《物联网与嵌入式技术》

大作业课题报告

文字-盲文转换器之文字识别模块设计

学 生 姓 名：李勇

学科、 专业：智能制造

学　　号：22304177

完 成 日 期：2024/5/6

大连理工大学

Dalian University of Technology

目　　录

1　技术背景.....1
2　技术实现.....3
2.1　技术流程说明.....3
2.2　文字识别工具选择.....3
2.2.1　主流 OCR 库对比..... 3
2.2.2　Tesseract 简介.....5
2.3　服务端 OCR 功能实现..... 6
2.3.1　Tesseract 库安装配置与测试.....6
2.3.2　程序实现..... 10
3　总结与展望.....13

1 技术背景

一套盲文读物相比同样内容的普通读物，价格可贵数倍乃至更多。记者现场看到，一本盲文版的小学语文教材，不但纸张宽大，厚度也堪比一本字典。“盲文图书制作工序复杂、出版周期很长、信息承载量少、且易磨损不耐用。”之江实验室类人感知研究中心工程专员陶文韬说，纸质盲文书依然是国内盲人学习知识和获取信息的主要工具，盲文数字化设备普及率非常低。

相关统计显示，我国视障人士人数达到 1700 多万，其中，800 多万为全盲，占世界失明人口的 20%，每年会出现新盲人大约 45 万。由于多方面原因，我国视障群体不会盲文的现象较为普遍，盲文师资同样短缺。虽然目前文字转成语音的技术较为成熟，但对视障群体来说，盲文依然是他们仰赖的主要阅读工具。随着要求社会更加包容多样的呼声日渐高涨，残障人群的权益，也比以往的到更多关注，为残障人群创造无障碍社会环境的意识正逐渐被重视。

2017 年，财政部、中国残联等多部门联合启动“盲人数字阅读推广工程”，促成 1000 台盲文电脑和盲文电子显示器被配置到全国 100 所盲人教育机构，仍难以满足我国广大视障群体的巨大需求。大多数盲文翻译器都很昂贵，而且仅限于翻译电子文本（即电子设备可以识别的文本）。比如， 一款名为 HumanWare Braillant 的盲文翻译器，连接电脑或移动设备后，才能将文本内容翻译成盲文，其售价高达 2595 美元！也有翻译器的相机只能拍摄到它视图范围内的一张图片，如果让这款设备向与手持扫描仪看齐，这样就能让用户一次扫描到材料的整个页面。

盲文点字显示器。我们知道，盲文一般是纸质型的显示，顶多刻在硬板上；但这是一个机器，在机器上可以显示盲文。以下图中这台为例，是十年前由清华大学研制的，型号是 V3PRO，长 33 厘米，宽大约 9 厘米，厚大约是 2 厘米，就是一个比较薄的长方体。平放在桌面上，点显器下方二分之一的地方，是稍微凹进去的区域，这地方有一排盲文点，对应的每一方点位上方有一个小按键，靠近长方体的正前方还有两边窄边缘，也有一些按键。机器外壳有的是金属的，有的是塑料的。重量不算重，大概是五百克。

图 1.1 点显器

它的功能就是在机器上显示盲文。它有一排盲文显示区域，从左往右，每一方有八个点。这种机器规格有一行十六方的，甚至还有更小的，一行只有八方的；大一点的有一行二十四方的，还有一行四十方的，甚至还有一行八十方的。目前这些规格都是只能显示一行盲文。它上面并排有很多的点位，这些点位就可以通过软件的指挥，根据出现不同的字，让这些点位台起来，凹下去，显示出不同的盲文字，因此它是显示盲文的一种机器。等于八个点或小于八个点的盲文都能显示，你制造一个每方有十个点的盲文，那是不能显示的，因为最多每方是八个点。

它的工作原理，就是通过程序指挥，以四十方为例，它有三百二十个点，这三百二十个点都是一个盲文的点位，每一个点的点位用一个数字来代表，那么我就可以通过程序指挥着机器工作，比如当出现字母 B 的时候，让一二点显示，当出现字母 L 的时候，就让一二三点显示，因为是软件或者说是电子器械，它就可以实现由代码去控制，说白了，它就可以显示任何一种盲文，就是我想让它显示什么样的点位，只要我会改，我会设计，那么都能显示出来。你像现在全球有几十种上百种盲文，都可以通过点显器显示出来，只要你给它做出来对应的盲文表，碰到这个字，让它显示什么样的点位，那么它就可以显示出来，就是它能显示所有的盲文种类。

点显器有传统的，也就是只能连接电脑使用的，必须借助电脑这个平台把电脑上的汉字转换成盲文显示出来的；现在有些新型的机器，包括国内外都有，它就像读书机一样，就是内部有这么一个功能，它可以转换盲文，不需要借助电脑，你只要把内存卡或者 U 盘插上去，如果里面有电子书，那么就能转换出盲文来，咱们就能查看。

但点显器有个缺点就是无法抓取 PDF 等图片文件。

“在知识爆炸且快速迭代的信息时代，盲文数字资源严重缺乏，视障群体的‘数字鸿沟’问题越显突出。”杨文珍表示。目前将物联网与嵌入式技术用于盲文信息无障碍产品还很少。结合我国视障群体的国情，本团队应用通用物联网与嵌入式技术和设备以及盲文翻译算法等，专门研制了一款初步具备基本文字-盲文转换功能的产品。该技术能够提高视障人士的盲文学习效率， 能够输出文字和盲文点位相同内容的信息,为视障人士无障碍学习盲文提供条件。

2 技术实现

2.1 技术流程说明

本团队总体技术流程大体为：分别设置客户端程序和云端服务器程序，用户运行客户端程序，通过鼠标、数位板等外设框选抓取屏幕或利用摄像头等手段获得内容为待识别文字的图片，由于时间精力等不可抗因素，本团队暂只考虑识别英文文本，包含 0-9 数字及标点符号，暂未对汉语言文字转换进行开发，感兴趣的读者可以进一步完善，原理类似；图片文件经过一定处理并通过 TCP 协议传输至云端服务器，由服务端进行图片文字识别和文字-盲文转换，将文字的 ASCII 码转换成相应国际标准下的盲文点数信息，以 6 位二进制数表示；转换后的信息通过串口或 WIFI 模块传输至单片机，由单片机程序控制配置好的舵机机构运动凸起，以向盲文用户传递信息。本人负责文本识别模块的开发。

2.2 文字识别工具选择

2.2.1 主流 OCR 库对比

在接口自动化工作中，经常需要处理文字识别的任务，而 OCR（Optical Character Recognition，光学字符识别）库能够帮助我们将图像中的文字提取出来。本次设计服务端和客户端采用 Python 语言开发，故而只对比支持 Python 环境的 OCR 库进行选择，主要有以下几种。

EasyOCR。EasyOCR 是一个功能强大且开源、易于使用的 OCR 库，适用于各种文字识别任务，包括文档扫描、图像处理、自然语言处理等。它可以帮助开发者快速实现文字识别功能，并应用于各种应用领域。与其他 OCR 库相比，EasyOCR 具有以下特点：多语言支持：EasyOCR 支持超过 80 种语言的文字识别，包括中文、英文、日文、韩文等。它可以处理多种语言混合的文本，适用于全球范围的应用。高精度识别：EasyOCR 使用了深度学习模型和先进的 OCR 技术，能够提供高精度的文字识别结果。它在多个公开数据集上进行了训练和测试，具有较高的准确率和鲁棒性。简单易用：EasyOCR 提供了一个简单的 API，使得文字识别变得容易。只需几行代码，即可将图像中的文字转换为可用的文本。跨平台支持：EasyOCR 可以在多个平台上运行，包括 Windows、Linux 和 Mac OS。它支持 Python 和命令行界面，可以与其他编程语言和工具集成。

使用 EasyOCR 进行文字识别的步骤如下：安装 EasyOCR 库：可以使用 pip 命令安装 EasyOCR 库，例如 pip install easyocr。导入 EasyOCR 库：在 Python 代码中导入 EasyOCR 库，例如 import easyocr。创建 OCR 对象：创建一个 OCR 对象，例如 reader = easyocr.Reader(['en', 'zh'])，指定要识别的语言。识别文字：使用 OCR 对象的 readtext 方法对图像中的文字进行识别，例如 result = reader.readtext('image.jpg')。处理识别结果：根据需要处理识别结果，例如提取文字内容、位置和置信度等。

其缺点是识别速度很慢，不支持训练。

PaddleOCR。PaddleOCR 是一个可以识别图片中文字的工具，可以将图片中的文字转换成电脑可以认识的文字。简单来说，它的原理是使用深度学习技术，通过训练模型来识别图片中的文字。具体来说，它会通过一系列处理，比如缩放、灰度化、去噪等操作，来提高文字识别的准确率。然后，它会使用深度学习模型来检测图片中的文字区域，并将其转换成电脑可以识别的边界框。最后，它会使用另一个深度学习模型来识别边界框中的文字，并将其转换成电脑可以识别的文字。这样，就可以实现将图片中的文字转换成电脑可以识别的文字的功能了。

其特点如下：支持多种 OCR 任务：PaddleOCR 支持多种 OCR 任务，包括文字检测、文字方向检测、多语种 OCR、手写体 OCR 等，可以满足不同场景下的 OCR 需求。识别精度高：PaddleOCR 的深度学习模型经过大量的训练和优化，可以在各种复杂场景下实现高精度的 OCR 识别，具有较高的识别准确率。可准确识别不同字体、字号、字形的文字图像，实现超越人眼识别率的准确率。易于使用：PaddleOCR 提供了丰富的预训练模型和模型优化技术，可以快速部署和使用 OCR 功能，同时也提供了简单易用的 API 接口和开发文档，方便用户进行二次开发和定制化。开源免费：PaddleOCR 是一个开源免费的 OCR 工具，用户可以免费获取源代码和训练数据，自由使用和修改，方便用户进行二次开发和定制化。

但有时会不可控出现部分内容丢失的情况。

CnOCR。CnOCR 是 Python 3 下的文字识别（Optical Character Recognition，简称 OCR）工具包，支持简体中文、繁体中文（部分模型）、英文和数字的常见字符识别，支持竖排文字的识别。自带了 20+个训练好的识别模型，适用于不同应用场景，安装后即可直接使用。同时，CnOCR 也提供简单的训练命令供使用者训练自己的模型。

以下是 CnOCR 的主要特点和功能：中文文本识别：CnOCR 专注于中文文本的识别和提取。它能够处理印刷体中文字符，并能够在各种图像中准确识别和提取中文文本信息。基于深度学习：CnOCR 使用深度神经网络模型进行文本识别。这种模型能够学习和理解字符的特征，并能够对复杂的中文文本进行准确的识别。简单易用：CnOCR 提

供了简单易用的 API 和命令行界面，使用户能够轻松集成和使用该工具。无需复杂的配置和调优，即可进行快速的中文文本识别。高准确率：由于采用了深度学习模型，CnOCR 具有较高的准确率，能够识别出复杂字形和字体的中文字符。快速识别：CnOCR 经过优化，能够在较短的时间内处理图像并进行实时的中文文本识别。这对于需要快速处理大量图像或实时应用的场景非常有用。

其缺点主要是训练比 PaddleOCR 麻烦,极少更新维护。

此外，经济实力允许的开发人员还可以选择付费 API，如百度通用文字识别。基于业界领先的深度学习技术，提供多场景、多语种、高精度的整图文字检测和识别服务，多项 ICDAR 指标居世界第一。对图片中的文字进行检测和识别，支持中、英、法、俄、西、葡、德、意、日、韩、中英混合等 10 种语言，并支持中、英、日、韩四语种的类型检测。提供标准版、高精度版、标准含位置版、高精度含位置版 4 种版本，适应不同业务场景对识别精度、识别速度及位置信息的需求，模型针对图片模糊、倾斜、翻转等情况进行专项优化，鲁棒性强，且支持 2W+大字库，总体识别准确率高，依托百度智能云技术实力，提供高可靠性、弹性可伸缩、高并发承载的文字识别公有云服务，服务可用性高达 99.9%以上。有三种使用方式：公有云服务、离线 SDK、私有化部署。标准版请求 URL 为 https://aip.baidubce.com/rest/2.0/ocr/v1/general_basic。

因此，本团队另选择第 5 种 OCR 工具，即 Tesseract 库。

2.2.2 Tesseract 简介

Tesseract 库最初由惠普实验室于 1985 年开发，后来被 Google 收购并于 2006 年开源。自那时以来，Tesseract 库经历了多个版本的迭代和改进，现在已经成为 OCR 领域的先驱之一。它支持超过 100 种语言，并且在各种操作系统上都能够运行，包括 Windows、Linux 和 Mac OS。Tesseract 有 13 种图片分割模式、4 种引擎模式。

Tesseract 库的核心功能是将输入的图像转换为可编辑的文本。它能够处理各种图像格式，包括 JPEG、PNG 和 TIFF 等。Tesseract 库使用了一种称为“光学字符识别”的算法，该算法通过分析图像中的像素信息来识别和提取文本。它能够识别不同字体、大小和颜色的文本，并且在处理扫描文档或摄影图像时表现出色。

Tesseract 库的使用非常简单，它提供了丰富的 API 和命令行工具。作为一个程序员，可以使用 Tesseract 库的 API 将其集成到应用程序中。如果更喜欢命令行界面，可以使用 Tesseract 库的命令行工具来进行文本识别。无论是想要识别单个图像还是批量处理大量图像，Tesseract 库都能够满足需求。

除了基本的文本识别功能，Tesseract 库还提供了一些高级特性。例如，它支持文本方向检测和自动校正，可以自动识别和修复图像中的文字方向，利用字符白名单和黑名

单功能还可以指定只检测何种字符或不检测何种字符。它还支持多种语言模型，可以根据需要加载不同的语言模型来提高识别准确性。此外，Tesseract 库还支持字典和格式规则，可以用于提高特定领域的文本识别效果。

尽管 Tesseract 库是一个强大的 OCR 引擎，但它并不是完美的。在某些情况下，它可能会出现识别错误或无法处理特定的图像。然而，Tesseract 库具有开源的优势，这意味着可以自己修改和改进它，以满足你的特定需求。

其优点有：准确性：Tesseract 在文本识别方面具备较高的准确性，尤其对于印刷体文字的识别效果较好。它经过多次改进和优化，可以提供可靠的结果。多语言支持：Tesseract 支持超过 100 种不同语言的文本识别，包括中文、英文、日文、法文等。这使得它适用于全球范围内的多语言场景。跨平台支持：Tesseract 可以在多个平台上运行，包括 Windows、Linux、macOS 和 Android。这使得它可以被广泛应用于各种应用程序和系统中。自由开源：Tesseract 是一个开源项目，使用和修改都是免费的。这意味着开发人员可以自由地访问和定制 Tesseract 的代码，以满足特定需求。易于集成：Tesseract 提供了多种编程语言的 API 接口，如 C++、Python、Java 等，使得开发人员可以方便地将 Tesseract 集成到他们的应用程序中。

综上，Tesseract 库能够满足本团队的开发目的。

2.3 服务端 OCR 功能实现

2.3.1 Tesseract 库安装配置与测试

首先安装 Tesseract OCR 引擎：pytesseract 依赖于 Tesseract OCR 引擎。根据官方介绍我们需要知道：有两个部分需要安装，引擎本身和语言的训练数据。语言训练的数据包称为“tesseract-ocr-langcode”和“tesseract-ocr-script-scriptcode”，其中 langcode 是三个字母的语言代码， scriptcode 是四个字母的脚本代码。例如：tesseract-ocr-eng（英语），tesseract-ocr-ara（阿拉伯语），tesseract-ocr-chi-sim（简体中文），tesseract-ocr-script-latn（拉丁字母），tesseract-ocr-script-deva（梵文）等。官方训练模型主要采用的是 LSTM。本次设计我们直接采用官方提供的训练好的数据，能够满足对英文文本的高准确率识别，数据集下载地址：https://tesseract-ocr.github.io/tessdoc/Data-Files，无须重新训练。具体安装和配置步骤遵循安装指导逐步进行即可，在此不作赘述。本项目在 Windows 系统下开发，配置相应的环境变量。其次要在项目环境下安装 pytesseract 包，直接用 pip 命令即可，其包含多种函数和类可供用户调用，另外需要安装一些其他依赖库，如 Pillow、opencv 等，用作图像辅助处理。

按照如下程序进行测试：

```
# import the necessary packages
from PIL import Image
import pytesseract
import argparse
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to input image to be OCR'd")
ap.add_argument("-p", "--preprocess", type=str, default="thresh",
    help="type of preprocessing to be done")
args = vars(ap.parse_args())

# load the example image and convert it to grayscale
image = cv2.imread(args["image"])
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow("Image", gray)

# check to see if we should apply thresholding to preprocess the
# image
if args["preprocess"] == "thresh":
    gray = cv2.threshold(gray, 0, 255,
        cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

# make a check to see if median blurring should be done to remove
# noise
elif args["preprocess"] == "blur":
    gray = cv2.medianBlur(gray, 3)

# write the grayscale image to disk as a temporary file so we can
# apply OCR to it
filename = "{}.png".format(os.getpid())
cv2.imwrite(filename, gray)

# load the image as a PIL/Pillow image, apply OCR, and then delete
# the temporary file
text = pytesseract.image_to_string(Image.open(filename))
os.remove(filename)
print(text)

# show the output images
# cv2.imshow("Image", image)
cv2.imshow("Output", gray)
cv2.waitKey(0)
```

上面的 Python 脚本对输入图像先进行了简单的图像处理，比如模糊和二值化。然后将处理后的图片使用 tesseract 进行文字识别。测试图片为：

图 2.1 测试图片

命令行运行 python ocr.py --image images/example_01.png，经过阈值分割后的图像如下，可以看到把背景阴影很好的去掉了。

图 2.2 分割后图像

命令行输出如下，正确的识别了结果。

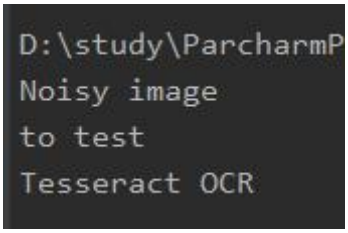

图 2.3 输出结果

2.3.2 程序实现

服务端检查来自客户端发送过来的图片类型和完整性，接收完毕后发送确认信息给客户端；利用 Tesseract 识别图片中文字，将识别结果发送回客户端。

接收并处理图像函数：

```
def handle(self):
        data = b''  # 初始化一个变量来保存数据
        while True:  # 添加循环以处理多个请求
            try:
                # 接收图像数据
                chunk = self.request.recv(1024)
                print("getting data")
                if not chunk:
                    print("getting chunk end")
                    break  # 如果没有更多的数据被接收，就跳出循环
                # 检查是否是心跳信息
                if chunk.decode() == 'HEARTBEAT':
                    print("got heartbeat")
                    self.request.sendall('HEARTBEAT_ACK'.encode())
                    print("sent heartbeat ACK")
                    continue
                # 检查是否是图片数据的标识符
                if chunk.decode() == 'IMAGE_DATA':
                    print("got image data")
                    # 如果是，那么接下来的数据就不需要解码
                    while True:
                        print("start receiving image data")
                        chunk = self.request.recv(1024)
                        print("receiving")
                        if not chunk:
                            print("getting img chunk end")
                            break  # 如果没有更多的数据被接收，就跳出循环
                        # 检查是否是结束标识符
                        if chunk.decode(errors='ignore') == 'IMAGE_END':
                            print("got image end")
                            break
                        data += chunk
                    # 在开始处理图像之前，发送一个确认消息给客户端
                    self.request.sendall('IMAGE_RECEIVED'.encode())
                    print("image received")
```

```
                except Exception as e:
                    print("Error: ", str(e))
                    print(traceback.format_exc())
                    response = {'error': str(e)}
                    self.request.sendall(json.dumps(response).encode())
                    return  # 如果发生错误，返回以退出函数
            # 在接收到所有数据后，处理图像
            try:
                image = Image.open(io.BytesIO(data))
                # 发送确认收到信息给客户端
                self.request.sendall('ACK'.encode())
                print("sent image ACK")
                # 存个 debug 文件
                image.save('debug_image.jpg')
                print('image saved')
                text = pytesseract.image_to_string(image)
                print('got text')
                # 获取文本对应的坐标，暂时没用到
                # coordinates = pytesseract.image_to_boxes(image)
                binary_text = MangWen.ascii_to_braille(text)  # 转盲文
                response = {'text': text, 'binary_text': binary_text}
                print('text:', response['text'], 'binary_text:', response['binary_text'])
                # self.request.sendall(json.dumps(response).encode()) # 发回客户端
                # 调用前面的发送方法
                # self.send_to_microcontroller(response['binary_text'])
            except Exception as e:
                print("Error: ", str(e))
                print(traceback.format_exc())
                response = {'error': str(e)}
                # self.request.sendall(json.dumps(response).encode())
"""
服务器端代码。
监听 5000 端口上的 POST 请求,使用 Flask 接收图像。
当收到一个包含图像的请求时，运行 OCR 代码，并将结果返回给客户端。
当前返回的参数是一个字典，键值包括文本和坐标。
```

```
"""
from PIL import Image
import pytesseract
import io
import os
from flask import Flask, request
import traceback
app = Flask(__name__)
# 设置 tesseract.exe 的路径，保存在.env 文件中
tesseract_path = os.getenv('TESSERACT_PATH')
pytesseract.pytesseract.tesseract_cmd = tesseract_path
# 定义一个路由，用于处理 POST 请求
@app.route('/ocr', methods=['POST'])
def ocr_image_with_coordinates():
    try:
        # 从请求中获取图像文件
        file = request.files['image']
        # 打开图像文件
        image = Image.open(io.BytesIO(file.read()))
        # 保存图像以进行调试
        image.save('debug_image.jpg')
        # 提取图像中的文字
        text = pytesseract.image_to_string(image)
        # 获取每个字符的坐标，即边框信息
        coordinates = pytesseract.image_to_boxes(image)
        # 将提取的文字和坐标返回给客户端
        return {'text': text, 'coordinates': coordinates}
    except Exception as e:
        # 如果出现异常，打印错误信息和堆栈跟踪
        print("Error: ", str(e))
        print(traceback.format_exc())
        return {'error': str(e)}, 500
if __name__ == '__main__':
    # 启动服务器，监听 5000 端口，打开 debug
    app.run(host='0.0.0.0', port=5000, debug=True)
```

3 总结与展望

本设计基本具备了英文文本转换为盲文的功能，但还存在诸多不足，如：未针对汉语言文字进行识别转换；转换效率不够高；利用舵机实现凸起功能有待寻找更优替换方案等。

本设计本着方便盲人阅读的出发点，虽不完美，但却是一次有意义的实践，随着数字化进程发展，盲文乃至整个盲人市场都有着较好的前景，政府也会给予相应政策支持，期待更多的研究人员加入此类研究中，促进技术进步和成熟。