基于 SVM 的车牌图像识别

1. 实验内容

设计一个系统对车牌图像进行识别

2.实验平台与开发环境

笔者采用 Microsoft Visual Studio 2010 与 OpenCV2.4.10 在 Windows 10 预览版下开发完成，并在 Windows 7 与 8 下测试可用。

3.实验内容

3.1 实验步骤：

主要流程为：

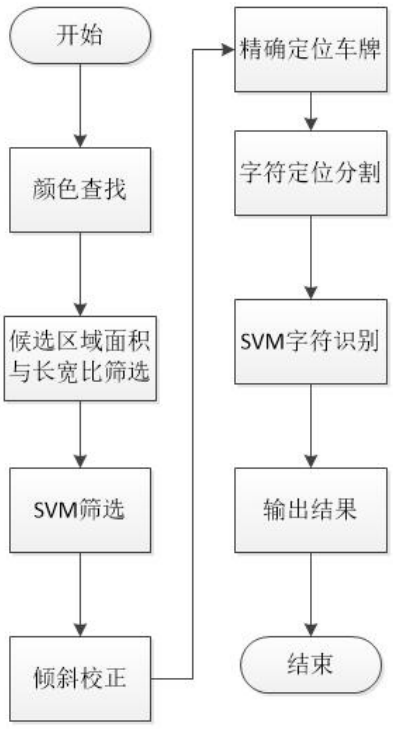

图 1 主要流程

3.1.1 颜色查找

一般说来，对于一幅含有车牌的图像，人眼识别时首先是根据颜色特征定位到了车牌区域。而HSV颜色模型最为贴近的描述人眼对于颜色的感受特征，将车

牌颜色特征定量描述下来，并选取一定的误差阈值对图像进行检测则可以将颜色特征区域找到。常见车牌颜色为蓝底白字和黄底黑字，本实验主要考虑蓝底白字车牌。

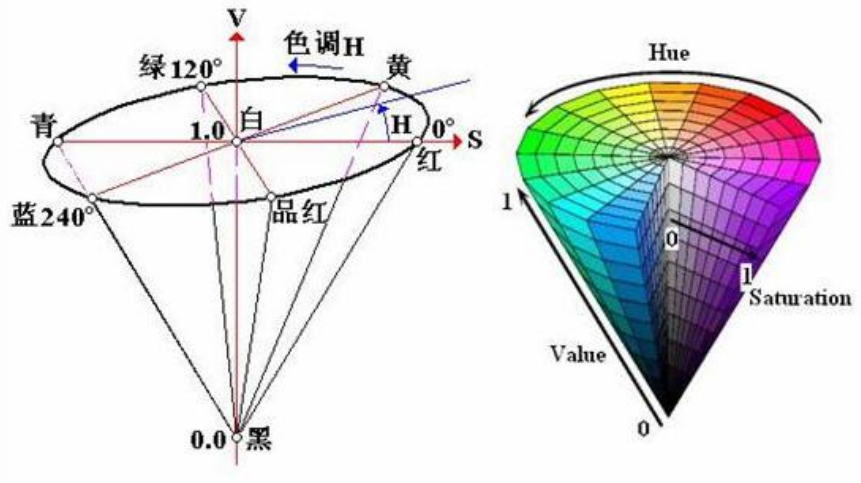

图 2 HSV 模型

HSV 颜色模型如上所示，有三个分量：H(色调)、S(饱和度)、V(亮度)，其中 H 为影响颜色的主要分量，其范围是 0-360,利用 HTML 色彩提取利器人眼大致判断出蓝颜色所取得 H 范围值为 200-280，S 与 V 范围值为 35%-100% ，图 3 位 HTML 色彩提取利器所示界面。

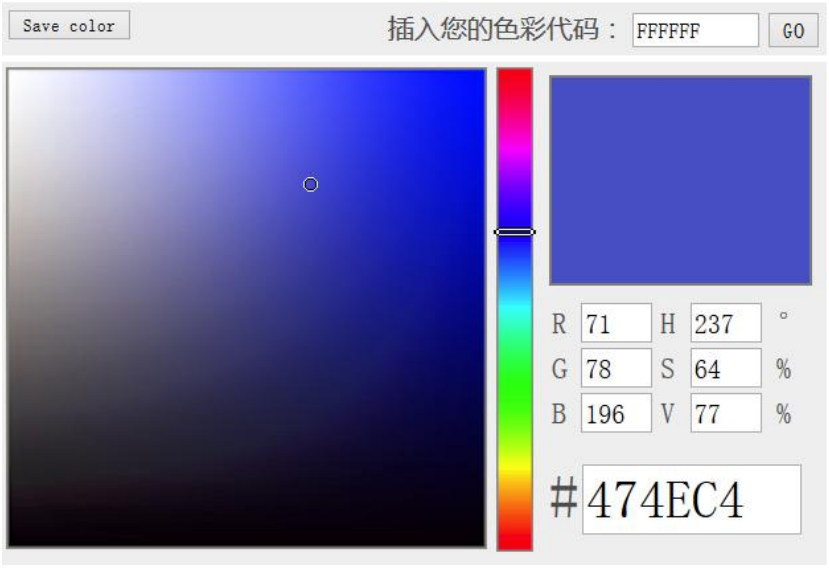

图 3　HTML 色彩提取利器

相应的实现步骤及细节为：

1.读取图片，将图像的颜色空间从 RGB 转为 HSV，并添加直方图均衡化处理来减少光照的影响；

2.依次遍历图像的所有像素，当 H 值落在 200-280 之间并且 S 值与 V 值也落在 0.35-1.0 之间，标记为白色像素，否则为黑色像素； opencv 为保证 HSV 分量都在 0-255 区间，对相应分量做了变换处理，编程现实时 H 范围为 100-140，S、V 对应范围 64-255。处理效果图如下所示：

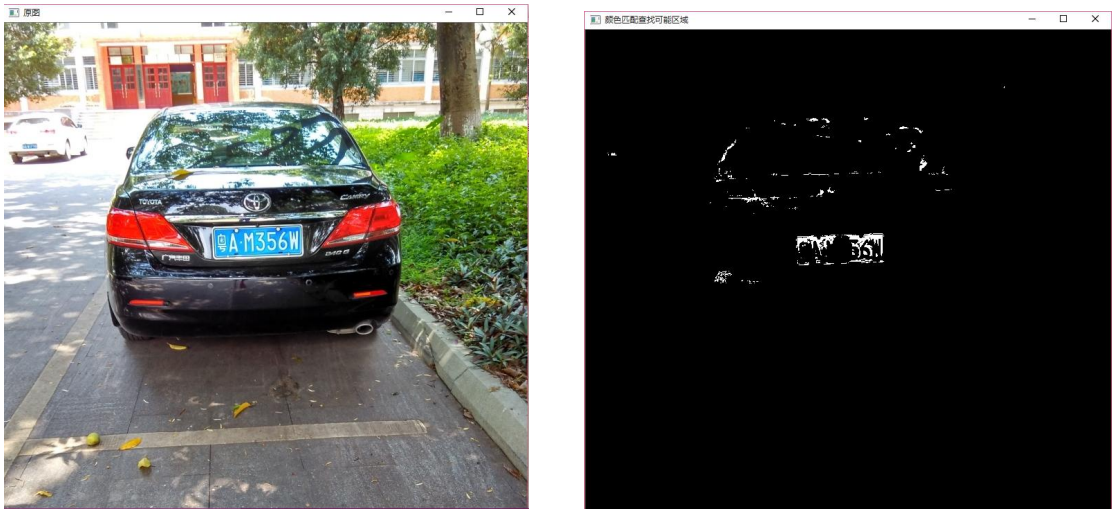

图 4　颜色特征查找效果

3.1.2 候选区面积与长宽比筛选

通过上一步只是查找到特征蓝色所在的区域，而由于车牌中有不是蓝色的字符使得车牌区域不容易提取，为方便将车牌完整的分割出来，对上一步的图像进行 sobel 处理、高斯模糊、形态学闭操作后，找出各个轮廓的最小包围矩形，车牌区域就在这些最小包围矩形之中。

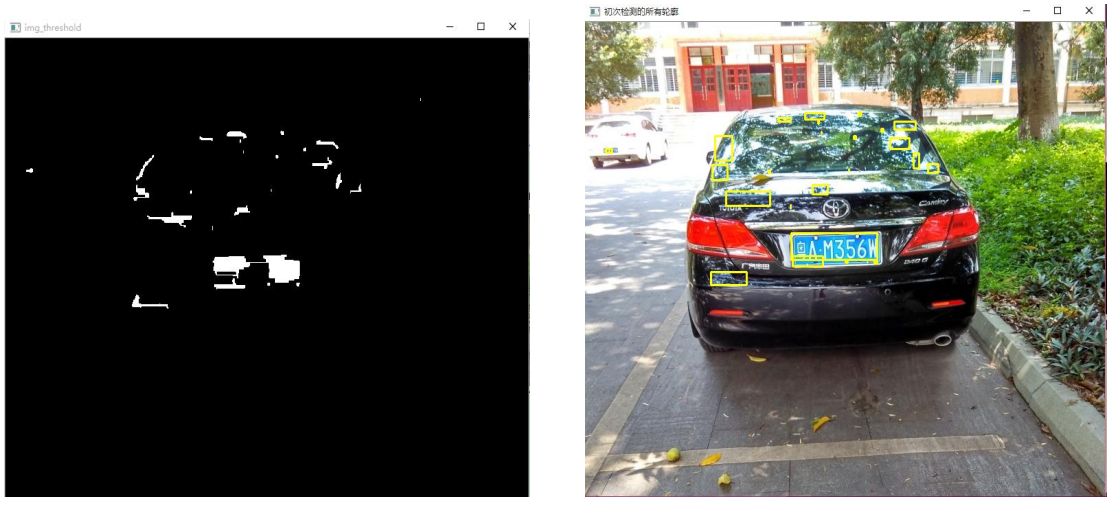

图 5 候选区域 1

但这样找出来的矩形数量相当之多，所示图片中的矩形就有 25 个。由于车牌的长宽比是固定的，假若车辆距离摄像头的距离也是固定的则拍摄图像中车牌的面

积也是固定的，考虑车辆拍摄时的距离会有一定变化，选取了一定的变化阈值。根据这两个特征对找到的矩形一一进行条件判别，则可以排除大量的非车牌区域。如下图所示，矩形有 25 个减少到了 3 个。为方便后面的识别，将得到的候选区域分割出来进行简单的预处理，图 6 为处理后的效果图。

图 6 候选区域 2

3.1.3 SVM 筛选

通过 svm 训练车牌图像和非车牌图像后，对候选图像进行判别则可以找到车牌。svm 实现细节为：采用了 opencv 封装好的 svm 算法，将样本图像转化成相应的要求格式进行训练，并将训练结果以 xml 文件形式保存。所采集的特征直接是样本的像素值，将统一大小的二值化矩阵转成一个一行的向量作为样本的特征向量。其中样本来自于对自己所采集的车牌图片的分割，标签分为两类（1 和 0 分别代表是否为车牌），所采集的样本数量比较少，特征也很简单，但从识别结果来看，效果比较好。

图 7 车牌样本

对候选区域所筛选出的结果为： 

3.1.4 倾斜校正

由于拍摄角度及路面等因素，车牌并不一定是正的，必须将车牌进行倾斜校正，算出车牌的倾斜角度，然后按照此角度对图像进行旋转。实现细节为：在第一步颜色查找的图中找出车牌的最小包围倾斜矩形，求出此倾斜矩形的倾斜角度，然后将原图旋转此角度，在相同区域分割出车牌。实现效果如下图所示：

图 8 倾斜校正 1

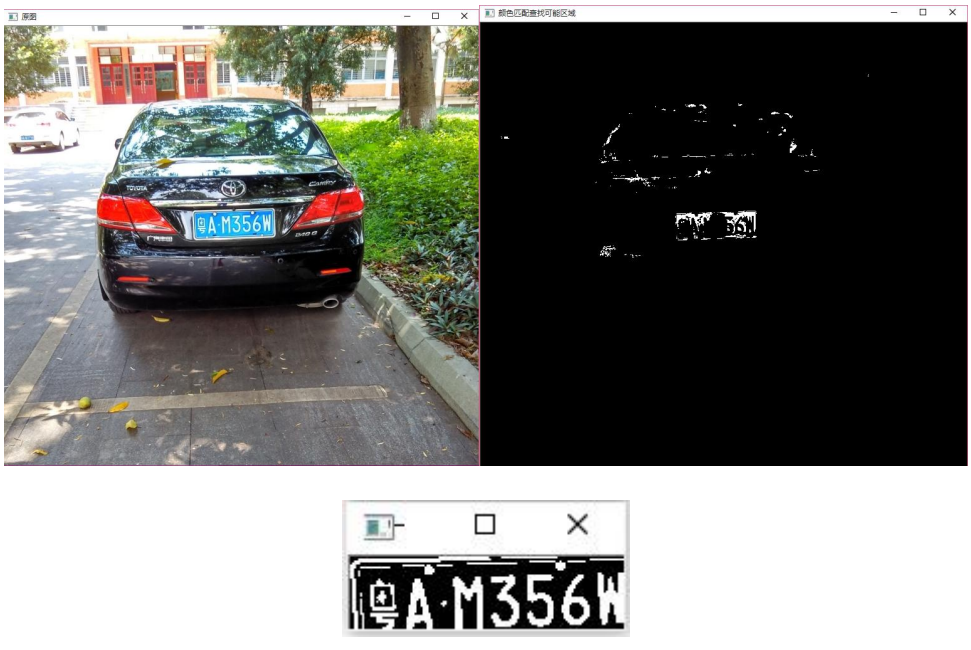

图 9 倾斜校正 1

3.1.5 精确定位

可以看出经过倾斜校正后得到的车牌仍然含有车牌四周的边框部分，需要采用一定方法将边框去除掉。水平方向上看，字符所在的区域白色像素点较多，且每一行所在的白色像素和值较为连续，与车牌边界接触时白色像素和值会有一个较大的下降值，由此可以从中间行开始分别往上和往下扫描判断每行的白色像素和的值，当它第一次小于某个和阈值时认为是分割行。对于左右边框，经观察大部分图像的左右边框位置较为固定，直接进行了固定位置的切割。将切割后的区域作为ROI放在新图片的中间即得到精确定位后的车牌。效果如下所示：

图 10 精确定位

3.1.6 字符分割

对精确得到后的车牌进行字符分割，这里采用的是寻找连通域的包围矩形，可以知道字符所在的包围矩形为前 7 个最大的矩形，找出前 7 个最大的矩形即可以分割出字符。在寻找之前考虑到第一个汉字可能会分割成上下两个部分，在这里进行了一次垂直方向上的膨胀操作。由于 opencv 利用轮廓包围矩形时并没有标记顺序，切割出的矩形无法对应字符顺序，在程序中添加了一个对矩形横坐标比较的功能，按照横坐标从小到大输出从左到右的字符。

图 11 字符分割

3.1.7 SVM 字符识别

得到分割后的字符后，对每个字符进行 SVM 分类判别。在此之前同样进行了

SVM 的样本特征训练，样本来自于网络资源，共 34 类，不含汉字，且数字 0 与字母 O，数字 1 与字母 I 分别作为一类来处理。

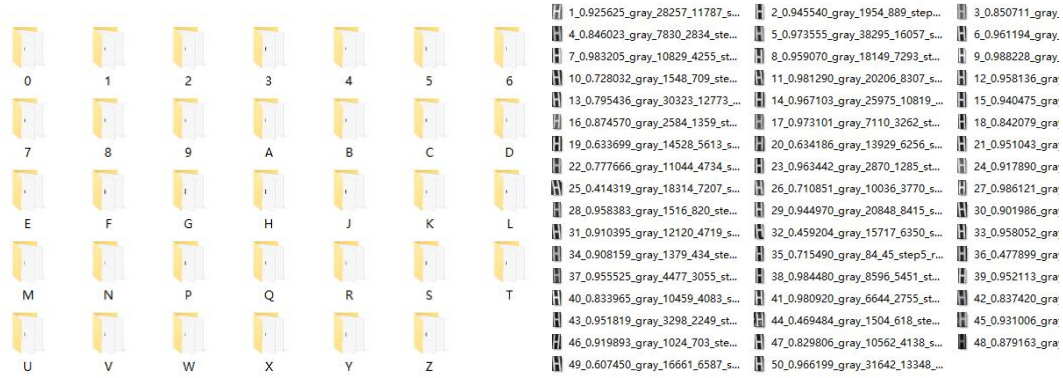

图 12 字符样本

对于字符的特征提取采用的是 hog 特征点提取，HOG 即 histogram of oriented gradient，是用于目标检测的特征描述子，该技术将图像局部出现的方向梯度次数进行计数，该方法和边缘方向直方图、scale-invariant feature transform 类似，不同的是 hog 的计算基于一致空间的密度矩阵来提高准确率。Navneet Dalal and Bill Triggs 首先在 05 年的 CVPR 中提出 HOG，用于静态图像 or 视频的行人检测。在 opencv2.2+版本里面已经实现，封装在 HOGDescriptor 类里。HOG 特征提取方法的实现步骤为：

1. 灰度化（将图像看做一个 x,y,z（灰度）的三维图像）；
2. 划分成小 cells；
3. 计算每个 cell 中每个 pixel 的 gradient（即 orientation）；
4. 统计每个 cell 的梯度直方图（不同梯度的个数），即可形成每个 cell 的 descriptor；

在 opencv 中创建一个 HOG 描述子的格式为：

```
HOGDescriptor *hog=new
HOGDescriptor(cvSize(28,28),cvSize(14,14),cvSize(7,7),cvSize(7,7),9);
```

其相应参数为：

检测窗口大小为 28*28

Block 大小为 14*14；

Cell 大小为 7*7；

Block 在检测窗口中上下移动尺寸为 7*7；

1 个 cell 的梯度直方图化成 9 个 bin；

代码中的一个 hog 描述子是针对一个检测窗口而言的，所以一个检测窗口共有((28-14)/7+1)*((28-14)/7+1)=9 个 block；一个 block 中有(14/7)*(14/7))=4 个 cell，而一个 cell 的 hog 描述子向量的长度为 9；所以检测窗口的 hog 向量长度=9*4*9=324 维。

得到样本 HOG 特征进行 SVM 训练，并保存训练结果为 HOG_SVM.xml。然后对分割出的字符一一进行识别，并输出结果。

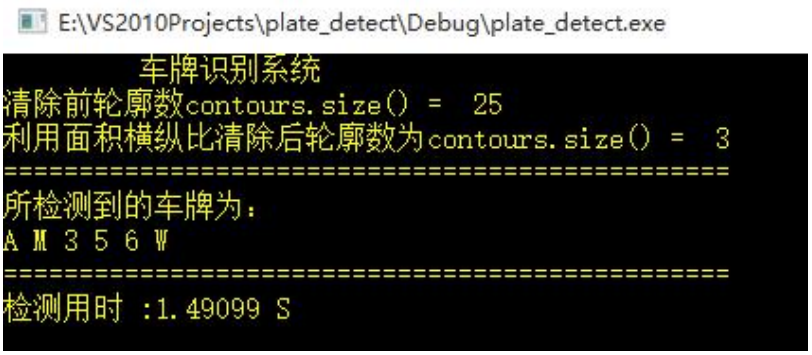

图 13 识别结果

4 实验效果

搭建 MFC 的 GUI，运行效果如下，对于拍摄角度合适的车牌图像能够很好地识别，但由于识别时按照颜色匹配来寻找车牌的，对于蓝色车牌并不能有效识别，对于一些拍摄视角非正向面对车牌的图片也不能识别，需要后续进行改进。

图 14 GUI 界面

5 问题与改进

1.为了方便调试程序，最开始并没有制作 MFC 的界面，调试好了之后将程序转化到 MFC 中，出现了图中所示问题。

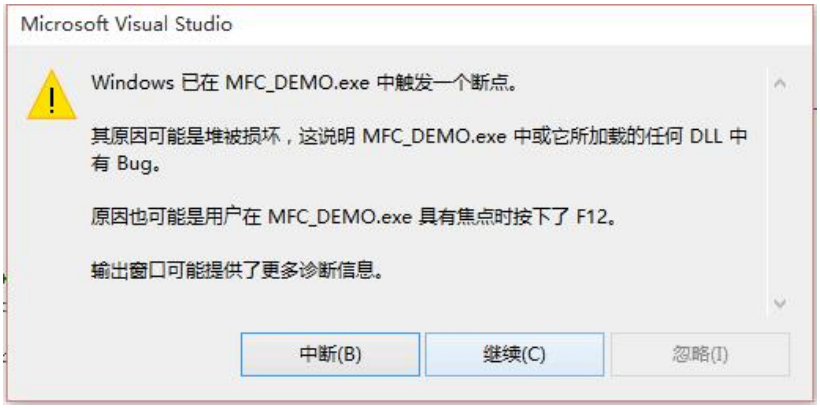

图 15 调试错误

找了很久才发现是因为在程序中使用了 vector 容器导致的崩溃，网络上找到了一个解决方法。vs 编译器没有配置好，debug 环境下：配置属性-》常规-》mfc 中使用-》共享 dl；C++-》代码生成-》运行库-》多线程调试 dll（MDd）。按照此方法进行修改配置后，即可正常运行。

2.由于整个识别是基于颜色识别的，对于蓝色车并不能有效识别，如下图所示并不能找到车牌，需要对程序进行该进，利用 SVM 判断检测不到车牌时利用另一种方法来找出车牌。

图 16 蓝色车识别错误

3. 由于训练样本时样本是正的，对于倾斜的车牌若无法校正准确会导致字符识别出错，甚至无法分割出字符，需要对倾斜校正部分的程序进行该进。

图 17 倾斜校正