This toolbox contains primarily Matlab source codes implementing the robust speaker identification (SID) system proposed in:

> Xiaojia Zhao, Yang Shao and DeLiang Wang, "CASA-based Robust Speaker Identification," *IEEE Trans. on Audio, Speech and Language Processing,* vol.20, no.5, pp.1608-1616, 2012.

This README was written by Xiaojia Zhao in Oct'11, adapted in May'13.

There are 6 parts of this toolbox and the detailed description of each part is given below.

1. Experimental Setup (Corpus Generation)

Folder: MixtureGeneration

Description: This folder contains files that create mixtures at different signal-to-noise ratios (SNR) as well as different noisy conditions. The Ideal binary mask (IBM) is also derived.

Structure:

MixtureGeneration

|-- cochleagram.m	Program to create cochleagram
|-- CTL	Folder to store test file lists
| |-- ...	
|-- erb2hz.m	Convert equivalent rectangular bandwidth (ERB) scale to linear hertz (Hz) scale
|-- f_af_bf_cf.mat	Contains parameter values of equal-loudness functions from BS3383, "Normal equal-loudness level contours for pure tones under free-field listening conditions", table 1
|-- gammatone.m	Use a bank of gammatone filters to decompose input speech signal with equal loudness compression
|-- gammatoneNorm.m	The same as above excluding equal loudness compression and control the overall gain of the filterbank to be 1
|-- hz2erb.m	Convert linear scale to ERB scale
|-- loadData.m	Load data in HTK format
|-- loudness.m	Compute loudness level in Phons on the basis of equal-loudness functions
|-- mixSRE.m	Mix clean test files with noises at different SNR levels
|-- mixTraining.m	Mix clean speech data with noises at different SNR levels for mask estimation purposes
|-- synthesis.m	Resynthesize target speech out of mixture with CASA mask
|-- synthesisNorm.m	Resynthesize target speech out of mixture with CASA mask without equal loudness compression and thus not distort energy level
|-- writeHTK.m	Write data in HTK format

2. Feature Extraction

Folder: FeatureExtraction

Description: This folder consists of programs to extract speaker features; particular gammatone features (GF) and gammatone frequency cepstral coefficients (GFCC) as well as their delta features.

Structure:

FeatureExtraction

|-- calDelta.m	Program to calculate delta features of input
|-- cochleagram.m	Program to create cochleagram
|-- erb2hz.m	Convert ERB scale to linear scale
|-- f_af_bf_cf.mat	Contains parameter values of equal-loudness functions from BS3383, "Normal equal-loudness level contours for pure tones under free-field listening conditions", table 1
|-- fgammaton.m	Use gammatone filterbank to decompose input signal and generate GF features
|-- gammatone.m	Use a bank of gammatone filters to decompose input speech signal with equal loudness compression
|-- gen_gammaton.m	Create the gammatone filterbank
|-- gtf2gtfcc.m	Convert GF to GFCC
|-- gtfeatures.m	Main file to batch generating GFs and GFCCs for a list of files
|-- hz2erb.m	Convert linear scale to ERB scale
|-- loadData.m	Load data in HTK format
|-- loudness.m	Compute loudness level in Phons on the basis of equal-loudness functions
|-- writeHTK.m	Write data in HTK format

3. Pitch Tracking

Folder: PitchTracking

Description: This part of the toolbox is used to estimate pitches of an input signal. The pitch tracking algorithm is proposed in

> Zhaozhang Jin and DeLiang Wang, "HMM-based multipitch tracking for noisy and reverberant speech," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 19, 2011.

Pitch estimation is important because in the following mask estimation part, pitch-based features are used to classify time-frequency (T-F) units that are dominated by speech or noise.

Structure:

PitchTracking

|-- JpitchHP_2pass.jar	Java programs that are developed by Zhaozhang Jin to do pitch tracking of the input signal
|-- PitchTracking.m	Main program to batch tracking pitches of a list of files

4. Mask Estimation

Folder: MaskEstimation

Description: Programs in this folder are aimed at training multi-layer perception (MLP) classifiers to estimate CASA masks. The core idea is proposed in the following paper:

> Zhaozhang Jin and DeLiang Wang, "A supervised learning approach to monaural segregation of reverberant speech," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 17 2009.

First of all, a 6-dimensional pitch-based feature vector is derived for every single T-F unit. A MLP is trained for each frequency channel using the 6-d feature as input and IBM as desired output. Specifically, a universal MLP will be trained for each channel first across a variety of SNRs and noisy conditions. An initial mask will be derived for a mixture using universal MLPs. With this mask, input SNR is estimated and the closest SNR-specific MLPs (trained at the same time with universal MLPs) will be chosen to refine mask estimation. A folder called "RefineMask" in the section 6 actually sequentially does mask refinement on the initial masks generated by universal MLPs.

Structure:

MaskEstimation

|-- Code	
| |-- GetTestFeatures.m	Batch generating 6-dimensional pitch-based features for test data
| |-- GetTrainFeatures.m	Batch generating 6-dimensional pitch-based features for training data of mask estimation
| |-- 16 kHz	All the necessary java source and class files for 6-dimensional pitch-based feature generation for 16 kHz sampling frequency signals
| -- *.java	

| -- *.class

| |-- 8 kHz   All the necessary java source and class files for 6-dimensional pitch-based feature generation for 8 kHz sampling frequency signals

| -- *.java

| -- *.class

|-- praat_pd.m   Program to get ground truth pitch value

|-- testAll.m   Use universal MLPs to derive voiced masks for a number of files

|-- testSNR.m   Use SNR-specific MLPs to derive voiced masks for a number of files

|-- trainAll.m   Train universal MLPs for every channel

|-- trainSNR.m   Train SNR-specific MLPs for every channel

|-- UsePraat.m   Batch generating ground truth pitch contours for a number of files

5. Model Training

Folder: ModelTraining

Description: This part of the toolbox trains speaker models by adapting a universal background model (UBM). The main idea is proposed by Reynolds et al. in

> Reynolds et al., "Speaker verification using adapted Gaussian mixture models", *Digital Signal Processing*, vol. 10, 2000.

We assume you have experience of using HTK to train a gigantic UBM. If not, please refer to HTK book or anyone familiar with HTK-based speech recognition for how to train a Gaussian mixture model (GMM).

Structure:

ModelTraining

|-- AdaptGMM.m   Adapt the UBM to get a speaker model. During this adaptation process, only means of Gaussian components are adapted

|-- calMixtureLL.m   Calculate likelihood of an observation given a Gaussian component

|-- GetPosterior.m   Get posterior probability of each Gaussian component given an observation

|-- loadData.m   Load data in HTK format

|-- loadGMM.m   Load a GMM created by HTK in a HMM prototype

|-- spkrList   A list of enrolled speakers

|-- TrainGMM.m   Main file of training speaker models

6. Evaluation

Folder: testing

Description: This folder has main programs to perform robust SID. It can evaluate not only the proposed combined system, but also individual modules as well as baseline systems.

Structure:

testing

```
|-- calGMMLL_hub_AdaptGMM.m  Calculate the SID score of an input signal given a speaker model
|-- calMixtureLL.m            Calculate likelihood of a frame given a Gaussian component
|-- CTL                       Main folder to store file lists
|-- DetectSNR.m               Estimate input SNR of a mixture
|-- GetMLP.m                  Get the closest SNR-specific MLPs given an input SNR
|-- GetRunningMask.m          Use selected SNR-specific MLPs to do finer mask estimation
|-- getSID_ssn.m              Get the ground truth speaker ID
|-- gtf2gtfcc.m               Program to convert GF to GFCC
|-- loadData.m                Load data in HTK format
|-- loadGMM.m                 Load a GMM created by HTK under its HMM definition
|-- MDL                       Main folder to store model files (speaker models, UBMs, etc.)
|-- preCptLBound.m            Pre-compute lower bound of bounded marginalization to speed up
                              score calculation
|-- readmodelfile_diagonal.m  Read a speech prior model / UBM with diagonal covariance matrix
                              provided no Gaussian component is dropped by HTK during training
|-- readPrior.m               Read a speech prior model / UBM with diagonal covariance matrix
                              provided some Gaussian components are dropped by HTK during
                              training
|-- recon_gtf.m               Perform reconstruction on GF
|-- RefineMask
|   |-- WriteMask.m           This program batch refines initial masks created by universal MLPs
|-- RmBottom10.m              Remove the first 10 channels of GF as they correspond to frequency
                              range below 200 Hz. This processing is for telephone recordings and it is
                              unnecessary for high quality microphone recordings
|-- speakerID_hub_AdaptGMM_Comb.m   Main file to get SID scores using the proposed combined
                              system
|-- speakerID_hub_AdaptGMM.m        Main file to get SID scores using individual modules
|-- spkrList                  A list of enrolled speakers
|-- testBaseline
|   |-- main_etsi.m           Main file of baseline system using ETSI_D features
|   |-- main_gfcc.m           Main file of baseline system using GFCC features
```

```
|   |-- main_mfc_12.m          Main file of baseline system using 12-dimensional MFCC features
|   |-- main_mfc.m             Main file of baseline system using 22-dimensional MFCC features
|-- testClean
|   |-- main_ETSI.m            Main file of a SID system using clean ETSI_D features
|   |-- main_GFCC.m            Main file of a SID system using clean GFCC features
|   |-- main_GTF.m             Main file of a SID system using clean GF features
|   |-- main_MFC_12.m          Main file of a SID system using clean 12-dimensional MFCC features
|   |-- main_MFC.m             Main file of a SID system using clean 22-dimensional MFCC features
|-- testCombination
|   |-- main_Combination.m     Main file of the proposed combined system
|-- testIndividualModule
|   |-- main_REC.m             Main file of the proposed reconstruction module
|   |-- main_MAR.m             Main file of the proposed marginalization module
|-- var2ccVar.m                Program to convert spectral uncertainties to cepstral uncertainties
```

7. How to use this toolbox?

Each part of this toolbox is relatively independent. You can easily find main programs in each folder to start with. Be careful with the programs as there are places that you need to make necessary changes such as designated folders. These programs have been reorganized based on our initial code so there could be some minor bugs, though we have done a quick sanity check to fix any obvious errors. Please feel free to contact us if you have any problem using the code.