

中科院声学所叠音检测引擎

概要设计说明

目录

[1.概述 5](#_Toc82528635)

[1.1简介 5](#_Toc82528636)

[1.2 读者对象 5](#_Toc82528637)

[1.3术语定义 6](#_Toc82528638)

[2.总体设计 7](#_Toc82528639)

[3. 数据结构设计 9](#_Toc82528640)

[4.模块设计 12](#_Toc82528641)

[4.1 特征提取模块 12](#_Toc82528642)

[4.2 DNN模型概率计算 13](#_Toc82528643)

[4.3起尾点判断 15](#_Toc82528644)

[5. 接口设计 16](#_Toc82528645)

[5.1 叠音检测引擎调用流程 16](#_Toc82528646)

[5.2初始化引擎 16](#_Toc82528647)

[5.3 叠音检测 17](#_Toc82528648)

[5.4 结果返回 18](#_Toc82528649)

[6可靠性设计 19](#_Toc82528650)

[7. 扩展性设计 20](#_Toc82528651)

[8.维护性设计 21](#_Toc82528652)

[9.易用性设计 22](#_Toc82528653)

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

叠音自动检测在语音技术领域中具有十分重要的地位。该技术是指从声音信号中寻找出多个说话人语音的重叠部分。进行有效的叠音自动检测能够对声音信号中多个说话人互相打断对话的情况进行检测。

## 1.2 读者对象

本文档是对叠音检测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

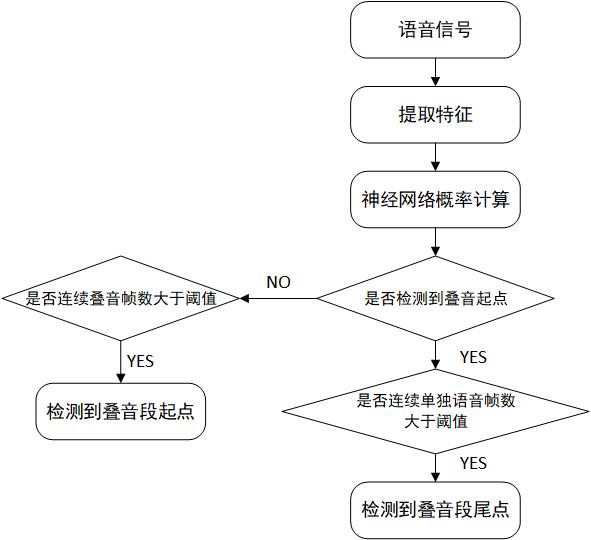
## 1.3术语定义

DNN(Deep Neural Networks）：深度神经网络

# 2.总体设计

2

叠音检测引擎是基于深度神经网络建模的方法对录入连续的语音数据进行有效的叠音自动检测并告知叠音片段语音数据的起始位置和结束位置，从而可以利用检测出有效的叠音数据为某些特定场景应用提供信息，总体流程图如下：



语音数据送入引擎后，经过“特征提取->DNN概率计算 ->结果判决”，最终得到叠音的起尾时间点。

* 特征提取：输入的语音数据分别经过预加重、分帧、加窗、FFT、梅尔滤波、对数运算、DCT后形成每帧40维的MFCC语音特征，语音特征数据将被送入DNN概率计算模块；
* DNN概率计算：在引擎初始化阶段加载DNN模型，构建神经网络；将特征送入神经网络中，计算出后验概率，送入结果判决模块；
* 结果判决：根据连续的语音帧或者叠音帧是否超过阈值，来判断叠音的开始以及结束。

# 3.模块设计

3

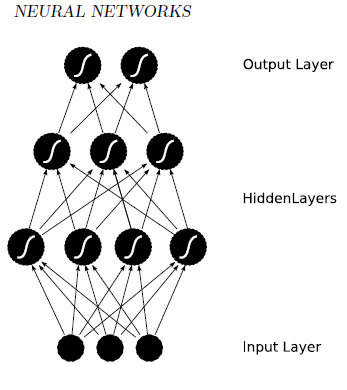
## 3.1 特征提取模块

引擎使用梅尔频率倒谱系数（MFCC）作为声学特征，梅尔刻度是一种基于人耳对等距音高变化的感官判断而定的非线性频率刻度，与人耳听觉相符，是语音识别中常用的声学特征。MFCC特征提取模块主要流程图如下，语音输入后分别经过预加重、分帧、加窗、FFT、梅尔滤波、对数运算、DCT后形成每帧40维的MFCC语音特征，特征提取流程图如下：



## 3.2 DNN模型概率计算

提取特征之后，输入DNN模型中的，模块中的DNN模型执行了一个二分类的任务，即输出某一原始语音帧为有效音帧概率和为噪音帧的概率。我们目前采用全连接的深度神经网络（之后会考虑替换其他结构的神经网络）来得到每帧的概率输出。下图是一个典型的神经网络结构示意图，语音特征先后通过输入层、隐含层、输出层，与有效音帧概率与叠音帧之间建立复杂的映射关系。在实际运行中，将每一帧语音特征联合其前后数帧的特征作为输入矢量，检测该帧的所属的类别，之后将每帧的输出概率做为一个二维向量存储下来。



## 3.3起尾点判断

利用DNN输出概率之后，原始输入语音变为了一个单人语音与叠音的一维序列。之后根据起尾点寻找算法来判断该序列中叠音的起点与尾点，其中步骤主要有：

起点寻找算法：当某一有叠音帧之后连续的叠音帧超过阈值后，则认为该帧为叠音段的起点；

尾点寻找算法：当某一非叠音帧之后连续的非叠音帧超过阈值后，则认为该帧为叠音段的尾点；

# 4. 接口设计

4

### 4.1 叠音检测引擎调用流程

引擎的流程如下:

1. 初始化引擎
2. 读入语音数据，并进行DNN判别，以链表的形式返回叠音的起尾点；

### 4.2初始化引擎

|  |  |
| --- | --- |
| 函数原型 | **bool InitDetectOverlap(const char \*systemDir, const char \*cfg);** |
| 功能 | 读取配置并初始化引擎 |
| 参数说明 | const char \*systemDir：系统目录地址 |
| const char \*cfg: 配置文件名 |
| 返回值 | 成功返回true，失败返回false |
| 说明 |  |

### 4.3 叠音检测

|  |  |
| --- | --- |
| 函数原型 | **bool Process(float \*newSet, int nFrameNum, int startFrm, int nDuration, short nFrameSize);** |
| 功能 | 接收数据并进行DNN判别 |
| 参数说明 | **float \*newSet：语音特征** |
| **int nFrameNum：语音特征帧数** |
| **int startFrm：输入语音起点绝对时间（在整通语音中）** |
| **int nDuration, short nFrameSize：特征系数** |
|  |
| 返回值 | 成功返回true，失败返回false |
| 说明 |  |

### 4.4 结果返回

|  |  |
| --- | --- |
| 函数原型 | **textResult \*GetResult();** |
| 功能 | 返回结果链表 |
| 参数说明 |  |
| 返回值 | **textResult链表头** |
| 说明 | typedef struct textResult\_{  char pText[128]; //存储输出结果  struct textResult\_ \*next;  }textResult; |

# 5可靠性设计

5

叠音检测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。

# 6. 扩展性设计

6

叠音检测引擎支持多路并发，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。

# 7.维护性设计

7

叠音检测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。

叠音检测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：

MSBias ：叠音帧概率加权

SSBias ：单人语音帧概率加权

MSpeechMiniLen：单人语音帧最小长度阈值

MNoneSpeechMiniLen：叠音帧最小长度阈值

MOverLapSpeechLen：叠音片段最大长度

# 8.易用性设计

8

叠音检测引擎采用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。