代　号	10701	学　号	0920421337
分类号	TP393	密　级	公开

题（中、英文）目　基于 Hadoop 平台的字符识别的研究

Research of Character Recognition Based on Hadoop

作者姓名	杨超	指导教师姓名、职务	王凯东 副教授
学科门类	工学	学科、专业	计算机系统结构

提交论文日期　二〇一二年二月

创新性声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：　　日期：

关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。本人保证毕业离校后，发表论文或使用论文（与学位论文相关）工作成果时署名单位仍然为西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。（保密的论文在解密后遵守此规定）

本学位论文属于保密，在___年解密后适用本授权书。

本人签名：　　日期：

导师签名：　　日期：

摘 要

字符识别是模式识别的一个重要分支，它是一门研究如何利用计算机系统自动识别各种媒介上印刷的或手写的字符的技术。随着社会信息化进程不断提高，我们在日常工作中经常需要将大量的字符信息转换成计算机可存储可处理的信息。字符自动识别技术的发展帮助我们自动完成这项工作。但是，随着数据量的不断增大，传统的单机字符识别系统的效率难以达到要求。

针对这个问题，本文对字符识别进行了研究，以提高字符识别的效率。首先本文以车牌识别为例研究了字符识别系统的一般识别流程和实现方法，包括车牌图像预处理、分割、特征提取、字符识别等操作，探讨了使用模板匹配识别和神经网络来进行字符识别的方法；然后本文阐述了字符识别与分布式系统结合的意义，研究了在 Hadoop 平台下基于 MapReduce 编程框架的分布式系统的实现，提出了分布式字符识别的方法并给出了设计方案；最后，本文将分布式字符识别系统与单机字符识别系统进行了实验，对其执行任务的效率进行了分析对比。

关键词：字符识别 Hadoop MapReduce 分布式

ABSTRACT

Character recognition is an important branch of pattern recognition, which is mainly to study how to use computers to recognize a printed or handwritten character automatically. With the continuous improvement of social information, we now need to store information form a lot of characters in our world into computers. The development of automatic character identify systems helps us to do this work automatically. However, with the increasing of data amount, the efficiency of traditional systems is difficult to meet the requirements of us.

For solving this problem, we study generally processes of character recognition by using the license plate recognition system as an example. We go deep into character recognition by using template matching and neural network. Then we discuss the significance of the combination of character recognition and distributed systems, research the implementation of a distributed system based on hadoop, and we give a design of distributed character recognition. In the end, we analyzed the efficiency of a distributed character recognition system and a normal character recognition system.

Keywords: Character recognition Hadoop MapReduce Distributed system

目 录

第一章 绪论

1.1 研究背景

文字的产生经历了一个漫长的过程。在文字产生以前，人们用各种实物手段，例如结绳，来记录信息、辅助记忆，这是不利于传播和传承的，有很大的局限性。而文字的产生无疑对社会的进步、人类的文明发展产生了不可磨灭的贡献，它极大地促进了信息的传播与知识的传承，对人类的发展起到了巨大的推进作用。

当今时代是一个高度信息化的时代。借助于计算机和网络的发展，人类资源共享与交流日益频繁与快捷，而且借助于计算机以及其他电子设备，人们可以更快捷简单的处理很多繁杂的工作。计算机是大规模大批量数据、信息等的最好的辅助处理设备。为了更好的达到资源共享的目的以及更快捷的处理各种信息，有时就需要将其他媒介上的信息录入计算机中。

一般将外界文本保存到计算机中最为快捷的方式就是使用摄像机等成像设备拍摄实物图片存入计算机。但是这种方式存储占用的存储空间巨大，冗余信息多，不利于处理。而使用人工将这些信息输入计算机虽然可以解决这一问题，但是却是非常繁琐的，不但任务繁重，而且各种人为失误无法避免，这在一定程度上减缓了社会信息化的进程。因而，如何快速高效准确地将这些信息录入计算机，是我们需要解决的重要问题。而使用各种自动化的录入设备将外界文本信息直接转化为计算机内部文字存储方式统一存储处理，是解决这个问题的重要方法。

使用计算机进行文字识别的总体来说可以分为两个步骤：第一步是通过扫描仪、数码相机等电子设备将纸质的信息转换成计算机可存储的图像信息；第二步是利用图像处理技术对图像信息进行处理并利用模式识别的方法来将图像信息转换成文字信息[1]。

光学字符识别(Optical Character Recognition，OCR)技术是目前普遍采用的一种重要技术。它是模式识别的一个分支，是通过扫描仪以及其他成像设备将纸质的或其他方式记录的印刷体或手写体的文本字符扫描成图像，输进电脑，然后通过一系列的处理，识别成相应的计算机可直接处理的字符，从而将字符信息化。它能大大提高信息的采集录入速度，减轻人们的工作强度，减少人为误差。伴随着多年来计算机技术的飞速发展以及各种理论算法的研究，OCR 技术已经得到了长足的发展，经过不断的改进和完善，现在已经广泛应用于各个领域，在许多重要领域，如银行票据、财务报表、车牌识别、邮政编码等，正发挥这重要作用，

使得大量的文档资料能快捷、省力和及时地自动输入计算机，实现信息处理的电子化。

车牌识别(License Plate Recognition，LPR)是字符识别的一个广泛应用的案例，用于交通中车辆车牌号的识别，其待识别的字符是面向包括大写英文字母、数字以及少量汉字组成的小字符集。车牌识别技术是模式识别和计算机视觉的一种应用，在现代智能交通系统的研究中是一项重要研究课题，有利于实现交通管理智能化，它使用了多种技术手段，以数字图像处理、计算机视觉、模式识别等技术为基础，是这些技术的综合应用。它的实现是在不影响汽车状态的情况下使用摄像机等设备拍摄车辆图像，然后由计算机自动完成车牌的识别，识别出汽车等车辆所具有的唯一车牌号码，从而可以智能自动管理交通运输，简化交通管理工作的复杂度。LPR 技术作为智能交通系统研究中的最为关键的技术之一，它在交通监控中占有非常重要的地位。现如今 LPR 技术已经广泛的应用于各级公路和城市道路交通管理，具有非常巨大的经济价值和重要的现实意义。因而对包括车牌识别在内的各种字符识别手段是我们需要持续研究不断发展的任务。

当今社会随着信息的爆炸式增长，字符识别要处理的数据及任务可能达到海量级别，传统的单机存储与计算的识别方法已经远远不能满足我们的需求。传统的各种字符识别的研究都是基于小批量模式进行识别研究，所产生的各种字符识别系统针对一些特定应用场景的有限的识别任务进行开发。而针对海量识别任务、大批量分类模式的识别，这些系统识别效率就差强人意了。研究针对海量识别任务、大批量分类模式的字符识别系统是很有意义的。分布式计算方法是研究的一个方向，把大批量的识别计算任务交由多个计算设备分布式处理，是提升识别效率的一个有用的方法。

Hadoop 是目前应用最为广泛的分布式计算平台，利用它能够分布式的高速处理海量的数据。Hadoop 实现了一个分布式文件系统称为 HDFS，其容错性高，可以部署在大量廉价的机器上。同时，其 Map/Reduce 编程模型使用户不需要了解 Hadoop 底层的基础架构就能够进行分布式计算程序的开发，充分利用集群的分布式可靠存储和高效运算[2][3][4]。

将字符识别与 MapReduce 编程模型结合起来是提升字符识别系统效率的一个可行的研究方向，也是本文的主要研究内容。

1.2 国内外研究状况

字符识别技术发展由来已久，最早可以追溯到 19 世纪末期的一项盲人阅读辅助装置的发明专利。1929 年，德国的科学家陶舍克利首先提出了 OCR 的概念，并且申请了专利[5]。但是对于当时的科技发展情况而言，一切仅仅是个设想，直到计

算机的诞生才使得这种设想成为现实，经过近百年的发展，真正的 OCR 系统才在电子计算机诞生后变成为现实[6]。现在 OCR 含义通常是指利用电子计算机来识别各种形式的文字及符号。

早在 60、70 年代，世界各国就开始研究 OCR，初期多以文字的识别方法研究为主，且识别的文字仅为 0 至 9 的数字。以日本为例，1960 年左右开始研究 OCR 基本理论，初期以数字为对象，之后开始有一些简单的产品，如印刷文字的邮政编码识别系统，帮助邮局作邮件自动分拣作业[7]。

1960 年到 1965 年，第一代 OCR 中最早的 OCR 产品应该是 IBM 公司的 IBM1418，可识别印刷体数字、英文字母及部分符号；第二代 OCR 是 60 年代中期到 70 年代初期的一些产品，这些系统能识别印刷体字符和部分手写字符，但识别范围较小，包括数字和少数字母等符号；第三代 OCR 产品起始于二十世纪 70 年代中期，主要解决的就是对于质量较差的文档及大字符集的识别，以及达到非常高的识别精度[8]。

60 年代初期，麻省理工学院的 Murray Eden 指出，所有拉丁字符可以由 18 种笔划组成，而这 18 种笔划又可分解成四个基本笔划，其工作的重要之处在于阐述了所有字符都可由有限数量的基本特征构成，文字识别中结构模式识别的方法来源于这一思想；70 年代初，Parks 等介绍了一种抽取拓扑特征的特征抽取法以及多级结构链接的识别方法；日本对汉字识别进行了研究，并于 1980 年进行了印刷体汉字识别的公开表演;大约在 80 年代初,许多研究者将其它领域的人工神经网络、小波变换、分形、模糊理论等新技术及研究手段引入到 OCR 技术的研究中，并取得了不错的成果；到了 90 年代，Vapnik 等人在有限样本的机器学习问题的研究上取得了很大的进展，并建立了一整套完整的支持向量机的理论体系[9]。

我国在这方面的研究起步较晚，在 70 年代开始对数字、英文字母及符号的识别进行研究；70 年代末开始进行汉字识别的研究；到 80 年代中期汉字识别的研究有了较大的进步，取得了一些成果，一直发展到现在，可以达到对印刷体汉字的较高的识别率，可识别多种字体，对手写体汉字的识别率也达到了比较高的识别率[8]。近年来，通过国内大量研究学者的努力，我国已取得了较大成果，研究开发了各种功能完善并且实用的识别软件或识别系统。

Hadoop 最早起源于 2002 年的 Apache Nutch 项目。2004 年，Google 首先提出了 Map/Reduce 编程模型，并发表了 GFS(Google File System)、BigTable 等相关论文。Hadoop 受其影响，借鉴其思想，开发了分布式文件系统 HDFS 和 MapReduce 编程计算框架等。Hadoop 目前已证明它是成功的，它成功地被雅虎以及雅虎以外的很多公司应用，例如 Last.fm、Fackbook 和《纽约时报》等。

Hadoop 技术在互联网领域得到了广泛的应用，同时也是很多大型互联网公司仍然在研究的领域。如 Yahoo!使用 4000 节点的机群运行 Hadoop，支持广告系统

和 Web 搜索的研究;Facebook 使用 1000 节点的机群运行 Hadoop,存储日志数据,支持其上的数据分析和机器学习；百度用 Hadoop 处理每周 200TB 的数据，进行搜索日志分析和网页数据挖掘工作；中移动研究院基于 Hadoop 开发了大云系统，不但用于相关数据分析，还对外提供服务；淘宝的 Hadoop 系统用于存储并处理电子商务的交易相关数据；国内的高校和科研院所基于 Hadoop 在数据存储、资源管理、作业调度、 性能优化、系统高可用性和安全性方面进行研究，相关研究成果多以开源形式贡献给 Hadoop 社区[11]。

1.3 本文研究内容

本文基于 Hadoop 平台进行了字符识别的研究，首先以面向小字符集的车牌识别为例研究探讨了字符识别的一般流程与方法，然后介绍了 Hadoop 平台的相关知识，最后研究了如何将 Hadoop 与字符识别结合起来。

首先是车牌识别系统。车牌识别所面向的是小字符集，是字符识别的一个特定的应用领域，得到了广泛的研究和使用，它分为几个步骤：图像获取，图像预处理，字符特征提取，字符识别分类，后期处理，输出识别结果，等。

1．图像获取

车牌图像的获取一般是使用交通路段中或一些停车场、小区入口等预先固定设置的摄像机或其他设备实时获取车辆牌照图片并录入计算机或者其他处理识别设备中，等待下一步处理。这一步获得的图片直接影响后续识别步骤的效果。天气状况、光线明暗、拍摄角度以及摄像头分辨率等因素均会影响图片的质量，一般相对清晰可辨的车牌图片自动识别率会更高一些。可以通过使用辅助光源、使用分辨率高的较好的摄像设备、限制车辆行驶速度、限制车辆行驶角度等获得更好的摄像效果。

2．图像预处理

预处理效果的好坏也会直接影响到后续其他步骤识别的性能。预处理的目的是为了滤除噪声、增强有用信息、定位车牌字符的区域、分割车牌、分割车牌字符、对字符进行处理便于识别等。

预处理方法一般车牌定位有图片的灰度化、图片的二值化、字符分割、字符细化、字符方向校正、字符规范化等等.

首先通过对现场采集的各种图像的分析，把采集到的图像进行了分类，使用各种技术手段定位到车牌区域，然后提取出车牌信息并分割车牌字符，最后再对车牌字符进行一些优化处理，包括字符方向校正、字符规范化、字符细化等等。

3．字符特征提取

特征的选取是字符识别的关键。选择出稳定的、有代表性的、数量尽可能少

的特征可以提高识别率、减少识别时间、提高识别效率。特征提取一般分为结构特征和统计特征；按照的特征生成的方式，可以将特征分成三类：局部特征、全局特征、结构特征。局部特征指不考虑字符结构信息，而通过局部变换得到的特征；全局特征指不考虑字符结构信息，而通过全部变换得到的特征；结构特征则指字符笔画结构的特征[12]。

4．字符分类识别

在选取了字符的特征之后，需要选择寻找合适的判别准则，从而判断出文字的特征的分类，进而能对文字进行分类识别。

分类器可分为模板匹配分类器、统计决策分类器、句法结构分类器、模糊判决分类器、神经网络分类器和逻辑推理(或人工智能)分类器等。单一分类器往往难以获得好的分类结果，因此， 多分类器融合的方法在实际应用中常被采用。为了提高分类器的工作速度，分类过程常分为粗分类及细分类[12]。

5．后期处理

经过字符分类识别后，识别的主要任务基本完成了，但是识别率难以达到100%，所以后期还需要人工根据人眼识别、根据上下文信息等，对识别的结果进行修正，这往往能改善和提高字符识别系统的整体性能。后处理可看作是分类器的补充。

6．输出识别结果

经过后期人工修正之后，就可以按照需求将结果以需要的方式输出。

本文将字符识别的过程分为两个部分进行介绍：第二章将介绍字符识别前期处理，包括图像预处理、特征提取等；第三章将介绍字符的分类识别等过程，这部分是识别的重点。

然后本文还针对大字符集的普遍的字符识别系统进行了研究，将字符识别与分布式系统结合，在 Hadoop 平台基础上研究提高字符识别系统的效率的方法。本文介绍了 Hadoop 平台的相关信息。Hadoop 系统是由 Apache 基金会开发的一个分布式系统基础架构，用户可以在不了解分布式底层细节的情况下，开发分布式程序，充分利用了集群的威力高速运算和存储，大大提高了大规模任务处理的效率[13]。本文重点介绍了 Hadoop 的文件系统 HDFS 和 Hadoop 系统的 MapReduce 过程。

Hadoop 的 HDFS(Hadoop Distributed File System)是专门为 MapReduce 作业所设计的文件系统，HDFS 是一个主从结构的体系，一个 HDFS 集群是由一个名称节点(Namenode)和多个数据节点(Datanode)构成。名称节点是一个管理文件的命名空间和调节客户端访问文件的主服务器，它将所有的文件和文件夹的元数据保存在一个文件系统树中，还保存了每个文件包括哪些数据块、数据块分布在哪些数据节点上等信息。数据节点是文件系统中真正存储数据的地方，客户端或者名称节点可以向数据节点请求写入或者读出数据块，数据节点周期性的向名称节点回报

其存储的数据块信息。在一个 PC 集群上部署 HDFS，一般都是使用一台单独的 PC 作为名称节点，集群剩下的 PC 各自作为一个数据节点。

本文介绍了 MapReduce 编程模型的原理，对使用 MapReduce 模型进行分布式程序开发的流程和方法进行了研究。MapReduce 编程模型分为 Map 和 Reduce 两部分。Map 过程中将输入数据划分成大量的数据片段并给每一个数据片段分配不同的 Map 任务。每一个 Map 节点通过 map 函数由分配到的键值对数据集计算生成中间结果集合，中间结果也还是键值对的形式， MapReduce 系统自动对所有中间值进行分组归并，并将其传到 Reduce 方法中。Reduce 操作是对 Map 传来的数据集进行归并，由归约函数指定归并的规则，将中间结果集合中具有相同关键字数据聚集在一起：Reduce 过程接收 Map 阶段输出的键值对，经过用户编写的 Reduce 代码处理后， 将这些值进行合并等操作形成一个更小的值的集合。等所有的 reduce 节点都完成以后，Master 节点会收集所有 reduce 节点返回的结果。根据结果判断是进行下一轮的 map 操作或是 reduce 操作还是返回最终结果。

最后本文对 Hadoop 平台下进行字符识别进行了研究和探讨。

1.4 论文的组织

本论文共分五章加以阐述，其结构如下：

第一章，介绍了课题研究的背景与意义、国内外研究现状、主要研究内容等，主要包括字符识别的研究背景和现状，字符识别常用方法手段等。

第二章，以面向小字符集的车牌字符的识别为例介绍了字符识别的一般步骤，并对其步骤，包括图像预处理、特征提取、字符识别分类等，进行了详细介绍。

第三章，介绍了字符识别的方法，包括模板识别、神经网络，等等。探讨了一些识别的具体方法的概念、发展与基本原理，研究了这些方法用于字符识别的可行性、优点以及实现方法。

第四章，介绍了 Hadoop 系统及 MapReduce 过程，针对面向大字符集的字符识别探讨了字符识别与 Hadoop 结合实现字符识别的分布式处理的意义，以及基于 Hadoop 平台的进行字符识别的识别方法。

第五章，对所做工作进行总结，提出了本文研究的局限性与不足之处，展望了进一步研究的方向与内容。

第二章 车牌字符识别预处理

2.1 引言

车牌识别是字符识别的一个应用，在当今高速发达的交通管理中得到了广泛的关注与使用。车牌识别就是使用计算机或其他设备对行驶车辆的车牌号进行自动识别的技术。这种技术在当今交通监管、车辆管理中发挥着重要作用，是字符识别应用的一个重要领域。

字符识别是模式识别的重要应用领域之一，它涉及到模式识别、图像处理、人工智能、模糊数学、组合论、信息论等多个学科，也涉及到语言文字学、心理学等学科，是一门综合性的技术[14][15]。

有两种基本的模式识别方法：即统计模式识别和结构模式识别。与之相对应的模式识别系统一般都由两个过程所组成[15]：设计过程与实现过程。设计过程就是指通过一定数量的训练样本来设计分类器，实现过程就是指用设计好的分类器对待识别样本进行分类识别。

车牌识别分为几个步骤：图像获取、图像预处理、字符特征提取、字符分类识别、后期处理、输出识别结果等。

1．图像获取

车牌图像的获取一般是使用交通路段中或一些停车场、小区入口等预先固定设置的摄像机或其他设备实时获取车辆牌照图片并录入计算机或者其他处理识别设备中，等待下一步处理。这一步获得的图片直接影响后续识别步骤的效果。天气状况、光线明暗、拍摄角度以及摄像头分辨率等因素均会影响图片的质量，一般相对清晰可辨的车牌图片自动识别率会更高一些。可以通过使用辅助光源、使用分辨率高的较好的摄像设备、限制车辆行驶速度等获得更好的摄像效果。

2．图像预处理

预处理效果的好坏会直接影响到后续其他步骤识别的性能。预处理的目的是为了滤除噪声、增强有用信息、定位车牌字符的区域、分割车牌、分割车牌字符、对字符进行处理便于识别等。

预处理方法一般有车牌定位、图片的灰度化、图片的二值化、字符分割、字符细化、字符方向校正、字符规范化等等。

首先通过对现场采集的各种图像的分析，把采集到的图像进行了分类，使用

各种技术手段定位到车牌区域，然后提取出车牌信息并分割车牌字符，最后再对车牌字符进行一些优化处理，包括字符方向校正、字符规范化、字符细化等等。

3．字符特征提取

特征的选取是字符识别的关键。选择出稳定的、有代表性的、数量尽可能少的特征可以提高识别率、减少识别时间、提高识别效率。一组好的特征需要满足以下三个条件：一是所提取的一组特征直接相互独立，或者说不相关，避免冗余；二是所提取的特征能够有效的减小类内距离，并能增加不同类的类间距离，可以提高识别率、降低误识率；三是选取的特征要简练，特征向量的维数要尽量小，可以提高识别效率、减少识别时间；特征提取一般分为结构特征和统计特征；按照的特征生成的方式，可以将特征分成三类[12]：局部特征、全局特征、结构特征。局部特征指不考虑字符结构信息，而通过局部变换得到的特征；全局特征指不考虑字符结构信息，而通过全部变换得到的特征；结构特征则指字符笔画结构的特征[15]。

4．字符分类识别

在选取了字符的特征之后，需要选择寻找合适的判别准则，从而判断出文字的特征的分类，进而能对文字进行分类识别。常用的判别准则有欧氏距离、相似度等判别准则。

分类器可分为模板匹配分类器、统计决策分类器、句法结构分类器、模糊判决分类器、神经网络分类器和逻辑推理分类器等。单一分类器往往难以获得好的分类结果，因此，多分类器融合的方法在实际应用中常被采用。为了提高分类器的工作速度，分类过程常分为粗分类及细分类。

5．后期处理

经过字符分类识别后，识别的主要任务基本完成了，但是识别率难以达到100%，所以后期还需要人工根据人眼识别、根据上下文信息等，对识别的结果进行修正，这往往能改善和提高字符识别系统的整体性能。后处理可看作是分类器的补充。

6．输出识别结果

经过后期人工修正之后，就可以按照需求将结果以需要的方式输出。

本文将字符识别的过程分为两个部分进行介绍：第二章将介绍字符识别前期处理，包括图像预处理、特征提取等；第三章将介绍字符的分类识别等过程，这部分是字符识别的重点。

2.2 车牌图片预处理

2.2.1 图像灰度化

从现场采集到车辆图片后，需要先进行处理后才能进行下一步操作。

首先需要对图像进行灰度化。灰度图没有颜色的差异，只有亮度的差异，每一个像素的灰度值范围为 0 到 255。其中最高亮度为 255，代表纯白，最低亮度为 0，代表纯黑。使用一个字节的存储空间就可以存储所有灰度值。图片转为灰度图后数据量大幅下降，处理更为快捷。对彩色图像进行灰度化一般有以下几种方法[16]：

1.分量法

将彩色图像的像素中的 R、G、B 三分量其中一个的亮度作为图像的灰度值，图像的 R、G、B 三分量都取这个值，实际选取哪个值作为图像灰度值可根据应用需要选取：

$$f1(i,j)=R(i,j) \quad \text{式(2-1)}$$

$$f2(i,j)=G(i,j) \quad \text{式(2-2)}$$

$$f3(i,j)=B(i,j) \quad \text{式(2-3)}$$

其中 fk(i,j)(k=1,2,3)为转换后的灰度图像在(i,j)处的灰度值。

2.最大值法

图像的灰度值取彩色图像中像素的 R、G、B 三分量的最大值：

$$f(i,j)=max(R(i,j),G(i,j),B(i,j)) \quad \text{式(2-4)}$$

3.平均值法

将彩色图像中的三分量亮度求平均得到一个灰度图：

$$f(i,j)=(R(i,j)+G(i,j)+B(i,j))/3 \quad \text{式(2-5)}$$

4.加权平均法

根据重要性及其它指标，将三个分量以不同的权值进行加权平均。由于人眼对绿色的敏感最高，对蓝色敏感最低，因此，常用下式对 RGB 三分量进行加权平均能得到较合理的灰度图像：

$$f(i,j)=0.30R(i,j)+0.59G(i,j)+0.11B(i,j)) \quad \text{式(2-6)}$$

图 2.1 原始图片

图 2.2 灰度图

2.2.2 图像平滑

图像在获取的过程中，会受到外界天气、光照、环境等因素的影响，在传输的过程中，也会受到各种各样噪声的干扰，包括系统外部的如电磁波等外部噪声，以及系统内部的如摄像机的热噪声、电器的机械运动而产生的抖动噪声等。这些噪声会干扰图像，使图像质量下降，不利于图像分析。

图像的平滑是一种实用的数字图像处理技术，主要目的是为了减少噪声。一个较好的平滑处理方法应该既能消除图像噪声，又不使图像边缘轮廓和线条变模糊[18]。实用的图像平滑算法多种多样，最常见的有：线性平滑、非线性平滑、自适应平滑等[19]。实际使用时应根据图片实际噪声选择合适的平滑算法进行去噪。

2.2.3 图象锐化

由于光线、天气、环境、车辆移动、成像设备自身性能等各种因素的影响，实际拍摄出来的图片可能是模糊不清的，这是不利于识别系统的性能的，给识别

带来一定的困难。因此需要增强图片的边缘信息，突出字符与背景的差异，增强有用的信息。

图像锐化就是补偿图像的边缘、轮廓，增强图像的边缘轮廓及灰度跳变的部分，使图像变得更加清晰可辨。

图像的平滑操作往往也会使得图像中的边界、轮廓变得模糊，为了减少这类不利效果的影响，这就需要利用图像锐化技术，使图像的边缘变的清晰。

图像处理中图像锐化的目的有两个：一是增强图像的边缘，使模糊的图像变得清晰起来；这种模糊不是由于错误操作，就是特殊图像获取方法的固有影响；二是提取目标物体的边界，对图像进行分割，便于目标区域的识别等[20]。

经过平滑的图像变得模糊的根本原因是因为图像受到了平均或积分运算，因此可以对其进行逆运算（如微分运算）就可以使图像变的清晰。

2.2.4 车牌定位

车牌定位就是在车辆图像中定位车牌的具体位置，并从车辆图片中截取出来，进行下一步的处理。车牌定位的方法多种多样，归纳起来主要有基于纹理特征分析的方法、基于边缘检测的方法、基于数学形态学定位、基于小波分析定位以及基于彩色图像定位等，这些方法各有所长[21]。

本文以边缘检测方法为例介绍车牌定位的方法。

所谓的边缘就是指其图片中某些像素的周围像素的灰度有阶跃变化的这些点的集合。边缘的两侧的像素区域分属于两个区域，每个区域内灰度基本均匀一致，而这两个区域之间的灰度在特征上可能相同，也可能存在一定的差异。边缘检测的方法有多种[22]，例如使用 Roberts 边缘算子、Prewitt 算子、Sobel 算子以及拉普拉斯边缘检测等各种方法来检测，这些方法都是利用图像中边缘处的像素灰度变化剧烈来检测图像的边缘。各个算子检测的效果不同，适用的情况也不尽相同。Roberts 边缘算子是一种利用局部方差算子寻找边缘的算子，定位比较精确；Prewitt 算子和 Sobel 算子对噪声有一定的抑制能力，但不能完全排除伪边缘；拉普拉斯算子是二阶微分算子，对图像中的阶跃型边缘点定位准确且具有旋转不变性，但容易丢失一部分边缘的方向信息，同时抗噪能力较差[21]。实际使用时可以针对不同的环境和要求来选择适合需求的情况的算子来对图像进行边缘检测。

以下是使用 Robert 算子进行边缘检测定位车牌的过程：

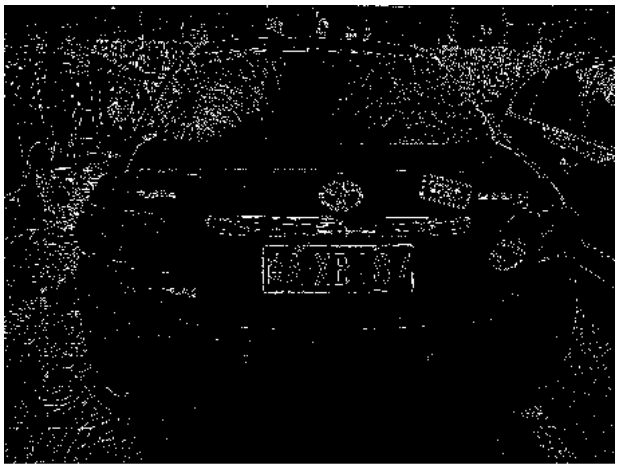


图 2.5 使用 Robert 算子进行边缘检测

图 2.6 形态学定位矩形

定位车牌区域后，就可以将车牌单独截取出来进行下一步处理。

图 2.7 灰度化后的车牌图片

2.2.5 图像二值化

图像的二值化是用来研究灰度图像的一种常用方法。

一幅图像包括目标物体、背景还有噪声，要想从多值的数字图像中直接提取出目标物体，最常用的方法就是设定一个阈值，用来将图片中的像素分为高于阈值的和低于阈值的两类，图片中像素高于给定阈值的作为前景，转化为白色像素（或黑色像素），像素值低于给定阈值的作为背景，转化为黑色像素（或白色像素）。这就是图像的二值化。经过二值化后的图片中数据量大大减少，目标与背景明显分离，可以减少背景和噪声的干扰。

二值化算法中阈值的选取有很多种方法，比较常用的阈值选取方法有：双峰法、迭代法和 OTSU 法等[23]。

1.双峰法

双峰法就是认为图像由前景和背景组成，在图像的灰度直方图上，前后二景形成两个高峰。首先计算图像的灰度直方图，然后找到直方图上的两个最高的高峰，然后找到两个高峰之间的波谷，这个波谷就是所求的阈值。

2.迭代法

迭代法就是使用迭代的方式，使得阈值逼近一个目标，这个目标将使得图像的前景和图像的背景的平均灰度值的平均值作为阈值并不再发生变化。其实现方法是首先求出图像的灰度直方图；然后找到最大灰度与最小灰度，取最大灰度值与最小灰度值的平均值为初始阈值；然后根据阈值将图像分为前景与背景，分别求出前景与背景的灰度平均值，然后以这两个灰度平均值的平均值为新阈值；如此重复迭代，直到阈值不再变化为止。

3.OTSU 法（大津法）

首先计算图像的灰度直方图，记 t 为前景与背景的分割阈值，前景点数占图像比例为 w_0，平均灰度为 u_0，背景点数占图像比例为 w_1，平均灰度为 u_1，图像的总平均灰度即为：$u=w_0*u_0+w_1*u_1$。从最小灰度到最大灰度遍历 t，找到使得 $g=w_0*(u_0-u)^2+w_1*(u_1-u)^2$ 最大的 t 值就是分割阈值。

图 2.8 OTSU 法二值化车牌图像

2.2.6 字符分割

在车牌区域定位之后，需要再将车牌区域分割成单个字符，然后再进行识别。

字符分割一般采用垂直投影的方法进行分割。因为车牌上每个字符之间都有一定的空隙，将车牌区域图像投影到垂直方向上，就会出现多个波峰与波谷，两个波峰之间的波谷就是两个字符之间的空隙，而且因为车牌字符是基本等间距的，因此两个波峰、两个波谷之间也基本是等间距的，利用这个，可以在实际检测时排除掉小块污点噪声的影响，进而达到较好的字符分割效果。利用垂直投影法对复杂环境下的汽车图像中的字符分割有较好的效果。

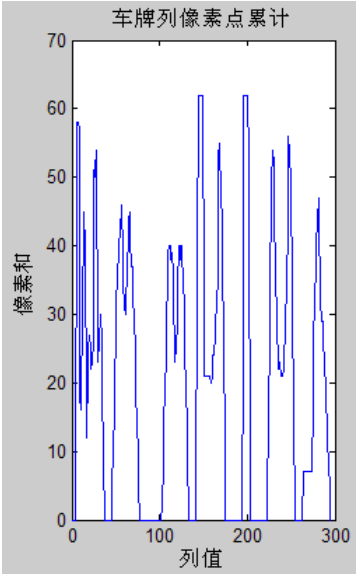


图 2.9 垂直投影图

图 2.10 字符分割后图像

2.2.7 其他处理

根据实际情况，图像的处理可能还需要其他一些步骤。

实际拍摄的图像会受到车辆倾斜方向、摄像设备拍摄角度的影响以及其他因素的干扰，因此可能还需要进行一些其他额外的处理，以便提高识别的效果：

1.字符倾斜校正

实际拍摄的图片可能会有一定角度的倾斜，这样会使得识别系统产生误差，甚至会造成错误的判别。因此，在车牌字符分割识别前，应该对车牌进行角度倾斜的校正。校正的方法可以首先根据车牌的边框算出车牌在水平角度上偏差的夹角，然后根据这个夹角对车牌图片进行水平方向的校正操作，使得识别系统识别效果更好。

2.字符归一化

由于实际拍摄环境多样、设备不同，因此拍摄的图像中截取的车牌大小各不相同，在识别的过程中，需要将截取的字符进行归一化，这样提取的特征值才有普遍性、比较性。

归一化是指将各个不同大小的字符图像统一到固定大小的字符点阵中，以便于下一步识别的需要。归一化包括字符位置的归一化与字符大小的归一化。字符位置归一化是为了消除各个字符点阵位置上的偏差，需要把整个字符点阵移动到统一的位置上。字符大小归一化则是指针对各个不同大小的字符做统一，使之成

为尺寸大小统一的字符。

归一化在字符的预处理中是比较重要的一步处理，有利于特征提取和识别的进行。

2.3 特征提取

在经过对原始车辆图片进行的一系列处理后，就得到了单个字符的较为清晰、统一的点阵图像，然后就可以对这些字符进行识别了。

字符识别首先需要进行字符特征值的提取操作。特征提取的主要目的是从原始数据中抽取出用于区分不同类别的本质特征。好的特征应该具备以下的几个条件：具有较好的类内一致性和类间区分度；具有较好的稳定性和较好的抗噪能力；具有较好的平移不变性、旋转不变性、尺度不变性等特点[24]。

按使用特征的不同，大体可以分为结构特征和统计特征两大类。

结构特征主要包括字符的笔画、字符轮廓、字符的骨架等，其中骨架特征在字符识别中，占有结构特征的重要地位；骨架特征还可以进一步划分为字符的特征点、字符笔画端点、字符笔画交叉点、字符笔画的转折点等等。字符的统计特征是将字符点阵看作一个整体，从这个整体上经过大量的统计而得到的特征。统计特征又可以分为全局统计特征和局部统计特征。全局统计特征就是对整个字符图像的全局信息进行某种变换得到的特征作为图像的一种特征，全局特征包括 Fourier 变换、余弦变换、小波变换、矩特征以及笔画的密度特征等。局部统计特征就是在局部特定的位置对一定大小的范围内的图像进行某种变换得到的特征，它包括局部网格特征、投影特征等等。

2.3.1 结构特征

结构特征直接反映了字符的笔画、字符轮廓、字符的骨架等，是字符结构的一种模型化的描述。使用结构特征进行字符识别的方法是：首先提取出字符的基本笔划或笔段作为结构特征基础单位，然后由这些笔划笔段的统计特征、位置特征等的组合来描述字符，进而识别出字符。下面给出在字符识别中比较常用的一些字符结构特征：

1.字符节点特征

字符的节点特征是字符的一种重要的结构特征，它是指笔划的端点（字符笔划中的起始点或终点）、拐点(字符笔划中方向明显转折的点)、叉点(多个笔划相交的点，包括三节点、四节点等)等。常用于字符识别的有字符的端点、拐点、三节点、四节点的个数与位置、字符的横向切割交点序列、字符纵向切割交点序列等

特征，这些特征在字符细化后是很容易得到的。

1	2	3
8	P	4
7	6	5

图 2.11 节点示意图

由上图来计算 P 点的节点数：1~8 是 P 点的八个邻接点，从八个邻接点中任一点开始沿顺时针或逆时针查看一周，记下由黑到白的次数（白色为字符，黑色为背景）。一次说明 P 为端点，两次并且不在同一条直线上说明是拐点，三次就是三节点，四次则对应四节点，等等。

而通过字符局部区域内的点特征进行计算，就可以得到字符的局部点特征。字符局部点特征在对相似的字符的区分中比较有效。

2.字符的空洞与缺口特征

字符的空洞和缺口特征也是字符的非常重要的结构特征。譬如“0”、“9”、“京”等字符内部就有明显的空洞。字符“3”、“5”、“豫”等则有明显的缺口。这些特征在字符的分类判断中也是很有效的字符结构特征。

3.字符笔道密度特征

字符笔道密度特征就是指沿在字符中某水平方向、垂直方向或者其他方向用一条或多条线来切割字符时得到的交截笔划的数量特征。这种特征的优点是无需进行细化处理就可以迅速得到。譬如，数字 0 的横向笔道密度在最上方、中间、最下方分别是 1、2、1，其纵向笔道密度在最左方、中间、最右方也分别是 1、2、1。字符的笔道密度特征反映了字符的内部结构特征，也是字符结构特征的重要的组成部分之一。

基于字符结构特征进行字符识别的方法多种多样，但是因为它的识别率与识别速度都不如基于字符统计特征的识别方法。其主要原因有：结构特征对噪声比较敏感，容易收噪声影响从而导致识别错误；字符的结构特征提取相对困难，稳定性不如字符的统计特征。因此，在使用字符的结构特征时，需要使用种种手段消除噪声影响，提高结构特征的获取准确率。

2.3.2 统计特征

字符的统计特征是将字符点阵看作一个整体，从这个整体上经过大量的统计而得到的特征。字符的统计特征根据特征抽取区域的不同又可分为全局字符统计特征和局部字符统计特征两大类[10]；字符的全局统计特征是将整个字符点阵整体作为统计识别研究对象，从整体上抽取特征；字符的局部统计特征是将字符点阵图像划分成不同区域或者叫做网格，然后在每个小区域内分别抽取统计特征。一

些常见的统计特征如下：

1．字符外围轮廓特征

字符的外围轮廓是指字符轮廓边缘中抽取的特征，这些特征不受字符内部结构的影响，即使在字符内部笔划粘连不清的情况下，其外围轮廓信息依然是比较完整的。字符的外围轮廓特征是字符统计特征的一个重要方面，可以作为字符识别的一个判定，这种特征非常适合于作为粗分类的特征，但细分的能力不强，可以配合其他特征进行字符识别，也可以只作为粗分类的特征。

2．字符的交截特征

字符的交截特征就是指某一行或者一列中字符像素与背景像素交替出现的次数及顺序。交截特征可分为横向交截特征和纵向交截特征。字符的交截特征也是字符的统计特征中重要的一种，他也反映了字符内部结构。

3．字符的跳变特征

字符的跳变特征是指字符的边缘发生不连续的跳跃变化所表现出来的一种特征。通过对相邻两行或者两列的像素数或者像素位置进行比较，若其变化大于某一阈值，即可认为这处发生了发生跳变。跳变特征又可分为左右跳变特征，它也是一种重要的统计特征。

4．字符宽度特征

字符的宽度特征可以分为全局宽度特征和局部宽度特征。从字符点阵图像的左右两个方向分别扫描字符，则每一列中最上面的字符像素和最下面的字符像素直接的距离就是字符在这列的宽度，当这个宽度小于给定阈值时，判断这些小于给定阈值的列数占总数的百分比，这样就得到了全局宽度特征。字符局部宽度特征则是指字符宽度小于某一阈值的列数占某一部分的比例。

5．字符变换特征

字符的变换特征是指通过对图像进行某种变换，如使用傅里叶变换、小波变换等对字符点阵进行变换，从而提取出来的特征值，这也是一种重要的字符统计特征。

6．字符笔划密度特征

字符的笔划密度特征一般认为是指：在字符点阵图像的某一范围内，以一定数量的扫描线沿水平、垂直或对角线等方向对字符点阵进行扫描时的穿透次数。字符笔划密度特征描述了字符的各部分笔划的疏密程度[25]，一定程度上反映了字符的结构，为识别工作提供了比较完整的信息。在图像质量较好的情况下，这种特征比较稳定，适合于进行字符的识别。

除此之外，字符还有许多其他的其他统计特征。由于统计特征对实际环境中的噪声有较好的适应性，受外界噪声的影响较小，其性能高于其它方法，因此统计方法是目前采用最多的方法。

2.3.3 统计特征与结构特征的结合

结构特征与统计特征各有其优缺点。使用结构特征对字符的结构比较敏感，对相似字符具有较强的区分能力，但是结构特征相对难以抽取，也不太稳定。使用统计特征在抗噪、抗干扰方面具有良好的性能，但难于描述字符内部复杂的几何及拓扑结构特性，因此对于相似的字符具有较差的区分能力。

由于两种特征各有其优缺点，因此目前很少单独使用其中的一种，通常是使用这两种特征的融合。

统计特征与结构特征的融合，一方面是特征的融合：在抽取特征的过程中，注意抽取能够反映字符的结构信息的统计特征，这种融合方式的其中一个典型产物就是网格化特征：首先将字符点阵图像等分或按结构等分的划分为若干个区域，每个区域就是一个网格，在每一个网格内寻找字符图像的各种结构的和统计的特征，其特征以网格为单位进行统计。

统计特征与结构特征的融合，另一方面就是识别方法的融合：结构特征和统计特征两种方法既可以采用串联的方式，即先使用统计特征的方法进行第一步的粗分类，再使用结构特征的方法进行第二步的细分类，进而分类识别出字符；还可以将统计特征和结构特征两种方法并联一起使用，综合集成、同时考虑。两种特征的融合是当今最常使用的方法，是研究的重点。

2.4 本章小结

本章以车牌识别为例对字符识别的一般流程进行了介绍，详细介绍了字符识别的前期预处理过程和字符特征的提取。车牌预处理过程包括车牌图像的灰度化、车牌的平滑、锐化、车牌的定位、字符的分割、字符的归一化、字符的细化等等。字符的特征包括结构特征和统计特征，实际使用时多将这两种特征融合使用，能够起到更好的效果。

字符特征提取后就可以进行识别，这将放在下一章里详细介绍。

第三章 字符识别方法

3.1 引言

本文在上一章中对车牌识别的整体流程进行了介绍，并详细介绍了车牌字符识别的前处理过程，包括对采集的图片进行灰度化、车牌定位、车牌二值化、字符分割、字符特征提取等等一系列的处理，最终得到字符的特征向量。在特征提取以后，便可以根据提取的特征对被识别字符的分类识别。

字符识别是模式识别的一种，属于模式识别的范畴，其基本思想是匹配判别。具体过程为：首先对分割后得到的字符进行特征提取，得到代表该字符模式的特征；然后将得到的特征和标准模板字符的特征逐一进行对比；接着根据给定的判决规则，找出与待识别字符最为接近的标准模板字符；最后就可以判断待识别字符属于这个标准模板字符类。

一般字符识别方法就是通过某种分类算法判别出待识别字符的所属类别，然后再将判别的结果输出。常用的字符分类识别的算法有很多种，最早的字符识别是利用模板匹配算法来实现的。后来随着技术的进步和发展，人们有发明了多种多样的不同识别算法，包括用几何矩、特征变换、直方图、Fisher 线性判别、非线性判别函数、人工神经网络、支持向量机等方法进行字符识别。

目前常用的字符识别方法有模板匹配法、贝叶斯法、几何分类法、人工神经网络识别法、支持向量机识别法等等。下面对其进行简要的介绍：

1．模板匹配法[26]

模板匹配法是将训练样本集中的每个样本都作为标准模板，用待测样本与所有模板做比较，找出最相似的模板，将这个模板所属类别作为自己的类别。例如 A 类有 5 个训练样本，B 类有 8 个训练样本，总共就有 13 个标准模板。待测样本所有标准模板进行对比，如果最相似的模板属于 A 类则待测样本属于 A 类，否则就属于 B 类。从原理上说模板匹配法实现上最简单，当字符较规则时，对字符图像的缺损、污迹、干扰适应力强。但是模板法有两个缺点：存储量大、计算量大。因为要存储的模板很多，所以存储量比较大。而每个测试样品都要对每个模板计算一次相似度，因此在模板数量很大时，计算量也很大。

2．贝叶斯法[7]

Bayes 法是模式识别的一个基本方法，其最有代表性的两种决策方法是基于最小错误率的 Bayes 决策与基于最小风险的 Bayes 决策。用 Bayes 方法进行分类

时首先需要知道各个类别的先验概率和总体的概率分布。Bayes 法中基于最小错误率的 Bayes 决策是根据样本库中的样品的先验概率及类条件概率密度函数，用 Bayes 公式计算出待识变量在各状态下的后验概率，哪个状态的后验概率最大，就认为它属于哪一类别。基于最小风险的 Bayes 决策则是根据样本库中的样品的先验概率、类条件概率密度函数、损失函数，依据 Bayes 公式算出后验概率，再根据后验概率和损失函数计算出每种识别结果的损失。其中，损失最小的那个就作为最后的识别结果。依据 Bayes 判别决策设计的分类器理论上讲具有最优的性能，但问题的困难是难以获得各个类别的先验概率、类条件概率密度、损失函数，这就导致用 Bayes 法作出的分类往往并不是最优的，有时甚至比简单方法作出的分类效果还差。

3．几何分类法[7]

一个模式通过某种变换映射为一个特征向量后，该特征向量可以理解为特征空间的一个点。在特征空间中，属于一个类的点集，在某种程度上总是与属于另一个类的点集相分离，即各个类之间是确定可分的。因此，如果能够找到一个分离函数，把不同类的点集分开，则分类任务就解决了。几何分类器不需要知道类条件概率密度，它是通过几何的方法，把特征空间分解为对应于不同类别的子空间。按照分界函数的形式，几何分类法又分为线性判别函数分类法和非线性判别函数分类法两类。由于线性判别函数计算简单，在计算机上容易实现，因此在实际中被广泛应用。

4．人工神经网络法[7]

人工神经网络就是以计算机网络系统对生物的神经网络进行模拟的智能计算系统，它是对生物神经网络的一种抽象和模拟，它反映了人脑的某些基本特征，但又并不是对人脑部分的真实再现，只是生物神经网络的某种抽象、简化和模仿。目前在字符识别中应用最多的是多层感知机网络。神经网络使用广泛，目前在很多工程应用中取得了成功，但其自身还有一些问题没有得到解决。首先，网络节点的选择比较困难。这是由于在网络中，每个输出节点代表一个类别，类别数太大会造成网络结构的异常复杂，类别数太小又不能准确进行分类。其次，网络的权重不易初始化。如果权重初始化的不合理，误差函数将会无法收敛。

5．支持向量机法[7]

支持向量机是 Vapnik 等人在统计学习理论的基础上提出的新一代机器学习算法，在解决小样本、非线性及高维模式识别问题中表现出了许多特有的优势，已经在手写数据库分类、人脸及 3D 图像识别、语音识别和机械故障诊断等领域取得了不错的效果。SVM 是机器学习领域若干标准技术的集大成者。它集成了最大间隔超平面、Mercer 核、凸二次规划、稀疏解和松弛变量等多项技术。支持向量机从本质上讲是一种前向神经网络，根据结构风险最小化准则，在使训练样本分

类误差极小化的前提下，尽量提高分类器的泛化推广能力。从实施的角度看，训练支持向量机的核心思想是：用线性变换将输入空间变换到一个高维空间，在这个高维空间中求最优线性分类面。由于它可以得到全局最优解，所以基于支持向量机的手写字符分类器能够吸收字符的变形，从而具有较好的泛化性和推广能力。

下面我们对字符识别中应用最为广泛的两种识别方法：模板识别、神经网络进行详细介绍。

3.2 模板识别

3.2.1 模板识别概述

模板匹配方法是一种经典的模式识别方法，是最直接的识别字符方法。它具有实现简单的优点，但是却有存储空间大、计算复杂的缺点。模板匹配法的实现方式取得待识别模式和模板模式之间的相似度或距离，则相似度最大或者距离最近的分类模式就是待识别的模式。对于字符识别来说，首先建立标准字符的模版库，将待识别字符图像进行预处理并归一化为和标准字符模板同样大小；然后将归一化后的字符图像与所有的模板字符图像进行匹配；最后根据匹配结果选择最为相似的模板，这个模板所属的分类就是所求结果。实际应用中为了提高字符识别的正确率往往需要使用大量的标准字符模板进行匹配，这样能够得到更好的识别结果。

模板匹配法有两种匹配方法：一种是进行特征提取后用特征与标准特征模板匹配；一种是不进行特征提取，直接用待识别字符点阵与模板字符点阵进行匹配[27]。显然后一种方法计算量大，但是实现简单，而前者需要复杂的前处理和特征提取，但是计算量相对较小。

模板匹配识别方法选取同一类字符中具有代表性的、共有的、相对稳定的并且分类性能好的特征作为特征向量来进行识别。常使用的特征包括字符的结构的特征和统计特征等，有字符二维平面的位置特征、字符在水平或者垂直方向投影的直方图特征、矩特征和字符经过频域变换或其它形式变换后的特征等。首先使用大量模版字符的特征经过提取、学习和分类形成关于字符的模板，这些模板信息存储在识别系统中。待识别字符图像在识别时首先经过相同的特征提取处理，得到同样的特征向量，然后与系统存储的字符模板特征进行匹配比较，根据比较结果确定字符最终的分类，达到识别的目的。

由于车牌字符中的字符集属于有限的少量字符集，总字符数不多于 100。如果前面的车牌图片预处理工作做得比较好，得到的字符图片质量较高，图片的水平倾斜度较小。这种方法的识别率基本能达到要求，可以获得较高的识别率。

3.2.2 模板匹配识别字符

模板匹配法有两种匹配方法：一种是进行特征提取后用特征与标准特征模板匹配；一种是不进行特征提取，直接用待识别字符点阵与模板字符点阵进行匹配[27]。

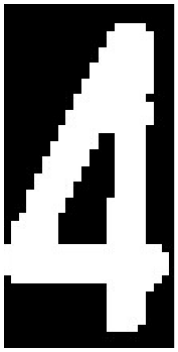
1.不进行特征提取进行匹配，就是将模板字符图片进行预处理，得到大小归一化后的二值图片直接作为模板，不需要提取其特征值；识别时将待识别字符进行同样处理得到大小归一化的点阵图像，然后直接与模板点阵图像进行匹配。

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	1	1	1	0	0
0	0	0	0	0	1	1	1	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	0	1	1	0	0
0	0	1	1	1	0	1	1	0	0
0	0	1	1	0	0	1	1	0	0
0	0	1	1	0	0	1	1	0	0
0	1	1	0	0	0	1	1	0	0
0	1	1	0	0	0	1	1	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0

(a)字符“4”模板图片　　(b)模板“4”归一化为 20×10 二值点阵

图 3.1 字符模板及归一化点阵

上图是作为标准模板的字符“4”的图像，以及模板图像进行预处理并归一化后得到的 20×10 的二值点阵。在识别字符时可以将待识别的字符图像做同样的处理，得到同样的点阵。



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	1	1	0
0	0	0	0	0	1	1	1	1	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	0	0	1	0	0
0	0	1	1	1	0	0	1	0	0
0	0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	1	1	0	0
0	1	1	0	0	0	1	1	0	0
0	1	1	0	0	0	1	1	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0

(a)待识别字符图片　　(b)待识别字符归一化后 20×10 二值点阵

图 3.2 待识别字符图片及其归一化后点阵

上图是待识别的字符图片及其归一化处理后点阵。将该 20×10 点阵和所有归一化后的模板字符点阵进行匹配，找到最接近的类别就是待识别字符所属类别。

将上面的点阵与归一化后字符“4”的点阵计算欧式距离：

$$D = [\sum_{i=1}^{20}\sum_{j=1}^{10}[A(i,j) - B(i,j)]^2]^{1/2} \quad \text{式(3-1)}$$

其中 A 是模板点阵，B 是待识别点阵。结果为 D=3.4641，在与所有模板的匹配中最小，最接近模板“4”，因此这个字符图像的识别结果为字符“4”。

在车牌识别中，由于车牌字符中的字符集有限且量少，总字符数不多于 100，而且车牌字符格式统一标准，因此使用这种模板匹配方法的识别率基本能达到要求，可以获得较高的识别率。但是这种不提取字符特征直接进行匹配的方法计算量较大，不适合大批量识别。通常识别时，都需要进行特征值提取，使用特征向量进行识别。

2.提取特征值进行匹配的方法，就是对模板字符图像进行预处理归一化后得到字符点阵，提取其结构的或者统计的或者二者融合的特征向量，作为标准模板。对待识别的字符图像进行同样的处理，得到的点阵提取与模板相应的特征向量，然后与标准模板进行匹配。

这里使用了结构特征和统计特征来进行字符识别，特征提取示意图 3.3 所示。

如图 3.3 中所示，提取特征值。其中，从上下左右 4 个方向提取字符轮廓特征，每个方向均匀取 4 个特征，共 16 个特征；从对角线和反对角线方向提取字符贯穿像素数，每个方向均匀取 4 个特征，共 8 个特征；将字符图像均分为 8 份，统计每一份中字符像素数，共 8 个特征。总共提取 32 个特征，组成 32 维特征向量，用来识别。

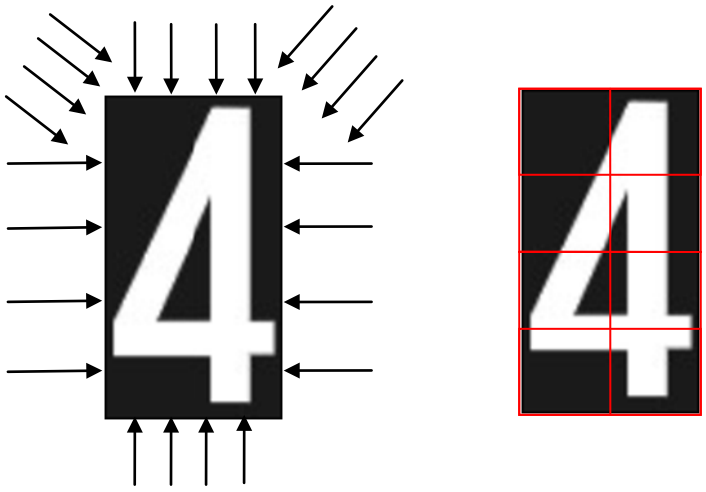


图 3.3 字符特征提取

用模板匹配法来识别字符方法为：将模板图片和待识别字符图片均进行处理后，提取其上诉 32 维特征向量，将待识别特征向量与所有模板特征向量进行比较，获取其欧氏距离，取其中最小的，作为识别结果。

3.3 神经网络识别

3.3.1 人工神经网络概述

计算机是近现代对人类社会做出最大贡献的发明之一，它的产生大大推动了社会的发展。计算机最突出的能力就是其计算能力。对计算机与人的计算能力进行比较，就可以发现，其差别是惊人的。2010 中国高性能计算机 TOP100 排行榜的榜首是经过技术升级优化后的“天河一号”超级计算机系统，它的峰值性能每秒 4700 万亿次、持续性能每秒 2507 万亿次。这个速度与人脑进行算术运算的能力差别巨大。但是，在另外一些方面，计算机却是远远比不上人脑。

现代计算机的计算速度是人脑的几百万倍，对于那些特征明确，推理或运算规则清楚的可编程问题，可以高速有效地求解。但是迄今为止，计算机在解决与形象思维和灵感思维相关的问题时，却显得无能为力。例如人脸识别、骑自行车、打篮球等涉及联想或经验的问题，人脑可以从中体会那些只可意会、不可言传的直觉与经验，可以根据情况灵活掌握处理问题的规则，从而轻而易举地完成此类任务，而计算机在这方面则显得十分笨拙[29]。

一个人能能够很容易的识别脸孔、理解语言、辨别物体，而使用计算机来做的话就会非常复杂。人脑能够大规模的并行处理数据，人脑神经元之间传递神经冲动是以毫秒计的，比普通的电子计算机都要慢得多。但人们通常能在一秒钟内对外界事物做出判断和决策，这是传统的计算机或人工智能所难以做到的。人脑还有很强的容错性，善于联想，概括，类比和推广。人脑每天有大量的细胞正常死亡，但这并不影响大脑正常的功能。而在常规计算机里，程序中微小的错误或者元器件的局部损坏，都可能引起严重的后果。因而，对人脑的研究势成必然，这就产生了人工神经网络这个全新的研究领域。

人工神经网络就是以计算机网络系统对生物的神经网络进行模拟的智能计算系统，它是对自然神经网络的一种抽象和模拟。网络上的每个结点相当于自然神经网络中的一个神经元，可以存储和处理一定的信息，并与其它结点进行并行工作。

神经网络理论自 20 世纪中期提出以来，取得了一系列的研究成果。近年来，随着计算机技术和非线性科学的发展，神经网络理论的研究又进入一个新的高潮，其应用已经渗透到各个领域，并在智能控制、模式识别、计算机视觉、生物医学工程等方面取得了巨大贡献[30]。

人工神经网络是对人脑功能的模仿，有很多人脑的优点。

从速度的角度来看，人脑神经元传递信息的速度要远低于计算机，但是由于人脑是一个大规模并行与串行处理系统，在许多问题上可以作出快速判断、决策和处理，速度则远高于串行结构的普通计算机。

人工神经网络具有初步的自适应与自组织能力，在学习或训练过程中通过改变突触权重值，适应周围环境的要求与变化。同一网络因学习方式及内容不同可具有不同的功能。人工神经网络是一个具有学习能力的系统，可以发展知识。通常，它的学习训练方式可分为两种：一种是有监督或称有导师的学习，是利用给定的样本标准进行分类或模仿；另一种是无监督学习或称无为导师学习，只规定学习方式或某些规则，具体的学习内容随系统所处环境而异，系统可以自动发现环境特征和规律性，具有更近似人脑的功能。

3.3.2 人工神经网络原理

生物神经元是脑组织的基本单元。其中树突是细胞的输入端，通过细胞体间联结的节点接受周围细胞传出的神经冲动。轴突相当于细胞的输出端，其众多的神经末梢为信号的输出端，用于传出神经冲动。

神经元是广泛互联与并行工作的，其分布式存储的结构特点会在两个方面表现出良好的容错性：一方面，由于信息的分布式存储，当网络中部分神经元损坏时不会对系统的整体性能造成影响，这一点就像人脑中每天都有神经细胞正常死亡而不会影响大脑的功能一样；另一方面，当输入模糊、残缺或变形的信息时，神经网络能通过联想恢复完整的记忆，从而实现对不完整输入信息的正确识别，这一特点就像人可以对不规范的手写字进行正确识别一样[29]。

生物的神经网络并非所有神经元每次都一样地工作，各神经元参与不同工作的程度不同，在人工神经网络中就使用权值来控制结点参与工作的程度，正权值相当于神经元突触受到刺激而兴奋，负权值相当于受到抑制而使抑制神经元兴奋。

人工神经网络的基础就是一个个模拟自然神经元的人工神经元。

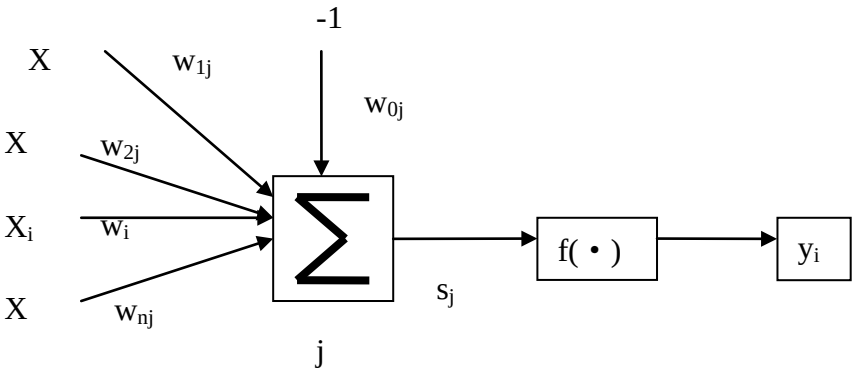


图 3.4 人工神经元模型

每个神经元有多个输入(X_i,i=1,2…n)一个输出 y_j。其中 w_{0j} 为阈值。其计算过程为：

$$y_j = f(\sum_{i=1}^{n} x_i \cdot w_{ij} - w_{0j}) \quad \text{式(3-2)}$$

人工神经元的输出是所有输入以及阈值的加权和的某种函数映射，受所以输入、所有连接权值以及阈值的影响。其中 *f(x)*为神经元的变换函数，最常见的变换函数[29]有以下四种形式：

1. 阈值型变换函数

阈值型变换函数采用了如下的单位阶跃函数：

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad \text{式(3-3)}$$

这种神经元称为阈值型神经元，这是神经元模型中最简单的一种

2. 非线性变换函数

非线性变换函数最常用的非线性变换函数是单极性的 Sigmoid 函数曲线，简称 S 型函数。其特点是函数本身及其导数都是连续的，因而在处理上十分方便。单极性 S 型函数定义如下：

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{式(3-4)}$$

3. 分段线性变换函数

神经元的输入与输出在一定区间内满足分段线性关系。单极性分段线性变换函数的表达式如下：

$$f(x) = \begin{cases} 0, x \leq 0 \\ cx, 0 < x \leq x_c \\ 1, x_c < x \end{cases} \quad \text{式(3-5)}$$

其中，c 为线性段的斜率

4. 概率型变换函数

概率型变换函数神经元模型输入与输出的关系是不确定的，需用一个随机函数来描述其输出状态为 1 或为 0 的概率。设神经元输出为 1 的概率为：

$$P(1)=\frac{1}{1+e^{-x/T}} \quad \text{式(3-6)}$$

神经网络是由大量神经元组成庞大的网络，能实现对复杂信息的处理与存储，并表现出各种优越的特性。根据神经元之间连接方式，可将神经网络结构分为两大类[29]：

1. 层次型结构

层次型结构的神经网络将神经元划分成若干层，包括输入层、中间层（或称隐层）和输出层，各层顺序相连。输入层神经元负责接收输入信息，并传递给中间各隐层神经元。隐层负责信息变换，可为一层或多层。最后一个隐层传递信息到输出层各神经元并经进一步处理后，由输出层输出信息处理结果。

2. 互连型结构

相对于层次型结构的神经网络将神经元分层划分，互连型网络结构的网络中任意两个节点之间都可能存在连接路径，其连接更加复杂。

根据神经网络内部信息的传递方向，可将神经网络分为两种类型[30]：

1. 前馈型网络

前馈神经网络，就是其网络信息处理的方向是从输入层到各隐层再到输出层逐层前向进行。前馈网络中某一层的输出是下一层的输入，信息的处理具有逐层传递进行的方向性，一般不存在反馈环路，因此这类网络很容易串联起来建立多层前馈网络。

2. 反馈型网络

在反馈网络中所有节点都具有信息处理功能，而且每个节点既可以从外界接收输入，同时又可以向外界输出信息，即存在反向信息流动，所以称之为反馈神经网络。

神经网络研究的重要的一方面是神经网络的学习和训练。人工神经网络的功能特性由其神经元连接的拓扑结构和神经元之间的连接强度决定。连接权值的整体反映了神经网络的知识存储。神经网络能够通过提供的学习样本进行不断的学习训练，不断改变网络的连接权值及其拓扑结构，使得网络的输出不断接近期望输出，这一过程就是神经网络的学习或训练。

神经网络的学习算法很多，根据一种广泛采用的分类方法，可将神经网络的学习算法归纳为 3 类[29]：有导师学习、无导师学习和灌输式学习。

有导师学习也称为有监督学习，采用的是纠错的学习模式。学习训练的过程中需要不断地提供成对的输入模式和期望输出模式，这就是教师信号。将实际输出结果同期望输出结果进行比较，按一定的规则调整神经网络各层连接权值，使得神经网络的输出更接近期望结果。当网络对于各种给定的输

入均能产生所期望的输出时，即认为网络可以用来进行工作了。

无导师学习也称为无监督学习，学习过程中不断地给网络提供输入，网络能根据其内部结构和学习规则，在输入信息流中发现可能存在的模式和规律，自动调整权值，这个过程叫做网络的自组织，最终使得网络能够进行自动分类。

灌输式学习是将网络设计成能识别特别的例子，当给定这些特定的输入信息时，网络就能回忆起来这些给定的例子。其网络权值不是通过训练逐渐形成的，而是通过某种设计方法得到的，权值一旦设计好就不再变动。

有导师学习和无导师学习神经网络网络一般分为训练阶段和工作阶段两部分。训练学习的目的是为了从训练数据中提取隐含的知识和规律，并存储于网络中供工作阶段使用。

3.3.3 BP 神经网络

BP (Back Propagation)神经网络全称为基于误差反向传播算法的人工神经网络，是 1986 年由 Rumelhart 和 McCelland 为首的科学家小组提出来的，是一种按误差反向传播算法进行学习训练的多层前馈网络，是目前应用最广泛的神经网络模型之一[28]。

BP 神经网络模型拓扑结构包括输入层（input）、隐层(hide layer)和输出层(output layer)（如图 3.5）。

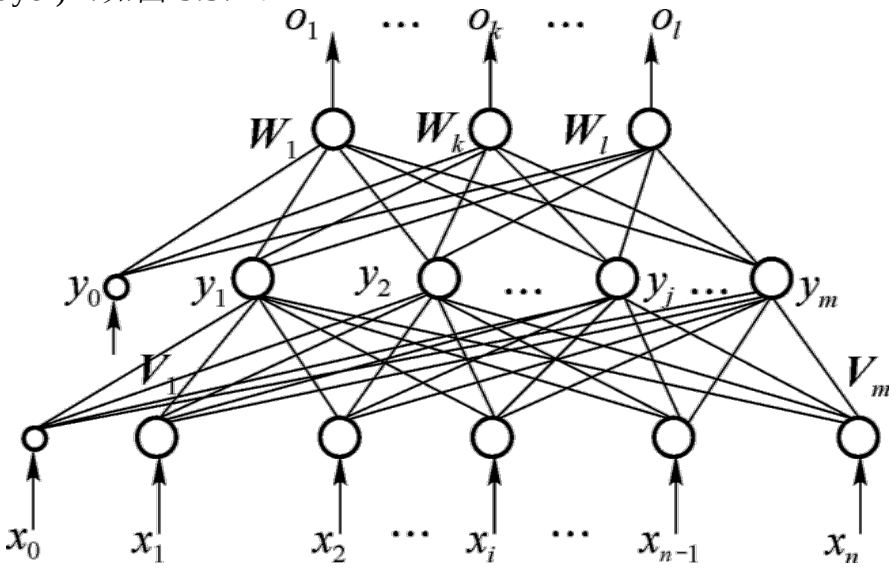


图 3.5 三层 BP 神经网络结构图

图 3.5 所示拓扑结构为单隐层 BP 神经网络网络的拓扑结构，一般称为三层前馈网或三层感知器，其三层为：输入层、中间层（也称隐层）和输出层。这种神经网络的特点是：各层神经元仅与相邻层神经元之间相互全连接，同层内神经元之间无连接，各层神经元之间无反馈连接，构成具有层次结构的

前馈型神经网络系统。Minsky 和 Papert 在颇具影响的《感知器》一书中指出，具有隐层的感知器神经网络能够求解非线性问题[29]，因此大部分普通非线性问题使用单隐层的神经网络就可以得到解决。

在人工神经网络发展历史中，很长一段时间里没有找神经网络各层之间连接权值的调整的有效算法。直到误差反向传播算法（即 BP 算法）的提出，成功地解决了求解非线性连续函数的多层前馈神经网络权值调整问题。

BP 算法的基本思想[29]是：学习过程由输入训练信号的正向传播与误差的反向传播两个过程组成。训练信号的正向传播就是将训练样本从输入层输入神经网络，然后经过神经网络内部输入层到隐层、各隐层之间、隐层到输出层的逐层运算，最终从输出层将运算结果输出。当输出层实际输出的结果与预先期望的输出不符时，就认为识别结果有误需要调整神经网络，这就需要进入误差的反向传播阶段。误差的反传是将输出结果和预期结果之间的误差通过输出层到输入层进行反向传播，经过神经网络内部输出层到隐层、各隐层之间、隐层到输入层的逐层运算，向着减小误差的方向调整各层间连接权值，最终将总误差分摊给各层的所有单元，调整所有层间连接权值达到神经网络的学习调整。在神经网络的训练过程中需要不断重复训练信号的正向传播与误差的反向传播调整过程，逐步调节各层连接权值，最终达到神经网络的输出误差减少到预设的可接受的程度为止。当神经网络训练到预先设定的学习次数任然没有将误差减少到预设的可接受的程度时也会停止训练。

3.3.4 BP 网络工作原理

设三层 BP 网络的输入向量为 $X=(x_1,x_2,\ldots x_i,\ldots x_n)^T$，$x_0=-1$ 为隐层输入阈值；隐层输出向量为 $Y=(y_1,y_2,\ldots y_j,\ldots y_m)^T$，$y_0=-1$ 为输出层阈值；输出层输出向量为 $O=(o_1,o_2,\ldots o_k,\ldots o_t)^T$；期望输出向量 $d=(d_1,d_2,\ldots d_k,\ldots d_t)^T$；输入层与隐层之间权值矩阵为 $V=(V_1,V_2,\ldots V_j\ldots V_m)$，其中 V_j 为隐层第 j 个神经元对应权向量；隐层到输出层权值矩阵为 $W=(W_1,W_2,\ldots W_k\ldots W_l)$，其中 W_k 为输出层第 j 个神经元对应权向量；输入层到隐层和隐层到输出层的变换函数 $f(x)$都为单极性 S 型函数。

BP 网络的权值调整过程如下：

网络输出与期望的输出误差为：

$$E=\frac{1}{2}(d-O)^2=\frac{1}{2}\sum_{k=1}^{l}(d_k-O_k)^2 \qquad \text{式(3-7)}$$

隐层到输出层误差调整公式为：

$$\Delta w_{jk}=\eta(d_k-O_k)O_k(1-O_k)y_j \qquad \text{式(3-8)}$$

其中常数$\eta\in(0,1)$，表示比例系数，在训练中反映了学习速率。

输入层到隐层误差调整公式为：

$$\Delta v_{ij} = \eta\left(\sum_{k=1}^{l} \delta_k^O w_{jk}\right) y_j (1 - y_j) x_i \quad \text{式(3-9)}$$

其中$\delta_k^O = (d_k - O_k) O_k (1 - O_k)$。

目前在实际应用中有两种权值调整方法：在标准 BP 算法中，每输入一个样本，都回传误差并调整权值，这种权值调整方法又称为单样本训练，由于单样本训练只针对每个样本产生的误差进行调整，难免顾此失彼，使整个训练的次数增加，导致收敛速度过慢；另一种方法是在所有样本输入之后，计算网络的总误差 $E_{总}$：

$$E_{总} = \frac{1}{2} \sum_{p=1}^{P} \sum_{k=1}^{l} (d_k^p - O_k^p)^2 \quad \text{式(3-10)}$$

然后根据总误差计算各层的误差信号并调整权值，这种累积误差的批处理方式称为批(Batch)训练或周期(Epoch)训练[29]。

3.3.5 BP 神经网络识别字符

用神经网络进行字符识别，主要有两种方法[30]：一种方法是先对待识别字符进行特征提取，然后用所获得的结构特征或者统计特征来训练神经网络分类器。这种方法会降低运算的复杂度，也会节省存储空间。但是需要首先提取字符特征，然后利用神经网络对待识别字符进行分类识别。这种方法提取特征需要靠人们的经验，其识别效果依赖于字符特征的提取，提取的特征不好会增加无用信息或得不到有用信息，会导致识别效果不理想。因此，字符特征的提取就成为研究的关键，需要谨慎提取特征。另一种方法[31]是充分利用神经网络的特点，直接把待处理图像输入网络，由网络自动完成特征的提取并进行字符的识别。这种网络待处理信息量大，网络庞大，但是这种方法无需特征提取，由神经网络自动提取特征并识别字符，抗干扰性能好，识别率高。但由于该方法网络结构比较复杂，输入模式维数的增加可能导致网络规模庞大，计算复杂。

1. 不提取特征值直接识别

首先，通过图像的预处理，包括图像的灰度花、二值化、字符定位、字符分割、字符图像平滑去噪、字符图像归一化等操作，得到大小统一的二值点阵图像，包括训练样本和待识别样本的字符点阵图像；然后，将训练样本字符点阵图像送入神经网络中进行训练，经过不断训练的神经网络最终达到能够正确将训练样本分类，这样得到的神经网络就可以进行字符的识别了。识别方法就是将待识别的字符图像送入神经网络，经过运算输出其所属类别。

2. 提取特征值后识别

首先，对训练样本图像进行预处理，图像的灰度花、二值化、字符定位、字符分割、字符图像平滑去噪、字符图像归一化等操作，得到大小统一的二值点阵

图像；然后，从字符点阵中提取特征值；接着，将提取的特征值送入神经网络中进行训练，经过不断训练的神经网络最终达到能够正确将训练样本分类，这样得到的神经网络就可以进行字符的识别了。识别时就是将待识别的字符图像进行同样处理和提取特征值，然后将特征值送入神经网络，经过运算输出其所属类别。

特征向量的选取同 3.2.2 节中一样，分别取其 16 个外围轮廓特征、8 个倾斜贯穿特征、8 个网格像素统计特征，共 32 维特征向量。

识别效果如下：

图 3.6 车牌识别结果

基于神经网络的车牌字符识别方法，对于解析度较高和图像比较清晰的车牌，这些方法能有效识别车牌中的字符，但对于较低解析度和较为模糊的车牌还有很多工作要做，因为这些方法只有在车牌中的每个字符被独立分割出来的前提下才能完成识别工作。而独立分割车牌取得字符，对较低解析度和较为模糊的车牌来说是非常困难的。虽然，神经网络识别速度较慢，不能满足实时性的要求，但其在识别效果上提高的余地较大，具有较强的容错能力，还可进一步训练学习，识别率较高，是广泛使用的一种方法。

3.4 本章小结

本章对字符的识别方法进行了介绍，详细介绍了模板匹配方法和神经网络方法的概念、原理及使用方法，并使用这两种方法进行字符的识别，包括不提取特征值将待识别字符图像与模板图像直接匹配的方法、提取特征值与模板图像进行匹配的方法、不提取特征值将图像直接送入 BP 神经网络进行识别的方法、提取特征值后将特征值送入 BP 神经网络进行识别的方法。

下一章将介绍字符识别与分布式计算的结合，将复杂的任务分别交由多个系统分布完成，以提高识别的效率。

第四章 字符识别与分布式计算

4.1 引言

前面我们以车牌识别为例对字符识别过程进行了介绍。车牌识别与其他文字识别技术类似，它与其他字符识别技术，例如书本中的印刷字符的识别、一张报纸的识别，其处理的差别之处就在于图片的处理、字符的分割上。当单个字符被分割提取并归一化之后，其识别的方法就殊途同归了。车牌字符是属于小字符集，其总的字符数不超过 100，但是一本书或一份报纸中的字符就远远不是车牌字符所能比拟的。单以汉字来说，清代《康熙字典》包含汉字 47000 多字；1990 年徐仲舒主编的《汉语大字典》，收字数为 54678 个；1994 年冷玉龙等的《中华字海》，收字数更是惊人，多达 85000 字。这只是汉字字符的类别，实际上字符的样式远不止于此，汉字有简体繁体之别，每个字符都有多种多样的字体，像是楷书、行书、草书、隶书等等，而每个人的书写风格也是不同的；而且世界上各国文字也是不同的，千变万化、不一而足。这样算起来，字符的个数、总类就以海量为计了，实际中要进行大字符集的字符识别所需要的对应模板就需要海量的模板了，其运算量巨大，传统单机字符识别系统就难以满足实时性的要求，因此需要改进字符识别系统，提升效率。

随着时代的发展，信息化进程的飞跃，人们对数据的海量存储和超级计算能力提出了更高的要求，这在过去几十年里促进了硬件的发展，使芯片集成度符合摩尔定律呈指数增长，但是硬件的发展受到了物理极限的约束[32]。由于晶体管电路已经逐渐接近其物理上的性能极限，摩尔定律在 2005 年左右开始失效，多核以及互联网时代到来，迫使软件编程方式不得不考虑并行问题，因此基于多核的多线程并发编程以及基于大规模计算机集群的分布式并行编程已成为提升计算性能的主要途径[33]。

字符识别始终是研究的热门，人们对字符识别的研究和改进始终未曾止步。评价一个字符识别的系统的性能有两方面的因素：系统的识别率和系统的效率。随着人们对字符识别的要求的提升，字符识别面临更加复杂的任务，包括针对海量识别任务的处理，包括针对多种多样字体的识别，包括针对各种不同国家的语言中巨量复杂字符的识别等等，这就对字符识别任务提出了更加艰巨的要求，对字符识别效率提出了更高需求。针对字符识别效率的改进，有多种多样的方法，包括对识别算法的优化、使用更好更专业的硬件系统、简化识别系统的冗余信息

量等等。还可以使用分布式计算的方法，将识别任务交给多个分布系统进行处理，从而可以利用大量廉价微机集群来完成海量数据的复杂运算，使得系统可以更高效的完成繁重的任务。

传统高性能计算中的并行编程模型抽象度不高，开发人员需要了解底层的配置和并行实现细节，需要了解并行编程中的种种复杂问题，如分布式存储，工作调度，负载平衡，容错处理，网络通信等，这些工作会占用程序员大量精力，降低开发效率。MapReduce 是 Google 实验室提出的一个分布式并行编程模型或框架，主要用来处理和产生大规模数据集，它已在 Google 内部得到了广泛的运用[33]，包括分布 grep、分布排序、web 连接图反转、每台机器的词矢量、web 访问日志分析、反向索引构建、文档聚类、机器学习、基于统计的机器翻译等。

2004 年 Dean 和 Ghemawat 第一次发表了这一新型分布式并行编程模型[34]，Hadoop 系统是 Apache 开源社区的项目，它用 Java 语言实现了 MapReduce 编程模型，这是一个开源实现平台。

本章将探讨在 Hadoop 平台下使用 MapReduce 编程模型与分布式字符识别系统的开发结合，分布式处理字符识别任务，提高字符识别系统的执行效率。

4.2 Hadoop 系统

Hadoop 是 Apache 开源组织下的一个分布式计算开源框架，它可以运行在大量普通廉价硬件设备构成的集群之上，实现对集群的控制和管理，方便快捷的处理海量数据。Hadoop 系统为应用程序提供了一组稳定可靠的接口，并屏蔽了底层并行应用开发的细节，可以使用户更加便捷地构建企业级的应用，并且能够实现海量数据的管理和分布式数据处理。用户可以在不了解分布式底层细节的情况下，开发分布式程序，充分利用集群的威力高速运算和存储。

Hadoop 是一个能够对大批量数据进行分布式处理的软件框架，它以一种可靠、高效、可伸缩的方式对大批量数据进行分布式处理。Hadoop 系统是一种可靠的系统，它预先假设计算元素和存储会失败，假设某些节点会失效，因此它维护多个数据的副本，确保当某一节点失效后能够迅速针对失败的节点重新分布处理，重新部署存储与运算，保证任务高效可靠完成。Hadoop 系统具有高效的特点，因为它是使用一个集群来完成任务，以并行的方式工作，将任务均衡分配给集群中的所有节点共同完成，通过并行处理加快处理速度。Hadoop 系统还具有扩展简单的特点，加入新的存储计算节点只需要修改配置文件就可以了。此外，Hadoop 使用了大批廉价机器构成的集群来完成任务，因此它的成本比较低，不需太多投入就可以使用。

Hadoop 系统最初是使用 Java 语言编写，天生支持 Java 语言程序，可以理想

稳定的运行在 Linux 平台上。目前 Hadoop 上的应用程序也可以使用其他语言编写，比如 C++。

Hadoop 最为核心的就是其分布式文件系统 HDFS(Hadoop Distributed File System)和 MapReduce 算法模型，它受到最先由 Google Lab 开发的 MapReduce 和 Google File System 的启发。

Hadoop 是一个广泛使用的分布式处理软件框架，它的广泛使用是由于它的以下优点[35]：

(1)开源性：Hadoop 是一个开源的软件平台，它是 Apache 根据 Google 提出的 GFS、MapReduce 等开发出来的开源项目，得到了 Yahoo、FaceBook 等大型公司及开源社区的支持，发展迅速。

(2)可扩展性：Hadoop 集群扩展简单便利，往集群中增加节点，只需简单修改配置文件，不需要改变现有 Hadoop 集群的结构。

(3)可靠性：Hadoop 假设计算元素和存储会失败，因此它维护多个工作任务以及数据的副本，确保当集群中某个节点失败时，能够针对失败的节点重新分布处理，重新部署存储与运算。

(4)高效性：Hadoop 通过分布式文件系统以及 MapReduce 框架，使得集群中的各节点能够并行的处理任务，从而大大提升了整体系统执行任务的效率。

(5)廉价性：Hadoop 集群可以部署在大量普通廉价机器的集群上，对硬件没有过高的特殊要求，因此构建 Hadoop 集群可以利用大量现有廉价集群，节省资源。

(6)跨平台性：Hadoop 是基于 Java 开发的一个开源软件项目，它可以跨平台的运行在多种操作系统和机器上。

4.2.1 HDFS

分布式文件系统 HDFS 是专门为 MapReduce 作业所设计的文件系统，是谷歌 GFS 的开源实现。GFS 即 Google File System，是 Google 公司为了存储海量搜索数据而设计的专用文件系统，是一个可扩展的分布式文件系统，它运行于廉价的普通硬件组成的集群上，可以用于处理大型、分布式的大量访问数据的应用，它可以给大量的用户提供总体性能较高的服务。HDFS 是 GFS 的开源实现，它并不是用来处理随机存取数据的。HDFS 的设计的初衷更多的考虑到了数据批处理，而不是用户交互处理，比之数据访问的低延迟问题，更关键的在于数据访问的高吞吐量。HDFS 是一个给应用提供高吞吐量的分布式文件系统，可能由成百上千的机器所构成，每个机器上存储着文件系统的部分数据。

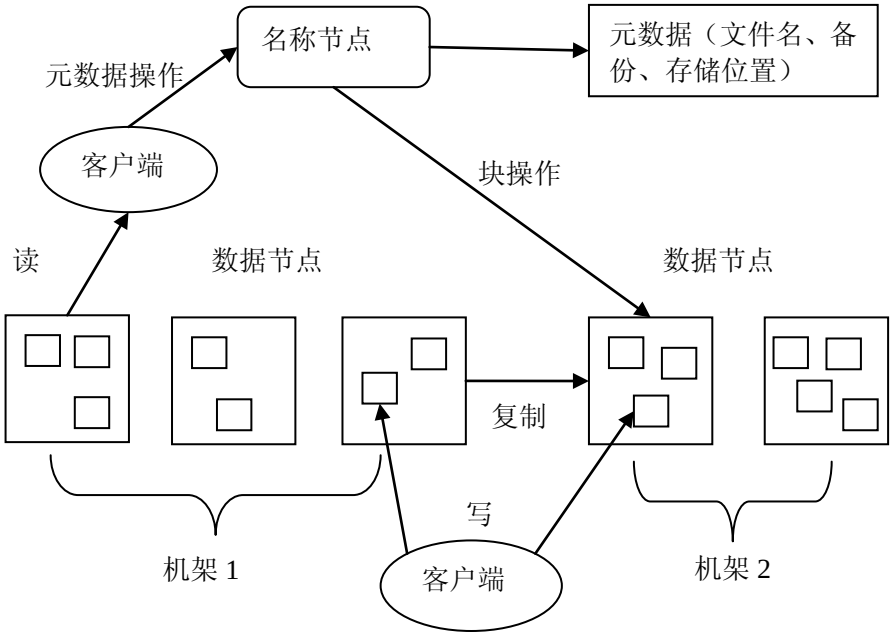


图 4.1HDFS 框架图

HDFS 默认的最基本的存储单位是 64M 大小的数据块，HDFS 中的文件默认是被分成 64M 一块的数据块存储的，不同于普通文件系统的是，在 HDFS 中如果一个文件小于一个数据块的大小，并不占用整个数据块存储空间。

HDFS 是一个主从结构的体系，一个 HDFS 集群是由一个名称节点(Namenode)和多个数据节点(Datanode)构成。名称节点是一个管理文件的命名空间和调节客户端访问文件的主服务器，它将所有的文件和文件夹的元数据保存在一个文件系统树中， 这些信息也会在硬盘上保存成以下文件：命名空间镜像(namespace image)及修改日志(edit log)，还保存了一个文件包括哪些数据块，分布在哪些数据节点上，这些信息并不存储在硬盘上，而是在系统启动的时候从数据节点收集而成的。数据节点是文件系统中真正存储数据的地方，客户端(client)或者名称节点可以向数据节点请求写入或者读出数据块，数据节点周期性的向名称节点回报其存储的数据块信息。在一个 PC 集群上部署 HDFS，一般都是使用一台单独的 PC 作为名称节点，集群剩下的 PC 各自作为一个数据节点。在硬件资源有限的情况下，有时也会让一台 PC 运行多个数据节点。

存储在 HDFS 中的文件被切分成块，然后系统将这些块复制到多个数据节点中，数据块的大小和块的复制数量在创建文件时由客户机决定，用户可以自行进行配置。名称节点可以控制所有文件操作，它可以对命名空间中的文件和目录进行操作，可以打开、关闭、重命名 HDFS 中的文件目录等等。数据节点是实际存储文件的节点，它负责处理来自名称节点或者客户端对 HDFS 进行的文件操作。数据节点还可以对块进行创建、删除等命令，还可以进行数据块的复杂操作等。名称节点和数据节点都是以软件的形式运行的，可以运行在普通的机器之上。每个数据节点都运行在一台单独的机器上；而名称节点可以单独运行在独立的机器

上，也可以和一个数据节点共同运行在一台机器上。Hadoop 集群的机器一般使用的都是 linux 操作系统；HDFS 最初是用 java 来写的，可以完美的支持 java 程序的运行，任何支持 java 的机器都可以很好的作为名字节点或数据节点来运行。由于 java 语言具有轻便的特点，因此很容易将 HDFS 部署到大范围的机器上。最典型的集群部署方法是用一个专门的独立机器来作为名称节点运行，集群中的其他机器运行数据节点实例。集群中只有一个名称节点极大地简单化了系统的体系。系统设计成用户的实际数据不经过名称节点。HDFS 内部的所有通信都基于标准的 TCP/IP 协议。

HDFS 有着高容错性的特点，它被设计用来部署在大规模低廉的硬件上，而且它能够提供高传输率来访问应用程序的数据，适合那些有着超大数据集的应用程序。

4.2.2 MapReduce[36]

Hadoop 系统中的 MapReduce 框架是一种编程模型，是 Google 实验室提出的一个分布式并行编程模型或框架，它是 Google 三大核心技术的一项重要技术，用于大规模数据集的并行运算。并行计算是海量数据的计算处理中的一种重要方法，目前，并行计算对很多编程人员来说还是一个比较难以实现的问题。传统高性能计算中的并行编程模型抽象度不高，开发人员需要了解底层的配置和并行实现细节，需要了解并行编程中的种种复杂问题，如分布式存储，工作调度，负载平衡，容错处理，网络通信等，这些工作会占用程序员大量精力，降低开发效率。MapReduce 是 Google 实验室提出的一个分布式并行编程模型或框架，其概念 Map 和 Reduce 是该模型中的两大基本操作，他们的主要思想，是从函数式编程语言里借来的，还有从矢量编程语言里借来的特性。他极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式系统上。

MapReduce 是被设计用来处理海量数据，是一种简化的并行计算编程模型，可以利用它高效地开发并行计算应用程序。MapReduce 的设计要求把计算任务分配到集群中的若干节点，所以节点之间不可以随意的对数据进行共享，因为同步节点之间数据的通信开销会降低集群的效率和可靠性。

其工作原理图如下：

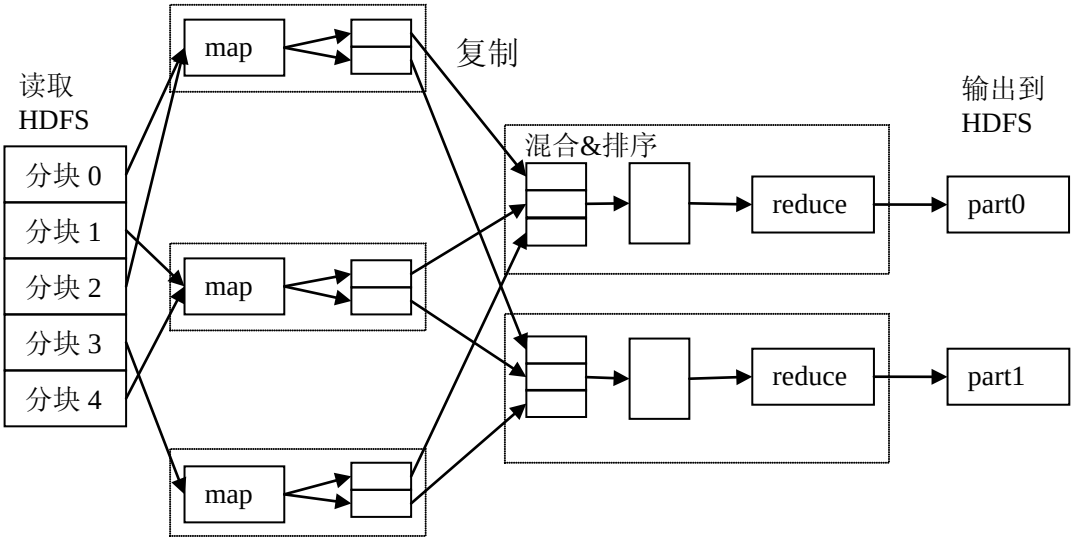


图 4.2 MapReduce 工作原理

从用户的角度讲，MapReduce 将计算过程分为两个最基本的阶段：Map 和 Reduce，每个阶段的输入都是一系列的键值对(key/value)，每个阶段的输出也是一系列的键值对。Hadoop Map/Reduce 是 Google MapReduce 的开源实现，通过构建计算机集群将用户需要执行的 Map/Reduce 任务分布到集群中的各个节点计算机上并行执行，实际上 Master 并没有把数据传输到各个 Map 和 Reduce 节点上，而是在每次调度中告知 Map 节点和 Reduce 节点要处理的数据分布所在的节点位置，将计算移动到数据上，而移动计算往往比移动数据的效率高，这样就能降低网络流量减少网络拥塞，提高系统效率。

Map 过程中将输入数据划分成大量的数据片段并给每一个数据片段分配不同的 Map 任务。每一个 Map 节点通过 map 函数由对输入数据片段的键值对进行计算并生成中间结果，这个中间结果也是以键值对的形式来表示的。Map 过程的操作是由用户自己定义的。用户编写相应的 Map 映射函数，由 Map 过程接收的键值对(k1,v1)经过处理后得到中间结果输出。中间结果表示为键值对(k2,v2)。MapReduce 系统对所有 Map 过程输出的中间键值对进行分组归并，将结果以键值对(k2,list(v2))的形式传送到 Reduce 方法中进行 Reduce 操作，其中 list(v2)是所有键为 k2 的值的集合：

Map: (k1,v1)→list(k2,v2)

Reduce 操作是对 Map 传来的数据集进行归并，由归约函数指定归并的规则，将中间结果集合中具有相同关键字数据聚集在一起：

Reduce: (k2,list(v2))→list(k3,v3)

Reduce 过程的输入是 Map 阶段输出的键值对 (k2,list(v2))。这些键值对经过用户编写的 Reduce 代码处理后，将进行合并等操作形成一个更小的值的集合[34]。

等所有的 reduce 节点都完成以后，Master 节点会收集所有 reduce 节点返回的

结果。根据结果判断是进行下一轮的map操作或是reduce操作还是返回最终结果。

MapReduce 的处理过程主要涉及以下四个部分：1.客户端 Client：用于提交 Map-reduce 任务 job；2. JobTracker：协调整个 job 的运行，是一个 Java 进程，其 main class 为 JobTracker；3.TaskTracker：运行此 job 的 task，处理 input split，其为一个 Java 进程，其 main class 为 TaskTracker；4.HDFS：hadoop 分布式文件系统，用于在各个进程间共享 Job 相关的文件。

MapReduce 执行任务过程如下[37]：

1.任务提交

使用 JobClient.runJob()创建一个新的 JobClient 实例，调用其 submitJob()函数提交任务，之后 runJob 每隔一秒钟轮询一次 job 的进度，将进度返回，直到任务运行完毕。

2.任务初始化

当 JobTracker 收到 submitJob 调用的时候，将此任务放到一个队列中，job 调度器将从队列中获取任务并初始化任务。

3.任务分配

TaskTracker 周期性的向 JobTracker 发送 heartbeat 心跳信息。在 heartbeat 中，TaskTracker 告知 JobTracker 其已经准备运行一个新的 task，JobTracker 将给其分配一个 task。在 JobTracker 为 TaskTracker 选择一个 task 之前，JobTracker 必须首先在最高优先级的 Job 中选择一个 task。

4.任务执行

TaskTracker 被分配了一个 task 之后便要运行此 task。

首先，TaskTracker 将此 job 的 jar 从共享文件系统中拷贝到 TaskTracker 的文件系统中。TaskTracker 从分布式缓存中将 job 运行所需要的文件拷贝到本地磁盘。然后为每个 task 创建一个本地的工作目录并将 jar 解压缩到文件目录中。最后创建一个 TaskRunner 来运行 task。TaskRunner 创建一个新的 JVM 来运行 task。被创建的 child JVM 和 TaskTracker 通信来报告运行进度。

(1)Map 过程

MapRunnable 从输入分块中读取一个个的记录，然后依次调用 map 函数，将结果输出到缓存中。当缓存中数据的到达一定的大小，将数据开始写入硬盘。在写入硬盘之前，内存中的数据通过 partitioner 分成多个 partition。在同一个 partition 中，背景线程会将数据按照 key 在内存中排序。每次从内存向硬盘存入数据，都生成一个新的 spill 文件。当此 task 结束之前，所有的 spill 文件被合并为一个整的被 partition 的而且排好序的文件。reducer 可以通过 http 协议请求 map 的输出文件。

(2)Reduce 过程

当 map task 结束后，通知 TaskTracker，TaskTracker 通知 JobTracker。对于一

个 job，JobTracker 知道 TaskTracer 和 map 输出的对应关系。reducer 中一个线程周期性的向 JobTracker 请求 map 输出的位置，直到其取得了所有的 map 输出。reduce task 需要其对应的 partition 的所有的 map 输出。reduce task 中的 copy 过程即当每个 map task 结束的时候就开始拷贝输出，因为不同的 map task 完成时间不同。reduce task 中有多个 copy 线程，可以并行拷贝 map 输出。当很多 map 输出拷贝到 reduce task 后，一个背景线程将其合并为一个大的排好序的文件。当所有的 map 输出都拷贝到 reduce task 后，进入 sort 过程，将所有的 map 输出合并为大的排好序的文件。最后进入 reduce 过程，调用 reducer 的 reduce 函数，处理排好序的输出的每个 key，最后的结果写入 HDFS。

5.任务结束

当 JobTracker 获得最后一个 task 的运行成功的报告后，将 job 的状态改为成功。当 JobClient 从 JobTracker 轮询的时候，发现此 job 已经成功结束，则向用户打印消息，从 runJob 函数中返回。

4.3 分布式计算与字符识别

当处理数据量足够大时，使用传统的单处理器的单机系统就难以保证系统实时性的要求，而使用分布式系统进行处理就可以将海量数据处理任务交由多个系统共同完成，可以显著提高处理效率。将分布式计算与字符识别系统进行结合，是提升系统处理大批量任务的效率的一种手段。

前面章节分别介绍了字符识别的流程和用于开发分布式程序的 Hadoop 平台。下面将介绍将 Hadoop 与字符识别系统结合，使用 Hadoop 系统进行字符识别的方法。

4.3.1 分布式识别算法设计

使用 Hadoop 平台开发分布式程序可以将负责的计算任务分布给集群中的各个节点并行计算，可以减少机器负担，提高运算速度，提升系统整体性能。字符识别系统使用 MapReduce 编程模型进行实现，关键就在于将任务进行分解，划分为各个分布式任务进行分布式运算。

这里我们将整体系统框架划分为两部分：前端客户端和后台运算系统两部分，其系统结构图如下：

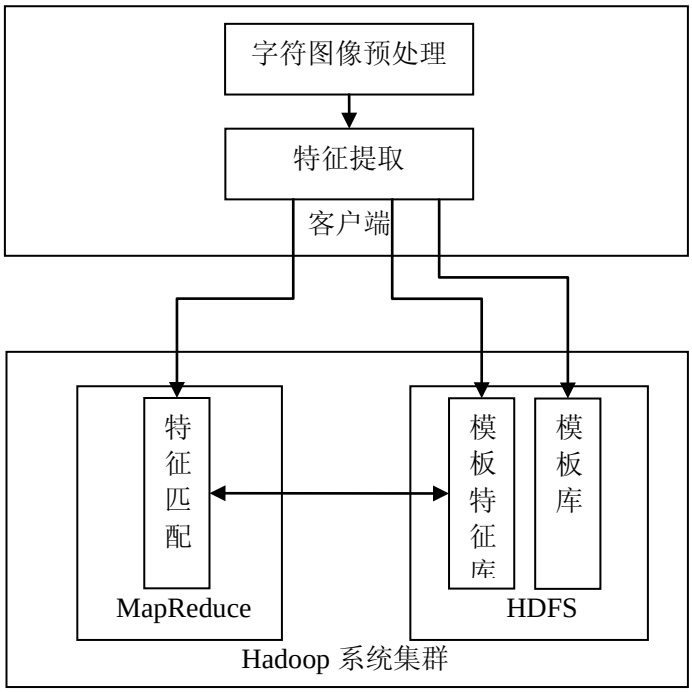


图 4.3 字符识别系统结构图

其中，在客户端中负责人机交互，供用户使用系统，提供系统功能及后台调用。在客户端中还将进行字符图像的预处理以及特征的提取。首先进行字符图像的灰度化、二值化、定位、字符分割等操作，然后提取字符图像的特征向量，供后台系统识别调用。

客户端提取字符图像后，将特征值输入 MapReduce 系统中，然后从 HDFS 中读取模板特征值，进行特征匹配，从中选出匹配最近似的结果，输出识别结果。

其中后台识别系统包括 Map 过程和 Reduce 过程，Map 节点上存储所有模板的信息，使用客户端将待识别字符图像进行预处理、提取其特征值，然后送入 MapReduce 进行匹配运算，最终由匹配结果得到字符识别的分类结果：

1. Map 过程

Map过程负责读入由客户端发来的字符特征数据与Map节点上保存的所有模板进行匹配运算，得到待识别字符与模版的欧氏距离值作为中间数据，将中间数据发送到输出结果收集器中。

2. Reduce 过程

将输入图像在所有 Map 节点上运算得到的结果键值对复制到 Reduce 节点上，对所有键相同的键值对进行合并，得到最为接近的分类，输出图像与图像所属分类信息的键值对。

整个系统 Job 流程如下：

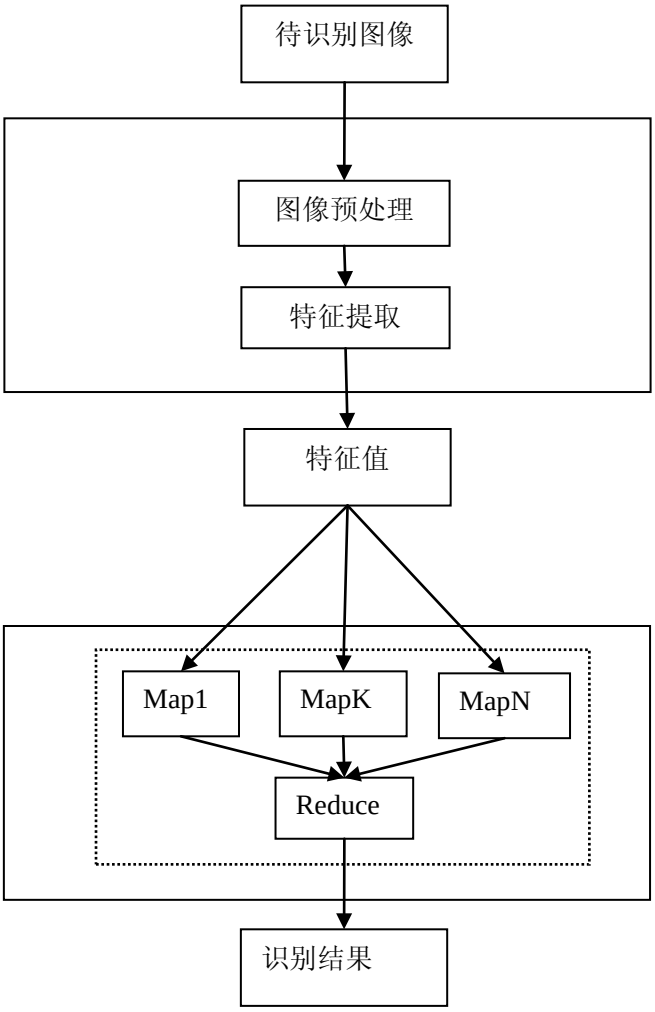


图 4.4MapReduce 作业执行流程

在系统中还可以使用 Combine 函数对流程进行优化。集群上的可用带宽限制了 MapReduce 作业的数量，因此最重要的一点就是尽量避免 map 任务和 reduce 任务之间的数据传输[36]；Hadoop 允许用户针对 map 任务的输出指定一个合并函数(combiner)，合并函数的输出作为 reduce 函数的输入。Combine 函数是在每个数据节点的 Map 任务执行完毕后执行的，只在本地环境执行，而与集群中其它节点没有无关，它与 Reduce 的不同之处在于，Combine 函数的结果输出为中间键值对作为 Reduce 的输入。

4.3.2 系统详细设计

系统客户端界面如下：

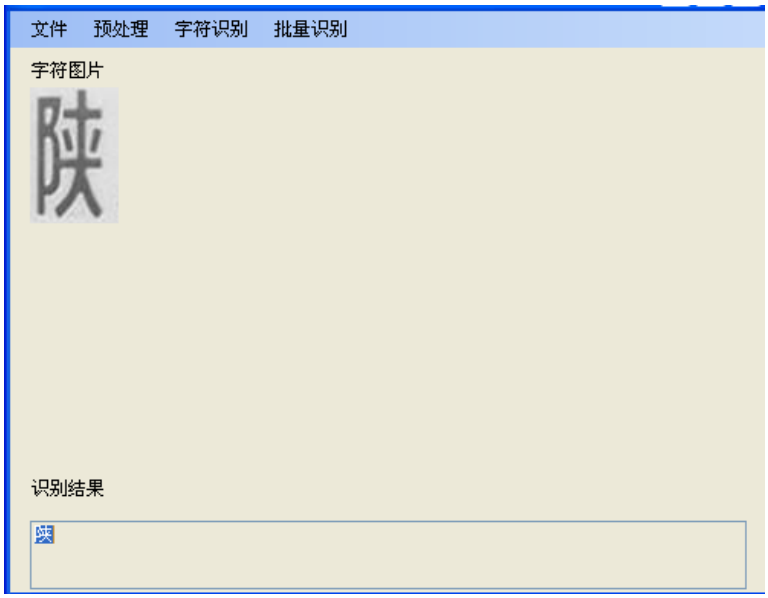


图 4.5 程序客户端界面

在客户端中完成字符图像的预处理以及特征向量的提取。

其中字符图像的预处理过程使用第二章中介绍的方法进行处理，其中图像灰度化使用第 2.2.1 节中的公式(2-6)中的加权平均法计算灰度值并将图像灰度化，每一像素灰度值是其 RGB 三分量的加权和。

图像的二值化使用第 2.2.5 节中的迭代法二值化图像，其算法流程如下：

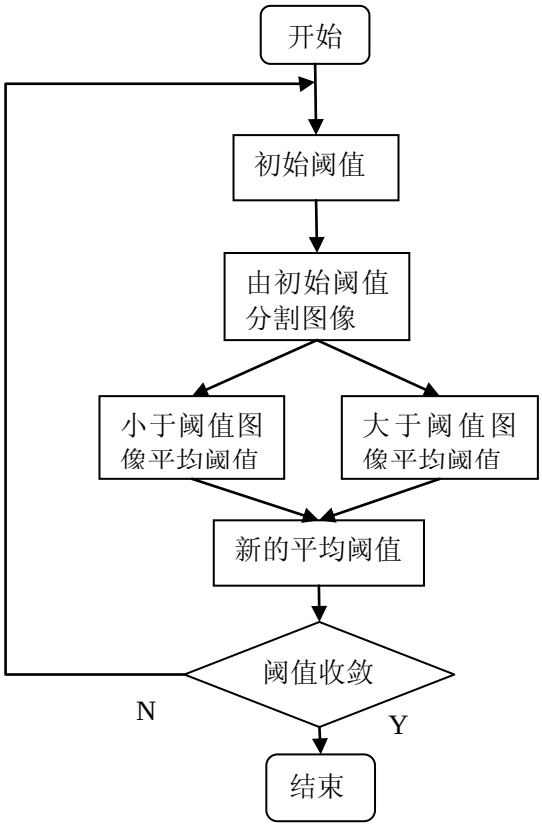


图 4.6 迭代二值化算法流程

对分割后的单个字符图像进行归一化，转化为大小位置统一的字符点阵图像；然后提取特征值，以 3.2.2 节中介绍的特征提取方法提取字符点阵的 32 维特征向量。

识别的过程是将待识别字符特征向量与所有模版进行匹配，计算其欧氏距离，找到距离最近的作为识别结果。模板也是使用同样方式提取的 32 维特征向量，存储在 HDFS 中。

4.3.3 关键代码

Map 过程伪代码：

```
Map()
{
    string picno=图像编号;
    int pic[32]=图像 32 维特征向量;
    int modnum=模板数;
    for (i=0;i<modnum;i++)
    {
        string modno=模板编号;
        int mod[32]=模板 32 维特征向量;
        double sum=0;
        for(int j=0;j<32;j++)
        {
            sum+=(pic[j]-mod[j])*(pic[j]-mod[j]);
        }
        double value=sqrt(sum);
        string key=picno+modno;
        输出键值对<key, value>;
    }
}
```

Reduce 过程伪代码：

```
Reduce()
{
    string key=Map 过程输出中间数据 key 值;
    double value=Map 过程输出中间数据 value 值;
    找出 value 最小的<key,value>对;
```

string key1=从 key 中分离出来的图像编号；
string value1=从 key 中分离出来的模板编号；
输出键值对<key1,value1>;
}

4.4 实验与分析

4.4.1 环境搭配[38][39]

本实验中使用的是 Hadoop 0.20.2 版本系统，使用的系统是 3 台虚拟器，使用的操作系统是 ubuntu 11.10，机器配置如下：

表 4-1 机器配置表

Hadoop 版本	Hadoop 0.20.2
Java 版本	JDK1.6
操作系统	Ubuntu 11.10
开发工具	Eclipse

Hadoop 集群包括一个 Namenode 节点和两个 Datanode 节点，其中 Namenode 节点也作为一个 Datanode 节点使用。两台机器配置信息如下：

表 4-2 Hadoop 集群节点信息

节点类型	IP	主机名
Namenode/Datanode	192.168.1.10	master
Datanode	192.168.1.11	slave1
Datanode	192.168.1.12	slave2

从 HDFS 的角度上来看，机器节点分为名称节点和数据节点，其中名称节点只有一个，而数据节点可以有若干个。从 MapReduce 编程模型的角度看，机器节点分为作业服务器和任务服务器，其中作业服务器只有一个，而任务服务器可以有若干个。本实验点和任务服务器中我们将 master 主机部署为名称节点和作业服务器，而 master，slave1，slave2 主机部署为数据节点和任务服务器。

集群中的每台机器的 IP 地址和主机名之间必须能正确解析。如果不能正确解析，则要修改相应的/etc/hosts 文件。对于名称节点机器，要将名称节点以及数据节点的 IP 地址和主机名添加到 hosts 文件中，对于数据节点机器，则将名称节点以及本机的 IP 地址和主机名添加到 hosts 文件中。

表 4-3 Hadoop 节点配置

HDFS	master	NameNode,DataNode
	slave1	DataNode
	slave2	DataNode
MapReduce	master	JobTracker ,TaskTracker
	slave2	TaskTracker
	slave2	TaskTracker

在 Hadoop 中名称节点通过 SSH 来启动和停止所有节点上的守护进程，各个节点使用不需要输入密码的方式对其它节点进行访问，因此需要配置 SSH 使用无密码公钥认证。以名称节点为例，首先执行 ssh-keygen –t rsa 命令，生成 id_rsa 和 id_rsa.pub 的密钥对，然后将 id_rsa.pub 的内容拷贝到本机的 /home/.ssh/authorized_keys 文件中，如果文件 authorized_keys 已经存在，则将 id_rsa.pub 的内容添加到文件结尾。接着把 authorized_keys 文件拷贝到另外两个数据节点中，这样名称节点就能通过无密码输入的 SSH 方式访问另外两个数据节点了。数据节点的配置过程也一样。

Hadoop 必须安装在所有机器的相同目录路径下，本实验中我们将 Hadoop 安装在 home 路径下。因为整个集群内的机器的环境完全一样，因此，实际操作中，我们先在名称节点下安装配置 Hadoop，然后将整个 Hadoop 文件夹复制到另外两个数据节点中的相同位置。

4.4.2 实验与分析

下面使用字符识别系统进行大量字符识别，由其结果来分析系统性能。

首先对其识别率进行分析对比。识别率就是系统识别结果正确数所占总识别字符的比率，分别以 1000、2000、3000 的识别量进行识别率的比较，其对比分析图如下所示：

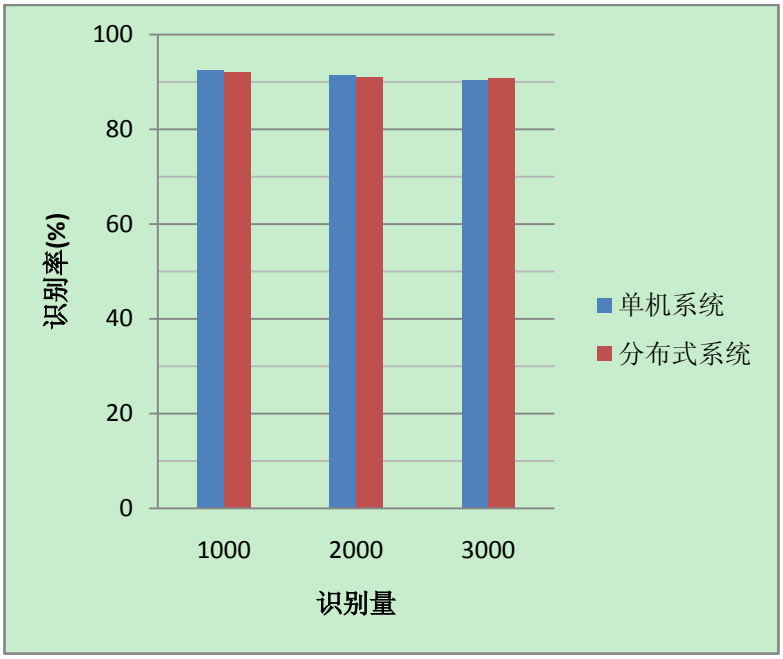


图 4.7 识别效果对比图

由上图可知单机字符识别系统识别率与分布式字符识别系统识别率波动不大，这是因为分布式系统只是将任务分配给各个节点分别执行最后结果汇总输出，其识别率由算法本身决定，与是否是分布式处理关系不大。系统的识别率并不算太高，这是由于模板数量不足、特征向量选取因素以及图像处理因素等等影响所致。一般拥有越多的模板就会使得系统的识别效率越高。

使用 Hadoop 系统进行字符识别，其目的主要提升字符识别系统执行任务的效率，下面使用了大批量的样本数据输入识别系统进行字符的识别，分析其对任务执行的效率，运行时间曲线图如下所示：

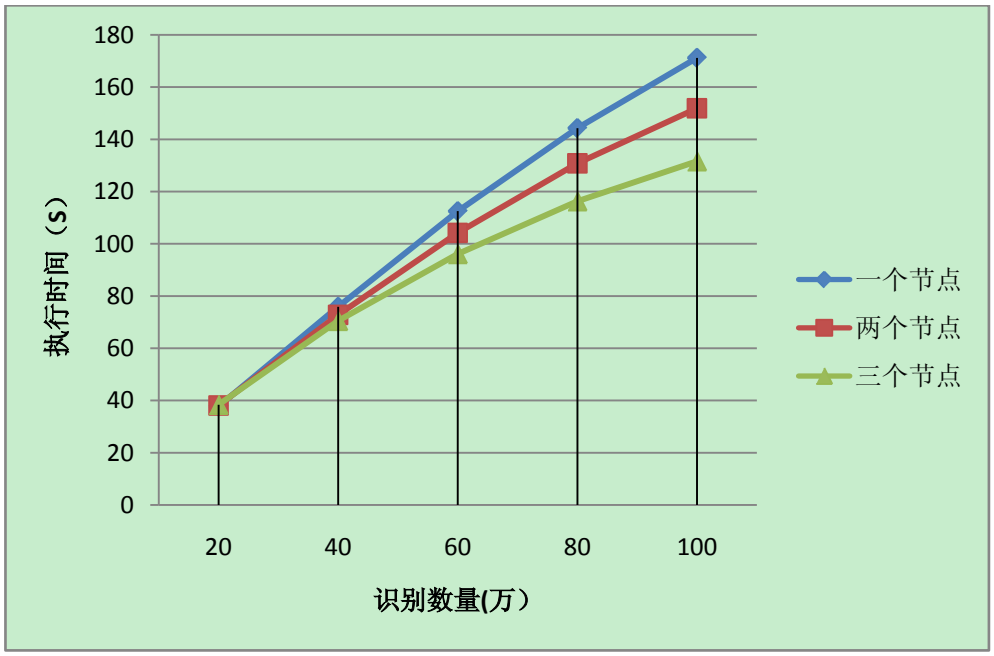


图 4.8 字符识别效率比较

由上图实验结果可以分析得到：

在待识别数据量较小时，使用 Hadoop 集群解决任务，其效率并没有多少提升，甚至可能效率还不如普通的单节点运算，这是因为分布式运算有节点间的通信、负载的均衡控制等等操作，当处理数据量不多时显示不出来分布式系统的优势，反而会多耗费一定的时间；但是随着识别任务量的增加，Hadoop 集群的分布式运算的优点就会逐步体现，其识别效率将会逐步由于单节点的识别系统。当识别量较小时，集群中的节点数目的增加对系统效率没什么影响；但是当识别量达到一定程度时，随着集群中节点数的增加，系统的执行效率也会增加。

因此，在处理大批量字符识别任务时，使用 Hadoop 集群的分布式识别系统将会拥有更高的效率。同时 Hadoop 系统的安全性也由其系统内部机制进行保证，其识别的安全性受到分布式运算节点故障、网络故障等的影响不大，这些是 Hadoop 内部封装好的，不需要我们考虑其实现细节。所以使用 Hadoop 分布式系统是一种提升字符识别效率的好的方案。

4.5 本章小结

本章在前两章的基础上，研究了字符识别与分布式运算的结合，探讨了在 Hadoop 系统性实现分布式字符识别的方法。首先介绍了 MapReduce 编程模型和 Hadoop 系统的相关知识，然后介绍了使用 Hadoop 系统进行字符识别的过程。最后结合实验结果对其性能进行了分析对比。

第五章 总结与展望

5.1 本文总结

本文首先介绍了字符识别的概念、历史和研究现状，然后介绍了 Hadoop 系统的概念与研究现状。然后本文对 Hadoop 中的 HDFS 和 MapReduce 两个重要方面进行了介绍，包括 HDFS 的组成和框架、MapReduce 的概念、组成及框架等。

然后本文对字符识别进行了深入的研究探讨。

字符识别是模式识别的一个分支，是当今广泛研究及使用的技术，在很多场合下有着成熟的应用，例如车牌识别、文本 OCR、邮件分拣等等。对字符识别进行研究有很大的理论意义与实践意义，因此本文对字符识别系统进行了研究并试图提升字符识别系统的效率。

本文先以面向小字符集的车牌识别为例介绍字符识别的一般流程和方法，对字符图像的前处理过程的一般步骤进行了详细介绍，包括图片的灰度化、二值化、字符分割、字符的特征提取等的详细实现方法。

然后本文介绍了字符分类识别的方法。字符分类识别的方法多种多样，研究的对象也不限于小字符集的车牌字符集，可以应用于所有字符集。这里主要介绍了字符识别中常用的两种方法：模板匹配法和神经网络法识别字符，这是字符识别中最经常使用的两种方法。

随着社会的发展、信息化的加速、识别任务的扩展，传统的单机运行任务的效率已经不能满足用户实时性的要求，基于 Hadoop 的分布式字符识别系统就是提升任务执行效率的一个解决方案。

本文首先对 Hadoop 系统进行了介绍，详细分析了 HDFS 的组成、框架、使用及其优缺点等，然后本文对 MapReduce 编程框架进行了深入研究，基于字符识别的研究和 Hadoop 的研究，本文提出了基于 Hadoop 的字符识别系统的设计方案与实现，最后经过实验测试和结果的分析，本文分析了 Hadoop 下分布式处理方案对字符识别系统执行任务的效率的提升。

5.2 进一步的工作

本文对基于 Hadoop 的字符识别系统进行了一些研究，分析论证了 Hadoop 平台下分布式程序效率的提升，但是本文的研究还有很多不足之处，需要后续深入

研究改进：

首先，本文研究的重点只在于字符识别系统效率上的提升，没有深入研究字符识别的正确率提升的方法，因此需要进一步完善字符识别系统的研究，从字符识别系统的效率、识别率等多方面进行进一步的深入研究。

其次，本文的研究的过程中简化了很多流程，使用了车牌识别系统对字符识别的一般流程进行了探讨研究，但这只是针对字符识别中的一种应用，面向的是小字符集，因此下一步需要进一步扩展研究的范围，使得系统具有普适性，能适应更大范围的任务与需求。

最后，字符识别系统应该包含一个完整的流程，不但包括其预处理、识别等过程，还需要考虑其后续处理过程，根据用户反馈提升系统性能等，因此还需要进一步的研究，修正完善整个系统。

致谢

本文的研究工作是在我的导师的精心指导和悉心关怀下完成的。无论是论文的立题，还是资料的搜寻，论文工作的研究，导师都给了我以极大的关怀和指导，并提出了宝贵的意见和建议。导师在我的学业和论文的研究工作中给予了无私的指导和帮助，使我深受的启迪，他的严谨的学风给予了我极大的指引。同时要感谢我的同学和朋友们。他们在我的学业和生活中给与了热心帮助和无私鼓励。在此我要向我的导师以及我的同学、朋友致以最诚挚的感谢和深深的敬意。

在日常生活和学习中，我的家人们也给予了我很大帮助。他们的默默支持和鼓励，是我行动的最大助力。在此，向所有关心和帮助过我的老师、同学、家人和朋友表示由衷的谢意！

最后真诚地感谢参与评阅论文工作和参加答辩的各位老师、教授！

参考文献

[1] 胡家忠.计算机文字识别技术.气象出版社.1994 年

[2] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. Http://hadoop.apache.org/core/docs/current/hdfs_design.html.2011.10

[3] Konstantin Shvachko，Hairong Kuang，Sanjay Radia．The Hadoop Distributed File System[J]．IEEE.2010,978-1-4244-7153-9(10)

[4] Jeffrey Dean, Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM,2008,51(1):107-113

[5] 沙建辉.无处不在的 OCR.中国计算机用户.2004 年第 23 期

[6] 张旋. OCR 技术研究进展及前瞻.中国科技纵横.2010 年第 8 期

[7] 百度百科. OCR. http://baike.baidu.com/view/17761.htm

[8] 孟侃良.认清 OCR.中国计算机报.1998 年第 69 期

[9] M.E.Stevens. Introduction to The Special Issue on Optical Character Recognition(OCR). Pattern Recognition 2,147-150(1970)

[10] 周奇.基于支持向量机的脱机手写字符识别研究.重庆大学硕士学位论文.2007 年 4 月

[11] 洪钊峰.Hadoop 发展现状与 Hadoop in China 大会. http://cloud.it168.com/a2010/0812/1089/000001089612.shtml.2010 年

[12] 文颖.数字、字符识别及其应用研究.上海交通大学博士论文.2009 年

[13] 邱康敏.电子资源分布式存储子系统的设计与实现.北京大学硕士学位论文.2009 年

[14] 尹芳.印刷体英文字符识别系统. 哈尔滨工业大学硕士学位论文. 2004 年

[15] 边肇祺.模式识别（第二版）. 清华大学出版社.2000 年 1 月

[16] 周天弋.基于计算机视觉的车辆与前方车距检测.浙江工商大学学位论文.2010 年

[17] R.O.Duda, P.E.Hart. Pattern Classification and Scene Analysis. John Wiley & Sons Inc. 1973.6

[18] 闫娟.数字图像的平滑处理方法研究.软件导刊.2009 年第 8 卷第 1 期

[19] 梁一江.图像平滑处理方法初探及简单的算法介绍.才智 2009 年 4 月

[20] 金献珍,吴艳.MATLAB 实现数字图像锐化处理.商场现代化.2008 年

36 期

[21] 闫青.常用车牌定位算法浅析.微型机与应用.2010 年 02 期

[22] 闫青.车牌识别系统中车牌定位算法的研究. 山东大学学位论文.2010 年

[23] 吕俊哲.图像二值化算法研究及其实现.科技情报开发与经济.2004 年第 12 期

[24] Trier OD, Taxt T, Jain AK. Feature extraction methods for character recognition-A survey. Pattern Recognition.1996,29(4)

[25] 洪必海.车牌识别算法的研究与实践. 厦门大学学位论文.2007 年

[26] 杨淑莹.图像模式识别—VC++技术实现.清华大学出版社.2005 年

[27] 徐海兰,刘彦婷,杨磊.模式识别中三种字符识别的方法. 北京广播学院学报.2005 年 12 月第 12 卷第四期

[28] Rumelhart, McCellland. Parallel Distributed Processing. Vol.1,2, MIT Press, Cambridge, MA. 1986

[29] 韩力群.人工神经网络教程. 北京邮电大学出版社.2006 年 12 月

[30] 刘静.几种车牌字符识别算法的比较. 电脑与电信.2008 年第 8 期

[31] 李振恒.基于各向异性扩散方程的图像降噪技术和车牌识别技术研究. 山东大学学位论文.2010 年

[32] 李响.基于 Hadoop 的云计算基础架构分析.计算机时代.2011 年 11 期

[33] 李成华，张新访，金海，向文.MapReduce: 新型的分布式并行计算编程模型.计算机工程与科学.2011 年 33 卷 3 期

[34] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 2005, 51(1) : 107- 113.

[35] 王贤伟.基于 Hadoop 的外观专利图像检索系统的研究与实现.广东工业大学硕士论文.2011 年 6 月

[36] Tom White. Hadoop 权威指南(第 2 版).2011 年 7 月

[37] Map-Reduce 入门. http://www.cnblogs.com/forfuture1978/archive/2010/11/14/1877086.html.

[38] Running Hadoop On Ubuntu Linux (Single-Node Cluster). http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

[39] Running Hadoop On Ubuntu Linux (Multi-Node Cluster). http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/