**广东工业大学研究生课程考试试卷封面**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学院： 自动化学院** | |  |  | | --- | --- | | **题号** | **分数** | | **1** |  | | **2** |  | | **3** |  | | **4** |  | | **5** |  | | **6** |  | | **7** |  | | **8** |  | | **合计** |  | | **平时成绩** |  | | **总评成绩** |  |   **开课单位：自动化** |
| **专业： 控制工程** |
| **姓名： 邹恒** |
| **学号： 211904351** |
| **考试科目:机器视觉检测与识别** |
| **学生类别：硕士**  **考试时间:第11周星期 二 (2020 年4月2日)** |
| **开课学期：2020 年春季** | **任课教师：刘焕军** |
|  |  |
| **说明：**1.以上左栏学生填写,右栏老师填写;  2.学位课考试的答题均写在答题纸上,考查课若无课堂考试可不用答题纸，但仍应填写此页封面;  3.平时成绩根据任课教师所写的课程教学大纲规定评定；  4.任课教师评完分后从网上录入成绩，再打印成绩单一式两份连同试卷、答题纸交教务员保管（保管四年）。 | |

目录

[一、 题目说明 2](#_Toc44018298)

[二、 车牌定位 3](#_Toc44018299)

[(一) 灰度化处理 4](#_Toc44018300)

[(二) 高斯滤波 4](#_Toc44018301)

[(三) 形态学处理 4](#_Toc44018302)

[(四) 边缘检测 5](#_Toc44018303)

[(五) 轮廓查找 6](#_Toc44018304)

[(六) 轮廓拟合 6](#_Toc44018305)

[(七) 轮廓筛选 6](#_Toc44018306)

[(八) SVM分类 6](#_Toc44018307)

[(九) 仿射变换 7](#_Toc44018308)

[三、 字符分割 9](#_Toc44018309)

[(一) 去除冗余的空间 9](#_Toc44018310)

[(二) 字符分割 9](#_Toc44018311)

[四、 字符识别 10](#_Toc44018312)

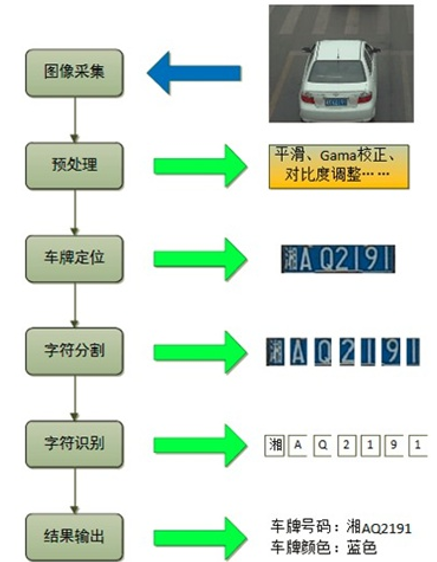
[(一) 数据集的准备 10](#_Toc44018313)

[(二) 训练与检测 11](#_Toc44018314)

[五、 总结与感想 11](#_Toc44018315)

# 题目说明

本报告的题目为车牌识别，设计语言为python3.6，设计平台为PyCharm，opencv版本为3.4.0.12，scikit-learn版本为0.22.1。处理的原始图片来自于中科大整理的数据集CCPD。车牌识别是基于图像分割和图像识别理论，对含有车牌识别车辆号牌的图像进行分析处理，从而确定车牌在图像中的位置，并进一步提取和识别出文本字符。车牌识别过程包括图像采集、预处理、车牌定位、字符分割、字符识别、结果输出等一系列算法运算。大致的处理流程如下图所示：



# 车牌定位

车牌定位就是把车牌区域完整的从一副具有复杂背景的车辆图像中分割出来，本报告借用开源库opencv-python对图片进行相应的预处理操作，主要包括滤波、形态学处理、边缘检测、阈值分割、支持向量机分类等方法，完成了对车牌的定位。主要流程如下：



## 灰度化处理

其中**灰度化处理**主要是将原始三通道彩色图像转换为单通道的灰度图，方便后续的处理 。通过cv2.cvtColor实现。结果如下：



## 高斯滤波

高斯滤波是一种线性平滑滤波，主要是为了消除图像噪声。为了尽可能减少噪声对边缘检测结果的影响，所以必须滤除噪声以防止由噪声引起的错误检测。利用高斯核与图像进行卷积运算，得到的结果就是滤波的结果。高斯卷积核大小的选择将影响Canny检测器的性能。尺寸越大，检测器对噪声的敏感度越低，但是边缘检测的定位误差也将略有增加。高斯滤波通过调用函数cv2.GaussianBlur实现。得到如下结果：



## 形态学处理

形态学处理包括腐蚀与膨胀操作，灰度形态学中的腐蚀就是类似卷积的一种操作，设结构元素B覆盖住的图像A的区域记为P，用P减去结构元素B形成的小矩形，取其中最小值赋到对应原点的位置即可。灰度形态学中的膨胀就是类似卷积的一种操作，用P加上B，然后取这个区域中的最大值赋值给结构元素B的原点所对应的位置。腐蚀和膨胀分别通过cv2.erode 与cv2.dilate两个函数实现。形态学处理结果如下：

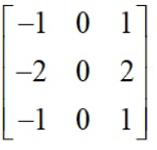
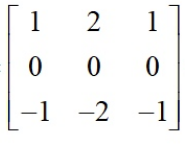


## 边缘检测

在阈值分割之后就可以进行边缘检测了，Canny边缘检测算法可以分为以下4个步骤：

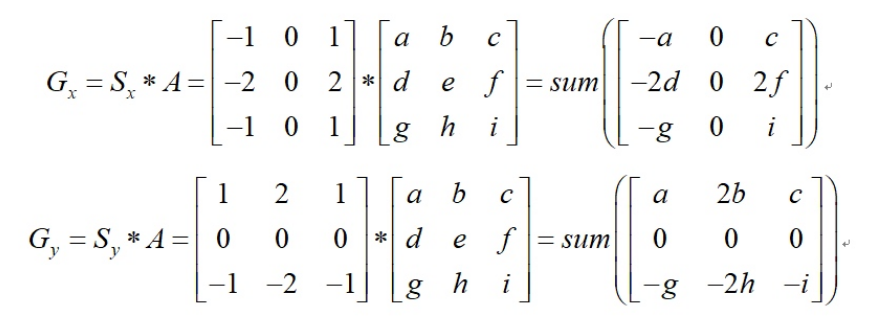
**1、计算图像中每个像素点的梯度强度和方向。**

边缘检测主要是利用sobel算子与图像进行卷积，提取图像中的边缘特征，图像中的边缘可以指向各个方向，因此经典Canny算法使用四个算子来检测图像中的水平、垂直和对角边缘。但是通常都不用四个梯度算子来分别计算四个方向。常用边缘差分算子（如Rober，Prewitt，Sobel）计算水平和垂直方向的差分Gx和Gy。

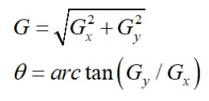
 

**水平方向的sobel算子 垂直方向梯度的sobel算子**

边缘检测的算子返回水平Gx和垂直Gy方向的一阶导数值，由此便可以确定像素点的梯度G和方向theta 。如下为卷积计算：



梯度计算：



在opencv中利用函数cv2.Canny来进行边缘检测，Canny算法应用双阀值，这里的阈值指的是梯度阈值，而不是灰度阈值。一般的边缘检测算法用一个阀值来滤除噪声或颜色变化引起的小的梯度值，而保留大的梯度值。即一个高阀值和一个低阀值来区分边缘像素。如果边缘像素点梯度值大于高阀值，则被认为是强边缘点。如果边缘梯度值介于高阀值与低阀值之间，则标记为弱边缘点，用于下一步的处理。小于低阀值的点则被抑制掉。

**2、应用非极大值抑制[5]，以消除边缘检测带来的杂散响应。**

非极大值抑制是一种边缘稀疏技术，非极大值抑制的作用在于“瘦”边。对图像进行梯度计算后，仅仅基于梯度值提取的边缘仍然很模糊（边缘比较粗）。而非极大值抑制则可以帮助将局部最大值之外的所有梯度值抑制为0，对梯度图像中每个像素进行非极大值抑制的算法是：

1. 将当前像素的梯度强度与沿正负梯度方向上的两个像素进行比较。梯度方向垂直于边缘方向。但实际上，我们只能得到像素点邻域的8个点的值，用于比较的两个像素点并不在其中，要得到这两个值就需要对该两个点两端的已知灰度进行线性插值。如下图的P1 与P2就是插值的结果。
2. 如果当前像素的梯度强度与另外两个像素相比最大，则该像素点保留为边缘点，否则该像素点将被抑制。
3. 完成非极大值抑制后，会得到一个二值图像，非边缘的点灰度值均为0，可能为边缘的局部灰度极大值点可设置其灰度为128。检测结果还是包含了很多由噪声及其他原因造成的假边缘。因此还需要进一步的处理。

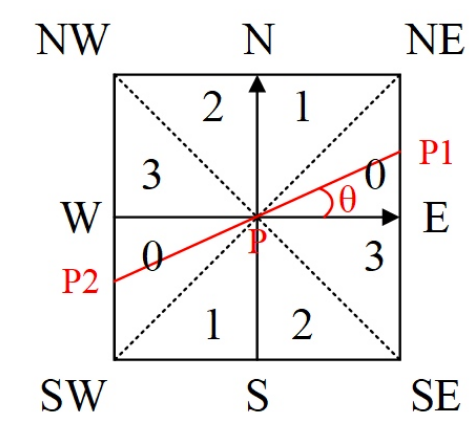


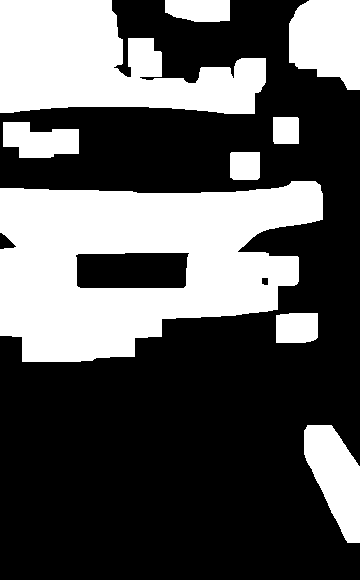
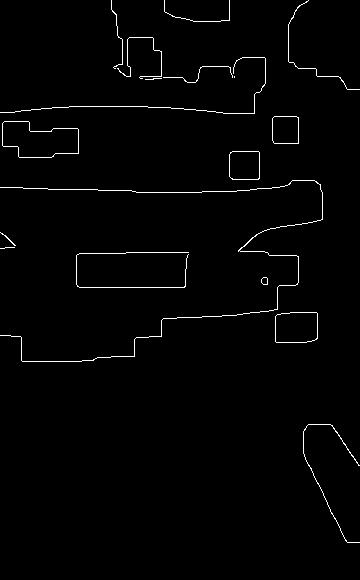
图10. 像素P领域的非极大值抑制

**3、应用双阈值（Double-Threshold）检测来确定真实的和潜在的边缘。**

Canny算法应用双阀值，这里的阈值指的是梯度阈值，而不是灰度阈值。一般的边缘检测算法用一个阀值来滤除噪声或颜色变化引起的小的梯度值，而保留大的梯度值。即一个高阀值和一个低阀值来区分边缘像素。如果边缘像素点梯度值大于高阀值，则被认为是强边缘点。如果边缘梯度值介于高阀值与低阀值之间，则标记为弱边缘点，用于下一步的处理。小于低阀值的点则被抑制掉。

**4、通过抑制孤立的弱边缘最终完成边缘检测。**

到目前为止，被划分为强边缘的像素点已经被确定为边缘，因为它们是从图像中的真实边缘中提取出来的。然而，对于弱边缘像素，将会有一些争论，因为这些像素可以从真实边缘提取也可以是因噪声或颜色变化引起的。为了获得准确的结果，应该抑制由后者引起的弱边缘。通常，由真实边缘引起的弱边缘像素将连接到强边缘像素，而噪声响应未连接。为了跟踪边缘连接，通过查看弱边缘像素及其8个邻域像素，只要其中一个为强边缘像素，则该弱边缘点就可以保留为真实的边缘。处理过程如下：

## 轮廓查找

根据上一步检测出来的边缘，利用**轮廓查找**函数cv2.findContours()函数找出图像中所有的轮廓, 函数返回三个参数，其中的 hierarchy便代表了检测到的轮廓在原图中的与部分其他轮廓层级关系： hierarchy是一个ndarray，其中的元素个数和轮廓个数相同，每个轮廓contours[i]对应4个hierarchy元素：hierarchy[i][0] ~hierarchy[i][3]，分别表示：后一个轮廓的序号、前一个轮廓的序号、子轮廓的序号、父轮廓的序号。在实际使用中是使用了contours这个参数。

## 轮廓拟合

利用opencv中的cv2.minAreaRect函数**对轮廓进行拟合**，找出轮廓的最小外接矩形。其输出结果是一个由三个元素组成的元组，依次代表旋转矩形的中心点坐标、尺寸和旋转角度（根据中心坐标、尺寸和旋转角度可以确定一个旋转矩形）。为了方便观察，可以结合cv2.boxPoints与cv2.drawContours函数将拟合的矩形画出来。结果如下：



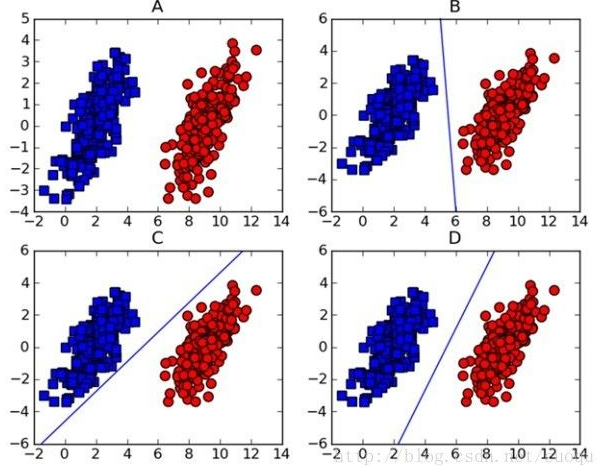
## 轮廓筛选

设定筛选条件，初步筛选出符合车牌长宽比（接近3:1）的矩形，如下图红色矩形框：



## SVM分类

利用**SVM分类器**，对剩下矩形框中的图像进行分类。关于SVM算法：对于线性可分两类数据，支持向量机就是条直线(对于高维数据点就是一个超平面), 两类数据点中的分割线有无数条，SVM就是这无数条中最完美的一条，即这条线距离两类数据点越远，则当有新的数据点的时候我们使用这条线将其分类的结果也就越可信。例如下图中的三条直线都可以将A中的数据分类：



要找到最优的结果，我们需要线找到数据点中距离分割超平面距离最近的点(找最小)然后尽量使得距离超平面最近的点的距离的绝对值尽量的大(求最大)。

支持向量机就是用来分割数据点那个分割面，他的位置是由支持向量确定的(如果支持向量发生了变化，往往分割面的位置也会随之改变), 因此这个面就是一个支持向量确定的分类器即支持向量机。

python库的scikit-learn整合了多种机器学习算法，利用scikit-learn中的sklearn.svm.SVC调用SVM算法，利用svc.fit(dataArr, label\_list)语句进行训练。其中参数dataArr为二维特征向量（将图片展平后堆叠），label\_list为对应的标签列表。

其中用于训练的数据集包括2800张蓝色车牌正样本（120\*30），以及8000张负样本（120\*30）。

正样本 负样本

经过训练之后得到对应的模型：based\_SVM\_plate\_train\_model.m用于检测车牌。定义检测函数：GetRes\_plate加载模型并得到车牌结果：

def GetRes\_plate(box):

gray = cv2.cvtColor(box,cv2.COLOR\_BGR2GRAY)

character\_Arr = np.zeros((1,3600))

img = cv2.resize(gray, (120, 30), interpolation=cv2.INTER\_LINEAR)

new\_character\_ = img.reshape((1,3600))[0]

character\_Arr[0,:] = new\_character\_

clf = joblib.load("based\_SVM\_plate\_train\_model.m")

predict\_result = clf.predict(character\_Arr)

middle\_route = ['0', '1']

return middle\_route[predict\_result.tolist()[0]]

## 仿射变换

上一步中已经找到了车牌的位置，考虑到不同图片中的车牌可能会发生视角倾斜， 这里采用透视变换，从原图中将车牌切割出来。

仿射变换后平行四边形的各边仍操持平行，透视变换结果允许是梯形等四边形，所以仿射变换是透视变换的子集，仿射变化是二维的二维的变换，透视变换是三维到三维的变换。仿射变换包括：平移、旋转、放缩、剪切、反射等。图像变换的实质就是矩阵的变换。下面是透视变换部分的实现：



至此完成对车牌的定位。下图为得到的车牌：



# 字符分割

## 去除冗余的空间

经过上一步之后得到的车牌图像，进行灰度化处理与和二值化处理得到二值图像：

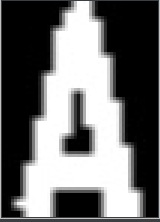
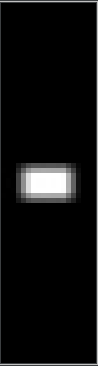
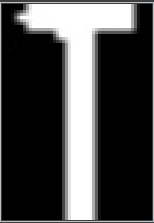
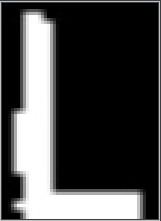
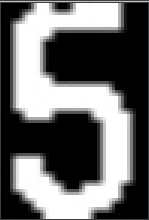
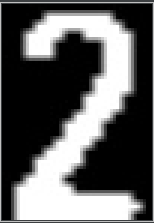


观察图像可以发现，图像中上部有一部分冗余的空间，为了不对后续识别处理的结果产生影响，现在将图片上部空间处理掉。主要思路是统计每一行的像素值的和，建立像素和值的直方图，然后设定一个阈值，找出在阈值之上的波峰区域作为字符的区域，利用该区域对图像进行裁剪，得到如下结果：



## 字符分割

现在需要对每个字符进行分割，然后分别进行识别，字符分割的思路是遍历图像的每一列，统计黑色像素（一定阈值范围内的像素值内的像素）的数目，然后设定像素数目阈值，当黑色像素数目大于数目阈值时，即认定其为可切割列，继续遍历即可获得连续的黑色像素区间，然后定位区间中位数进行切割，得到如下结果：

至此字符分割已经完成，去掉字符“·”，则获得了期望的7个字符。

# 字符识别

这里字符识别，仍然采用SVM训练的分类器，虽然SVM是一种用于二分类的机器学习方法，但是 sklearn中的SVM仍然可以完成多分类的任务。sklearn中的SVM算法底层的实现采用了台湾大学林智仁教授等发表的LibSVM运行库，LibSVM中采用了一对一（ovo）的多分类方法。

## 数据集的准备

省份字符的数据集与数字、字母字符的数据集来源不同，内容也相差比较大，为了不弄混淆，现在将其分别进行训练，数据集也分别准备。省份数据集共包含省份字符3216张（20\*20），数字字母字符共34\*50张。利用脚本将图片名读取并保存至txt文件中:

import os

filepath1 = "C:\\Users\\zh\\Desktop\\data\_province\\data\\"

filepath2 = "C:\\Users\\zh\\Desktop\\data\_province\\txt\\"

path\_list1 = os.listdir(filepath1) # data文件列表

print(path\_list1)

for i in range(len(path\_list1)):

path\_list2 = os.listdir(filepath1+path\_list1[i])

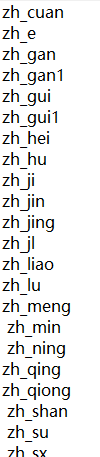
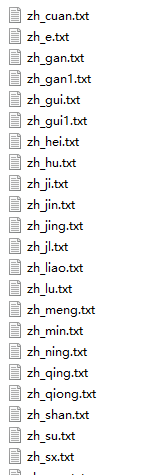
for j in range(len(path\_list2)):

with open(filepath2+path\_list1[i]+".txt","a") as f:

f.write(path\_list2[j] + "\n")

f.close()

得到如下结果：



## 训练与检测

训练得到相应的模型，并对对省份字符以及数字字母字符进行分类，将得到的结果进行拼接，得到最终识别结果：



至此字符识别已经完成。

# 总结

本次课程设计中的车牌定位单纯借助的是opencv开源库，导致本车牌识别系统泛化能力太差。本设计的改进就是在车牌定位环节运用目标检测算法进行检测。然后在进行后续的检测。考虑到数据集中的车牌颜色为蓝色，曾在车牌定位环节考虑将图片转换为HSV空间，试图通过设定颜色阈值筛选出车牌，但是考虑到这种方法会受到光线照射的影响，所以没有采用。也尝试过借用yolov3官方权重去做检测，但是该模型并未包含车牌这个类别，受疫情等客观因素的影响，暂时还没完成改进。