

# *Reference System based on speech modality*

## *ALIZE/LIA\_RAL*

version 2.0 - 01/02/2007



This reference system is based on the ALIZE software platform and uses tools developed by LIA. This paper describes this system and explains how to use it in practical to make an automatic speaker recognition experiment. This paper is divided into four parts. In the first part, we present how to install the system. Then, in the second part, the databases used and the protocol which have been defined are described. In the third part, we move on to the explanation of the method. And finally, the scripts used to launch a NIST experiment and the results obtained in authentication are shown.

## *I. Installation*

### *1. Software Requirements*

The full system has been tested under Linux (Red Hat 3.3.3-7). For the compilation step, we have used g++ 3.3.3.

The softwares used to make a text-independent speaker authentication are :

- **ALIZE** : a software platform that aims to facilitate application development in the speech and speaker recognition field.
- **LIA\_RAL** : a package that aims at providing automatic speaker detection related programs based on the ALIZE toolkit. It contains three sub-packages : LIA\_SpkDet related to speaker detection, LIA\_Seg related to speaker diarization and acoustic segmentation and LIA\_Utills which contains uncategorized programs like score fusion or score calculation.  
A library containing useful methods is also provided in LIA\_SpkTools.
- **SPro** : a toolkit which provides runtimes commands implementing standard feature extraction algorithms for speech and speaker recognition applications.
- **SPHERE** : a library used to be able to read sphere format (.sph).

The versions of each software tested are summarized in table (1).

software	version
ALIZE platform	1.04
LIA_RAL package	1.2
SPro toolkit	4.0.1
SPHERE library	2.6

TAB. 1 – software used for automatic speaker recognition

## 2. Downloading

The source code of each software can be downloaded at :

- ALIZE :  
<http://www.lia.univ-avignon.fr/heberges/ALIZE/>
- LIA\_RAL :  
[http://www.lia.univ-avignon.fr/heberges/ALIZE/LIA\\_RAL/index.html](http://www.lia.univ-avignon.fr/heberges/ALIZE/LIA_RAL/index.html)
- SPro :  
<http://www.irisa.fr/metiss/guig/spro/download.html>
- SPHERE :  
[http://www.nist.gov/speech/tools/sphere\\_26atarZ.htm](http://www.nist.gov/speech/tools/sphere_26atarZ.htm)

Download these archives in your home directory */home*. Then, you have to decompress these four tar files as follows :

```
>tar zxvf alize-1.04.tar.gz
>tar zxvf LIA_RAL_v1.2.tgz
>tar zxvf spro-4.0.1.tar.gz
>zcat sphere_2.6a.tar.Z | tar xvf -
```

At the end of this step, you have four folders (and the tar files) in your home directory : *alize-1.04*, *LIA\_RAL\_v1.2*, *spro-4.0* and *nist*.

## 3. Compilation

### a. ALIZE

In a bash shell, you have to make the following steps in order to generate the ALIZE library :

```
>cd alize-1.04
>touch ltmain.sh
>./autogen.sh
```

After that, the ALIZE library *libalize.a* can be found in */home/alize-1.04/src/*.

### b. LIA\_RAL

To install the LIA\_RAL package, follow these steps :

- enter in the LIA\_RAL directory :
  - >cd LIA\_RAL\_v1.2
- copy the *config.txt.ex* file as *config.txt* :
  - >cp config.txt.ex config.txt
- read the *config.txt* file :
  - >gedit config.txt
- fill in the *config.txt* file with your own parameters :
  - ALIZE\_DIR=/home/alize-1.04
  - ALIZE\_DIR=/home/alize-1.04
  - USER\_BIN\_DIR=/home/alize-1.04/bin
  - USER\_LIB\_DIR=/home/alize-1.04/lib
  - TAGS\_DIR=/usr/bin/ctags
- then, proceed as the following to generate dependencies, compile and install the packages :
  - >make deps
  - >make
  - >make install

At the end of these steps, the executables and the library *libliatools.a* are respectively located at */home/alize-1.04/bin/* and */home/alize-1.04/lib/*.

### ***c.SPHERE***

The installation of the SPHERE library can be accomplished with the next commands :

- enter in the *nist* directory :
  - >cd nist
- execute the installation script :
  - >sh src/scripts/install.sh

Before installation, this script needs information concerning the computing environment on which this package is being compiled. The install script has definitions for several operating system environments on which this package may be installed. The list of these OS environments is reminded below :

- 1 :Sun OS-4.1.[12]
- 2 :Sun Solaris
- 3 :Next OS
- 4 :Dec OSF/1 (with gcc)
- 5 :Dec OSF/1 (with cc)
- 6 :SGI IRIX :cc -ansi
- 7 :HP Unix (with gcc)
- 8 :HP Unix (with cc)
- 9 :IBM AIX
- 10 :Custom

As our computing environment (linux) is not listed, we choose the option 10 : custom. So, the custom installation prompts for the necessary information :

What is/are the Compiler Command?

```
>gcc
```

What is/are the Compiler Flags?

```
>-g
```

What is/are the Install Command?

```
>cp
```

What is/are the Archive Sorting Command?

```
>ranlib
```

What is/are the Archive Update Command?

```
>ar ru
```

What is/are the Architecture?

```
>UNIX
```

If the compilation step is successful, you will find the two libraries *libsp.a* and *libutil.a* in */home/nist/lib/*.

N.B : The success of the compilation step depends on the OS environment and the version of the compiler. So, if you encounter a compilation error, you will have to change the source code. For example, in your case, we had to change the code of *exit.c* (we added the code line *#include <errno.h>* and removed the declarations of variables *errno* and *sys\_errlist*).

### ***d.SPro***

To install the SPro toolkit, you have to follow these traditional steps :

- enter in the package root directory :

```
>cd spro-4.0
```
- configure the package for your system :

```
>./configure --prefix=/home/spro-4.0 --with-sphere=/home/nist
```

The option 'prefix' mentions where the toolkit will be installed whereas the option 'sphere' mentions the path to the SPHERE library.
- compile the code :

```
>make
```
- install the toolkit :

```
>make install
```

At the end of these steps, the executables and the library *libspro.a* are respectively located at */home/spro-4.0/bin/* and */home/spro-4.0/lib/*.

## II. Theory

### 1. Database

Experiments were based on NIST speaker verification data from the evaluation 2003 to 2005.

### 2. Protocol

The data protocol used to carry out the experiments consists of :

- a world set : subset from NIST03 and NIST04 data used for training the gender dependent background models.
- a development set : subset from NIST04 used to train the fusion and threshold.
- a normalisation set : subset from NIST04 for pseudo-impostors (77 male, 113 female) used for T-normalization.
- an evaluation set : primary task data (1conv-1conv) of the NIST05 speaker recognition evaluation campaign.

N.B : For the following, we suppose that the audio data have been installed at `/home/sph` location.

## III. Presentation of the method

The figure (1) represents a classical automatic speaker recognition experiment with its different steps and tools linked with each step. This part describes these different steps.

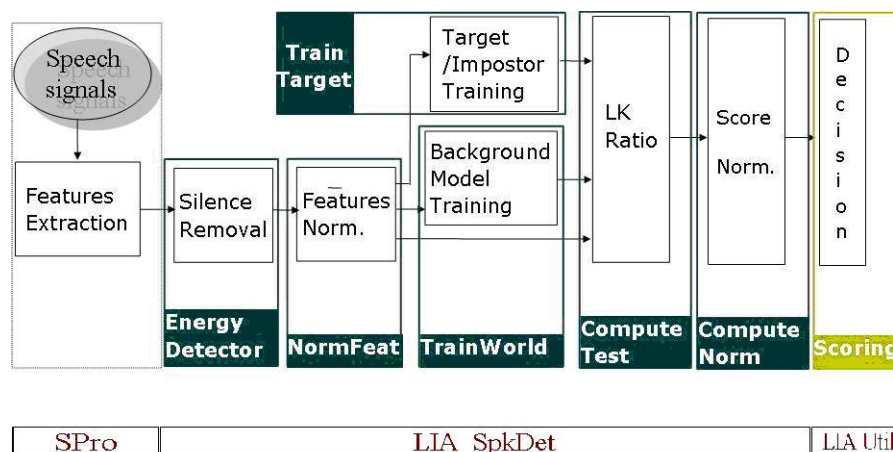


FIG. 1 – a classical ASR experiment

## 1. Features extraction

### a. Theory

The speech parametrisation is done with Linear Frequency Cepstral Coefficients (LFCC), calculated on 20ms (Hamming) windows with a 10ms shift. For each frame, a 16 element cepstral vector is computed and appended with first order deltas. The energy and delta-energy are used in addition. Bandwidth is limited to 300 – 3400Hz range.

### b. Practice

The features extraction is carried out by the filter-bank based cepstral analysis tool, *sfbcep* from SPro. The executable usage is given in the following :

```
>./sfbcep [options] inputfile outputfile
```

The *sfbcep* executable takes as input a waveform (.sph) and output filter-bank derived cepstral features (.tmp.prm). The options that we have used are described below (see tab.(2)) :

```
options= -F sphere -l 20 -d 10 -w Hamming -p 16 -e -D -k 0  
-i 300 -u 3400
```

Option	Description
-F	Specify the input waveform file format
-l	Specify the analysis frame length
-d	Specify the interval between two consecutive frames
-w	Specify the waveform weighting window
-p	Specify the number of output cepstral coefficients
-e	Add log-energy to the feature vector
-D	Add first order derivatives to the feature vector
-k	Specify the pre-emphasis coefficient
-i	Specify the lower frequency bound
-u	Specify the upper frequency bound

TAB. 2 – Description of the options used for features extraction

At the end of this step, the features are vectors (size 34) which have the next form :

$$\begin{pmatrix} \vdots \\ \cdot \\ \vdots \\ \cdot \end{pmatrix} \begin{array}{l} \left. \begin{array}{l} \vdots \\ \cdot \\ \vdots \\ \cdot \end{array} \right\} \text{Linear Frequency Cepstral Coefficients (LFCC) (size 16)} \\ \rightarrow \text{energy (size 1)} \\ \left. \begin{array}{l} \vdots \\ \cdot \\ \vdots \\ \cdot \end{array} \right\} \text{first order derivatives of the LFCC (size 16)} \\ \rightarrow \text{delta energy (size 1)} \end{array}$$

A full description of all *sfbcep* options is available with SPro documentation :

<http://www.irisa.fr/metiss/guig/spro/spro-4.0.1/spro-4.0.1.pdf>

## 2. Silence removal

### a. Theory

Once all the feature vectors have been calculated, a very important step is to decide which vectors are useful and which are not. One way of looking at the problem is to determine frames corresponding to speech portions of the signal versus those corresponding to silence. So to select frames corresponding to speech portions, we use an approach based on the log energy distribution of each speech segment :

1. The energy coefficients are first normalized using zero mean and unit variance normalization.
2. Then the normalized energies are used to train a three component GMM.
3. Finally  $N\%$  of the most energized frames are selected through the GMM, whereas the frames with lowest energy are discarded.

### b. Practice

To normalize the energy coefficients, the executable *NormFeat.exe* from LIA\_SpkDet has been used. The executable usage is given in the following :

```
> ./NormFeat.exe [options] --inputFeatureFileName inputfile
```

*NormFeat.exe* aims at processing input speech related features by applying any kind of normalization. *inputfile* is the name of the features to work with and it can be a list with a *.lst* extension. The main interesting options that we have used are described below (see tab.(3)) :

Option	Our value	Description
<code>--mode</code>	norm	Specify the kind of normalization (norm is for feature normalization on a gaussian distribution $N(0, 1)$ )
<code>--vectSize</code>	34	Specify the dimension of the features
<code>--loadFeatureFileFormat</code> <code>--saveFeatureFileFormat</code>	SPRO4	Specify the loading/saving format of feature files
<code>--loadFeatureFileExtension</code>	.tmp.prm	Specify the extension in order to load a feature file (which corresponds to the output of the features extraction step)
<code>--saveFeatureFileExtension</code>	.enr.tmp.prm	Specify the extension in order to save a feature file
<code>--featureFilesPath</code>	/home/sph/	Specify the path where to load and save feature files
<code>--writeAllFeatures</code>	true	When set to true, all features on the original file are written

TAB. 3 – Description of the main options used for energy normalization

Then, in order to select the frames with highest energy, we have used the executable *EnergyDetector.exe* from LIA\_SpkDet. The executable usage is given in the following :

```
> ./EnergyDetector.exe [options] --inputFeatureFileName inputfile
```

*EnergyDetector.exe* aims at analysing the energy component of input features by producing an output label file labeling features with highest energy. This is a typical speech / non-speech detection. *inputfile* is the name of the features whose energy has been normalized. The main interesting options that we have used are described below (see tab.(4)) :

Option	Our value	Description
--labelOutputFrames	speech	Label affected to selected frames
--labelFilePath	/home/lbl/	Specify the path where to save label files
--saveLabelFileExtension	.lbl	Specify the extension in order to save a label file
--loadFeatureFileExtension	.enr.tmp.prm	Specify the extension in order to load a feature file (which corresponds to the output of the energy normalization step)
--nbTrainIt	8	Number of EM iterations to estimate energy distribution
--mixtureDistribCount	3	Number of gaussian distributions in the mixture
--alpha	0.0	Percentage of frames selected in the central gaussian
--baggedFrameProbabilityInit	0.001	Specify the ratio between selected frames and the total number of frames used for initialisation

TAB. 4 – Description of the main options used for frame removal

A sample of an output label file is given in the figure (2). The first and second column contain the beginning and the end (in second) of the selected frames. The third column contains the label affected to selected frames.

13.75	13.88	speech
15.64	15.71	speech
16.8	16.8	speech
16.84	16.98	speech
17.08	17.9	speech
17.93	17.93	speech
17.97	18.04	speech
18.06	18.06	speech
18.09	18.09	speech
18.15	18.38	speech
...		

FIG. 2 – A sample of label file

### 3. Features Normalization

#### a. Theory

The parameter vectors are normalized to fit a zero mean and an unit variance distribution. The mean and variance used for the normalization are computed file by file on all the frames kept after applying the frame removal processing.

#### b. Practice

The executable used for the features normalization is the same than one used for the energy normalization. The only difference is that *NormFeat.exe* loads label files. So, in order to make the features normalization, *NormFeat.exe* uses only the frames with highest energy. We mention below the options which have changed in comparison with those used for energy normalization (see tab.(5)) :

Option	Our value	Description
<code>--saveFeatureFileExtension</code>	<code>.norm.prm</code>	Specify the extension in order to save a feature file
<code>--labelSelectedFrames</code>	<code>speech</code>	Specify the label to work with
<code>--labelFilesPath</code>	<code>/home/lbl/</code>	Specify the path where to load label files

TAB. 5 – Description of the main options used for features normalization

### 4. World model training

#### a. Theory

The goal of this step is to create a model which must represent the entire space of possible alternatives to the hypothesized speaker. The selected approach pools speech from several speakers and trains a single model which is called a world model (or a universal background model). The Gaussian Mixture Model is estimated using the EM (Expectation Maximization) algorithm. 2048 mixtures are used for the GMM. The log-energy coefficient is discarded to build the model.

#### b. Practice

To train the world model, the executable *TrainWorld.exe* from LIA\_SpkDet has been used. The executable usage is given in the following :

```
>./TrainWorld.exe [options] --inputFeatureFileName inputfile
--outputWorldFilename worldfile
```

*TrainWorld.exe* aims at learning a GMM via the EM algorithm. *inputfile* is the name of the normalized features to work with and it can be a list with a *.lst* extension. *worldfile* is the name of the resulting file model. The main interesting options that we have used are described below (see tab.(6)) :

Option	Our value	Description
<code>--distribType</code>	GD	Specify the type of distribution (GD is for diagonal covariance matrix)
<code>--saveMixtureFileExtension</code>	.gmm	Specify the saving format of the distribution
<code>--loadFeatureFileExtension</code>	.norm.prm	Specify the extension in order to load a feature file
<code>--MixtureDistribCount</code>	2048	Specify the number of gaussian distributions in the mixture
<code>--mixtureFilesPath</code>	/home/gmm/	Specify the path where to save mixtures
<code>--baggedFrameProbabilityInit</code>	0.0005	Specify the ratio between selected frames and the total number of frames used for initialization
<code>--baggedFrameProbability</code>	0.1	Specify the ratio between selected frames and the total number of frames used for training
<code>--nbTrainIt</code>	25	Number of EM iterations related to <code>--baggedFrameProbability</code>
<code>--nbTrainFinalIt</code>	4	Number of EM iterations with no <code>--baggedFrameProbability</code>
<code>--featureServerMask</code>	0-15,17-33	Used to select a subset of features in the vector. Example : "1,2,5-10" to select the following parameters 1, 2, 5, 6, 7, 8, 9, 10.
<code>--vectSize</code>	33	Specify the dimension of the features
<code>--normalizeModel</code>	true	Apply a $\mathcal{N}(0, 1)$ mapping at the end of the learning process

TAB. 6 – Description of the main options used for world model training

## 5. Target model training

### a. Theory

In order to train the target model, we use the following approach : the speaker model is derived by adapting the parameters of the background model using the speaker’s training speech and the maximum a posteriori (MAP) estimation. 2048 mixtures are used for the GMM. As previously, the log-energy coefficient is discarded to build the model.

### b. Practice

To train the target model, the executable *TrainTarget.exe* from LIA\_SpkDet has been used. The executable usage is given in the following :

```
> ./TrainTarget.exe [options] --targetIdList inputfile
--inputWorldFilename worldfile
```

*TrainTarget.exe* aims at training target speakers by adapting a world model via a MAP method. *worldfile* is the name of the world model used for adaptation. *inputfile* is the input list which associates a name of model to each feature filenames. A sample of a such list is given in the figure (3).

2003	tbvn
2007	tigu
2023	tfkq
2026	tgqg
2035	tclg
2048	tiba
2058	tcte
...	

FIG. 3 – A sample of input list (model on the first column and input feature filenames on the second)

The main interesting options that we have used are described below (see tab.(7)) :

Option	Our value	Description
<code>--baggedFrameProbability</code>	0.1	Specify the ratio between selected frames and the total number of frames used for adaptation
<code>--nbTrainIt</code>	0	Number of EM iterations related to <code>--baggedFrameProbability</code>
<code>--nbTrainFinalIt</code>	1	Number of EM iterations with no <code>--baggedFrameProbability</code>
<code>--MAPAlgo</code>	MAPOccDep	Specify the adaptation method to use (see below for description)
<code>--MAPRegFactor</code>	14	Parameter used by the MAPOccDep adaptation technique

TAB. 7 – Description of the main options used for target model training

The MAP method used is the MAPOccDep approach : the random variable to estimate is computed by a linear combination of its value in the world model and its value obtained by an EM algorithm on the data. This method takes into account the *a posteriori* probability  $n$  for each gaussian. The weights of this combination are provided by the option MAPRegFactor  $r$  ( $\frac{n}{n+r}$  for the world model and  $1 - \frac{n}{n+r}$  for the clien model).

## 5. Testing

### a. Theory

The goal of this step is simply to calculate a score for each test feature vector given a target model and a background model (the score is an estimate of the probability that the test segment contains speech from the target speaker). To compute this score, we consider only the ten "top" gaussian distributions of the models (N.B : for this step, we have also discarded the log-energy coefficient of each test feature vector).

### b. Practice

To calculate the score, the executable *ComputeTest.exe* from LIA\_SpkDet has been used. The executable usage is given in the following :

```
>./ComputeTest.exe [options] --ndxFileName inputfile
--worldModelName worldfile
--outputFile outputfile
```

*ComputeTest.exe* aims at giving a score related to a test segment and a target model. *inputfile* is the file which contains the list of experiments. A sample of a such file is given in the figure (4). *worldfile* is the name of the background model. *outputfile* is the resulting score file.

```
xaiV 2017 2172 2237 2345 2441 2514 2768 3426 3487 3497 3583 3757 3937 3944 3990 4015 4373 4641 4702 4865 4910 5009 5010 5012 5081
xazu 2017 2258 2434 2489 2500 2930 2943 2989 3097 3213 3487 3494 3540 3545 3998 4326 4538 4679 4702 4769 4849 4910 5009 5322
xaiu 2017 2073 2077 2345 2495 2591 2728 3071 3142 3259 3540 3811 3885 3927 3990 4101 4702 4849 4870 4937 4948 5010 5012 5081 5284
...
```

FIG. 4 – A sample of input file (test file on the first column, models to test with on the others)

The main interesting options that we have used are described below (see tab.(8)) :

Option	Our value	Description
--computeLLKWithTopDistrib	COMPLETE	Computation with "top" distributions
--topDistribCount	10	Number of gaussians used to compute the score
--loadMixtureFileExtension	.gmm	Specify the extension in order to load a model
--loadFeatureFileExtension	.norm.prm	Specify the extension in order to load a feature file
--featureFilesPath	/home/sph/	Specify the path where to load feature files
--mixtureFilesPath	/home/gmm/	Specify the path where to load mixtures
--gender	F	gender of the input file

TAB. 8 – Description of the main options used for testing

A sample of an output score file is given in the figure (5). The columns contain in order : the gender of the target speaker, the target model identifier, a binary number (0 or 1) which informs on the sign of the score, the test file identifier and the score.

```
F 2003 0 xaey -0.0438823
F 2214 0 xaey -0.0295948
F 2266 0 xaey -0.204703
F 2288 0 xaey -0.0819359
F 2519 1 xaey 0.0526875
...
```

FIG. 5 – A sample of output score file

## 6. Score Normalization

### a. Theory

The last step in speaker verification is the decision making. This process consists in comparing the likelihood resulting from the comparison between the claimed speaker model and the incoming speech signal with a decision threshold. If the likelihood is higher than the threshold, the claimed speaker will be accepted, else rejected.

The tuning of decision threshold is very troublesome in speaker verification. Indeed, its reliability cannot be ensured while the system is running. This uncertainty is mainly due to the score variability between trials, a fact well known in the domain.

So, score normalization has been introduced explicitly to cope with score variability and to make speaker-independent decision threshold tuning easier.

The normalization technique used here is *tnorm* (which uses impostor scores to normalize client scores).

### b. Practice

To apply score normalization, the executable *ComputeNorm.exe* from LIA\_SpkDet has been used. The executable usage is given in the following :

```
>./ComputeNorm.exe [options] --tnormNistFile impostorfile
--testNistFile testfile
--outputFile outputfile
```

*ComputeNorm.exe* aims at applying normalization technique on speaker detection scores. It has to be noted that the impostor trials and related impostor scores, necessary for the *tnorm* normalization technique, are not performed by this executable (*ComputeTest.exe* has been used for this task). *impostorfile* is the impostor score file in NIST format (performed by *ComputeTest.exe*). *testfile* is the client score file in NIST format (performed by *ComputeTest.exe*). *outputfile* is the nist normalized score file. The main option that we have used are described below (see tab.(9)) :

Option	Our value	Description
<code>--normType</code>	<code>tnorm</code>	Normalization technique to apply

TAB. 9 – Description of the main options used for score normalization

A sample of a nist normalized score file is given in the figure (6). The columns are the same than those described above (but the information of the third column is not used here : the binary is always put to zero).

F	2003	0	xaey	0.832226
F	2214	0	xaey	1.08527
F	2266	0	xaey	-2.01602
F	2288	0	xaey	0.158272
F	2519	0	xaey	2.54254
...				

FIG. 6 – A sample of nist normalized score file

A full description of all *LIA\_SpkDet* tools and options is available at the following address :

[http://www.lia.univ-avignon.fr/heberges/ALIZE/Doc/DOC\\_LIA\\_SpkDet.pdf](http://www.lia.univ-avignon.fr/heberges/ALIZE/Doc/DOC_LIA_SpkDet.pdf)

## 7. Decision making

### a. Theory

The goal of this step is simply to take the decision accept or reject according to the normalized score and a decision threshold. If the score is higher than the threshold, the claimed speaker will be accepted, else rejected.

### b. Practice

To make a decision, the executable *Scoring.exe* from LIA.Utils has been used. The executable usage is given in the following :

```
>./Scoring.exe [options] --inputFile inputfile
                        --outputFile outputfile
```

*Scoring.exe* aims at taking the decision accept or reject according to the confidence score found in the score file. *inputfile* is the nist normalized score file (performed by *ComputeNorm.exe*). *outputfile* is the output decision file. The main interesting options that we have used are described below (see tab.(10)) :

Option	Our value	Description
--mode	NIST	Specify the type of scoring and output files
--threshold	2	The threshold to take a decision

TAB. 10 – Description of the main options used for decision making

A sample of an output decision file is given in the figure (7). This file contains eight fields, separated by white space and in following order :

1. The training type of the test.
2. Adaptation mode. **n** for no adaptation and **u** for unsupervised adaptation.

3. The segment type of the test.
4. The sex of the target speaker : **m** or **f**.
5. The target model identifier.
6. The test segment identifier.
7. The decision : **t** or **f** (whether or not the target speaker is judged to match the speaker in the test segment).
8. The confidence score.

```

1side n 1side f 2003 xaey f 0.832226
1side n 1side f 2214 xaey f 1.08527
1side n 1side f 2266 xaey f -2.01602
1side n 1side f 2288 xaey f 0.158272
1side n 1side f 2519 xaey t 2.54254
...
```

FIG. 7 – A sample of an output decision file

A full description of all *LIA\_Utils* tools and options is available at the following address :

<http://www.lia.univ-avignon.fr/heberges/ALIZE/Doc/DOC-LIA-Utils.pdf>

## ***IV. Experiment***

### ***1. Scripts***

A set of scripts is available to easily run experiment on the common protocol with Alize/LIA\_RAL. It requires a minimum amount of setup (specify where the sound samples are stored) and run a full NIST evaluation with a standard state of the art GMM based system. These scripts are described below :

1. For NIST 2005 data, the stereo files have to be split into two files adding an extension **.A** or **.B** to the original name (xxxx.sph split to xxxx.A.sph and xxxx.B.sph). To do that, we use the script *extract1channel.pl* as follows :

```
>./extract1channel.pl inputlist inputfile ext exe output
```

*inputlist* is the list containing the audio files which have to be split. *inputfile* is the location where the sound samples are stored. *ext* is the extension of the audio files. *exe* is the location of the executable *w\_edit* (for us, it is */home/nist/bin*). *output* is the location where the two files (**.A** and **.B**) are saved.

2. Decompress the tar file *uws\_alz\_scripts.tar.gz* in your home directory.
3. Then, you have to edit the script *setupdirs.sh* to specify where the NIST sphere files are located (only the lines 11, 12, 13 and 14 have to be changed). After that, run *setupdirs.sh*.

4. To create a new experiment, execute the script *new\_exp.sh*.
5. Go to *./exp/\*\*\*\*\*/* and edit the config files if required.
6. Run *run\_exp.sh* to launch a full Nist experiment.

## 2. Experimental results

We sum up here the main options used to make a speaker recognition experiment :

- parameter alpha (for frame removal) : 0.0
- feature normalization : zero mean and unit variance
- features : 16 LFCC + 16 first order derivatives of the LFCC + delta-energy
- number of gaussians used to build the models : 2048
- verification : this step is performed using the 10 best Gaussian components
- score normalization technique : tnorm
- database :
  - the database used for evaluation is NIST 2005
  - data used for the development come from previous campaigns (see part II.2).

The corresponding DET curve is displayed in fig.(8).

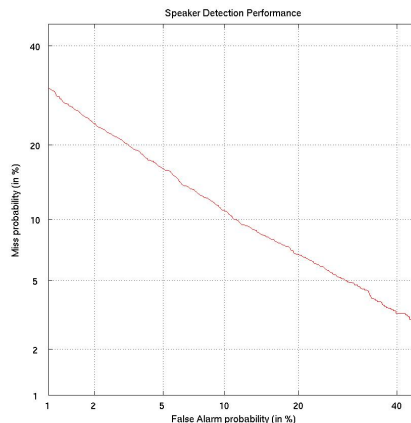


FIG. 8 – Results using ALIZE/LIA\_RAL on Nist 2005 data

## 3. Comparative results

To test the influence of the number of gaussians, we have repeated the previous experiment using only 512 gaussians to create the different models. The corresponding DET curve is displayed in fig.(9) (in comparison, the DET curve obtained using 2048 gaussians have also been displayed (in blue) on the same figure). The use of 2048 mixtures instead of 512 for the speaker modelling improves slightly the performances. Regarding the fact that GMMs with 2048 mixtures need more computation, a model with 512 mixtures is a good compromise.

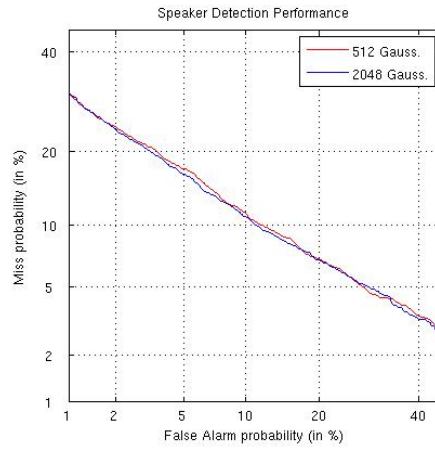


FIG. 9 – Comparative results using ALIZE/LIA\_RAL on Nist 2005 data

### *Conclusion*

The ALIZE/LIA\_RAL reference system has been described and tested on the NIST database. This reference system is made of replaceable modules which is of great interest. From now, a researcher could indeed show the improvement of his own new module (features extraction, silence removal, score normalization, ...) simply by replacing the corresponding module in the reference system.

### *Acknowledgments*

I express here my sincere gratitude to Asmaa El Hannani for their availability, collaboration and help.