**基于OpenCV视觉库提取并识别车牌号码系统的设计与实现**

**摘 要**

车牌识别是计算机视觉领域应用的代表，是智能交通、智慧城市的重要组成部分。随着国内城市化进程发展加快，个人汽车的增加，交通压力会越来越重，因此智能交通将是未来交通发展的大方向。而智能车牌识别系统作为智能交通体系中的核心，也将会不断的更新发展。

本文针对车牌识别，在设计中主要是以开源计算机视觉库（OpenCV）为基础，对目标图像进行过滤、提取操作，再使用人工神经网络的多层感知器对所提取图像进行识别，从而实现从目标图像中车牌号码识别的功能。设计中主要实现了两大功能模块，一个是对提供图像中车牌号码区域的提取功能，另一个是对提取出的车牌区域进行字符分隔和识别的功能。

程序是使用C++语言以及OpenCV函数库编写，利用程序获取到相应路径下的目标图像，再经过程序处理后，最终在命令行窗口输出车牌号码（没有可视化的界面，设计是以纯代码实现）。在程序设计中，针对车牌提取程序中使用了矩形区域中心点筛选算法来确定车牌在图像中的位置；针对号码识训练样本使用矩阵存储数据，使用BP算法矩形进行字符预测。

【关键词】计算机视觉；人工神经网络；多层感知器；OpenCV

目 录

[第1章 绪 论 1](#_Toc513193638)

[1.1 相关技术简介 1](#_Toc513193639)

[1.1.1 人工神经网络 1](#_Toc513193640)

[1.1.2 图像处理 1](#_Toc513193641)

[1.1.3 OpenCV 2](#_Toc513193642)

[1.2 系统开发目的和意义 2](#_Toc513193643)

[1.3 系统开发工具 2](#_Toc513193644)

[第2章 需求分析 3](#_Toc513193645)

[2.1 业务需求 3](#_Toc513193646)

[2.2 用户需求 3](#_Toc513193647)

[2.3 功能需求 4](#_Toc513193648)

[第3章 系统设计 5](#_Toc513193649)

[3.1 模块设计 5](#_Toc513193650)

[3.2 流程设计 5](#_Toc513193651)

[3.2.1 图像车牌获取 5](#_Toc513193652)

[3.2.2 车牌号码识别 6](#_Toc513193653)

[第4章 系统实现 8](#_Toc513193654)

[4.1感兴趣区域提取 8](#_Toc513193655)

[4.2 滤波操作 8](#_Toc513193656)

[4.2.1 噪声 8](#_Toc513193657)

[4.2.2 滤波处理 9](#_Toc513193658)

[4.2.3 双边滤波 9](#_Toc513193659)

[4.3 图像灰度化 11](#_Toc513193660)

[4.3.1 灰度图 11](#_Toc513193661)

[4.3.2 像素灰度化 11](#_Toc513193662)

[4.4 图像二值化 13](#_Toc513193663)

[4.4.1 二值化 13](#_Toc513193664)

[4.4.2 二值化处理 13](#_Toc513193665)

[4.5 边缘检测 14](#_Toc513193666)

[4.5.1 高频滤波 14](#_Toc513193667)

[4.5.2 Canny 14](#_Toc513193668)

[4.5.3 图像边缘检测 14](#_Toc513193669)

[4.6 轮廓的选取与描述 15](#_Toc513193670)

[4.6.1 轮廓的选取 15](#_Toc513193671)

[4.6.2 轮廓的筛选 16](#_Toc513193672)

[4.7 确定图像ROI 17](#_Toc513193673)

[4.7.1 目标矩形区域的筛选 17](#_Toc513193674)

[4.7.3 确定感兴趣区域 19](#_Toc513193675)

[4.7.4 图像字符分隔 20](#_Toc513193676)

[4.8 人工神经网络参数设置 21](#_Toc513193677)

[4.9 神经网络的训练 21](#_Toc513193678)

[4.9.1训练样本 21](#_Toc513193679)

[4.9.2参数设置 22](#_Toc513193680)

[4.9.3训练神经网络 23](#_Toc513193681)

[4.10 神经网络识别结果 23](#_Toc513193682)

[第5章 系统测试 25](#_Toc513193683)

[5.1 测试方式 25](#_Toc513193684)

[5.2 测试数据 25](#_Toc513193685)

[5.3 测试结果 25](#_Toc513193686)

[5.4 系统局限性分析 26](#_Toc513193687)

[总 结 27](#_Toc513193688)

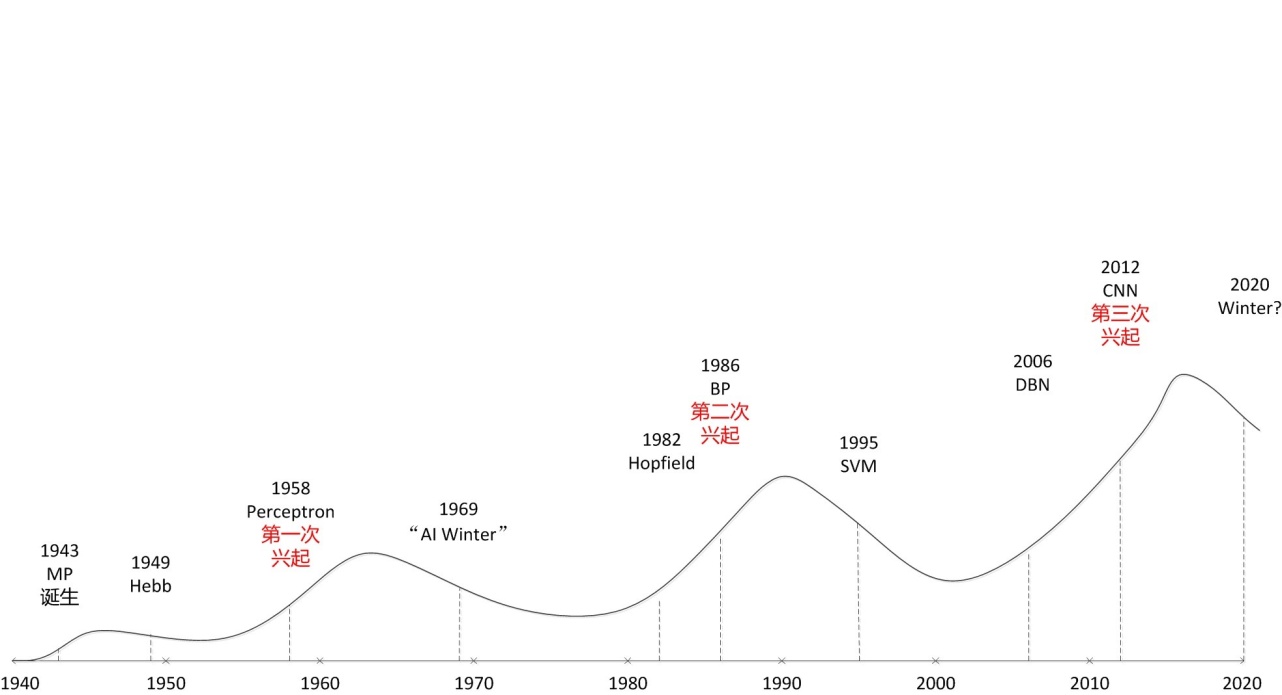
[参考文献 28](#_Toc513193689)

第1章 绪 论

## 1.1 相关技术简介

### 1.1.1 人工神经网络

人工神经网络发展的几十年来并不顺利，通过观察人工神经网络发展历史图1-1可以知道，人工神经网络的发展有过三次大的兴起,也有两次大的衰落。但是现在由于GPU、云等技术的出现，人工神经网络又一次迎来技术性突破发展。



1-1 人工神经网络发展历史图

在20世纪40年代的时候，M-P神经元的数学模型开启了人工神经网络（ANN）的时代[1]。著名生理学家Hebb在1949年提出神经心理学理论，为人工神经网络提供了学习规则[2]。1957年后，研究学者提出“感知器”模型，人工神经网络第一次出现在风口之上。1982年，人们又提出离散的人工神经网络模型，神经网络第二次兴起。在这之后BP神经网络模型、并行分布处理理论、Darwinism模型等上百种的人工神经网络模型被提出。2006年，深度学习的出现，使得机器学习出现了一个新的领域。而人工神经网络经过几十年的不断改进发明变得日趋完善。

### 1.1.2 图像处理

图像处理是在20世纪60年代随着计算机技术的发展进步而产生的一个新的技术领域。20世纪70年代，随着计算机技术和人工神经网络、思维科学的迅速发展，人们已经开始研究如何使用计算机系统来解释处理图像信息，这被称为图像理解或者计算机视觉。70年代末期 Marr提出了视觉计算理论，这个理论在这之后的十年中是计算机视觉的主要思想[3]。到了21世纪，随着计算机技术的飞速发展和计算机相关理论的不断完善扩充，图像处理技术引起了很多应用领域的广度关注，同时也取得了重大进展。

### 1.1.3 OpenCV

OpenCV的全称是Open Source Computer Vision Library，是一个跨平台的开源计算机视觉库，其中包括的许多常用的算法供人们免费试用。OpenCV已经在计算机视觉的每个领域都有涉猎，它的功能几乎包含了所有的研究方向。它实现的算法更新跟随最新的技术发展，为图像处理、目标跟踪、三维建模、机器学习等提供了大量的优化算法。OpenCV目前经历了3个版本，当从OpenCV1.x时代进入2.x时代的时候，它变的更加方便，如imread函数和imwrite函数等等的很多函数和MATLAB类似。2.x具有良好的封装性，基本实现了所有想要实现的功能，使用相应的功能的时候直接调用其API，简单明了。OpenCV3.x大部分是语言的变动， API由C语言转变为C++，代码简洁、错误率低。

## 1.2 系统开发目的和意义

车牌识别是计算机视觉技术在智能交通应用领域的重要研究课题之一，是实现智能交通不可缺少的一个环节，可以说车牌识别系统应用十分广泛。车牌识别系统加上一些后续处理手段可以实现：不停车自动收费；机场、港口等出入口车辆管理；小区车辆管理；闯红灯等违章车辆监控和交通流量检测公共停车场安全防盗管理等功能，它的市场价值和应用潜力巨大，可以产生很大的社会效益和经济效益。

## 1.3 系统开发工具

系统平台：Windows XP以上系统

集成开发工具：Visual Studio 2015

计算机视觉库：OpenCV 2.4.9

编程语言：C++

第2章 需求分析

## 2.1 业务需求

城市在发展进程中总会遇到各种各样的问题，如今看来，例如停车，停车难已经成为一线二线城市，城市化过程中一个畅聊的话题，停车位少、停车位不好找、停车收费慢等等，所以今天我们要讨论的话题就是车牌识别一体机对提高城市停车管理的重要性，车牌识别一体机不仅可以减轻停车场负担，收费难管理难的压力，还能为安防等行业，提供便利。

城市在设计的过程中，优化了停车场的布局，在建设停车场的时候更加的人性化、方便化，比如立体车库的不断涌现。当然，我们如何知道停车场内是否有空余车位、哪个位置有停车位。这就离不开车牌识别的作用。在停车场门口，设立车牌识别一体机，对进出的车辆扫描识别车牌号，与相关系统对接，记录车辆的信息。车牌识别，加强了对车辆的管理，有利于优化停车，方便用户找到停车位。

车牌识别系统(Vehicle License Plate Recognition，VLPR) 是计算机视频图像识别技术在车辆牌照识别中的一种应用。车牌识别在高速公路车辆管理中得到广泛应用，电子收费（ETC）系统中，也是结合DSRC技术识别车辆身份的主要手段。在停车场管理中，车牌识别技术也是识别车辆身份的主要手段。在深圳市公安局建设的《停车库（场）车辆图像和号牌信息采集与传输系统技术要求》中，车牌识别技术成为车辆身份识别的主要手段。

停车收费、高速路收费也是交通管理中常见的事情，人工收费、取卡收费，对车辆的行进产生了不小的影响，车辆常常堵在收费口处，赶上节假日因为收费堵车一两个小时，是常有的事情。有了车牌识别技术后，在车辆上安装射频卡，车牌识别之后，通过卡扣费，车辆直接通过，车牌识别技术使车辆通过用时不到1秒钟。车牌识别技术的应用，优化了车辆收费进程，使收费管理不在困难。

车牌识别技术结合电子不停车收费系统（ETC）识别车辆，过往车辆通过道口时无须停车，即能够实现车辆身份自动识别、自动收费。在车场管理中，为提高出入口车辆通行效率，车牌识别针对无需收停车费的车辆（如月卡车、内部免费通行车辆），建设无人值守的快速通道，免取卡、不停车的出入体验，正改变出入停车场的管理模式。

## 2.2 用户需求

车牌识别技术要求能够将运动中的汽车牌照从复杂背景中提取并识别出来，通过车牌提取、图像预处理、特征提取、车牌字符识别等技术，识别车辆牌号、颜色等信息。车牌号码提取识别系统主要为了解决现实生活交通对车辆的监控，管理等问题。实际应用中，要从图像或者影像中识别出车牌号码。

在系统实现的过程中，车牌的提取和号码的分析相互依赖，缺一不可。提取车牌直接影响号码分析，而号码分析又是最终结果，只有两项技术都成功实现，这个系统才算是最终完成。

所以，设计的程序要求能准确定位到车牌位置，以及能够快速识别车牌字符并将识别的结果输出到命令行窗口中，识别结果要求精确度够高。

## 2.3 功能需求

综合来看，车牌识别产品的竞争将更多的强调成像、图像预处理、算法、业务逻辑等系统各环节的相互配合。特别是物理成像以及图像预处理的水平可能会成为下阶段发展的关键，其原因在于当识别算法的发展达到一定的高度，短期内很难会出现非常突破性的变革，这时如何能够通过成像、光源、降噪等环节来为核心算法扫清障碍就成为了影响产品最终性能的制衡点。

停车场出入口安装车牌识别系统将是个趋势，智慧城市的建设，离不开车牌识别系统。通过图像的检测识别，可以实现交通违规现象的检测、违规车辆的车牌号码的识别等功能。对于图像中的车牌号码能准确定位，定位之后可以分割识别，最后输出识别结果。为了满足项目需求，系统的基本功能要求如下：

（1）加载图像：可以加载文件夹下的车牌图像。

（2）车牌获取：可以从图像中获取到车牌位置。

（3）车牌提取：从获取的车牌位置提取出车牌号码。

（4）号码识别：根据提取出的车牌号码字符进行识别。

（5）结果输出：将识别的结果进行输出。

第3章 系统设计

## 3.1 模块设计

车牌提取模块：本模块通过对图像进行灰度化处理、模糊处理、膨胀或腐蚀处理等操作对图像进行降噪过滤，再通过图像二值化和图像的边缘检测进行轮廓的选取和绘制，最后通过算法进行最后的降噪处理，得到最后的感兴趣区域（车牌）。

号码识别模块：本模块是通过将分类的数据进行参数设置，通过反向传播算法进行数据训练，再通过训练结果对得到的车牌字符进行识别。

系统功能模块结构图如图3-1所示：



3-1 系统需求模块结构图

## 3.2 流程设计

### 3.2.1 图像车牌获取

车辆图像获取是车牌识别的第一步，也是很重要的一步，车辆图像的好坏对后面的工作有很大的影响。如果车辆图像的质量太差，连人眼都没法分辨，那么肯定不会被机器所识别出来。车辆图像都是在实际现场拍摄出来的，实际环境情况比较复杂，图像受天气和光线等环境影响较大，在恶劣的工作条件下系统性能将显著下降。现有的车辆图像获取方式主要有两种：一种是由彩色摄像机和图像采集卡组成，其工作过程是：当车辆检测器（如地感线圈、红外线等）检测到车辆进入拍摄范围时，向主机发送启动信号，主机通过采集卡采集一幅车辆图像，为了提高系统对天气、环境、光线等的适应性，摄像机一般采用自动对焦和自动光圈的一体化机，同时光照不足时还可以自动补光照明，保证拍摄图片的质量；另一种是由数码照相机构成，其工作过程是：当车辆检测器检测到车辆进入拍摄范围时，直接给数码照相机发送一个信号，数码相机自动拍摄一幅车辆图像，再传到主机上，数码相机的一些技术参数可以通过与数码相机相连的主机进行设置，光照不足时也需要自动开启补光照明，保证拍摄图片的质量。

图像车牌的获取，主要分为以下的九个步骤，首先要获取目标图像，即源图像；获取到目标图像之后我们对目标图像进行滤波操作，过滤掉图像中的一些噪音；之后对图像进行灰度化，将图像彩色部分过滤剩下灰度色彩，更易于图像信息获取操作；再对灰度化的图像设定一个阀值，将其转化为黑白二值图像，也就是图像的二值化操作；对二值化图像进行边缘检测保留边缘部分；对边缘部分进行轮廓选取，选取之后进行筛选，就获得了我们想要的车牌区域；最后进行车牌号码的分隔操作。

相关步骤的流程图如图3-2所示：

3-2 车牌提取结构图

### 3.2.2 车牌号码识别

由于中国大部分的车牌是第一个是汉字，第二个到第七个是字母或数字，这就可以将车牌图像识别过程分成两部分处理，第一部分是识别汉字的过程，第二部分是识别字母和数字的过程，由于汉字笔画较多，同字母或数字的处理过程有所不同，所以程序这里就只是针对字母或数字的识别。

除汉字外，在第一个字母和第二个数字的中间有个一点，所以在字符分割的时候要考虑去掉中间的点。由于有些车牌的上面和下面也有螺丝之类的东西将车牌固定，所以在将车牌分割的时候，通过筛选的方法，可以去除掉，以便最后将车牌进行分割，去除这些干扰。

程序中，图像车牌号码的识别，主要分为以下的八个步骤。首先是将我们要训练的图像从相应的路径中读取；读取的图像抽象为输入的矩阵；过后计算机会将训练结果输出为矩阵；为待训练的矩阵设置人工神经网络参数；设置完成后继续人工神经网络训练，将所有的图像训练完成之后会生成一个xml数据文件；之后读取上述分隔的字符图像；将读取的图像进行识别；最后输出识别的结果。

相关步骤的流程图如图3-3所示：

3-3 车牌号码识别结构图

第4章 系统实现

## 4.1感兴趣区域提取

感兴趣区域（Region of Interest，ROI）指的是图像中的目标区域，也就是要重点操作的图像区域。而感兴趣区域的提取是对目标图像的一系列操作，主要的目的是通过过滤噪点从而得到想要的目标区域。在这个系统中使用到了图像灰度化、均值模糊、图像的膨胀、图像的二值化、边缘检测以及轮廓的选取与绘制等处理方式降噪，最后通过一定的算法去获取感兴趣区域。对于对图像的相关操作会以图4-1目标原图像例进行操作以及效果展示。



4-1 目标原图像

## 4.2 滤波操作

### 4.2.1 噪声

图像的噪声（noise）指的是图像中的粗糙部分，也指图像中不该出现的外来干扰像素，比如斑点等，包含噪声的图像布满一些细小的糙点，它们的存在使得图像质量下降，影响图像信息以及图像信息的提取。噪声在图像上常表现为有较强视觉效果的孤立像素点或像素块。

对于图像中的噪声一般采取滤波操作来处理，这样处理会将图像中的斑点像素剔除或者将斑点像素模糊化，从而大大降低图像的干扰因素，也就能更好的进行下一步操作。

包含噪声的图像如图4-2所示：



4-2 噪声图像

### 4.2.2 滤波处理

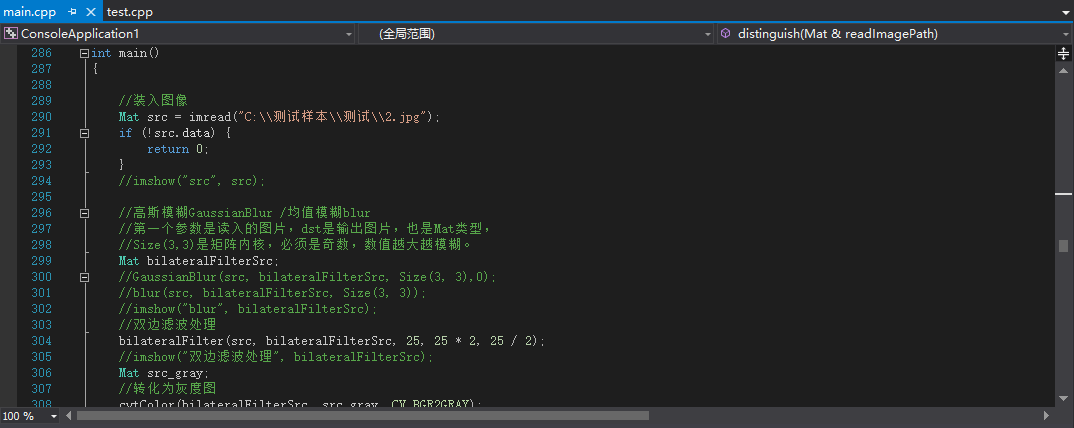
OpenCV中的过滤方法有很多，比如线性滤波方法（方框滤波、均值滤波与高斯滤波）和非线性滤波方法（中值滤波、双边滤波）。最常用的是高斯模糊和双边滤波（Bilateral filter）。将两者进行效果对比，高斯模糊效果图如图4-4，双边滤波的效果图如图4-5所示。

对比可以看出，双边滤波的一个重要特点就是能够保持物体的边缘像素，使得图像依然清晰，也就是说高斯模糊和均值模糊的图像效果都是使图像的像素模糊化，而双边滤波则是使图像的整体区域依然清晰。

### 4.2.3 双边滤波

双边滤波是由Tomasi [3] 提出，它作为一种非线性的滤波方法，是结合了图像的像素相似度和空间邻近度的一种折衷处理方法。Fleishman 等, 将图像处理中双边滤波器的思想推广到三维网格模型，设计了基于双边滤波[4-6]。的各向异性光顺去噪算法其中中值滤波的原理是用图像像素点邻域的灰度值的中值来代替该像素点的灰度。由于中值滤波是基于排序统计理论，所以它对脉冲噪声、椒盐噪声会有很好的滤波效果，并且能够保留图像的边缘细节像素，但是它的缺点就是效率低。设计中选择使用保留边缘细节的双边滤波进行操作。

OpenCV中C++关于双边滤波的函数是bilateralFilter()函数，配置参数后程序如图4-3所示：



4-3 双边滤波程序图

将路径下的图像导入程序中，再对图像进行双边滤波操作，最后得到的效果图如图4-5所示，其核心的代码是：

Mat src = imread("C:\\2.jpg");

Mat bilateralFilterSrc;

bilateralFilter(src, bilateralFilterSrc, 25, 25 \* 2, 25 / 2);

imshow("双边滤波处理", bilateralFilterSrc);

  
4-4 高斯模糊效果图

## 4.3 图像灰度化

### 4.3.1 灰度图

在日常生活中能看到的图像一般都是彩色图像，即红绿蓝（RGB）三色图像。也就是说彩色图像是由像素为单位组合而成，而每个像素又都是由R、G、B三种颜色分量决定的。图像的深度是由内存中每个像素点的字节数来共同决定，而每个字节的范围是0到255，所以每个像素有256\*256\*256种可能。对于信息量大的图像进行分析时困难会很大，所以要将彩色图像转化为灰度图来进行图像操作。

灰度将彩色转化为灰色，图像的像素分为256种情况（0到255）。灰度也可以认为是亮度，是对色彩的深浅程度的描述。与彩色图像相比，灰度图像虽然失去了一些颜色信息，但是它仍然可以描述目标图像局部和整体的色度、亮度等级的分布和特征。



4-5 双边滤波效果图

### 4.3.2 像素灰度化

生活中大多数看到的彩色图片都是RGB类型，但是在进行图像处理时，需要用到灰度图、二值图、HSV、HSI等颜色制式，OpenCV提供了cvtColor函数来实现这些功能，函数的作用是将一个图像从一个颜色空间转换到另一个颜色空间。

cvtColor函数在C++中的定义如下：

void cvtColor

(

InputArray src, // 输入的图像序列

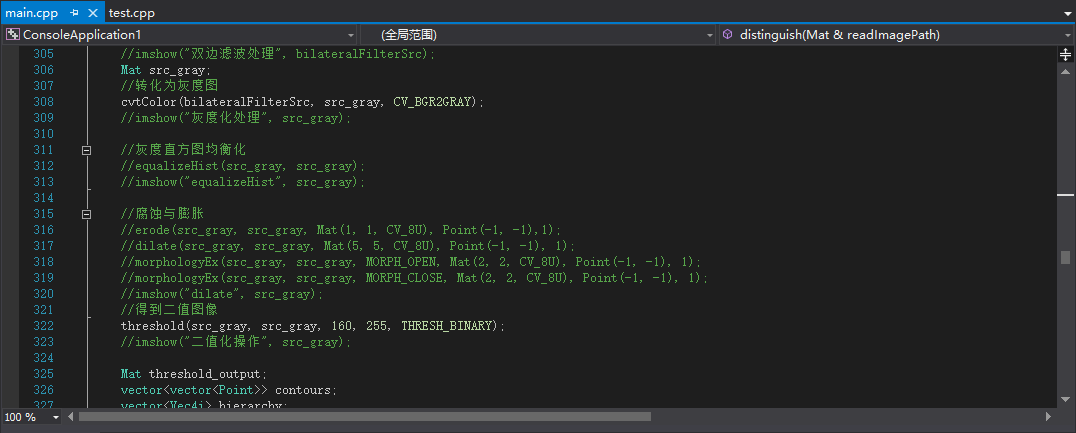
OutputArray dst, // 输出的图像序列

int code, // 颜色映射码

int dstCn = 0 // 输出的通道数（默认值为0）

);

因为OpenCV中默认的彩色图像的颜色空间不是RGB而是BGR，所以选择CV\_BGR2GRAY作为code参数，第四个参数dstCn选择默认值。使用OpenCV的cvtColor()函数对上边图4-5进行下边的函数操作，其代码截图如图4-6所示：



4-6 图像灰度化程序图

将滤波处理后的图像经过灰度操作之后的效果图如图4-7所示：



4-7 灰度效果图

## 4.4 图像二值化

### 4.4.1 二值化

图像的二值化操作和图像灰度化有些类似，图像二值化是把图像按照设定的阀值将图像分为0或255两种像素，也就是说图像二值化操作之后图像会变成黑白图像。通过图像的二值化操作，就可以区分图像前景和背景，或者得到感兴趣部分。

### 4.4.2 二值化处理

在OpenCV中，使用C++函数threshold()，threshold()函数包括了5个参数，其中第三个参数是阀值，也就是线程黑白像素的临界值，第五个参数是阀值类型（取阀值类型为THRESH\_BINARY时，当前点值大于阈值，取第四个参数Maxval）：

double threshold

(

InputArray src, // 输入的图像序列

OutputArray dst, // 输出的图像序列

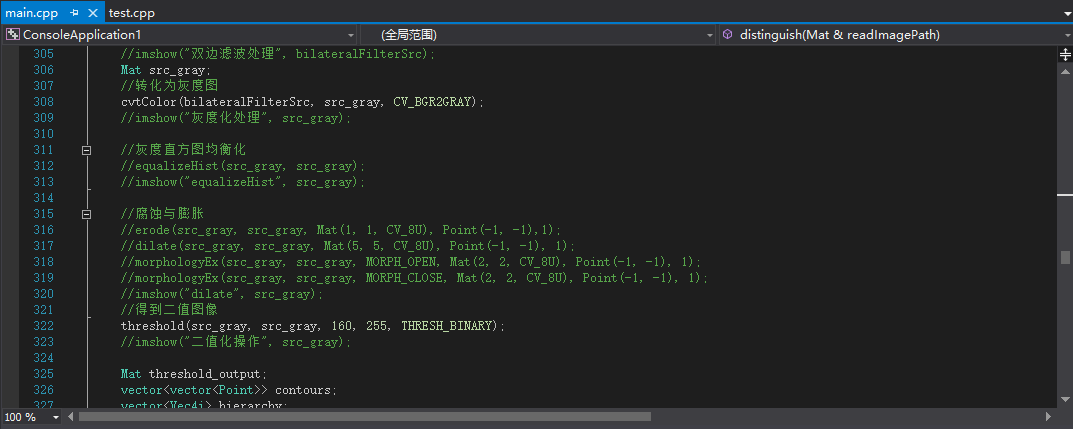
double thresh, // 阈值的具体值

double maxval, // 取得阀值

int type // 阈值类型

);

将阀值设定为160，Maxval设置为最大255，类型为THRESH\_BINARY，设置后的程序如图4-8所示。



4-8 图像二值化程序图

执行程序后，得到的二值化灰度图效果如图4-9所示：



4-9 二值化效果图

## 4.5 边缘检测

### 4.5.1 高频滤波

图像的边缘信息大部分都集中在高频段部分，一般的图像锐化或者边缘检测实质上就是对高频波的过滤，即高频滤波。通过扫描图像中的像素来检测灰度级或者结构具有巨大变化的地方，这就表明在这个地方是不连续的，是区域的分割线，这种不连续称之为边缘。图像边缘检测的目的是标识目标图像中具有明显变化的像素点，去除图像中不需要的信息部分, 它的结果直接影响进一步图像处理效果[4]。

### 4.5.2 Canny

Canny 算子作为一种优化的边缘检测算子[5]，先是使用高斯滤波器平滑图像，之后使用Sobel等梯度算子计算梯度幅值和方向并且对梯度幅值进行非极大值抑制，最后用双阈值算法检测和连接边缘。Canny边缘检测满足了低错误率的边缘检测、最优定位以及图像中的任意边缘应该只被标记一次的三个要求。经过Canny边缘检测的图像效果是很明显的，对比一般的梯度算法跟大程度上抑制了噪声所引起的伪边缘，对图像的像素边缘细化更细，也容易进行后边的处理。

### 4.5.3 图像边缘检测

OpenCV中关于C++实现的Canny的算法包含6个参数。

void Canny

(

InputArray image, // 输入的图像序列

OutputArray edges, // 输出的图像序列

double threshold1, // 第一个滞后性阈值

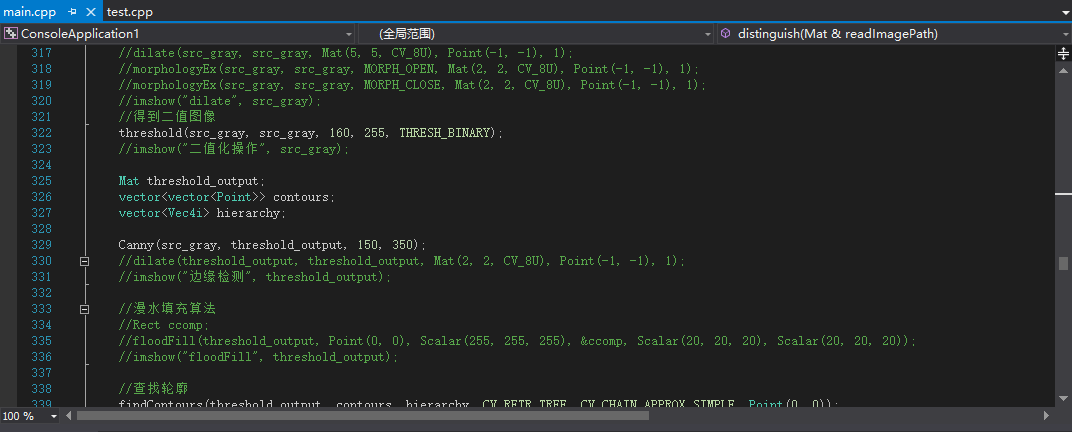
double threshold2, // 第二个滞后性阈值

int apertureSize=3, // 应用Sobel算子的孔径大小

bool L2gradient=false // 计算图像梯度幅值的标识

);

对于第一个滞后性阈值和第一个滞后性阈值，两者中数值较大者用来控制强边缘的初始段，较小者用于边缘连接，对于高低阈值比例一般处在2:1到3:1之间。默认最后两个参数值，具体使用函数如图4-11所示，执行程序后得到的效果图如图4-12所示。



4-11 边缘检测程序图

## 4.6 轮廓的选取与描述

### 4.6.1 轮廓的选取

图像经过边缘检测之后信息大量减少，想要将车牌号码识别就必须在目前的图像中圈定的目标区域。现在要将剩余的像素使用findContours函数选取轮廓。

OpenCV中使用C++实现的findContours函数结构如下：

findContours

(

InputOutputArray image, // 输入的图像序列

OutputArrayOfArrays contours, // 输出的图像序列

OutputArray hierarchy,

int mode, // 定义轮廓的检索模式

int method, // 定义轮廓的近似方法

Point offset=Point() // Point偏移量

);

使用findContours函数设定参数，选取轮廓值，代码如下：

Mat threshold\_output;

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

findContours(threshold\_output, contours, hierarchy, RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, Point(0, 0));

使用函数操作二值图像的效果图：



4-12 边缘检测效果图

### 4.6.2 轮廓的筛选

获取到图像的轮廓后需要将图像轮廓绘制描述为矩形以便后续操作。因此要对轮廓进行多边形近似处理求得矩形包围框RECT，核心代码如下：

for (int i = 0; i< contours.size(); i++)

{

Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.uniform(0, 255));

Rect rect((boundRect[i].tl().x + boundRect[i].br().x) / 2, (boundRect[i].tl().y + boundRect[i].br().y) / 2, 0, 0);

ListRect matrix(boundRect[i].tl(), boundRect[i].br(), rect.tl(), i);

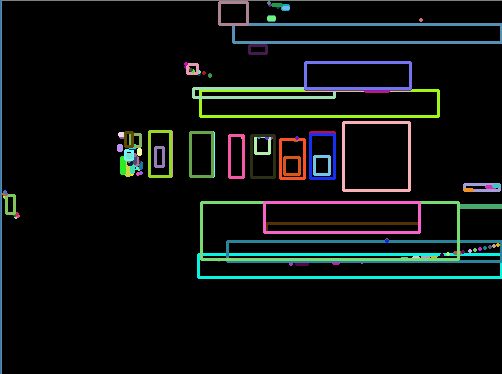
rects.push\_back(matrix);

rectangle(drawing, boundRect[i].tl(), boundRect[i].br(), color, 2, 8, 0);

count++;

}

得到了矩形包围框之后要对矩形进行描述，使用函数rectangle绘制出每一个包围框，轮廓绘制的图像如图4-13所示：



4-13 轮廓描述效果图

## 4.7 确定图像ROI

### 4.7.1 目标矩形区域的筛选

因为图像中包围框数量很多，所以要对包围框进行初步的筛选，参考车牌上号码的字符比例，将目标矩形凡是宽的长度大于高的长度三分之二或者小于高度四分之一的矩形去除；轮廓图中包含大量的非常小的矩形，没有实际意义，可以将它们去除，所以将目标矩形宽长小于6像素或者高度小于10的矩形去除；图中也包含了一些非常大的矩形，参考车牌上号码的数量可以按照比例将图像中大于比例的矩形去除，因此将矩形大于原图像宽度六分之一的矩形也去除，最后按照中心点的纵坐标降序将所有符合条件的包围框进行排列，剩下的矩形就是包含车牌信息的初步筛选结果，核心代码如下：

int count = 0;

vector<ListRect> rects;

for (int i = 0; i< contours.size(); i++)

{

//初步筛选

if (boundRect[i].width <= boundRect[i].height \* 2 / 3 &&

boundRect[i].width > 5 && boundRect[i].height > 9&&

boundRect[i].width > boundRect[i].height / 4 &&

boundRect[i].width < src.cols / 6) {

Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.uniform(0, 255));

Rect rect((boundRect[i].tl().x + boundRect[i].br().x) / 2, (boundRect[i].tl().y + boundRect[i].br().y) / 2, 0, 0);

ListRect matrix(boundRect[i].tl(), boundRect[i].br(), rect.tl(), i);

rects.push\_back(matrix);

//绘制包围框,tl是左上角坐标, br是右下角坐标,包围框为矩形

rectangle(drawing, boundRect[i].tl(), boundRect[i].br(), color, 2, 8, 0); count++;

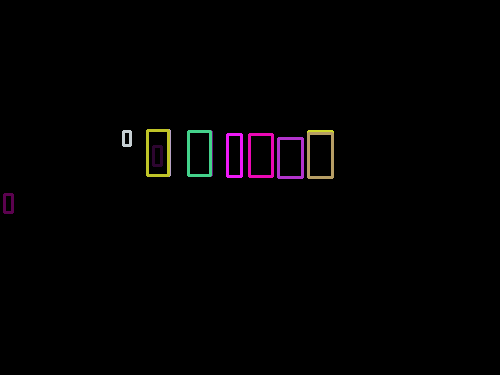
}

}

//按照中心点的纵坐标降序

sort(rects.begin(), rects.end(), sortDown);

初步筛选的区域图像如图4-14所示：

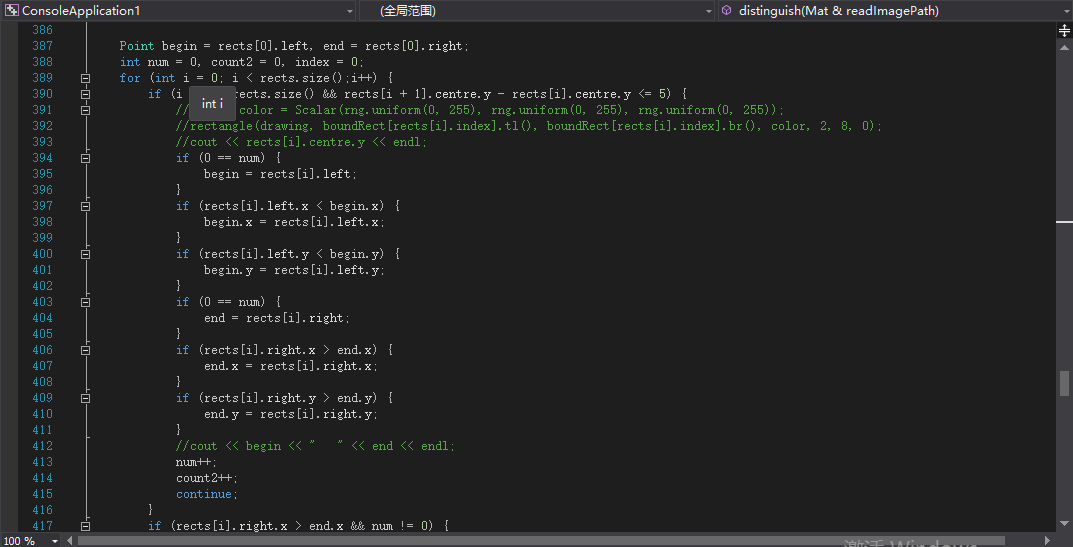


4-14 初步筛选轮廓效果图

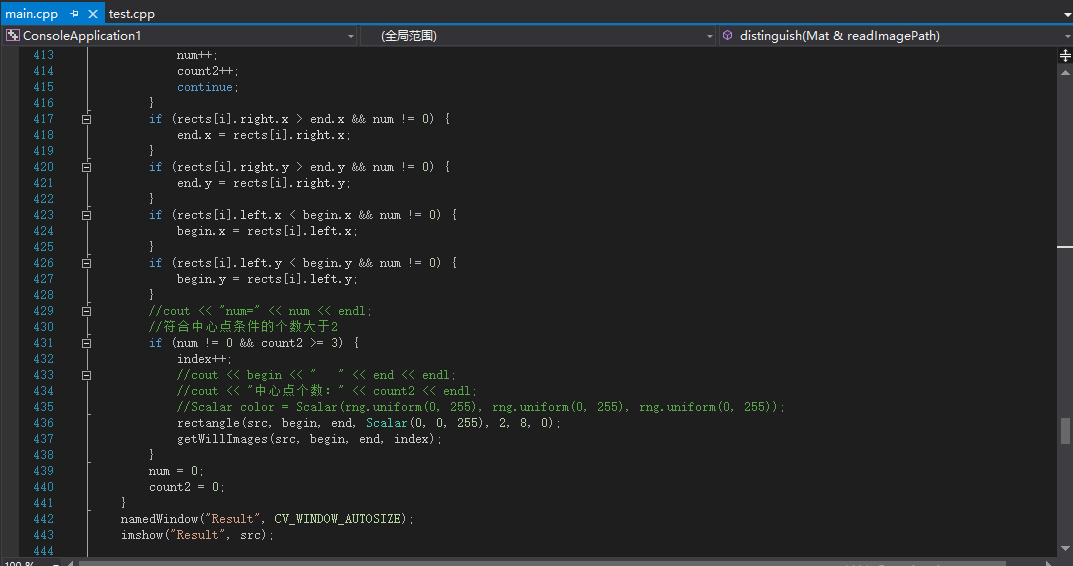
### 4.7.3 确定感兴趣区域

剩下的矩形中包含感兴趣区域，考虑到车牌号码是包含在一个矩形中，也就是说各个号码是线性排列的，即它们的中心点是在一条线上的。为了更加精确的定位，也为了增加容错率，会增加一些像素误差来平衡结果：

将找到的矩形中心点存放在一个容器中，并且按照由小到大的顺序排列，根据像素误差以及设定中心点最少个数（3个），找出在像素误差范围的几组数据。并按照原矩形上下左右像素将这几组矩形绘制，程序如图4-15和4-16所示：



4-15 中心点筛选程序图一



4-16 中心点筛选程序图二

最终获取的目标区域如图4-17所示：



4-17 目标区域效果图

目标区域在原图像中的区域位置如图4-18所示：

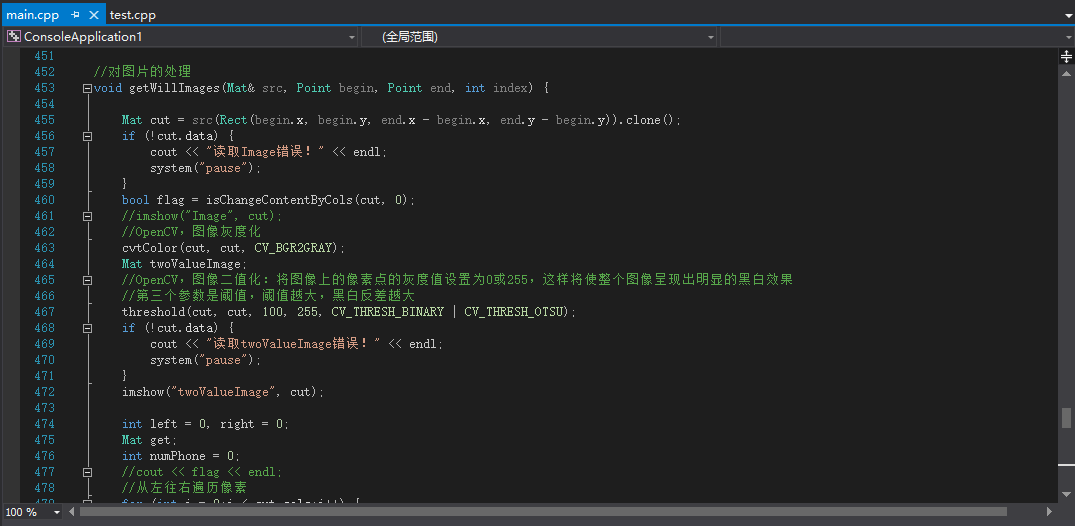


4-18 原图像目标区域效果图

### 4.7.4 图像字符分隔

获取感兴趣区域的图像之后，对图像进行灰度化和二值化处理，并将图像上的像素点的灰度值设置成黑白像素，也就是0或255像素。根据像素的黑白变化将图像分隔成三种，一种是包含车牌号码的有用图像，一种是无用的号码间的黑色分隔区域，以及没有筛选干净的“假”信息图像。

对应没用的黑色分隔区域之间判断其是否包含白色像素直接去除，没有筛选干净的干扰图像在识别时增加准确率的判别来去除干扰，程序如图4-19所示。



4-19 字符分隔程序图

分隔后的效果图如图4-20所示：



4-20分隔效果图

## 4.8 人工神经网络参数设置

对于人工神经网络参数的设置，设计选择使用BP算法，BP 算法的基本思想是，学习过程由信号的正向传播与误差的反向传播两个过程组成[9]，对BP算法设置了两个参数，一个是权值更新率bp\_dw\_scale和一个是权值更新冲量bp\_moment\_scale。设置人工神经网络的迭代次数为10000次，最小误差值为0.0001，层数为5层。核心代码为：

//OpenCV，神经网络训练参数的类型

CvANN\_MLP bp;

CvANN\_MLP\_TrainParams params;

//OpenCV，设置算法：反向传播算法BACKPROP

params.train\_method = CvANN\_MLP\_TrainParams::BACKPROP;

//OpenCV，针对于反向传播法，主要是两个参数，一个是权值更新率bp\_dw\_scale和权值更新冲量bp\_moment\_scale

params.bp\_dw\_scale = 0.001;

params.bp\_moment\_scale = 0.1;

//OpenCV，迭代次数和误差最小值

params.term\_crit=cvTermCriteria(CV\_TERMCRIT\_ITER+ CV\_TERMCRIT\_EPS, 10000, 0.0001);

//OpenCV，创建一个5层的神经网络

Mat layerSizes = (Mat\_<int>(1, 5) << imageRows\*imageCols, 128, 128, 128, classNum);

//OpenCV，设置一个5层的神经网络

//bp.create(layerSizes,CvANN\_MLP::SIGMOID\_SYM,1.0, .0);

//CvANN\_MLP::SIGMOID\_SYM,CvANN\_MLP::GAUSSIAN ,CvANN\_MLP::IDENTITY

## 4.9 神经网络的训练

### 4.9.1训练样本

训练的数据样本有34组，（因为0和O，1和I相似，所以样本中不包含O和I），每组平均包含200个训练数据，如图4-21所示。

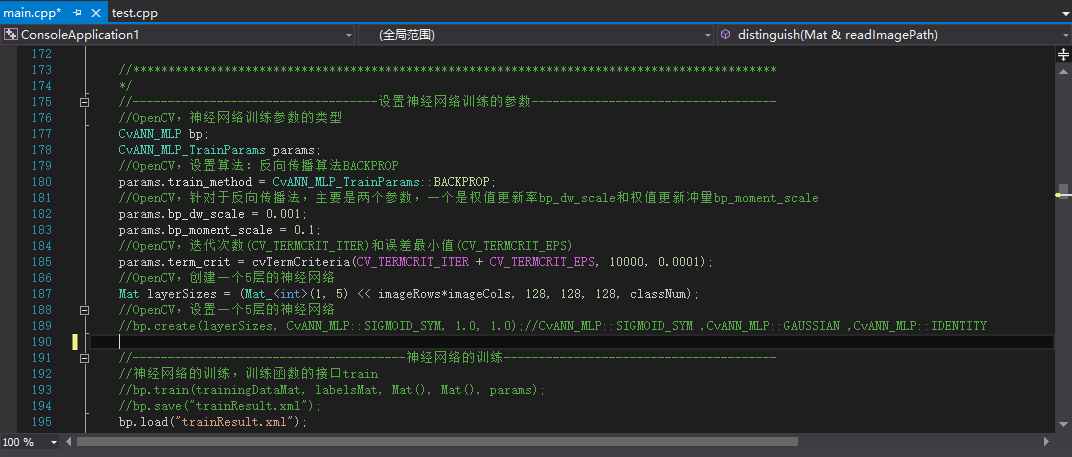
将放在路径下的待训练数据信息遍历取出获取，每次获取信息时简单处理图像，并在读取图像时生成训练的矩阵，将按照BP算法进行操作的数据保存于trainResult.xml文件中，再次使用直接加载配置文件即可。



4-21训练样本

### 4.9.2参数设置

使用OpenCV中多层神经网络进行相关的参数设置。在神经网络中使用反向传播算法（BP算法），针对于反向传播法，主要是两个参数：权值更新率bp\_dw\_scale设置为0.001，权值更新冲量bp\_moment\_scale设置为0.1；设置网络迭代次数(CV\_TERMCRIT\_ITER)和误差最小值分别为10000和0.0001；最后设置神经网络层数为5层，参数设置后如图4-22所示。



4-22程序参数设置

核心代码如下：

CvANN\_MLP bp;

CvANN\_MLP\_TrainParams params;

params.train\_method = CvANN\_MLP\_TrainParams::BACKPROP;

params.bp\_dw\_scale = 0.001;

params.bp\_moment\_scale = 0.1;

params.term\_crit = cvTermCriteria(CV\_TERMCRIT\_ITER + CV\_TERMCRIT\_EPS, 10000, 0.0001);

Mat layerSizes = (Mat\_<int>(1, 5) << imageRows\*imageCols, 128, 128, 128, classNum);

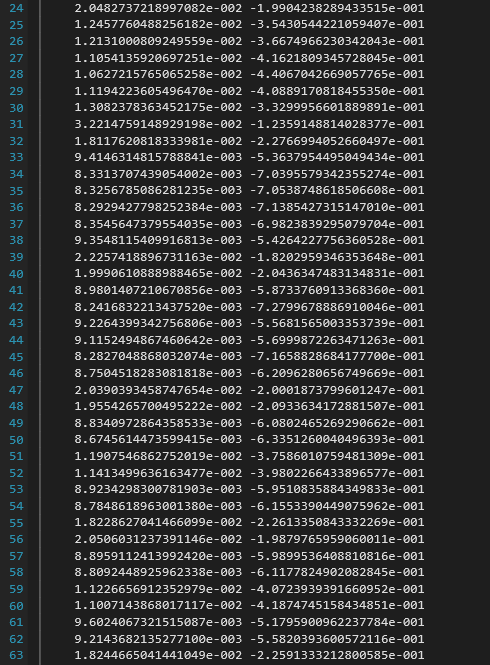
### 4.9.3训练神经网络

将包含所有训练样本特征的矩阵进行算法操作并保存在trainResult.xml文件中：

bp.train(trainingDataMat, labelsMat, Mat(), Mat(), params);

bp.save("trainResult.xml");

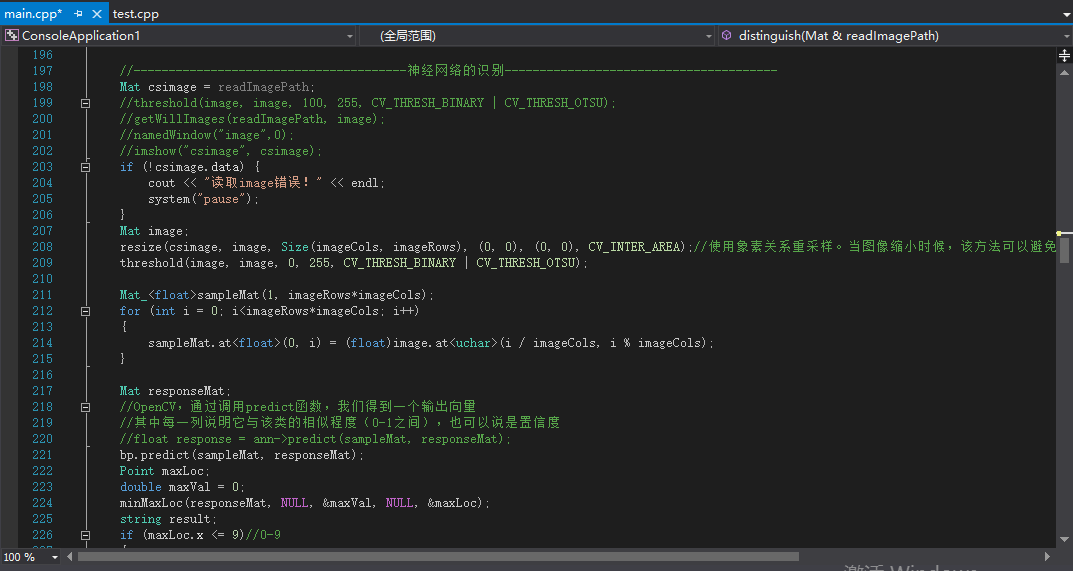
训练后生成的xml文件如图4-23所示：



4-23训练结果部分图

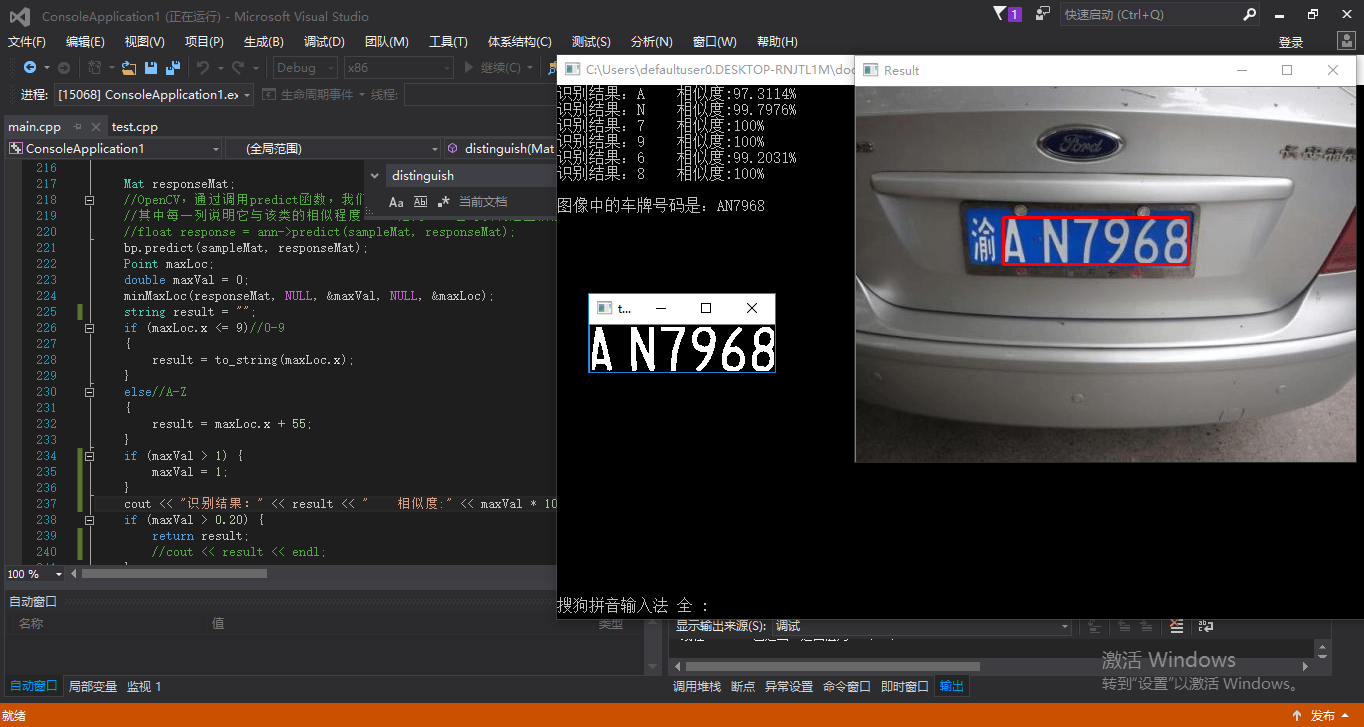
## 4.10 神经网络识别结果

通过OpenCV，调用predict函数，我们得到一个输出向量，这个输出向量包含了两个我们需要的结果，相似结果以及相似度，根据这两个结果我们将相似度（置信度）低的剔除，最后再将结果输出到命令行窗口，相关程序如图4-24所示。



4-24结果预测程序图

将识别结果输出到命令行窗口上，对应测试用例的输出结果如图4-25所示：



4-25 输出结果图

第5章 系统测试

## 5.1 测试方式

一般常见的测试方式包括黑盒测试和白盒测试。

黑盒测试指的是在测试时，在完全不考虑程序内部结构与内部特性的情况下，测试者在程序接口进行测试[10-12]。

本文测试使用了黑盒测试方式对识别车牌进行测试。

## 5.2 测试数据

测试的数据样本来源于两个方面：网络图片以及手机拍摄。测试数据共选取了21个，如图5-1所示：

5-1 测试样本

## 5.3 测试结果

一共选取了21个测试样本，其中正确识别的样本数量为15，一个字符误差数量为4，其他错误数量为2，测试结果统计表如表5-1所示：

表5-1 测试结果统计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 车牌号码 | 识别车牌 | 是否完全识别 | 错误原因 | 其他 |
| KR9888 | KR9888 | 是 | 没有错误 |  |
| AN7968 | AN7968 | 是 | 没有错误 |  |
| H59590 | H5959D | 否 | 0识别为D | 区域多余 |
| RCYCLED | RCVCLED | 否 | Y识别为V |  |
| B27X99 | B27X9D | 否 | 9识别为D |  |
| B0009V | 0009 | 否 | 区域选取不当 |  |

表5-1 测试结果统计表（续表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AT0H69 | AT0H69 | 是 | 没有错误 |  |
| EN78R7 | EN78R7 | 是 | 没有错误 |  |
| ANT989 | ANT989 | 是 | 没有错误 |  |
| A88888 | 488888 | 否 | A识别为4 |  |
| EG26J6 | EG26J6 | 是 | 没有错误 | 区域有两处 |
| B99999 | B99999 | 是 | 没有错误 | 区域多余 |
| K44444 | N4444 | 否 | 区域选取不当 | 区域左多余 |
| H59590 | H59590 | 是 | 没有错误 |  |
| A92HD7 | A92HD7 | 是 | 没有错误 |  |
| AR8887 | AR8887 | 是 | 没有错误 |  |
| FSA449 | FSA449 | 是 | 没有错误 |  |
| F0N234 | F0N234 | 是 | 没有错误 |  |
| F379A6 | F379A6 | 是 | 没有错误 |  |
| F678K2 | F678K2 | 是 | 没有错误 |  |
| AQ9277 | A09277 | 否 | Q识别为0 |  |

## 5.4 系统局限性分析

此设计仅仅是最基本的车牌提取与识别设计，实际应用中需要考虑的因素非常繁杂，比如图像角度、图像的噪声差异、车牌中文的掺杂、车牌的不同种类等等。明显的不足包含：所用的训练样本少，准确性有待提高，对相似的字符识别依然存在错误。这里只是简单的实现了简单的图像英文车牌的识别或者是不包含汉字的中国车牌识别，因此此项目还要针对更加具体的细节情况去完善。所以基于设计本身，还要一些后续能展开的工作，比如车牌的透视处理，更加精确的降噪算法的实现，增加中文汉字的识别等等。