# 低成本手语手套设计与中等词汇量连续手语识别

计算机科学与信息技术

高中组　个人项目

# 摘要

手语是聋哑人与世界交流的桥梁，随着计算机技术的发展与智能化的普及，对手语识别的研究也层出不穷，现今主要的研究方法有基于机器视觉与基于佩带式输入设备两种。相较于基于机器视觉的手语识别，基于佩带式输入设备的手语识别有可以实时得到手指的弯曲、外展情况与手型信息的优点。论文运用了语音识别中的新技术，建立了深度神经网络-隐马尔可夫混合模型（DNN-HMM），设计并制作了低成本低功耗的手语手套，以动态时间规整（DTW）、单高斯隐马尔可夫模型（GMM-HMM）、深度神经网络-隐马尔可夫混合模型（DNN-HMM）作为识别方法，探究最适合中等词汇量连续手语识别的算法。在电脑上完成模型的训练后将模型参数反灌至智能终端，通过编写的译码程序完成实时手语识别。论文研究发现在涉及的模型中，可以使用去除左手弯曲传感器的 17 维数据代替包含左手弯曲传感器的 22 维数据进行模型的训练与识别，不会影响最终的准确率。这样不仅可以降低算法时间复杂度，还可以降低成本；使用深度神经网络代替单高斯模型进行隐马尔可夫模型的观测概率计算可以提高识别准确率，在没有语法时提高效果更为显著，符合手语识别的发展趋势。论文通过设计的训练模型与译码程序，在智能终端上完成了手语的实时识别，识别准确率可以达到 97%以上。

**关键词：低成本；低功耗；DTW；GMM-HMM；DNN-HMM；中等词汇量；连续手语识别**

# 目录

**摘要** ..... 1
第 1 章 引言 ..... 4
**1.1 研究意义** ..... 4
**1.2 手语识别研究概况** ..... 4
**1.2.1 基于佩带式输入设备的手语识别** ..... 4
**1.2.2 基于机器视觉的手语识别** ..... 5
**1.3 论文目的** ..... 6
**1.4 主要创新点** ..... 7
第 2 章 手语手套硬件设计 ..... 8
**2.1 设计综述** ..... 8
**2.2 电阻向电压的转换** ..... 11
**2.3 串口的使用** ..... 14
**2.4Arduino 数据的传送** ..... 15
**2.5 电源的设计** ..... 18
**2.6 功能的验证** ..... 19
第 3 章 数据采集与识别系统设计 ..... 21
**3.1 设计综述** ..... 21
**3.2 数据的接收与存储** ..... 22
**3.3 数据的处理与分析** ..... 24
**3.4 样本录制与显示** ..... 26
**3.5 类与类之间的调用关系** ..... 28
第 4 章 算法的应用与分析 ..... 30
**4.1 算法设计综述** ..... 30
**4.2 动态时间规整的应用** ..... 31
**4.3 隐马尔可夫算法的应用** ..... 32

**4.4 深度神经网络的应用** ..... 39
**4.5 数据分析** ..... 41
第 5 章 实时译码 ..... 43
**5.1 实时译码的意义** ..... 43
**5.2 DTW 的实时译码** ..... 44
**5.3 HMM 的实时译码** ..... 46
**5.4 实时译码小结** ..... 52
第 6 章 结论与展望 ..... 53
**6.1 结论** ..... 53
**6.2 展望** ..... 54
第 7 章 参考文献 ..... 55
第 8 章 致谢 ..... 57

# 第 1 章 引言

## 1.1 研究意义

手语是由手形、手臂运动并辅之以表情、唇动以及其它体势来表达思想的人体语言，具有规范的语法、明确的语义和完备的词汇体系。[1]中国手语主要分为两类：手指语和手势语。手指语是用手指的轨迹描述一个汉语拼音字母，并按照汉语拼音规则构成的语言。手势语是以模拟事物的形状、动作为主要手段，并辅以姿态和表情来表达的语言。手语识别的目标主要是将手指语与手势语转化为文本或者语音一类的自然语言，从而使聋人与听力正常人之间的交流畅通无阻。我国目前有 2000 多万聋人，对手语识别的研究，无疑将直接造福于这个群体，为其提供一种更加自然、更加方便快捷地与健听人交流的途径，以便他们更好地融入社会，这也将对构建多元关爱的和谐社会产生积极影响。

随着计算机技术的高速发展，智能化已经成为发展的主要方向之一，能让机器像人一样思考并作出判断始终是计算机智能化的研究目的。目前，在图像识别及语音识别方面，深度学习技术已经取得了显著的发展。手语手势识别研究与语音识别有着千丝万缕的联系。与语音相似，手语中的手势具有数据量大、时序性强等特点，故可以很好的继承使用语音识别中的前沿研究成果。

## 1.2 手语识别研究概况

研究手语识别现今主要有两种较为成熟的技术：基于佩戴式输入设备的手语识别与基于机器视觉的手语识别。

### 1.2.1 基于佩带式输入设备的手语识别

在基于佩带式输入设备的手语识别方法中，常使用的输入设备主要有数据手套和三维跟踪装置等。数据手套可以实时地得到手指的弯曲、外展和手形变化等方面的信息，三维跟踪装置可以实时给出物体的空间位置。根据这一特点，通过将数据手套与三维跟踪装置结合，借助于适当的算法就能得到手掌和手臂在空间中的位置

和姿态方面的信息。基于佩戴式输入设备的手语识别系统根据被识别的对象进一步可以分为手指语识别、手语词识别、连续手语识别。

手指语作为一种静态的手势词，具有易识别、易尝试新算法的特点，许多早期的研究者都以其为起点进行手语识别的研究。手指语识别的代表性工作有：日本ATR 研究室的高桥（Takahashi）和岸野（Kishino）设计的一个 VPL 数据手套系统，利用关节角度及手的方向编码技术可识别与用户有关的 46 个日本字母中的 34 个。

与手指语相比，手语词的表述是一个动态的时间序列，因此在识别和建模时必须考虑时序性。手语孤立词识别的代表性工作有：弗尔斯（S.S. Fels）和辛顿（G.E. Hinton）使用 VPL 数据手套和 Polhemus 位置跟踪器作为输入设备，神经网络作为手势分类器，根据手的运动轨迹、手形、手的运动方向、手的偏移量以及手的运动速度为特征形成 5 个功能网来识别 203 个手语词。

连续手语识别方面，代表性工作包括：梁容辉（R.H.Liang）和欧阳明（Ouhyoung）利用隐马尔可夫模型实现了连续的台湾手语识别翻译器，该系统针对台湾手语课本里的基本字条及练习句，词汇量为 71-250，手语数据是利用 VPL 公司制作数据手套采集的。中国科学院计算所高文等人[6]采用左右两只 CyberGlove 数据手套作为手势输入设备，配合于三维跟踪装置，开发了一套 CSL(Chinese Sign Language)识别系统，该系统在特定人方面，5100 个词汇的孤立词识别率为 94%左右；在非特定人方面，共采集了 6 个人的数据，词汇同样为 5100 个，平均识别率为 91%。

### 1.2.2 基于机器视觉的手语识别

由于二维视觉图像的遮挡、投影以及光线的影响，使得基于视觉的方法很难精确跟踪到各个手指的弯曲信息。因而，基于视觉的手语识别系统研究工作都集中在小词汇量和中等词汇量之间。该种识别方法的优点是可以同时检测发话者手部的动作和面部的表情，在二维空间中可以在较大范围内实时检测出发话者手部姿态和位置，可以识别体势(如点头、摇头等)以及用户不需要佩戴输入设备即可进行识别。因此基于机器视觉的手语识别系统成本相对较低。

基于机器视觉的手语词识别方面，代表性工作有：凯拉雅凡（C.Charayaphan）和马布尔（A. Marble）使用图像处理方法理解美国手语中 31 个孤立手势词，该方法能正确识别其中的 27 个。杨明玄（M. H. Yang）等人使用运动轨迹在图像序列中提取和分类两维的运动，通过延时神经网络识别 40 个美国手语，识别率达到 98.14%。德国的格罗贝尔（K. Grobel）和埃森（M. Assan）利用隐马尔可夫模型方法识别 262 个孤立荷兰手语词，识别率为 91.3%。该系统的识别方法是基于视觉的，方法是让打手语者带上有色手套，然后通过视频提取二维特征。

连续手语识别方面，代表性工作有：美国麻省理工学院多媒体试验室的斯特纳（T.Starner）等对美国连续手语识别进行的研究。他们采用手形、方向及运动轨迹信息的特征向量作为隐马尔可夫模型的输入来识别美国手语。为了便于跟踪，要求用户戴上有色手套，其中右手手套是黄色，左手是桔黄色。测试是在由 40 个词汇随机组成的短句子上进行，系统的识别率为 91.3%，增加一定的语法约束后，实时识别率为 98%。鲍尔（B. Bauer） 和海因茨（H. Hienz）使用一个彩色摄像机作为输入设备、隐马尔可夫模型作为识别方法去识别由 97 个词汇组成的连续德国手语句子，手语识别率达到 91.7%。

## 1.3 论文目的

尽管前人已经对基于数据手套的手势识别有了较为详尽的研究，但是市面上至今仍然没有一款产品广泛地被聋人所使用。一个可能的原因是数据手套高昂的造价。市面上的数据手套主要是供 VR 使用的，因此价格普遍很高。以主营数据手套的 DataGlove Inc.的产品为例，最便宜的一款数据手套也需要$585.00，约合 3800 元人民币。如果要推广这一技术，能否使用更低成本的数据手套达到相同的识别效果是一个关键所在。

本论文旨在通过设计低成本的手语手套，完成中等词汇量的实时连续手语识别。在手语识别中，小于 100 词的属于小词汇量，100-500 词属于中词汇量，500 词以上属于大词汇量。[1]由于时间的限制，本论文选择的词汇量范围为中词汇量，识别范围限于针对特定人的识别。论文中所识别的语句来自于《中国日常手语会话》一书，该书中手语词共 500 词，符合中词汇量的标准。

## 1.4 主要创新点

本论文的创新点是：

1. 使用简易而较为常见的元器件（蓝牙模块、陀螺仪等）设计并制作出了一副低成本的手语手套，测试证明可以完成数据的采集功能且可以达到较高的识别率。
2. 应用语音识别中的新技术，建立深度神经网络-隐马尔可夫混合模型进行手语识别。
3. 建立了基于动态时间规整、单高斯隐马尔可夫模型、深度神经网络-隐马尔可夫混合模型的训练与实时识别系统，完成了三种不同识别方法在时间复杂度与正确率上的比较。
4. 在智能终端上完成了手语的实时识别。

# 第 2 章 手语手套硬件设计

## 2.1 设计综述

为了实现手语识别的目标，手语手套至少需要采集手指与手势两方面的数据。手指数据用来判断手姿，使用的是弯曲传感器，手势数据用来判断手的运动轨迹，使用的是三轴陀螺仪传感器，为了能够对上述两组数据进行接收与预处理，需要使用单片机；为了将数据以无线的方式传送至智能终端，需要使用蓝牙模块。

采集系统的框图如图 2.1.1

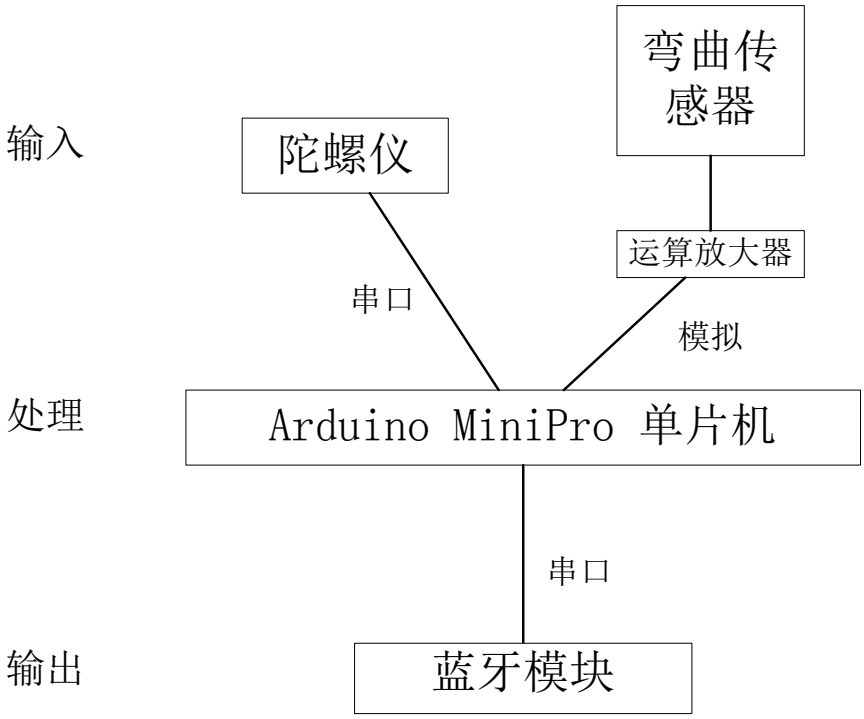


图 2.1.1：系统整体框图

弯曲传感器方面，市面上较为普及的是 Spectra Symbol®的产品，论文使用的尺寸为 2.2"与 4.5"，2.2"用在大拇指与小指上，4.5"用在食指，中指与无名指上。

陀螺仪的选择方面，论文使用的是 MPU6050 三轴陀螺仪传感器，提供的数据类型为加速度，角速度与角度三种。该种陀螺仪的坐标系是时刻以自身为原点的，测量的加速度为重力加速度与外界施加的加速度的合成，因此可以通过加速度在坐标轴上的投影判断运

动的方向，另一方面，通过角速度可以判断运动的过程，两者的数据都不会受到朝向的影响，因而可以较为有效地描绘运动轨迹。陀螺仪发送出的数据是 3 个顺序输出的数据包，每包 11 个字节，分别为加速度、角速度和角度包。数据格式如表 2.1.1、2.1.2、2.1.3。

| 数据编号 | 数据内容 | 含义 |
|---|---|---|
| 0 | 0x55 | 包头 |
| 1 | 0x51 | 标识这个包是加速度包 |
| 2 | AxL | x轴加速度低字节 |
| 3 | AxH | x轴加速度高字节 |
| 4 | AyL | y轴加速度低字节 |
| 5 | AyH | y轴加速度高字节 |
| 6 | AzL | z轴加速度低字节 |
| 7 | AzH | z轴加速度高字节 |
| 8 | TL | 温度低字节 |
| 9 | TH | 温度高字节 |
| 10 | Sum | 校验和 |

表 2.1.1：加速度包数据格式

| 数据编号 | 数据内容 | 含义 |
|---|---|---|
| 0 | 0x55 | 包头 |
| 1 | 0x52 | 标识这个包是角速度包 |
| 2 | wxL | x轴角速度低字节 |
| 3 | wxH | x轴角速度高字节 |
| 4 | wyL | y轴角速度低字节 |
| 5 | wyH | y轴角速度高字节 |
| 6 | wzL | z轴角速度低字节 |
| 7 | wzH | z轴角速度高字节 |
| 8 | TL | 温度低字节 |
| 9 | TH | 温度高字节 |
| 10 | Sum | 校验和 |

表 2.1.2：角速度包数据格式

| 数据编号 | 数据内容 | 含义 |
|---|---|---|
| 0 | 0x55 | 包头 |
| 1 | 0x53 | 标识这个包是角度包 |
| 2 | RollL | x轴角度低字节 |

| | | |
|---|---|---|
| 3 | RollH | x轴角度高字节 |
| 4 | PitchL | y轴角度低字节 |
| 5 | PitchH | y轴角度高字节 |
| 6 | YawL | z轴角度低字节 |
| 7 | YawH | z轴角度高字节 |
| 8 | TL | 温度低字节 |
| 9 | TH | 温度高字节 |
| 10 | Sum | 校验和 |

表 2.1.3：角度包数据格式

在 0-10 号数据中，实际处理中需要用到的只有 2-7 号数据，包头（0-1 号数据）仅用于定位和确定数据类型，而校验和（10 号数据）仅用于检验数据有效性。陀螺仪的尺寸为 15.24*15.24mm，支持串口与 I2C 接口，可以在 115200/9600Baud 下工作，工作电压为 3.3V，工作电流为 10mA，封装为表面贴装。

单片机方面，为了达到便于穿戴的目的，电路板的尺寸需要较小，尤其整体厚度必须要薄。常用的 Arduino Uno 尺寸较大而且由于存在排针的缘故高度很高，不适合本论文中的电路板设计。在搜索了硬件库后，论文选择了 Arduino 系列尺寸最小且没有排针的一款单片机——Arduino ProMini。为达到低功耗的目的，选择的工作电压为 3.3V 版本，主处理器为 ATmega328P，时钟频率 8MHz，尺寸为 33*18mm，耗电约为 3.2mA。由于封装上没有采用邮票孔封装，无法直接焊接在电路板上。因此在实际制作时将单片机的外端做了切割处理，将圆孔切为了邮票孔以便于焊接。Arduino ProMini 有 8 个模拟输入口与 1 对串口收发端，可以正常接收弯曲传感器的数据，但是在串口通信上必须另辟蹊径，否则无法满足陀螺仪与蓝牙模块对于串口的需求。

蓝牙模块方面，为保证低功耗，论文选用的是 BLE4.0 模块，尺寸为 15.1*11.2mm，工作电压为 3.3V，工作电流<6mA，封装为表面贴装。使用蓝牙数据通道传送时，每个数据包最大载荷为 20 字节。

## 2.2 电阻向电压的转换

弯曲传感器相当于可随弯曲度改变阻值的电阻，而由于使用的处理核心为单片机，故需要将电阻值转化为电压值接入模拟口进行测量。弯曲传感器的变化范围为 7kΩ-20kΩ（4.5″）与 20kΩ-40kΩ（2.2″），常用的转换方法有两种，使用简单分压电路转换或使用运算放大器转换。

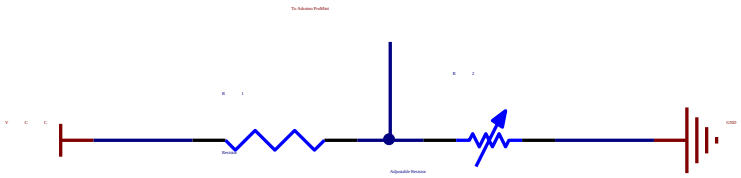

图 2.2.1：简单分压电路

简单分压电路如图 2.2.1：

记弯曲传感器两端的电压为$U_2$，则

$$U_{2max} = \frac{V_{cc}}{R_1 + R_{2max}} \times R_{2max} < V_{cc} \ (1)$$

$$U_{2min} = \frac{V_{cc}}{R_1 + R_{2min}} \times R_{2min} > 0 \ (2)$$

由式(1),(2)可知在使用简单分压电路时，弯曲传感器上的分压变化区间小，无法达到电源和地。

运算放大器电路如图 2.2.2：

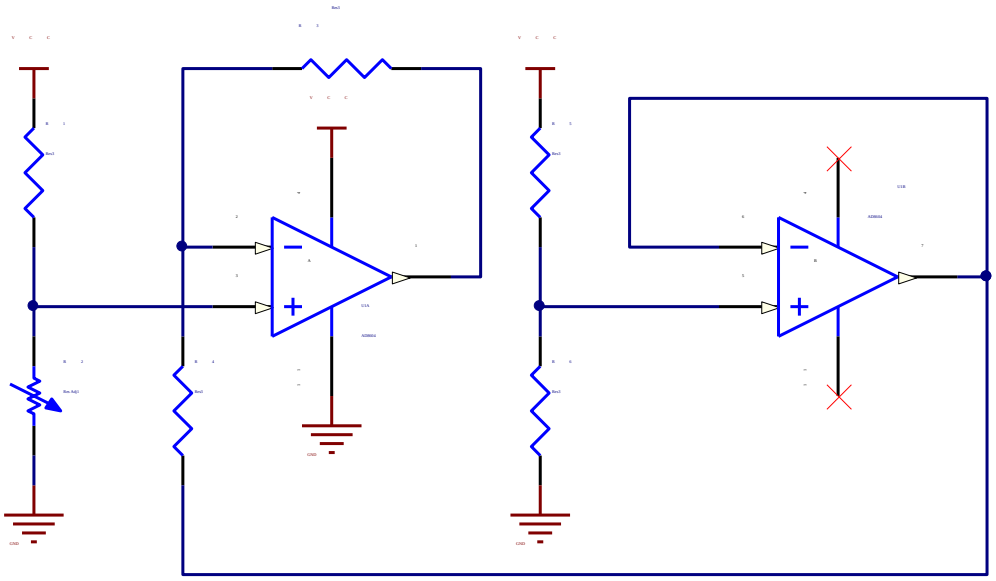

图 2.2.2：运算放大器电路

该电路中，运算放大器的 1 脚接单片机的模拟输入口。记运算放大器各引脚处相对于地的电压为$U_1$、$U_2$、$U_3$……$U_7$，电阻两端电压为$U_{R1}$、$U_{R2}$、$U_{R3}$……$U_{R6}$。利用运算放大器的虚短与虚断特性，可得

$$U_7 = U_6 = U_5 = \frac{V_{cc}}{R_5 + R_6} \times R_6 \quad (3)$$

$$U_2 = U_3 = \frac{V_{cc}}{R_1 + R_2} \times R_2 \quad (4)$$

$$U_{R4} = U_7 - U_2 = V_{cc} \times \left(\frac{R_6}{R_5 + R_6} - \frac{R_2}{R_1 + R_2}\right) \quad (5)$$

$$\mathrm{I} = \frac{U_{R4}}{R_4} = \frac{V_{cc} \times \left(\frac{R_6}{R_5 + R_6} - \frac{R_2}{R_1 + R_2}\right)}{R_4} \quad (6)$$

$$U_1 = U_2 + I \times R_3 = V_{cc} \times \left[\frac{R_3}{R_4} \times \frac{R_6}{R_5 + R_6} + \left(1 - \frac{R_3}{R_4}\right) \times \frac{R_2}{R_1 + R_2}\right] \quad (7)$$

取$R_5 = R_6$，$R_3 = 3R_4$, $R_1 = R_{2max}$，由于$R_{2min} = \frac{1}{2}R_{2max}$，可知当$R_2$取最小值时，$U_1 = 0$，当$R_2$取最大值时，$U_1 = V_{cc}$。即在$R_2$变化时，输出电压$U_1$的变化区间可以达到电源和地。

简单分压电路的输出电压无法达到电源和地，而运算放大器电路的输出电压可以达到电源和地，说明运算放大器的电路在输出电压的变化区间上优于简单分压电路，因此论文选择了运算放大器电路作为电阻向电压的转换方式。

在运算放大器的选择中，应选择轨到轨的运算放大器。普通运算放大器的输入和输出均不能到达电源电压和地，而轨到轨运放的输入和输出可以到达电源电压和地。例如在本论文中，电源电压为 3.3V，在使用相同电路的情况下，使用 4.2″ 弯曲传感器进行测试。若使用非轨到轨类型的运算放大器 LM324，在弯曲传感器阻值发生变化时，电压的变化为 1.0V-2.5V。而在使用轨到轨的运算放大器 AD8604 时，电压的变化为 0V-3.3V，有明显的变化量提升。论文最终选择了 AD8604 作为运算放大器，在 3.3V 的工作电压下，其电流约为 0.75mA。

## 2.3 串口的使用

论文中使用三轴陀螺仪收集手势变化，蓝牙模块将数据传至上位机，两个模块均需要通过串口的方式连接至单片机，而由于论文中使用的 Arduino ProMini 为小型单片机，只有一个串口，故无法使用常规方法连接。

论文起初使用的连接方法为串行外设接口（Serial Peripheral Interface, SPI）。将两块单片机分别作为主机（Master）、从机（Slave），主机连接陀螺仪，从机连接蓝牙模块。由于 Arduino ProMini 本身没有 USB 接口，直接调试较为麻烦，故使用拥有 USB 接口的 Arduino Uno 代为调试。Arduino Uno 在处理速度、主频晶振上均高于 Arduino ProMini，在可行性测试方面可以胜任。在使用 Arduino Uno 进行可行性测试时，通过 SPI 的方式连接得到的数据丢包情况严重，可能的原因是 Arduino 本身对协议的支持性不好导致异步通信中时钟频率不统一。鉴于以上原因，论文只能放弃 SPI 连接，改用其他方法。

在分析实际情况后可以发现，陀螺仪和蓝牙模块的串口都分为 Tx（传输）和 Rx（接收）两个接口，但实际上，陀螺仪只需要传送数据，蓝牙模块只需要接收数据，都只用到了串口的一半，因此可以考虑将串口一分为二，将陀螺仪的 Tx 接至 Arduino 的 Rx，将 Arduino 的 Tx 接至蓝牙模块的 Rx（图 2.3.1）。

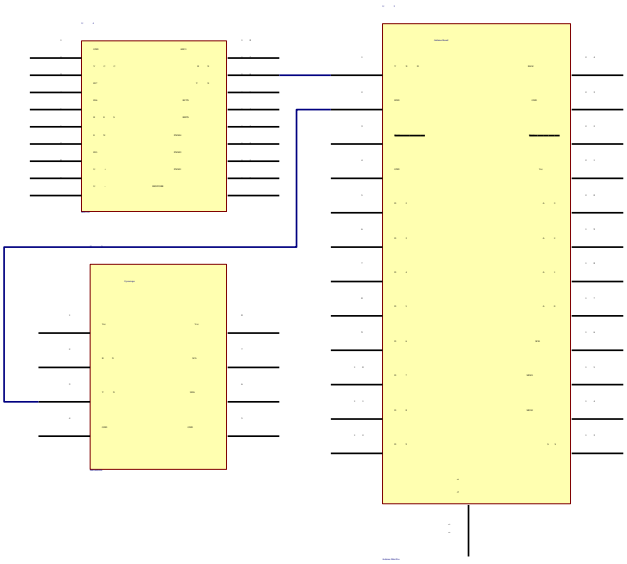

图 2.3.1：陀螺仪与蓝牙模块通过串口独立连接到 Arduino ProMini

使用 Arduino Uno 进行可行性试验发现该种传输方法可以实现数据的正常传输，丢包率为零。试验事实证明了这种连接方式可以简单有效地实现串口的收发独立使用。

## 2.4Arduino 数据的传送

在完成串口使用方式的可行性研究后，使用 Arduino ProMini 进行实际测试。试验中传输速率正常，但收到的数据中仍然存在部分丢失或错位情况。在经过了多次芯片资料的阅读后，发现问题可能是由波特率的误差引起的。

单片机实际发送数据的波特率由其主频与自定义的参数决定，计算公式为 $BAUD = \frac{fosc}{16(UBRRn+1)}$，其中$f_{osc}$是系统的晶振时钟频率，UBRRn 为自定义的 0-4095 的整数。在本论文中，Arduino ProMini 的晶振时钟频率为 8MHz。若使用通常的 115200Baud 进行传输，实际能达到的最接近波特率为$\frac{8\times10^6}{16\times(3+1)} = 125000Baud$，误差为 8.5%。查阅芯片手册后发现，在论文的数据传送状态（8 bit, no parity） 下，允许的最大误差为±4.5%，推荐的最大误差为±2.0%，实际的误差已经远远超出了允许的最大误差，故会发生数据的丢包问题。由于陀螺仪的输出波特率只有 115200Baud 与 9600Baud 两种，考虑选用 9600Baud 这一更低的波特率进行传输以减小误差。当使用 9600Baud 进行传输时，实际能达到的最接近波特率为$\frac{8\times10^6}{16\times(51+1)} = 9615Baud$，误差仅有 0.16%，在推荐的最大误差范围以内，可以保证传输的稳定性。使用新的波特率进行传输后，再次进行丢包率检测。试验数据表明数据传输速率稳定，校验和正确率为 100%，丢包率为零，符合传输要求。

在完成了对于传送方法的研究后，论文设计了 Arduino 上的数据收集与预处理程序。程序框图如图 2.4.1。

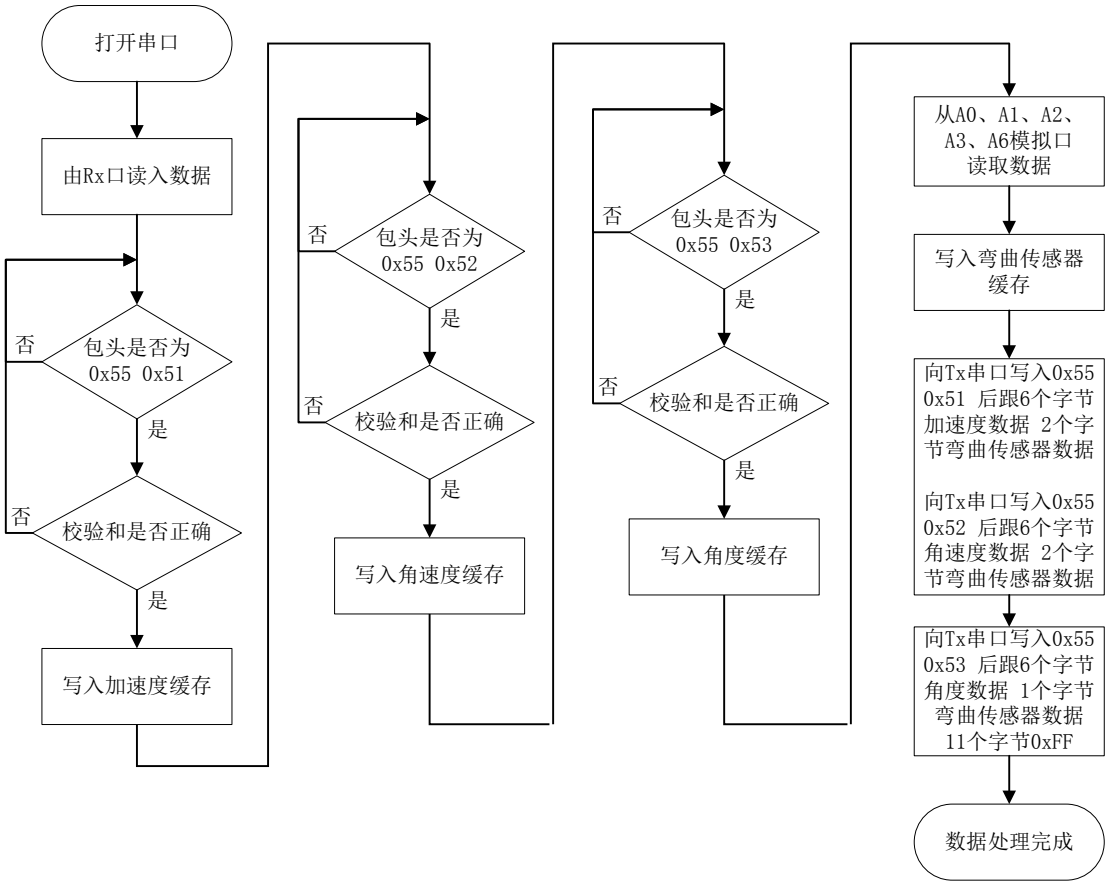


图 2.4.1：数据接收与预处理程序框图

在传送的过程中，程序会先检验每包数据的校验和，由于陀螺仪数据本身已经经过打包处理，为减小数据量，对陀螺仪数据进行解包后作进一步处理。每个弯曲传感器数据为一个字节的十六进制整型。收齐全部三包陀螺仪数据后，将其与弯曲传感器数据打成两包，传至蓝牙模块发送。数据格式如表 2.4.1。

| 数据编号 | 数据内容 | 含义 |
|---|---|---|
| 0 | 0x55 | 包头 |
| 1 | 0x51 | 标识这个包是加速度包 |
| 2 | AxL | x轴加速度低字节 |
| 3 | AxH | x轴加速度高字节 |
| 4 | AyL | y轴加速度低字节 |
| 5 | AyH | y轴加速度高字节 |
| 6 | AzL | z轴加速度低字节 |
| 7 | AzH | z轴加速度高字节 |
| 8 | Flex1 | 弯曲传感器数据1 |

| | | |
|---|---|---|
| 9 | Flex2 | 弯曲传感器数据2 |
| 10 | 0x55 | 包头 |
| 11 | 0x52 | 标识这个包是角速度包 |
| 12 | wxL | x轴角速度低字节 |
| 13 | wxH | x轴角速度高字节 |
| 14 | wyL | y轴角速度低字节 |
| 15 | wyH | y轴角速度高字节 |
| 16 | wzL | z轴角速度低字节 |
| 17 | wzH | z轴角速度高字节 |
| 18 | Flex3 | 弯曲传感器数据3 |
| 19 | Flex4 | 弯曲传感器数据4 |

| 数据编号 | 数据内容 | 含义 |
|---|---|---|
| 0 | 0x55 | 包头 |
| 1 | 0x53 | 标识这个包是角度包 |
| 2 | RollL | x轴角度低字节 |
| 3 | RollH | x轴角度高字节 |
| 4 | PitchL | y轴角度低字节 |
| 5 | PitchH | y轴角度高字节 |
| 6 | YawL | z轴角度低字节 |
| 7 | YawH | z轴角度高字节 |
| 8 | Flex5 | 弯曲传感器数据5 |
| 9-19 | 0xFF | 补位 |

表 2.4.1：蓝牙数据格式

## 2.5 电源的设计

电路中主要的耗电元器件为单片机、蓝牙模块、陀螺仪与运算放大器，根据之前选择的各元器件参数，计算得电源输出电压为 3.3V 时干路电流小于 22.2mA，为保证连续工作 10 小时的设计要求与不超过 30*30mm 的尺寸限制，论文最终选择的电池为 300mAh 的锂电池，尺寸为 30*25mm，标称电压 3.7V。为了更方便地给电池充电，电路板配有一个符合 micro-USB 标准的接口，装于电路板的外端。为了控制电池的充电或是供电，电路板上还装有一个开关。为保证整个系统的可穿戴性，对开关的尺寸要求为高度<1mm，行程<2mm，宽度<4mm，封装为表面贴装。最终选择的标贴开关尺寸为高度 0.7mm，行程 1.4mm，宽度 3.1mm，符合设计要求。

## 2.6 功能的验证

最终制成的电路板尺寸为 36.8*34.3mm，简图如图 2.6.1.

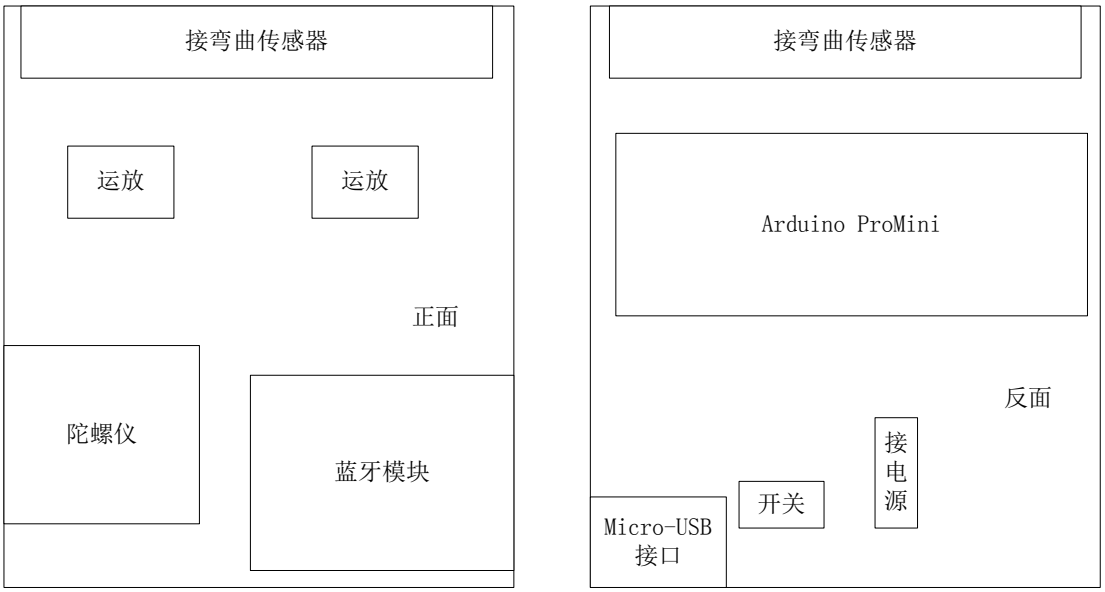


图 2.6.1：电路板简图（左侧为正面，右侧为反面）

每只手套的造价为 660.5 元（BOM 表如图 2.6.2），整个系统（两只手套）造价为 1321 元，远远低于市面上现有的数据手套售价。

# Component list

手语手套

**Source Data From:** 手语手套.SchDoc
**Project:** Free Documents
**Variant:** None

Altium Think it, Design it, Build it™

Report Date: 2016/11/27 15:24:22
Print Date: 04-Feb-17 4:33:06 PM

| # | LibRef | Description | Footprint | Quantity | Unit Price |
|---|---|---|---|---|---|
| 1 | Flex 4.5" | Flex sensor 4.5" | 6-0805_N | 3 | 118 |
| 2 | Flex 2.2" | Flex sensor 2.2" | HDR1X2 | 2 | 92.5 |
| 3 | LM324D | Quadruple Operational Amplifier | D014_N | 2 | 7.5 |
| 4 | Arduino Mini Pro | Arduino Mini Pro | Arduino Mini Pro | 1 | 8.8 |
| 5 | BLE 4.0 | Bluetooth Low energy | BLE4.0 | 1 | 13.9 |
| 6 | Battery | Lithium battery(small) | | 1 | 28 |
| 7 | Charger | USB charger | Micro_USB | 1 | 7 |
| 8 | Charger Port | Header, 5-Pin | Micro_USB | 1 | 0.14 |
| 9 | Gyroscope | Gyroscope Module | Gyroscope | 1 | 47.25 |
| 10 | SW-SPDT | SPDT Subminiature Toggle Switch, Right Angle Mounting, Vertical Actuation | Switch | 1 | 1.48 |
| | | | | 14 | **660.57** |

图 2.6.2：系统 BOM 表

根据设计图搭建电路，测试得干路电流 20.5mA，与计算值的 22.2mA 误差小于 10%，考虑到蓝牙芯片的特性，该误差在可以接受的范围以内。在弯曲传感器弯曲时，大拇指、中指、食指的电压变化区间为 0V-3.3V，小拇指为 0V-3.1V，无名指为 0V-2.5V，基本符合设计要求。无名指的电压变化幅度相对较小可能是由于本身弯曲幅度受限的原因。

在确定数据的收集方面无误后，在 iPad 上编写程序，利用蓝牙协议读取数据并写为文件作进一步处理。之后使用电脑读取这些文件并通过检查校验和确定数据的有效性。在初步采集的 2679 组数据中，校验和正确率为 100%，且在弯曲传感器弯曲时，可以在数据中看到电压值的改变，变化量与使用万用表测量值相同。

使用 iPad 进行编程是因为苹果对蓝牙协议的支持性优于安卓，传输数据更为稳定。其次，苹果的在硬件上的兼容性很好，该程序可以轻而易举地移植到 iPhone 等其他苹果智能终端上。

# 第 3 章 数据采集与识别系统设计

## 3.1 设计综述

系统需要达成的设计目标主要是三方面的：为了能实时监测蓝牙方面的连接与数据情况，需要蓝牙数据接收与存储模块；为了能确定数据的有效性，并保证在手套出现异常（如出现断线情况）时能及时发现，需要数据处理与分析模块；为了方便地录制样本与了解自己所录制的样本内容，需要样本录制与语料标注显示模块。系统的界面图如图 3.1.1。


图 3.1.1：数据采集与识别系统界面图

## 3.2 数据的接收与存储

论文所设计的程序通过 CBCentralManagerDelegate 协议接收蓝牙数据，框图如图 3.2.1。

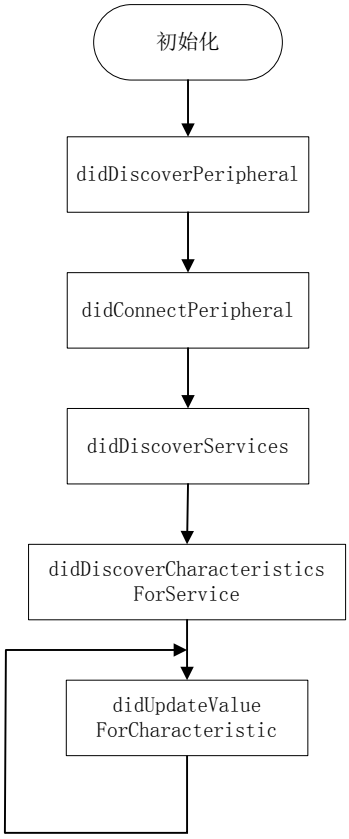


图 3.2.1：蓝牙数据接收框图

当 CBCentralManager 调用任何一个函数时，程序都会更新系统所处的状态并在用户界面上显示。具体流程为：Peripheral Discovered→Peripheral Connected→Services Discovered→Characteristic Discovered。

didDiscoverPeripheral 函数的传出参数包含(RSSI*)RSSI，调用该函数时会记录此时的信号强度值并显示在用户界面。

didUpdateValueForCharacteristic 函数的传出参数包含(CBCharacteristic*)characteristic，即经过封装后被包装成 CBCharacteristic 对象的蓝牙数

据。论文实际使用了数据中的 characteristic.value.description 作为收到的蓝牙数据做后续处理。在每次调用该函数时，都会使帧数+1 并显示在用户界面上。

由于蓝牙与 Arduino ProMini 的通信方式为异步通信，可能会导致 Arduino 传来的一个数据包被蓝牙拆为两包发送的情况，如果不经处理直接将数据储存的话，在后续的识别中会发生数据的丢失与错位，影响整体识别效果。程序中使用了自定义的 DataBuffer 类处理这一情况。

DataBuffer 提供的外部接口为

```
- (void)writeBuffer:(NSString*) incomingStr;
- (NSString*)readBuffer;
```

writeBuffer 函数通过环形数组的结构实现了蓝牙数据的初步寄存，readBuffer 函数通过正则表达式的查找方式在环形数组中提取完整的数据包，并以 NSString*的方式传出。

## 3.3 数据的处理与分析

在实际测试中发现左手的蓝牙模块数据发送速率快于右手，因此在数据接收中可能出现左手已经发送了 3 包数据而右手只发送了 1 包的情况。如果简单地将左右手数据一包对一包地拼接起来，可能会出现左手数据溢出的情况。程序中使用自定义的 MessageBuffer 类处理这一问题。

MessageBuffer 提供的外部接口为

```
- (MessageBuffer*)init;
- (NSString*)getFullMessage:(NSString*)message forHand:(NSString*)hand;
```

init 函数的作用为生成计数器与缓存用的环形数组。getFullMessage 函数的作用为缓存左右手的数据并将其拼接在一起，当左手数据溢出时，不再使用实时的右手数据，而是使用最近一次右手传来的数据进行拼接。拼接后的结果以 NSString*的形式传出。

完成了数据的拼接后，对数据进行解包处理。程序定义了名为 gloveData 的结构体，定义语句如下

```
typedef struct{
    double acceleration[6];
    double angularVelocity[6];
    double angle[6];
    int flex[10];
}GloveData;
```

结构体由四个数组组成，分别存放加速度、角速度、角度和弯曲传感器数值。解包处理在 ViewController 中完成。过程为将 MessageBuffer 处理完成的数据中分离出加速度、角速度、角度和弯曲传感器数据，并利用定义的结构体储存。分离数据后利用给定算法解算数据，得出数据的实际值。

线图的绘制由 DataShowView 类完成。

DataShowView 不提供外部接口，仅包含系统自动调用的 drawRect 函数，其作用为从 gloveData 结构体中调取数据并绘制折线图，显示在用户的界面上，如图 3.3.1。

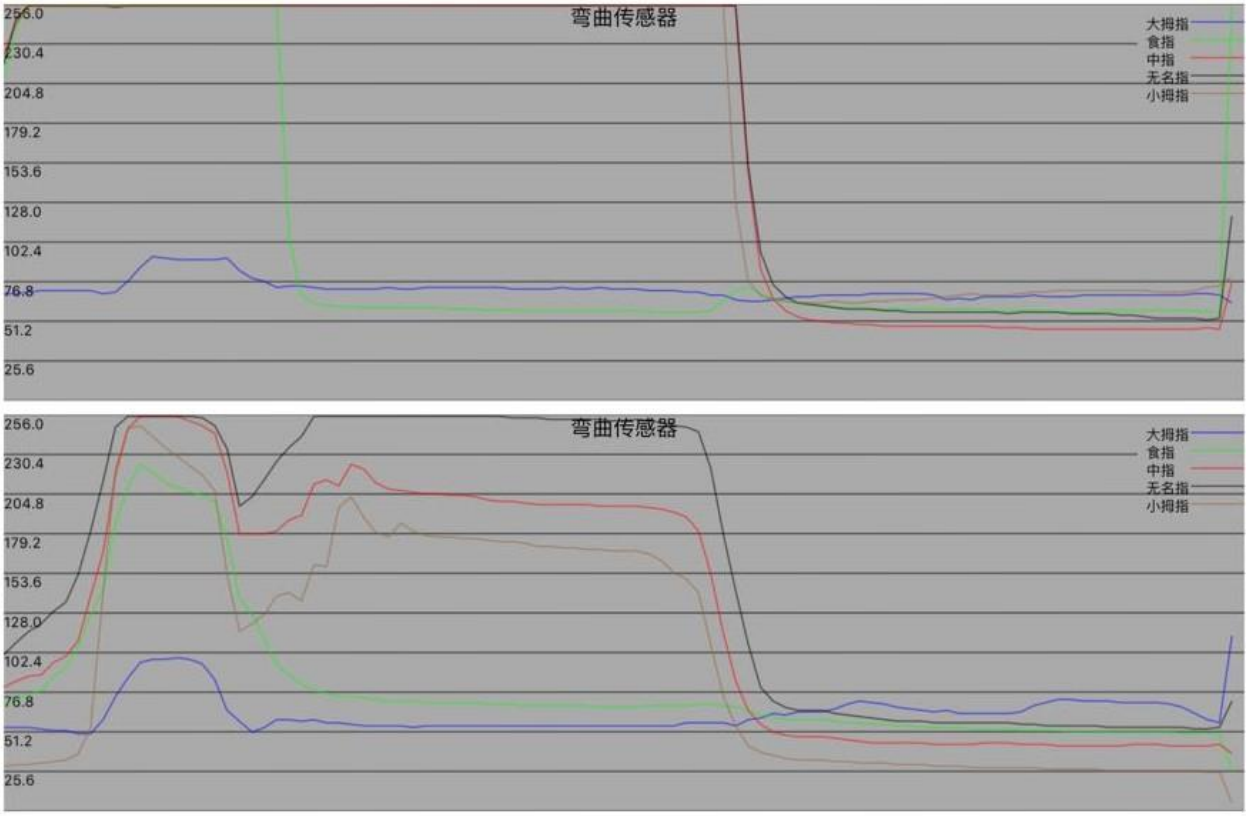


图 3.3.1：弯曲传感器折线图

## 3.4 样本录制与显示

在样本录制的过程中，需要提供的功能包括显示并朗读当前录制的句子，显示/修改重复次数，显示/修改录制的句子序号范围，抹去先前一次动作的错误数据以及判断用户的录制状态。前三种功能的实现函数在 ViewController 中，朗读功能是使用 AVSpeechSynthesizerDelegate 协议实现的。最后两种功能是使用自定义的 ActivityDetection 类实现的。

在程序中判定动作的方法如下：对于弯曲传感器和角速度数据，在静止状态时的值都应接近于零，所以需要判断数据的最大值是否超过阈值以确定是否为动作状态。对于加速度，通过滑动窗口处理数据，窗长为 5。由于在静止状态时有重力加速度 g，因此判断窗口内的数据最大值与最小值的差量是否超过阈值以确定是否为动作状态。如果角速度、弯曲传感器与加速度有任意一者被判定为动作状态，则数据就会被判定为动作数据。

类中通过枚举定义了五种可能的录制状态，分别为 INIT（初始化）,SENTENCE_START（动作开始）,SENTENCE_ONGOING（动作过程）,SENTENCE_END（动作结束）,IDLE（闲置）。状态转移图如图 3.4.1。

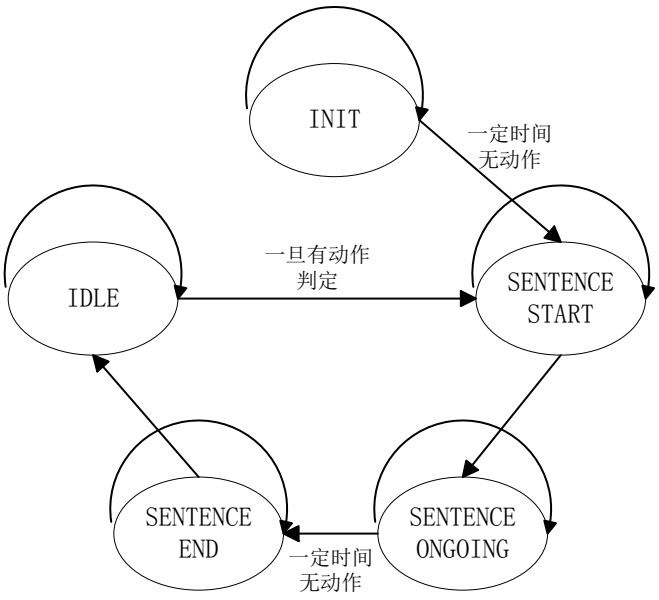


图 3.4.1 录制状态的转移图

INIT 状态后需要持续一定时间无运动才进入 SENTENCE_START 状态，这是因为在单人录制过程中，在按下录制键之后到手回到身体两侧开始录制有一定的时间，为了不记录下这段数据，需要有一定的延迟。而 IDLE 状态下一旦检测到运动状态便会进入 SENTENCE_START 状态，这是为了保证不会漏失数据。在不同的状态下，置于用户界面的方块会显示出不同的颜色以提醒用户。

ActivityDetection 提供的外部接口为

- (ActivityDetection*)init;

- (NSInteger)detectActivity:(GloveData)gloveData;

- (void)resetState;

init 函数的作用为生成各类数据动作判定的阈值，用于初始化。resetState 函数的作用是将状态重置为 INIT，该函数不仅用于初始化，也用在抹去先前一次的错误数据功能上。在录制样本时，由于每个样本需要重复的次数很多，难免会出现出错的情况，当发现出错时再次按下录制键可以抹去之前一次动作的数据，这时会调用 resetState 函数将状态恢复至 INIT。detectActivity 的作用是判定一段数据是否为动作，并根据判定结果更新状态。

## 3.5 类与类之间的调用关系

程序中类与类之间的调用关系如图 3.5.1。

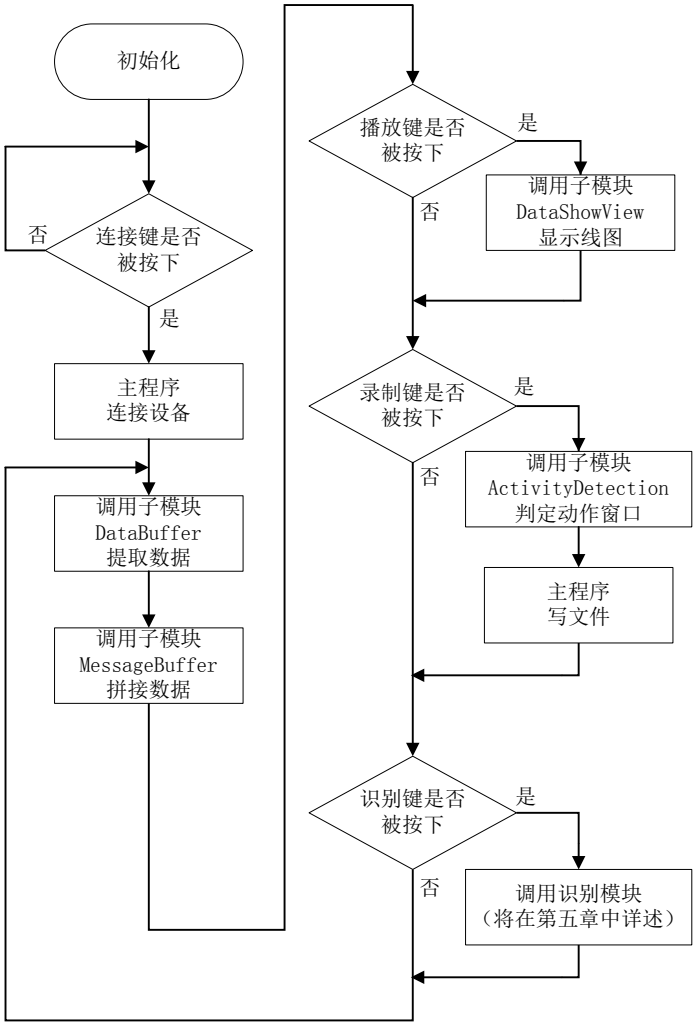


图 3.5.1 类与类之间的调用关系图

写文件时，文件名为 GloveData+4 个字符起始句子序号（包括）+4 个字符终止句子序号（不包括）+4 个字符句子重复次数，为 txt 文件。以第 1 句的录制文件为例，重复次数为 30 次，则文件名如图 3.5.2。

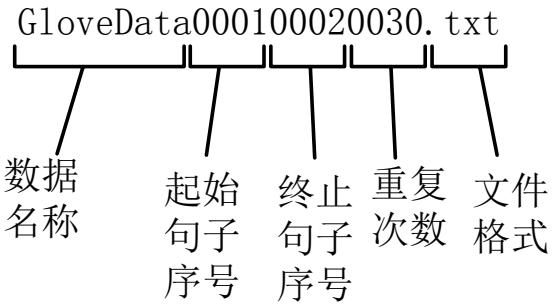


图 3.5.2 文件名示例

文件内容是由左右手数据拼接而成的，每行共 128 个字符，格式如表 3.5.1。

| 数据编号 | 数据内容 | 含义 |
|---|---|---|
| 1-4 | sentenceIndex | 句子序号 |
| 5-8 | repetitionNumber | 句子重复次数 |
| 9-68 | leftData | 左手数据 |
| 69-128 | rightData | 右手数据 |

表 3.5.1 文件数据格式

论文共使用手语手套与采集程序录制了 335 句句子的样本，每句句子重复次数为 30 次。

# 第 4 章 算法的应用与分析

## 4.1 算法设计综述

现今手语识别的算法主要有动态时间规整（下简称 DTW），深度神经网络（下简称 DNN）和隐马尔可夫算法（下简称 HMM）。论文旨在通过比较三种算法之间的复杂度与准确率，寻找最适合中词汇量连续手语识别的算法。

在进行算法的应用前，论文首先对先前采集到的样本数据进行了预处理。原始数据中包含有定位用的包头 0x55 0x51/0x52/0x53，且格式为 txt，不方便直接分析数据，因此在处理中将包头去除，并将剩余数据写为 Htk 指定的 mfc 格式的二进制文件。[11]在之后的数据分析中，先前录制的每句 30 组数据中 85%作为训练数据，15%作为识别测试数据。

除此之外，论文还对《中国手语日常会话》中的所有句子进行了分词处理，即将一句句子中的手语词区分开，根据提取到的手语词制作字典 handDict，key 为手语词名称，对应的 value 为根据 GBT24435-2009 中国手语基本手势得出的手语词编号。由于论文使用的 HMM 库 Htk 不支持汉语，论文还制作了 HMM 中需要使用的词语别名表 wordAlias。

数据维度上，每只手的弯曲传感器数据均为 5 维，陀螺仪数据中加速度 3 维，角速度 3 维，故每只手的数据维度为 11 维，双手的数据总维度为 22 维。

在对手语手势进行了分析后，论文发现在手语中，左手主要作为辅助手使用，其手指的弯曲度在识别过程中可能影响不大。如果能去除左手的弯曲传感器而不影响最终的识别结果，不仅能减少算法复杂度，更能大大降低系统的成本，契合低成本的研究目的。

为了验证这个设想，在分析数据时将提供包含左手弯曲传感器（22 维）的结果与不包含左手弯曲传感器（17 维）的结果，并做出比较。

## 4.2 动态时间规整的应用

动态时间规整（DTW）是基于动态规划（DP）的算法，解决了发音长短不一的模板匹配问题，通过使用满足一定条件的时间规整函数描述测试模板和参考模板的时间对应关系，求解两模板匹配时累计距离最小所对应的规整函数[错误!未找到引用源。]。

由于 DTW 算法中只需要一个模板，故在 25 个训练数据中随机抽取一个作为模板。测试数据与选出的所有模板进行逐一比对，选出对应距离最小的一个模板作为识别结果。在测试中，由于模板选择的随机性，选出的模板可能不是最有代表性的，故正确率会有一定的浮动，论文在 22 维与 17 维中均进行了 5 轮测试，取平均值作为测试结果。

22 维测试中平均正确率为 99.17%，即在 1687 组数据中平均有 14 组出错。17 维测试中平均正确率为 99.20%，即在 1687 组测试数据中平均有 13.5 组出错。分析上述数据可以得出结论，在基于 DTW 的手语识别中，22 维数据与 17 维数据的识别准确率基本相仿，17 维数据的准确率略高出 0.03%，说明在基于 DTW 的手语识别中可以使用 17 维数据代替 22 维数据而不影响识别准确率。

## 4.3 隐马尔可夫算法的应用

隐马尔可夫模型（HMM）是统计模型的一种，用来描述一个含有隐含未知参数的马尔可夫过程。具体算法是通过从可观察的参数中确定该过程的隐含参数，然后利用这些参数作进一步的分析[9]。

论文中使用 Htk 工具包进行单音素与三音素隐马尔可夫模型的训练与识别，训练需要的参数包括训练配置、HMM 模型、模型列表、字典（之前准备好的 wordDict）、训练语料标注以及训练文件列表，识别使用 Viterbi 译码，需要的额外参数是识别语法。

训练配置中与缺省值不同之处如表 4.3.1

| |
|---|
| TARGETKIND = MFCC |
| CEPLIFTER = 22* |
| NUMCEPS = 22* |
| FORCECXTEXP = T |
| ALLOWXWRDEXP = T |

表 4.3.1 训练配置

以上配置声明了训练数据的维度与对三音素识别的支持情况。如果使用 17 维数据，则将*处的数字替换为 17.

论文中对每个手语词进行建模，使用的 HMM 模型为单高斯（GMM）左右模型，简写为 GMM-HMM 模型。每个模型的状态数均为 5，模型名称为手语词对应的编号，据此制作单音素模型列表。例如表 4.3.2

| |
|---|
| G100011 |
| G100034 |
| …… |
| G998 |
| sil |

表 4.3.2 单音素模型列表样例

根据先前制作的词汇别名表 wordAlias 与每个手语词的对应编号制作单音素训练语料标注列表。以“很高兴认识你”这句句子为例,左侧为词语级的语料标注，右侧为音素级的语料标注，如表 4.3.3。

| | |
|---|---|
| SENT-START | sil |
| hen3 | G172 |
| gao1xing4 | G1856 |
| ren4 | G269 |
| shi2 | G209 |
| ni3 | G166 |
| SENT-END | sil |
| . | . |

表 4.3.3 词语级与单音素级语料标注

训练语法为根据 Htk 要求书写的句子列表，如表 4.3.4

```
$sentence-body=
hen3 gao1xing4 ren4shi2 ni3
| hen3 gao1xing4 ren4shi2 ni3 men2
……
| zi4zhu4 you2 hai2 shi4_2 can1jian1 lv3you2 tuan2
;
(SENT-START $sentence-body SENT-END)
```

表 4.3.4 训练语法样例

使用 Htk 提供的软件工具，在 Matlab 中编写脚本进行 HMM 的训练与识别，并输出测试结果。脚本代码如下。

```
system(['HCompV -C ','myConfig -f 0.01 -m -S ','myTrain_Mono.scp -M ','Hmms
','myProto']);
system(['HERest -C ','myConfig -I ','myLabels_Mono -t 250.0 150.0 1000.0 -S
','myTrain_Mono.scp -H ', ...
```

```
    'Hmms/macros_Mono -H ','Hmms/hmmdefs_Mono -M ','Hmms/Mono
','myPhones_Mono']);
    for n=1:4
        system(['HERest -C ','myConfig -I ','myLabels_Mono -t 250.0 150.0 1000.0 -S
','myTrain_Mono.scp -H ', ...
            'Hmms/Mono/macros_Mono -H ','Hmms/Mono/hmmdefs_Mono ','myPhones_Mono']);
    end
    system(['HHEd -H ','Hmms/Mono/macros_Mono -H ','Hmms/Mono/hmmdefs_Mono
','G0.hed ','myPhones_Mono']);
    for n=1:36
        system(['HERest -C ','myConfig -I ','myLabels_Mono -t 250.0 150.0 1000.0 -S
','myTrain_Mono.scp -H ', ...
            'Hmms/Mono/macros_Mono -H ','Hmms/Mono/hmmdefs_Mono ','myPhones_Mono']);
    end
    system(['HVite -H ','Hmms/Mono/macros_Mono -H ','Hmms/Mono/hmmdefs_Mono -S
','myTest_Mono.scp -l * -i ', ...
        'myResult_Mono.mlf -w ','myWordNet_Mono -p 0.0 -s 5.0 ','myDict_Mono
','myPhones_Mono']);
    system(['HResults -I ','myTestLabels_Mono ','myPhones_Mono ','myResult_Mono.mlf']);
    system(['HVite -H ','Hmms/Mono/macros_Mono -H ','Hmms/Mono/hmmdefs_Mono -S
','myTest_Mono.scp -l * -i ', ...
        'myResult_MonoNoContext.mlf -w ','myWordNet_MonoNoContext -p 0.0 -s 5.0
','myDict_Mono ','myPhones_Mono']);
    system(['HResults -I ','myTestLabels_Mono ','myPhones_Mono
','myResult_MonoNoContext.mlf']);
```

22 维单音素测试结果如表 4.3.5（无语法）和表 4.3.6（有语法）

```
====================== HTK Results Analysis =======================
  Date: Tue Jan 31 14:43:25 2017
  Ref : D:/gestureRecognize22/myTestLabels_Mono
  Rec : D:/gestureRecognize22/myResult_MonoNoContextFix.mlf
---------------------- Overall Results -------------------------
```

```
SENT: %Correct=24.36 [H=411, S=1276, N=1687]
WORD: %Corr=84.27, Acc=69.89 [H=9241, D=252, S=1473, I=1577, N=10966]
================================================================
```

表 4.3.5 22 维单音素测试结果（无语法）

```
====================== HTK Results Analysis =======================
  Date: Tue Jan 31 13:35:30 2017
  Ref : D:/gestureRecognize22/myTestLabels_Mono
  Rec : D:/gestureRecognize22/myResult_Mono.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=97.93 [H=1652, S=35, N=1687]
WORD: %Corr=99.73, Acc=99.64 [H=10936, D=15, S=15, I=9, N=10966]
================================================================
```

表 4.3.6 22 维单音素测试结果（有语法）

17 维单音素测试结果如表 4.3.7（无语法）和表 4.3.8（有语法）

```
====================== HTK Results Analysis =======================
  Date: Tue Jan 31 12:18:18 2017
  Ref : D:/gestureRecognize/myTestLabels_Mono
  Rec : D:/gestureRecognize/myResult_MonoNoContextFix.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=32.13 [H=542, S=1145, N=1687]
WORD: %Corr=88.34, Acc=69.65 [H=9672, D=83, S=1193, I=2047, N=10948]
================================================================
```

表 4.3.7 17 维单音素测试结果（无语法）

```
====================== HTK Results Analysis =======================
  Date: Sun Jan 29 13:28:52 2017
  Ref : D:/gestureRecognize/myTestLabels_Mono
  Rec : D:/gestureRecognize/myResult_Mono.mlf
```

| |
|---|
| ---------- Overall Results ---------- |
| SENT: %Correct=97.51 [H=1645, S=42, N=1687] |
| WORD: %Corr=99.63, Acc=99.53 [H=10908, D=9, S=31, I=12, N=10948] |
| ============================================================ |

表 4.3.8 17 维单音素测试结果（有语法）

其中 H 代表正确数，N 代表总测试数，D、S、I 均为错误，D 代表删除错误（例如“很高兴认识你”→“很高兴认你”），S 代表替换错误（例如“很高兴认识你”→“很高兴认识他”），I 代表插入错误（例如“很高兴认识你”→ “很高兴认识你们”），SENT 代表句子正确率，一句句子中只要有一个词语识别有错便判定为错误。WORD 中 Corr 代表不考虑插入错误的词语正确率，Acc 代表考虑插入错误的词语正确率[11]。

可以看出，语法对于 GMM-HMM 的手语识别有很大的影响，主要原因在于识别语法可以减少 Viterbi 译码过程中很多尝试的路径，从而减少了出错的可能性。

在完成单音素的测试后，论文进行了三音素的测试。三音素 HMM 手语识别将一个音素的前一个音素与后一个音素均计入考虑，故与单音素在准备文件的不同主要体现在模型列表与语料标注上。

模型列表格式如表 4.3.9：

| |
|---|
| sil |
| sil-G172+G1856 |
| …… |
| G136-G509+G2131 |
| G509-G2131+sil |

表 4.3.9 三音素模型列表样例

语料标注格式如表 4.3.10：

| |
|---|
| sil |
| sil-G172+G1856 |

```
G172-G1856+G269
G1856-G269+G209
G269-G209+G166
G209-G166+sil
sil
.
```

表 4.3.10 三音素语料标注样例

22 维三音素测试结果如表 4.3.11

```
====================== HTK Results Analysis =======================
  Date: Tue Jan 31 15:00:28 2017
  Ref : D:/gestureRecognize22/myTestLabels_Mono
  Rec : D:/gestureRecognize22/myResult_TriPhone.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=99.70 [H=1682, S=5, N=1687]
WORD: %Corr=99.97, Acc=99.95 [H=10963, D=0, S=3, I=2, N=10966]
================================================================
```

表 4.3.11 22 维三音素测试结果

17 维三音素测试结果如表 4.3.12：

```
====================== HTK Results Analysis =======================
  Date: Sun Jan 29 15:05:57 2017
  Ref : D:/gestureRecognize/myTestLabels_Mono
  Rec : D:/gestureRecognize/myResult_TriPhone.mlf
---------------------- Overall Results -------------------------
SENT: %Correct=99.29 [H=1675, S=12, N=1687]
WORD: %Corr=99.95, Acc=99.86 [H=10942, D=2, S=4, I=9, N=10948]
================================================================
```

表 4.3.12 17 维三音素测试结果

对于三音素 HMM 无法直接做无语法测试，假设单因素模型共有 $n$ 个，则三音素模型的排列组合有$n^3$个，这些组合情况在训练数据中不一定都存在，如果在训练中有一组或多组排列组合不存在，便无法生成译码时需要用到的参数，在译码时便会出错，因此只能通过聚类等方法做近似。因为论文时间有限，故没有进行这部分的研究。

对比表 4.3.6 与表 4.3.8、表 4.3.7 与表 4.3.9、表 4.3.11 与表 4.3.12，可以得出结论，17 维数据与 22 维数据在 GMM-HMM 模型中识别准确率基本相同，在有语法的情况下 17 维数据的准确率略低于 22 维数据 0.43%（单音素）和 0.41%（三音素），可以忽略不计，在单音素无语法的情况下准确率甚至高出 22 维数据 31.90%。因此就 GMM-HMM 模型而言，可以使用更为简单的 17 维数据代替 22 维数据而不影响识别准确率。这样不仅降低了数据复杂度，而且将系统的总造价由 1321 元降低至了 782 元，降低了 40.8%。

## 4.4 深度神经网络的应用

深度神经网络（DNN）是一种从信息处理角度对人脑神经网络进行抽象，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的[12][14]。尽管神经网络方法具有分类特性及抗干扰性，然而由于其处理时间序列的能力不强，因而目前仅应用于静态手势的识别[13]。考虑到 DNN 更善于处理静态问题，论文考虑使用 DNN 代替 GMM（单高斯模型）完成样本的观测概率计算。目前较为常用的 DNN-HMM 组合分为串联式与混合式，由于时间有限，课题选择的是混合式模型。

为使训练获得更好的结果，训练中将原始的 1 个样点扩充为一个由相邻的 9 个样点（紧邻样点的前 4 个样点、样点本身、紧邻样点的后 4 个样点）形成的样本进行训练，对于起始与终止的样点则通过将起始或终止样点的数据重复 4 次的方式将样点数补足。训练的数据数共$190 \times 335 \times 25 = 1591250$ 个，每个数据包含$9 \times 17 = 153$个维度的样本与 1182 个维度的标签。数据量过大，无法利用神经网络一次性训练完成。而 MATLAB 不支持神经网络的分批次训练，Htk 支持神经网络的 3.5 Ver.又尚未发布 windows 版本，故论文最终使用了支持分批次神经网络训练的基于 Python 的 Keras 库进行 DNN 训练，训练用的神经网络是多层感知机（MLP），dropout 率为 10%，采用 3 层配置。第一层与第二层均有 1500 个神经元，激励函数为 relu，第三层有 1182 个神经元，激励函数为 softmax。第三层的神经元数量为所有手语词的状态（state）总数。

由于需要将数据在 Matlab（HMM 训练）、Python（DNN 训练）与 C++（Viterbi 译码）之间传递，需要选用三者皆支持的文件格式储存数据，论文选择的是 h5（hdf5）格式。文件中包括数据的 17 维最大值与最小值，手语词状态总数、训练数据、训练数据标签、测试数据、测试数据标签。测试结果如表 4.4.1（无语法），表 4.4.2（有语法）。

```
====================== HTK Results Analysis =======================
  Date: Sun Jan 29 16:27:51 2017
  Ref : D:/gestureRecognize/myTestLabels_Mono
  Rec : D:/gestureRecognize/viterbiDecodeResultFix
---------------------- Overall Results -------------------------
SENT: %Correct=85.93 [H=1447, S=237, N=1684]
WORD: %Corr=97.39, Acc=95.71 [H=10635, D=77, S=208, I=183, N=10920]
==================================================================
```

表 4.4.1 DNN-HMM 测试结果（无语法）

```
====================== HTK Results Analysis =======================
  Date: Mon Jan 30 10:21:01 2017
  Ref : D:/gestureRecognize/myTestLabels_Mono
  Rec : D:/gestureRecognize/viterbiDecodeResult
---------------------- Overall Results -------------------------
SENT: %Correct=97.92 [H=1649, S=35, N=1684]
WORD: %Corr=99.75, Acc=99.67 [H=10893, D=12, S=15, I=9, N=10920]
==================================================================
```

表 4.4.2 DNN-HMM 测试结果（有语法）

## 4.5 数据分析

在完成了多种算法模型的应用后，将各种模型的识别准确率汇总为图 4.5.1

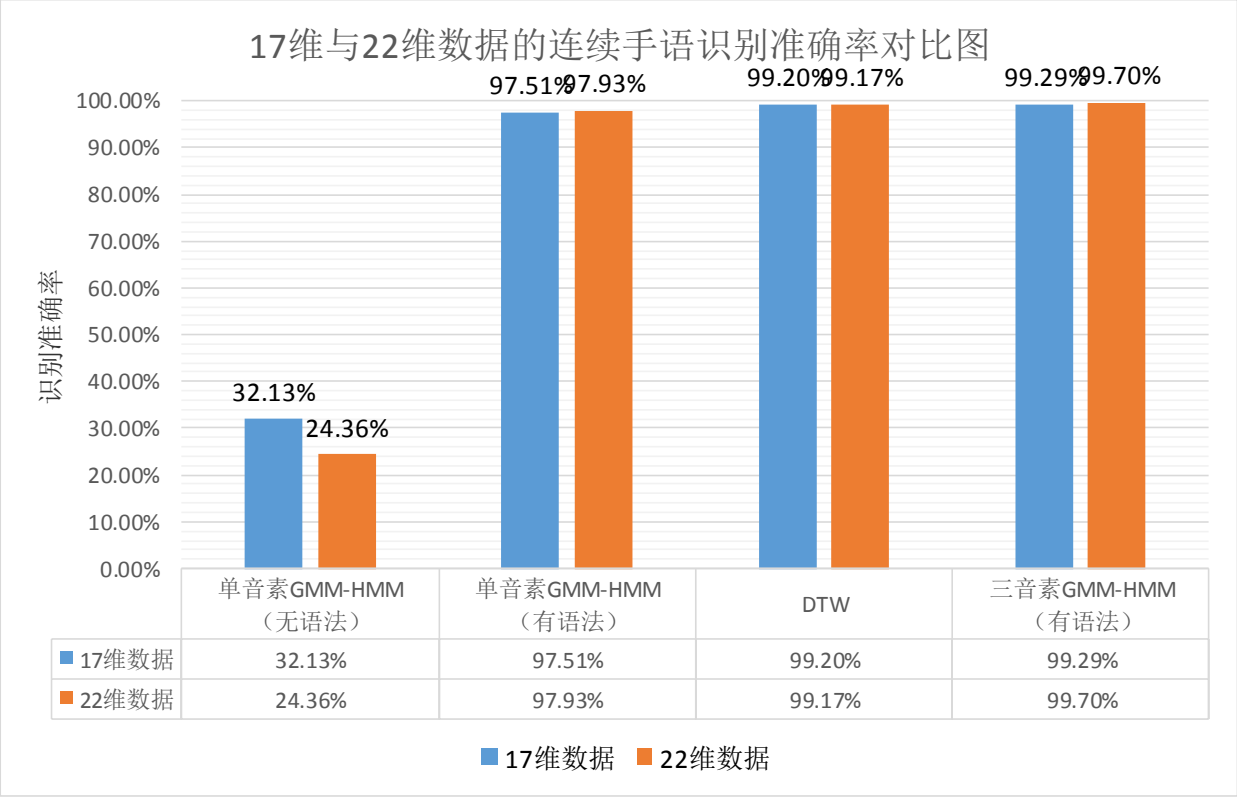


图 4.5.1 17 维与 22 维数据的连续手语识别正确率对比图

由图 4.5.1 可以看出在基于 DTW 与 HMM 的识别中，使用 17 维数据得到的准确率结果均与使用 22 维数据得到结果相仿，最大相差不超过 0.50%，在 DTW 与单音素无语法 HMM 识别中，使用 17 维数据甚至会得到更高的准确率。以上数据说明在本论文所涉及的范围以内，可以使用 17 维数据代替 22 维数据进行模型的训练与识别，不会影响最终的准确率，因此在之后的分析中均使用了 17 维数据。

基于不同模型的连续手语识别准确率比对图如图 4.5.2。

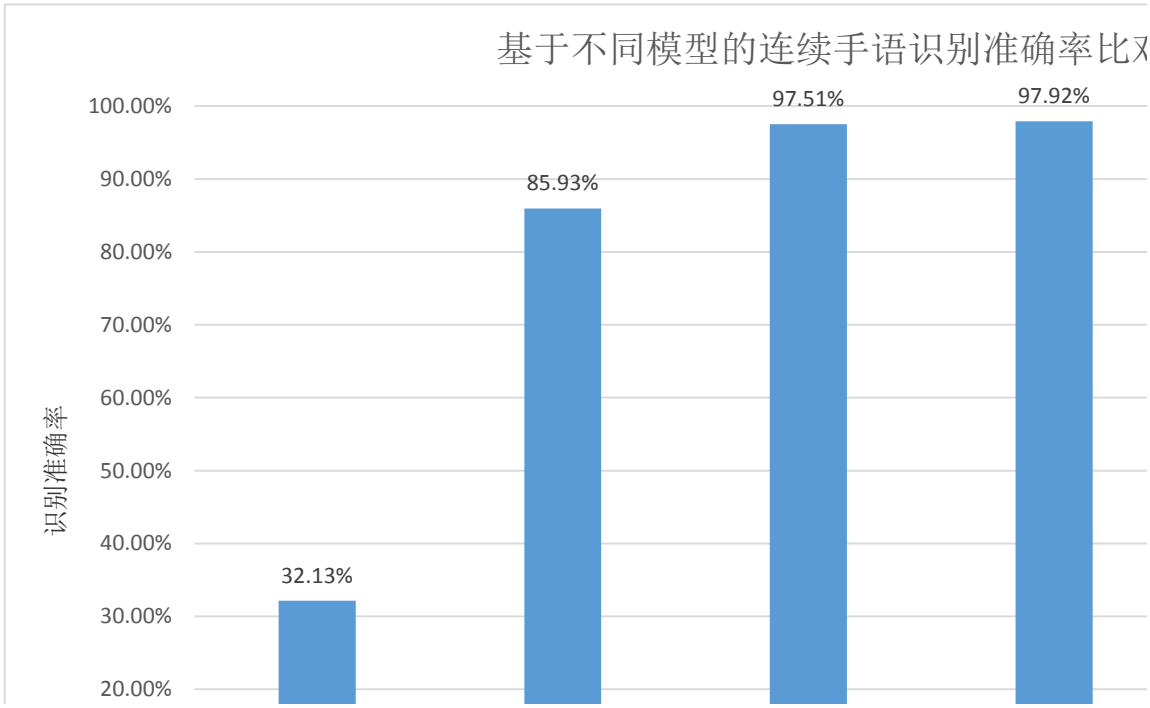


图 4.5.2 基于不同模型的连续手语识别正确率对比图

分析图 4.5.2 可以得出结论，使用 DNN 代替 GMM 计算隐马尔可夫模型的观测概率可以提升识别准确率，在无语法时提升效果更为显著，可以提升最多 167.44%的准确率，在有语法时准确率的提升相对较小，只能提升 0.42%。在有语法时，增加音素的数量可以更为有效地提升识别准确率，比较三音素 GMM-HMM 与单音素 GMM-HMM 识别准确率可以得出，三音素相较于单音素在 GMM-HMM 模型中可以提升 1.82%的准确率。

# 第 5 章 实时译码

## 5.1 实时译码的意义

由于本论文目的为完成实时连续手语识别，在将训练好的模型参数反灌回智能终端后，需要在智能终端上进行实时的译码以达到识别的目的。在实时译码的过程中，论文将着重研究译码算法的复杂度，并通过测试得出实际处理需要的时间作为结果。论文设计的数据发送速率为每秒 20 帧，即 50 毫秒一帧，对于 HMM 而言，实际处理时间必须小于 50 毫秒以确保数据处理的实时性；而对于 DTW 而言，需要收齐一句完整句子的数据才能开始处理数据，因此对于处理时间的要求可以略低一些，设计时间为每句句子处理时间不超过 1 秒。

论文中主要使用了两种译码方式，一种为针对 DTW 的译码，另一种为针对 HMM 的 Viterbi 译码，上文中提到的 DNN-HMM 模型与 GMM-HMM 模型使用的均为 Viterbi 译码，区别仅为读取观测概率时的方法不同，故归为一类进行讨论。

## 5.2 DTW 的实时译码

程序中使用 DTW 类进行 DTW 的实时译码，外部接口如下

-(DTW*)init;

-(int)recognize:(NSMutableArray *)testData;

-(void)dealloc;

Init 函数用于初始化，dealloc 函数用于释放通过 malloc 申请的内存，recognize 函数用于进行实时识别。

DTW 的实时译码框图如图 5.2.1.

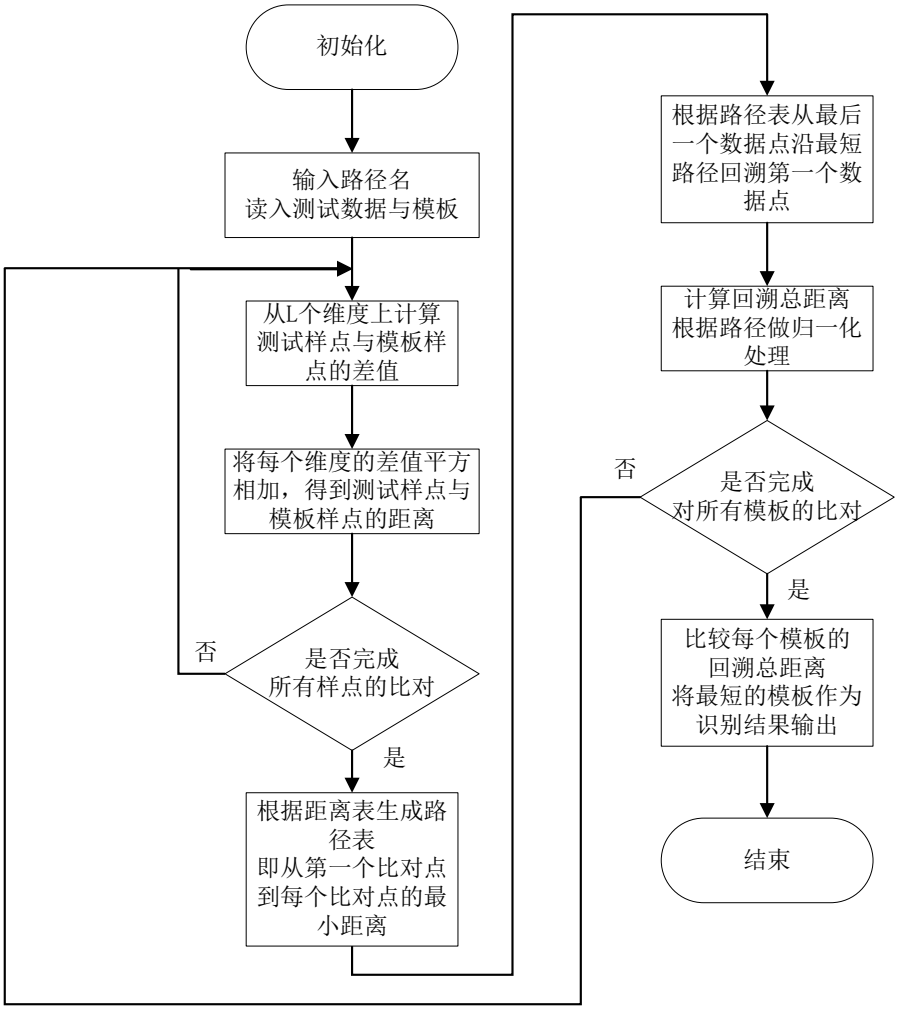


图 5.2.1 DTW 的实时译码程序框图

由框图可以得出，假设模板长度为 n，则译码程序的复杂度具体值为 $3 \times n^2 \times SampleDimension \times TemplateNumber \approx 6 \times 10^8$，其中 3 是每次规整中需要进行的操作数；n 是样本的样点数，平均值为 190；SampleDimension 为数据的维度数量，此处为 17；TemplateNumber 为模板总数，论文中为 335。

实际测试中，处理包含 323287 个样点的 1683 句句子共耗时 1410.091588 秒，平均每句用时 0.837844 秒，每个样点用时 0.004410 秒，即 4.410 毫秒。

## 5.3 HMM 的实时译码

程序中使用 HViterbi 类进行 HMM 的实时译码，外部接口如下

-(HViterbi*)init;

-(void)initLattice:(double *)obs type:(NSUInteger)recognizeType;

-(void)updateLattice:(double*)obs sampleIndex:(NSUInteger)sampleIndex type:(NSUInteger)recognizeType;

-(NSString*)decode:(NSUInteger)sampleNumber;

-(NSMutableArray *)readMFC:(NSString*)fileName;

-(double *)getMaxVector;

-(double *)getMinVector;

-(void)dealloc;

Init 函数用于初始化，dealloc 函数用于释放通过 malloc 申请的内存，readMFC 函数用于从文件中读入数据，getMaxVector 和 getMinVector 用于寻找数据在每个维度上的最大值，归一化后供神经网络使用。

为增快译码速度，论文定义并使用了名为 Lattice（格点）的结构体，结构体中包含格点名称、手语词序号、状态序号、当前格点观测概率、到下一格点的链表（存有可能到达的下一个格点序号与转移矩阵）、上一格点到当前格点最可能的路径编号等内容。定义语句如下:

```
struct latticeNode{
    double logTransP;
    unsigned int latticeIndex;
    struct latticeNode *next;
};
typedef struct latticeNode LatticeNode;
struct latticeList{
    LatticeNode *head;
};
typedef struct latticeList LatticeList;
struct lattice{
```

```
    char *latticeName;
    char *word;
    unsigned int wordIndex;
    unsigned int stateIndex;
    double prob;
    LatticeList *nextLatticeListP;
    bool used;
    bool nextUsed;
    double currentMaxProb;
    int maxLatticeIndex;
};
typedef struct lattice Lattice;
```

论文在此之上还定义了用于存放所有 Lattice 的结构体数组，之后所有的操作均针对于结构体与数组进行。在通过 initLattice 函数初始化 Lattice 结构体后，程序通过 updateLattice 函数计算每一个格点到达下一个格点的概率，之后通过 decode 函数回溯，寻找最佳路径。

译码程序框图如图 5.3.1。

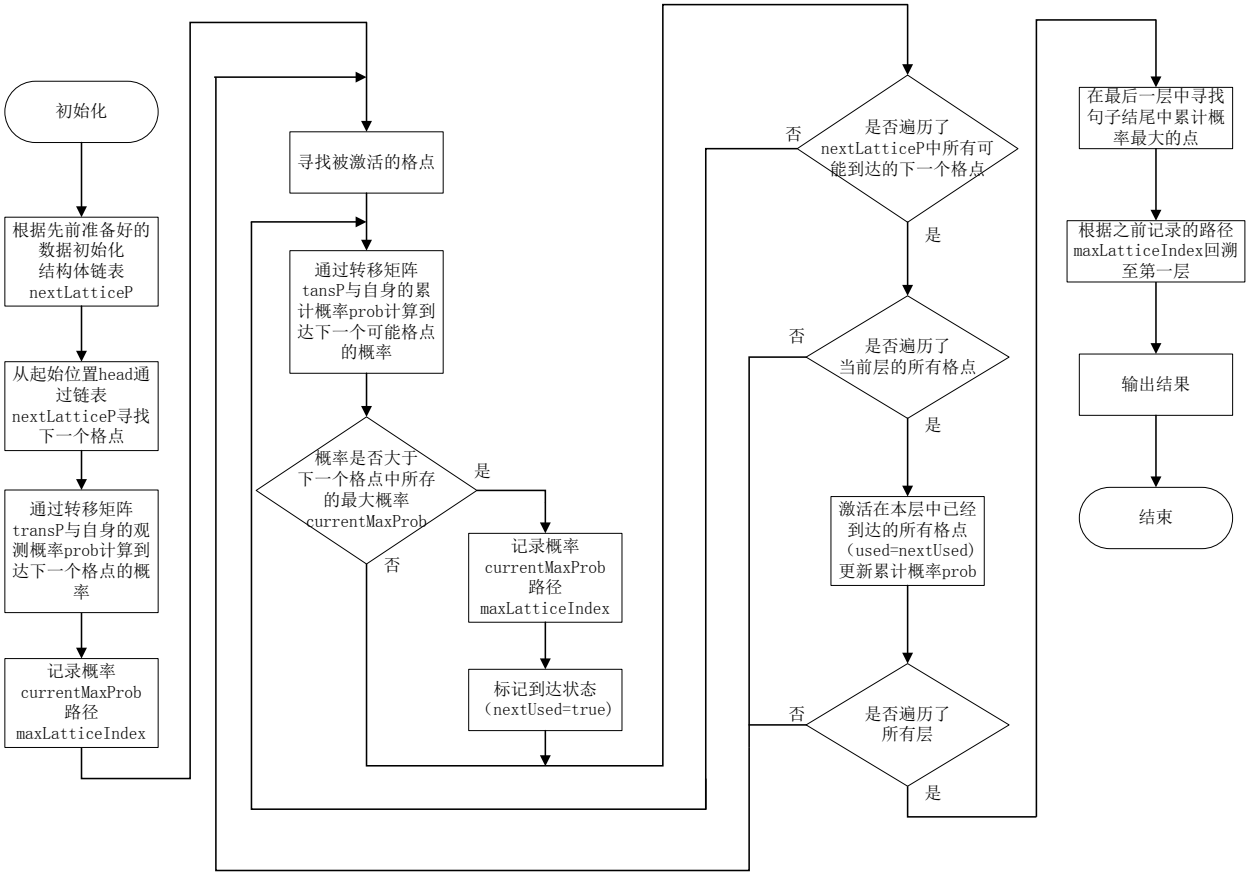


图 5.3.1 HMM 的实时译码程序框图

DNN-HMM 与 GMM-HMM 译码主要的不同点在于当前格点观测概率的计算上。DNN-HMM 的计算方法是根据训练结果中的权重（Weight）与偏置（Bias），在程序中运行 relu 与 softmax 函数，计算出观测概率的具体数值。GMM-HMM 的计算方法是根据单高斯模型得出的平均值（Mean）、方差（Variance）与高斯模型常数（Gconst），利用公式(8)计算出观测概率。

$$N(o;\mu,\Sigma)=\frac{e^{-(o-\mu)'\Sigma^{-1}(o-\mu)}}{\sqrt{(2\pi)^n|\Sigma|}} \quad (8)$$

GMM-HMM 核心代码语句如下

```
double *temp=(double *)malloc(sizeof(double)*_vectorLength);
    double *temp1=(double *)malloc(sizeof(double)*_vectorLength);
    for(int i=0;i<_stateNumber;++i){
```

```
        int meanStartIndex=i*(_vectorLength*2+1);
        int varStartIndex=meanStartIndex+_vectorLength;
        int gConstIndex=varStartIndex+_vectorLength;
        _stateProb[i]=0;
        vDSP_vsubD(data, 1, _stateArray+meanStartIndex, 1, temp, 1, _vectorLength);
        vDSP_vsqD(temp, 1, temp1, 1, _vectorLength);
        vDSP_dotprD(temp1, 1, _stateArray+varStartIndex, 1, _stateProb+i, _vectorLength);
        _stateProb[i]*=-0.5;
        _stateProb[i]-=0.5*_stateArray[gConstIndex];
    }
    free(temp);
    free(temp1);
```

DNN-HMM 核心代码语句如下

```
double *input=data;
    double zero=0;
    for(int i=0;i<_layerNumber-1;++i){
        double *result=(double *)malloc(sizeof(double)*_row[i]);
        double *temp=(double *)malloc(sizeof(double)*_row[i]);
        vDSP_mmulD(_weight[i], 1, input, 1, result, 1, _row[i], 1, _column[i]);
        vDSP_vaddD(result, 1, _bias[i], 1, temp, 1, _row[i]);
        vDSP_vthrD(temp, 1, &zero, result, 1, _row[i]);
        free(temp);
        if(input!=data)
            free(input);
        input=result;
    }
    double *temp=(double *)malloc(sizeof(double)*_row[_layerNumber-1]);
    vDSP_mmulD(_weight[_layerNumber-1], 1, input, 1, _stateProb, 1, _row[_layerNumber-
1], 1, _column[_layerNumber-1]);
    vDSP_vaddD(_stateProb, 1, _bias[_layerNumber-1], 1, temp, 1, _row[_layerNumber-1]);
```

```
        double sumProb=0;
        for(int j=0;j<_row[_layerNumber-1];++j){
            sumProb+=exp(temp[j]);
        }
        sumProb=-log(sumProb);
        vDSP_vsaddD(temp, 1, &sumProb, _stateProb, 1, _row[_layerNumber-1]);
        free(input);
        free(temp);
```

共用的核心代码语句如下

```
    for(int i=0;i<_latticeNumber;++i){
        if(_latticeArray[i].used){
            LatticeNode *nextLatticeP=_latticeArray[i].nextLatticeListP->head;
            while(nextLatticeP){
                Lattice *nextLattice=_latticeArray+nextLatticeP->latticeIndex;
                if(nextLattice->currentMaxProb<_latticeArray[i].prob+nextLatticeP-
>logTransP){
                    nextLattice->nextUsed=true;
                    nextLattice->currentMaxProb=_latticeArray[i].prob+nextLatticeP->logTransP;
                    nextLattice->maxLatticeIndex=i;
                }
                nextLatticeP=nextLatticeP->next;
            }
        }
    }
```

假设模板长度为 n，则 GMM-HMM 译码程序的复杂度为 $n \times (6 \times \frac{List}{Lattice} \times Lattice + StateNumber \times VectorLength \times 3) \approx 2.3 \times 10^7$，其中 6 和 3 分别为更新格点数据与计算观测概率需要的指令数；$\frac{List}{Lattice}$为平均每个格点可以到达的下一个状态数，具体值为$\frac{10831}{5247}$；$Lattice$为总格点数，具体值为 5247；$StateNumber$ 为总状

态数，具体值为 1182；$VectorLength$ 为数据维度，具体值为 17；n 是样本的样点数，平均值为 190。DNN-HMM 译码程序的复杂度主要集中在利用 DNN 训练参数计算观测概率上，共用的核心代码部分复杂度可以忽略不计，具体值为 $n \times (1500 \times 1500 \times 2 + 1182 \times 1500 \times 2) \approx 1.5 \times 10^9$，1500 与 1182 分别为第二层和第三层的神经元数量。

在实际测试中，每处理一帧数据，GMM-HMM 需要的时间最大值为 0.003427 秒，即 3.427 毫秒；DNN-HMM 需要的时间最大值为 0.025529 秒，即 25.529 毫秒，均在规定的 50 毫秒以内，符合设计要求。

## 5.4 实时译码小结

DTW、GMM-HMM、DNN-HMM 的算法复杂度与处理时间对照如表 5.4.1。

| 算法名称 | 算法复杂度（条指令/句） | 处理速度（毫秒/帧） |
|---|---|---|
| DTW | $6 \times 10^8$ | 4.410（平均值） |
| GMM-HMM | $2.3 \times 10^7$ | 3.427 |
| DNN-HMM | $1.5 \times 10^9$ | 25.529 |

图 5.4.1 不同算法的算法复杂度与处理时间对照表

从表中可以看出，GMM-HMM 与 DNN-HMM 算法均可以在规定的 50 毫秒内完成一帧数据的处理，DTW 算法的处理速度也在预期时间之内。三种算法均可以运用于本论文的实时手语识别。

DTW 算法的优势在于需要的训练样本很少且识别准确率较高，缺点在于算法复杂度高，无法实时处理数据，延展性极差。使用 DTW 进行识别的先决条件在于被识别的数据与模板在手势上必须完全一致，否则识别效果便会大打折扣。GMM-HMM 算法的优势在于算法复杂度低，延展性较好，缺点在于训练样本大，在无语法的情况下识别准确率极低（如 4.5 所述）。DNN-HMM 的特点是延展性好，在有语法与无语法的情况下都能做到较好的识别准确率。未来，手语识别的趋势将是更大的词汇量与更少的语法，也正因为这个原因，DNN-HMM 相较于 DTW 与 GMM-HMM 有着更良好的发展前景。

# 第 6 章 结论与展望

## 6.1结论

1. 论文实现了设计并制作低成本低功耗的手语手套的目的，利用常见的低成本元器件，如蓝牙模块、陀螺仪、弯曲传感器完成了数据的采集工作，总成本仅 782 元，远低于市场上销售的同功能数据手套。制作出的手语手套功率较低，保守估计连续工作时间为 10 小时。
2. 在本论文所涉及的模型中，可以使用去除左手弯曲传感器的 17 维数据代替包含左手弯曲传感器的 22 维数据进行模型的训练与识别，不会影响最终的准确率。这样不仅可以降低算法复杂度，还可以降低成本。
3. 使用 DNN-HMM 混合模型代替 GMM-HMM 模型可以有效地增加识别准确率，且在无语法的情况下提高倍数更为显著，符合手语识别的发展趋势，是未来更有发展前景的手语识别方法。
4. 实现了在智能终端上完成实时识别的课题目的，实时译码系统可以在规定时间内完成数据处理，可以保证数据的实时性。

## 6.2 展望

本论文仍存在诸多不足之处，可以在以下方面继续推进：

1. 使用更好的算法。由于设备与时间原因，论文没有研究三音素 DNN-HMM，应该在之后继续推进研究。论文仅仅研究了全连接（MLP）的神经网络，可以运用语音识别中的新技术，如卷积神经网络（CNN）、循环神经网络（RNN）代替 MLP。
2. 深入无语法研究，论文仅仅对无语法单音素 HMM 模型进行了研究，对于三音素的情况，可以使用 SOM 或 kmeans 等聚类方法研究。
3. 改进电路板接口。论文中使用跳线将弯曲传感器连接至电路板上，但是这种连接方式极为不牢固，在测试的过程中经常出现断线的情况。可能的解决方案是寻找一种较小的表贴接插件连接弯曲传感器，这样能使连接更加稳定。

# 第 7 章 参考文献

[1] 张良国，陈熙霖.手语识别研究综述[J].信息技术快报.2009，7(3):28-41

[2] 范会敏，王浩.模式识别方法概述[D].西安.西安工业大学计算机科学与工程学院，2012

[3] 江勇军.基于 Kinect 的孤立词手语识别系统研究[D].合肥.中国科学技术大，2015:9-14

[4] 余晓婷，贺荟中.国内手语研究综述[J].中国特殊教育，2009，4：36-41

[5] 陈振华，余永权，张瑞.模糊模式识别的几种基本模型研究[J].计算机技术与发展，2010，20(9)：32-35

[6] 吴江琴，高文.基于 ANN/HMM 的中国手语识别系统[J].计算机工程与应用，1999，37(9):1-4

[7] 倪训博，赵德斌，姜峰，程丹松. Viterbi 和 DTW 算法的关系分析——在非特定人手语识别中的应用[J].计算机研究与发展，2010，47(2):305-317

[8] 张露.基于 DTW 的单个手语识别算法[J].现代计算机，2016(8):77-80

[9] Sakoe H, Chiba S. Dynamic programming algorithm optimization for spoken word recognition[J]. IEEE Transactions on Acoustics Speech & Signal Processing, 1978, 26(1):43-49.

[10] Rabiner L R. A tutorial on hidden Markov models and selected applications in speech recognition[J]. Proceedings of the IEEE, 1989, 77(2):267-296.

[11] Young S, EvermDNN G, Gales M, et al. The HTK book (for HTK version 3.4)[J]. 2006.

[12] Fels.S.S.,Hinton.G.E.,GloveTalk:A neural network interface between a dataglove and a speech synthesizer[J], IEEE Trans On Neural Network，1993，4(1)：2—8.

[13] 邹伟. 中国手语单手词汇识别方法和技术研究[D]. 中国科学院自动化研究所, 2003.

[14] Michael A. Nielsen. "Neural Networks and Deep Learning" [M].Determination Press. 2014

# 第 8 章 致谢

本论文是在我的指导老师亲切关怀和悉心指导下完成的，他从暑假开始便开始指导我的论文。在论文的构想过程中，他解答了我的很多问题，让我对论文的实现有了信心；在试验方面，他十分关心我的研究进程，努力为我创造最好的条件，教授我仪器的操作方法，让我的试验得以顺利进行。

同时还要感谢这篇论文所涉及的各位学者，本文引用了数位学者的成果，给我的研究过程带来了许多的帮助。如果没有他们的辛勤研究，我们将很难完成这一论文。

最后需要感谢我的家人、同学与朋友，是他们的帮助与鼓励支持着我将论文精益求精，做到最好。

由于学术水平有限，论文写作中不免有许多不足之处，恳请评委老师与专家们修改指正！