**车牌识别系统**

**摘要**

本文采用**两种不同方式**实现车牌识别系统，第一种方法是传统的模板匹配，第二种采用深度学习。

对于第一种传统车牌识别方法，本文使用 Keil for ARM 作为该系统软件部分设计的集成开发环境，使用ARM Cortex-M3内核的STM32F103ZET作为实验平台。首先，对识别的对象车牌进行特征分析，包括几何特征、字符特征、灰度变化特征、投影特征等，掌握车牌特征，可以利用车牌图像生成器，按照要求生成任意的车牌，作为识别阶段的数据来源之一。**然后**，分模块设计系统，硬件模块主要组成部分包括:电源模块、STM32F103ZET最小系统、ST-LINK调试器（使用SWD接口）、摄像头OV7725、4.3英寸TFT；软件模块中采用二值化进行图像预处理，颜色特征的方法定位车牌，投影法分割字符，归一化分割得到的字符，然后采用模板匹配识别车牌。**最终**，先对车牌图像生成器生成的不带环境背景的车牌进行识别，然后再采集带有环境背景的车牌进行再次识别，同样发现，车牌识别的效果较好，但是那些特征相似的字符的识别结果较差，如E和F，同样有些汉字的识别率较低，如“沪”。

传统的车牌识别方法存在识别率低，单张用时长，鲁棒性差等特点。为解决此问题，本文接着采用深度学习的方法进行端到端的车牌识别，本文选用Inter公司的LprNet网络进行复现，LPRNet的主要优点有三个，第一不需要字符预先分割、第二部署于嵌入式设备、第三鲁棒性好。

**首先**采用LocNet网络对车牌进行空间变换预处理，校正车牌倾斜，提高后续车牌识别的准确率。**其次**通过主干网络，对输入的RGB图像进行特征提取，下采样等处理，最终输出一个代表对应字符可能性的序列；**然后**由于解码器的输出与目标字符序列的长度是不一致的，因此采用了CTC损失函数。模型训练采用CCPD数据集，选取其中11696车牌图像,然后以9:1的比例将数据集划分为训练子集和验证子集。在NVIDIA GeForce GTX 1080显卡运算每个车牌3ms，在英特尔酷睿i7-6700K上每个车牌1.3ms。车牌识别结果和方法对比如表4所示，LprNet方法实时识别准确率达到95%。相比于传统的车牌识别方法，准确率大大提高，单张车牌识别所用时间大大减少；与YOLOV3等方法相比，LprNet算法的效率更高。**最终**采用图形程式框架PyQt5进行识别结果可视化，通过Qt Designer来完成，实现了视图和逻辑的分离。

**关键词**：模式识别；模板匹配；车牌识别；STM32F103ZET；LprNet网络；PYQT5

**目录**

[一、引言 2](#_Toc21168)

[二、基于STM32F103ZET的车牌识别 3](#_Toc22866)

[2.1研究对象分析 3](#_Toc14542)

[2.2系统模块设计 4](#_Toc17127)

[2.2.1硬件模块 4](#_Toc25315)

[2.2.2软件模块 5](#_Toc24647)

[2.3车牌识别算法实现原理 5](#_Toc18310)

[2.3.1.车牌图像预处理——图像二值化 5](#_Toc19767)

[2.3.2.车牌定位 5](#_Toc15557)

[2.3.3.车牌字符分割 6](#_Toc4971)

[2.3.4车牌识别 6](#_Toc27130)

[2.4实验结果 8](#_Toc14204)

[三、基于LPRNET的车牌识别系统 10](#_Toc31064)

[3.1环境配置 10](#_Toc6858)

[3.2 LprNet网络结构 10](#_Toc3797)

[3.2.1空间变换预处理LocNet 10](#_Toc16702)

[3.2.2骨干网络 11](#_Toc30550)

[3.3 CCPD数据集 11](#_Toc16964)

[3.4识别结果 12](#_Toc23113)

[3.4.1识别结果分析 12](#_Toc9850)

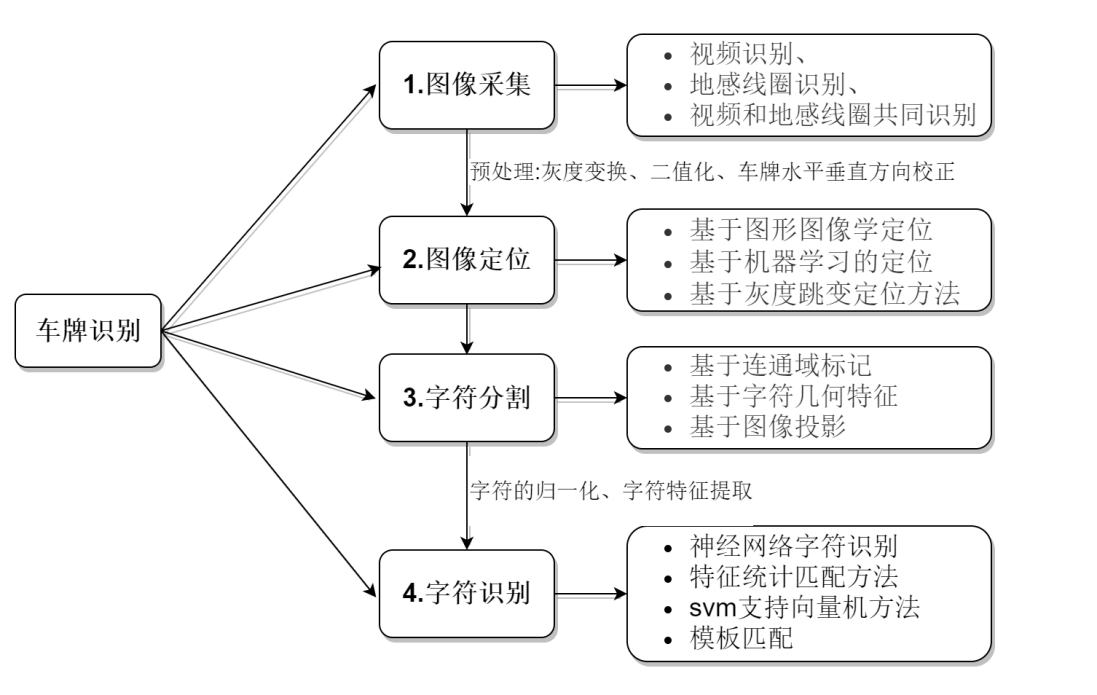
[3.4.2识别结果可视化——Python图像界面编程 12](#_Toc18712)

[四、结束语 13](#_Toc16353)

**一、引言**

智慧城市利用先进的信息技术实现智慧化地管理城市，为人们创造和谐美好而便捷的生活。而车牌识别系统是智能城市的组成部分之一，采用数字图像处理、模式识别和计算机视觉技术，在不影响车辆行驶的条件下，准确而迅速地获得车辆的数字化信息，最终实现车辆的智能化管理。

车牌识别系统其流程图如图 1 所示



**图1.车牌识别系统实现方法**

**二、基于STM32F103ZET的车牌识别**

基于STM32F103ZET的车牌识别系统是一个嵌入式系统，在结构上，嵌入式系统是软件和硬件的结合体。嵌入式系统的程序一旦烧写到终端，一般不会进行改变。具有功耗低、目的性强执行效率高等很多的优点。

本文采用基于ARM Cortex-M3内核的STM32F103ZET作为本文研究设计平台。

**2.1研究对象分析**



要进行车辆牌照的提取和分割，必须了解车辆牌照的特征。目前的车牌定位算法都或多或少利用了车牌自身的特征。车牌自身具有许多固有的特征，我国车牌的特征主要包括以下几个方面：

**图2.车牌对象特征**

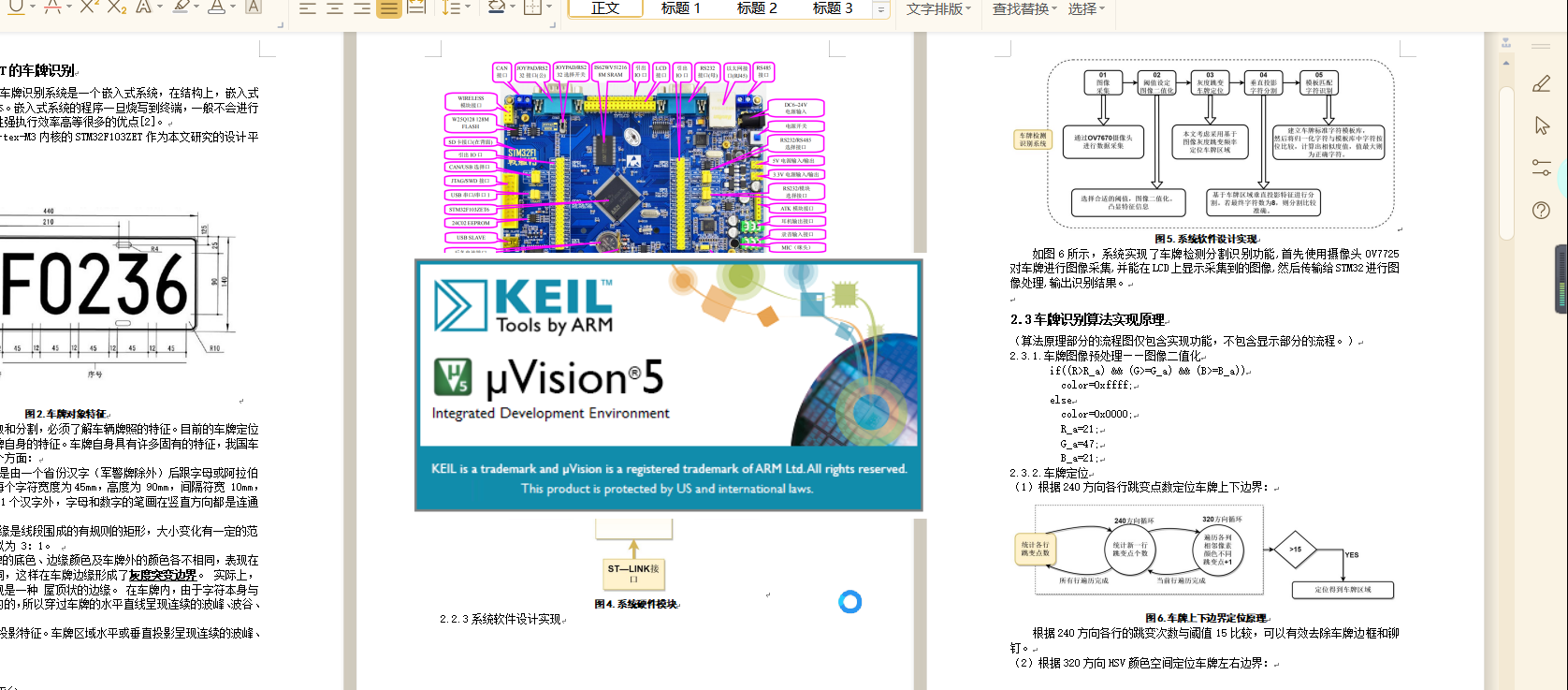
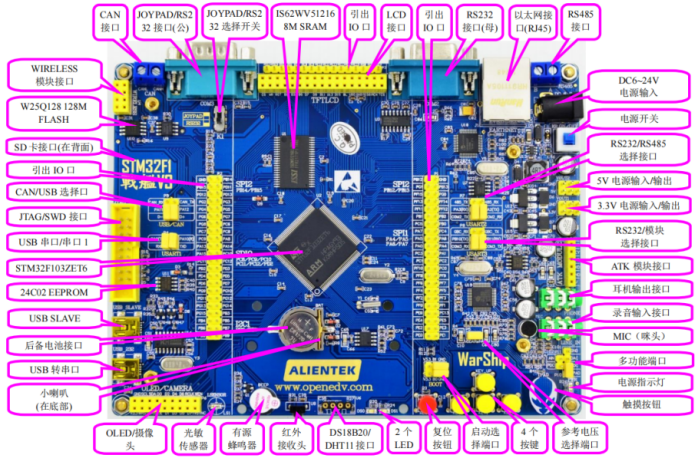
（1）字符特征。

车辆牌照是由一个省份汉字（军警牌除外）后跟字母或阿拉伯数字组成的7个字序列。每个字符宽度为45mm，高度为 90mm，间隔符宽 10mm，每个单元间隔12mm。除第1个汉字外，字母和数字的笔画在竖直方向都是连通的。（2）几何特征。车牌的边缘是线段围成的有规则的矩形，大小变化有一定的范围，整个车牌的宽高比近似为 3：1。 （3）灰度变化特征。 车牌的底色、边缘颜色及车牌外的颜色各不相同，表现在图像中就是灰度级各不相同，这样在车牌边缘形成了灰度突变边界。 实际上，车牌的边缘在灰度上的表现是一种 屋顶状的边缘。 在车牌内，由于字符本身与牌照底色的内部灰度是均匀的，所以穿过车牌的水平直线呈现连续的波峰、波谷、波峰的分布。（4）车牌区域水平或垂直投影特征。车牌区域水平或垂直投影呈现连续的波峰、波谷、波峰的分布。

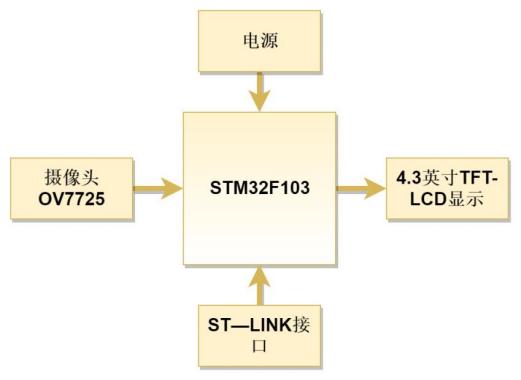
**2.2系统模块设计**

2.2.1硬件模块

（1）开发环境Keil5与实验平台STM32F103ZET如图3.1和图3.2所示

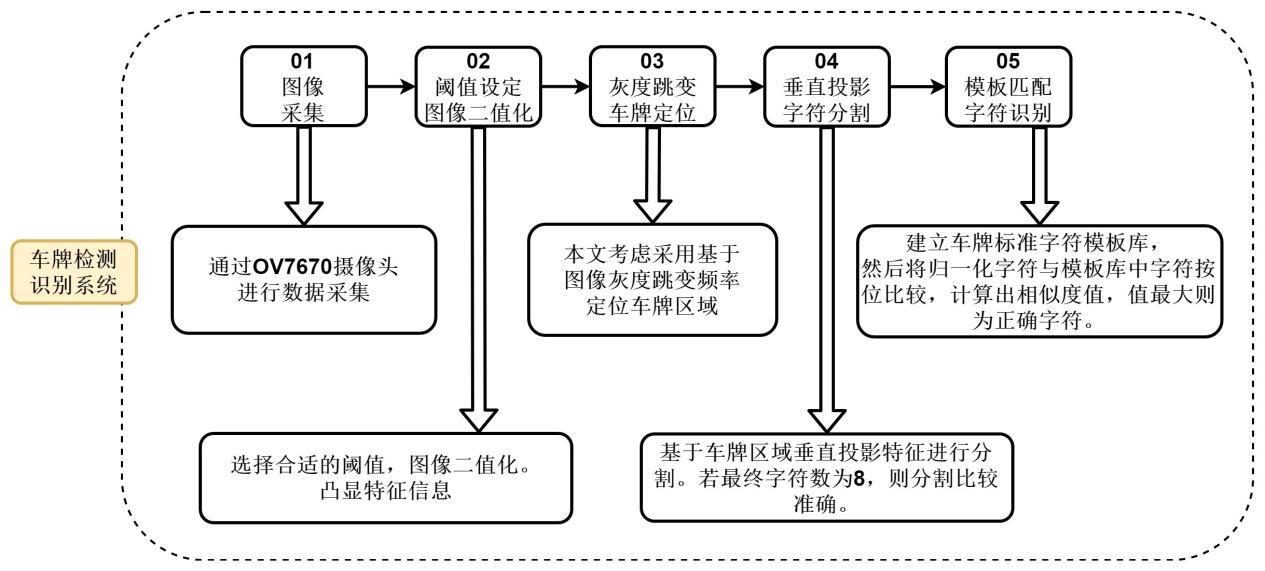
**图3.1.开发环境Keil5 图3.2.实验平台STM32F103ZET**



按照本系统功能及指标的要求,本系统的主要组成部分包括:电源模块、STM32F103ZET最小系统、ST-LINK调试器（使用SWD接口）、摄像头OV7725、4.3英寸TFT。该系统的系统框图如图4所示。

**图4.系统硬件模块**

2.2.2软件模块



**图5.系统软件设计实现**

如图6所示，系统实现了车牌检测分割识别功能,首先使用摄像头OV7725对车牌进行图像采集,并能在LCD上显示采集到的图像,然后传输给STM32进行图像处理,输出识别结果。

**2.3车牌识别算法实现原理**

2.3.1.车牌图像预处理——图像二值化

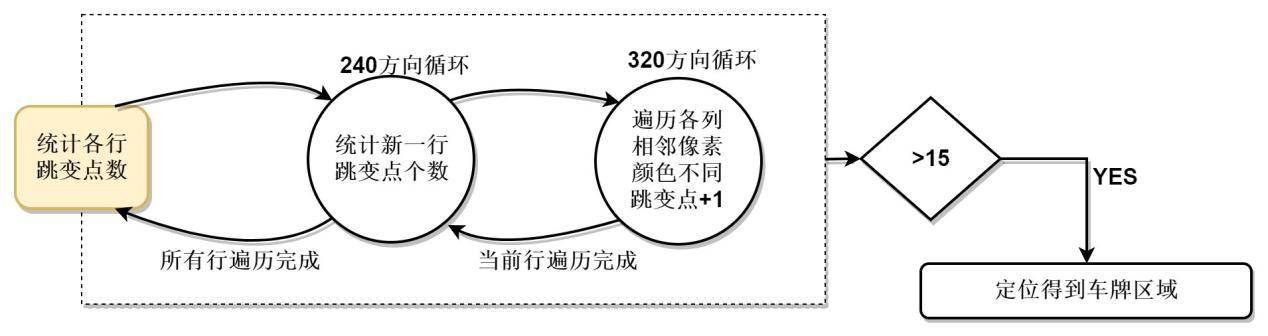
if((R>R\_a) && (G>=G\_a) && (B>=B\_a)) color=0xffff;

else

{color=0x0000;R\_a=21;G\_a=47;B\_a=21; }

2.3.2.车牌定位

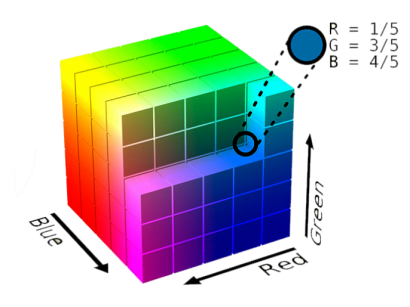
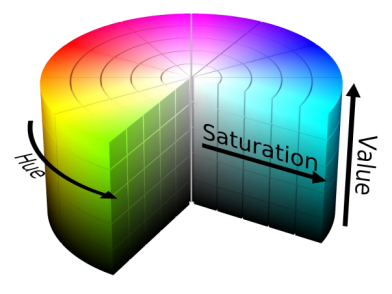
（1）根据240方向各行跳变点数定位车牌上下边界：



**图6.车牌上下边界定位原理**

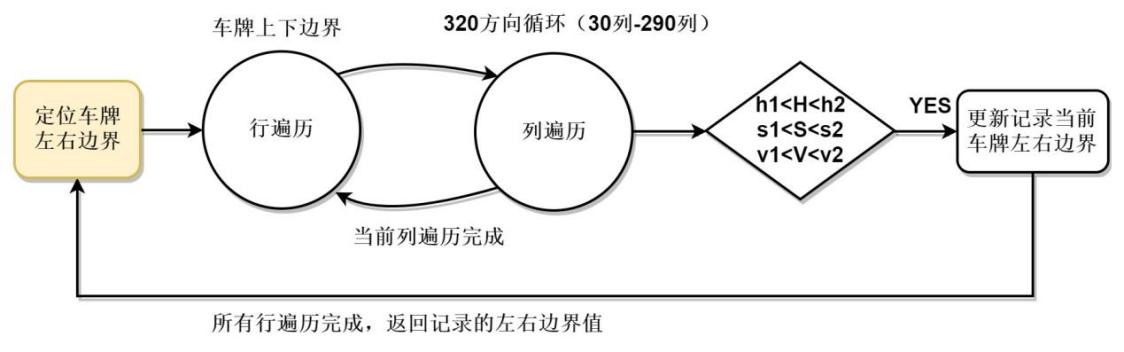
根据240方向各行的跳变次数与阈值15比较，可以有效去除车牌边框和铆钉。

（2）根据320方向HSV颜色空间定位车牌左右边界：

**图7.RGB颜色空间 图8.HSV颜色空间**

采用HSV颜色模式识别蓝色区域更符合人眼的颜色识别。HSV即色相、饱和度、明度。HSV中显示出来大多是比较鲜艳的绿色或者荧光黄色。HSV是一种将RGB色彩模型中的点在圆柱坐标系中的表示法，这种表示法试图做到比RGB基于笛卡尔坐标系的几何结构更加直观。



**图9.车牌左右边界定位原理图**

2.3.3.车牌字符分割

**根据320方向各列跳变点数实现车牌字符分割，**320方向各列的跳变点的统计同理于240方向，此处不再赘述。

本文采用**垂直投影法**的字符划分方法来划分车牌字符区域，依据车牌特点可知，每个字符之间都存在一定的纯黑区域。二值化后，字符区域为白色，车牌中的非字符区域为黑色。垂直投影后得到直方图为:



**图10.车牌及其垂直投影图**

若最终分割的字符数为8，则分割正确。

2.3.4车牌识别

（1）字符图像归一化

需要对划分出的单个字符进行归一化处理，使得从采集图像中定位出的车牌在字符分割操作完成后，所获取的单个字符图像大小都为25\*50像素，基于我国车牌的标准尺寸，每个字符宽度为45mm，高度为 90mm，即宽高比例为1:2。



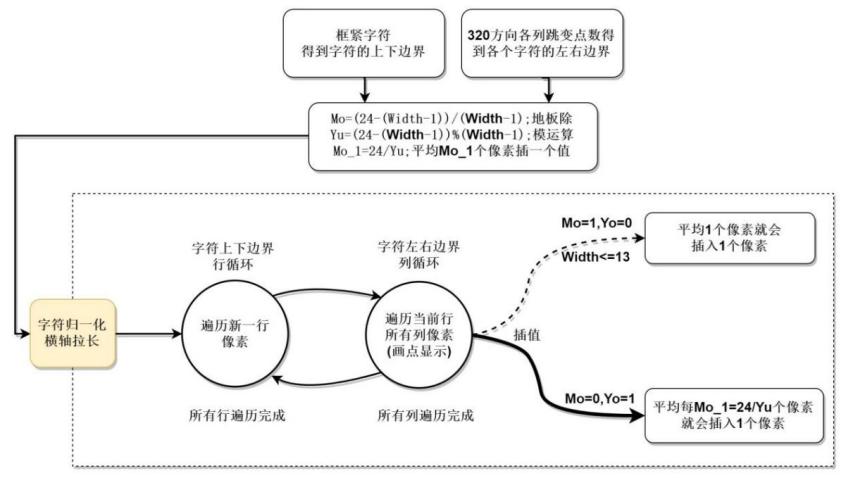
**图11.字符归一化长宽比为1:2**

假设： 

容易证明：当Mo=1时，Yu=0；即当图片的宽度只有指定宽度的一半时（宽度只能取一半或者以下），平均1个像素就会插入1个像素，插入像素的颜色值取前一个像素的颜色值；相对应的，当Mo=0时，Yu=1，平均每Mo\_1=24/Yu个像素就会插入一个像素。

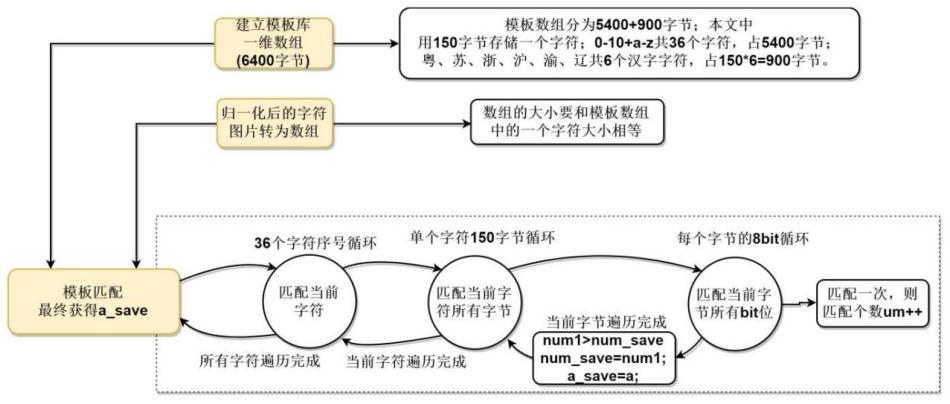
因此对于两种情况，需要分别编写代码实现进行横向（纵向）拉伸到25（50）像素，即进行横向和纵向的归一化。具体的归一化流程如下，以横向拉伸为例，纵轴方向的拉伸同理。

具体归一化实现原理如图12所示:



**图12.字符归一化流程**

（2）基于模板匹配的字符识别

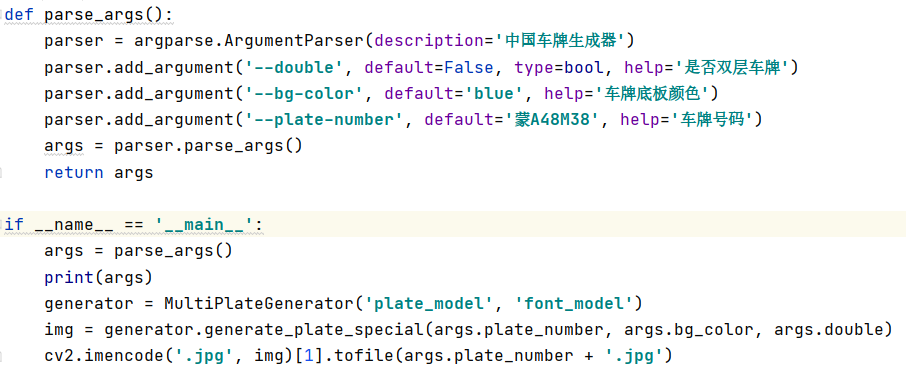
字符归一化完成后，即可进行模式识别，模板匹配是最早出现的模式识别方法，也是模式识别中最容易的一种，数学模型很容易建立。它对每个类别建立一个或多个标准模板，

**图13.模板匹配原理图**

分类决策时将待识别的样本与每个类别的模板进行比对，根据与模板的匹配程度将样本划分到最相似的类别中。严格来说，模板匹配不能算是模式识别的范畴，在建立模板的时候需要人工的干预，但由于它直接、简单，在类别特征稳定、明显，类间差距大的时候仍然可以使用，只是它的适应能力比较差。在计算机视觉中，图像的轮廓、灰度、纹理和边缘等都可以作为模板。在实际的应用中，人脸识别、条码识别、字符识别、零件检测、车牌识别、苹果识别和指纹识别等都可以使用模板匹配方法。

**2.4实验结果**

2.4.1车牌图像生成器（部分代码如下）



**图14.车牌生成器生成标准车牌**

2.4.2识别结果

|  |  |  |  |
| --- | --- | --- | --- |
| 辽B98436 |  | 渝A48M38 |  |
| sue05ev8 |  | yueE33333 |  |

2.4.2结果分析

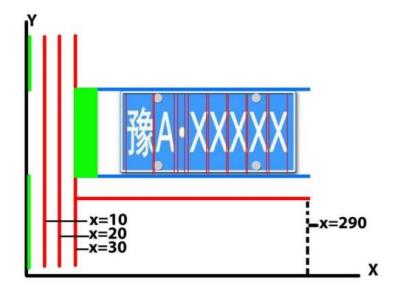


图15是实验结果图的标准化表示方法,图中x=10,x=20,

x=30三条红线作为参考线；绿色部分代表当前行统计的灰度跳变点个数；与参考线垂直的红线规定在识别过程中车牌区域不能超过该线，而要限定于上下两条蓝线之间，这两条蓝线可以按照车牌特征

**图15.车牌识别显示理想效果图**

进行自动锁定。最终实验结果表明，当车牌区域位于两条蓝线之前，且下边界不超过红线时，识别结果较为准确，但对于字符F、E的识别准确率较低，原因之一是他们的字符特征交相似。

2.4.3.难点分析

**表1.基于STM32F103ZET车牌识别系统的难点及对应解决方案**

|  |  |
| --- | --- |
| **难点分析** | **解决方案** |
| 显示：LCD设定显示区域、显示数字、汉子、字母，画点划线、及LCD的8种扫描方向的确定 | 使用ST-LINK调试器（使用SWD接口）反复调试，对比实际显示效果和设定的显示效果，不断的进行修改。 |
| 显示：对ov7725摄像头的功能方法不明确 | 查阅《STM32F1开发指南-库函数版本》，ov7725摄像头，了解到只能输出格式为RGB565的图片。因此需要读写两个8bit像素进行移位操作拼接为1个16bit |
| 识别：车牌识别效果不理想，不知道错误的原因 | 识别过程显示过快，无法明确每个字符的识别过快。因此在调试的过程中使用延时函数，便有足够的时间仔细分析整个识别过程。发现LCD镜像显示车牌字符，可能导致识别效果不理想。 |
| **经验教训** | |
| 1)调试：通过flymcu开始编译执行hex文件时，忘记选择编译前重装文件选项，导致多次调试的结果没有任何变化；  2)进一步熟悉按位操作：  如：<<，有符号左移位，将运算数的二进制整体左移指定位数，低位用0补齐。>>，有符号右移位，将运算数的二进制整体右移指定位数，整数高位用0补齐，负数高位用1补齐（保持负数符号不变）；  又如:在模板匹配时，进行按位比较:**for(e=0;e<8;e++){s1=st1&(1<<e);  s2=st2&(1<<e);}**  3)注重基础运算模运算和地板除基础的运算完成图像归一化 | |

2.4.4工作不足

**表2.工作不足**

|  |  |
| --- | --- |
| **1** | 4.3英寸 TFT 分辨率为800\*480, 而现在采用320\*240的分辨率 |
| **2** | 按键蜂鸣器等外设没有添加任何功能 |
| **3** | 对于特征相似的字符识别率较低，如F和E |
| **4** | 基于模板匹配的车牌识别相比于端到端的识别算法，流程十分繁琐，过程也及其复杂 |

**三、基于LPRNET的车牌识别系统**

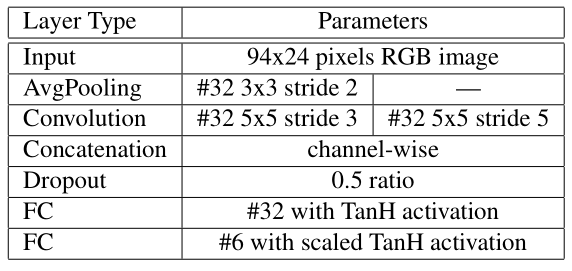
**3.1环境配置**

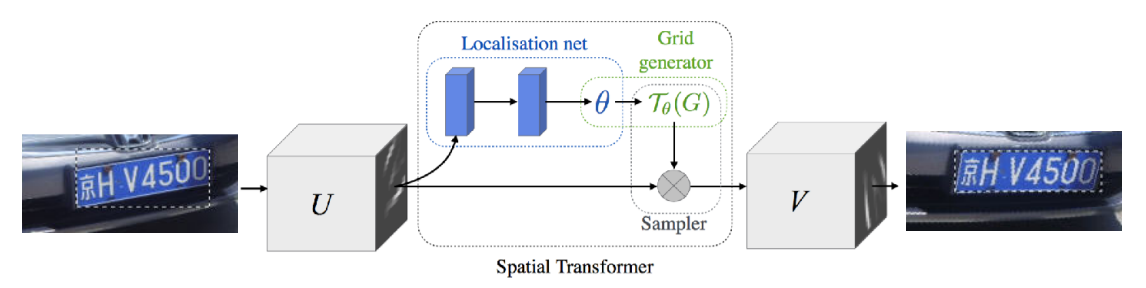
pytorch=1.4.0、python=3.6.13、pyqt5=5.15.4

**3.2 LprNet网络结构**

3.2.1空间变换预处理LocNet

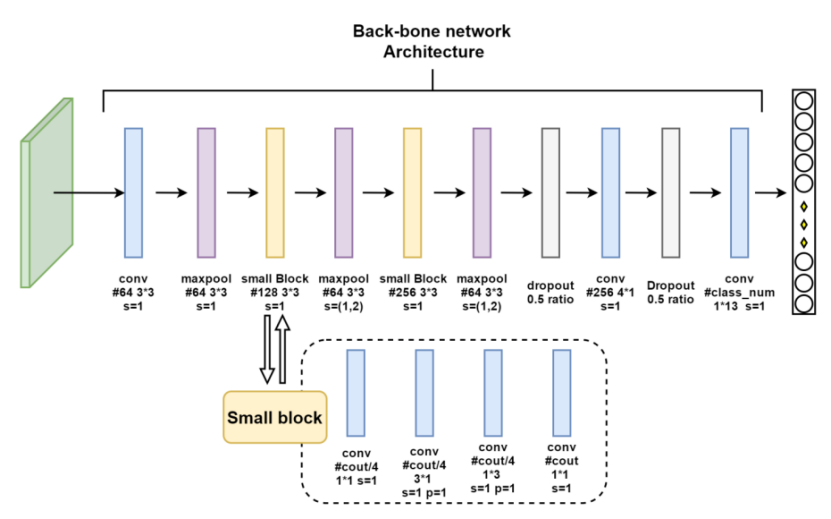
**表3.LocNet网络结构**

使用Spatial Transformer Layer对检测到的倾斜车牌进行校正，校正前后的效果如表3所示。因此利用该模块，将会提高图像识别准确率。



**图16.LocNet网络校正效果**

3.2.2骨干网络

骨干网络的结构在图17中进行了描述。如图17中所示，原始的RGB图片作为骨干网络输入，计算出大量特征的空间分布。宽卷积(1\*13的卷积核)利用本地字符的上下文从而取代了基于LSTM的RNN网络。骨干子网络的输出是一个代表对应字符可能性的序列，它的长度刚到等于输入图像的宽度。

**图17.主干网络LPR-Net**

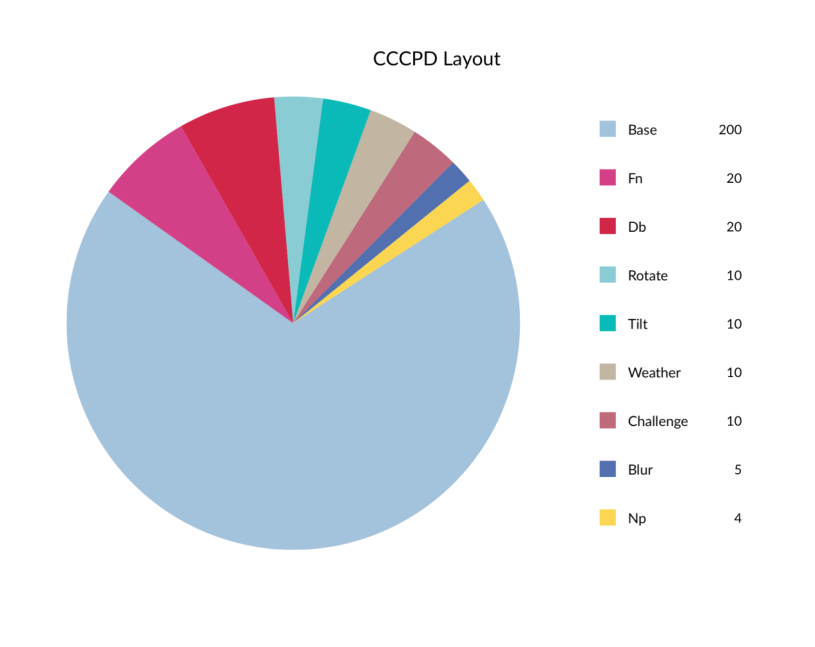
由于解码器的输出与目标字符序列的长度是不一致的，因此采用了CTC损失函数,无需分割的端到端训练。

CTC 损失函数是一种广泛地用于处理输入和输出序列不对齐的方法。pytorch内置torch.nn.CTCLoss的方法。即在Pytorch 1.0.x版本内早就有内置ctc loss接口了，我们完全可以直接使用，使用大致步骤如下：

首先，获取CTCLoss()对象，其次，在迭代中调用CTCLoss()对象计算损失值，对应代码如下:ctc\_loss = nn.CTCLoss(blank=len(CHARS)-1, reduction='mean')；loss = ctc\_loss

(log\_probs, targets, input\_lengths, target\_lengths)。

**3.3 CCPD数据集**

**图18.**

车牌数据集CCPD如图18所示，该数据集由中国科学技术大学开源，数据集庞大。数据集采集城市道路场景，数据集极不平衡。

该数据集在合肥市的停车场采集得来的,采集时间早上7:30到晚上10.00.涉及多种复杂环境共包含超多25万张图片,每种图片大小720x1160x3。一共包含9项。每项占比如下图

**车牌数据集CCPD**

选取11696车牌图像,然后以9:1的比例将数据集划分为训练子集和验证子集。

**3.4识别结果**

3.4.1识别结果分析

**表4.不同方法实验结果对比**

|  |  |  |
| --- | --- | --- |
| **方法** | **识别率/%** | **单张用时/ms** |
| 模板匹配 | 80.35 | 49.15 |
| HOG+SVM | 90.12 | 52.64 |
| BP神经网络 | 83.24 | 50.12 |
| 多级细粒度特征融合车牌识别 | 94.56 | 53.83 |
| YOLOV3 | 99.32 | 17 |
| 边缘检测定位与连通域分割 | 83.2 | —— |
| License Plate Recognition via  Deep Neural Networks | 95.0 | 1.3 |

LprNet方法只需要0.34gflops就可以完成一次前向传播，在NVIDIA GeForce GTX 1080显卡运算每个车牌3ms，在英特尔酷睿i7-6700K上每个车牌1.3ms。

如表4所示，LprNet方法实时识别准确率达到95%。相比于传统的车牌识别方法，准确率大大提高，单张车牌识别所用时间大大减少；与YOLOV3等反方相比，LprNet算法的效率更高。

**表2.LprNet方法优点分析**

|  |  |
| --- | --- |
| **优点** | **分析** |
| LPRNet不需要字符预先分割 | 车牌识别的准确率高、算法实时性强、支持可变长字符车牌识别。对于字符差异比较大的各国不同车牌均能够端到端进行训练 |
| 部署于嵌入式设备 | LPRNet是第一个没使用RNN的实时轻量级算法，可以实现端到端的识别，能够在包括嵌入式设备在内的各式设备上运行。例如LPRNet可以部分移植到FPGA上，这可以为其他部分释放CPU资源 |
| 鲁棒性好 | LPRNet在实际交通监控视频中的应用表明，该算法在视角和摄像畸变、光照条件恶劣、视角变化等复杂的情况下仍表现出很好的识别效果。 |

3.4.2识别结果可视化——Python图像界面编程

PyQt5是基于Digia公司强大的图形程式框架Qt5的python接口，由一组python模块构成。PyQt5本身拥有超过620个类和6000函数及方法。在可以运行于多个平台，包括：Unix, Windows, and Mac OS。

在PyQt中编写UI界面通过Qt Designer来完成，如图20所示。Qt Designer的设计符合MVC的架构，其实现了视图和逻辑的分离，从而实现了开发的便捷。Qt Designer生成的.ui文件（实质上是XML格式的文件）也可以通过pyuic5工具转换成.py文件。 将.ui文件转换到.py文件很简单，在前面我们曾设置了pyuic5这个工具。

**图20.Qt Designer操作界面**





**图21.车牌识别测试结果（环境背景包括夜晚灯光、加雾、加遮掩）**

**四、结束语**

从选题阶段的犹豫不决，到文献综述阅读研究的乏味难耐，到与硬件设备的不断磨合，再到软件代码的不断调试，到实验结果不尽人意，再到程序代码的返工修改，最终，实验结果终于一步步改进，但是任然存在诸多不足之处，如基于STM32F103ZET的车牌识别方法中预处理较为简单、PPT制作、报告创新点不足、LprNet未部署到FPGA、深度神经网络学习难度大等。但是我必须感谢老师给我们这次机会，并且在实验过程中督促每一个小组的进度、耐心解答每一个小组的疑问。一句话做结尾，那便是**道固难且艰,亦不阻我志矣**!