**电子科技大学成都学院**

**实验报告**

课程名称 数字图像处理

分 院 计算机学院

年 级 2020

专　　业 人工智能

班 级 一班

任课教师 田军 陈东祥

学生姓名

学　　号

2021年12月制

摘 要

随着科技的发展，电子与计算机技术的进步，答题卡的出现大大减轻教学工作者们批改试卷的工作量。答题卡是光标阅读机输入信息的载体，是配套光标阅读机的各种信息录入表格的总称。答题卡将用户需要的信息转化为可选择的选项，供用户涂写。OMR是用光学扫描的方法来识别按一定格式印刷或书写的标记，并将其转换为计算机能接受的电信号的设备，并根据信息点的涂与未涂和格式文件设置将信息还原。因此，如何将答题卡填涂的黑色区域识别出来并使用计算机进行处理是极为关键的。本论文探索了有效识别答题卡的方法，以opencv为工具，基于数字图像处理技术对答题卡填涂区域进行了识别，并对识别的结果进行了处理，得到了结果。本论文利用Hough变换的直线检测技术检测图像的倾斜度，判断图像是否倾斜，对存在倾斜的图像进行旋转校正。最终实现答题卡答案的定位和检测。论文使用像素统计方法进行识别，利用黑白颜色灰度值的巨大差异对二值图像的灰度值进行累加并进行阈值判定，识别错误效率极低，能够准确的识别答题卡的涂卡标记。

关键词：Houg，变换，答题卡识别，opencv.

Abstract

With the development of science and technology and the progress of electronic and computer technology, the emergence of answer cards greatly reduces the workload of teaching workers in correcting test papers. Answer card is the carrier of input information of cursor reader and the general name of various information input forms supporting cursor reader. The answer sheet converts the information required by users into optional options for users to scribble. OMR is a device that uses the optical scanning method to identify the marks printed or written in a certain format and convert them into electrical signals acceptable to the computer, and restore the information according to the coated and uncoated information points and format file settings. Therefore, how to recognize the black area filled in the answer sheet and process it by computer is very key. This paper explores the method of effectively identifying the answer card. Using OpenCV as a tool, the filling area of the answer card is identified based on digital image processing technology, and the recognition results are processed and the results are obtained. In this paper, the line detection technology of Hough transform is used to detect the inclination of the image, judge whether the image is inclined, and rotate and correct the inclined image. Finally, the positioning and detection of the answers in the answer sheet are realized. In this paper, the pixel statistical method is used for recognition. The gray value of the binary image is accumulated and the threshold value is determined by using the huge difference of the gray value of black and white color. The recognition error efficiency is very low, and the coated mark of the answer card can be recognized accurately.

Key words: Houg, transformation, answer sheet recognition, OpenCV

目录

[第一章：引言 1](#_Toc5170_WPSOffice_Level1)

[1.1课题意义 1](#_Toc982_WPSOffice_Level2)

[1.2发展历史 1](#_Toc25220_WPSOffice_Level2)

[1.3 识别难点 2](#_Toc13685_WPSOffice_Level2)

[1.4 现有的答题卡识别案例参考 2](#_Toc31141_WPSOffice_Level2)

[1.5答题卡识别案例问题 4](#_Toc6204_WPSOffice_Level2)

[1.6 简单介绍算法需求 4](#_Toc21615_WPSOffice_Level2)

[第二章：方法 4](#_Toc982_WPSOffice_Level1)

[2.1整体流程 5](#_Toc2971_WPSOffice_Level2)

[2.2 所提方法 5](#_Toc18362_WPSOffice_Level2)

[2.2.1canny边缘检测算子 5](#_Toc982_WPSOffice_Level3)

[2.2.2 高斯滤波 5](#_Toc25220_WPSOffice_Level3)

[2.2.3透视变换 6](#_Toc13685_WPSOffice_Level3)

[2.2.4 Hough变换 6](#_Toc31141_WPSOffice_Level3)

[2.2.5Sobel 算子 6](#_Toc6204_WPSOffice_Level3)

[第三章:实验设计 6](#_Toc25220_WPSOffice_Level1)

[3.1预处理、轮廓检测 7](#_Toc16295_WPSOffice_Level2)

[3.2 轮廓排序，透视变换 7](#_Toc8054_WPSOffice_Level2)

[3.3寻找圆圈轮廓 10](#_Toc1012_WPSOffice_Level2)

[3.4 输出每个轮廓，对比答案 11](#_Toc31401_WPSOffice_Level2)

[第四章:结果展现 13](#_Toc13685_WPSOffice_Level1)

[第五章:结语 18](#_Toc31141_WPSOffice_Level1)

[致谢 18](#_Toc6204_WPSOffice_Level1)

第一章：引言

# 1.1课题意义

答题卡识别系统在考试领域的应用， 对答题卡识别技术的研究保证**了阅卷的质量，提高了阅卷的速度，减少不必要 的错误，坚持了考试的原则，弥补了传统方式中人工阅卷存在的不足，如阅卷 质量存在的问题，阅卷速度的效率低等弊端。 通过图像识别技术的使用，在阅 卷过程中把客观题交给计算机，留给教师更多教学管理与思考教学问题的时间。 同时，学生**在普通的考试当中，充分体验了考试的模式，正规考试时也可以从 容面对。

保证了阅卷的质量，提高了阅卷的速度。减少了老师的工作量，减少了不仔细造成的改卷错误。

# 1.2发展历史

随着计算机技术的飞速发展，阅卷系统也得到了成熟的发展。自动阅卷系统涉及多个学科领域，其中[数字图像处理](https://www.zhihu.com/search?q=%E6%95%B0%E5%AD%97%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b)是这个学科中最关键最重要的一部分，它是国内外教育事业发展领域的研究热点之一。

国外是率先采用计算机技术自动阅卷系统，1966 年在美国的[杜克大学](https://www.zhihu.com/search?q=%E6%9D%9C%E5%85%8B%E5%A4%A7%E5%AD%A6&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":397441281}" \t "https://zhuanlan.zhihu.com/p/_blank)，Ellis Page 等团队经过努力研发了第一个自动评分的系统叫做 PEG(Project Essay Grade)[1]，它主要是针对于文章的内在特征进行句子意思的分析，将文章的特征进行量化，采用人工评估得分的方式给出结果，从这些结果中找出一系列的特征进行训练，得出一个与之相关的分析性值，用这个值对待批阅的文章进行批阅。在此后还出现了 LSA(Latent Semantic Analysis)系统[2]，这个系统最重要的功能就是将每一篇文章看作是一个在空间的向量，向量的每一行表示文档的特征，而每一列表示文档的向量，每一个文本都会产生一个[矩阵](https://www.zhihu.com/search?q=%E7%9F%A9%E9%98%B5&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":397441281}" \t "https://zhuanlan.zhihu.com/p/_blank)。再用余弦的方法计算文本向量与将要批阅的文本向量的相似度，最后给出评价。1970 年，欧美几个发达国家开始研制光标阅读机，在答题卡识别系统方面研究比较成熟的系统是 Remark Office 软件[2]，填涂的区域是[圆形](https://www.zhihu.com/search?q=%E5%9C%86%E5%BD%A2&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":397441281}" \t "https://zhuanlan.zhihu.com/p/_blank)，这些系统具有较好的经济价值，然而当这些系统投入市场后，得到了大多数使用者的青睐，然而在相关技术方面公布的资料都很少，限制了研究者以及开发者其研究和开发。

在 80 年代初期，我国才开始对自动阅卷系统进行研究。在 1988 年的高考的阅卷中进行了试点，取得了不错的成绩。这个阅卷系统就是[山东大学](https://www.zhihu.com/search?q=%E5%B1%B1%E4%B8%9C%E5%A4%A7%E5%AD%A6&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":397441281}" \t "https://zhuanlan.zhihu.com/p/_blank)和重庆大学首先研发的。此后 1990 年，国家教育部的考试中心把这项阅卷技术列为教育发展的重要项目，并大力鼓励各科研单位进行研究这个阅卷系统，1992 年已有多个科研单位研制的自动阅卷系统通过国家审核，并用于各种标准化考试中。由于随着计算机科学技术的高速发展和数字图像处理技术的成熟，近年来人们开始研究基于图像处理的答题卡阅卷系统，处理效果也相对比较好，已经有不少公司开发的阅卷系统投入了市场。

# 1.3 识别难点

在答题卡识别的过程中有一定的错误和多选漏选的情况因此对于答题卡精准的识别需要经过不断的训练才能达到我们需要的目的

# 1.4 现有的答题卡识别案例参考

以matlab为工具利用数字图像处理技术对答题卡进行了识别，并对识别结果进行了处理。注意到答题卡在采集图像的过程中由于各种原因可能会产生图像倾斜、水平或垂直错位，要进行正确识别首先必须对其进行校正，再依据像素检索技术进行识别。

答题卡在各种考试中，作为一种更方便的答题判题的工具，得到了非常广泛的应用

Hough变换主要是用来检测直线的。是答题卡判别要依靠的一个重要技术

在平面空间坐标中，表示一条直线有很多种方法，最常见的就是直线的斜截表达式，即：

这种表达式是在x-y空间中的，也就是x为横轴y为纵轴的情况下。那么如果把k和b，即直线的斜率和截距作为横轴和纵轴，直线会变成什么样呢？因为一条在x-y空间中的直线，他的斜率和截距都是定值，所以在k-b空间中他表现为一个点。而在x-y空间中的一个点，我们可以把它视为无数条通过它的直线的集合，这些直线过定点但斜率和截距可变，所以在k-b空间表现为一条直线。

这种方法是存在问题的，在x-y空间中，直线的斜率和截距不同时的情况下都可以趋近于无穷大，这样的直线再转换到k-b空间中，就难以直观表现了。所以这里要采用另外一种方式来表现x-y空间中的直线。

这种方式有点类似于极坐标，通过原点向一条直线做垂线，通过这条直线的长度和角度（从x轴正向逆时针旋转），可以唯一的表示这条直线。这里将直线的长度记为ρ，直线的角度记为θ。就可以得到这样的表达式：

（自己画图推演很重要）

这样的话，x-y空间中的直线对应到ρ-θ空间，就变成了一个点，那么x-y空间中的一个点呢？回忆之前定义一个点的方法，是所有经过该点的直线的集合，这些直线在ρ-θ空间中，就表现为正弦曲线。

（此处可以通过数学证明来理解，也不复杂，画图再演算便可推得，但是凭空间想象也可以理解，一条直线以直线上某一定点旋转一周，ρ和θ始终跟随直线变化，便可看到ρ和θ之间的正弦关系）

了解了Hough变换的原理之后，再回到图像上，因为Hough是针对点，而不是针对值，所以将一幅图像送去Hough变换之前，应该转换为二值图像。我们将二值图像上的每一个点都进行Hough变换，就能得到一个由多个正弦曲线组成的美妙图像。而这些正弦曲线的交点又是一个很重要的信息，两个正弦曲线的交点，代表有一组相同的ρ和θ同时满足这两个点，也就意味着在x-y空间中，有一条直线经过了这两个点。越多的正弦曲线相交于一点，代表着在x-y空间中有一条直线贯穿所有这些点，也就是说，这是一条x-y空间中的直线的可能性就越大。

# 1.5答题卡识别案例问题

1.答题卡区域的分割问题：进行答题卡识别得先把答题卡区域和环境区域分割出来。

2.答题卡纸张背景和答案的分离问题：我们需要的只有答案的区域，因此需要解决答案和答题卡背景的分割问题。

3.轮廓的筛选问题：筛选出需要的轮廓，排除那些不需要的轮廓信息。

4.轮廓的排序和定位问题：如何对轮廓进行行和列的定位。

5.检测答题者所选择的选项：检测漏选、多选的情况。

# 1.6 简单介绍算法需求

使用Hough变换检测出答题卡边缘直线，调用lines函数的参数得到倾斜角度并进行旋转变换从而消除答题卡倾斜状态。

将扫描部分分为7个扫描区域，分别使用像素检索技术，对各个检索区域内的灰度值进行累加，并对累加和进行阈值判定，从而得出检索区域颜色，完成判定。

第二章：方法

# 2.1整体流程

  读取图片

  图片转化为灰度图

 图片设定阈值

开运算(先腐蚀，后膨胀)

  指定答题区域

 找到涂选框

  根据涂选框的坐标确定所涂选的选项及题号

# 2.2 所提方法

## 2.2.1canny[边缘检测](https://so.csdn.net/so/search?from=pc_blog_highlight&q=%E8%BE%B9%E7%BC%98%E6%A3%80%E6%B5%8B" \t "https://blog.csdn.net/qiao_lili/article/details/_blank)算子

Canny边缘检测基本原理

(1)图象边缘检测必须满足两个条件：一能有效地抑制噪声；二必须尽量精确确定边缘的位置。

(2)根据对信噪比与定位乘积进行测度，得到最优化逼近算子。这就是Canny边缘检测算子。

(3)类似与Marr（LoG）边缘检测方法，也属于先平滑后求导数的方法。

2.Canny边缘检测算法：

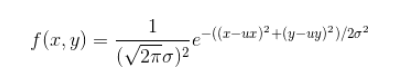
step1:用高斯滤波器平滑图象；

step2:用一阶偏导的有限差分来计算梯度的幅值和方向；

step3:对梯度幅值进行非极大值抑制；

step4:用双阈值算法检测和连接边缘。

## 2.2.2 高斯滤波

高斯滤波是一种线性平滑滤波器，对于服从正态分布的噪声有很好的抑制作用。在实际场景中，我们通常会假定图像包含的噪声为高斯白噪声，所以在许多实际应用的预处理部分，都会采用高斯滤波抑制噪声。

## 2.2.3透视变换

透视变换(Perspective Transformation)是指利用透视中心、像点、目标点三点共线的条件，按透视旋转定律使承影面(透视面)绕迹线(透视轴)旋转某一角度，破坏原有的投影光线束，仍能保持承影面上投影几何图形不变的变换。简而言之，就是将一个平面通过一个投影矩阵投影到指定平面上。

## 2.2.4 Hough变换

Hough变换的基本原理是将影像空间中的曲线(包括直线)变换到参数空间中，通过检测参数空间中的极值点，确定出该曲线的描述参数，从而提取影像中的规则曲线。

## 2.2.5Sobel 算子

Sobel算子是一种用于边缘检测的离散微分算子，它结合了高斯平滑和微分求导。该算子用于计算图像明暗程度近似值，根据图像边缘旁边明暗程度把该区域内超过某个数的特定点记为边缘。Sobel算子在Prewitt算子的基础上增加了权重的概念，认为相邻点的距离远近对当前像素点的影响是不同的，距离越近的像素点对应当前像素的影响越大，从而实现图像锐化并突出边缘轮廓

第三章:实验设计

# 3.1预处理、轮廓检测

import cv2

import numpy as np

# 正确答案

ANSWER\_KEY = {0: 1, 1: 4, 2: 0, 3: 3, 4: 1}

def cv\_show(name,img):

cv2.imshow(name, img)在窗口中显示图像， 第一个参数是一个窗口名称（也就是我们对话框的名称），它是一个字符串类型。第二个参数是我们的图像

cv2.waitKey(0)等待一个键盘的输入（因为我们创建的图片窗口如果没有这个函数的话会闪一下就消失了，所以如果需要让它持久输出，我们可以使用该函数）。

cv2.destroyAllWindows()销毁我们创建的所有窗口

# 读取输入

image = cv2.imread("test\_01.png")读取图片

contours\_img = image.copy()复制图像

### gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)将BGR格式转换成灰度图片

blurred = cv2.GaussianBlur(gray, (5, 5), 0)高斯滤波处理

cv\_show('blurred',blurred)输出灰度图进行高斯滤波处理后的图片

edged = cv2.Canny(blurred, 75, 200)使用函数 cv2.Canny()获取图像的边缘，进行边缘处理

cv\_show('edged',edged)输出边缘处理后的图像

# 轮廓检测

cnts = cv2.findContours(edged.copy(), cv2.RETR\_EXTERNAL,\

cv2.CHAIN\_APPROX\_SIMPLE)进行轮廓检测.使用edged图片的复制，因为findcontours函数会“原地”修改输入的图像

cv2.drawContours(contours\_img,cnts,-1,(0,0,255),3) 在图像（contours\_img）上绘制轮廓，**-**1，绘制其中的所有轮廓

cv\_show('contours\_img',contours\_img)输出绘制轮廓后的图像

# 3.2 轮廓排序，透视变换

def order\_points(pts):

# 一共4个坐标点

rect = np.zeros((4, 2), dtype = "float32")返回来一个给定形状和类型的用0填充的数组

# 按顺序找到对应坐标0123分别是 左上，右上，右下，左下

# 计算左上，右下

s = pts.sum(axis = 1)将一个矩阵的每一行向量相加

rect[0] = pts[np.argmin(s)]给出水平方向最小值的下标

rect[2] = pts[np.argmax(s)]给出水平方向最大值的下标

# 计算右上和左下

diff = np.diff(pts, axis = 1)执行的是后一个元素减去前一个元素

rect[1] = pts[np.argmin(diff)]给出水平方向最小值的下标

rect[3] = pts[np.argmax(diff)]给出水平方向最大值的下标

return rect

def four\_point\_transform(image, pts):

# 获取输入坐标点

rect = order\_points(pts)

(tl, tr, br, bl) = rect

# 计算输入的w和h值

widthA = np.sqrt(((br[0] - bl[0]) \*\* 2) + ((br[1] - bl[1]) \*\* 2))

widthB = np.sqrt(((tr[0] - tl[0]) \*\* 2) + ((tr[1] - tl[1]) \*\* 2))

maxWidth = max(int(widthA), int(widthB))

heightA = np.sqrt(((tr[0] - br[0]) \*\* 2) + ((tr[1] - br[1]) \*\* 2))

heightB = np.sqrt(((tl[0] - bl[0]) \*\* 2) + ((tl[1] - bl[1]) \*\* 2))

maxHeight = max(int(heightA), int(heightB))轮廓排序步骤

# 变换后对应坐标位置

dst = np.array([

[0, 0],

[maxWidth - 1, 0],

[maxWidth - 1, maxHeight - 1],

[0, maxHeight - 1]], dtype = "float32")轮廓排序步骤

# 计算变换矩阵

M = cv2.getPerspectiveTransform(rect, dst)

warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))轮廓排序步骤

# 返回变换后结果

return warped

# 确保检测到了

docCnt = None

if len(cnts) > 0:

# 根据轮廓大小进行排序

cnts = sorted(cnts, key=cv2.contourArea, reverse=True)

# 遍历每一个轮廓

for c in cnts:

# 近似

peri = cv2.arcLength(c, True) 计算轮廓的周长

approx = cv2.approxPolyDP(c, 0.02 \* peri, True)

 用于获得轮廓的近似值

# 准备做透视变换

if len(approx) == 4:

docCnt = approx

break

# 执行透视变换

warped = four\_point\_transform(gray, docCnt.reshape(4, 2))

cv\_show('warped',warped)

# 3.3寻找圆圈轮廓

def sort\_contours(cnts, method="left-to-right"):

reverse = False

i = 0

if method == "right-to-left" or method == "bottom-to-top":

reverse = True

if method == "top-to-bottom" or method == "bottom-to-top":

i = 1

boundingBoxes = [cv2.boundingRect(c) for c in cnts]获取轮廓点，分别是x，y，w，h

(cnts, boundingBoxes) = zip(\*sorted(zip(cnts, boundingBoxes),

key=lambda b: b[1][i], reverse=reverse))

return cnts, boundingBoxes

# Otsu's 阈值处理

thresh = cv2.threshold(warped, 0, 255,

cv2.THRESH\_BINARY\_INV | cv2.THRESH\_OTSU)[1] 阈值处理

cv\_show('thresh',thresh)输出阈值处理后的图像

thresh\_Contours = thresh.copy()复制阈值处理后的图像

# 找到每一个圆圈轮廓

cnts = cv2.findContours(thresh.copy(), cv2.RETR\_EXTERNAL,

# cv2.CHAIN\_APPROX\_SIMPLE)[0]轮廓检测

# cv2.drawContours(thresh\_Contours,cnts,-1,(0,0,255),3) 轮廓绘制

cv\_show('thresh\_Contours',thresh\_Contours)输出轮廓绘制后的图像

questionCnts = []

# 3.4 输出每个轮廓，对比答案

# 遍历

for c in cnts:

# 计算比例和大小

(x, y, w, h) = cv2.boundingRect(c)返回x,y,w,h的值

ar = w / float(h)

# 根据实际情况指定标准

if w >= 20 and h >= 20 and ar >= 0.9 and ar <= 1.1:

questionCnts.append(c)计算比对答案标准

# 按照从上到下进行排序

questionCnts = sort\_contours(questionCnts,

method="top-to-bottom")[0]

cv2.drawContours(warped, questionCnts, 1,(0,255,255),2)轮廓绘制

cv\_show("warp",warped)输出答案比对轮廓绘制后的图片

correct = 0

# 每排有5个选项

for (q, i) in enumerate(np.arange(0, len(questionCnts), 5)):

# 排序

cnts = sort\_contours(questionCnts[i:i + 5])[0]

bubbled = None

# 遍历每一个结果

for (j, c) in enumerate(cnts):

# 使用mask来判断结果

mask = np.zeros(thresh.shape, dtype="uint8")

cv2.drawContours(mask, [c], -1, 255, -1) #-1表示填充。绘制轮廓

cv\_show('mask',mask)输出绘制轮廓后的图

# 通过计算非零点数量来算是否选择这个答案

mask = cv2.bitwise\_and(thresh, thresh, mask=mask)

total = cv2.countNonZero(mask)

# 通过阈值判断

if bubbled is None or total > bubbled[0]:

bubbled = (total, j)判断答案正确

# 对比正确答案

color = (0, 0, 255)

k = ANSWER\_KEY[q]

# 判断正确

if k == bubbled[1]:

color = (0, 255, 0)

correct += 1

# 绘图

cv2.drawContours(warped, [cnts[k]], -1, color, 3)对判断正确的答案进行轮廓绘制

score = (correct / 5.0) \* 100计算得分

print("[INFO] score: {:.2f}%".format(score))输出得分

cv2.putText(warped, "{:.2f}%".format(score), (10, 30),在图像上显示文字（得分）

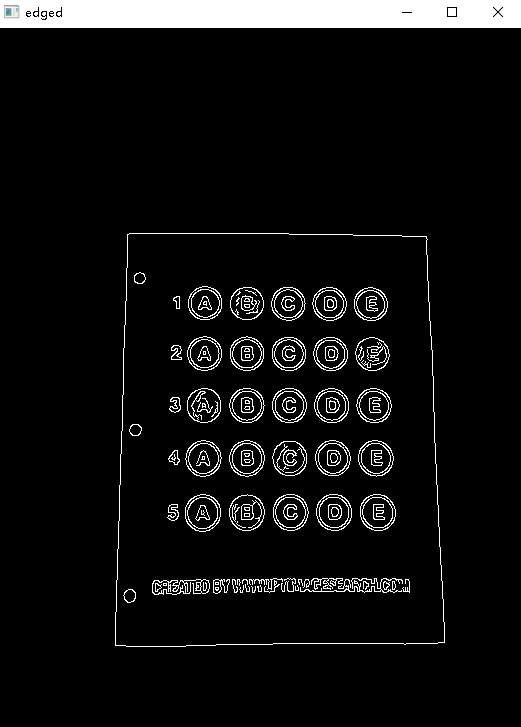
cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 0, 255), 2)字体设置

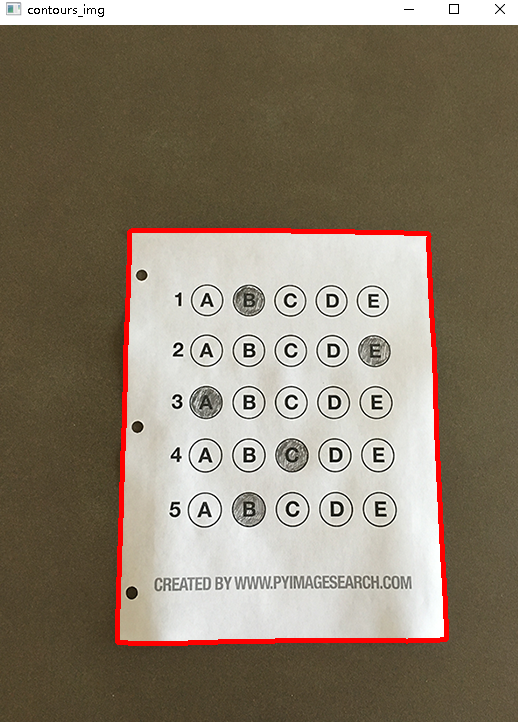
cv2.imshow("Original", image)输出图像

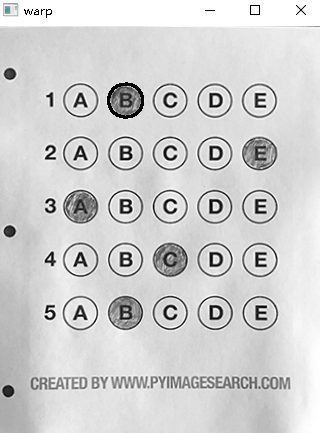
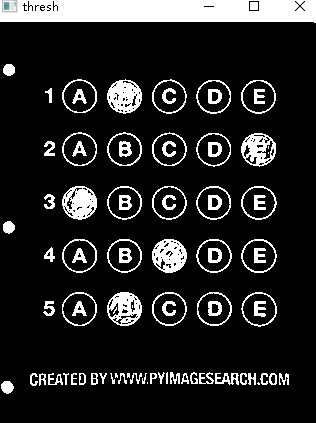
cv2.imshow("Exam", warped)输出图像

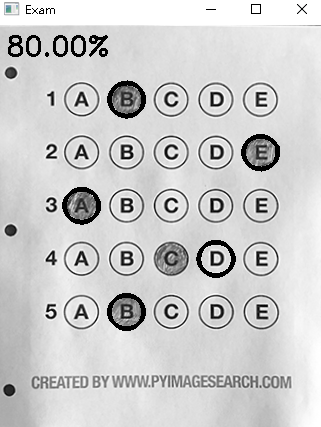
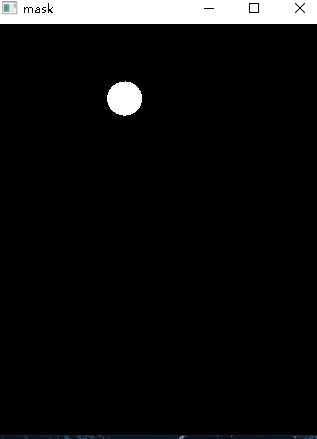
cv2.waitKey(0)

第四章:结果展现









第五章:结语

通过这次的学习和对实验的探讨和研究，我们学到了很多关于数字图像处理的很多类容，比如各种各样的算法和变换等等，使我们收益匪浅，也让我们明白了团队协助的重要性，每一个团队中的每一个人都有至关重要的作用，每一个人都有自己的分工，在自己的位置上发挥着自己的光芒，同时在这次学习探讨中我们学习了论文怎么写，还有ppt如何精美的制作，和对于思维导图的创建十分重要，总之这次的学习类容让我们收获颇丰。

致谢

禾路的博客园：

<https://www.cnblogs.com/jsxyhelu/p/9790979.html>

opencv convertTo用法：

<https://blog.csdn.net/qq_22764813/article/details/52135686>

vector<Point> 到 vector<Point2f>的类型转化：

<https://stackoverflow.com/questions/7386210/convert-opencv-2-vectorpoint2i-to-vectorpoint2f>

opencv 中的Rect类：

<https://blog.csdn.net/qq_30214939/article/details/65648273>

opencv中copyTo的应用：

<https://www.cnblogs.com/phoenixdsg/p/8420716.html>

开辟二维的vector矢量：

<https://blog.csdn.net/zchlww/article/details/44678757>

<https://blog.csdn.net/qiao_lili/article/details/83176480>

<https://blog.csdn.net/w11599/article/details/118495790?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_title~default-0.essearch_pc_relevant&spm=1001.2101.3001.4242.1>

https://blog.csdn.net/weixin\_39875161/article/details/103050272

<https://blog.csdn.net/Q1321814823/article/details/114466228>