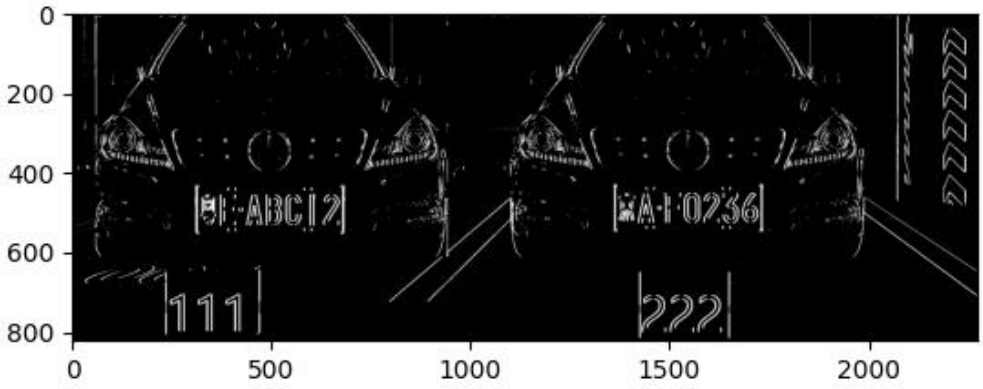
3. x方向上的边缘检测（增强边缘信息）；

```
Sobel_x = cv2.Sobel(gray_image, cv2.CV_16S, 1, 0)
absX = cv2.convertScaleAbs(Sobel_x)
image = absX
```



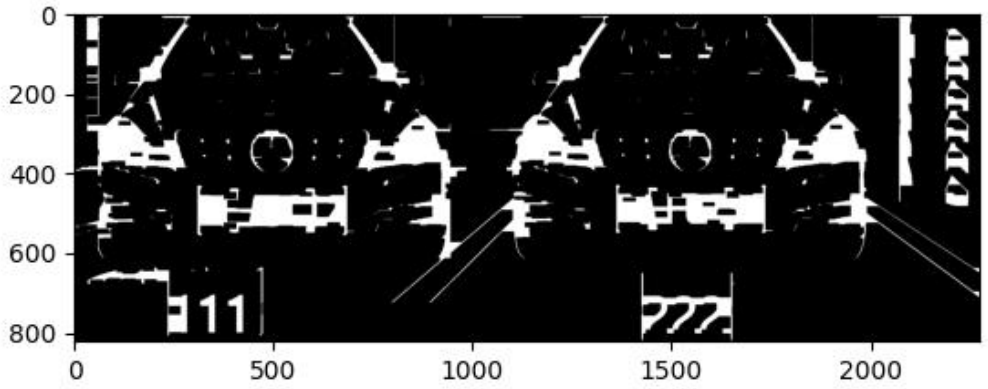
4. 将图像二值化；

```
ret, image = cv2.threshold(image, 0, 255, cv2.THRESH_OTSU)
```



5. 进行闭运算操作，获得小连通域；

```
kernelX = cv2.getStructuringElement(cv2.MORPH_RECT, (30,
10))
image = cv2.morphologyEx(image, cv2.MORPH_CLOSE,
kernelX,iterations = 1)
```



6. 进行腐蚀（erode）和膨胀（dilate）；

```
# 腐蚀（erode）和膨胀（dilate）
kernelX = cv2.getStructuringElement(cv2.MORPH_RECT, (50,
1))
kernelY = cv2.getStructuringElement(cv2.MORPH_RECT, (1,
20))
#x方向进行闭操作（抑制暗细节）
image = cv2.dilate(image, kernelX)
image = cv2.erode(image, kernelX)
#y方向的开操作
image = cv2.erode(image, kernelY)
image = cv2.dilate(image, kernelY)
```

```
# 中值滤波（去噪）
image = cv2.medianBlur(image, 21)
```

7. 使用findContours获取车牌和标号的轮廓；

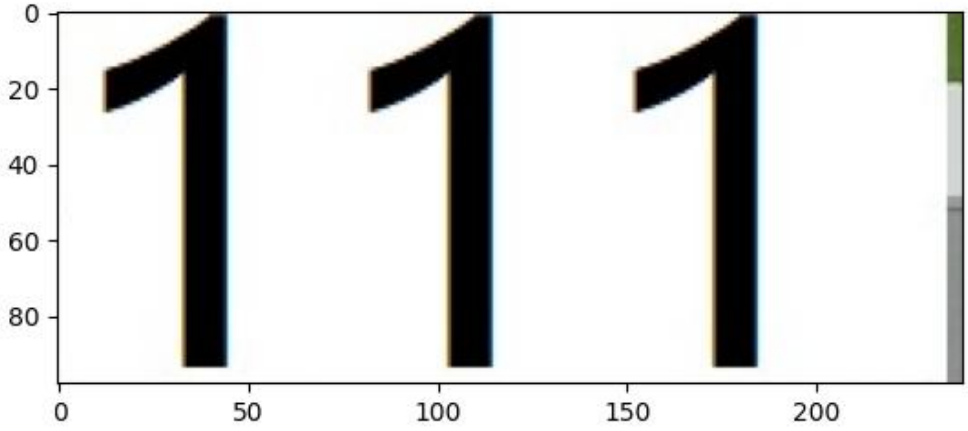
```
contours, hierarchy = cv2.findContours(image,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

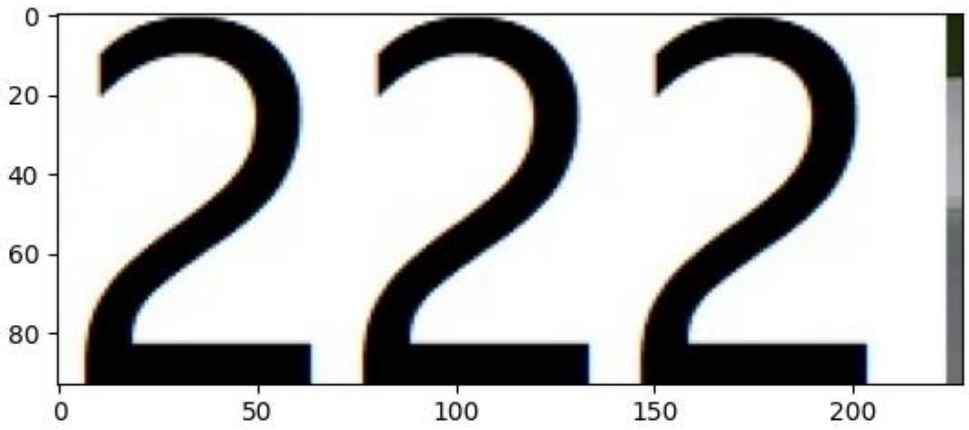
8. 根据轮廓的面积进行筛选，找到可能属于车牌和标号的轮廓；

```
for j, item in enumerate(contours):
    rect = cv2.boundingRect(item)
    x = rect[0]
    y = rect[1]
    weight = rect[2]
    height = rect[3]
    # 根据轮廓的形状特点，确定车牌和标号的轮廓位置并截取图像

    if (weight > (height * 3)) and (weight < (height *
5)):

        cut_image = origin_image[y:y + height, x:x +
```

```
weight]

plates.append((x,y,weight,height,cut_image.copy()))
        plt_writeRGB(cut_image,
img_name+str(j)+"_car_plate.jpg")

    elif(weight>height*1.5 and weight<height*2.5):
        cut_image = origin_image[y:y + height, x:x +
weight]
        nums.append((x,y,weight,height, cut_image.copy()))
        plt_writeRGB(cut_image.copy(),
img_name+str(j)+"_plate.jpg")
```


京A·F0236


1 1 1



2. 停车场编号、车牌字符分割

1. 对停车场编号、车牌字符进行去噪灰度处理；

```
gray_image = gray_guss(image)
```

2. 对停车场编号、车牌字符进行二值化处理；

```
ret, image = cv2.threshold(gray_image, 0, 255,
cv2.THRESH_OTSU)
```

3. 膨胀操作，为分割做准备

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2))
image = cv2.dilate(image, kernel)
```

4. 使用findContours函数找到图像中的轮廓；

```
contours, hierarchy = cv2.findContours(image,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    words = []
    word_images = []
    #对所有轮廓逐一操作
    if(contours==[]):
        return
    for item in contours:
        word = []
        rect = cv2.boundingRect(item)
        x = rect[0]
        y = rect[1]
        weight = rect[2]
        height = rect[3]
        word.append(x)
        word.append(y)
        word.append(weight)
        word.append(height)
        words.append(word)
```

```
    # 排序，车牌号有顺序。words是一个嵌套列表
    words = sorted(words,key=lambda s:s[0],reverse=False)
```

5. 对轮廓进行筛选，找到可能属于车牌和停车场编号的字符轮廓；

```
for word in words:
    # 筛选字符的轮廓
    i = i+1
    testimg = image[word[1]:word[1] + word[3],
word[0]:word[0] + word[2]]
    plt.subplot(1, len(words)+1, i+1)
    plt.imshow(testimg,cmap='gray')
      # 添加一个图像来展示

    if (word[3] > (word[2] * 1.5)) and (word[3] < (word[2]
* 3.5)) and (word[2] > 25):

        splite_image = image[word[1]:word[1] + word[3],
word[0]:word[0] + word[2]]
        word_images.append(splite_image)
```

3. 匹配车牌和停车场编号

1. 初始化停车场编号和车牌字符的模板；

```
# 准备模板(template[0-9]为数字模板；)
template = ['0','1','2','3','4','5','6','7','8','9',

'A','B','C','D','E','F','G','H','J','K','L','M','N','P','Q','

'藏','川','鄂','甘','赣','贵','桂','黑','沪','吉','冀','津','晋

'青','琼','陕','苏','皖','湘','新','渝','豫','粤','云','浙']

# 读取一个文件夹下的所有图片，输入参数是文件名，返回模板文件
地址列表
def read_directory(directory_name):
    referImg_list = []
    for filename in os.listdir(directory_name):
        referImg_list.append(directory_name + "/" +
filename)
    return referImg_list

# 获得中文模板列表（只匹配车牌的第一个字符）
def get_chinese_words_list():
    chinese_words_list = []
    for i in range(34,64):
        #将模板存放在字典中
        c_word = read_directory('./refer1/'+ template[i])
        chinese_words_list.append(c_word)
    return chinese_words_list
chinese_words_list = get_chinese_words_list()


# 获得英文模板列表（只匹配车牌的第二个字符）
def get_eng_words_list():
    eng_words_list = []
    for i in range(10,34):
        e_word = read_directory('./refer1/'+ template[i])
        eng_words_list.append(e_word)
    return eng_words_list
eng_words_list = get_eng_words_list()
```

```
# 获得英文和数字模板列表（匹配车牌后面的字符）
def get_eng_num_words_list():
    eng_num_words_list = []
    for i in range(0,34):
        word = read_directory('./refer1/'+ template[i])
        eng_num_words_list.append(word)
    return eng_num_words_list
eng_num_words_list = get_eng_num_words_list()
```

2. 使用matchTemplate方法识别车牌字符和停车场编号字符，并返回得分；

```
# 读取一个模板地址与图片进行匹配，返回得分
def template_score(template,image):
    #将模板进行格式转换

template_img=cv2.imdecode(np.fromfile(template,dtype=np.uint8
    template_img = cv2.cvtColor(template_img,
cv2.COLOR_RGB2GRAY)
    #模板图像阈值化处理——获得黑白图
    ret, template_img = cv2.threshold(template_img, 0,
255, cv2.THRESH_OTSU)
#     height, width = template_img.shape
#     image_ = image.copy()
#     image_ = cv2.resize(image_, (width, height))
    image_ = image.copy()
    #获得待检测图片的尺寸
    height, width = image_.shape
    # 将模板resize至与图像一样大小
    template_img = cv2.resize(template_img, (width,
height))
    # 模板匹配，返回匹配得分
    result = cv2.matchTemplate(image_, template_img,
cv2.TM_CCOEFF)
    return result[0][0]
```

3. 选择得分最高的作为匹配到的字符；

```
def template_match_number(word_images):
    results = []
    for index,word_image in enumerate(word_images):
        best_score = []
        for eng_num_word_list in eng_num_words_list:
            score = []
            match_time = 0
            for eng_num_word in eng_num_word_list:
                result =
template_score(eng_num_word,word_image)
                score.append(result)
                match_time+=1
                if(match_time==100):
                    break
            best_score.append(max(score))
        i = best_score.index(max(best_score))
        # print(template[i])
        r = template[i]
        results.append(r)
    return results

# 对分割得到的字符逐一匹配
def template_matching(word_images):
    results = []
    for index,word_image in enumerate(word_images):
        if index==0:
            best_score = []

            for chinese_words in chinese_words_list:
                score = []
                match_time = 0
                for chinese_word in chinese_words:
                    result =
template_score(chinese_word,word_image)
                    score.append(result)
                    match_time+=1
                    if(match_time==100):
                        break
                best_score.append(max(score))
            i = best_score.index(max(best_score))
            # print(template[34+i])
```

```
            r = template[34+i]
            results.append(r)
            continue
        if index==1:
            best_score = []
            for eng_word_list in eng_words_list:
                score = []
                match_time = 0
                for eng_word in eng_word_list:
                    result =
template_score(eng_word,word_image)
                    match_time+=1
                    score.append(result)
                    if(match_time==100):
                        break
                best_score.append(max(score))
            i = best_score.index(max(best_score))
            # print(template[10+i])
            r = template[10+i]
            results.append(r)
            continue
        else:
            best_score = []
            for eng_num_word_list in eng_num_words_list:
                score = []
                match_time = 0
                for eng_num_word in eng_num_word_list:
                    result =
template_score(eng_num_word,word_image)
                    score.append(result)
                    match_time+=1
                    if(match_time==100):
                        break
                best_score.append(max(score))
            i = best_score.index(max(best_score))
            # print(template[i])
            r = template[i]
            results.append(r)
            continue
    return results
```

4. 将停车场编号与车牌编号进行匹配，得到车牌对应的车位编号；

```
for number in numbers:

        distance = 10000000.0
        match_number = ""
        for car in cars:
            xycar = np.array([float(car[0]),
float(car[1])])
            xynumber = np.array([float(number[0]),
float(number[1])])

            # Calculate Euclidean distance
            n_dis = np.linalg.norm(xycar - xynumber)

            if(n_dis < distance):
                distance = n_dis
                match_number = number[2]
            # 如果这个编号没有被匹配，则匹配
        if(nummp.get(match_number)==None):
            nummp.update({match_number:car})
            dismp.update({match_number: distance})
        else:
            if(distance < dismp.get(match_number)):
                dismp.update({match_number:distance})
                nummp.update({match_number:car})

    for number in numbers:
        if(nummp.get(number[2])==None):
            print(f"编号{number[2]}上没有停放车辆")
        else:
            print(f"编号{number[2]}上停着
{nummp.get(number[2])[2]}")
```

四、实验结果

```
在文件.\pics\0.jpg中
编号001上没有停放车辆
在文件.\pics\1.jpg中
编号111上没有停放车辆
编号222上停着京AF0236
在文件.\pics\2.jpg中
编号001上停着宁AF0236
在文件.\pics\3.jpg中
编号001上停着粤EAAZZI
编号002上停着粤EABC12
在文件.\pics\4.jpg中
编号001上停着粤EAAZZI
编号002上停着云SBCOZZ
在文件.\pics\5.jpg中
编号001上没有停放车辆
编号OO2上停着琼EB112C
在文件.\pics\6.jpg中
编号001上没有停放车辆
编号002上停着辽EBAA7Z
在文件.\pics\7.jpg中
编号001上没有停放车辆
编号002上没有停放车辆
在文件.\pics\8.jpg中
编号555上停着云SBC0ZZ
在文件.\pics\9.jpg中
编号001上停着粤EAAZZI
```

比较好地识别出车牌和停车场编号，但是对于车牌字符的识别效果不是很好，有时候会识别错误或者无法识别出停车场编号或车牌号，对于车牌字符的识别还有待改进。

五、总结

通过这个实验，我加深了对数字图像处理的理解，了解了基本的图像处理方法，使用了形态学操作，轮廓检测，模板匹配等方法，对于图像的处理有了更深的理解。我也学习了使用OpenCV，pyplot，numpy等库进行图像处理，学习了使用Python进行图像处理的方法。

六、参考文献

OpenCV文档

OpenCV-Python Tutorials

车牌识别参考