目录

车牌定位与识别

一、 系统简介

本系统用于在图片中定位车牌位置并进行车牌号码的识别。

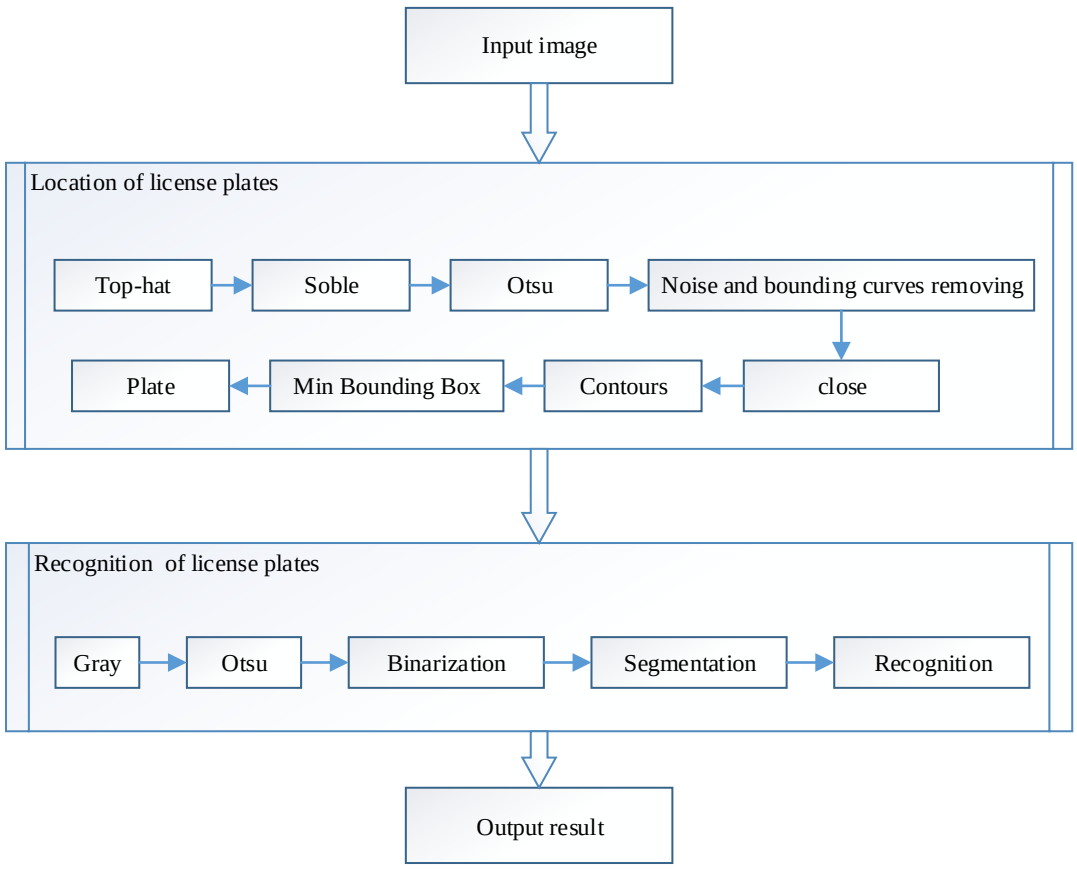
系统用 C++.NET 实现，开发工具选用 Visual Studio 2010，平台为 .NET Framework 4.0。系统所处理的图片为 400*300 像素 JPG 图片，角度为车辆正前方/正后方或者车辆稍微倾斜。车牌为常规的 92 式车牌样式，车牌的背景色皆可。

系统的输入是通过界面的打开图片按钮，选择图片作为系统处理的对象；系统的输出为车牌号码识别的结果，作为文本显示在最后一个按钮上。

系统界面及处理过程如下图所示：

二、 系统流程

整体系统实现的流程如下：



三、 系统参数和数据格式

系统需要的数据分为三类：矩阵表示图像数据、最小包围盒、大小参数。

图像数据中，RGB 位图数据用 Byte[行][列][RGB]三维数组来表示图像三维矩阵；灰度图的为徒数据用 Byte[行][列]二维数组表示图像二维矩阵；二值图也用 Byte[行][列]二维数组表示图像二维矩阵，取 0 值表示黑色，取 255 值表示白色。而图像本身的宽度和高度信息另有辅助变量表示。系统中使用的图像数据有 bmp_src_rgb[MAX_HEIGHT][MAX_WIDTH][3]、bmp_src_gray[MAX_HEIGHT][MAX_WIDTH]等。

最小包围盒数据包括{ Xmin, Xmax, Ymin, Ymax, Number}，分别表示某个连通区域的最小 X 坐标、最大 X 坐标、最小 Y 坐标、最大 Y 坐标、识别的结果数值。如 minBoundingBox[100][5]。

大小参数如 MAX_HEIGHT、MAX_WIDTH、MAX_PLATE_WIDTH、MAX_PLATE_HEIGHT，分别限定了系统处理的对象的一些阈值。系统中使用到的全部参数设定如下表。

参数值/阈值	意义
MAX_WIDTH=400 MAX_HEIGHT=300	加载的图片的最大宽度和高度
MAX_PLATE_WIDTH=150 MAX_PLATE_HEIGHT=50	对于设定的最大图片，所处理的最大车牌的宽度和高度
X=120 Y=240	在数字识别时，将数字库的特征 X 向量设置为 120 元，特征 Y 向量设置为 240 元；分割的数字样本也相应调整至(240,120)矩阵
Soble_X[3][3]={-1,0,1,-2,0,2,-1,0,1}	Soble 滤波中选用的列向量算子
[1/20,1/6]	有效边缘的竖直方向的长度； 竖直长度低于图像总行数的 1/20 的边缘是随机噪音； 而高于图像总行数的 1/6 的边缘则是背景线条，如车身线条
[1/30,1/7]	车牌中每个数字的宽度所占车牌宽度的比例范围
[0.4,0.7]	每个数字的宽度与高度的比例范围
[0.3,0.75]	每个二值化数字的绘制像素点占数字区域的面积比例范围
[1/4,3/4]	所截取的车牌的[1/4,3/4]的宽度范围和[1/4,3/4]的高度范围，此部分属于车牌本身，不会含有因车牌略倾斜而引起的车牌区域图片含有车身背景区
0.4	车牌本身区域中前景所占的面积比； 低于车牌面积 0.4 的属于前景，即字符的颜色； 高于 0.4 的属于背景，即车牌的颜色

四、 系统基本函数介绍

函数：void Bitmap2Byte(Bitmap^,Byte [][][])
参数：24位位图，矩阵
作用：将24位的位图转为RGB的图像矩阵

函数：Bitmap^ rgbByte2Bitmap(Byte [][][],int,int)
参数：矩阵，列数，行数
返回值：24位的位图
作用：将RGB的图像矩阵转为24位的位图

函数：Bitmap^ grayByte2Bitmap(Byte [][],int,int)
参数：矩阵，列数，行数
返回值：24位的位图
作用：将灰度的图像矩阵转化为24位的位图

函数：void seg_sort(int,int [][])
参数：最小包围盒的有效个数，包围盒
作用：对最小包围盒数据结构进行排序，按照Xmin升序

函数：void out(char*,Byte [][],int,int)
参数：文件名，矩阵，矩阵的列数，矩阵的行数
作用：将矩阵写入指定文件中

函数：void in_f(int [][][])
参数：矩阵
作用：加载数字库文件至矩阵

函数：int max_4(int, int, int, int)
参数：四个整型数
返回值：最大值
作用：获取四个整型数值中的最大值

函数：int max_6(int, int, int, int, int, int)
参数：六个整型数
返回值：最大值
作用：获取六个整形数据中的最大值

五、 车牌定位

1. Gray

Gray 过程，将 RGB 图像数据的三位矩阵转化为灰度图像数据的二维矩阵。本系统中采用的灰度化公式为：Y=0.299*rgbRed+0.587* rgbGreen+0.114*rgbBlue。处理过程如下：

```
灰度图像数据的二维目标矩阵置零
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        目标矩阵[][]=0.299*RGB 矩阵[][][0]+0.587*RGB 矩阵[][][1]+0.114*RGB 矩阵[][][2]
    结束
结束
```

RGB 图像的灰度化效果如下图所示，左图为 24 位原图像，右图是 8 位灰度图像。

2. Top-hat

Top-hat 过程在灰度图像数据的矩阵上进行，是一种数学形态学变换，属于非线性滤波，用于消除复杂背景图像的噪音，并提取图像的明亮边缘。Top-Hat 的性能取决于其滑动窗口的宽度。本系统采用的是 3*3 的滑动窗口。

Top-Hat 公式是：Top-Hat(A)=A-(A◦B)=A- (A $\ominus$ B) $\oplus$ B，即 Top-Hat(A)=A-（A 先进行腐蚀操作，再进行膨胀操作的结果）。Top-Hat 处理的过程如下：

```
灰度图像数据的二维目标矩阵置零

//先腐蚀
```

```
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        目标矩阵[][]=以当前位置为中心的 3*3 滑动窗口中像素最小值
    结束
结束

//再膨胀
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        目标矩阵[][]=以当前位置为中心的 3*3 滑动窗口中像素最大值
    结束
结束



//求帧差
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        目标矩阵[][]=取绝对值(原矩阵[][]-目标矩阵[][])
    结束
结束
```

对上图中右侧的灰度图进行 Top-Hat 处理后的图像如下所示，其原图中的边缘被提取出。

3. Soble

Soble 滤波，用于获取特定边缘信息。本系统的实现是基于车牌中字符的竖直排列特征的，所以在获取的图像边缘图中，要继续进行滤波，以保留系统后续需要的竖直边缘信息。系统选中的 Soble 算子是 Soble_X[3][3]={-1,0,1,-2,0,2,-1,0,1}。

处理过程如下：

```
灰度图像数据的二维目标矩阵置零
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        temp=以当前位置为中心的 3*3 滑动窗口中像素与 Soble 算子的卷积
        如果 temp 大于 255，则 temp=255
        如果 temp 小于 0，则 temp=0
        目标矩阵[][]=temp;
    结束
结束
```

如下图所示，边缘中的竖直边缘被保留下，而水平方向的边缘被清除掉了。

4. Otsu

在车牌识别使用的图像中，拍照的时间、地点、角度等不同导致每张图像的整体明亮度不同，所以固定的二值化阈值无法满足，因此需要自适应的局部二值化方法。

系统采用的是 Otsu 大津法，即最大类间方差法。最大类间方差法按照图像的

灰度特性，将图像中像素分为前景和背景，而前景与背景之间的类间方差越大则分割效果越好。Otsu 二值化分为计算阈值和图像二值化两部分。

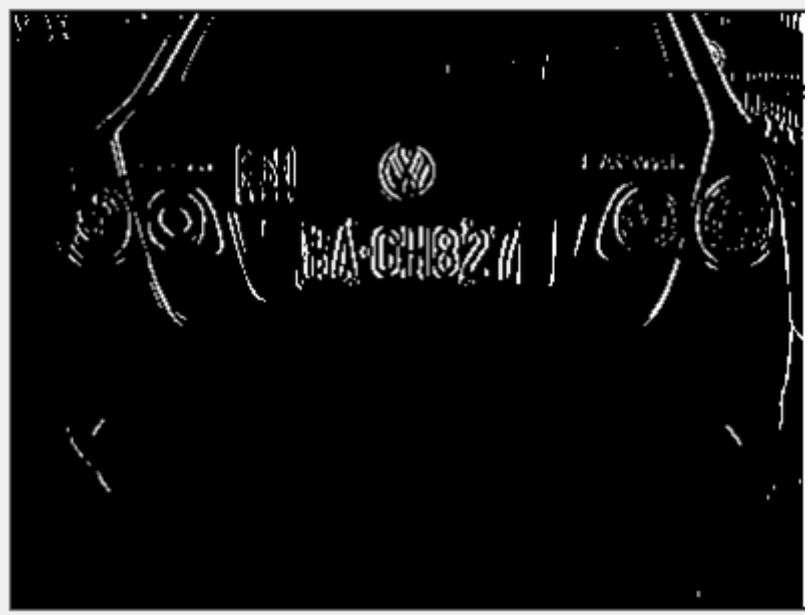
阈值的计算过程如下：

```
灰度图像数据的二维目标矩阵置零
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        对灰度值(原矩阵[][])进行累计
    结束
结束

//计算从 0 到当前数值的均值和概率分布
对 256 色的每一各颜色值
    此数值的像素数占总体的比例=此数值的像素数/图像总像素数
    图像均值+=此数值像素数占总体的比例*此数值

    此数值的概率分布+=此数值像素数占总体的比例
    此数值为止值的像素灰度均值=图像均值/从 0 到此数值的像素的累积比例
结束

初始化阈值为 0
对 256 色的每一各颜色值
    前景色 w0=从 0 到此数值的像素的累积比例
    背景色 u0=此数值为止值的像素灰度均值
    类间方差 g=(w0*(u-u0)*(u-u0))/(1.0-w0)
    如果类间方差 g 大于最大类间方差 gmax
        更新最大类间方差值
        记录当前像素值为更合适的阈值
    结束
结束

返回阈值
```

二值化的过程如下：

```
灰度图像数据的二维目标矩阵置零
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        如果原矩阵[][]大于阈值，则目标矩阵[][]=255
        否则，目标矩阵[][]=0;
    结束
结束
```

对上图进行最大类间方差法二值化后的结果如下图所示。



5. Noise and Bounding Curves removing

此过程的目标是去除边缘噪音和背景线条。边缘噪音是指线条较短的边缘，背景线条是指较长的边缘。

车辆照片中，车身本身的轮廓所产生的边缘线条一般比较长，而背景的噪音所产生的边缘一般比较短，这两种边缘都会妨碍合理车牌边缘的进一步处理，所以需要将其清除掉。

线条长度的计算基于像素点的连通性，像素点之间的连通性定义如下：

对于从左上开始的扫描，如下表，红色区表示当前像素点的第一传播点区，绿色区表示当前像素点的第二传播点区。

	(i-2,j-1)	(i-2,j)	(i-2, j+1)	
(i-1,j-2)	(i-1,j-1)	(i-1,j)	(i-1, j+1)	
(i ,j-2)	(i ,j-1)			
(i+1,j-2)				

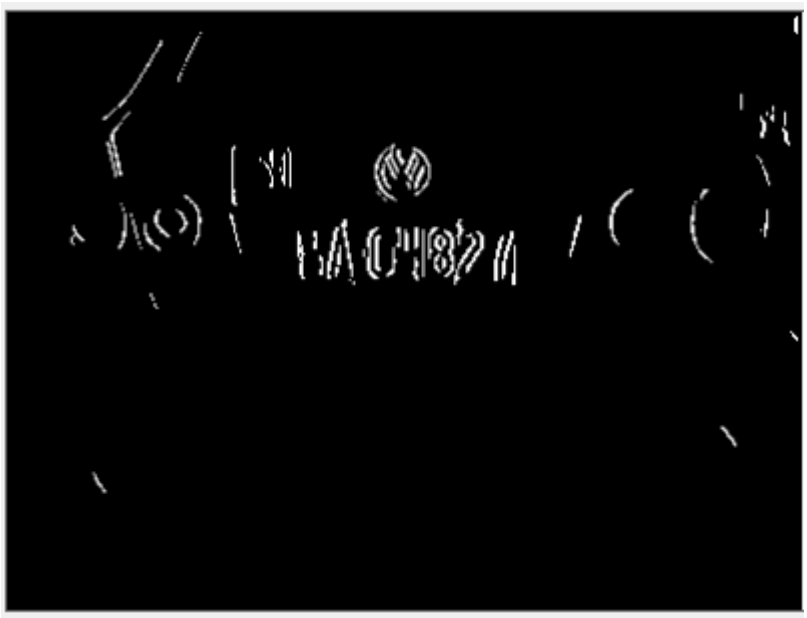
对于从右下开始的扫描，如下表所示，红色区表示当前像素点的第一传播点区，绿色区表示当前像素点的第二传播点区。

				(i-1, j+2)
			(i ,j+1)	(i ,j+2)
	(i+1,j-1)	(i+1,j)	(i+1,j+1)	(i+1, j+2)
	(i+2,j-1)	(i+2,j)	(i+2,j+1)	

清除噪音和背景线条基于合理长度的过滤，首先计算连通线条的高度，然后判断其是否合理，最后将不合理的线条清除掉。处理的过程如下：

```
矩阵 M、N 置零

//第一次扫描 从左上开始
对图像数据矩阵的每一行
    对每一行的每一位像素点
        如果此像素点为前景色
            如果此点的第一传播点区内有点是前景色
                M[][]=第一传播区内最大的 M 值+1
            否则
                M[][]=第二传播区内最大的 M 值+1
    结束
结束

//第 2 次扫描 从右下开始
对图像数据矩阵的每一行
    对每一行的每一位像素点
        如果此像素点为前景色
            如果此点的第一传播点区内有点是前景色
                N[][]=第一传播区内最大的 N 值+1
            否则
                N[][]=第二传播区内最大的 N 值+1
    结束
结束

//第 3 次扫描 进行合理性判断
对图像数据矩阵的每一行
    对每一行的每一位像素点
        如果此像素点为前景色
            如果此点的 M 值+N 值处于[row/20,row/6]范围内
                此点在有效的边缘线条中，E[][]=255（前景色）
            否则
                E[][]=0（背景色）
    结束
结束
```

在二值化图像中去除边缘噪音和背景线条的结果如下，与上图相比，过长的竖直方向边缘和零散过短的噪音线条已被清除，剩下的竖直边缘接近于车牌的高度。可以看出车牌区域的竖直边缘很明显地集中在一起。



6. Close

此过程是边缘连通环节，对二值化图像中的线条进行处理，达到将临近的竖直边缘连接起来的目的。由于车牌上字符位置相邻，其边缘会比较接近。适当的进行水平方向的膨胀就可以使车牌字符的边缘相互连通成车牌区域。

本过程分为两步，先对边缘的二值化图像进行闭操作，然后再一次进行膨胀，以保证完整的形成连通的车牌区域。

```
目标矩阵 N 置零
//反色
对矩阵 M 的每一行
    对每一行的每一位像素点
        如果此像素点 M[][]==0
            M[][]=255
        否则
            M[][]=0
    结束
结束

对矩阵 M 的每一行
    对每一行的每一位像素点
        此像素点 N[][]== M[][]的 3*3 窗口内最大像素值
    结束
结束

对矩阵 N 的每一行
    对每一行的每一位像素点
```

```
            此像素点 M[][]== N[][]的 3*3 窗口内最小像素值
        结束
结束

目标矩阵 N 置零
对矩阵 M 的每一行
        对每一行的每一位像素点
            如果此像素点 M[][]==0
                将此像素值扩散到矩阵 N[][]的 6*3 窗口内
        结束
结束
```

对二值化边缘图像进行边缘连通的结果如下所示。左图是进行 3*3 窗口的闭操作结果，右图是对闭操作结果进行进一步的水平方向上的 6*3 窗口的膨胀操作的结果图。

7. Contours and Min Bounding Box

此过程是检测连通区、计算连通区的轮廓和最小包围盒。这三个过程是依次进行的，每检测到一个连通区，随之就计算此连通区的轮廓和最小包围盒。

连通区域的检测是基于染色原理。对一个像素点的周边进行染色的顺序如下：

6	7	8
5		1
4	3	2

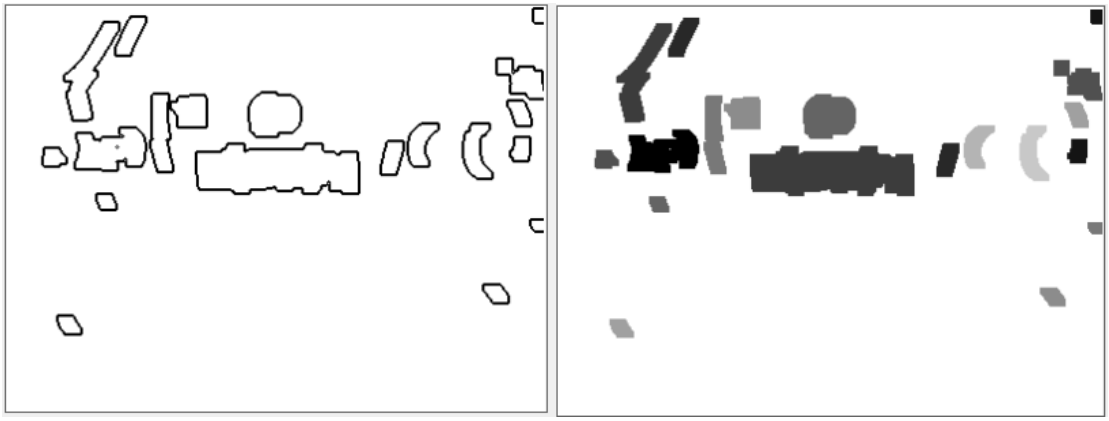
轮廓检测用于获取二值图像中连通区域的边界。其基于的原理是当前有效像素（颜色值==0）的上下两个像素或者左右两个像素的颜色值不同，即一个前景一个背景。边界点也是连通区域本身的像素点，而非向外扩展了一层。

```
目标矩阵 N 置零
对矩阵 M 的每一行
    对每一行的每一位像素点
        如果 M[][]的上下两点像素值不一致或者左右两点的像素值不一致
            N[][]=0
    结束
结束
```

最小包围盒计算是为了进行连通区的合理性判断，从而下一步处理中可以给出车牌的位置。最小包围盒选取的是连通区的最小 X 坐标、最大 X 坐标、最小 Y 坐标、最大 Y 坐标所组成的长方形区域。

```
目标矩阵 M 置零
对矩阵 E 的每一行
    对每一行的每一位像素点
        //连通区检测
        如果图像数据矩阵 E[][]像素值为前景色且未被着色标记
            此像素点加入队列末尾
            如果队列中有数据
                取出队首数据
                队首数据做着色标记
                更新此连通区的最大 X 坐标、最小 X 坐标、最大 Y 坐标、最小 Y 坐
标
                连通区的面积+1
                将此数据位置的周边的属于前景色但是未标记的像素点依次加入队
列
            结束

            lenX=maxX-minX,lenY=maxY-minY
            lenArea=lenX*lenY
            如果  lenX>60 && lenX<150 && lenY>10 &&lenY<50 && locArea>800 &&
locArea<5000 && (locArea*1.0)/lenArea>0.5
                此连通区为合理的车牌范围
                此连通区的信息加入到最小包围盒
                包围盒数量+1
            结束
    结束
结束
```

此过程处理的结果如下。左图是对上图中连通区进行检测到的轮廓边界图，右图是为了方便地区分连通区而进行的区域着色处理。



8. Plate

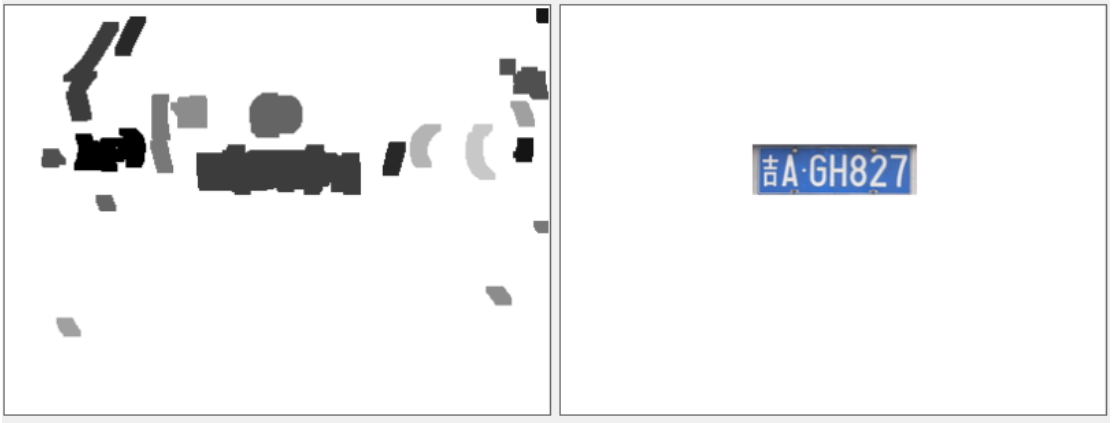
此过程是获取车牌图像。首先对每个连通区计算得到的最小包围盒进行合理性判断；如果合理，则此连通区代表着车牌上的字符位置，此最小包围盒代表着车牌位置。判断的标准如下：

包围盒的宽度在[60,150]个像素范围内

包围盒的高度在[10,50]个像素范围内

包围盒内连通区的面积占包围盒的面积比例在[0.5,1.0] 范围内

如果符合此标准，则此连通区为合理的车牌范围。处理的结果如下图所示，右图是获取的车牌图像。



六、 车牌识别

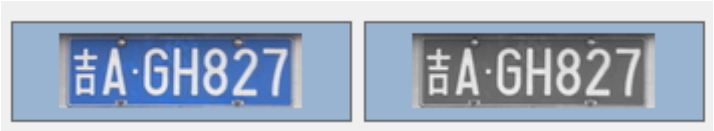
1. Gray

将 RGB 图像数据的三位矩阵转化为灰度图像数据的二维矩阵。本系统中采用

的灰度化公式为：Y=0.299*rgbRed+0.587* rgbGreen+0.114*rgbBlue。处理过程如下：

```
灰度图像数据的二维目标矩阵置零
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        目标矩阵[][]=0.299*RGB 矩阵[][][0]+0.587*RGB 矩阵[][][1]+0.114*RGB 矩阵[][][2]
    结束
结束
```

RGB 图像的灰度化效果如下图所示，左图为 24 位原图像，右图是 8 位灰度图像。



2. Otsu

与车牌定位中一样，在二值化处理时，系统采用的是 Otsu 大津法，即最大类间方差法。最大类间方差法按照图像的灰度特性，将图像中像素分为前景和背景，而前景与背景之间的类间方差越大则分割效果越好。Otsu 二值化分为计算阈值和图像二值化两部分。

阈值的计算过程如下：

```
灰度图像数据的二维目标矩阵置零
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        对灰度值(原矩阵[][])进行累计
    结束
结束

//计算从 0 到当前数值的均值和概率分布
对 256 色的每一各颜色值
    此数值的像素数占总体的比例=此数值的像素数/图像总像素数
    图像均值+=此数值像素数占总体的比例*此数值

    此数值的概率分布+=此数值像素数占总体的比例
    此数值为止值的像素灰度均值=图像均值/从 0 到此数值的像素的累积比例
结束

初始化阈值为 0
对 256 色的每一各颜色值
```

```
    前景色 w0=从 0 到此数值的像素的累积比例
    背景色 u0=此数值为止值的像素灰度均值
    类间方差 g=(w0*(u-u0)*(u-u0))/(1.0-w0)
    如果类间方差 g 大于最大类间方差 gmax
        更新最大类间方差值
        记录当前像素值为更合适的阈值
    结束
结束

返回阈值
```

二值化的过程如下：

```
灰度图像数据的二维目标矩阵置零
对 RGB 矩阵的每一行
    对每一行的每一位像素点
        如果原矩阵[][]大于阈值，则目标矩阵[][]=255
        否则，目标矩阵[][]=0;
    结束
结束
```

对上图进行最大类间方差法二值化后的结果如下图所示

3. Binarization

此过程称为二值化后处理，目标是实现二值化颜色归正，以黑色为背景色，白色为前景色，白色代表文字。

车牌具有多种底色，一般蓝底的车牌在二值化处理后文字会显示为白色，但黄底的车牌正好相反，处理后的文字显示为黑色。为了统一进行数字的分割和识别，需要将二值化图像的格式进行调整：统一将文字显示为白色，车牌底色显示为黑色。

二值化后处理基于的原理是统计，在车牌中，字符所覆盖的面积占车牌总面积的比例小于未覆盖区域的面积占总面积的比例。处理的过程如下：

```
//计算指定中心区域内 0 值像素点的面积
中心区域的行 限定在[row/4,(row*3)/4]
中心区域的列 限定在[column/4,(column*3)/4]
面积=0
```

```
对限定的中心区域内的每一行
    对限定的中心区域内的每一行的每一位像素点
        如果[][]为前景色 则面积+1
    结束
结束

//如果有必要 则反色 保证文字为前景色（白色 255）
如果中心区域内的 0 值像素点面积占中心区域的面积的比例不到 0.4
    对矩阵的每一行
        对每一行的每一位像素点
            如果像素值为 0，则设为为 255 //前景色
            否则 设为 0
        结束
    结束
结束
```

二值化后处理的结果如下所示，下面两张图从左到右的四个子图依次是：车牌原图像、灰度化处理图像、最大类间方差法二值化后的图像、二值化后处理得到的二值化图像。

图 蓝底+白字式车牌

图 白底+红字式车牌

4. Segmentation

字符分割过程的目标是在二值化图像中获取每个字符的最小包围盒，并从图像中分割出单个字符区域。步骤如下：

- 获取连通区域，方法与车牌定位中的连通区着色算法一致
- 判断连通区是否是字符。使用最简单的过滤法，根据字符合理的宽度、字符的宽高比和字符覆盖区域占其最小包围盒的面积比，对检测到的连通区进行筛选

- 对合理的字符区进一步处理。计算其包围盒，并可视化其包围盒的边框。

下图是分割结果，将二值化图像中的前景字符的最小包围盒绘制出进行了可视化。

5. Recognition

字符识别的目标是对分割出的字符图像进行识别，判断其属于哪一个数字的图像。步骤如下：

- 加载数字的图形库，计算其特征 X 向量和 Y 向量
- 对待识别图像的图片，计算其的 X 向量和 Y 向量
- 计算待识别图像的特征向量和每个数字的特征向量的欧几里得距离
- 取距离最小的数字特征为该待识别图像的标记
- 显示识别结果

识别的结果如下图所示，数字串显示在分割图像下面的文本框中。

七、 参考文献

车牌定位的流程是根据 Vehicle License Plate Recognition Based on Extremal Regions and Restricted Boltzmann Machines （Chao Gou, Kunfeng Wang , Yanjie Yao, and Zhengxi Li ）一文进行设计和实现的。

在车牌定位部分中，边缘噪音和背景线条的去除算法（Noise and bounding curves removing）参照了 An efficient method of license plate location（Danian Zheng , Yannan Zhao, Jiaxin Wang）一文的 2.3 Background curve and noise removing 一节所设计的算法。