江西理工大学软件学院

《数字图像处理》期末大作业报告

2021—2022 学年第 2 学期

课程名称 数字图像处理

题　　目 Python 学号识别

专业班级 虚拟现实 191

姓　　名 皮寡

学　　号 5720192022

2022 年 5 月 22 日

目　录

1. 绪论

随着时代的发展，计算机技术越来越深入各行各业，为广大的用户提供了更为周到和便捷的服务。目前各行各业广泛使用计算机系统，其内容范围跨越了教育科研、文化事业、金融、商业、新闻出版、娱乐、体育等各个领域，其用户群十分庞大。因此，设计开发好一个专用程序对一个机构(或部门)的发展十分重要。近年来，随着用户要求的不断提高及计算机科学的迅速发展，特别是数据库技术的广泛应用，向用户提供的服务将越来越丰富，越来越人性化。

本学期经过约一学期对于数字图像处理的学习，在魏勋老师的带领下，我对于图像处理和分析了解颇深。我认为了解清楚图像处理和分析到底是一种怎么样的存在，是解决这门课程中一切问题的关键。 这种事实对本人来说意义重大。学习数字图像处理不仅仅是一个重大的事件，还可能会改变我的人生。

2. 环境搭建

2.1 硬件环境介绍

电脑型号 Shinelon Computer CNH5L 准系统电脑
操作系统 Windows 11 专业版 64 位(DirectX 12)
处理器 AMD Ryzen 5 3600 6-Core 六核
主板 Shinelon Computer CNH5L (AMD PCI 标准主机 CPU 桥)
内存 16 GB (镁光 DDR4 3200MHz)
主硬盘三星 MZVLB512HAJQ-00000 (512 GB /固态硬盘)
显卡 Nvidia GeForce RTX 2070 (8 GB /蓝天(CLEVO))

2.2 系统环境

版本　Windows 11 家庭中文版
版本　21H2
操作系统版本　22000.348

图 2.1　个人 Windows 规格

2.3 编程环境

Python 3.8.12

图 2.2　Python 版本

opencv4.5.4

图 2.3　opencv 版本

numpy 1.16.6

图 2.4　numpy 版本

scipy 1.7.3

图 2.5　scipy 版本

matplotlib 3.4.3

图 2.6　matplotlib 版本

2.2 Python 语言简介

Python 是一门简单、现代、面向对象和类型安全的编程语言，在 Python 之前，C 和 C++、Java 已经成为在软件的开发领域中使用最广泛的语言。然而对于很大一部分的应用来说，这些中级语言的实现过于复杂。

最重要的是，Python 使得程序员可以高效的开发程序，Python 易于扩展，可以使用 C 语言或 C++（或者其他可以通过 C 调用的语言）扩展新的功能和数据类型。Python 也可用于可定制化软件中的扩展程序语言。Python 丰富的标准库，提供了适用于各个主要系统平台的源码或机器码。

其特点有：

(1) Python 的底层是用 C 语言写的，很多标准库和第三方库也都是用 C 写的，运行速度非常快；

(2)可以把 Python 嵌入 C/C++程序，从而向程序用户提供脚本功能；

(3) Python 的扩展交互性很强；

(4) Python 对版本的更新的支持使得其使用更加方便。

3.程序概要设计

3.1 程序思路大致流程

图 3.1 代码设计思路图

3.2 程序流程图

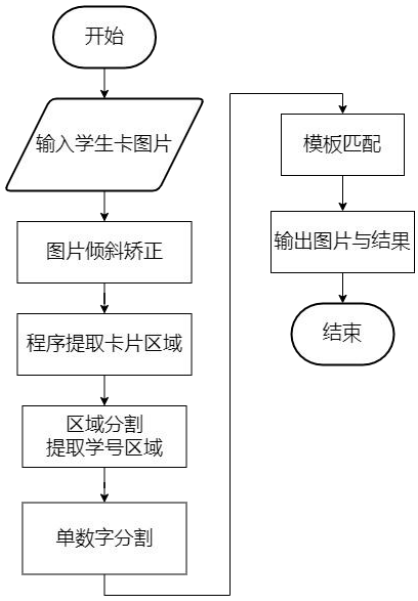

图 3.2　程序流程图

4.图像增强

4.1 图像倾斜矫正

输入图片，通过霍夫变换，监测线条将输入的图像进行校正。

图 4.1　输入图形示例

图 4.2　输入图形调整示例

代码如下：

```
#输入图片并进行调整
img = cv2.imread('H:/sample/sample10.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150, apertureSize=3)

# 霍夫变换进行校正
lines = cv2.HoughLines(edges, 1, np.pi / 180, 0)
rotate_angle = 0
for rho, theta in lines[0]:
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a * rho
    y0 = b * rho
    x1 = int(x0 + 1000 * (-b))
    y1 = int(y0 + 1000 * (a))
    x2 = int(x0 - 1000 * (-b))
    y2 = int(y0 - 1000 * (a))
    if x1 == x2 or y1 == y2:
        continue
    t = float(y2 - y1) / (x2 - x1)
    rotate_angle = math.degrees(math.atan(t))
```

```
        if rotate_angle > 45:
            rotate_angle = -90 + rotate_angle
        elif rotate_angle < -45:
            rotate_angle = 90 + rotate_angle

    image = ndimage.rotate(img,rotate_angle)
```

4.2 学生卡区域检测并裁剪

对之前矫正过的图片进行一系列处理，包括灰度处理、高斯平滑、blur 图像过滤、二值处理、边缘检测、腐蚀膨胀、轮廓检测、面积排序

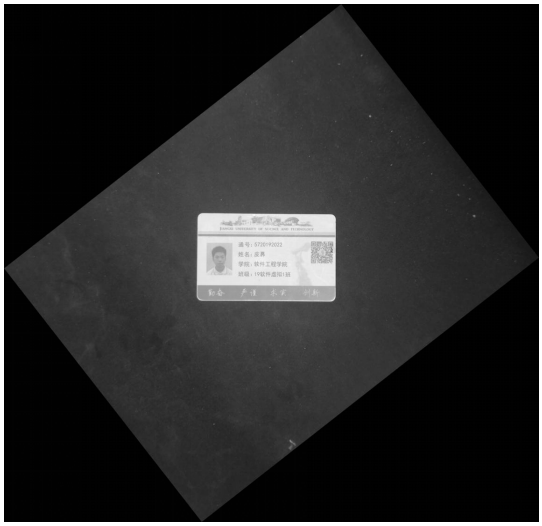

图 4.3　灰度处理示例

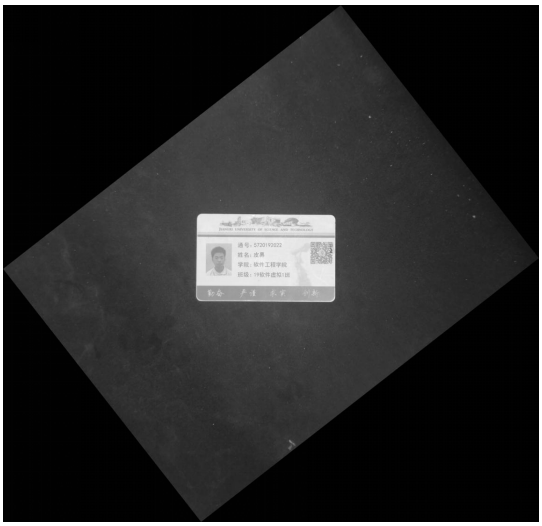

图 4.4　高斯平滑处理示例

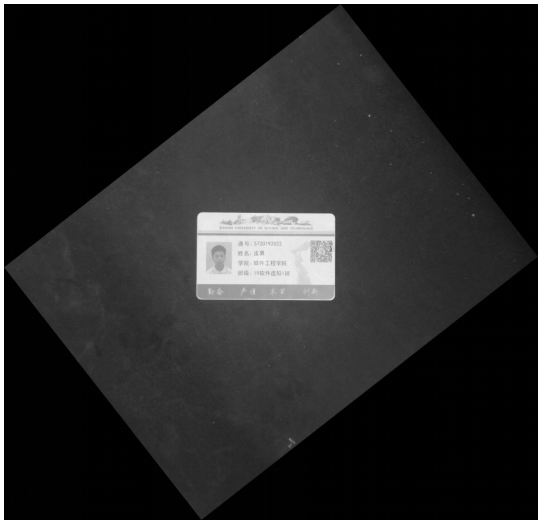

图 4.5　blur 图像过滤处理示例

图 4.6　二值处理示例

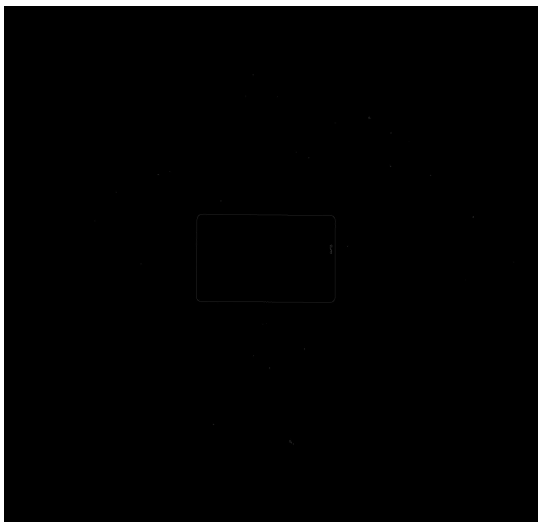

图 4.7　边缘检测处理示例

图 4.8　腐蚀膨胀处理示例

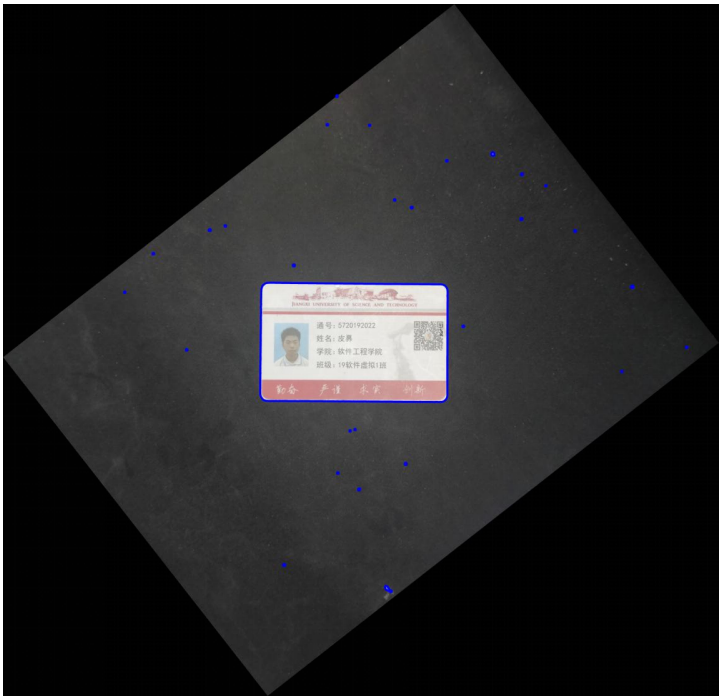

图 4.9　轮廓检测处理示例

代码如下：

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
#图像转灰度
gaosi =cv2.GaussianBlur(gray, (1,1), 0)
#高斯平滑
blur = cv2.medianBlur(gaosi, 7)
#图像过滤
```

```
threshold = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)[1]
#二值处理
canny = cv2.Canny(threshold, 100, 200)
#边缘检测
kernel = np.ones((3, 3), np.uint8)
dilate = cv2.dilate(canny, kernel, iterations=5)
#为了使边缘检测的边缘更加连贯，使用膨胀处理，对白色的边缘膨胀，即边缘线条变得更加粗一些。
contours, hierarchy = cv2.findContours(dilate, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
image_copy = image.copy()
res = cv2.drawContours(image_copy, contours, -1, (255, 0, 0), 20)
#使用 findContours 对边缘膨胀过的图片进行轮廓检测
contours = sorted(contours, key=cv2.contourArea, reverse=True)[0]
image_copy = image.copy()
res = cv2.drawContours(image_copy, contours, -1, (255, 0, 0), 20)
#经过对轮廓的面积排序，我们可以准确的提取出校园卡的轮廓。
```

4.3 提取一卡通区域

检测出一卡通区域后，对此区域进行裁剪，并进行长方形矫正（比较长与宽，将竖置图片横置）。

图 4.10　倾斜竖直图片示例

图 4.11　矫正裁剪图片示例

代码如下：

```
epsilon = 0.02 * cv2.arcLength(contours, True)

approx = cv2.approxPolyDP(contours, epsilon, True)
n = []
for x, y in zip(approx[:, 0, 0], approx[:, 0, 1]):
    n.append((x, y))
n = sorted(n)
sort_point = []
n_point1 = n[:2]
n_point1.sort(key=lambda x: x[1])
sort_point.extend(n_point1)
n_point2 = n[2:4]
n_point2.sort(key=lambda x: x[1])
n_point2.reverse()
sort_point.extend(n_point2)
p1 = np.array(sort_point, dtype=np.float32)
h = sort_point[1][1] - sort_point[0][1]
w = sort_point[2][0] - sort_point[1][0]
pts2 = np.array([[0, 0], [0, h], [w, h], [w, 0]], dtype=np.float32)
```

```
# 生成变换矩阵
M = cv2.getPerspectiveTransform(p1, pts2)
# 进行透视变换
dst = cv2.warpPerspective(image, M, (w, h))
# print(dst.shape)
#提取出轮廓的四个顶点，并按顺序进行排序，对所选的卡片图像进行校正处理

if w < h:
    dst = np.rot90(dst)
resize = cv2.resize(dst, (1084, 669), interpolation=cv2.INTER_AREA)

#图像宽高检测变换，以确定图像是否为正
temp_image = resize.copy()
```

5. 区域分割

5.1 整体区域分割

对学生卡进行处理，将学生卡上的区域进行分割，提取出学号区域。

图 5.1　区域划分图片示例

图 5.2　区域划分直线框选图片示例

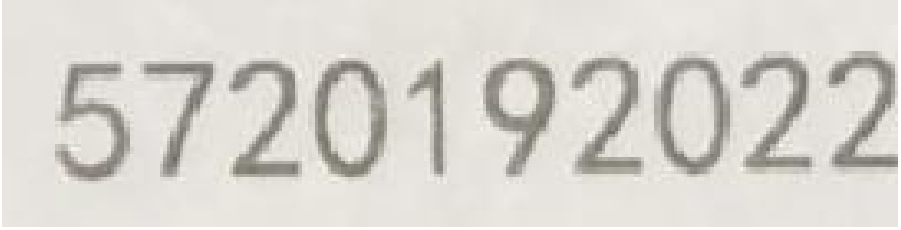

图 5.3　提取出的学号区域示例

代码如下：

```
gray = cv2.cvtColor(resize, cv2.COLOR_BGR2GRAY)
gaosi =cv2.GaussianBlur(gray, (1,1), 0)
threshold = cv2.threshold(gaosi, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
blur = cv2.medianBlur(threshold, 5)
kernel = np.ones((3, 3), np.uint8)
morph_open = cv2.morphologyEx(blur, cv2.MORPH_OPEN, kernel)
#将图像中需要的区域显现出来。
kernel = np.ones((7, 7), np.uint8)
dilate = cv2.dilate(morph_open, kernel, iterations=6)
contours, hierarchy = cv2.findContours(dilate, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
resize_copy = resize.copy()
res = cv2.drawContours(resize_copy, contours, -1, (255, 0, 0), 2)

labels = ['']
positions = []
data_areas = {}
resize_copy = resize.copy()
#长方形区域框选
for contour in contours:
    epsilon = 0.002 * cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, epsilon, True)
    x, y, w, h = cv2.boundingRect(approx)
    if h > 50 and x < 670:
        res = cv2.rectangle(resize_copy, (x, y), (x + w, y + h), (0, 255, 0), 2)
        area = gray[y:(y + h), x:(x + w)]
        blur = cv2.medianBlur(area, 3)
        data_area = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
        positions.append((x, y))
        data_areas['{}-{}'.format(x, y)] = data_area
crop_img = res[206:265,432:665]
```

5.2 单数字分割

对裁剪下来的区域进行调整，以便于令区域内的每个数字分割一块矩形区域，为后续的识别做准备。

图 5.4　学号区单数字分割示例

代码如下：

```
gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
blur = cv2.medianBlur(gray, 5)
ret1,gray=cv2.threshold(blur, 200, 255, cv2.THRESH_BINARY)
threshold = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

blur = cv2.medianBlur(threshold, 5)
canny1 = cv2.Canny(blur, 100, 150)
kernel = np.ones((3, 3), np.uint8)
morph_open = cv2.morphologyEx(threshold, cv2.MORPH_OPEN, kernel)
#将图像中需要的区域显现出来。
kernel = np.ones((7, 7), np.uint8)
dilate = cv2.dilate(morph_open, kernel, iterations=1)
contours, hierarchy = cv2.findContours(canny1, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

res1 = cv2.drawContours(crop_img, contours, -1, (255, 0, 0), 1)
print(len(contours))
for c in contours:
    x1,y1,w1,h1=cv2.boundingRect(c)
    caiji=cv2.rectangle(threshold,(x1,y1),(x1+w1,y1+h1),(255,255,255),1)
```

6. 模板采集与处理、识别

6.1 模板采集与处理

首先采集模板，采集字体字号为黑体，80 号，白字黑底，采集图如下：

图 6.1　采集模板图片示例

之后对模板进行处理，以便于进行单数字分割。

图 6.2　采集模板图片处理示例

代码如下：

```
test_img = cv2.imread('H:/sample/modetest.jpg')
graytest = cv2.cvtColor(test_img, cv2.COLOR_BGR2GRAY)
ret,graytest=cv2.threshold(graytest, 200, 255, cv2.THRESH_BINARY)
thresholdtest = cv2.threshold(graytest, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)[1]

blurtest = cv2.medianBlur(thresholdtest, 5)
cannytest = cv2.Canny(blurtest, 100, 150)
morphtest_open = cv2.morphologyEx(cannytest, cv2.MORPH_OPEN, kernel)
dilatetest = cv2.dilate(morphtest_open, kernel, iterations=1)
contourstest, hierarchytest = cv2.findContours(cannytest, cv2.RETR_EXTERNAL,
```

```
cv2.CHAIN_APPROX_SIMPLE)
    res2 = cv2.drawContours(test_img, contourstest, -1, (255, 0, 0), 1)

    print(len(contourstest))
    for c in contourstest:
        x1,y1,w1,h1=cv2.boundingRect(c)
        muban=cv2.rectangle(thresholdtest,(x1,y1),(x1+w1,y1+h1),(0,255,255),1)

    muban1=255-muban
```

6.2 模板匹配

首先构造一个函数，通过列表获取处理好的裁剪图片与处理好的模板图片数据中每个框的 X,Y,W,H,将这些数据存入列表中，再对这些数据根据 X 进行排序，之后将处理好的裁剪图片与处理好的模板图片数据进行模板匹配。

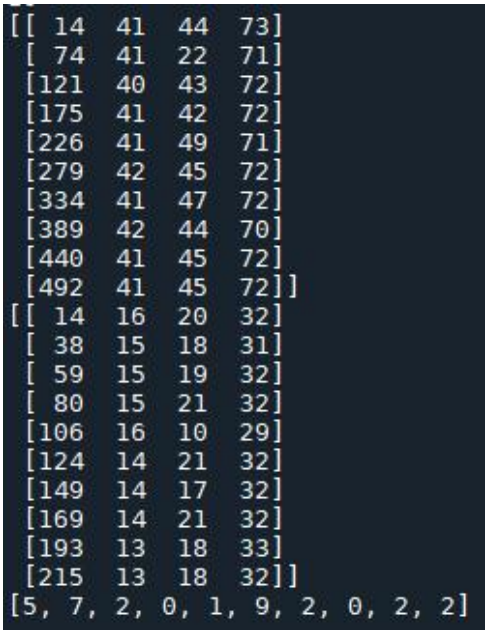

图 6.3　数据显示记录示例

最后将识别出的结果放置到图片上，输出：

图 6.4　输出图片示例

代码如下：

```
def sequence_contours(image,width,height):

    contours,hierarchy = cv2.findContours(image, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    n = len(contours)
    RectBoxes0 = np.ones((n,4),dtype=int)
    for i in range(n):
        RectBoxes0[i] = cv2.boundingRect(contours[i])

    RectBoxes = np.ones((n,4),dtype=int)
    for i in range(n):
        sequence = 0
        for j in range(n):
            if RectBoxes0[i][0]>RectBoxes0[j][0]:
                sequence = sequence + 1
        RectBoxes[sequence]=RectBoxes0[i]
    ImgBoxes = [[]for i in range(n)]
    for i in range(n):
            x,y,w,h = RectBoxes[i]
```

```
                ROI = image[y:y+h,x:x+w]
                ROI = cv2.resize(ROI,(width,height))
                thresh_val, ROI = cv2.threshold(ROI, 200, 255, cv2.THRESH_BINARY)
                ImgBoxes[i] = ROI

        return RectBoxes,ImgBoxes

RectBoxes_temp,ImgBoxes_temp = sequence_contours(graytest,120,200)
print(RectBoxes_temp)
cv2.imshow('ImgBoxes_temp[1]', ImgBoxes_temp[3])
cv2.waitKey()     #要加的两行代码
cv2.destroyAllWindows()
RectBoxes, ImgBoxes = sequence_contours(caiji, 120, 200)
print(RectBoxes)

result = []
for i in range(len(ImgBoxes)):
        score = np.zeros(len(ImgBoxes_temp),dtype=int)

        for j in range(len(ImgBoxes_temp)):

                score[j] = cv2.matchTemplate(ImgBoxes[i], ImgBoxes_temp[j],
cv2.TM_SQDIFF)

        min_val, max_val, min_indx, max_indx = cv2.minMaxLoc(score)
        result.append(min_indx[1])
print(result)

result = str(result)
im=res
cv2.putText(im,result,(451, 199),cv2.FONT_HERSHEY_SIMPLEX,0.75,(0,255,255),2)
#模板匹配
cv2.imshow("final", im)
cv2.waitKey()     #要加的两行代码
cv2.destroyAllWindows()
```

7. 算法测试与优化

7.1 不同个人图片测试

使用十张不同的图片来进行测试，检测效果，图片如下：

图 7.1　样本一示例

图 7.2　样本二示例

图 7.3　样本三示例

图 7.4　样本四示例

图 7.5　样本五示例

图 7.6　样本六示例

图 7.7　样本七示例

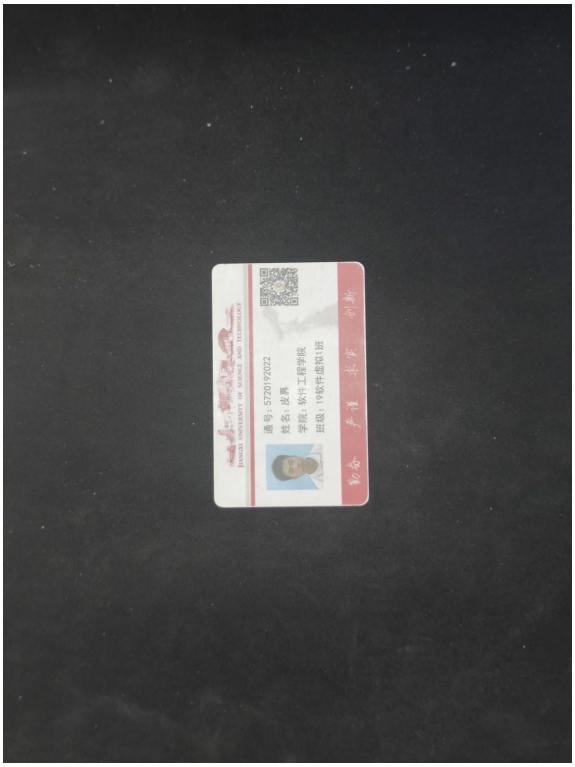

图 7.8　样本八示例

图 7.9　样本九示例

图 7.10　样本十示例

测试结果如下（测试 10 张图，共有 9 张正确检测出学号，这 10 张样本测试的正确率为 90%）：

图 7.11　结果一示例

图 7.12　结果二示例

图 7.13　结果三示例

图 7.14　结果四示例

图 7.15　结果五示例

图 7.16　结果六示例

图 7.17　结果七示例

图 7.18　结果八示例

图 7.19　结果九示例

图 7.20　结果十示例

7.2 算法优化

因为以前的课程曾经使用 winform 构建过完整的软件程序，所以现在程序的主要问题其一是无法像一般图形检测软件一样在框体内输入检测；其二是构建中存在一些重复性的代码，应该写进一个方法或一个函数，这样就能提高代码的精简性，其三是由于代码基本上只调用图形处理方面与数学运算方面的包与工具，在识别上不太准确，存在识别字符失误的情况，通过这段时间的学习，我发现可以通过调用百度或者阿里巴巴的 OCR 字符检测库通过大数据来进行识别，可以精简许多的操作，以上为算法优化的三个方向。

8. 总结

随着时代的发展，计算机技术越来越深入各行各业，信息的自动处理以及网络式的信息交互方式越来越被人们认可和应用。让计算机来处理各种各样的信息是我们软件工程专业学习的重点，用软件来处理学生卡信息相比传统人工抄录更加节约资源，更加省时、便捷。

就我个人来说，图形图像处理对图像识别学习的意义，不能不说非常重大。 带着这些问题，我们才开始根据要求对程序进行开发。 学生卡号程序开发到底需要如何做到，使用 Python 程序开发的发生，又会如何产生。 那么，我们不得不面对这些问题。 在安排各种各样的要求与代码之间的不同构造， 一般来讲，我们都必须务必慎重的考虑考虑。 培根曾经提到过，合理安排时间，就等于节约时间。我在空余时间钻研代码，钻研图形图像的各种知识。 我们都知道，只要有意义，那么就必须慎重考虑。 俾斯麦曾经说过，失败是坚忍的最后考验。这句话语虽然很短，但令我感受颇深，在这两周的时间里，我经历了许多的代码报错，但还是成功完成了。 这次的大作业既让我提升了自己的 Python 代码编程水平，也锻炼自己处理问题的能力，实在是让人印象深刻。