**北 京 林 业 大 学**

**2022 学年— 2023 学年第 一 学期 数字视频技术及应用（课程设计）实验报告书**

实验地点： 学研 任课教师： 刘文萍

实验题目： **视频中车牌号识别**

实验环境： **VS+OpenCV**

目录

[分工： 2](#_Toc155560984)

[实验内容: 2](#_Toc155560985)

[实现方法: 2](#_Toc155560986)

[一、 使用mfc框架编写界面： 2](#_Toc155560987)

[1. 视频和图像显示： 2](#_Toc155560988)

[2. 打开文件 4](#_Toc155560989)

[3. 最终结果显示 5](#_Toc155560990)

[二、 对视频帧进行预处理： 5](#_Toc155560991)

[三、 车牌区域提取： 6](#_Toc155560992)

[四、 切割车牌区域： 6](#_Toc155560993)

[五、 投影法分割字符： 8](#_Toc155560994)

[1. 预处理和二值化： 9](#_Toc155560995)

[2. 裁剪出包含字符区域的子图像 9](#_Toc155560996)

[3. 垂直投影： 10](#_Toc155560997)

[4. 将投影得到的子图像存到DisplayImages： 12](#_Toc155560998)

[六、 字符识别： 12](#_Toc155560999)

[1. 模板匹配 12](#_Toc155561000)

[2. SVM向量机 12](#_Toc155561001)

[实验结果: 12](#_Toc155561002)

[总结：（难点以及解决方案） 15](#_Toc155561003)

# 分工：

我负责实现整体MFC程序的框架实现，对话框的制作，视频帧预处理车牌区域提取，切割车牌提取等功能的实现以及在字符识别svm框架的实现，程序的测试等任务。

# 实验内容:

编程实现视频中车牌号的识别。输入为含有车辆车牌的视频，视频可以自己拍摄或者使用网上资源，经过图像预处理、车牌定位、字符分割和字符识别等一系列处理，输出为具体车牌号。两人一组共同完成，完成后展示实验结果。

# 实现方法:

本课设项目中的所有工程文件放在如下文件夹中，配置好opencv环境即可运行。



## 使用mfc框架编写界面：



使用MFC对话框类实现界面可视化

### 视频和图像显示：

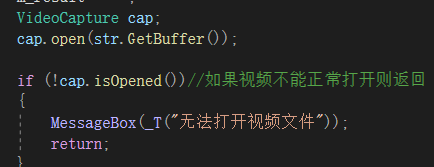
视频和显示图像使用picture control控件，结合opencv进行定位显示。

* **首选我在自己设置的对话框中添加picture control组件：ID为： IDC\_VIDEO。**



* **之后，在代码中逐帧读取视频：**

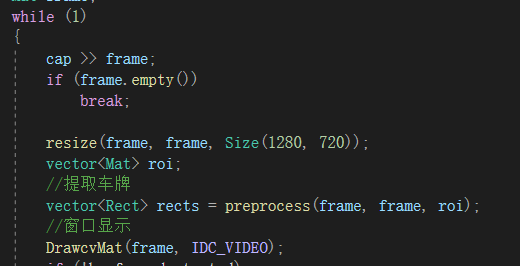
在OnBnClickedOpenvideo这个函数中，我使用OpenCV打开视频文件



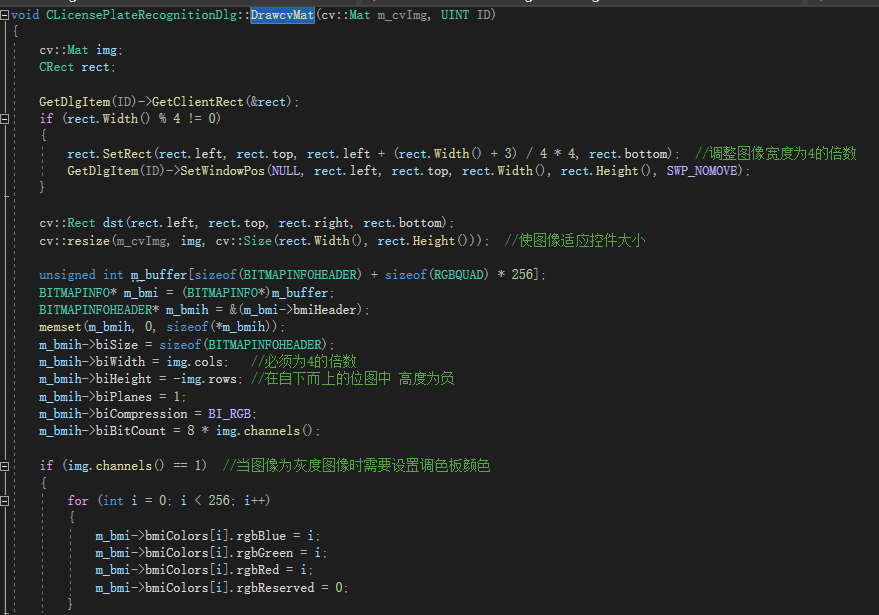
如果无法打开，则显示“无法打开视频文件”，退出整个系统。

* **循环读取视频帧并显示：**

不断地读取视频帧，对每一帧进行处理，然后使用DrawcvMat函数将处理后的图像显示在IDC\_VIDEO控件上。

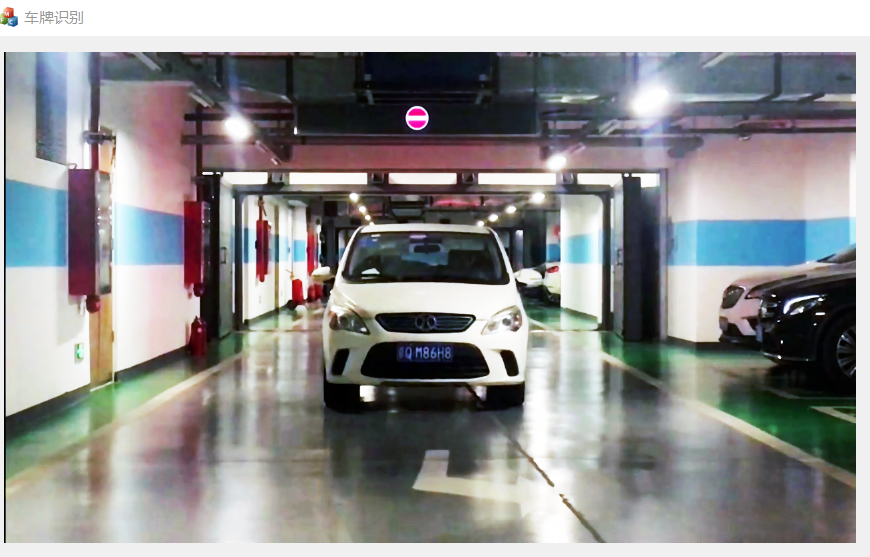


* **定义DisplaycvMat函数，将OpenCV的Mat图像绘制到MFC控件上**

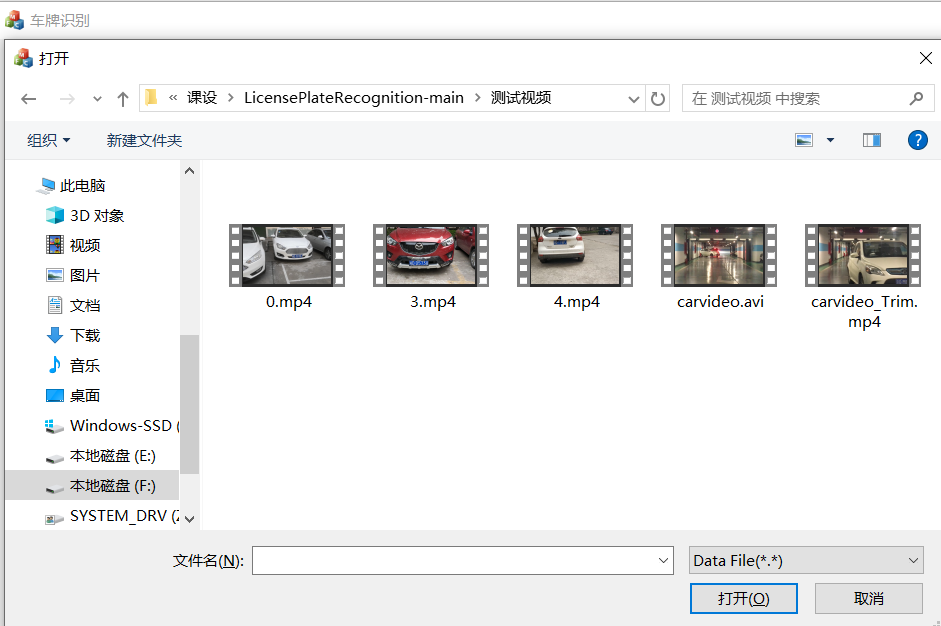


函数通过获取控件的客户区域大小，创建内存DC和兼容位图，将OpenCV图像通过StretchDIBits函数绘制到内存DC上，然后再将内存DC的内容绘制到控件DC上，实现图像的显示。

得到如图所示的结果：

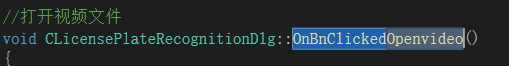


### 打开文件

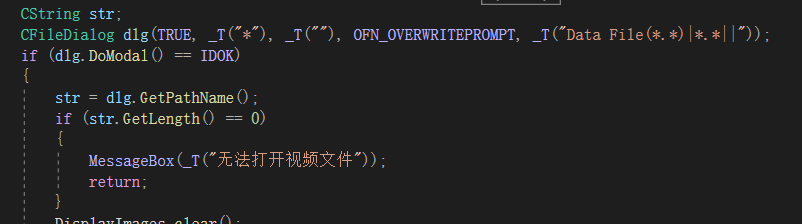


点击打开视频按钮之后，出现如图所示的界面。

在mfc框架中，我添加了如下函数，实现了打开文件的操作。

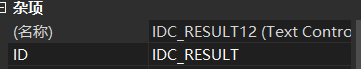
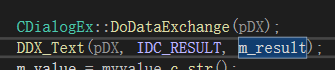


* + 创建 CFileDialog 对话框对象，设置参数（OFN\_OVERWRITEPROMPT 表示如果用户指定的文件名已经存在，将提示用户确认是否覆盖）。
  + 调用 DoModal 函数显示文件对话框，等待用户选择文件或取消操作。
  + 若用户选择了文件（IDOK），通过 GetPathName 获取选中的文件路径。



### 最终结果显示

显示最终结果使用Edit control动态获取显示结果字符串

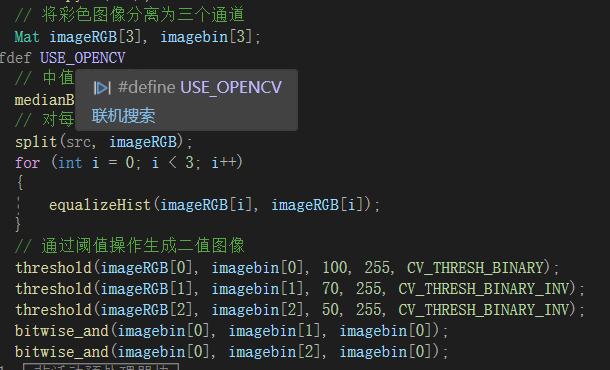
 



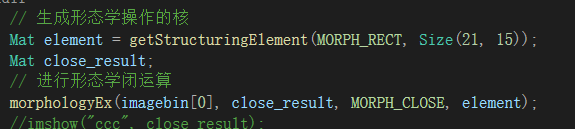
## 对视频帧进行预处理：

对图像进行预处理我写在了preprocess函数中，具体的实现步骤如下：

1. 将彩色图像分离为三个通道，并对每个通道进行直方图均衡化。
2. 使用阈值操作生成二值图像。



1. 进行形态学闭运算，突出车牌区域。

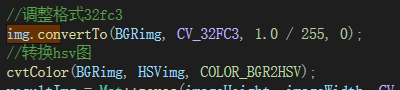


## 车牌区域提取：

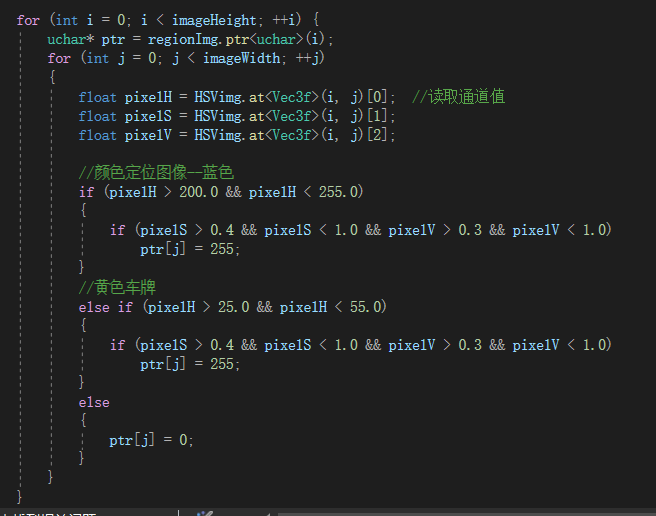
根据先验知识，中国车牌为长 45cm,宽 15cm的矩形车牌，长宽比 3:1。车牌颜色通常为蓝底白字、黄底黑字等，所以采用基于HSV颜色模型的方法对车牌定位效果会比RGB要好，从而进一步判断，提取含有车辆的图像。

具体的算法步骤如下：

首先将图像从rgb转换成hsi空间：



然后通过阈值过滤找到蓝色和黄色车牌的区域，蓝色车牌H在200~255，S在0.4~1.0，V在0.3~1.0的范围内过滤，黄色车牌同理。

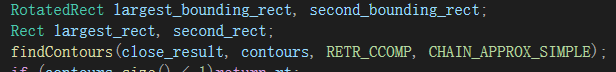


## 切割车牌区域：

采用基于形状的方法。利用OpenCV的findContours函数检测物体的轮廓，通过判断物体的面积大小、长宽比来提取含车辆的帧图像。findContours函数的输入为二值图像，黑色为背景，白色为目标，该函数会修改原图像。

具体的实现步骤如下：

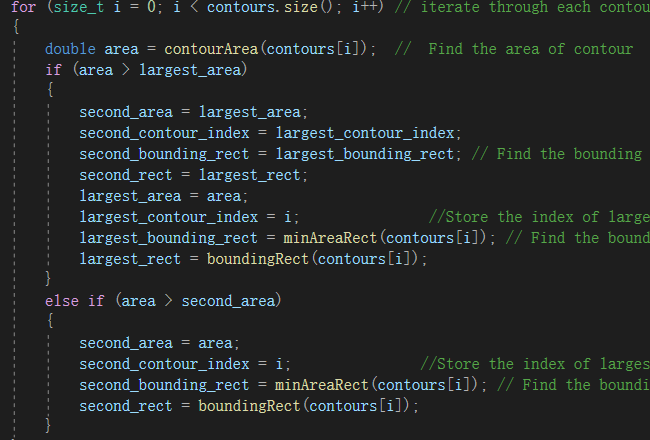
1. 使用findContours函数找到二值化图像中的所有轮廓。



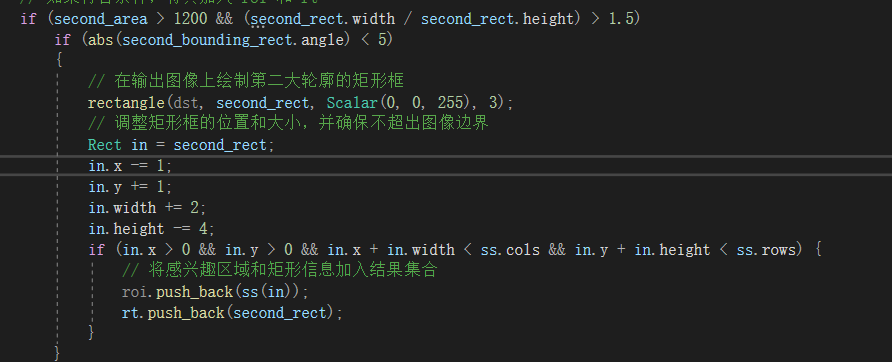
1. 如果没有找到轮廓，那么直接返回，空的图像矩阵



1. 遍历每个轮廓，计算轮廓的面积，更新最大和第二大的轮廓信息。为提取的每个轮廓绘制包围矩形。有2种矩形：boundingRect最小正矩形和minAreaRect最小斜矩形。



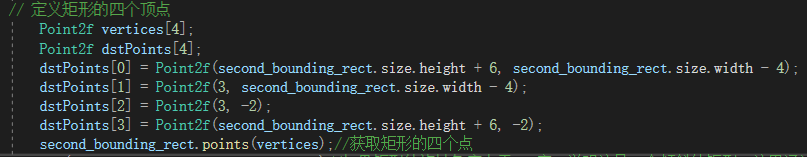
1. 对潜在的符合条件的轮廓进行处理，如果轮廓区域大小超过了设定值，以及长宽比大于设定值，也会加入到roi和rt这两个向量数组中。
2. 为符合条件的轮廓绘制矩形框、调整位置和大小，确保不超出图像边界。



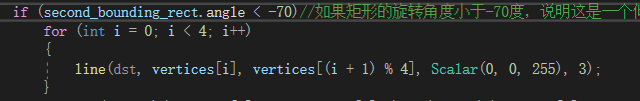
1. 对旋转的车牌进行仿射变换，将其调整为水平状态。

* 获取矩形的四个顶点：

这里second\_bounding\_rect是包围第二大轮廓的最小斜矩形，通过points方法获取了矩形的四个顶点坐标。



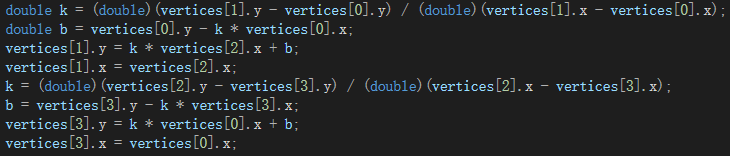
* 根据旋转角度进行绘制：



如果矩形的旋转角度小于-70度，说明这是一个倾斜的矩形，这里通过绘制红色线段来显示出原始矩形的倾斜状态。

* 计算仿射变换矩阵：

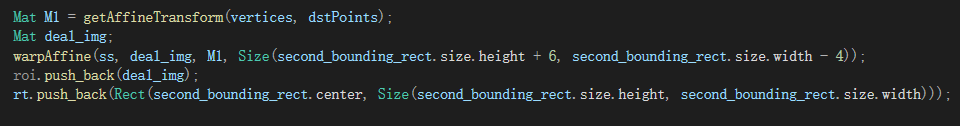
这里通过计算两条对角线的斜率和截距，然后重新设置两个端点的坐标，从而将倾斜矩形调整为水平状态。



* 获取仿射变换矩阵并应用变换：

使用getAffineTransform函数获取仿射变换矩阵，然后通过warpAffine函数将原始图像ss按照计算得到的仿射变换矩阵进行变换，得到调整为水平状态的车牌图像deal\_img。

* 将处理后的图像和矩形信息加入结果集合：



1. 将处理后的图像和矩形信息加入结果集合。



1. 返回最终的结果集合，包含可能的车牌图像和对应的矩形信息。

## 投影法分割字符：

这一步主要是对切割好的车牌图片进行预处理二值化，进行字符分割。字符分割主要是通过投影法实现的。投影法是一种基于图像垂直和水平方向上的像素投影信息来检测文本行或字符的方法。

### 预处理和二值化：

* 将车牌图像转换为灰度图：

车牌图像首先被转换为灰度图像，这是为了便于后续的处理，因为字符的分割通常在灰度图上进行。

* 使用大津法进行二值化：

大津法是一种自适应阈值选择的方法，用于将灰度图二值化，将字符与背景分离。



### 裁剪出包含字符区域的子图像

这里调用cut\_bounder函数，对于图像区域进行分割，裁剪出包含字符区域的子图像，并将结果存储到 dst 中。

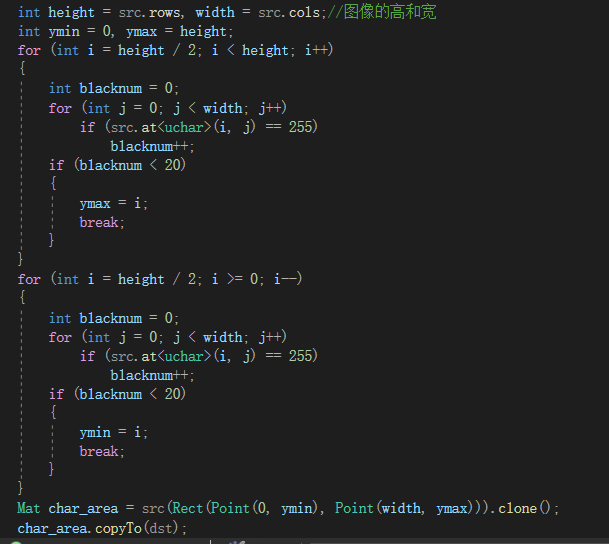
1. **初始化变量：**

获取图像的高 height 和宽 width。初始化变量 ymin 和 ymax，它们表示字符区域的上下边界，默认为整个图像的上下边界。

1. **寻找字符区域的下边界：**
   * 从图像的下半部分向上查找，遍历每一行。
   * 统计当前行中白色像素的数量（像素值为255）。
   * 如果白色像素数量小于20（可调整的阈值），则认为找到字符区域的下边界，更新 ymax 值，并退出循环。
2. **寻找字符区域的上边界：**
   * 从图像的上半部分向下查找，遍历每一行。
   * 统计当前行中白色像素的数量。
   * 如果白色像素数量小于20，认为找到字符区域的上边界，更新 ymin 值，并退出循环。
3. **裁剪字符区域：**
   * 使用 Rect 构造函数，基于计算得到的上下边界，定义一个矩形区域。
   * 使用 clone 函数从原图 src 中复制该区域，确保 dst 中存储的是独立的图像。
4. **输出结果：**

将裁剪得到的字符区域存储在 dst 中，函数执行完毕。

源代码如下：



### 垂直投影：

通过垂直投影直方图分析，找到字符区域的边界，并将字符分割成单个图像。最后返回包含分割字符图像的 **vector**。

1. **反色处理：**

将输入的二值化图像 srcImg 进行反色处理，将黑色背景变为白色，白色字符变为黑色。

1. **垂直投影直方图计算：**

创建一个数组 projectValArry 用于储存每一列的白色像素个数。

遍历图像的每一列，统计白色像素的数量。

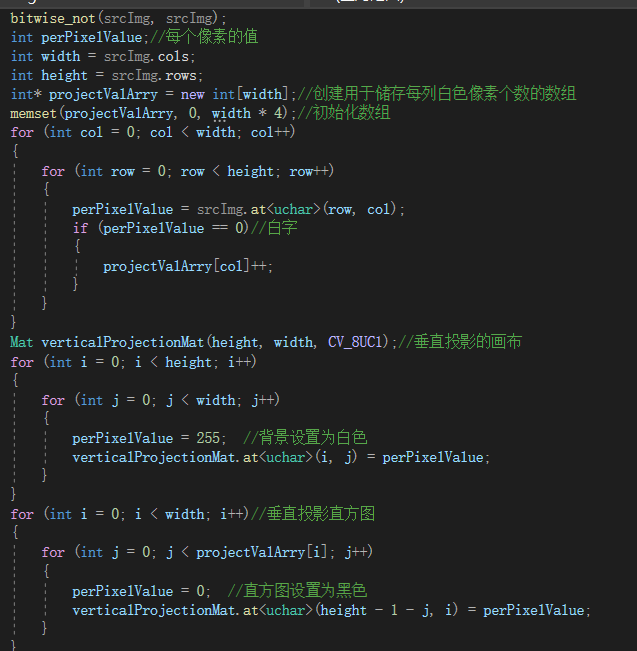
1. **创建垂直投影画布：**

创建一个大小与原图相同的画布 verticalProjectionMat，用于绘制垂直投影直方图。

将整个画布初始化为白色（像素值为255）。

1. **绘制垂直投影直方图：**

根据每一列的白色像素数量，在 verticalProjectionMat 上绘制相应的黑色直方图。



1. **字符分割：**

创建一个空的 vector<Mat> roiList 用于储存分割出来的每个字符。

使用变量 startIndex 和 endIndex 记录字符区域的起始和结束索引。

使用 inBlock 标志位记录是否在字符区域内。

遍历垂直投影直方图，根据白色像素的存在与否，确定字符区域的边界。

将字符区域提取为 roiImg。

对提取的字符区域进行处理，确保它符合一定的条件，如宽度大于图像宽度的1/16或者字符间距不太窄。

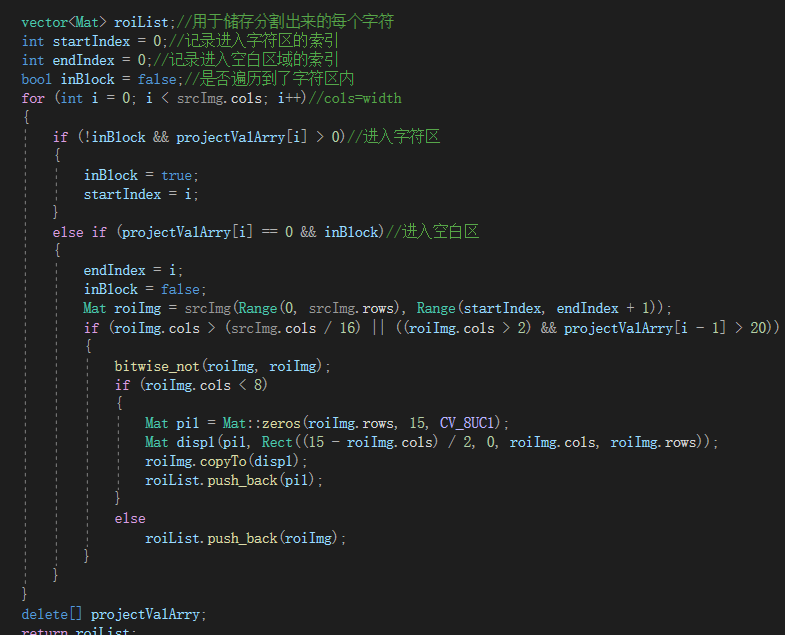
将处理后的字符区域添加到 roiList 中。

1. **释放内存：**

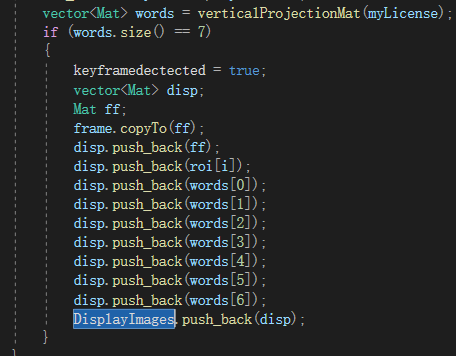
释放动态分配的数组内存。

1. **返回结果：**

返回储存分割出的字符图像的 roiList。



### 将投影得到的子图像存到DisplayImages：

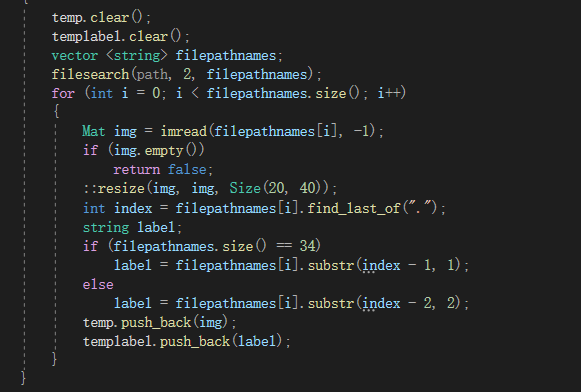


将分割好的子图像存入到DisplayImages中，方便后续的车牌识别。

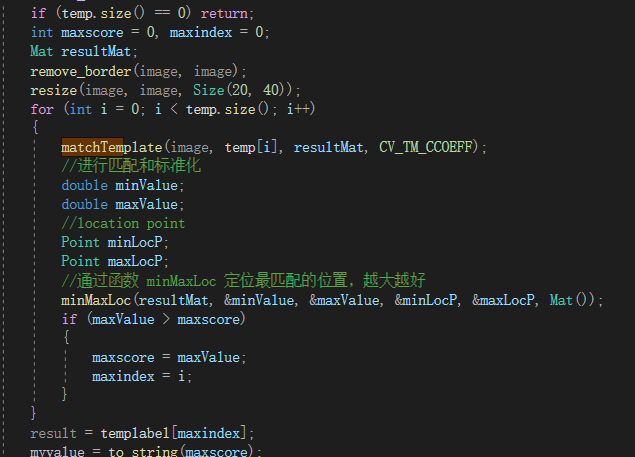
## 字符识别：

### 模板匹配

1. **载入模板：**将模板库中的字符模板和数字模板加载，存入到temp数组中。



1. **去除边框和调整大小**：通过调用remove\_border函数去除图像边框，然后使用resize函数将图像调整为大小为20x40。
2. **遍历模板集合**：通过一个for循环，遍历模板集合中的每个模板。
3. **模板匹配**：对于每个模板，使用OpenCV的matchTemplate函数在输入图像上进行匹配，得到匹配结果矩阵resultMat。
4. **寻找最大匹配得分**：通过minMaxLoc函数，找到匹配结果矩阵中的最大得分以及对应的位置。
5. **更新最大匹配得分和索引**：如果当前模板的匹配得分大于之前的最大得分，更新maxscore和maxindex。
6. **获取匹配结果**：在循环结束后，根据最大得分对应的模板索引，获取对应的字符标签，存储在**result**中。
7. **附加信息**：根据匹配得分maxscore，将其转换为字符串形式，并存储在myvalue中。
8. **返回结果**：最终，字符识别的结果以字符串形式存储在result中，而匹配得分则存储在myvalue中。



### SVM向量机

支持向量机（support vector machines，SVM）是一种用来分类的机器学习算法，对字符分类要用多类分类的SVM方法。

实现的方法如下：

1. 初始化：

* 创建一个SVM（支持向量机）模型对象，并设置其参数。具体参数设置如下：

setType(SVM::C\_SVC)：设置SVM类型为C-Support Vector Classification。

setKernel(SVM::LINEAR)：设置核函数类型为线性核。

setDegree(0)：设置多项式核函数的阶数为0。

setGamma(1)：设置核函数的参数。

setCoef0(0)：设置核函数的独立项。

setC(1)：设置SVM的惩罚参数。

setNu(0)：设置SVM的Nu参数。

setP(0)：设置SVM的epsilon参数。

setTermCriteria(CvTermCriteria(CV\_TERMCRIT\_ITER, 1000, 0.01))：设置SVM的停止准则，包括迭代次数和epsilon。

* 初始化HOGDescriptor对象：

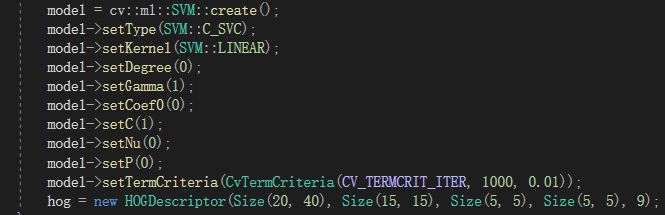
Size(20, 40)：设置窗口大小为20x40。

Size(15, 15)：设置块大小为15x15。

Size(5, 5)：设置块步幅为5x5。

Size(5, 5)：设置细胞大小为5x5。

‘9’：设置HOG描述符的维度。

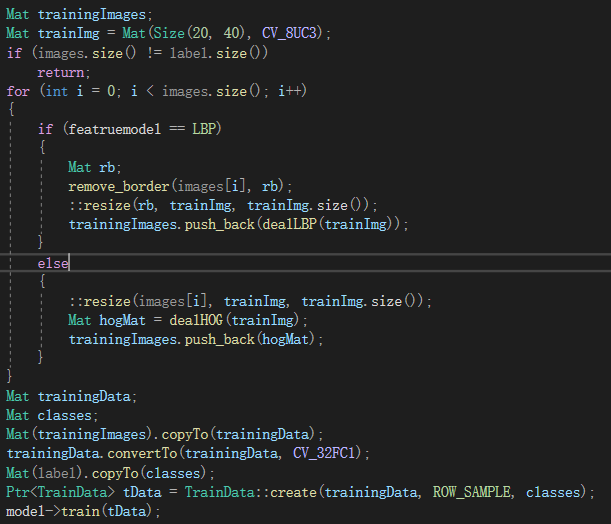


1. 训练：

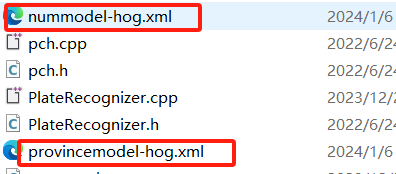
* **初始化变量：**创建Mat对象trainingImages用于存储训练图像的特征表示。
* **循环处理图像：**对输入的图像集合进行循环处理，每次处理一个图像。

调用remove\_border函数去除图像边框，然后将其调整为20x40的大小，并使用dealLBP函数提取LBP特征。将提取的特征添加到trainingImages中。

* **数据格式转换：**将trainingImages转换为Mat对象trainingData，并将其数据类型转换为CV\_32FC1，以适应SVM的训练要求。
* **构建类标签：**将输入的标签集合label转换为Mat对象classes。
* **创建训练数据对象：**使用TrainData::create创建训练数据对象，其中包括特征数据和相应的类标签。
* **训练模型：**调用SVM模型的train函数，使用训练数据对象进行模型训练。

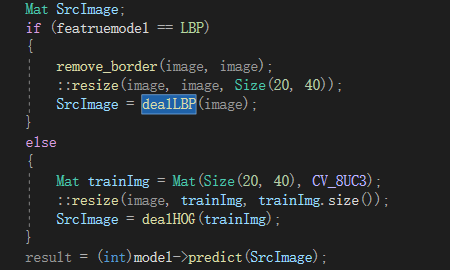


训练的模型会存储为.xml文件，存储在文件夹中，在使用深度模型框架的时候调用。



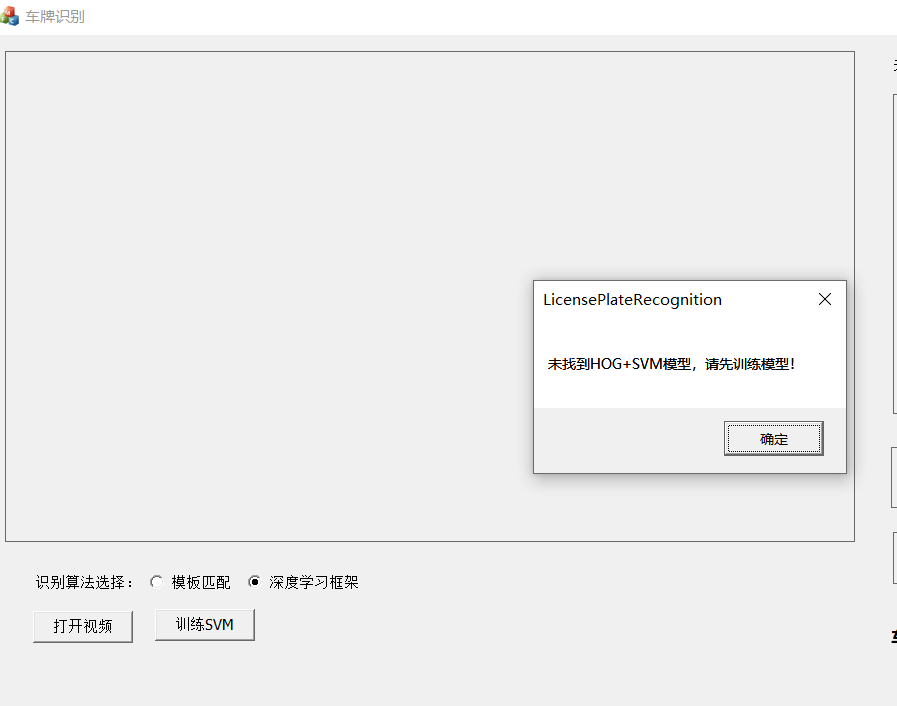
1. 预测：

将待检测模型用LBP或者HOG的模型进行预测，提取特征，进行预测。



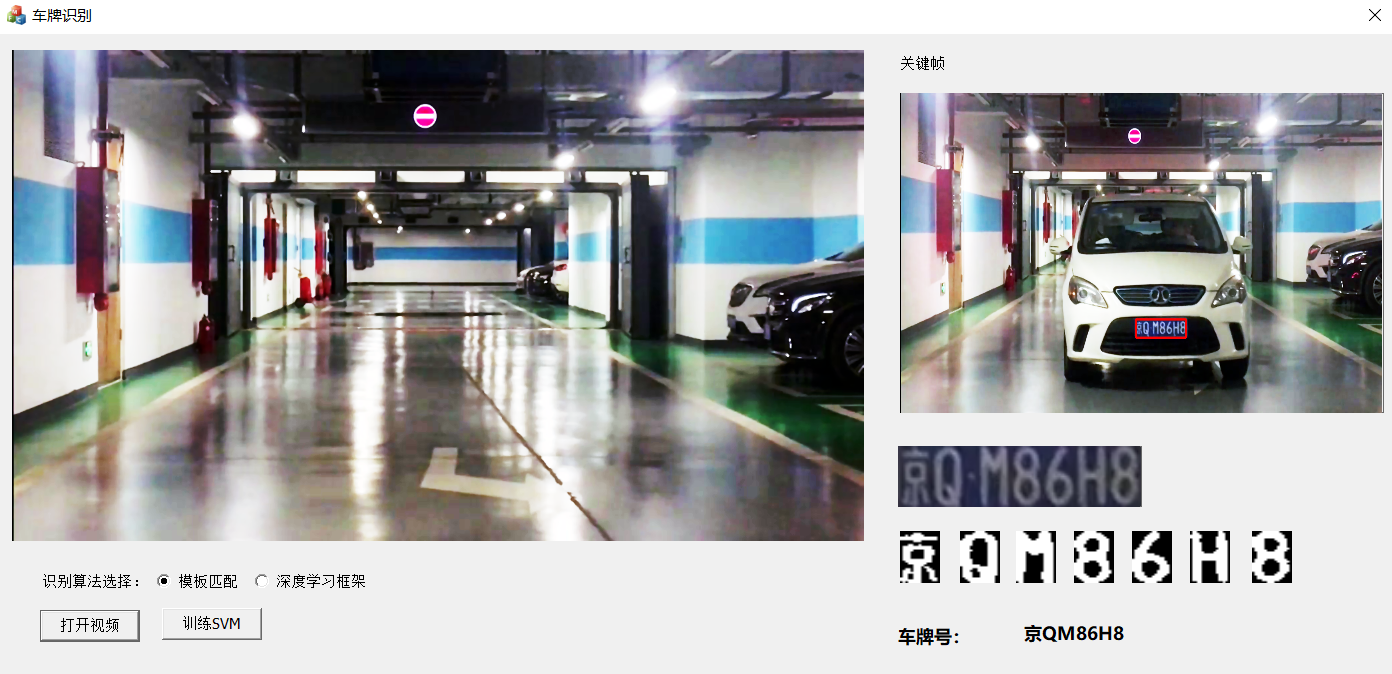
# 实验结果:

如下图所示：



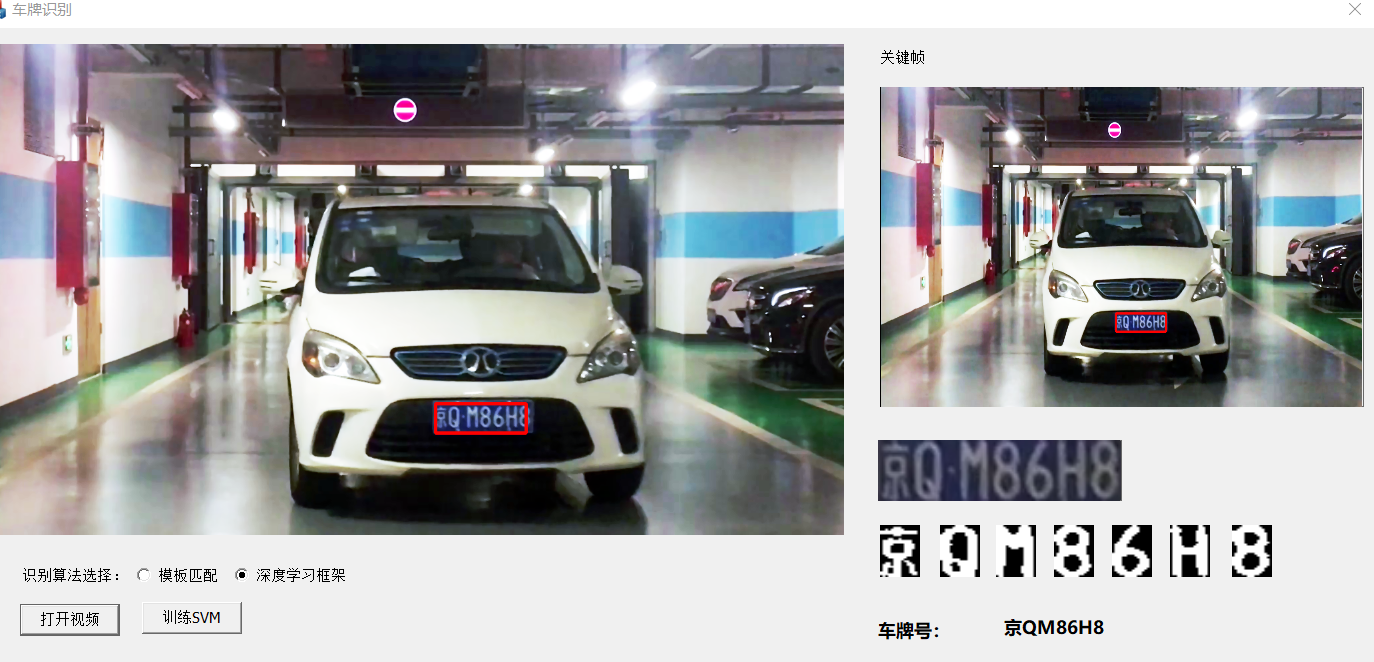
如果在指定文件夹中没有训练好的模型文件，会出现如上报错。

此时我们可以点击训练SVM，，在程序自动执行之后，就可以正常的利用深度学习框架进行训练了。





使用模板匹配实现的。





使用深度学习框架实现的

# 总结：（难点以及解决方案）

1. **问题即解决方案：**首先是完成整个任务需要学习很多相关的知识。就比如在切割车牌区域这一部分，我学习Rotaterect预定义变量的使用方法，这里由于最初不熟悉卡了很长时间，在截取时车牌区域时出现坐标为负数，或超出图像范围的异常，在学习了网上如何避免裁剪图块时出现这些异常，为旋转矩形的交点坐标进行一次重新赋值，由工程中的Safecontours函数实现。在偏斜变换时也是由于相关几何知识了解不够导致结果不太好，参考了很多资料修改代码才成功。
2. **问题：**此次课设实践过程中遇到了不少异常，图片大小和通道问题最常见，一般发生在处理过程中没有考虑一些特殊情况或方法调用时给了错误的图片形式。

**解决方案：**是不断调试，找到出问题的代码部分，为特殊情况编写特殊处理if语句，或找到产生不符合条件的图像的原因，修改之前的处理代码。这次课设与以往不同的是，每一个处理步骤几乎都依赖于前一个步骤的结果，环环相扣，所以在实现每一个功能时都要仔细测试，逻辑中往往前一个处理结果的true与false也是后一个步骤是否进行或者怎样进行的前提条件，因此逻辑的完整性与每一步结果的准确性都对结果造成了巨大的影响。通过不断地调试，测试多个视频文件，几乎做到了对大部分视频都有比较准确的识别结果。工程里的逻辑也经过严格推敲，确保不会因某一步的处理不当造成异常情况发生或影响最后结果。

1. **问题：**实现svm支持向量机较为繁琐在实现利用的时候，我本来还想着一些变量和函数全部都自己定义实现。

**解决方案：**学习了OpenCV官方文档中，发现了在这个函数库中有相应的API实现，我可以调用官方的函数库，大大简化了代码难度。

1. **问题：**关于颜色定位。

**解决方案：**首先是排除利用RGB分量判断，蓝色属性在255，只要另外两个分量不为0，最后得到的不是蓝色，而可能是青色或其他颜色。这还只是区分蓝色的问题，黄色更麻烦，它是由红色和绿色组合而成的，这意味着需要考虑两个变量的配比问题。非常复杂，故不用。

而采取用hsi 的色彩模型可以有效的解决这个问题。简化了判断方法，也简化了实现难度。

1. **问题：**关于预处理图像之后识别不准确的问题

**解决方案：**在这里我一直在不断的利用不同的形态学操作处理图像，修改形态学操作的参数，多次实验之后得到了比较准确的分割出来的图像。