基于STM32的孤立词语音识别

这是我毕业设计的论文，当年花了几个月来做，最终算是做出来个基本的功能样机。本来最开始时想做一个图像识别进而实现体感操控，后来考虑到当年用的比较顺手的MCU中功能最强的就是STM32，处理速度和内存容量都难以实现图像识别。于是就换成语音识别，图像识别留作以后再来吧。

OK，废话不多说，上论文：

摘要：语音识别是机器通过识别和理解过程把人类的语音信号转变为相应文本或命令的技术，其根本目的是研究出一种具有听觉功能的机器。本设计研究孤立词语音识别系统及其在STM32嵌入式平台上的实现。识别流程是：预滤波、ADC、分帧、端点检测、预加重、加窗、特征提取、特征匹配。端点检测(VAD)采用短时幅度和短时过零率相结合。检测出有效语音后，根据人耳听觉感知特性,计算每帧语音的Mel频率倒谱系数(MFCC)。然后采用动态时间弯折(DTW)算法与特征模板相匹配,最终输出识别结果。先用Matlab对上述算法进行仿真，经多次试验得出算法中所需各系数的最优值。然后将算法移植到STM32嵌入式平台，移植过程中根据嵌入式平台存储空间相对较小、计算能力也相对较弱的实际情况，对算法进行优化。最终设计并制作出基于STM32的孤立词语音识别系统。

关键词：STM32 孤立词语音识别 VAD MFCC DTW

目录

引 言

从技术上讲，语音识别属于多维模式识别和智能接口的范畴。它是一项集声学、语音学、计算机、信息处理、人工智能等于一身的综合技术，可广泛应用在信息处理、通信和电子系统、自动控制等领域。

国际上对语音识别的研究始于20世纪50年代。由于语音识别本身所固有的难度，人们提出了各种条件下的研究任务，并有此产生了不同的研究领域。这些领域包括：针对说话人，可分为特定说话人语音识别和非特定说话人语音识别；针对词汇量，可划分为小词汇量、中词汇量和大词汇量的识别，按说话方式，可分为孤立词识别和连续语音等。最简单的研究领域是特定说话人、小词汇量、孤立词的识别，而最难的研究领域是非特定人、大词汇量、连续语音识别。

在进入新世纪之前，语音识别技术大都只在特定行业或场所中使用或者仅仅停留在实验室，处于探索和试验中。最近十年由于消费电子行业的兴起和移动互联网技术的爆发。越来越多的自动化和自能化产品走进人们的日常生活。语音识别技术也随之进入大众的视线，并开始为更多人所了解和使用。例如语音门禁、智能电视上的语音换台、智能手机上的语音拨号、语音控制等等。语音识别技术正在由过去的实验探索迈入实用化阶段。我们有理由相信会有越来越多的产品用到语音识别技术，它与人工智能能技术的结合将会是一个重要的发展方向。语音识别技术最终会改变人与机器之间的交互方式，使之更加自然、便捷、轻松。

本设计的孤立词语音识别是语音识别技术中较为基本的，算法实现也较简单，适合于在嵌入式平台中实现一些简单的语音控制功能。以往类似系统大都基于ARM9、ARM11、DSP、SOC等。这些平台系统规模较大、开发和维护的难度较大、成本也相对较高。STM32是意法半导体(ST)公司推出的基于ARM Cortex-M3内核的高性能单片机。上市之后，由于其出色的性能、低廉的价格，很快被运用到众多产品中。经测试，STM32F103VET6单片机拥有能够满足本系统孤立词语音识别所需的运算和存储能力。所以在本系统中采用STM32F103VET6作为主控制器，采集并识别语音信号。以低廉的成本，高效的算法完成了孤立词语音识别的设计目标。本系统主要涉及的内容如下述：

语音信号的采集和前端放大、防混叠滤波、模数转换。

语音信号预处理，包括预加重、分帧、加窗。

语音信号端点检测，检测输入信号中有效语音的起始和结束点

语音信号特征提取。提取有效语音中每帧语音信号的Mel频率倒谱系数(MFCC)系数。

模板训练，对每个语音指令采集多个语音样本，根据语音样本获取每个语音指令的特征模板。

特征匹配，使用动态时间规整（DWT）算法计算输入语音信号与各模板的匹配距离。识别输入的语音信号。

系统硬件电路设计，人机界面设计。

第一章 方案论证及选择

1.1系统设计任务要求

本系统利用单片机设计了一个孤立词语音识别系统，能够识别0~9、“上”、“下”、“左”、“右”14个汉语语音指令。系统通过触摸式LCD与用户交互。

本设计的主要要求如下：

1．采集外部声音信号，转换为数字信号并存储。

2．在采集到的声音信号中找出有效语音信号的开始和结束点。

3．分析检测到的有效语音，得出语音信号特征。

4．对每个待识别的语音指令，建立特征模版。

5．比较输入语音信号特征与特征模版，识别输入的语音信号

6．显示系统操作界面，并能够接受用户控制。

1.2硬件选择

1.2.1 硬件方案总体介绍

系统硬件由音频放大模块、MCU、触摸屏、电源四部分组成。音频放大模块完成对外部声音信号的采集和放大。将声音信号转化为电信号，并放大到0~3V。MCU的ADC参考电压为其电源电压3.3V。音频放大模块的输出信号不超出MCU ADC的电压范围，并且能够获得最大的量化精度。MCU对音频放大模块输入的声音信号进行AD转换。然后提取并识别信号特征。另外，MCU还控制触摸屏的显示和读取触摸屏点击位置。触摸屏负责显示操作界面，并接收用户操作。电源为电池供电。

系统硬件结构图如图1.1所示。

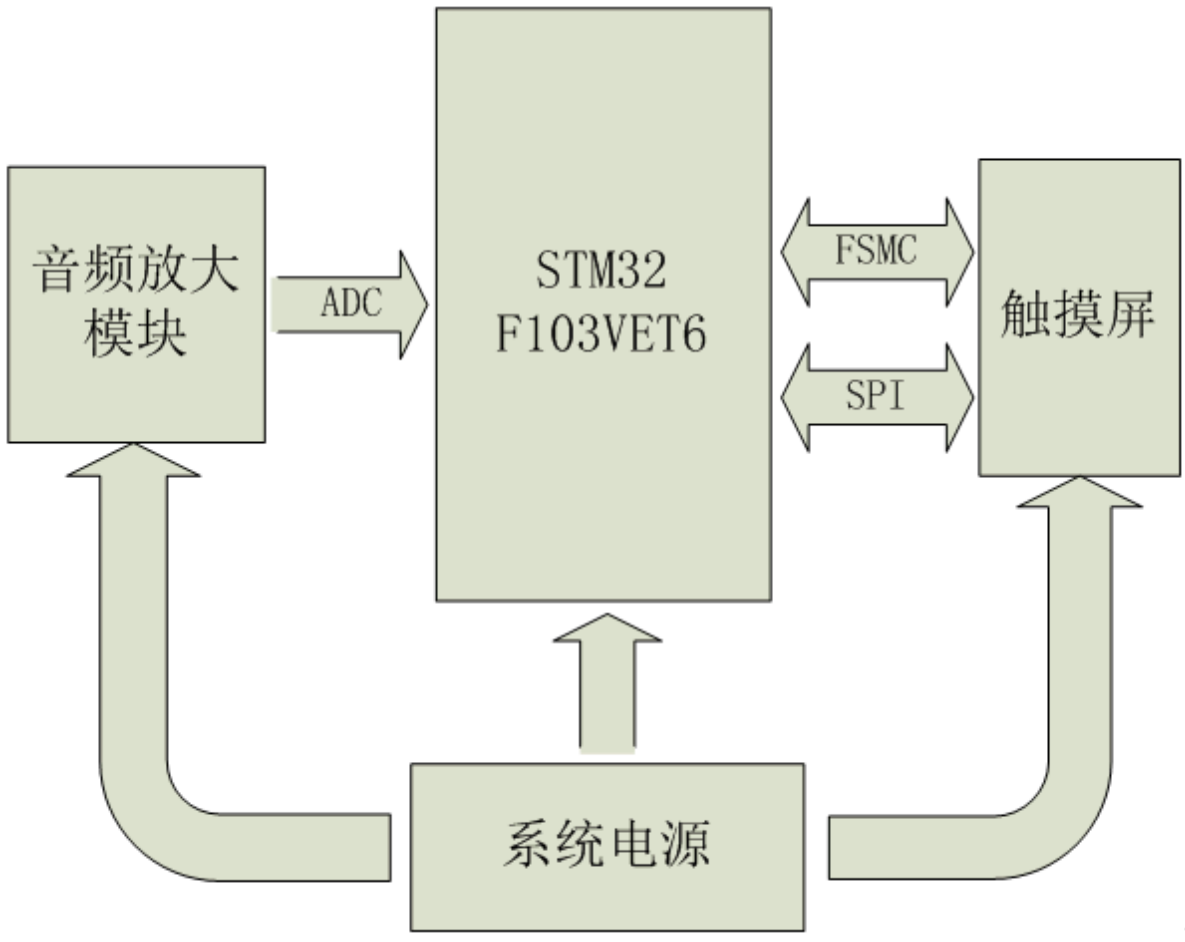


图1.1系统硬件总体结构图

1.2.2 MCU选择

传统上孤立词语音识别多采用语音识别专用芯片，例如凌阳SPCE061A、LD3320等。此种方案设计简单，开发周期较短，但可拓展性较差，一般只能识别特定的语音，或者识别语音指令的个数有限制。且专用芯片价格一般相对较高，对系统成本控制不利。

STM32F103VET6是意法半导体（ST）推出的高性能32位Cortex-M3内核单片机，带有ADC、DAC、USB、CAN、SDIO、USART、SPI、IIC、FSMC、RTC、TIM、GPIO、DMA等大量片上外设。Cortex-M3内核属于ARM公司推出的最新架构ARMv7中的M系列，侧重于低成本、低功耗、高性能。其最高主频可达72MHz， 1.25 DMIPS/MHz的运算能力，三级流水线另加分支预测，并且还带有单周期乘法器和硬件除法器。相比较ARM7TDMI内核，Cortex-M3在性能上有较大的提升。

STM32F103VET6内置3个一共21通道的12位ADC，采样频率最高可达1MHz。12通道DMA控制器，可访问系统Flash、SRAM、片上外设，能够处理内存到外设、外设到内存的DMA请求。11个16位定时器，其中T1、T2、T3、T4、T5、T8可连接到ADC控制器，在每次定时器捕获/比较事件到来时自动触发ADC开始一次A/D转换。A/D转换完成后可自动触发DMA控制器将转换后的数据依次传送至SRAM的数据缓冲区。因此STM32F103VET6能够进行精确且高效的A/D转换。能够满足音频信号采集的需求。

STM32F103VET6的FSMC(Flexible Static Memory Controller，可变静态存储控制器)能够根据不同的外部存储器类型，发出相应的数据/地址/控制信号类型以匹配信号的速度。FSMC连接至LCD控制器，可将LCD控制器配置为外部NOR Flash。在系统需要访问LCD时，自动生成满足LCD控制器要求的读写时序，能够精确、快速地完成对LCD界面显示的控制。内置3个最高可达18Mbit/s的SPI控制器，与触摸屏控制器相连能够实现触摸屏点击位置检测。

本系统中采集一个汉语语音指令。录音时间长度2s，以8KHz 16bit采样率对语音进行采集，所需存储空间为32KB，另外加上语音处理、特征提取及特征匹配等中间步骤所需RAM空间不会超过64KB。而STM32F103VET6带有512KB Flash和64KB RAM。所以STM32F103VET6在程序空间上能够满足。语音识别中最耗时的部分是特征提取中的快速傅立叶变换换。一般来说，孤立词语音识别中有效语音时间长度小于1s。语音信号一般10~30ms为一帧，本系统中按20ms一帧，帧移（相邻两帧的重叠部分）10ms，这样一个语音指令不超过100帧。在8KHz 16bit的采样率下，20ms为160采样点 。STM32固件库所提供的16位、1024点FFT，在内核以72MHz运行时每次运算仅需2.138ms。完成100帧数据的FFT所需时间为213.8ms，加上其他处理所需时间，识别一个语音指令耗时不会超过0.5s。所以在程序运行时间上STM32F103VET6也能够满足需要，能够进行实时的孤立词语音识别。

基于以上论证，本系统选用STM32F103VET6作为主控MCU。

1.2.3音频信号采集方案选择

音频信号采集多采用音频编解码芯片，例如UDA1341、VS1003等。此类芯片能够提供丰富的功能，且系统一致性较好，但它们成本较高。本系统是一个低成本解决方案，并且只需要采集音频信号。因此不宜采用那些专用的音频编解码芯片。

在本系统的音频放大模块中使用小型话筒完成声电信号转换，两个9014三极管构成两级共基极放大电路。在每一级中加电压负反馈，稳定放大倍数。

语音信号的频带为300~3400Hz，根据抽样定理，抽样频率设为8000Hz就足以完成对语音信号的采集。在本系统中TIM1被设置为ADC触发信号源。TIM时钟源为系统时钟72MHz。经100分频，变为720KHz。计数模式为向上递增，自动重载值为90，即计数值从0递增到90再返回0。比较匹配值设为0~90间任意一个数值 ，则每秒可发出8000次比较匹配事件。ADC每秒完成8000次A/D转换，即抽样频率为8KHz。

1.2.4显示及操作界面选择

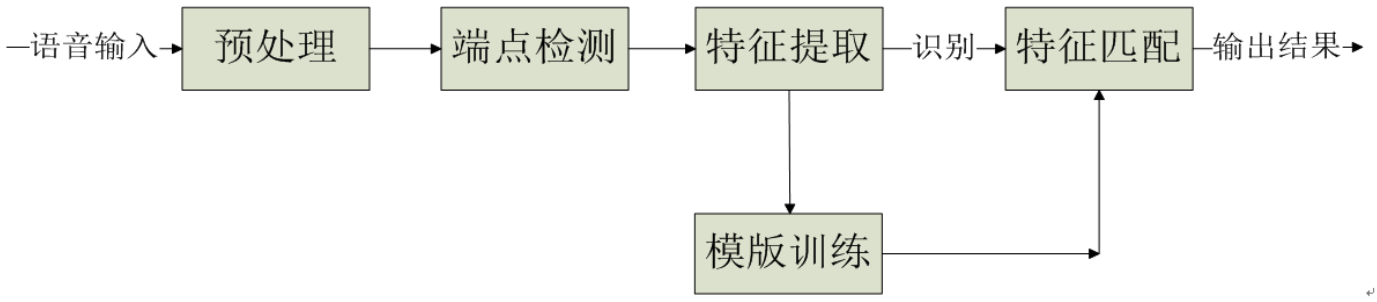
触摸屏作为一种新的输入设备，它是目前最简单、方便、自然的一种人机交互方式。LCD触摸屏是一种可接收触摸点击输入信号的感应式液晶显示装置。当接触或点击屏幕时，触摸控制器可读取触摸点位置，如此可通过屏幕直接接受用户的操作。相比较机械式按钮，触摸屏在操作上更加直观生动。综合考虑，本设计中采用2.5寸240×320分辨率的LCD触摸屏实现界面显示和操作。

1.3算法选择

1.3.1软件算法总体介绍

对采集到的音频信号进行预处理、端点检测、特征提取、模板训练、特征匹配的一些列处理，最终识别输入语音。

系统软件流程图如下图所示。



1.3.2预处理算法选择

语音信号的预处理主要包括： ADC、分帧、数据加窗、预加重。

语音信号的频率范围通常取100Hz~3400Hz，因为这个频段包含绝大部分的语音信息，对语音识别的意义最大。根据采样定律，要不失真地对3400Hz的信号进行采样，需要的最低采样率是6800Hz。为了提高精度，常用的A / D采样率在8kHz到12kHz。

语音信号有一个重要的特性：短时性。由于人在说话中，清音与浊音交替出现，并且每种音通常只延续很短的一段时间。因此，从波形上看，语音信号具有很强的"时变特性"。在浊音段落中它有很强的周期性，在清音段落中又具有噪声特性，而且浊音和清音的特征也在不断变化之中。如图1.4所示，其特性是随时间变化的，所以它是一个非稳态过程。但从另一方面看，由于语音的形成过程是与发音器官的运动密切相关的，这种物理性的运动比起声音振动速度来说是缓慢的（如图1.5所示）。因此在一个短时间范围内，其特性变化很小或保持不变，可以将其看做一个准稳态过程。我们可以用平稳过程的分析处理方法来分析处理语音信号。

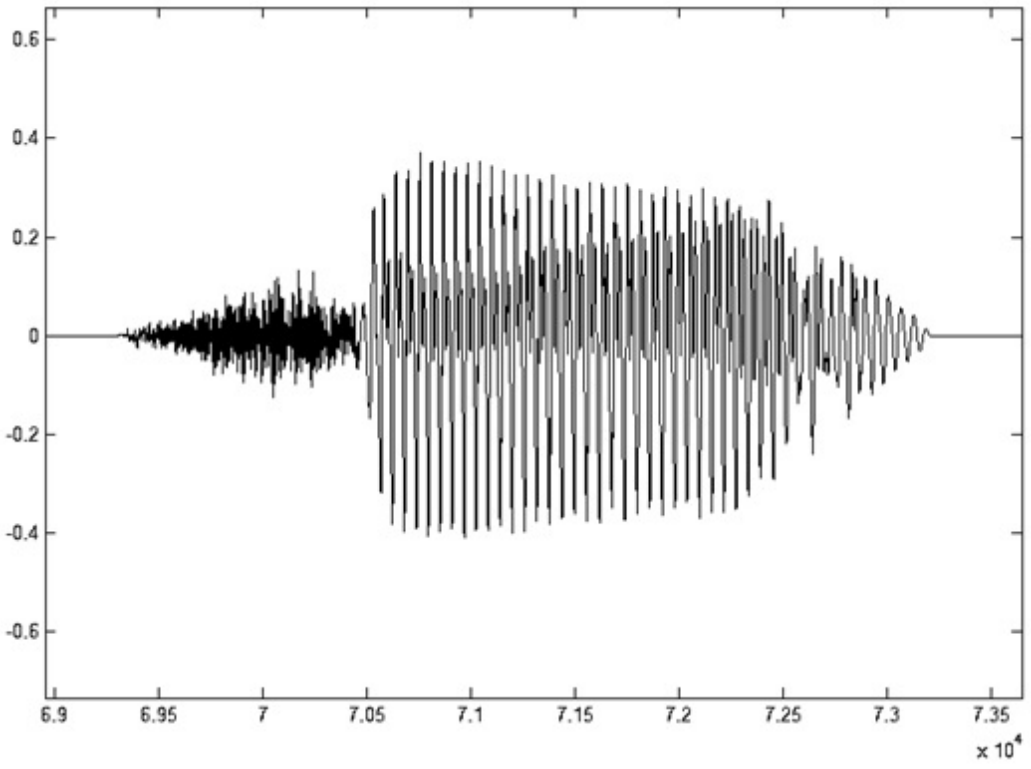


图1.4 语音"7"的时域波形

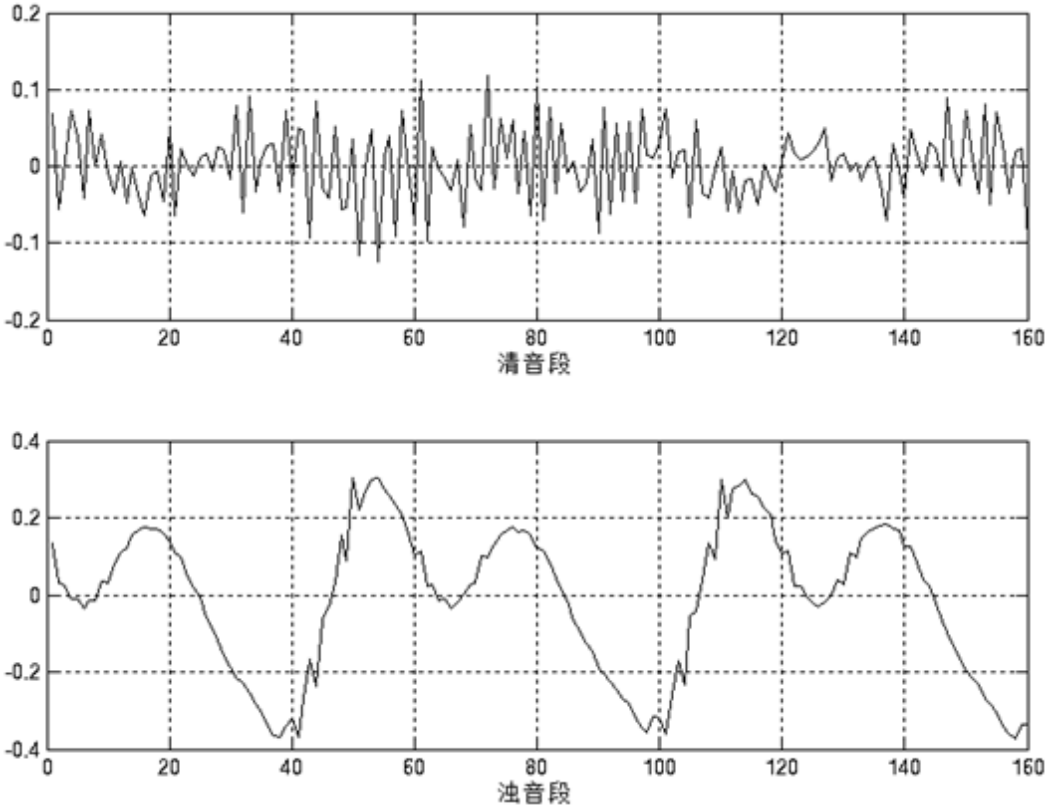


图1.5 语音“7”清音段和浊音段的20ms短时波形

基于以上考虑，对语音信号的分析处理必须采用短时分析法，也就是分帧。语音信号通常在10ms~30ms之间保持相对平稳。在本设计中，每帧取20ms。为了使前后帧之间保持平滑过渡，帧移10ms，即前后帧之间交叠10ms。

为了便于后续语音处理，需对分帧后的信号加窗。加窗方式如式(1-1)。

$$Y(n) = y(n)w(n)\ ,\ \ 0 \le n \le N-1 \qquad (1\text{-}1)$$

式中$Y(n)$是加窗后的信号，$y(n)$是输入信号，$w(n)$是窗函数，N是帧长。

窗函数可以选择矩形窗，即

$$w(n) = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{其他} \end{cases} \qquad (1\text{-}2)$$

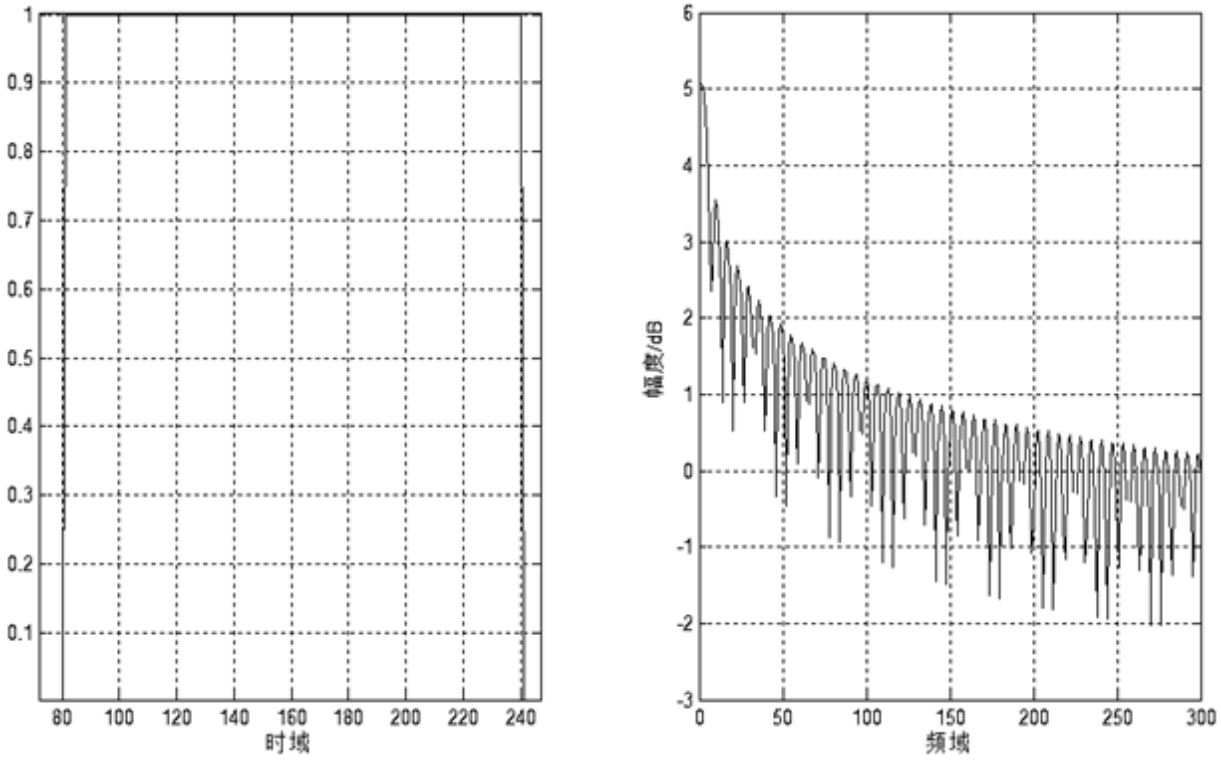


图1.6 矩形窗时域、频域示意图

或其他形式窗函数，如汉明窗

$$w(n)=\begin{cases}0.54-0.46\cos[2\pi n/(N-1)], & 0\le n\le N-1\\ 0, & \text{其他}\end{cases} \quad (1\text{-}3)$$

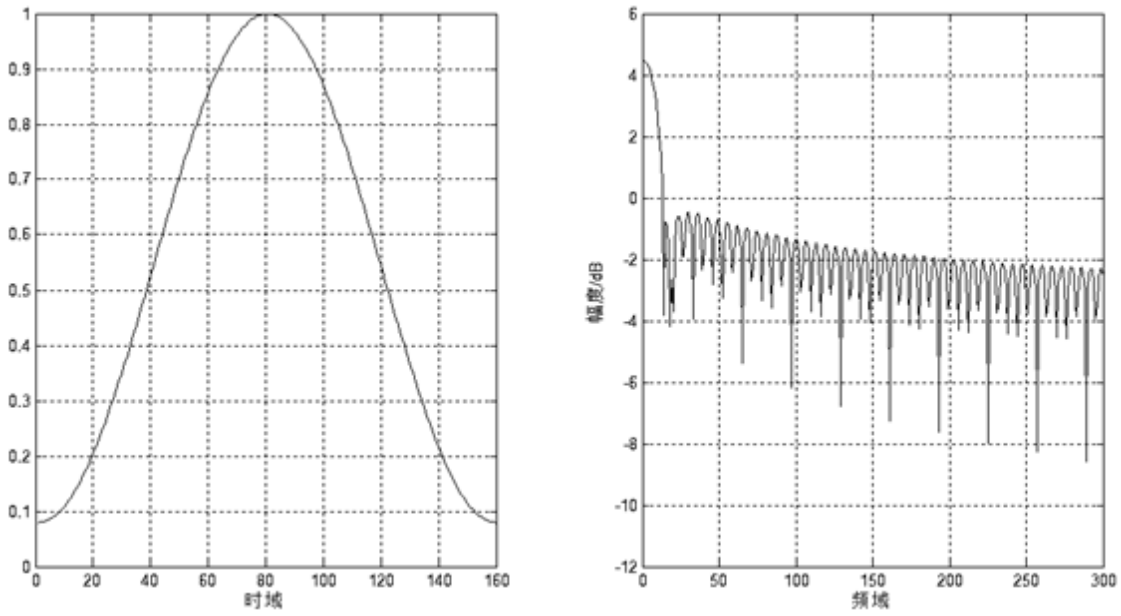


图1.7 汉明窗时域、频域示意图

这些窗函数的频率响应都具有低通特性，但不同的窗函数形状将影响分帧后短时特征的特性。图1.7和图1.8分别给出了160点矩形窗和汉明窗的时域和频域示意图。从图中可以看出汉明窗的带宽大约是同样宽度矩形窗带宽的两倍。同时，在通带外汉明窗的衰减比矩形窗大得多。矩形窗的主瓣较小，旁瓣较高；而汉明窗具有最宽的主瓣宽度和最低的旁瓣高度。

对语音信号分析来说，窗函数的形状是非常重要的，矩形窗的谱平滑性较好，但波形细节易丢失，并且矩形窗会产生泄露现象。而汉明窗可以有效地克服泄露现象，应用范围也最为广泛。基于以上论述，本设计选用汉明窗作为窗函数。图1.9和图1.10分别给出了一帧浊音加矩形窗和汉明窗后的时域和频域效果。

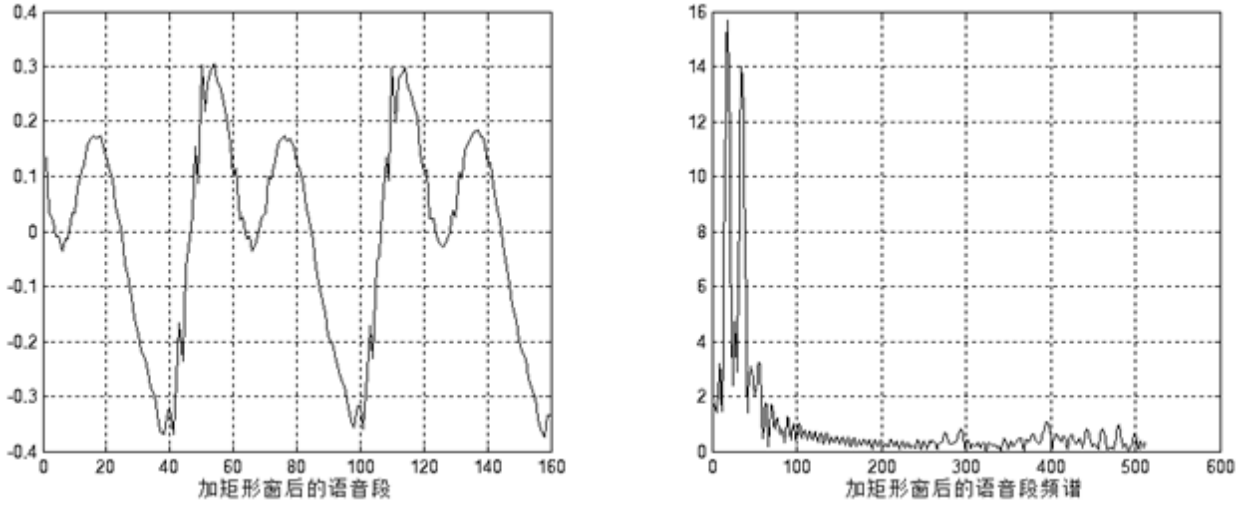


图1.8 加矩形窗

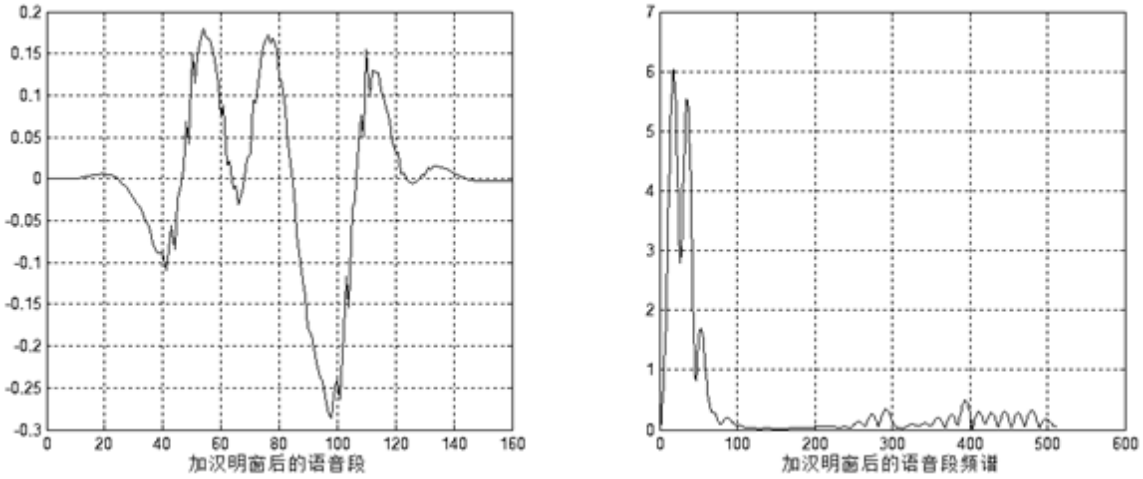


图1.9 加汉明窗

由于人的发声器官的固有特性，语音从嘴唇辐射将有6dB / 倍频的衰减。此种效应主要表现在高频信息的损失，对语音信号的特征提取会造成不利的影响。因此，必须对信号进行高频提升，即对信号进行高频的补偿，使得信号频谱平坦化，以便于进行频谱分析或声道参数分析。预加重可以用具有

6dB/倍频提升高频特性的预加重数字滤波器实现。预加重滤波器一般是一阶的，其系统函数和差分方程如式(1-4)

$$H(z) = 1 - uz^{-1}$$
$$y(\mathrm{n}) = x(\mathrm{n}) - ux(\mathrm{n}-1) \quad (1\text{-}4)$$

其中y(n)为提升后的输出值，x(n)和x(n-1)分别为当前时刻和前一时刻的输入值。*u*接近于1，典型取值在0.94~0.97之间。本设计取0.95。预加重效果如图1.11所示。

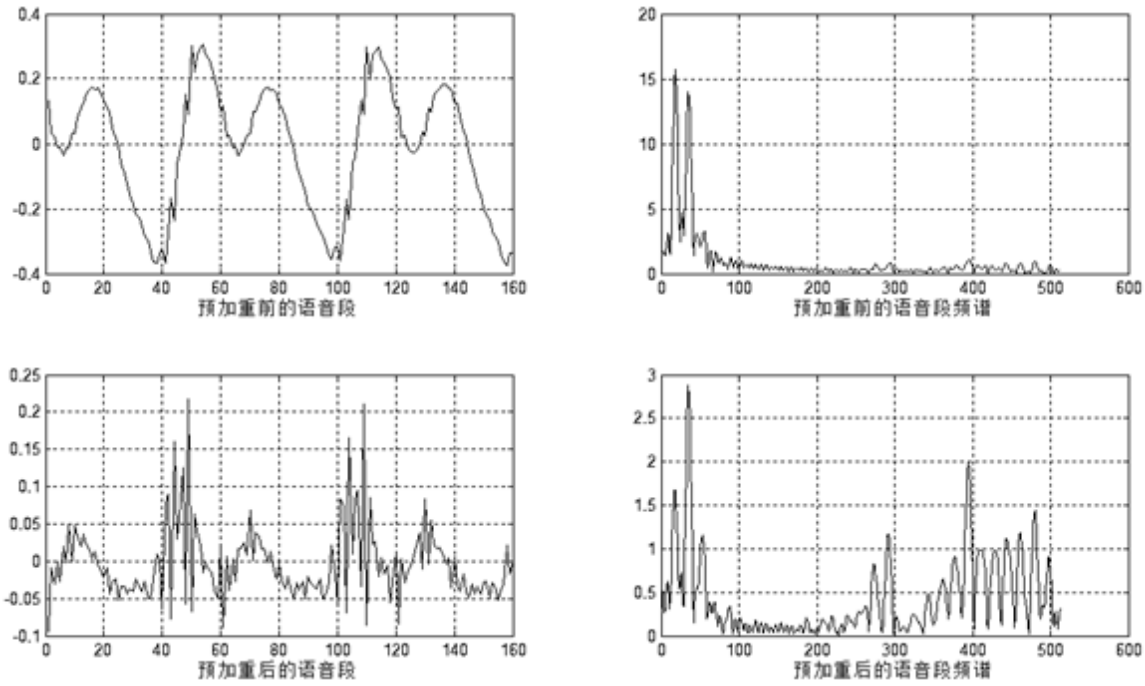


图1.10 预加重效果图

1.3.3端点检测算法选择

语音端点检测(VAD)，也称为语音活动性检测，主要应用在语音处理中的语音编解码，语音识别及单信道语音增强等领域。语音端点检测的基本方法可以用一句话来表达：从输入信号中提取一个或一系列的对比特征参数，然后将其和一个或一系列的门限阀值进行比较(如图3-2)。如果超过门限则表示当前为有音段；否则表示当前为无音段。门限阀值通常是根据无音段时的特征确定的。但是由于语音和环境噪声的不断变化，使得这一判决过程变得非常复杂。通常语音端点检测是在语音帧的基础上进行的，语音帧的长度在10ms~30ms不等。一个好的语音端点检测算法必须具有对各种噪声的鲁棒性，同时要简单、适应性能好、时延小、且易于实时实现。

在高信噪比的情况下，常用的检测方法大体上有以下几种：短时能量、短时过零率。这些方法都是利用了语音和噪声的特征参数，因此判别效果较好。并且它们实现简单，计算量相对较小，因而得

到广泛的应用。

短时能量定义如下式：

$$E = \sum_{n=0}^{N-1} x^2(n) \tag{1-6}$$

式中N为帧长，E为一帧的短时能量值。

短时能量主要有以下几个方面的应用：首先短时能量可以区分清音和浊音，因为浊音的能量要比清音的大得多；其次可以用短时能量对有声段和无声段进行判定，以及连字分界等。短时能量由于是对信号进行平方运算，因而人为增加了高低信号之间的差距。更重要的的是平方运算的结果很大，容易产生数据溢出。解决这些问题的简单方法是采用短时平均幅度值来表示能量的变化。其定义如下：

$$E = \sum_{n=0}^{N-1} |x(n)| \tag{1-7}$$

短时过零率是语音信号时域分析中最简单的一种特征，它指每帧内信号通过零值的次数，定义如下：

$$Z = \frac{1}{2} \sum_{n=0}^{N-2} |\mathrm{sgn}[x(n+1)] - \mathrm{sgn}[x(n)]| \tag{1-8}$$

式中，sgn(x)是符号函数，即

$$\mathrm{sgn}[x(n)] = \begin{cases} 1, & x(n) \geq 0 \\ -1, & x(n) < 0 \end{cases} \tag{1-9}$$

根据以上定义，清音由于类似于白噪声，所以过零率较高。浊音的能量集中于低频段，所以浊音信号的短时过零率较低。噪声的短时过零率较高，这主要是因为语音信号的能量主要集中在较低的频率范围内，而噪声信号的能量主要集中于较高的频段。这样计算的短时过零率容易受到噪声干扰。解决这个问题的方法是对上述定义稍作修改，即设置一个门限T，将过零率的含义修改为跨过正负门限的次数。修改后的定义如下式：

$$Z=\frac{1}{2}\sum_{n=0}^{N-2}\{|\mathrm{sgn}[x(n+1)-T]-\mathrm{sgn}[x(n)-T]|+|\mathrm{sgn}[x(n+1+T)]-\mathrm{sgn}[x(n)+T]|\}$$

(1-10)

这样计算的短时过零率就有一定的抗干扰能力，即使存在随机噪声，只要它不超过正负门限所构成的带，就不会产生虚假过零率。

综合考虑设计需要和系统处理能力，本设计采用短时幅度值和改进后的短时过零率判断语音起始和结束点。分别为短时幅度和短时过零率设置门限值。每次识别前，选定语音段前300ms作为背景噪声，用以确定这两个门限值，实现对背景噪声的自适应。具体的端点检测方法如下。

判断语音起始点，要求能够滤除突发性噪声。突发性噪声可以引起短时能量或过零率的数值很高，但是往往不能维持足够长的时间，如门窗的开关，物体的碰撞等引起的噪声，这些都可以通过设定最短时间门限来判别。超过两门限之一或全部，并且持续时间超过有效语音最短时间门限，返回最开始超过门限的时间点，将其标记为有效语音起始点。判断语音结束点，要求不能丢弃连词中间短暂的有可能被噪声淹没的“寂静段”。这可以通过设定无声段最长时间门限来判别。同时低于两门限，并且持续时间超过无声最长时间门限，返回最开始低于门限的时间点，将其标记为有效语音结束点。

图1.11和图1.12分别给出了上述算法在一般信噪比和低信噪比情况下的端点检测效果。从图中可以看出上述算法能够适应一般的背景噪声。在背景噪声较高时，上述算法无法准确判断语音起始结束点.但经过试验，当信噪比低至图1.12所示时时人耳也很难准确辨识语音。所以上述算法在实际使用中能够满足端点检测的需求。

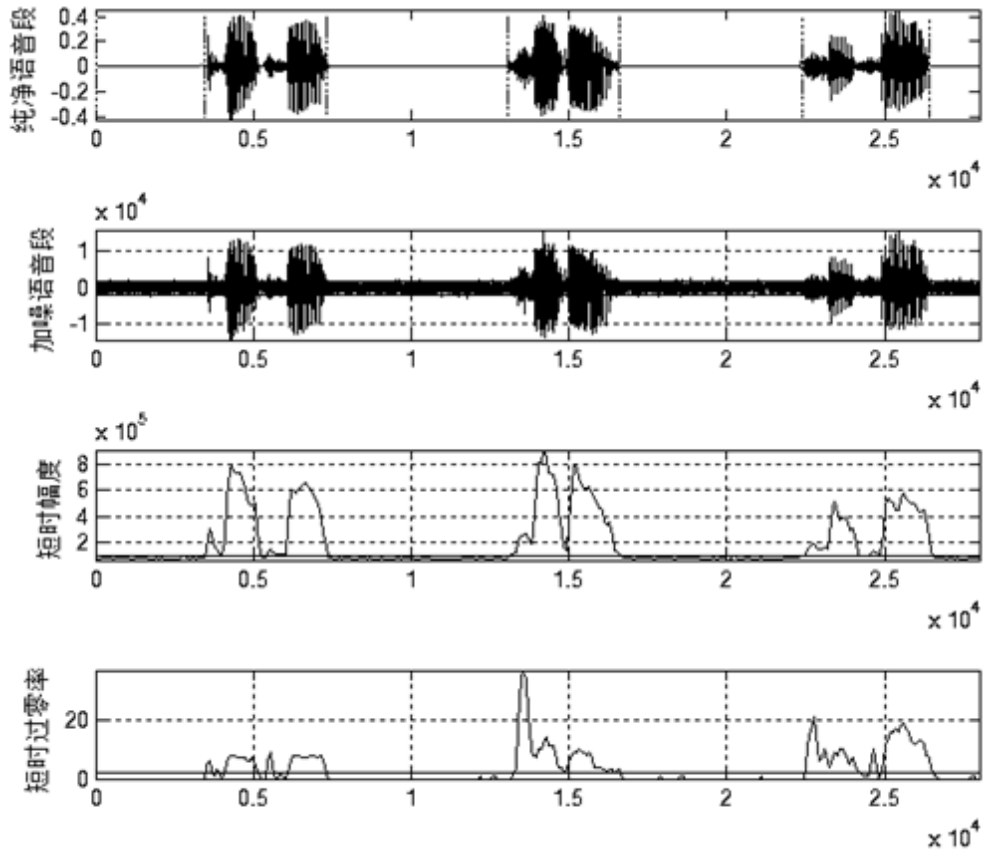


图1.11 一般信噪比下的端点检测效果

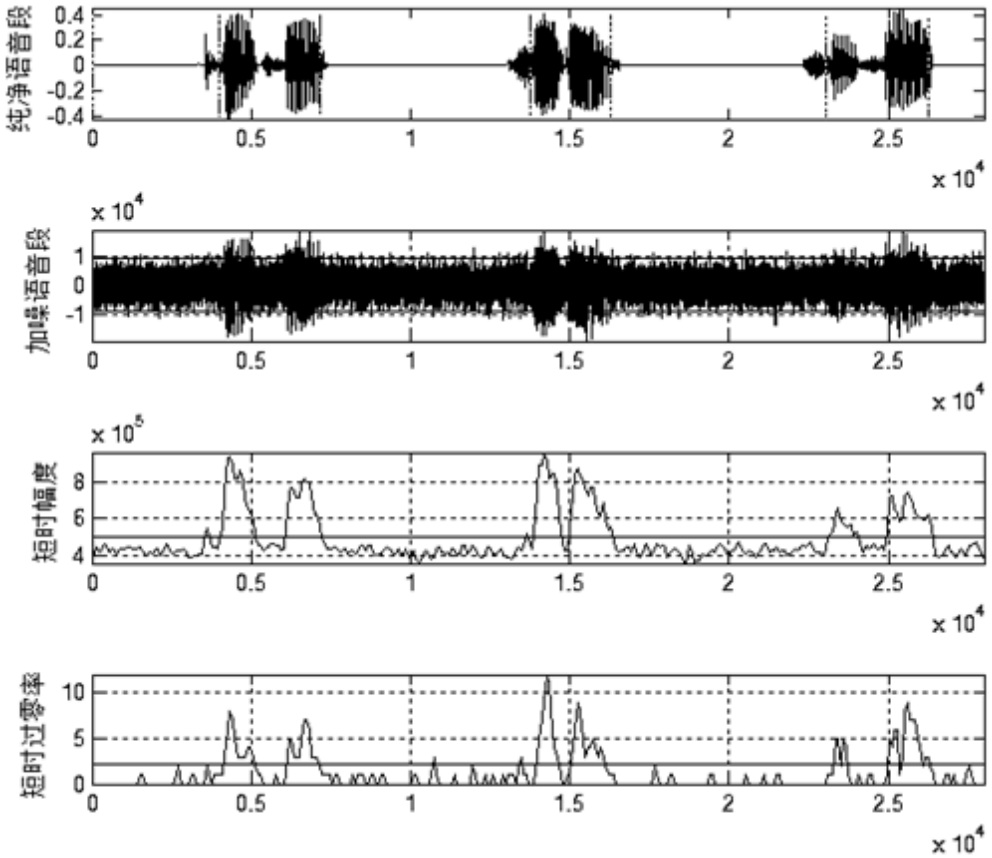


图1.12 低信噪比下的端点检测效果

1.3.4特征提取算法选择

在语音识别系统中，模拟语音信号在完成A / D转换后成为数字信号。此时的语音信号为时域的信号，时域的信号难以进行分析和处理，而且数据量庞大。通常的做法是对时域信号进行变换，提取其中某种特定的参数，通过一些更加能反映语音本质特征的参数来进行语音识别。特征提取是识别过程中一个非常重要的环节，选取的特征直接影响到识别的结果。不同的特征对不同语音的敏感度也不一样，优秀的语音特征应该对不同字音距离较大，而相同字音距离较小。

另外，特征值的数目也是一个重要的问题。在满足使用要求的情况下，所使用的特征数应该尽量减少，以减少所涉及的计算量。但是过少的特征有可能无法恰当的描述原始语音，以致识别率下降。语音特征的提取方法是整个语音识别的基础，因此受到了广泛的重视。通过近几十年的发展，目前语音特征的提取方法主要有以下三类：

1.基于线性预测分析的提取方法。这一类的典型代表是线性预测倒谱系数LPCC。

2.基于频谱分析的提取方法。这一类的典型代表是Mel频率倒谱系数MFCC。

3.基于其它数字信号处理技术的特征分析方法。如小波分析、时频分析、人工神经网络分析等。

目前的孤立词语音识别系统大多采用前两种语音特征提取方法。在本文中，借鉴前人对LPCC系数和MFCC系数的总结对比，采用Mel频标倒谱系数MFCC。

人类的耳蜗实质上相当于一个滤波器组，耳蜗的滤波作用在1000Hz以下为线性尺度，而1000Hz以上为对数尺度，这就使得人耳对低频信号的分辨率高于对高频信号的分辨率。根据这一特性，研究者根据心理学实验得到了类似于耳蜗作用的一组滤波器组，这就是Mel频率滤波器组。Mel频率可以用如下公式表示：

$$f_{\mathrm{Mel}} = 2595 \times \log(1 + f/700) \qquad (1\text{-}11)$$

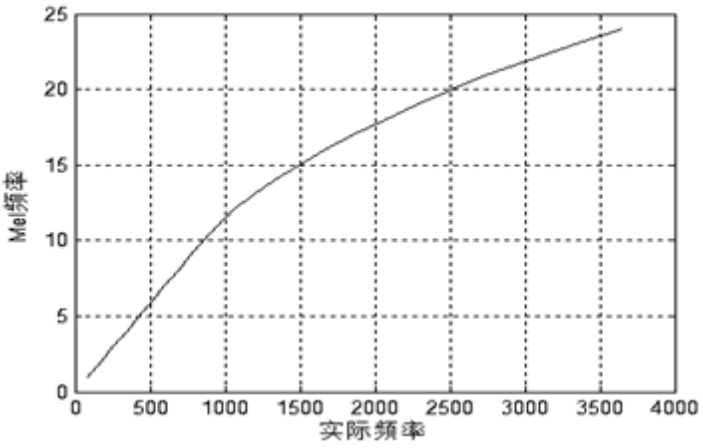


图1.13 Mel频率与实际频率的对应关系

对频率轴的不均匀划分是MFCC特征区别于普通倒谱特征的最重要特点。将频率按照式(1-11)和图1.13变换到Mel域后，Mel带通滤波器组的中心频率是按照Mel频率刻度均匀排列的。在本设计中，MFCC倒谱系数的计算过程如下述。

1.对语音信号预加重、分帧、加汉明窗处理，然后进行短时傅里叶变换，得出频谱。

2.取频谱平方，得能量谱。并用24个Mel三角带通滤波器进行滤波；由于每个频带的分量在人耳中是叠加的，因此将每个滤波器频带内的能量进行叠加，输出Mel功率谱。

3.对每个滤波器的输出值取对数，得到相应频带的对数功率谱。然后对24个对数功率进行反离散余弦变换得到12个MFCC系数，反离散余弦变换如式(1-12)，式中M=24，L=12。

$$C_n = \sum_{k=1}^{M} x(n)\cos[\pi(k-0.5)n/M] \quad , \quad n = 1,2,\cdots,L \tag{1-12}$$

在本设计中采集语音信号的抽样频率是8000Hz，频率范围是0Hz~4000Hz。在此频率范围内的Mel三角带通滤波器组如下图所示：

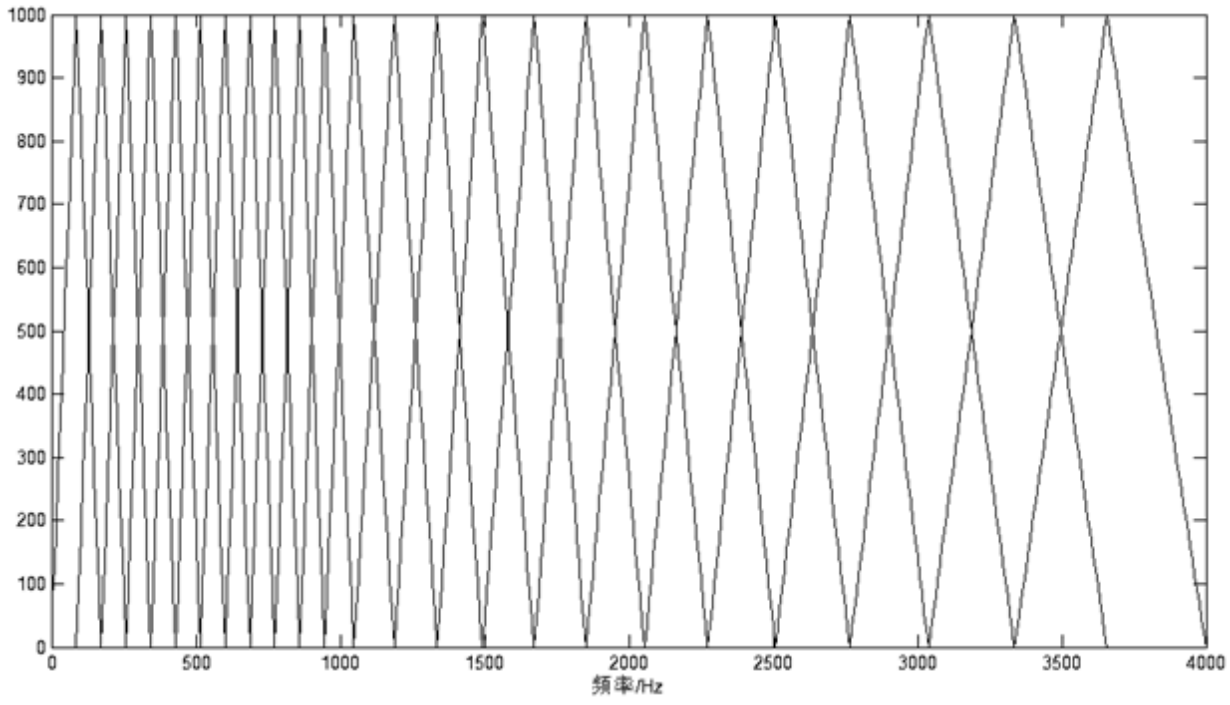


图1.14 Mel三角滤波器组

与LPCC参数相比，MFCC参数具有以下优点：

1．语音的信息大多集中在低频部分，而高频部分易受环境噪声干扰。MFCC参数将线性频标转化为Mel频标。强调语音的低频信息，从而突出了有利于识别的信息，屏蔽了噪声的干扰。

2．MFCC参数没有任何前提假设，在各种情况下均可使用。而LPCC参数需要假定所处理的信号为AR信号，对于动态特性较强的辅音，这个假设并不是严格成立的。

因此，MFCC参数的抗噪声特性是优于LPCC参数的。在本设计中，采用的语音特征参数均为MFCC参数。

1.3.5特征匹配算法选择

要建立一个性能良好的语音识别系统仅有好的语音特征还不够，还要有适当的语音识别的模型和算法。在现阶段，语音识别的过程是根据模式匹配的原则，计算未知语音模式与语音模板库中的每一个模板的距离测度，从而得到最佳的匹配模式。目前，语音识别所应用的模型匹配方法主要有动态时间弯折(DTW：Dynamic Time Warping)、隐马尔可夫模型(HMM：Hidden Markov Model)和人工神经网络(ANN：Artificial Neural Networks)等。当今孤立词识别领域最常用的识别算法是DTW和HMM。

DTW算法是较早的一种模式匹配和模型训练技术，它应用动态规划的方法成功解决了语音信号特征参数序列比较时时长不等的难题，在孤立词语音识别中获得了良好的性能。DTW算法是建立在动态规划(DP：Dynamic Programming)的理论基础上的。动态规划是一个很有效的方法来求取一个问题的最佳解。其中心思想简单的说可以描述为：在一条最佳的路径上，其中任意一条子路径也都必须是相关子问题的最佳路径，否则原路径就不是最佳路径。

HMM算法是数学上一类重要的双重随机模型，用概率统计的方法描述时变语音信号，很好的描述了语音信号的整体非平稳性和局部平稳性。HMM的各状态对应语音信号的各平稳段，各状态之间以一定转移概率相联系，是一种较为理想的语音模型。HMM模型属于统计语音识别，适用于大词汇量、非特定人的语音识别系统。随着现代计算机技术的迅猛发展，计算机的运算速度迅速提高，隐马尔科夫模型分析方法也得到了广泛利用。该算法在识别阶段计算量较少，适应性强，但是需要大量的前期训练工作，对系统资源的要求较多。

用于孤立词识别，DTW算法与HMM算法在相同的环境条件下，识别效果相差不大，但是HMM算法要复杂得多，这主要体现在HMM算法在训练阶段需要提供大量的语音数据，通过反复计算才能得到模型参数，而DTW算法的训练中几乎不需要额外的计算。所以在孤立词语音识别中，DTW算法得到更广泛的应用。

综合比较DTW算法的工作量小，不需要大量的语音数据，而且DTW算法适合孤立词语音识别，且容易实现，节省系统资源，比较方便移植到嵌入式系统中。所以本系统选择DTW算法作为语音识别的核心算法。下面介绍DTW算法及其实现方法。

假设参考模板的特征矢量序列为,输入语音特征矢量序列为$X = \{x_1, x_2, \cdots, x_I\}$,输入语音特征矢量序列为 $Y = \{y_1, y_2, \cdots, y_J\}, I \neq J$。DTW算法就是要寻找一个最佳的时间规整函数，使待测语音的时间轴j非线性地映射到参考模板的时间轴i上，使总的累积失真量最小。

设时间规整函数为

$$C = \{c(1), c(2), \cdots, c(N)\} \tag{1-13}$$

式中N为匹配路径长度，$c(n) = (i(n), j(n))$表示第*n*个匹配点是参考模板的第i(n)个特征矢量与待测模板的第j(n)个特征矢量构成。两者之间的距离$d(x_{i(n)}, y_{j(n)})$称为局部匹配距离。DTW算法就是通过局部优化的方法实现匹配距离总和最小。

一般时间规整函数满足一下约束：

1.单调性，规整函数单调增加。

2.起点终点约束，起点对起点，终点对终点。

3.连续性，不允许跳过任何一点。

4.最大规整量不超过某一极限值。$|i(n) - j(n)| < M$,*M*为窗宽。规整函数所处的区域位于平行四边形内，本设计中将平行四边形的约束区域端点放宽3点。局部路径约束，用于限制当第n步时，后几步存在几种可能的路径。本设计中DTW规整区域和局部路径如图1.16、图1.17所示。

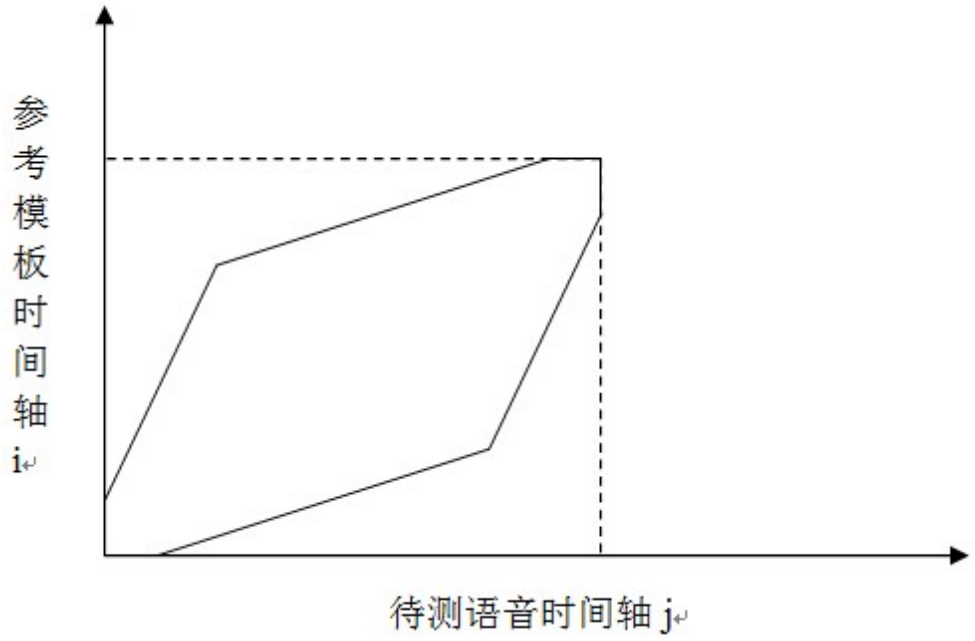


图1.16 放宽端点限制的DTW规整区域

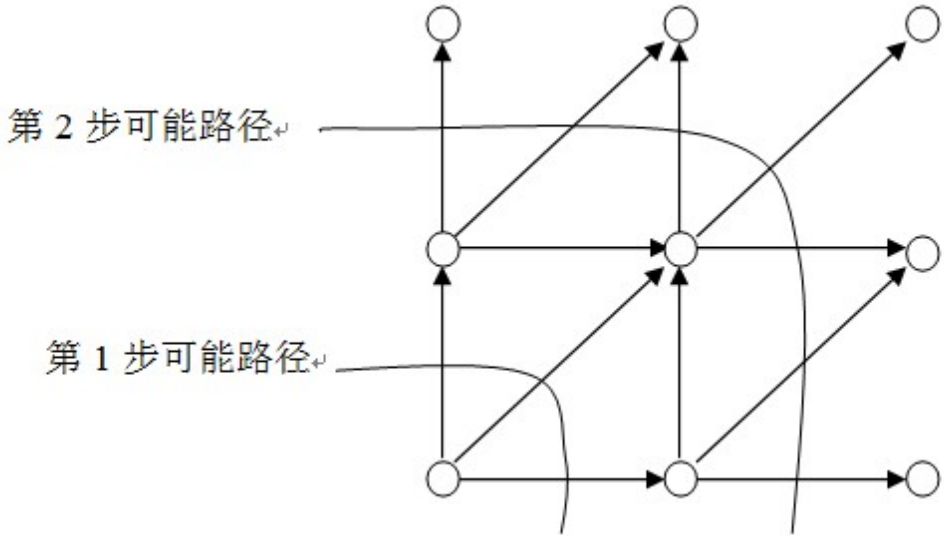


图1.17 DTW局部路径

本设计中DTW算法计算步骤：

1.初始化。令i(0)=j(0)=0，i(N)=I，j(N)=J，确定一个如图1.16所示的规整约束区域Reg。它由一平行四边形变化而来。此平行四边形有两个位于(1,1)和(I,J)的顶点，相邻两条边的斜率分别为2和1/2。

2.按照如图1.17所示的路径递推求累计匹配距离。第n步匹配距离如下式

$$g(n) = \min\{d(x_{i+1}, y_{j+1}); d(x_{i+1}, y_j); d(x_i, y_{j+1})\}$$

$$i = 2,3,\cdots,I; \quad j = 2,3,\cdots,J; \quad (i,j) \in \text{Reg} \qquad (1\text{-}14)$$

3.累计匹配距离除匹配步数，得归一化匹配距离。即输入特征与特征模板之间的匹配距离。计算输入特征与每一特征模板的匹配距离，匹配距离最小的特征模板与输入特征有最大的相似性。

第二章 系统设计

2.1硬件设计

2.1.1 MCU及其最小系统电路设计

经过第一章的论证，选用意法半导体公司的STM32F103VET6单片机。

MCU输入时钟由8MHz晶振提供，经MCU内部PLL倍频至72MHz。在每一个电源引脚上并接0.1uF去耦电容，以提高MCU电源稳定性和抗干扰性。

2.1.2 音频信号采集电路设计

音频信号采集电路原理图如下

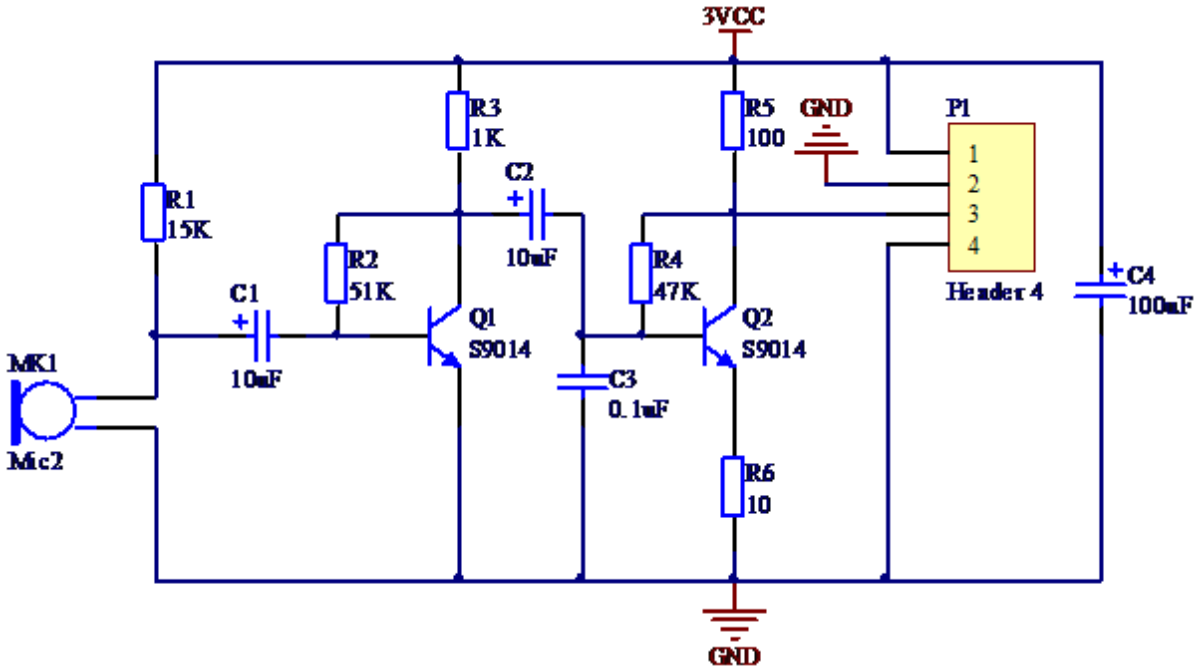


图2.6 音频信号采集原理图

2.1.3 LCD接口电路设计

本设计中显示器件选用2.4英寸TFT LCD显示屏，LCD驱动器是ILI9325 。

Thin Film Transistor (薄膜场效应晶体管)，是指液晶显示器上的每一液晶象素点都是由集成在其后的薄膜晶体管来驱动。从而可以做到高速度高亮度高对比度显示屏幕信息。

ILI9325 是一个262144色的单芯片TFT LCD SoC 驱动。 它提供240×320的分辨率, 172,800字节的图形数据RAM ，并且带有内部电源电路。它与控制器的接口可设置为16位并口、8位并口、SPI接口。在本设计中为了提高显示数据的传输速率，采用了STM32F103VET6的FSMC(可变静态存储控制器)的16位并口作为MCU和ILI9325的接口。将ILI9325的数据和控制接口映射为外部存储器。MCU传送控制命令或显示数据时，自动生成相应的时序，避免了传统上采用IO口模拟时序，提高了数据传输效率。

2.2软件设计

2.2.1 语音预处理算法设计

语音信号预处理包括： 语音信号采集、分帧、数据加窗、预加重。

语音信号采集就是将外部模拟的语音信号，转换为MCU可处理和识别的数字信号的过程。在本设计中，通过MCU内部的定时器、模数转换器以及DMA控制器实现了对音频信号采集模块输入语音信号的数字化。其处理流程如下图所示。

图2.9 语音信号数字化流程图

在程序中，控制语音信号采集的函数如下。

```
void record(void)

{

    delay_ms(atap_len_t);                    //延时，规避点击屏幕发出的噪声
```

```
    TIM_Cmd(TIM1, ENABLE);                  //开启定时器，开始信号采集

    GUI_ClrArea(&(Label[G_ctrl]));  //显示操作提示



    GUI_DispStr(&(Label[G_ctrl]),"录音中");

    delay_ms(atap_len_t);                   //开始说话之前，录制一小段背景声音，用以实现背景噪声自适应

                                            //提示开始说话

    set_label_backclor(&(Label[G_spk]), spk_clor);

                                            //等待缓冲区数据更新完毕

    while(DMA_GetFlagStatus(DMA1_FLAG_TC1)==RESET);

    TIM_Cmd(TIM1, DISABLE);                 //数据采集结束，关闭定时器

    DMA_ClearFlag(DMA1_FLAG_TC1);           //清数据传输完成标志，以备下次使用

                                            //提示开始处理采集到的数据

    set_label_backclor(&(Label[G_spk]), prc_clor);

}
```

分帧就是将采集到的语音数据分割成相同长度的片段，以用于短时分析。本设计中取20ms即160点为一帧，帧移10ms即80点。为了适应MCU存储空间有限的实际情况，分帧并没有被单独设计和占用单独的空间，而是在读语音数据缓冲区的时候按照帧长帧移的顺序依次读取。

由于端点检测属于时域分析，并不需要加窗和预加重，所以本设计中，分帧和预加重都加在端点检测之后提取MFCC之前。

2.2.2 端点检测算法设计

本设计采用短时幅度和短时过零率相结合的端点检测算法。

首先去缓冲区前300ms作为背景噪声，提取背景噪声参数。用于后续端点检测。背景噪声参数由以下结构体定义。

```
typedef struct

{

    u32 mid_val;  //语音段中值 相当于有符号的0值 用于短时过零率计算

    u16 n_thl;    //噪声阈值，用于短时过零率计算

    u16 z_thl;    //短时过零率阈值，超过此阈值，视为进入过渡段。

    u32 s_thl;    //短时累加和阈值，超过此阈值，视为进入过渡段。

}atap_tag;      //自适应参数
```

提取函数为void noise_atap(const u16* noise,u16 n_len,atap_tag* atap)，其提取过程如下。

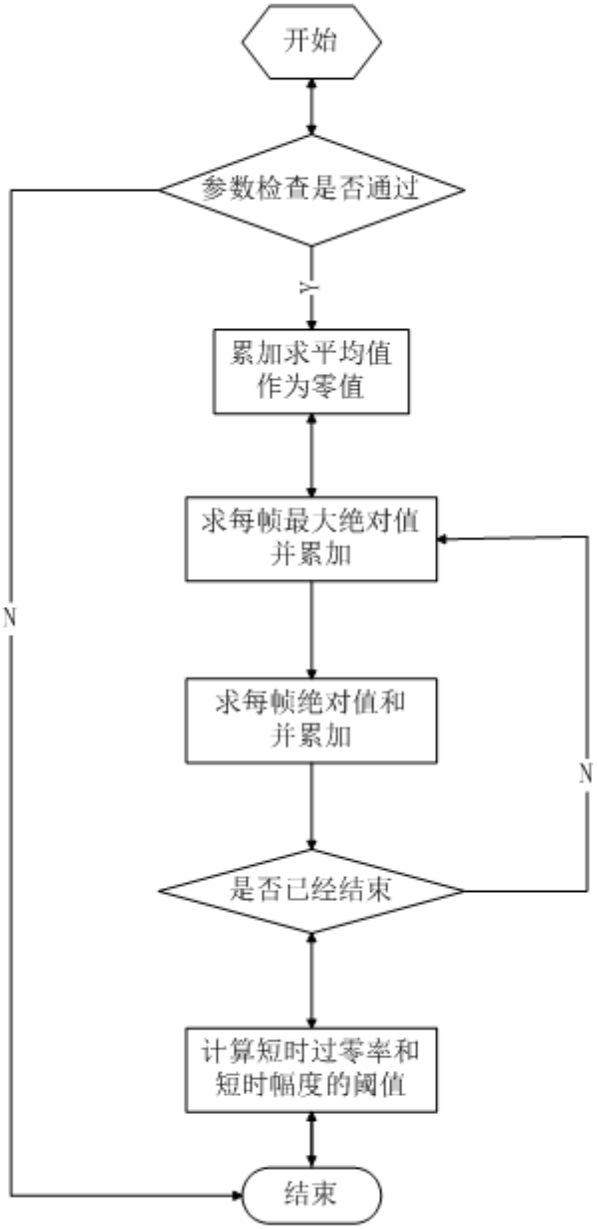


图2.10 背景噪声参数提取流程

然后根据提取到的短时过零率和短时幅度计算有效语音起始和结束点。有效语音端点由以下结构体定义。

```
typedef struct

{

    u16 *start;    //起始点

    u16 *end;      //结束点
```

```
}valid_tag;     //有效语音段
```

端点检测函数为void VAD(const u16 *vc, u16 buf_len, valid_tag *valid_voice, atap_tag *atap_arg)。其流程图如下。

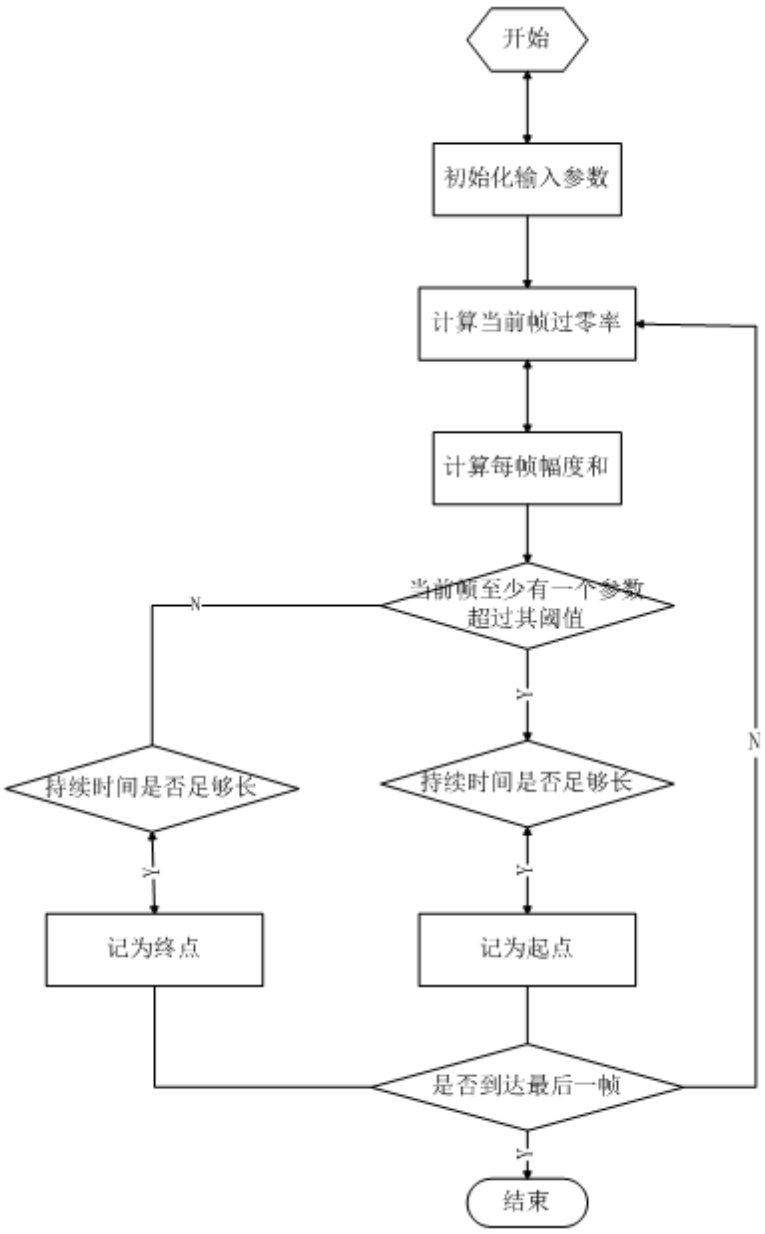


图2.11 端点检测流程

2.2.3 特征提取算法设计及优化

本设计选用12阶MFCC作为语音特征。此步是整个算法流程中最耗时也是优化空间最大的部分。因此，在程序设计中，沿用经典算法的同时做了大量的针对STM32嵌入式平台的优化工作。优化的中心思想是：尽量少使用或不使用浮点运算；使用整型数，其运算结果应尽量大以减少舍入噪声，但必须保证数据不会溢出；空间换时间。

FFT函数是u32* fft(s16* dat_buf, u16 buf_len)。它封装了了ST提供的STM32固件库里的void cr4_fft_1024_stm32(void *pssOUT, void *pssIN, u16 Nbin)函数。cr4_fft_1024_stm32()输入参数是有符号数，包括实数和虚数，但语音数据只包括实数部分，虚数用0填充，fft点数超出输入数据长度时，超过部分用0填充。cr4_fft_1024_stm32()输出数据包括实数和虚数，应该取其绝对值，即平方和的根。

语音特征用如下结构体定义。

```
typedef struct

{

    u16 save_sign;                      //存储标记 用于判断flash中特征模板是否有效

    u16 frm_num;                        //帧数

    s16 mfcc_dat[vv_frm_max*mfcc_num]; //MFCC转换结果

}v_ftr_tag;
```

获取MFCC的函数是void get_mfcc(valid_tag *valid, v_ftr_tag *v_ftr, atap_tag *atap_arg)。获取MFCC的一般步骤在上一章已有论述，在此介绍移植到MCU上需做的优化。

预加重的高通滤波系数为0.95，如果直接使用，则需要进行浮点运算，尽量避免，故使用y(n)=x(n)-x(n-1)×95/100。加汉明窗窗函数值如果每次都要重新计算，则需要进行三角函数运算，耗时严重，效率低下。但其数值是一定的，因此事先计算好160点的汉明窗值。存于数组中const u16 hamm[]，使用时直接读取。FFT函数直接输入ADC转换过的值-2048~2047，其输出频谱幅值过小，舍入误差较大。数据输入前需作放大处理。vc_temp[i]=(s16)(temp*hamm[i]/(hamm_top/10));此句代码在实现加窗的同时，将语音数据放大10倍。Mel三角滤波器的中心频率和数值的计算涉及到对数运算，不宜直接计算，也实现计算好的数值存于Flash中，使用时直接读取。还有其他的优化措施，详见附件代码。

void get_mfcc(valid_tag *valid, v_ftr_tag *v_ftr, atap_tag *atap_arg)函数流程如下。

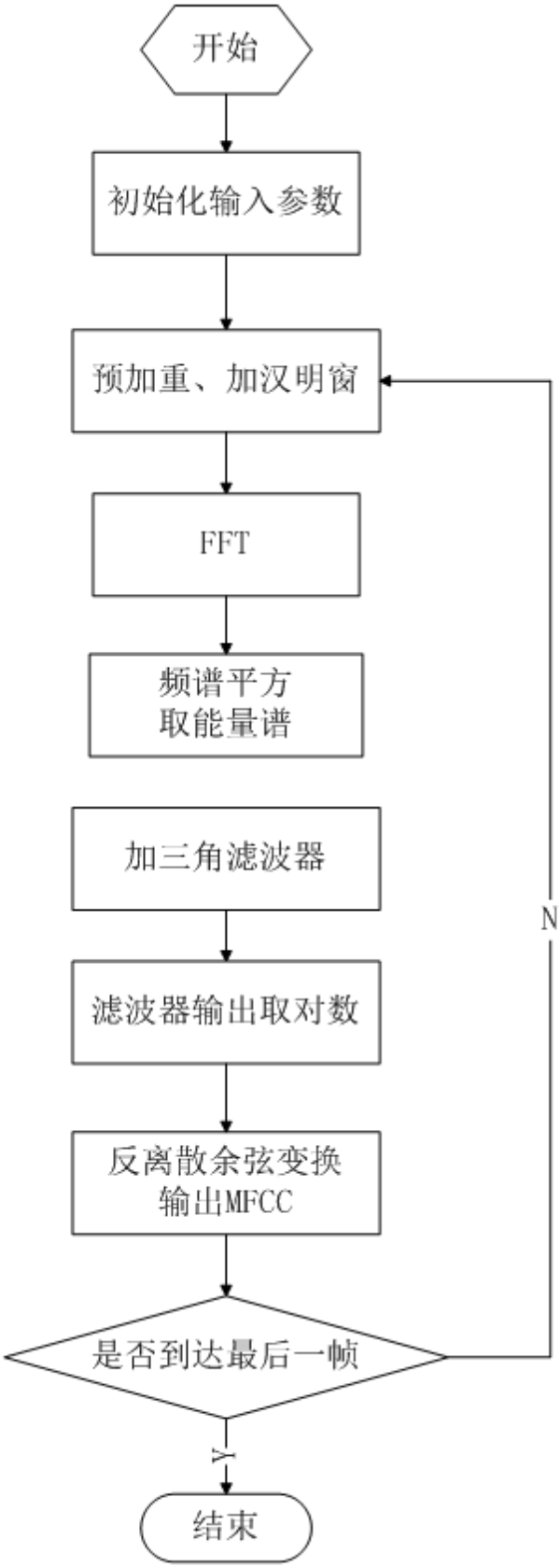


图2.12 特征提取流程

2.2.4模板训练算法设计

本设计模板训练采用冗余模板算法，即每个语音指令存储4个特征模板，识别时输入特征分别与每个特征模板相比较，匹配距离最小的，就是识别结果。这4个特征模板存储于MCU Flash后端，模

板训练时，将模板存于指定的Flash地址。为了保证保存的特征模板不被擦除或被其他代码或数据占用，需设置编译器的地址范围。

2.2.5特征匹配算法设计

本设计特征匹配算法采用 DTW（动态时间弯折）。其原理在上一章已有论述，在此不再赘述。其流程如下。

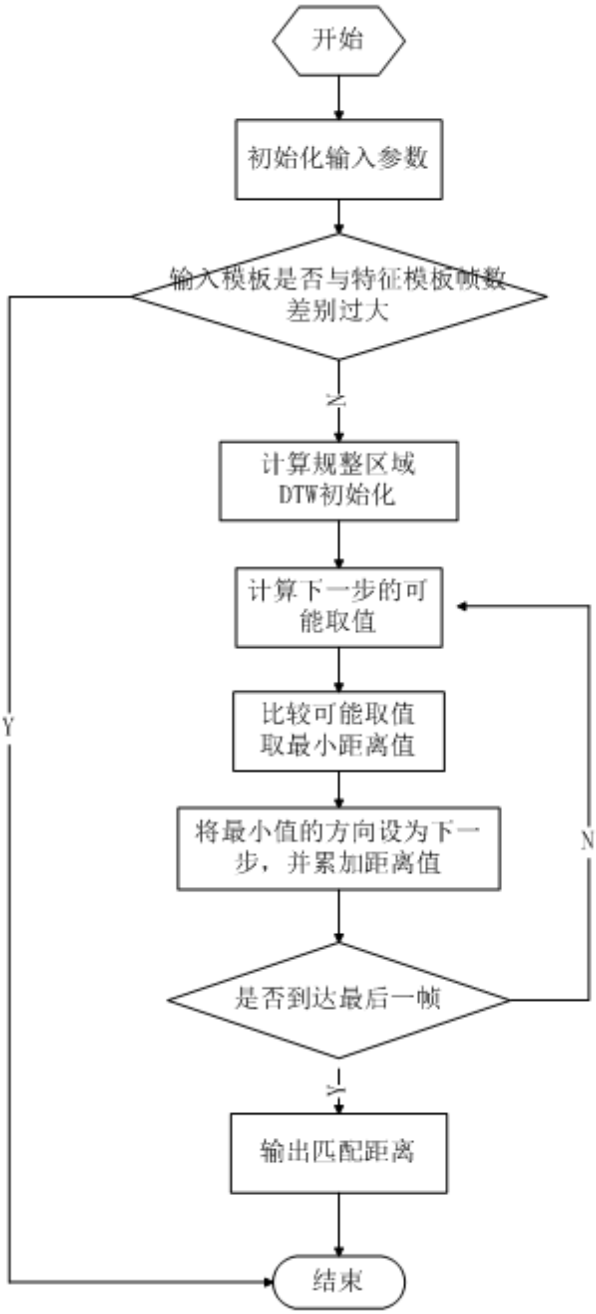


图2.13 特征匹配流程

2.2.6显示界面设计

本设计在触摸式LCD上实现了简单的GUI操作界面。能够显示中英文文本框、按钮。

最基本元素为GUI_Area，定义如下。

```
typedef struct

{

    u16 Left;         //区域离屏幕左边界的距离  像素

    u16 Top;          //区域离屏幕上边界的距离  像素

    u16 Width;        //区域宽度     像素

    u16 Height;       //区域高度     像素

    u16 BackColor;    //区域背景色

    u16 ForeColor;    //区域前景色

}GUI_Area;
```

在此基础上实现了以下函数。

```
void wait_touch(void);                          //等待屏幕点击

u8  touch_area(GUI_Area *area);                  //判断是否点击指定区域

void GUI_HideArea(GUI_Area *Area);              //隐藏区域  显示屏幕前景色

void GUI_ClrArea(GUI_Area *Area);               //清除区域  显示区域背景色
```

```
void GUI_DispStr(GUI_Area *Area,const u8 *str);    //在区域内显示字符串
```

```
void GUI_printf(GUI_Area *Area,char *fmt, ...);    //printf函数在区域内的实现
```

配合显示界面，主函数流程如下。

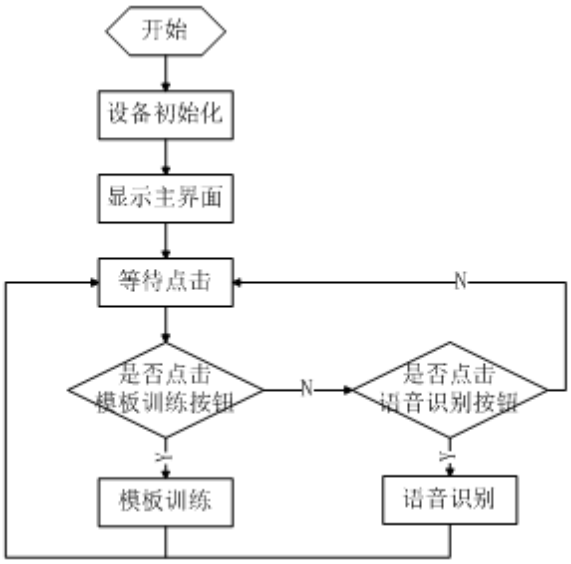


图2.14 主程序流程

第三章 系统制作及调试结果

3.1系统制作与调试

本系统的制作调试主要分为Matlab仿真、硬件调试、软件调试。

经过初步的分析设计后，Matlab中仿真算法。调节算法细节，直至能够较好地实现所需功能，再将其移植到MCU平台上。在设计制作硬件电路的同时，调试穿插进行，应用系统的硬件调试和软件调试是分不开的，许多硬件故障是在调试软件时才发现的。但通常是先排除系统中明显的硬件故障后才和软件结合起来调试，如此有利于问题的分析和解决，不会造成问题的积累，从而可以节约大量的调试时间。软件编程中，首先完成单元功能模块的调试，然后进行系统调试，整体上采用硬件调试的调试方法。

3.2制作与调试结果

经过制作与调试，实现了系统预设功能。实物图如下。

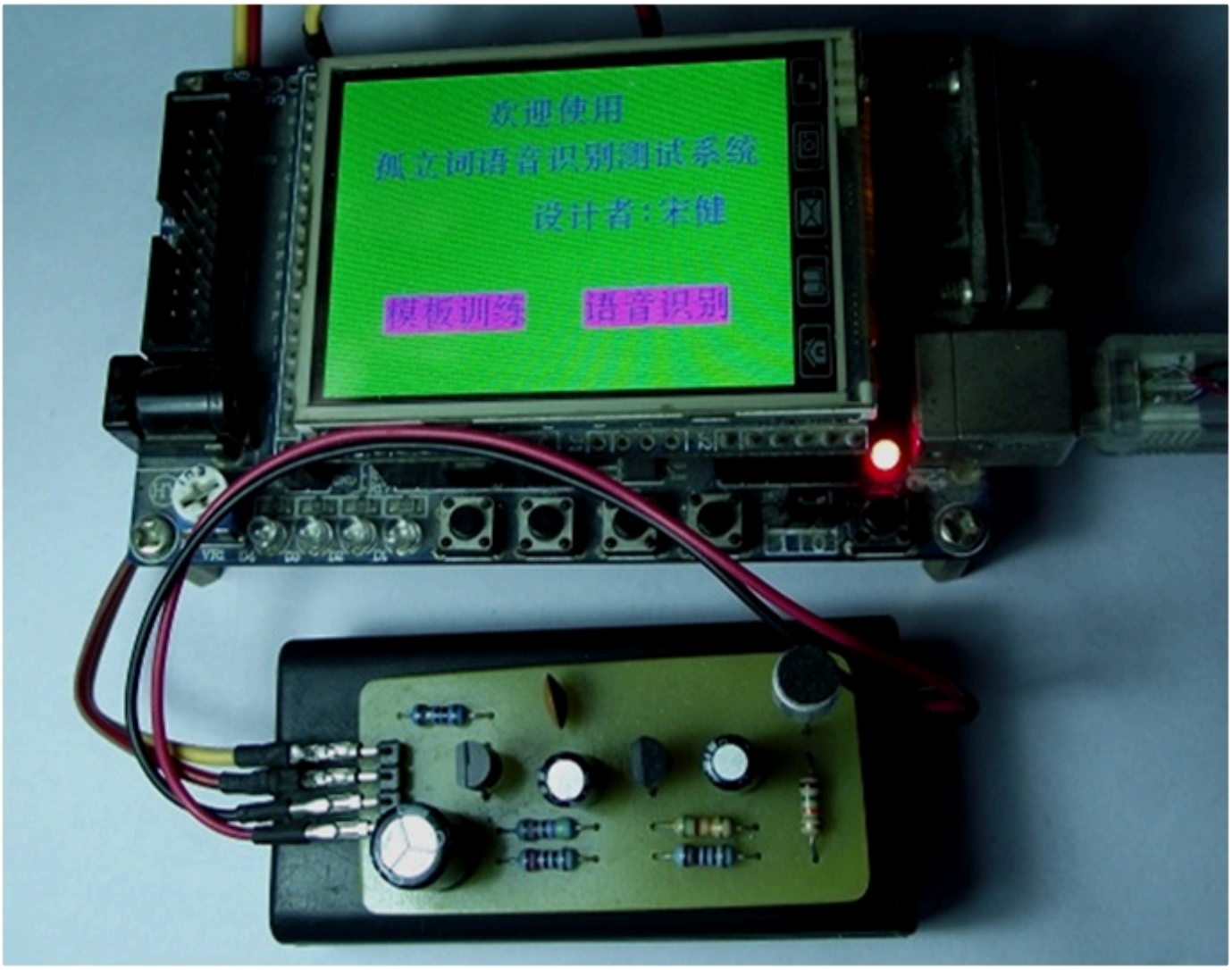


图3.1 实物图 欢迎界面

图3.2 实物图 模板训练界面

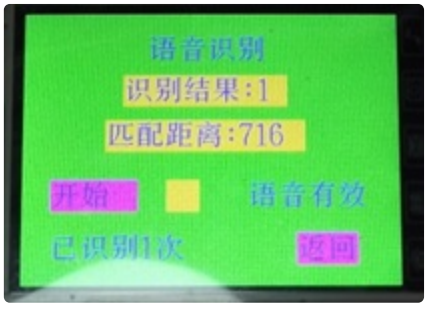


图3.3 实物图 语音识别界面

结 论

原理样机经过设计方案论证，设计了相应的硬件电路和系统软件，制作了电路原理样机并进行单机调试，结果表明，所设计的电路和软件能完成基本的测试功能。

采用STM32F103VET6单片机构建语音识别系统，通过此系统对语音信号进行采集、前端放大、AD转换、预处理、MFCC特征提取、模板训练、DTW特征匹配的一系列步骤，完成孤立词语音识别的预期目标。

本设计目前也存在一些不足，例如语音信号采集模块的动态范围不足，当说话声音较大或较小时，会出现无法识别的现象，需加上自动增益控制功能。语音识别时，录音控制不方便，最好能够改进为完全通过语音控制。特征模板仅仅用12阶MFCC略显不足，可添加MFCC一阶差分。

参考文献

[1] 韩纪庆、张磊、郑铁然. 语音信号处理. 北京：清华大学出版社[M]，2004年9月

[2] 董辰辉、彭雪峰. MATLAB 2008 全程指南. 北京：电子工业出版社[M]，2009年3月

[3] 张雪英. 数字语音处理及MATLAB仿真. 北京：电子工业出版社[M]，2011年7月

[4] 赵力. 语音信号处理 第2版. 北京：机械工业出版社[M]，2011年6月

[5] 陈程. 机载环境下的语音识别技术及实现［J］.电子科技大学硕士学位论文,2008年5月

[6] 蒋子云. 基于ARM嵌入式孤立词语音识别系统研究与实现［J］. 中南大学硕士学位论文, 2009年5月

[7] 白顺先. 汉语孤立字语音识别技术的研究［J］. 西南交通大学硕士学位论文, 2009年6月

[8] 童红. 孤立词语音识别系统的技术研究［J］. 江苏大学硕士学位论文, 2009年6月

[9] 汪冰. 小词汇非特定人的孤立词语音识别系统的研究与设计［J］. 广东工业大学硕士学位论文, 2008年5月

[10] 黄振华. 孤立词识别中的说话人归一化技术［J］. 上海大学硕士学位论文, 2009年1月

开源

代码链接：https://github.com/gk969/stm32-speech-recognition