哈 尔 滨 理 工 大 学

毕 业 设 计

题　　目：基于 opencv 的即时翻译软件

院、　系：计算机科学与技术学院 计算机系

姓　　名：蒋烻吉

指导教师：高尚民

系 主 任：林克正

2018 年 6 月 20 日

哈尔滨理工大学毕业设计（论文）评语

学生姓名：蒋烻吉　　学号：1404011005
学　　院：计算机科学与技术　　专业：计算机科学与技术
任务起止时间：2018　年 3 月 1 日至　2018　年　6　月　20　日
毕业设计（论文）题目：基于 opencv 的即时翻译软件
指导教师对毕业设计（论文）的评语： 指导教师签名：_____ 指导教师职称：_____
评阅教师对毕业设计（论文）的评语： 评阅教师签名：_____ 评阅教师职称：_____
答辩委员会对毕业设计（论文）的评语： 答辩委员会评定，该生毕业设计（论文）成绩为：_____ 答辩委员会主席签名：_____　职称：_____ 年　月　日

基于opencv的即时翻译软件

摘　要

随着计算机软件的发展，人工智能(AI)也在近年得到了飞速的发展，人工智能是计算机科学领域的一个分支，它识图揭开智能的本质，并产生一个种新的能以人类智慧相似的方式做出反应的智能机器。该领域包括，图像识别、文字识别、自然语言处理。人工智能自从诞生以来，理论与技术日趋成熟，前景也在不断扩大。

可以说人工智能程序的“眼睛”就是图像识别程序，图像识别教会了AI 程序看东西，而文字识别程序认识东西。本系统就是为了解决非文本的光学字符识别而开发的工具。

本系统使用在 Windows 下的 Visual Studio 2015 中使用 C#高级程序语言开发，图形界面使用了 WPF 框架，XAML 负责绘制图形界面，C#负责后台逻辑代码编写，图像引擎为基于 opencv 的 tesseract，由后台调用。用户可以通过加载一张载有文字的图片或者使用相机获取，经过系统处理后得到文本状态的文字内容，这样再将文字进行二次处理就十分简单了，例如，自然语言识别、翻译。本系统也提供了翻译为多国语言的二次处理。

关键词　图像识别；文字识别；翻译；tesseract

Instant Translation Software Based on Opencv

Abstract

With the development of computer software, artificial intelligence (AI) has also developed rapidly in recent years. Artificial intelligence is a branch of the field of computer science. It uncovers the essence of intelligence and produces a new intelligent machine that can react in a similar way of human intelligence. This area includes image recognition, character recognition, and Natural Language Processing. Since the birth of artificial intelligence, the theory and technology have matured and the prospects are expanding.

It can be said that the "eye" of AI program is the image recognition program. Image recognition teaches the AI program to see things, and the word recognition program knows things. This system is a tool developed to solve non text optical character recognition.

This system is used in the Visual Studio 2015 under Windows to use the C# advanced programming language, the graphical interface uses the WPF framework, the XAML is responsible for drawing the graphical interface, C# is responsible for the compilation of the back-end logic code, the image engine is opencv based Tesseract, and is called by the background. By loading a picture of a text or using a camera, the user can get the text content after the system is processed, so it is very simple to carry out the two processing of the text, for example, the natural language recognition and translation. The system also provides the two processing of translation for multi lingual languages.

Keywords image recognition, word recongnition, translation, tesseract

目　录

摘要 I
Abstract II

第1章 绪论 1
1.1 数字图像处理历史简介 1
1.2 图像处理技术的发展前景 2
1.3 光学字符识别简介 2
1.4 课题主要任务 3
第2章 系统分析 4
2.1 可行性分析 4
2.1.1 社会可行性 4
2.1.2 技术可行性 4
2.1.3 经济可行性 4
2.2 需求分析 4
2.3 程序框架设计 5
2.3.1 程序运行环境 5
2.3.2 开发环境 5
2.3.3 程序框架 6
2.3.4 数据库设计 6
2.3.5 数据库介绍 7
2.4 本章小结 7
第3章 系统主要功能模块设计 8
3.1 程序主界面的设计 8
3.2 相机模块的设计 9
3.3 主题模块的设计 11
3.4 设置模块的设计 12
3.5 本章小结 14
第4章 系统功能实现 15
4.1 程序主界面的实现 15
4.1.1 标题栏的实现 15
4.1.2 按钮逻辑的实现 16
4.1.3 多语言功能的实现 19
4.1.4 翻译框的实现 20
4.2 相机模块的实现 21

4.2.1 相机模块界面的绘制.....21
4.2.2 相机模块录入逻辑.....22
4.3 主题模块的实现.....23
4.3.1 主题样式切换功能.....23
4.3.2 不同窗体通信.....25
4.3.3 主题模块自动隐藏功能.....26
4.4 设置模块的实现.....27
4.4.1 设置模块的底层控件.....27
4.4.2 设置模块集成的主题功能.....28
4.4.3 反馈功能的数据库设计.....29
4.5 本章小结.....30
第5章 程序测试.....31
5.1 内存使用率分析.....31
5.2 CPU使用率分析.....31
5.3 GPU使用率分析.....32
5.4 本章小结.....32
结论.....33
致谢.....34
参考文献.....35
附录A.....36
附录B.....45

第1章 绪论

1.1 数字图像处理历史简介

数字图像处理最早出现于上个世纪五十年代，当时的电子计算机已经发展到一定水平，人们可以开始利用计算机来帮助处理图形和图像信息。数字图像处理作为一门学科大致形成于上个世纪六十年代前期。最早期的图像处理技术是用来提高图像的质量，它使用人作为对象，以改善人的视觉效果为目的。在图像处理过程中，输入端是低质量的图像，输出端是改善质量后的新图像，常用的图像处理方法有图像增强、复原、编码、压缩等。首次获得实际成功应用的是美国喷气推进实验室（JPL）。他们对航天探测器徘徊者 7 号在 1964 年发回的几千张月球照片使用了图像处理技术，如几何校正、灰度变换、去除噪声等方法进行处理，并考虑了太阳位置和月球环境的影响，由计算机成功地绘制出月球表面地图，获得了巨大的成功。随后又对探测飞船发回的近 10W 张照片进行更为复杂的图像处理，因此获得了月球的地形图、彩色图及全景镶嵌图，获得了举世瞩目的成果，为人类登月创举奠定了厚实的基础，也推动了数字图像处理这门学科的诞生。在以后的宇航空间技术，如对火星、土星等星球的探测研究中，数字图像处理技术都发挥了巨大的作用。数字图像处理取得的另一个巨大成就是在医学上获得的成果。

20 世纪 60 年代末，计算机视觉开始于倡导人工智能的大学。它的目的是模仿人类视觉系统，作为赋予机器人智能行为的垫脚石。在 1966，人们相信这可以通过一个项目来实现，通过将照相机连接到计算机上并使它“描述它所看到的”。

计算机视觉与当时流行的数字图像处理领域的区别在于，从图像中提取三维结构，目的是实现全面的场景理解。20 世纪 70 年代的研究形成了许多计算机视觉算法的早期基础，这些算法包括从图像中提取边缘、标记线、非多面体和多面体建模、将物体表示为较小结构的互连、光流和运动估计

接下来的十年，基于计算机视觉的更严格的数学分析和定量方面的研究。这些包括尺度空间的概念，从各种线索的形状推断，例如阴影、纹理和焦点，以及被称为蛇的轮廓模型。研究人员还认识到，这些数学概念中的许多可以在与正则化和马尔可夫随机场相同的优化框架下进行处理。90 年代以前的一些研究课题变得比其他研究更活跃。投影三维重建的研究，使更好地了解相机校准。随着摄像机标定优化方法的出现，人们已经从摄影测量学的角度认识到束调整理论中的许多思想。这导致从多个图像的场

景的稀疏 3D 重建的方法。对稠密立体对应问题和多视点立体技术进行了研究。同时利用图割的变化来解决图像分割问题。这十年也标志着第一次统计学习技术在实践中被用来识别图像中的人脸。20 世纪 90 年代末，随着计算机图形学和计算机视觉领域相互作用的增强，出现了重大变化。这包括基于图像的绘制、图像变形、视图插值、全景图像拼接和早期光场绘制。

最近的工作已经看到了基于特征的方法的复苏，结合机器学习技术和复杂的优化框架。

1.2 图像处理技术的发展前景

对图像的处理技术现在的发展已经十分火热了，目前最为重要的 AI 人工智能领域就设计到很多的对图像的处理，而图像文字识别就这里面的分类之一，机器 AI 的自我学习涉及的深度学习部分就会使用到对图像的分析，可以说图像识别技术的成熟加速了人工智能技术的发展。像之前很火热的围棋 alpha GO 就是一款人工智能程序。它基于“深度学习”的工作原理，它可以自己学习分析已有的成千上万的棋谱来提高自己的技术。

无独有偶，现在手机相机上都装有“美颜”功能，而这个功能发展到现在有了“AI 美颜”功能。所谓“AI 美颜”是指美颜程序本身变成了一个 AI 程序，不同于传统的程序在于，AI 程序可以通过自我学习来不断进化，而“AI 美颜”程序是怎么进化的呢，通过给程序提供巨大量的图片学习专业人士的调图方式，然后在真正工作时，通过上百个面部特征点,来选择合适的方案进行调整。

在以前对于图片的虚化大多依赖于拍照时的对焦，但是在手机上由于元件尺寸的限制很难做到这一点。于是在 AI 程序进行人像识别的训练之后，通过算法进行人像拍照的背景虚化也变得普遍起来。

摄像头是机器的眼睛，而对于图像的分析处理技术就是机器的视神经，中心程序只接受处理完得到的数据。

1.3 光学字符识别简介

本毕业设计图像处理引擎使用了基于 opencv 的 tesseract ocr 引擎。

OCR 的概念是在 1929 年由德国科学家图舍克最先提出来的，后来美国科学家亨德尔也提出了利用技术对文字进行识别的想法。而最早对印刷体汉字识别进行研究的是 IBM 公司的凯西和纳吉，1966 年他们发表了第一篇关于汉字识别的文章，采用了模板匹配法识别了 1000 个印刷体汉字。

早在上世纪六七十年代，世界各国就开始有 OCR 的研究，而研究的初期，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。以同样拥有方块文字的日本为例，上世纪六十年代初左右开始研究 OCR

的基本识别理论，初期以数字为对象，直至上世纪六十年代中叶开始有一些简单的产品，如印刷文字的邮政编码识别系统，识别邮件上的邮政编码，帮助邮局作区域分信的作业；也因此至今邮政编码一直是各国所倡导的地址书写方式。

上个世纪七十年代初，日本的学者开始研究汉字识别，并做了大量的工作。中国在 OCR 技术方面的研究工作起步较晚，在七十年代才开始对数字、英文字母及符号的识别进行研究，七十年代末开始进行汉字识别的研究，到上个世纪七十年代中叶，我国提出“863”高新科技研究计划，汉字识别的研究进入一个实质性的阶段，清华大学的丁晓青教授和中科院分别开发研究，相继推出了中文 OCR 产品，现为中国最领先汉字 OCR 技术。早期的 OCR 软件，由于识别率及产品化等多方面的因素，未能达到实际要求。同时，由于硬件设备成本高，运行速度慢，也没有达到实用的程度。只有个别部门，如信息部门、新闻出版单位等使用 OCR 软件。进入上个世纪九十年代以后，随着平台式扫描仪的广泛应用，以及我国信息自动化和办公自动化的普及，大大推动了 OCR 技术的进一步发展，使 OCR 的识别正确率、识别速度满足了广大用户的要求。

1.4 课题主要任务

在日常生活中，有时需要快速的将图片上的文字翻译为其他语言，如果使用手动输入再翻译的话，效率极其的低下，如果能有一个程序，能够批量的，识别图片上的文字，并且能够快速的翻译为想要的目标语言，并保存为文本，那将极大提高人们的效率，因此，为了方便日常生活，工作中文本阅读和翻译的需求，从实用的角度开发了一款能够识别图片文字信息，并且翻译为目标语言的软件，考虑到使用场景的不同，还设计了两种模式，一种是可以直接读取自己电脑本地上的图片进行处理，另一种是可以通过摄像头获取图片信息，再进行处理，这样最大化的包含了用户可能使用到的场景，帮助用户迅速快捷的处理信息。

第2章 系统分析

2.1 可行性分析

商业软件的开发和设计的根本目的是为了满足用户的需求。可行性是开发的系统具有使用的价值与较长的生命周期的重要保证。

2.1.1 社会可行性

当前社会是一个物联网的社会，在全球化的今天，我们不可避免也有必要与他国进行各种交流，在交流的过程中，语言是最为重要的一环，而文字作为语言的载体就显得尤为重要。人们需要阅读文档或者书籍与其他人进行交流，本软件正式应这一需求而诞生。为个人，为企业提供高效、快速的光学字符的识别和翻译功能。

2.1.2 技术可行性

本系统采用 C#语言，用 SQL server 数据库作为后台数据库。安装和使用简单易懂，容易操作。前端界面框架使用 WPF 框架。WPF 和 QT、Duilib、MFC 相比具有很大的优势，它可以轻松地搭建美观、高效图形用户界面。服务器使用了腾讯云的服务。开发工具使用了微软的 Visual Studio2015。开发人员在学习了这些工具和语言之后，可以有能力且轻松地开发本系统。

2.1.3 经济可行性

经济可行性是指软件所带来的经济效益与开发设计所需要的投资比是否合适，软件能否带来经济收益。首先，伴随着计算机硬件和计算机软件的不断发展，计算机硬件价格不断下降，而软件则使用了 visual studio 的社区版本。设计本套系统的成本下降。而本系统的应用前景广泛。因此，在经济上是可行的。

2.2 需求分析

需求分析是本系统计划阶段的非常重要的一环，并且也是软件生命周期中非常重要的一个环节，该环节主要是分析系统在功能上要去“实现什么”，而不是思考如何去“实现”。需求分析的目的是能把用户对开发软件提出的“要求”或“需求”进行整理与分析，确认之后形成描述完整、清晰与规范的文档，以确定系统需要实现哪些功能，完成哪些工作。此外，系统的一些非功能性需求(比如软件性能、软件可靠性、软件响应时间、软件可扩展性等)，软件设计的约束条件，运行的时候与其他软件之间的关系等也是软件需求分析的目标。

本系统主要实现以下功能：

- 图片文字识别：通过摄像头或者本地获取一张图片，经过算法处理得到图片上的文字。
- 翻译：将原图片文字翻译为其他国家语言。
- 多国语言识别：能够识别非中文文字再翻译为中文。
- 主题：可以修改系统主题。
- 设置：集成了辅助功能。

2.3 程序框架设计

2.3.1 程序运行环境

本毕业设计将运行于 windows10 操作系统上。

2.3.2 开发环境

开发语言：C#

开发环境：Windows10 .Net 框架 Visual Studio 2015

开发工具介绍：微软 Visual Studio 是一个来自微软的集成开发环境（IDE）。它用于开发计算机程序，以及网站、网络应用程序、Web 服务和移动应用程序。Visual Studio 使用微软软件开发平台，如 Windows API、Windows 窗体、Windows 演示基金会、Windows Store 和微软 Silverlight。它既可以生成本机代码，也可以生成托管代码。

Visual Studio 包括支持智能感知（代码完成组件）和代码重构的代码编辑器。集成调试器既是源级调试器，又是机器级调试器。其他内置工具包括代码分析器、用于构建 GUI 应用程序的表单设计器、Web 设计器、类设计器和数据库模式设计器。它接受几乎所有级别的功能，包括增加对源代码控制系统（如 Sub 和 Git）的支持，以及为软件开发的其他方面添加新的工具集，如编辑器和可视化设计器，用于特定于领域的语言或工具集。生命周期（如 Team Foundation Server 客户端：团队资源管理器）。

VisualStudio 支持 36 种不同的编程语言，并允许代码编辑器和调试器支持（不同程度）几乎任何编程语言，只要存在特定于语言的服务。内置的语言包括 C， C++，C++和 CLI，Visual Basic.NET，C#，F#，TypeScript，JavaScript，XML，XSLT，HTML 和 CSS。其他语言如 Python、Ruby、node.js 和 m 等的支持可以通过插件获得。java（J #）在过去支持。

2.3.3 程序框架

为了提供尽量好的用户体验，更好的视觉效果，本毕业设计采用了WPF 用户界面框架。

该框架采用 XAML+C#的分层设计。使用 XMAL 绘制用户界面，使用 C#编写底层逻辑代码。

如图 2-3 所示，为程序的功能框架设计示意图。

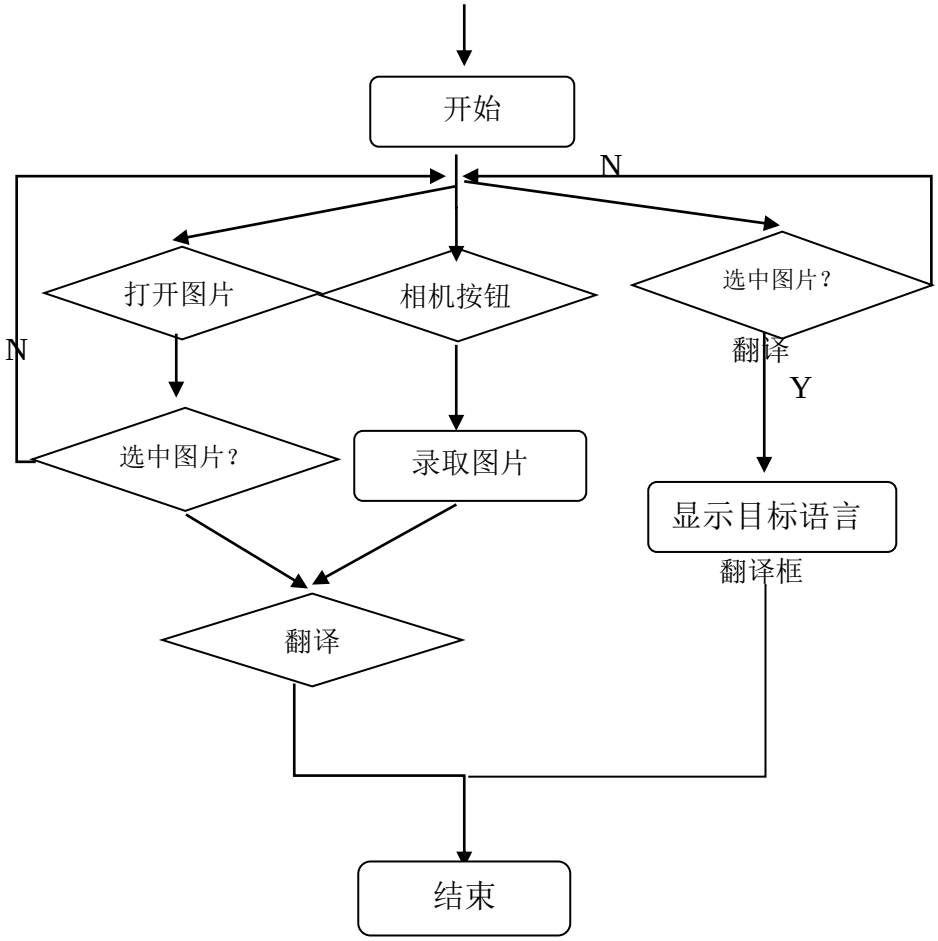

图 2-3 程序框架图

2.3.4 数据库设计

本系统的数据库主要用于程序的反馈部分，用户提交的意见或者建议将会提交到系统的数据库中记录。针对反馈功能的特点，设计了如下所示的数据项和数据结构。

反馈表数据实体：标题，正文。

表在程序初始化之前就已经添加好了。

数据库实体的 E-R 图如图 2-4 所示。

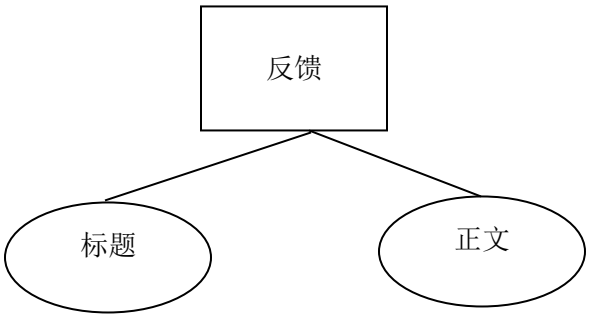

图 2-4 反馈实体 E-R 图

数据插入语句：insert into ref value(‘title’, ‘content’);

在数据库的设计过程中还内置了对提交信息的检测系统，系统内置了一份敏感词汇以及垃圾词汇的库，在上传之前系统会检测信息的有效性，避免垃圾信息占据数据库不必要的内存。

2.3.5 数据库介绍

本系统涉及到数据库的部分使用了微软 Microsoft 提供的 MicroSoft SQL Server 的数据库服务。

远程服务器采用了腾讯云提供的微软 Microsoft 的 Windows server 2012 操作系统，MicroSoft SQL Server 数据库就是搭载在该服务器上的。

SQL Server 数据库具有高性能，能充分利用 WindowsNT、系统管理先进，支持 Windows 图形化管理工具，支持本地和远程的系统管理和配置、事务处理功能强壮，采用各种方法保证数据的完整性、支持对称多处理器结构、存储过程、ODBC，并具有自主的 SQL 语言的特性。并且 SQLServer 以其内置的数据复制功能、强大的管理工具、与 Internet 的紧密集成和开放的系统结构为广大的用户、开发人员和系统集成商提供了一个出众的数据库平台。

2.4 本章小结

本章主要介绍了前两节介绍了系统设计之前的可行性分析和未来具体功能的设计方向，在第三节详细介绍了系统的运行环境、开发环境以及系统的框架设计和系统框架图示。

第3章 系统主要功能模块设计

本系统采用了模块化的设计思想，在设计的过程中，本系统被大致分为核心功能、辅助功能两个大的部分，而辅助功能也细分出了包含主题模块、设置模块、相机模块等三个部分。

3.1 程序主界面的设计

如图 3-1 所示，主界面采用了四层的分层设计。

图 3-1 程序主界面设计框架示意

第一层包含了最大化、最小化、关闭、主题、设置五个部分，其中前三个部分实现了程序的基础功能，后两个部分提供了对主题和设置进行操作的接口。

第二层包含了一个文本显示框，用户显示对图片信息读取后的文本内容。

第三层包含了打开图片、翻译、多语言识别功能、相机三个部分。分别提供了选择本地图片、翻译上方文本框文本、识别他国语言、打开摄像头获取图片三个方法的接口。

如图 3-2 所示，多语言识别接口提供识别他国（日语、英语、俄语）的功能，为了使界面美观，当点击图标之后，之前的菜单栏会进行重绘，提供为了提供多语言识别而特别设计的界面。虽然中间的菜单栏被重

绘了，但主界面的其他元素依然是共用的，这样的好处是可以尽量减少重复代码的编写，提高代码的简洁性。为了实现这个功能就需要解决多窗口通信的难题。

图 3-2 多语言功能界面

第四层为一个 tabControl 包含四个 tabItem 的复式控件。上层为四个语言的入口，下方将显示翻译文本内容。其中有一点小不同的是，两个文本框并不完全相同，下方的文本框采用了富文本框控件，可以显示丰富的自定义内容。如颜色、行距、换行等。

3.2 相机模块的设计

相机模块的设计分为两个阶段。

第一个阶段为功能的开发，这个阶段以实现基本的功能要求为目标，暂时放弃 UI 的设计。如图 3-3 所示，相机模块被分为了两个部分。

第一部分提供生成图片的接口，点击“录入”按钮，后台会在本机的桌面生成一张名为“photo.jpg”的图片，这张图片将被图像文字识别程序读取。

第二部分提供了相机界面的预览示意图，摄像头的即时图像将会显示在这个部分。

图 3-3 相机模块功能设计示意

第二个阶段为用户界面的设计，在这个阶段不仅要对用户界面进行重新设计，还要在底层上进行修改，使得该模块也实现对主题的支持。并且相机界面也要支持同步更换主题。

设计完成的相机模块分为三个部分。

第一部分，将系统自带的关闭按钮进行了重绘，左上角的圆形灰色按钮提供了关闭操作，中间部分添加了一个简单的文字介绍。

第二部分，为相机模块的主要部分，这里使用了.Net 封装的 wpfmedia 的 VideoCaptureElement 控件提供相机的预览界面，在这里了看到相机传回的实时界面，方便用户即时获取清晰的图片。

第三部分，用户选择“录入”图片后，将在用户使用的机器上的桌面生成名为“photo.jpg”的图片文件供图像文字处理程序解析。要求在不同机器上使用时也能正确地生成图片。

由于某些老旧的设备并无摄像头，因此在该模块启动之前还要提供摄像头的检测功能，一旦没有识别到摄像头，将会弹出警告通知，并且不会执行下面的功能。

图 3-3 相机模块界面设计示意

3.3 主题模块的设计

主题模块是本毕业设计主题变换的核心模块，为了实现了全局主题的同步统一，主题功能需要集成进所有模块的底层。

要实现全局统一的效果就离不开窗口间的通信问题。这里通信问题的解决方法是通过父子窗口绑定的方式实现的。

通过点击衣服按钮打开界面，鼠标脱离界面关闭。这种功能的实现是通过对窗口灵活的显式化和隐式化操作实现的。在程序初始化阶段，窗口默认隐藏，点击衣服按钮后，窗口显式化。鼠标离开窗口事件触发后再隐藏。

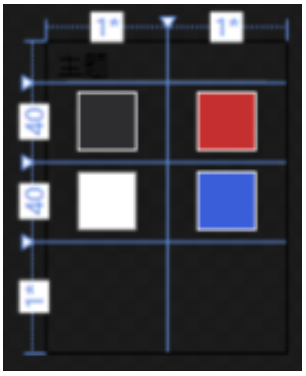

图 3-4 主题模块设计示意

这里主要分为两个部分。

第一部分为主题的 content，可以使用 label 控件或者 TextBlock 实现。

第二部分分为四个不同的主题，通过点击不同主题，就可以改变程序的风格。这个模块使用的是 page 页，使用 page 可以在当前窗体的某个部分开辟一个空间，将 page 页放置在这个地方。要实现全局主题的同步更改就需要解决窗体间通信的问题。

3.4 设置模块的设计

设置模块集中了程序的除核心功能之外的所有额外功能，像前面提到的主题功能也包含在这里面，除此之外，在设置模块里面还加入了反馈功能，以及版本说明，登陆的开机启动等常用功能。

在设置模块的设计过程中，采用了 4 层的分层设计。

第一部分，如图 3-5 所示，登录设置，加入了开机启动的功能，要求能够自动识别当前配置，能够正确的显示在设置界面中，并且不会因为重启软件而显示错误。

图 3-5 登陆设置设计示意

第二部分，如图 3-6 所示，主题功能，这里的主题功能集合了前面的主题模块的内容，两部分的操作互相共享，要求能够实现主题模块的所有功能，并且默认选项能够正确的显示当前主题，同样的，不能因为重启软件而导致显示错误。

图 3-6 主题设置设计示意

第三部分为版本说明，这里包含了程序的编译时间、开发者信息、开源地址以及版本号等信息。

图 3-7 反馈设置设计示意

第四部分，如图 3-7 所示，反馈功能，考虑到后期的维护，如果用户

在使用过程中发现了程序存在的 bug，可以通过反馈将 bug 提交给管理员。

3.5 本章小结

整体系统的完成离不开事先的未雨绸缪，在着手功能开发时首先对目标功能进行了详细的设计。而本章则分为四节分别详细地介绍了本系统的四个功能模块的设计思想，并且完整的记录了每个功能模块的分层和将要发挥的功能。在下一章将会介绍这些功能的具体开发过程。

第4章 系统功能实现

4.1 程序主界面的实现

如图 4-1 所示，首先去掉系统自带的边框，然后对关闭、最大化、最小化三个按钮进行了重绘，并赋予了相同的逻辑功能。

界面的右上角为设置、主题两个部分的接口。“打开图片”按钮提供本机图片的输入，右边相机按钮提供从摄像头获取图片。相机图标的左边按钮提供多语言识别的功能。三个接口都是作为图片信息的输入，经过图像文字识别程序的处理得到的文字信息将显示在上方的文本框中。

“翻译”按钮提供了对多国语言翻译的支持。下方的经过特别定制的文本框将按照原格式显示翻译之后的文本。

图 4-1 主界面图

4.1.1 标题栏的实现

图 4-2 标题栏

如图 4-2 所示，标题栏分为左右两个部分，左边为重绘的关闭、最小最大化三个按钮。

“关闭”按钮的 xaml 代码如下：

```
<Border x:Name="CLOSE"
ToolTip="关闭"
HorizontalAlignment="Left" Margin="20,10,0,0" Width="13" Height="13"
Background="#FFB8B8B8"
CornerRadius="13"
MouseDown="CLOSE_MouseDown"
VerticalAlignment="Top"/>
```

HorizontalAlignment 属性表示按钮的水平位置，VerticalAlignment 属性表示按钮的竖直位置，在 MouseDown 中定义了鼠标点击关闭窗体的方法。当鼠标悬空在按钮上时会显示 ToolTip 的文字。CornerRadius 定制了按钮的形状。

“关闭”按钮的关闭功能由下面的方法提供：

```
private void CLOSE_MouseDown(
                                object sender, MouseButtonEventArgs e)
{
    this.Close();
}
```

而“设置”和“主题”按钮的界面使用的空间有些许不同，以“设置”按钮为例：

```
<Border x:Name="setting"
ToolTip="设置"
HorizontalAlignment="Right"
Margin="0,12,30,0" Width="13" Height="13"
VerticalAlignment="Top"
MouseDown="setting_MouseDown"
Grid.Column="1">
    <Image Source="./img/setting.png"/>
</Border>
```

这是一个复合控件，在 Border 中还包含了 Image 图片控件，Source 指定了图片的来源路径。

而“设置”和“主题”两个部分也有一些不同，点击“设置”之后，会出现一个全新的 window 窗体，而点击“主题”按钮之后，主题模块的界面将会显示在主界面上，也就是说主题界面是内嵌在主界面上的。

4.1.2 按钮逻辑的实现

图 4-3 按钮界面

中间的“打开图片”和“翻译”按钮为两个 Button 控件。

```
<Button Name="btn1"
Content="打开图片"
FontFamily="幼圆" FontSize="13"
Background="#119EDA" Height="30"
Grid.Column="1" Width="100"
BorderThickness="0"
HorizontalAlignment="Left"
Click="btn1_Click"/>
```

这里打开图片的逻辑内容写在 Click 属性之中，包括了打开对话框和图片文字识别。

```
TesseractEngine engine = new TesseractEngine(@"C:\Program Files
   (x86)\Tesseract-OCR\tessdata\", "chi_sim", EngineMode.Default);
 engine.SetVariable("chop_enable", "F");
 engine.SetVariable("enable_new_segsearch", 0);
 engine.SetVariable("use_new_state_cost", "F");
 engine.SetVariable("segment_segcost_rating", "F");
 engine.SetVariable("language_model_ngram_on", 0);
 engine.SetVariable("textord_force_make_prop_words", "F");
 engine.SetVariable("edges_max_children_per_outline", 50);
```

这里对Tesseract引擎进行了配置。

```
 if (imageSource == null)
 {
     var file = new Microsoft.Win32.OpenFileDialog();
     file.Filter = "所有文件(*.*)|*.*";
     var image = file.ShowDialog();
```

这里将打开选择本地图片的对话框，选中的图片路径将存储在file.FileName之中，然后赋值给pathname。

```
     if (file.FileName != string.Empty)
     {
        try
        {
              pathname = file.FileName;
        }
        catch (Exception ex)
        {
              MessageBox.Show(ex.Message);
```

```
            }
        }
    }
        else
        {
            string dir = Environment.GetFolderPath(
                    Environment.SpecialFolder.DesktopDirectory);
            pathname = dir + @"\photo.jpg";
        }
```

dir存储了本机桌面路径，它会自动识别不同机器的桌面路径。

```
        if (pathname != null)
        {
            System.Drawing.Image img =
              System.Drawing.Image.FromFile(pathname);
             Bitmap p = (Bitmap)img;
            var page = engine.Process(p, pageSegMode:
              engine.DefaultPageSegMode);
             var testText = "";
             testText = page.GetText();
             tb1.Text = testText;
             if (!testText.Equals(""))
             {
                 btn2.IsEnabled = true;
             }
        }
}
```

这里方法的默认参数 imageSource 是为了将两种路径来源的图片文字识别实现写在同一个方法中，当参数为空时，图片来源为本地，不为空时，图片来源为相机获取，当然两种功能分别由不同的接口提供。

而“翻译”按钮的逻辑实现为：

```
private void GetEnglish()
{
            string result = GetJson("EN");
            Root rt = JsonConvert.DeserializeObject<Root>(result);
            this.tb2.Document.Blocks.Clear();
            for (int i = 0; i < rt.translation.Count(); i++)
            {
                tb2.AppendText(rt.translation[i] + "\n");
            }
}
```

第三行 result 存储了返回的 JSON 数据，再通过 rt 解析获取的 JSON 数据，最终将数据显示在文本框之中。

4.1.3 多语言功能的实现

多语言识别的功能是通过右侧的按钮来提供的，点击按钮后会生成一个新的界面覆盖中间的部分，如图所示。

图 4-4 重绘的按钮栏

这里提供了新的接口。

覆盖的区域为：

```
<ContentControl Name="MulLanguageModule"
Grid.ColumnSpan="3" Grid.Row="0" Margin="0,0,30,0"/>
```

后端代码实现为：

```
MulLanguageBar bar = new MulLanguageBar();
MulLanguageModule.Content = new Frame() { Content = bar };
```

因为不同文字的文字识别设计到不同的处理逻辑，因此需要在不同的地方编写不一样的代码，最早的方案是将主程序的上半部分包括逻辑按钮全部覆盖，重新绘图，每个不同的语言都再进行一次单独页面的逻辑编写，这样做的话整个程序其实也比较简单，但是希望能做到代码的复用，避免重复冗余地编写代码，使得代码显得尽量美观。这个方案只需要在对应的界面里将对应部分的方法进行重载或者部分修改，但是工程量非常大并且显得非常不完美，代码冗余。于是就放弃了这个方案。

后来第二个方案决定只修改按钮部分的实现，对话框还是使用同一个，这样避免了不必要的重复，在这个方案的前半部分，选择的是，将中间按钮部分分为两部分，左边以“打开图片”和“翻译”两个按钮为一部分，右边将多语言选择功能作为一部分，这样做的好处就是使得界面的改动尽量做到最小。点击多语言之后左边部分的按钮会进行重绘，在重绘的按钮里写入了对其他语言的文字识别支持。但这样又出现了一个问题，因为两部分都是不同的 page，所以在 page 到 page，再从 page 到 window 的信息交互就出现了问题，在最上层的 page 中获得的文字信息无法传递到底层 window 的文本框中。这里有一个最简单也是最粗暴的解决方案，在打开多语言功能的同时就打开一个新的隐藏的文本框界面，但是这样处理的话会导致出现上面相同的问题，会需要新建多个 page，这也违背最初保持代码简洁，最大化复用的初衷，希望能在一个 page 中完成所有语言的文字识别。

经过长时间的考虑最终选择了一种比较好的解决方案，将上面的两部分合并在同一个 page 中，这样就只存在 page 到 window 的信息交互，也避免了再绘制一个新的麻烦的文本框，只需要在不需要多语言部分后动态地将它移除。这样就有了图 4-5 和图 4-6 的显示。

图 4-5 重绘的按钮栏

图 4-6 重绘的按钮栏

4.1.3.1 多语言功能逻辑的实现

中文的文字识别和日语的文字识别使用的逻辑是有一些差别的，因此需要重新考虑日语识别逻辑部分的修改，在原代码的基础上，将固定话的变量修改为传参形式，这样既能减少代码的编写，也能获得更多的功能。

翻译部分将对应的方法进行了重载使得也支持了任何形式的语言到任何形式语言的翻译。

```
private void btn2_Click(object sender, RoutedEventArgs e)
{
        GetJapanese();
        this.Visibility = Visibility.Hidden;
        this.parentWindow.btn1.IsEnabled = true;
}

private void GetJapanese()
{
        string result = GetJson("ja", "zh-CHS");
      Root rt = JsonConvert.DeserializeObject<Root>(result);
        //解析 JSON 数据
     this.parentWindow.TbJapanese.Document.Blocks.Clear();
        //清空文本框
    for (int i = 0; i < rt.translation.Count(); i++)
    {
            this.parentWindow.TbJapanese.
                    AppendText(rt.translation[i] + "\n");
    }
}
```

4.1.4 翻译框的实现

最初使用 TextBox 文本框的时候会出现文本格式错乱，转义字符不能

识别的问题。为了使前后的文本方便对照，引用了 RichTextBox 来解决这个问题，但是原生的 RichTextBox 控件格式差异依旧很大，因此对 RichTextBox 控件进行了专门的自定义解决了文本格式不一致的问题。

为了实现多种语言的快速切换，翻译框在底层使用了 TabControl 标签栏，如图 4-6 所示。

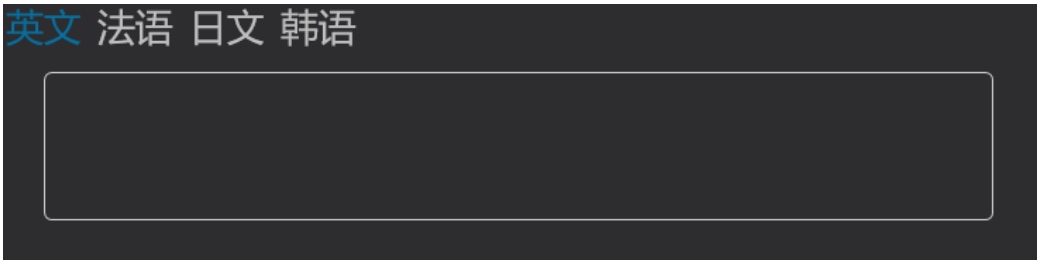

图 4-7 翻译标签栏

4.2 相机模块的实现

4.2.1 相机模块界面的绘制

这个部分提供了界面的相机组件，界面如题 4-8 所示。

```
<Border Width="240" Height="175"
BorderBrush="#FF3399FF"
BorderThickness="1">
        <wpfmedia:VideoCaptureElement Name="video"/>
</Border>
```

加载时的初始化工作，录入按钮在没有摄像头时设为不可用状态：

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    if(MultimediaUtil.VideoInputNames.Length > 0)
    {
        video.VideoCaptureSource =
          MultimediaUtil.VideoInputNames[0];
    }
    else
    {
        Warning warn = new Warning("没有检测到任何摄像头");
        btn.IsEnabled = false;
        isHasCamera = false;
        warn.ShowDialog();
     }
 }
```

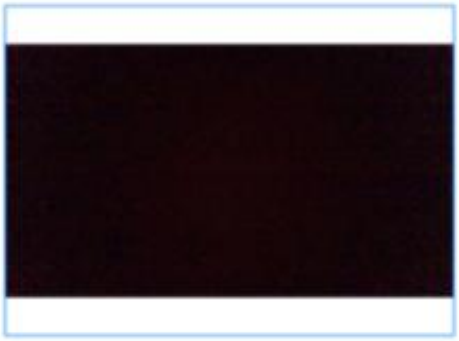

图 4-8 相机界面

程序初始化时，只有运行机器有摄像头时才会执行后面的功能，否则会出点“没有检测到任何摄像头”的警告，如图 4-9 所示。

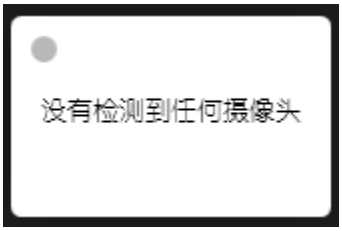

图 4-9 警告图

4.2.2 相机模块录入逻辑

Click 的点击事件：

```
public void SaveImage(string fileName)
{
        System.IO.FileStream fs =
                        new System.IO.FileStream(
                        fileName, System.IO.FileMode.Create);
        RenderTargetBitmap bitmap = new RenderTargetBitmap(
        (int)video.ActualWidth,
        (int)video.ActualHeight, 0, 0, PixelFormats.Default);
        bitmap.Render(video);
        BitmapEncoder encoder = new PngBitmapEncoder();
        encoder.Frames.Add(BitmapFrame.Create(bitmap));
        encoder.Save(fs);
        fs.Close();
}
```

该函数负责保存摄像头获取的图片，传入变量 filrName 为图片将要存储的目标路径，RenderTargetBitmap 定义了图片的尺寸。

4.3 主题模块的实现

4.3.1 主题样式切换功能

4.3.1.1 XAML 界面绘制

主题模块供选择了四个默认主题样式，xaml 的代码实现为：

黑色主题样式如下：

```
<Border Name="RgbBlack" Grid.Column="0"
Grid.Row="1"
Width="30" Height="30"
Background="#2D2D30"
   MouseDown="RgbBlack_MouseDown" />
```

红色主题样式如下：

```
    <Border Name="RgbRed" Grid.Column="1"
 Grid.Row="1"
   Width="30" Height="30"
   Background="#C62F2F"
   MouseDown="RgbRed_MouseDown" />
```

白色主题样式如下：

```
      <Border Name="RgbWhite"
   Grid.Column="0" Grid.Row="2"
   Width="30" Height="30"
   Background="White"
   MouseDown="RgbWhite_MouseDown"/>
```

蓝色主题样式如下：

```
      <Border Name="RgbBlue"
   Grid.Column="1" Grid.Row="2"
   Width="30" Height="30"
   Background="#3A5DD9"
   MouseDown="RgbBlue_MouseDown"/>
```

这里分别使用了 4 个 Border 控件来展示主题的样式，后面可以根据需要再增加新的主题样式，后台的逻辑代码通过 MouseDown 属性来实现，主题界面使用的是 page 类，page 可以嵌入在主界面之中，点击“主题”按钮之后将会显示在按钮下方，鼠标移出后该界面会自动隐藏，界面如图 4-10 所示。

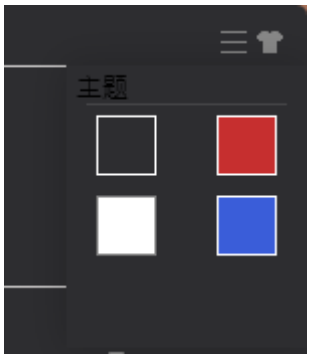

图 4-10 主题模块图

4.3.1.2 样式切换的逻辑代码

首先编写了一个修改默认配置文件的函数，每次修改主题后会在配置文件中自动同步当前主题配置，这么做的原因是为了保持所有界面的主题一致性。

```
private void ChangeDefaultColor(string str)
 {
        System.Configuration.Configuration config =
              ConfigurationManager.OpenExeConfiguration(
              ConfigurationUserLevel.None);
        config.AppSettings.Settings
                    ["Skin.Color.Default"].Value = str;
        config.Save(ConfigurationSaveMode.Modified);
        ConfigurationManager.RefreshSection("appSettings");
 }
```

然后通过将数据传递到父窗口（通过我们自己实现的多窗口通信功能），改变当前窗口的值。赋值的方式是赋为当前配置文件的默认值，因为我们首先修改的就是默认配置文件。

```
private void RgbBlack_MouseDown(
        object sender, MouseButtonEventArgs e)
 {
        this.parentWindow.border1.Background =
            GetColor(ConfigurationManager.
            AppSettings["Skin.Color.Black"]);
        ChangeDefaultColor(ConfigurationManager.
            AppSettings["Skin.Color.Black"]);
}
```

4.3.2 不同窗体通信

因为程序框架的原因，除了主界面之外还会有其他的窗体来协助实现一些其他的功能，因此如何在操作其他界面时将信息传送回主界面成为了要解决的第一大问题。为了实现这个功能期间做了很多的尝试。

1)采用 WPF 的 Binding 通过对数据操作，然后再不同界面读取这个数据实现数据通信。在实际操作中发现还是会出现需要两个不同实例才能解决的问题，因此被 pass 掉。

2)采用全局静态变量。因为不能即时地检测数据的变化，因此也被 pass。

3)将主窗体与其他窗体绑定为父子的关系。这样完美解决了数据变化的即时性和实例的单一性。最后采用的也是这个方案。

父窗体：

```
Skin skin = new Skin();
//全局申明，方便SKIN_MouseDown()每次点击加载配置文件
 public string DebugTime;
 public MainWindow()
 {
      InitializeComponent();
      //主题从默认配置文件加载
      cc . Content = new Frame() { Content = skin };
      skin . ParentWindow = this;
      //绑定Page的父窗口
      //cc . Visibility = Visibility . Collapsed;
      if (tb1 . Text . Equals(""))
      //文本框内容为空时，翻译按钮设为不可用
      {
           btn2 . IsEnabled = false;
      }
      DebugTime = DateTime . Now . ToString();
 }
```

第一行声明了一个皮肤类的实例，该类描述了皮肤模块的所有功能，第二行的 Debug Time 是为后面版本的设置功能里版本功能做准备的。cc . Content = new Frame() { Content = skin };cc . Content 是在主界面的某个部分添加了一个新的模块空间， content = skin 表示将实例 skin 赋值给该空间，skin 的界面功能将显示在该位置。skin . ParentWindow = this;表示将 skin 模块的父窗口设为当前窗体，这是为了不同窗体建信息传递做准备的。

子窗体：

```
MainWindow parentWindow;
```

```
public MainWindow ParentWindow
{
    get
    {
        return parentWindow;
    }
    set
    {
        parentWindow = value;
    }
}
```

第一行在子窗体中声明了父窗体的实例，这是为了绑定做准备，第二行建了一个父窗体的方法，用来获取子窗体的信息。通过调用子窗体的 ParentWindow 方法来将父窗体绑定给子窗体的方法来实现不同窗体间的信息传递。

4.3.3 主题模块自动隐藏功能

最初的功能是主题模块点击出现后会驻留在界面上，不会自动消失。后来为了让操作更人性化，于是修改为鼠标移出主题模块界面后，会自动关闭。

1)引入标记变量，记录每一次鼠标点击后窗口的状态。

2)将主题模块设为隐藏，当鼠标发生点击事件时变为可见，当鼠标触发 MouseLeave 事件时再设为不可见。本毕业设计采用了后一种方案。

```
Skin skin = new Skin();
//全局申明，方便SKIN_MouseDown()每次点击加载配置文件
public string DebugTime;
public MainWindow()
{
    InitializeComponent();
    cc . Visibility = Visibility . Collapsed;
}
private void SKIN_MouseDown(object sender,
                            MouseButtonEventArgs e)
{
    skin . border . Background = new SolidColorBrush((System .
                            Windows . Media . Color)System .
    Windows . Media . ColorConverter .
                    ConvertFromString(ConfigurationManager .
  AppSettings["Skin . Color . Default"]));
   cc . Visibility = cc . Visibility == Visibility . Collapsed ?
```

```
            Visibility . Visible : Visibility . Collapsed;
        }

        private void Page_MouseLeave(object sender, MouseEventArgs e)
        {
            this . parentWindow . cc . Visibility = Visibility . Collapsed;
        }
```

由于前面已经解决了界面的通信问题，因此主题修改后的变化是即时的，但是出现了一个问题就是，每次打开程序后都会重新初始化一次，修改后的主题并不能保存下来。于是这里直接引入了默认配置文件，每一次主题修改后都会直接作用到默认配置文件上，每次打开程序读取的也是配置文件的内容，这样就保证了前后主题的一致性，后期如果有需要的话可能会将数据保存在云端防止信息被篡改。

```
<configuration>
    <startup>
        <supportedRuntime version="v4 . 0" sku=" .
NETFramework,Version=v4 . 6 . 1" />
    </startup>
  <appSettings>
    <add key="Skin . Color . Black" value="#2D2D30" />
    <add key="Skin . Color . White" value="#FFFFFF" />
    <add key="Skin . Color . Blue" value="#3A5DD9" />
    <add key="Skin . Color . Red" value="#C62F2F" />
    <add key="Skin . Color . Default" value="#FFFFFF" />
    <add key="AutoStart" value="true" />
    <add key="Default" value="0" />
  </appSettings>
 </configuration>
```

在配置文件中对 4 个预设主题样式进行了定义，还额外添加了一个 default 配置，该配置会根据主题的改变同步改变，系统每次启动时都会读取该配置。

4.4 设置模块的实现

4.4.1 设置模块的底层控件

设置模块在底层使用了 TabControl 控件来应对需要多个标签页的需求。在 TabControl 中采用了内嵌 TabItem 来实现标签页的功能，无独有偶，在翻译框处也采用类似的设计。

```
<TabControl TabStripPlacement="Left"
Grid.ColumnSpan="2"
Background="Transparent"
Margin="20,30,10,20">
        <TabItem Header="登录">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="88*"/>
                    <ColumnDefinition Width="349*"/>
                </Grid.ColumnDefinitions>
                <StackPanel Grid.Column="1"
 Margin="0,20,0,0">
                        <CheckBox Name="CheckBox"
 Click="CheckBox_Click">
                            <TextBlock Text="开机自动启动"/>
                        </CheckBox>

                </StackPanel>
            </Grid>
            </TabItem>
</TabControl>
```

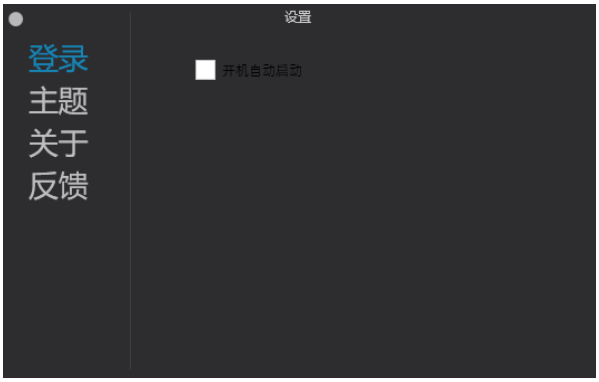

图 4-11 设置界面

4.4.2 设置模块集成的主题功能

设置功能本来就是各种附加功能的集合，因此也是存在重新集成的主题设置。

图 4-12 主题设置功能

如图 4-12 所示，这里集成的主题设置使用了 ComboBox 控件，它提供了下拉选项功能的单选框。值得一提的是，初始化时，默认的选项为当前设置的主题，并且这里的设置和全局一样是完全同步的。

4.4.3 反馈功能的数据库设计

如图 4-13 所示，为了提供良好的使用体验，在出现问题时及时补救，在本系统后面加入了反馈功能。用户可以将在使用过程中遇到的问题提交到服务器，以作后面系统的维护、修改的参考。

图 4-13 反馈功能

如表 4-1 所示，该表介绍数据库中使用到的数据表和数据类型。

表 4-1 反馈表

列名	数据类型	长度	说明
Title	Varchar	140	反馈标题
Content	Varchar	140	反馈内容
Date	DateTime	10	反馈日期

4.5 本章小结

程序的完美运行离不开各个功能模块的协同工作，而本章则分为四大节分别详细地介绍了本毕业设计四个功能模块的实现过程，并且完整的记录了每个功能模块发挥的作用，此外，还对每个模块的每个部分都进行了详实而充分地解释说明，甚至最微小的功能细节都做到了应有尽有的记录。

第5章 程序测试

在系统各功能完成之后，在代码整合的过程中对程序进行功能测试。检查是否实现了预定的功能设计，在系统运行过程中是否存在因内存泄漏，变量传参错误等导致的系统崩溃。在设计的最后阶段，检测用户界面是否和预定设计得一样美观友好，是否突出特色，是否达到风格一致。

5.1 内存使用率分析

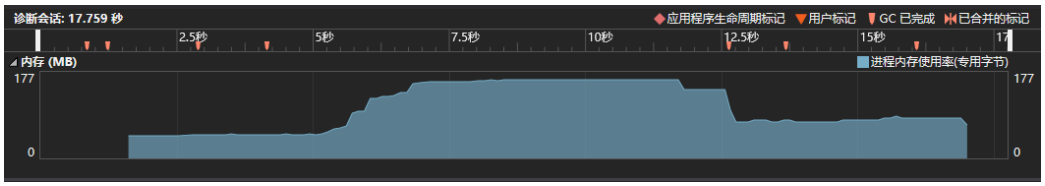

图 5-1 内存使用率分析图

如图 5-1 所示，可以看到在挂起状态，程序的内存占用量十分小且稳定。而在执行图片识别功能和翻译功能时内存占用量明显增加但也十分稳定，因此本系统并未出现内存泄漏的情况

5.2 CPU 使用率分析

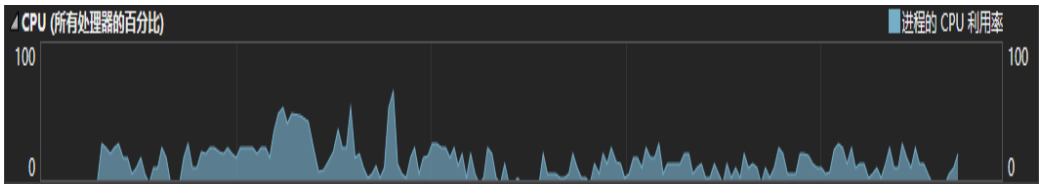

图 5-2 CPU 使用率分析图

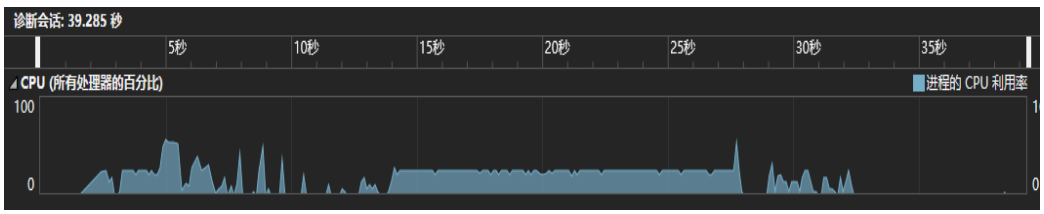

图 5-3 大段文字 CPU 使用率分析图

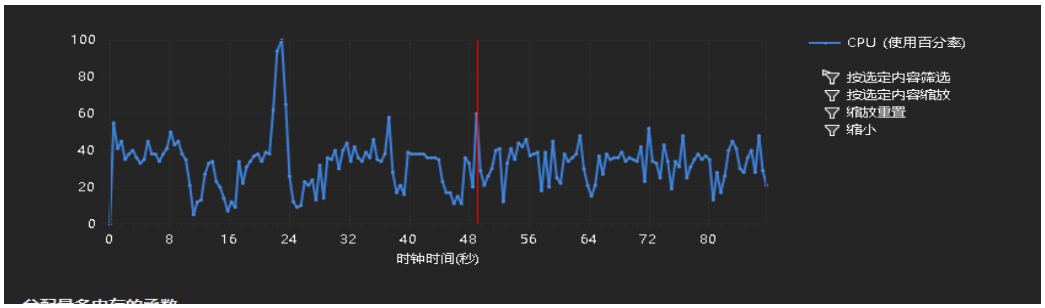

图 5-4 CPU 采样分析图

如图 5-2 所示，可以看到在调用系统底层组件的时候，cpu 占用明显升高，但这和使用者的 cpu 型号、频率也密切相关。如图 5-3 所示，在图片文字信息较长时，处理的时间也随之增长，但效率十分稳定。本次测试采用英特尔酷睿 2 代 i3 中央处理器。

5.3 GPU 使用率分析

图 5-5 GPU 使用率分析图

如图 5-5 所示，可以看到，在不涉及前半部分的系统对 GPU 的调用十分少，而在使用标签页时，图形渲染工作量增大，GPU 占用量也随之提高，但这和使用者所使用图形处理器密切相关，测试时使用的为英特尔处理器的核芯显卡。

5.4 本章小结

软件的测试对实现一个完整系统十分重要，软件测试保证了系统的正确性、完整性、安全性。是软件系统质量的重要保证。本章从内存、中央处理器，图形处理器的占用层面对本系统进行了测试。

结论

本文详细地介绍了图像文字识别的实现和文本的翻译。结合了自己学习的 Windows 操作系统，图形界面程序开发、软件工程多学习的知识，考虑到用户使用的具体情况，采用了多种便捷、高效的设计。通过 Windows 下的 Visual Studio 2015 进行了代码的编码工作和编译、性能分析。

在开发过程中，深刻地感受到完成一个大型的项目之前，进行调查、需求分析、系统设计的重要性。调查可以将项目的需求尽可能完成地记录，以便结合实际情况选择开发工具与开发环境。达到事半功倍的作用。保证了项目进行时编码工作能够稳定地展开，不会因为需求的改变和准备不完善而影响编码进度。系统设计能保证在总览全局的情况下实现系统的最优设计，不会出现严重的框架问题导致项目开发进程终止。在充分了解系统之后，再进行详细的功能设计，这样能够尽可能地完善项目的各种细节，做到尽善尽美。

在编码工作时对代码的整理也非常重要，完善的注释、清楚的文档可以很好地帮助回顾之前的工作，为后面的编码工作打下好的基础。大型项目的开发不同于小程序，合理地使用面向对象思想，代码尽可能做到能够复用，简洁。代码命名做到清晰、清楚、一目了然。

不足之处在于：

在某些算法地方的效率没有进一步地优化，例如数据库的垃圾信息过滤，直接使用了 string 库的方法，时间复杂度为 logn。一旦数据量过多，性能下降很明显。

界面程序只使用了一个线程，对 CPU 的利用率不高，如果采用多线程应该能提高图形界面的性能。

本次开发完成预定的需求，并按时提供给用户使用。在软件生命周期也会不断维护，根据需求添加新功能。

致谢

本毕业设计的最初构想基于我在大学本科学习的第二年，这个毕业设计，当然也包括这些功能得益于我的老师们对我专业知识的启发，当然还有我的朋友们，在万分繁忙的时候，对我的程序进行功能的测试，感谢他们百忙中挤出的时间，不仅进行了非常出色的测试，还对本毕业设计的草稿提出了非常宝贵的意见。

在开发此程序之前，我认为只要开发出功能合格的程序就行了，因此我也觉得本毕业设计只需要有完整的功能就行了，但是我错了，我需要合作者可以从用户的角度审视这个程序，并告诉我需要做什么、应该做什么和能够做什么，如果我有足够的经历的话。

我还要向我的大学老师们表示由衷的感谢，我在课上学习到了许多有用的知识，本毕业设计的基础 C#和 WPF 也是在老师们的课程上得到启发的。没有老师们的帮助我想我的程序也不能完成得如果迅速，用户界面也不能如此精致。

最后还要感谢我的指导老师，高尚民老师。从毕业设计到毕业论文，高老师全方位的指导让整个过程显得行云流水，老师精益求精的态度也让我受益颇深。没有高老师的指导，整个项目也不会如此让人满意。

参考文献

[1] [美]Stanley B, Lippman. C# Primer [M]. 华中科技大学出版社. 2018

[2] [美]JacquieBarke. Beginning C# Objects[M]. 电子工业出版社. 2018

[3] [美]Chris H.Pappas. Programming C# 4th[M].中国电力出版社. 2018

[4] [美]Charles Petzold. Programming Microsoft Windows Forms [M]. 人民邮出版社. 2018

[5] [美]Liberty, J. Programming C# 4th[M].电子工业出版社. 2018

[6] [6]TesseractOCR[DB/OL]. https://github.com/tesseractocr/tesseract/wiki/APIExample [美] Google. 2018

[7] [英]夏普 著，周靖 译. Visual C# 2005 从入门到精通[M]. 清华大学出版社. 2018

[8] [美]托马斯 等著，陈伟柱，陶文 译. 单元测试之道 C#版：使用 NUnit[M]. 电子工业出版社. 2018

[9] [美]伯克斯、赛欧司编 著，张晓坤 译.NET 本质论-第 1 卷：公共语言运行库(中文版) [M]. 中国电力出版社. 2018

[10] [美]格尔 等编著，李敏波 翻译. C＃高级编程(第 4 版) [M]. 清华大学出版社. 2018

[11] [德]Christian Holm, Mike Kruger, Bernhard Spuida 著 薛兴涛 袁勤勇 译. C#软件项目开发全程剖析[M]. 清华大学出版社. 2018

[12] [美]佐尔特 著，杨涛等 译. C#程序设计[M]. 机械工业出版社. 2018

[13] [美]om Archer 等著. C#技术揭秘[M]. 机械工业出版社. 2018

[14] [美]（Yao，Y.）,杜朗（Durant，D.）著. 刘新军 盛泉 李辛鹤 译.NET 精简框架程序设计:C#版[M]. 电子工业出版社. 2018

[15] [美]佩特佐德（Petzold，C.)著，天宏工作室 译. Microsoft C# Windows 程序设计：上下册[M]. 北京大学出版社. 2018

附录 A

Image-based Word Recognition in Oriental Language Document Images

Jason Zhu and Jonathan J. Hull
Center of Excellence for Document Analysis and Recognition
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260 USA hull@cs.buffalo.edu

Abstract

An algorithm for word recognition in Oriental languages such as Chinese, Japanese, and Korean is presented. The objective is to recognize words, that are composed of a number of consecutive characters, in document images where there are no explicit visually defined word boundaries. The technique exploits the redundancy in these languages that is expressed by the difference between the number of possible character strings of a fixed length and the number of legal words of that length.

Sequences of character images are matched simultaneously to lists of legal words and illegal strings that are likely to occur. A word is located if its image is more likely to occur in the current context than any of the illegal strings that are visually similar to it. No intermediate character recognition step is used. The application of contextual information directly to the interpretation of features extracted from the image overcomes noise that could have made isolated character recognition impmsible and the location of words with conventional postprocessing algorithms diffcult. Experimental results are presented that show the ability of this algorithm to correctly recognize text in the presence of noise.

1. Introduction

The recognition of words in Oriental languages such as Chinese. Japanese,

and Korean is an important part of extracting information about the content of the text. The identities of words provide clues about the topic of the text that are useful in translating from the source language to a target language such as English. The ability to recognize words in degraded images such as facsimile transmissions or poor photocopies will also provide an improved transcription of those image to an ASCII-like representation such as BIG-5 or JIS and will.

1051-4651/94 $04.OO 0 1994 IEEE

also improve the ability to automatically translate those documents.

The recognition of words in Oriental languages is a diffcult problem since words are composed of one or more characters and a running text contains no visual indication of word boundaries. Word recognition in Oriental languages has traditionally been solved by postprocessing the results of isolated character recogni tion [4, 6]. Typically, the top N best decisions about the identity of each character in a running text are matched to a list of legal words. A word decision is output if it has a high confidence in comparison to other legal words.

Word recognition techniques take advantage of the redundancy present in Oriental languages. For example, in Chinese there are approximately 3500 commonly used characters. However, out of the 35002 z 12 million posSible stings of two-character words, only about 259000 are legal words. Since legal words make up such a small percentage of all possible character strings, their occurrence in the output of a recognizer is a strong indication that they are correct.

A problem with postprocessing methods is that they requiæ most if not all of the characters in a word to be recognized correctly in the top-N decisions output by the character recognizer. This assumption can be diffcult to guarantee when the input image is subject to degradations that occur in facsimile transmissions or photocopies. This problem has been addressed in English word recognition by matching dictionary entries directly to image data

An algorithm is proposed in this paper for Oriental language word

recognition that overcomes the limitations of postprocessing techniques. The feature vectors extracted from sequences of individual characters are matched directly to the

sequences of features for words in lists (f legal and illegal words. Each enuy in the legal word list points to entries in the list of legal words that are visually similar. A word decision is output if the legal W(Td is significantly more similar to the input than to any of the associated illegal words.

The advantages of this technique include its application of word-level contextual information directly to the interpretation of the feature vectors. The direct use of image data allows for the recognition of words in context even though their individual characters may be unrecognizable in isolation. This overcomes a problem in traditional postprocessing techniques that require reliable character recognition results.

The rest of this paper presents the algorithm in more detail. The model of legal and illegal words is explained and an example is given of how it is used in practice. Experimental results are presented on a database of Chinese text that illustrates the ability of the algorithm to overcome noise that would be 'diffcult to compensate for in a postprocessing approach.

2. Algorithm Description

The implementation of the prop(ßed algorithm for word recognition is described in Figure 1. The input is composed of a stream of characters as they would appear, for example, in a Chinese language book. At each character position, all the words of each length are matched to the image data. Words that contain two to eight characters are used here.

Features extracted from consecutive characters in the image are matched directly to the feature vectors for the characters that compose words in the dicdonary. In Figure 1 the dictionary words are the centers of the clusters. The other entries in the clusters are the confusion set for that word.

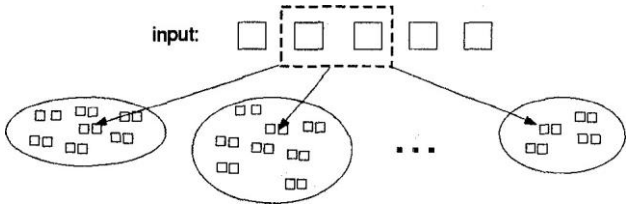

Figure 1. Implementation of word recognition.

If the similarity between the consecuüve characters in the input image and the characters that compose any word is above a threshold, the input characters are compared to the other entries in the confusion set for that word. A word decision is a plausible output only if the similarity between it and the input is less than the similarity between the input and any other word in the confusion set. This process is repeated for all the words in the dictionary and a decision is output only if a single word is plausible at a given location.

The confusion set is compiled from a large training corpus of running text. Each consecutive string of characters in the corpus is compared to every dictionary word. If a string of characters contains at least one character in common with a dictionary word and those characters occur consecutively more than a fixed number of ümes in the corpus, they are added to the confusion set. The reasoning behind using the confusion set is that false positives are most likely to happen when the input is visually similar to a dictionary word. The confusion set models the most likely false positives and provides a method to suppress their identification.

A precise statement of the algorithm is given below. A training phase shown in Figure 2 is used in which the confusion sets for each dictionary word are calculated. In step 1 all the character sü•ings of each length from two to six are extracted from a training corpus along with their frequencies. In step 2 the character strings are determined that al? the same length as each dictionary word and have one character in common with it. These strings are added to the confusion list for the dictionary word if the similarity between the character

string and the dictionary word is yeater than öo or it is greater than and the character string occurs more than tconf times in the corpus.

```
1.  for each length I —-2345678
      — extract all the character strings of length I and
        their frequencies from a training corpus;

2.  for each dictionary word dw
      — determine the character strings cw found in step 1 that
        are the same length as dw and have at least one
       • character in common;
      — calculate the similarity between each cw and dw ;
      — add those strings to the confusion list for dw
        with a similarity greater than öo(cw) or with
        a frequency greater than tc0M and a similarity
        greater than öl(cw);
```

Figure 2. Training phase.

```
for each character position p in the input
  { output set = NULL; for each dictionary
  word dw of length I { if (Similarity (I input
  chars, dw) > trecog ,l ) output word = dw ;
                —
        for each word cw in the confusion set of dw if (Similarity (l input
          chars, cw) < Similarity (l input chars, dw ) output word = cw ;
                        —
        if (output word == dw) output set = output set + dw ;
                    —
  } / * end for each dw *
  / if ( I output set l 1)
  { write (output set) i
    p = p + l;
```

In the testing phase shown in Figure 3 each character position p is inspected. Each dictionary word dw of a given length I is compared sequentially to the input sffeam starting at pßition p. Longer words are compared before shorter words. If the similarity between the next I characters from the input sffeam starting at position p and dw is greater than a threshold trecog,l, each word cw in the confusion set of dw is compared to the input. If none of the members of the confusion set match the input better than dw (i.e., the similarity between the input characters and every cw is less than the similarity between the

input and dw), then dw is

added to the output set. If after all the dictionary words have been considered, the output set contains a single word, that word is written out and the next I characters are skipped in the input stream.

The similarity between two sffings of character images is calculated as the average of the distance between their feature vectors. The local sffoke direcüon (LSD) vector is exffacted from the input characters as well as the characters that compose the dictionary words [2, 5]. The LSD divides each character image into a 6x6 gid and assigns each pixel the direction (vertiCal, horizontal, diagonal right, and diagonal left) (f the vector that covers the maximum number of consecuåve black pixels. The number of pixels in each grid cell that are assinde each direction ue output. This provides a feature vector of 144 elements for each isolatecharacter.

The distance calculation assumes that there exists image of each character in the fraining corpus and the dictionary. This image data is extracted from digitized fonts that contain labeled images of isolated characters.

3. Experimental Results

Experiments were conducted to investigate several aspects of the word recognition algorithm. An implementation of the algorithm was consü•ucted that located keywords in a Chinese language document. This image is the introduction from a book on Chinese OCR techniques that was scanned at 300 ppi in binary mode [7]. The document image was degraded to simulate the noise present in poor quality facsimiles or photocopies. The recogniüon performance of the proposed algorithm was compared to that of a convenüonal character recognition algorithm and a traditional postprocessing technique.

The training data for building the confusion tables was composed of two million characters of running text from the Pin-Yin Hanzi (PH) corpus [1] plus three million characters of running text from the China News Digest. The dictionary was composed of the twenty keywords shown in Table 1. Fourteen of

those words occur 48 times in the test image and the other six words do not occur. The length of each keyword is given as well as its English translation. The confusion lists on average contained 16.9 character suings for each word.

The distances between the characters in the test image and the characters in the dictionary words and the confusion sets were calculated as described above. A Kunlun 96x96 pixel Sung font was used as the font ing data.

Noise was introduced into the test image by downsampling it to 200 ppi and 100 ppi and corrupting the 300 ppi and 200 ppi versions with a Gaussian bit-flip model. Each pixel in the test image was processed sequentially. If a random number drawn from a GausSian distribution was greater than one standard deviation from the mean, the color of that pixel was changed from white to black or from black to white. Three levels of noise were used. At the first level, one iteration of noise was added. At second level two iterations of noise were added and at the third level three iterations were applied. At each iteration a unique seed was supplied to the random number generator.

Length of	Fnglish translation	I„ength of	English translation
2	recognition	4	formal language
2	Ginese character	6	automaton
6	computer	4	system electron
4	inforrnation	7	micr(Ncope
4	pattern	7	operating systern
4	Chinese characters	7	programming
4	mathematics	6	design
4	artificial intelligeree	2	percentage
7	I ua e character stud	2	workstation
8		2	chinesc language
			langua e character

Table 1. Keywords in the dictionary.

The experimental results are reported in Table 2. The proposed algorithm for word recognition is compared to a character recognition technique and a postprocessing algorithm. The character recognition algorithm compaled the LSD feature vector extracted from characters in the test image to characters in the font training data. The percentage of characters from the test image that are correctly located as the first choice of the character recognition technique are reported. The percentage correct in the five and ten best choices found by the LSD recognizer are also given.

The postprocessing algorithm was given the N best decisions from the character recognition algorithm. It was assumed that a keyword could be located if all its characters (excluding at most one) were found in the N best decisions. The percentage of keywords correctly located by this criterion are reported. This is a generous over-estimate of the expected performance of a well engineered pßtprocessing technique. However, it provides a reasonable upper bound to compare against the word recognition method.

The performance (f the keyword recognition algorithm is reported as the percentage of keywords in the test image that are correctly located as well as the number of errors that occurred. That is, the number of keywords that were incorrectly recognized. The percentage of keywords in the test image that were not located by this technique are also reported,

The experimental results show that the proposed algorithm for word recognition outperforms the postprocessing algorithm in all cases. Correct rates above 94 percent are achieved in the 300 ppi and 200 ppi images. When the 100 ppi image was processed, the postprocessing algorithm needed ten choices to achieve an 88 percent correct rate while the keyword recognition method found 96.5 percent of the target keywords correctly with a zero percent error rate.

4. Discussion and Conclusions

An algorithm for word recognition in Oriental language documents was presented. This technique compared the feature vectors extracted from sequences of characters directly to the feature vectors for words. A simultaneous comparison to character sffings that are likely to be confused with each dictionary word was used to suppless false positives. This provides a method that effectively uses word level contextual information directly at the image level. The bypassing of any intermediate character recognition step allows for the.

recogniüon of words in the presence of noise that would make successful postprocessing of those decisions versus a dictionary nearly impossible.

Current work on this method includes the development of a preprocessing step that redwes the üme required for dictionary matching. The word recognition algorithm is also being applied to Japanese language documents using a dictionary of several thousand words and a training corpus of over 30 million characters.

Acknowledgment

This work was partially supported by a grant from the California Research Center of the Ricoh Corporation.

References

4. S. Mori, C. Y. Suen and K. Yamamoto, "Historical review of OCR research and development," Proceedings of the IEEE 80, 7 (July, 1992), 1029-1058.
5. S. Naito, K. Komori and S. Mori, "SÜoke density feaülre for handprinted Chinese character recognition," Journal of IECE of Japan PRL 81-32 (1981).
6. K. Seino, Y. Tanabe and K. Sakai, "A linguistic post-processing based on word occurrence probability," in From Pixels to Features Ill: Frontiers in Handwriting Recognition, S. Impedovo and J. C. Simon (editor), Elsevier Science Publishers, Amsterdam, 1992, 191-199.
7. X. Z. Zhang, Chinese recognition techniques, QingHua University, Bejing, China, 1992. (in Chinese).

Table 2.Recognition performance comparison.

附录 B

基于图像的东方语言文档图像识别

Jason Zhu 和 Jonathan J. Hull
文件分析与识别卓越中心
计算机科学系
纽约州立大学水牛城分校
布法罗，纽约 14260 美国 Hull @ C.Bualal.EDU

摘要

提出了一种汉语、日语、韩语等东方语言的词识别算法。目标是识别在文档图像中由多个连续字符组成的单词，其中没有明确的视觉定义的字边界。该技术利用这些语言中的冗余，通过固定长度的可能字符串的数目和该长度的合法单词的数量之间的差异来表示。

字符图像序列同时匹配到可能发生的合法单词和非法字符串的列表。如果其图像比当前任何类似于其视觉上相似的非法字符串更容易发生在当前上下文中，则可以找到一个字。不使用中间字符识别步骤。上下文信息直接应用于从图像中提取的特征的解释克服了可能导致孤立字符识别不可及的噪声，并且用传统的后处理算法难以定位单词。实验结果表明，该算法的能力，正确地识别文本在噪声的存在。

1. 介绍

汉语东方语言中的词语识别。日语和韩语是提取文本内容信息的重要组成部分。单词的身份提供了关于文本主题的线索，这些文本有助于从源语言到目标语言如英语的翻译。在诸如传真传输或不良影印的退化图像中识别单词的能力也将提供这些图像到 ASCII 类表示的改进的转录，例如 BIG-5 或 JIS 和意愿。

1051-4651/94 $04.OO 0 1994 IEEE

汉语东方语言中的词语识别。日语和韩语是提取文本内容信息的重要组成部分。单词的身份提供了关于文本主题的线索，这些文本有助于从源语言到目标语言如英语的翻译。在诸如传真传输或不良影印的退化图像中

识别单词的能力也将提供这些图像到 ASCII 类表示的改进的转录，例如 BIG-5 或 JIS 和意愿。.

词汇识别技术利用了东方语言中存在的冗余。例如，在汉语中大约有 3500 个常用字符。然而，在两个字的 35002 个 Z 1200 万个可能的刺中，

只有大约 259000 个是合法的词。由于法律词汇构成了所有可能字符串中的一小部分，它们在识别器的输出中的出现强烈地表明它们是正确的。

词汇识别技术利用了东方语言中存在的冗余。例如，在汉语中大约有 3500 个常用字符。然而，在两个字的 35002 个 Z 1200 万个可能的刺中，只有大约 259000 个是合法的词。由于法律词汇构成了所有可能字符串中的一小部分，它们在识别器的输出中的出现强烈地表明它们是正确的。

词汇识别技术利用了东方语言中存在的冗余。例如，在汉语中大约有 3500 个常用字符。然而，在两个字的 35002 个 Z 1200 万个可能的刺中，只有大约 259000 个是合法的词。由于法律词汇构成了所有可能字符串中的一小部分，它们在识别器的输出中的出现强烈地表明它们是正确的。

这种技术的优点包括将字级上下文信息直接应用于特征向量的解释。图像数据的直接使用允许在上下文中识别单词，即使它们的单个字符可能是孤立地不可识别的。这克服了传统的后处理技术中需要可靠的字符识别结果的问题。

本文的其余部分更详细地介绍了该算法。解释了法律和非法词汇的模型，并举例说明了它在实践中的应用。实验结果显示在中文文本数据库上，说明了该算法克服噪声的能力，这将是难以在后处理方法中补偿的。

2. 算法描述

在图 1 中描述了用于单词识别的 PRED 算法的实现。输入是由一个字符流组成的，例如它们会出现在中文书中。在每个字符位置，每个长度的所有单词与图像数据匹配。这里包含两到八个字符的单词。

从图像中连续字符中提取的特征直接匹配到在字典中构成单词的字符的特征向量。在图 1 中，字典词是簇的中心。集群中的其他条目是那个单词的混淆集。

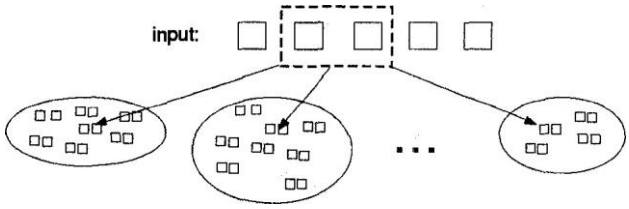

图 1 单词识别的实现

如果输入图像中的连续字符与构成任何单词的字符之间的相似性高于阈值，则将输入字符与该单词的混淆集中的其他条目进行比较。只有当它与输入之间的相似性小于混淆集中的输入和任何其他单词之间的相似性时，单词判定才是合理的输出。对于字典中的所有单词重复这个过程，并且只有在给定位置上单个单词是合理的情况下才输出决定。

混淆集是从一个大型的运行文本训练语料库编译的。语料库中的每个连续

字符串与每个字典字进行比较。如果一串字符包含至少一个与字典字相同的字符，并且这些字符连续地超过语料库中的固定数量的 UMES，则它们被添加到混淆集中。使用混淆集背后的推理是，当输入在视觉上类似于词典词时，最有可能出现假阳性。混乱集模型最有可能的假阳性，并提供了一种方法来抑制他们的身份。

下面给出算法的精确说明。使用图 2 所示的训练阶段计算每个字典单词的混淆集。在步骤 1 中，从训练语料库中提取每个长度从 2 到六的所有字符 S，以及它们的频率。在步骤 2 中，字符串被确定为 AL？每个字典单词的长度相同，并有一个共同的字符。如果字符串和字典词之间的相似性大于 OO，或者字符串大于语料库中的 TCONF 时间，则这些字符串被添加到字典单词的混淆列表中。

在图 3 所示的测试阶段中，检查每个字符位置 P。将给定长度 i 的每个字典字 DW 顺序地与从 P π P 开始的输入 SFFEAM 进行比较。较长的单词在较短的单词之前进行比较。如果从位置 P 和 DW 开始的输入 SFFEAM 的下一个 I 字符之间的相似性大于阈值 TrCOG L，则将 DW 的混淆集中的每个字 CW 与输入进行比较。如果混淆集的成员没有比 DW 更好地匹配输入（即，输入字符和每个 CW 之间的相似度小于输入和 DW 之间的相似度），那么 DW 是

添加到输出集。如果在考虑了所有的字典词之后，输出集包含一个单词，该单词被写入，并且在输入流中跳过下一个 I 字符。

将两个字符图像的相似度计算为它们的特征向量之间的距离的平均值。局部 SFOKO 直接向量（LSD）向量从输入字符以及组成字典词的字符[2, 5]中被导出。LSD 将每个字符图像划分为 6x6GID，并分配每个像素的方向（垂直、水平、对角线右、对角线左）（f）覆盖最大长度的连续像素的向量。每个网格单元中每个方向 UE 输出的像素数。这为每个孤立物提供了 144 个元素的特征向量。T 通过距离计算，假设在词典和词典中存在每个字符的图像。该图像数据是从数字化中提取的。

包含孤立字符的标记图像的字体。

实验结果

实验进行了调查的几个方面的单词识别算法。该算法的实现是在中文文档中定位关键词。这张图片是从一本关于中国 OCR 技术的书中介绍的，它以二进制模式（7）扫描在 300 ppi。文档图像被降级以模拟劣质传真或复印件中存在的噪声。将该算法的性能与传统的字符识别算法和传统的后处理技术进行了比较。

《拼音表》的训练数据由拼音汉字（PH）语料库（1）中的二百万个

汉字组成，加上《中国新闻文摘》三百万行文字。字典由表 1 所示的二十个关键字组成。十四的词在测试图像中出现 48 次，而其他六个词不出现。给出每个关键字的长度以及它的英文翻译。混乱列表平均包含每个单词的 16.9 个字符。

如上所述，计算测试图像中的字符与字典词中的字符和混淆集之间的距离。使用昆仑 96x96 像素 Sung 字体作为字体数据。

噪声被引入到测试图像中，通过将其降采样到 200 ppi 和 100 ppi，并用高斯位翻转模型破坏 300 ppi 和 200 ppi 版本。测试图像中的每个像素依次处理。如果从高斯分布中提取的随机数大于平均值的一个标准偏差，则该像素的颜色从白色变为黑色或从黑色变为白色。使用三个级别的噪声。在第一层，增加了一次噪声的迭代。在第二层添加两次噪声迭代，在第三级迭代三次。在每次迭代中，一个唯一的种子被提供给。

Length of	Fnglish translation	l„ength of	English translation
2	recognition	4	formal language
2	Ginese character	6	automaton
6	computer	4	system electron
4	inforrnation	7	micr(Ncope
4	pattern	7	operating systern
4	Chinese characters	7	programming
4	mathematics	6	design
4	artificial intelligeree	2	percentage
7	l ua e character stud	2	workstation
8		2	chinesc language
			langua e character

表 1 关键词辞典

```
1.  for each length I —-2345678
      — extract all the character strings of length I and
        their frequencies from a training corpus;

2.  for each dictionary word dw
      — determine the character strings cw found in step 1 that
        are the same length as dw and have at least one
       • character in common;
      — calculate the similarity between each cw and dw ;
      — add those strings to the confusion list for dw
        with a similarity greater than öo(cw) or with
        a frequency greater than tc0M and a similarity
        greater than öl(cw);
```

图 2.培训阶段

```
for each character position p in the input
  { output set = NULL; for each dictionary
  word dw of length I { if (Similarity (I input
  chars, dw) > trecog ,l ) output word = dw ;

            —
       for each word cw in the confusion set of dw if (Similarity (l input
         chars, cw) < Similarity (l input chars, dw ) output word = cw ;

                    —
       if (output word == dw) output set = output set + dw ;

                —
  } / * end for each dw *
  / if ( I output set l 1)
  { write (output set) i
    p = p + l;
    break;
```

图 3. 测试阶段

随机数发生器

实验结果报告在表 2 中。所提出的词识别算法与字符识别技术和后处理算法进行了比较。字符识别算法将从测试图像中的字符提取的 LSD 特征向量压缩为字体训练数据中的字符。报告了作为字符识别技术的第一选择的被正确定位的来自测试图像的字符百分比。还给出了 LSD 识别器找到的五和十最佳选择的正确百分比。

后处理算法给出了 N 个最佳决策字符识别算法。假设如果所有的字符（不包括最多）在 N 个最佳决策中找到关键字，则可以找到关键字。该标准正确定位的关键字的百分比报告。这是对一个精心设计的加工工艺的预期性能的过度估计。然而，它提供了一个合理的上限与词识别方法进行比较。

关键字识别算法的性能被报告为被正确定位的测试图像中的关键字的百分比以及所发生的错误的数目。也就是说，不正确识别的关键词的数量。还报道了未被该技术定位的测试图像中关键词的百分比。

实验结果表明，在所有情况下，所提出的词识别算法优于后处理算法。在 300 PPI 和 200 PPI 图像中，正确率达到 94%以上。当处理 100 PPI 图像时，后处理算法需要十个选择以达到 88%的正确率，而关键词识别方法正确地发现目标关键词的 96.5%，错误率为零。

4. 讨论与结论

提出了一种面向东方语言文档的单词识别算法。该技术将从字符序列直接提取的特征向量与单词的特征向量进行比较。同时比较可能与每一个字典字混淆的字符过滤来补充假阳性。这提供了一种有效地直接在图像级使用 Word 级上下文信息的方法。任何中间字符识别步骤的绕过允许。

在噪音存在的情况下识别单词

对这些决策进行成功的后处理

字典几乎是不可能的。

目前的工作方法包括：

一个预处理步骤

字典匹配所需的。单词识别

算法也被应用于日语

使用几千字词典的文件

和一个超过 3000 万个字符的训练语料库。

确认

这项工作部分得到了补助金的支持。

理光公司加利福尼亚研究中心。

参考

4. S. Mori, C. Y. Suen and K. Yamamoto,

OCR 研究的历史回顾与展望

发展，IEEE 80, 7 会议录

(July, 1992), 1029-1058.

5. S. Naito、K. Komori 和 S. Mori 的 S·K 密度

手印汉字

识别，《日本 PRL 81》杂志

32（1981）。

6. K. Seino、Y. Tanabe和 K. Sakai：基于词发生的语言后处理

手写识别领域中的前沿问题，S.

Errdovo 和 J. C. Simon（编辑），爱思唯尔科学出版社，阿姆斯特丹，1992，191-199。

7. X. Z. Zhang，汉语识别技术，清华大学，北京，中国，1992。（中文）。