Projet ANR- 11-IS02-001

MEX-CULTURE/ Multimedia libraries indexing for the preservation and dissemination of the Mexican Culture

Deliverable

Final version of Speech/Audio descriptors tools

Programme Blanc International II- 2011 Edition

A IDENTIFICATION

Project acronym	MEX-CULTURE
Project title	Multimedia libraries indexing for the preservation and dissemination of the Mexican Culture
Coordinator of the French part of the project (company/organization)	Centre d'Etude et de Recherche en Informatique et Communications – Conservatoire National des Arts et Métiers
Coordinator of the Mexican part of the project (company/organization)	Centro de Investigación y Desarrollo de Tecnología Digital – Instituto Politécnico Nacional
Project coordinator (if applicable)	Michel Crucianu : France Mireya Saraí García-Vázquez : México
Project start date Project end date	01/01/2012* 30/04/2016
Competitiveness cluster labels and contacts (cluster, name and e-mail of contact)	Cap Digital Paris-Région Philippe Roy Philippe.Roy@capdigital.com
Project website if applicable	http://mexculture.cnam.fr/

** The Mexican partners are only financed since November 2012.*

Coordinator of this report	
Title, first name, surname	*Prof. Mireya Sarai Garcia Vazquez*
Telephone	*(+52) 664 3 47 21 00*
E-mail	*freemgarciav@gmail.com*
Date of writing	*21/05/2016*

Redactors :	IPN

B OVERVIEW

B.1 INTRODUCTION

This documentation details the Application Programming Interface (API) for the Mex-Culture projet. The principal goal of these developed functions, is to give speech/audio descriptors tools a degree of abstraction for the purpose of integration across other moduls of software.

To perform the functions description developed in the framework of Mex-Culture project, we follow a very simple structure to describe functions. For that we will follow very basic rules for an API created.

According to Wikipedia:
An application programming interface (API) is a source code based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.

An API may describe the ways in which a particular task is performed.

MXC – Is the acronym used as a prefix in the names of Mex-Culture **funcitons, structures and enum**.

The reference documentation for an API is an intrinsic part of any API, and without it the API is unusable. Every aspect of the API, no matter how trivial, should be stated explicitly (source Wikipedia).

C PROPOSED METHODS FOR DESCRIPTION OF SPEECH/AUDIO CONTENT

C.1 AUDIO SEGMENTATION

In order to obtain the best performance in classifying and identifying audio content for the Mex-Culture project´s tasks, a method for audio segmentation was developed.

The objective of this method is to classify into three different categories (voice, music and silence), fragments of audio file content. This is based on two techniques called Root mean square (RMS) and Zero crossings (ZC). Employing the results of this segmentation method an improvement is done by creating new files that contain one of each categories [ED2-3].

Segmentation of an audio file into different categories like music and voice is a process that involves several stages as it shows in the diagram on figure C1.1.

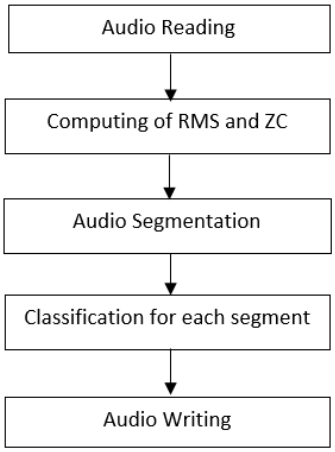

Figure C1.1. Audio segmentation stages.

The classification for each segment is done when a segment of signal is related to one of the classification types, see code in figure C1.2. First each segment enters into some conditions that verify the type assigned to the segment (1 = music, 2 = voice/speech, 3 = silence). If the type corresponds to music or voice, it first concatenates the values to the vector *finalMV* which contains music and voice. It then sees which exact classification is being handled to assign voice to vector *finalVoz* and music to vector *finalMusica.* In case that the type is silence it concatenates the values to vector *finalSilencio*.

```
if (type(i) == 1 || type(i) == 2) %option music-speech
   if pfpos(i) == 1
      finalMV = [finalMV; Y1(pfpos(i):(pfpos(i+1)-1)*windowms,:)];
   else
      finalMV = [finalMV; Y1(pfpos(i)*windowms:(pfpos(i+1)-1)*windowms,:)];
   end
   if type(i) == 2 %option speech
      if pfpos(i) == 1
         finalVoz = [finalVoz; Y1(pfpos(i):(pfpos(i+1)-1)*windowms,:)];
      else
         finalVoz = [finalVoz; Y1(pfpos(i)*windowms:(pfpos(i+1)-1)*windowms,:)];
      end
   else
      if pfpos(i) == 1 %option music
         finalMusica = [finalMusica; Y1(pfpos(i):(pfpos(i+1)-1)*windowms,:)];
      else
         finalMusica = [finalMusica; Y1(pfpos(i)*windowms:(pfpos(i+1)-1)*windowms,:)];
      end
   end
elseif type(i) == 3 %option silence
   if pfpos(i) == 1
      finalSilencio = [finalSilencio; Y1(pfpos(i):(pfpos(i+1)-1)*windowms,:)];
   else
      finalSilencio = [finalSilencio; Y1(pfpos(i)*windowms:(pfpos(i+1)-1)*windowms,:)];
   end
end %end if (type(i) == 1 || type(i) == 2)
```

Figure C1.2.- Vector creation that contains the audio segments.

Through the code, it can see that there are two ways to add the audio values to its vector, this depends on ***pfpos*** which is a previously generated vector that stores the final positions where a change is produced, and *windowms* is the window corresponding to 20 ms of audio which is the duration of the samples. These two variables are used to indicate the parts that shall be

concatenated. The first time the values are going to be added ***pfpos*** is equal to one which indicates the start of the audio, all the other times ***pfpos*** may have any other value. The difference resides in that usually it is needed to multiply the ***pfpos*** value by *windowms* to obtain the correct range of the original audio to be concatenated, but if this is done at the start, there will be a loss of data.

With the vectors for silence, music, voice and music with voice containing the appropriate values, the creation of the new audio files is done. Figure C1.3 demonstrates how wave audio file is created using the ***audiowrite*** function, it also demonstrates how the vector is verified not to be empty before creating it.

```
if isempty(finalMV) == false
    audiowrite('finalMV.wav',finalMV,FS);
end
```

Figure C1.3.- Code segment writing a .wav file in Matlab.

ERROR CATCHING

After the previous steps of updating the code, test with different audio files belonging to either the national sound library or our personal collection, are followed. Using Matlab code we modified the files to be of variant lengths of time, going from 5 seconds to 8 minutes.

While testing we found that in some cases, the files values would not be classified into the separated categories. After some analysis we found that the problem was mostly caused by short audio files which sounds are to sharp or acute for the software to recognize a possibility of change. If the program does not detect this it has a clause to automatically exit, this can be seen in figure C1.4.

```
%window size = 1 sec
%Selection of candicate windows where there is a change
j = 0;
for i=3:1-3,
   if (Pd(i) > 5 & P(i) > P(i+1) & P(i) > P(i-1)),
      j = j+1;
      pos(j) = i;
   end
end

if j == 0,
    return;
end
```

Figure C1.4.- Code where the function would exit if no change was detected.

We also noticed that the cycle goes from the third second of the audio to the total duration minus three, passing by all the seconds and verifying if there is a possible change, and if it is true *j* will increase. Therefore, if after doing this *j* is still 0, that means the code will be unable to actually analyze that file and it decides to exit at that moment. This where we also found that files with a duration of 6 seconds shall not be analyzed by the code at all. There is also a chance that even if a sharp sound passes by that part of code, a variable will be unable to be calculated and the program will crash.

To prevent this, we implemented a try module surrounding the code to prevent it from crashing, following it came a catch module for the error, this module shows us the reason

and creates a text file (if not already created) that contains a list of the files with which an error occurred. The catch module can be seen on figure C1.5.

```
catch error
    %Print the error cause
    disp(error.identifier);
    %if the file exist we append text
    if exist('Unsegmented_Files.txt','file')
        fid=fopen('Unsegmented_Files.txt','at+');
        fprintf(fid,'\n%s', info.Filename);
    %else the file is created
    else
        fid=fopen('Unsegmented_Files.txt','w');
        fprintf(fid,'%s', info.Filename);
    end
end
```

Figure C1.5.- Error catching module for the modrmszcmxc function.

In addition, in the segment code shown in figure C1.4 where the function exits, we wrote code shown on figure C1.6 to instead go directly to the catch module in figure C1.5.

```
if j == 0,
    %exception to exit when no changes are detected
    throw(MException('MyComponent:noSuchVariable',
        'No candicate windows where there is a change'))
    %return;
end
```

Figure C1.6.- Code to specify the reason of the error and it goes to the catch module.

C.2 DESCRIPTORS FOR SPEECH/MUSIC DETECTION SYSTEM

The features used in the LaBRI Speech/Music detection system are Perceptual Linear Prediction (PLP) coefficients which are widely used in speech processing. We train Gaussian Mixture Model (GMM) models for each class. Viterbi decoding is used during the test phase (ED2-1).

PLP models the human speech based on the concept of psychophysics of hearing. PLP discards irrelevant information of the speech and thus improves speech recognition rate.

C.3 MUSICAL AUDIO DESCRIPTORS

We describe the music video stream as a sequence of audio and visual features. As they are extracted from different modalities with different time resolutions, we choose to express them at a common time-scale, empirically set to a sampling period of 0.5 seconds.

We considered descriptions of the audio stream through Mel-Frequency Cepstral Coefficients (MFCC) and Chroma vectors. For the MFCC we consider the 13 first coefficients including 0th order. A Chroma vector constitute a description of the tonal content of the input signal (ED2-1).

C.4 LANDMARKS-BASED SONOROUS RECOGNITION DESCRIPTORS FOR IDENTIFICATION SONOROUS MEXICAN CONTENT

Fonoteca Nacional México makes the classification of its sonorous content in five classes: Sonorous Art, Music, Landscape Sound, Radio and Voice. These five classes are a significant part to make her catalog of sonorous files [ED2-3].

The integration of developed segmentation method voice / music / silence, allows useful information to process the audio content. The sound files for each sonorous class was processed obtaining theirs landmarks-based sonorous recognition descriptors. We have utilized the Application Programming Interfaces (APIs) used in the identification of five clases mexican sonorous content based on Fonoteca Nacional México classification [ED2-1].

C.5 LANDMARKS-BASED SPEECH FINGERPRINTING DESCRIPTORS FOR IDENTIFY MEXICAN INDIGENOUS LANGUAGES

The task in this part of Mex-Culture project is to implement the necessary resources developping tools that can help us identify an indigenous language of México among several other indigenous languages of México [ED2-3].

The integration of developed segmentation method voice / silence, allowed generate useful information to process the speech content. We have utilized the Application Programming Interfaces (APIs) used in the identification of four indigenous language of México [ED2-1].

C.6 SPEECH RECOGNITION AND SPEAKER RECOGNITION

Speech recognition is a challenging problem on which much work has been done the last decades. The extraction of the best parametric representation of acoustic signals is an important task to produce a better recognition performance. The method used to extract relevant information from each short frames of sound file is the mel-cepstrum method (Mel-Frequency Cepstrum Coefficients, MFCC). MFCC is perhaps the best known and most popular, and will be utilised here. MFCC is based on known variation of the human ear's critical bandwidth with frequency. The number of coefficients needed is 13. To increase the information of the human perception, the first and second time derivative are calculated. We have utilized the Application Programming Interfaces (APIs) used in the speech recognition [ED2-1].

The details of API documentation about the developed programs in the previous sections, can be seen in the attached document : AudioMexCulture.pdf

D REFERENCES

[ED2-1] Mid-term report on speech/audio descriptors. Mex-Culture project, 2014. ANR-CONACYT.

[ED2-3] Final report on speech/audio descriptors. Mex-Culture project, 2016. ANR-CONACYT.