

中科院声学所EPD引擎

详细设计说明

目录

[1.概述 5](#_Toc80350088)

[**1.1简介** 5](#_Toc80350089)

[1.2 读者对象 5](#_Toc80350090)

[1.3术语定义 6](#_Toc80350091)

[1.4 参考资料 6](#_Toc80350092)

[2.总体设计 7](#_Toc80350093)

[3.数据结构设计 8](#_Toc80350094)

[4.模块设计 11](#_Toc80350095)

[4.1 频域能量判断模块 11](#_Toc80350096)

[4.2 浊音（谐波）判断 16](#_Toc80350097)

[4.3起点细查找 18](#_Toc80350098)

[3.4终点判断 19](#_Toc80350099)

[5. 接口设计 20](#_Toc80350100)

[5.1、接口调用流程 20](#_Toc80350101)

[5.2、初始化引擎 21](#_Toc80350102)

[5.3、引擎参数重置 21](#_Toc80350103)

[5.4、EPD切分 22](#_Toc80350104)

[6.可靠性设计 23](#_Toc80350105)

[7. 扩展性设计 24](#_Toc80350106)

[8.维护性设计 25](#_Toc80350107)

[9.易用性设计 26](#_Toc80350108)

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 1.概述

1

**1.1简介**

EPD的主要功能：前端功能，对输入的原始语音进行有效音截取，使得输入到识别引擎中的语音为有效音从而减少噪音对识别引擎的干扰。本引擎主要依据音频信号的时域能量、频域能量以及谐波特征，来对音频信号中的语音以及非语音进行区分。

## 1.2 读者对象

本文档是对EPD引擎的内部代码流程和相关技术点的整体说明，因此读者最好满足如下几点条件后再读此文档：

* 有一定的C和C++语言基础
* 有基本的音频信号处理常识

## 1.3术语定义

EPD(EndPoint Detection）：语音端点检测

## 1.4 参考资料

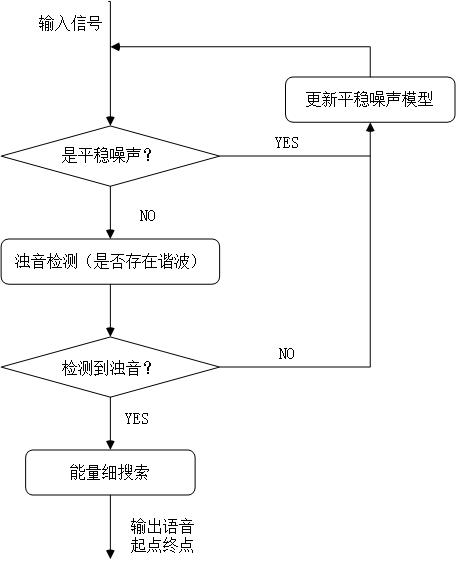
EPD算法主要是参考论文：

国雁萌, 付强, 颜永红. 复杂噪声环境中的语音端点检测[J]. 声学学报, 2006, 31(6): 549-554.

# 2.总体设计

2

EPD通过能量检测和浊音检测相结合，首先找到能量非平稳的段落，然后在其中检测浊音，并在发现浊音后搜索语音精确起止点，实现了复杂噪声环境中的语音端点检测。算法总体流程如下：



# 3.数据结构设计

3

本节主要讲解EPD类主要内容，端点检测主要的工作就是检测出原始语音中的有效语音，一方面可以减少后端模块的计算量，另一方面也可以一定程度上降低一些噪音对系统的影响。

* 实现源文件：EPD.h EPD.cpp
* 类名：C\_EPD
* 主要接口函数及关键类说明：

（1）C\_EPD(LogFile\* logFile, int SampleRate,char\* cfgFilename);

功能：

EPD模块构造函数，初始化

形参：

LogFile\* logFile：日志文件句柄

int SampleRate: 输入语音采样率

char\* cfgFilename：配置文件

返回值：无

（2）int doEPD(short \*buf\_in, int len\_in, int \*begin, int \*end);

功能：

分段函数，将原始语音中的有效语音起尾时间点标出来

形参：

short \*buf\_in：输入语音

int len\_in：输入语音长度

int \*begin：有效语音的起点时间点（采样点）

int \*end：有效语音的尾点时间点（采样点）

返回值：1

（3）int re\_initEPD();

功能：

重置分段函数，每检测到一句都需要重置。

形参：无

返回值：1

（4）CDetectVoice\_EPD \*detectVoice;

功能：

检测谐波的类，用来检测语音中是否有谐波，详情可参考4.2浊音判断章节。

（5）CDetectEnergy\_EPD \*detectEnergy;

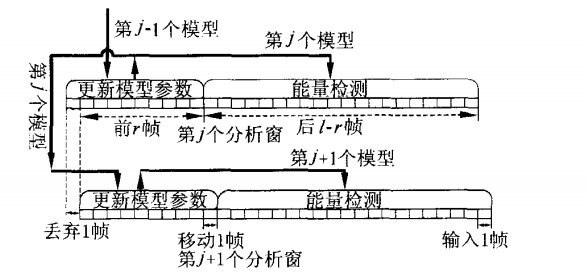
功能：

检测语音频域能量的类，用来计算语音的频域能量以及自适应计算频域能量阈值，详情可参考4.1频域能量判断模块。

# 4.模块设计

4

## 4.1 频域能量判断模块

首先分帧之后，由L帧语音构成一个分析窗，根据其中前r帧信号估计模型参数，并设定能量阈值C:\Users\thinkit\AppData\Local\Temp\1529025393(1).png，(其中C:\Users\thinkit\AppData\Local\Temp\1529025433(1).png是灵敏度系数，C:\Users\thinkit\AppData\Local\Temp\1529025457(1).png，可根据具体情况适当调节)，则可根据阈值，检验后l—r帧信号的能量是否平稳。每输入一帧信号，分析窗就后移一帧，并校正分布模型，重新计算C:\Users\thinkit\AppData\Local\Temp\1529025500(1).png。之后根据特定算法[1]对该模型进行初始化以及更新，这样就得到了一个背景噪声的模型。之后根据每一个分析窗的信号谐波特征，选取5条相邻的谐波，之后根据这五条谐波是否满足能量大于阈值的要求，且都大于噪声，如果满足则判定语音信号的起点在该分析窗中。大致流程如下图：

* 实现源文件：DetectEnergy.h DetectEnergy.cpp
* 类名：CDetectEnergy\_EPD
* 主要函数说明：·

（1）CDetectEnergy\_EPD(int SampleRate);

功能：

频域能量计算构造函数，初始化变量。

形参：

int SampleRate：输入语音采样率

返回值：无

（2）void InitNoise();void ReInit();

功能：

分别是初始化噪声模型以及重置能量模块。

形参：无

返回值：无

（3）int InitSensitivity(int FrameCount\_in);

功能：

初始化能量模块自适应灵敏度。

形参：

int FrameCount\_in：输入语音帧数

返回值：1

（4）double GetNewEnergy(float \*BinEnergy\_in, int FrameCount\_in);

功能：

计算各个子带的能量并存储,并返回该帧的总能量。

形参：

float \*BinEnergy\_in：输入语音频域能量

int FrameCount\_in：输入语音帧数

返回值：

该帧的总能量

（5）void UpdateNoise(int FrameCount\_in);

功能：

根据语音情况更新噪音均值、方差（寻找起点时）。

形参：

int FrameCount\_in：输入语音帧数

返回值：无

（6）void UpdateNoiseInSpeech(int FrameCount\_in);

功能：

根据语音情况更新噪音均值、方差（寻找尾点时）。

形参：

int FrameCount\_in：输入语音帧数

返回值：无

（7）int DetectRoughStart(int FrameCount\_in);

功能：

根据频域能量确定语音粗起点。

形参：

int FrameCount\_in：输入语音帧数

返回值：

语音起点

（8）int DetectRoughStart(int FrameCount\_in);

功能：

根据频域能量确定语音粗起点。

形参：

int FrameCount\_in：输入语音帧数

返回值：

语音起点

（9）int DetectRoughEnd(int FrameCount\_in);

功能：

根据频域能量确定语音尾点。

形参：

int FrameCount\_in：输入语音帧数

返回值：

语音尾点

## 4.2 **浊音（谐波）判断**

浊音是由周期性声门脉冲激励声道而产生的。对纯净浊音信号加窗，则加窗信号在基音及谐波频率存在能量峰起。因为基频变化较缓慢，此特性在窄带语谱图上表现为一系列均匀平行亮线。由于多数噪声不具有这种谐波特性，所以通过在输入信号中检测浊音谐波特性，可以检出浊音信号。

基音和谐波集中了浊音的主要能量，所以浊音在噪声干扰下仍能保持清晰的谐波特性。但是，语音的频域能量分布随基音和共振峰而变化，噪声的频域能量则随环境而改变。因此，在实际噪声环境中，语音的各频率分量所受干扰程度不同，并且处在不断变化中，浊音谐波特征较清晰的频段也随之变化。本算法根据人类浊音特性，在较宽的频率范围中自动搜索最清晰的谐波特征，并结合前后信息判断浊音是否存在。这样不仅可避免低频噪声和基音倍频的干扰，而且能自动适应噪声类型和强度的变化。

* 实现源文件：DetectVoice.h DetectVoice.cpp
* 类名：CDetectVoice\_EPD
* 主要函数说明：·

（1）CDetectVoice\_EPD(int SampleRate);

功能：

谐波判断构造函数，初始化变量。

形参：

int SampleRate：输入语音采样率

返回值：无

（2）int DetectVoice(float \*AllBinEnergy\_in, int VoiceNumber\_in);

功能：

接口函数，接受数据，返回是否有谐波。

形参：

float \*AllBinEnergy\_in：输入语音各频带能量

int VoiceNumber\_in：能组成谐波最短的语音帧数

返回值：

存在谐波返回1，否则返回0

（3）void UpdateBuf();

功能：

更新谐波检测的内部频带能量缓存。

形参：无

返回值：无

（4）void MatchPitch();

功能：

对当前一帧的局部极值做基本的匹配，看是否可以基本符合某些谐波。

形参：无

返回值：无

（5）int Find(int VoiceNumberRequest);

功能：

连接匹配出来的局部极值，如果超过阈值则认为语音中存在谐波。

形参：

int VoiceNumberRequest：谐波帧数阈值

返回值：

存在谐波返回1，否则返回0

## 4.3起点细查找

起点细查找，若在分析窗内检测到浊音，则结合能量检测的结果，通过能量细搜索寻找语音起点。从b开始，向前搜索语音起点帧b0，它满足如下条件：从b0的下一帧开始，信号能量逐帧递增。由于语音能量较大且变化迅速，所以语音起点附近的能量起伏很大，不会引起噪声模型的变化，能量阈值也不会升高。由于清音位于浊音前，且浊音能量远大于清音，所以根据前后h帧平均能量所找到的起点b一般处在比阈值能量更低的位置，即清音段或清浊音交界处，满足上述条件的b0就是语音起点。

## 3.4终点判断

之后进行终点判决，若分析窗中任何一段长度为h帧的连续信号都不满足每帧能量大于C:\Users\thinkit\AppData\Local\Temp\1529025500(1).png的条件（即背景噪音），则判定该分析窗的第1帧为语音终点。

# 5. 接口设计

5

### 5.1、接口调用流程

EPD引擎的流程如下:

1. 初始化引擎,分配引擎运行所需要资源；
2. 读入语音数据，分包送入引擎并进行EPD切分，实时返回语音起尾点，每找到一组起尾点对引擎进行重置操作，重置后数据需要从当前尾点之后重新开始送入；

### 5.2、初始化引擎

|  |  |
| --- | --- |
| 函数原型 | **C\_EPD(int SampleRate,char\* cfgFilename);** |
| 功能 | 加载配置文件 |
| 参数说明 | **SampleRate**：输入音频采样率（默认8kHz，只支持PCM格式） |
| **char\* cfgFilename：**配置文件名称 |
| 返回值 |  |
| 说明 | 确保该在其它函数之前调用，一个进程仅能调用一次 |

### 5.3、引擎参数重置

|  |  |
| --- | --- |
| 函数原型 | **int re\_initEPD();** |
| 功能 | 重置引擎内部参数 |
| 参数说明 |  |
| 返回值 |  |
| 说明 | 需要每一句都调用一次 |

### 5.4、EPD切分

|  |  |
| --- | --- |
| 函数原型 | **int doEPD(short \*buf\_in, int len\_in, int \*begin, int \*end);** |
| 功能 | 分包接收数据并获取分段结果 |
| 参数说明 | **short\* buf：输入数据包** |
| **int len：数据包长度（short类型长度）** |
| **int\* begin：返回起点（若无起点则返回-1）** |
| **int\* end：返回尾点（若无起点则返回-1）** |
|  |
| 返回值 |  |
| 说明 | 返回的起尾点为相对于上一个尾点的相对值 |

# 6.可靠性设计

6

EPD引擎设计考虑了应用场景的复杂性和输入的多样性的情况，内部加入了大部分实际情况中会出现的异常情况的处理方案，例如超长静音、超长连续语音等异常事件的处理；并且EPD引擎测试按照严格的7\*24小时压力测试标准进行测试，同时也针对所有异常数据进行异常情况测试，保证引擎的稳定性和可靠性。

# 7. 扩展性设计

7

EPD引擎支持多路并发，保证了不同cpu数和内存大小的机器拓展需求；同时支持多种配置参数，以适应不同的应用场景；引擎本身遵循模块化设计，可以随时添加定制化功能，满足不同应用场景的需要。

# 8.维护性设计

8

EPD引擎为了方便后期的引擎维护，采用标准c/c++函数同时在较低编译器上进行编译。

EPD引擎开放了一些内部参数，保证了引擎的可配性和可维护性。常见参数如下：

**harmonicMinNumber** #要判出语音,至少需要多少谐波帧

**MAX\_SILENCE** #尾点等待的静音时间长度（以帧为单位，可以认为是0.01s为单位）。如果说话人保持静音的时间长于此参数，程序将认为检测到尾点。

**MIN\_HARMO\_LEN** #最短的谐波长度

**MeanEnergy** #语音平均最小能量，小于该能量则改判为静音

**ENERGY\_DECREASE\_START** #起点能量增益，设为1则为原始值

**ENERGY\_DECREASE\_END** #尾点能量增益，设为1则为原始值

# 9.易用性设计

9

EPD引擎采用c/c++接口，接口方便灵活，支持c/c++，java，python等变成语言调用，只需掌握基础的语言学习知识，即可自主设计功能模块和功能的实现。