FP7-ICT-2013-C FET-Future Emerging Technologies-618067

SkAT-VG: Sketching Audio Technologies using Vocalizations and Gestures

D5.5.3 Integrated system that predicts the category of imitated sound sources (Companion Document)

First Author	Gabriel Meseguer Brocal
Responsible Partner	IRCAM
Status-Version:	Final-1.0
Date:	July 30, 2016
EC Distribution:	Consortium
Project Number:	618067
Project Title:	Sketching Audio Technologies using Vocalizations and Gestures

Title of Deliverable: (Companion Document)	Integrated system that predicts the category of imitated sound sources
Date of delivery to the EC:	01/08/2016

Workpackage responsible for the Deliverable	WP5
Editor(s):	Gabriel Meseguer Brocal, Frederic Bevilacqua, Geoffroy Peeters
Contributor(s):	Gabriel Meseguer Brocal, Frederic Bevilacqua
Reviewer(s):	Davide Rocchesso
Approved by:	All Partners

Abstract	The current deliverable contains the software package of WP5
Keyword List:	Audio classifier

#	Participant Name	Short-Name	Role	Country
1	Università Iuav di Venezia	IUAV	Co-ordinator	Italy
2	Institut de Recherche et de Coordination Acoustique/Musique	IRCAM	Contractor	France
3	Kungliga Tekniska Högskolan	KTH	Contractor	Sweden
4	Genesis SA	GENESIS	Contractor	France

Document Revision History

Version	**Date**	**Description**	**Author**
Draft	28/07/2016	First Draft	Meseguer Brocal
Draft	29/07/2016	Revised Draft	Bevilacqua
Final	29/07/2016	For reviewers	Bevilacqua

Table of Contents

1 Overview **6**

2 Processing and algorithms **6**

2.1 Audio descriptors 6

2.2 Classifier 6

3 Implementation **6**

3.1 Sound Descriptors 7

3.2 Downloading and installing 8

3.2.1 System requirements 8

3.2.2 Package content 8

3.2.3 Procedure 9

Index of Figures

1 Patch overview 9

List of Acronyms and Abbreviations

DoW Description of Work

WP Work Package

KNN Work Package

GMM Gaussian Mixture Models

HMM Hidden Markov Models

HHMM Hierarchical Hidden Markov Models

MuBu multi-buffer is a container for sound and motion data. The Max MuBu library contains a set of associated tools for real-time interactive audio processing

PiPo Plug-In-Plug-Out / Programming Interface for (Afferent Stream) Processing Objects. Implemented in the MuBu for Max library

XMM Probabilistic Models for Motion Recognition and Mapping. Implemented in the MuBu for Max library

1 Overview

This deliverable focuses on the integration of the classifiers into the Max environment (Cycling74), which is the environment chosen in the SkAT-VG project to develop the applications. The initial development of the audio classifier has been conducted in Matlab (Tasks 5.1 and 5.2), and specific adaptations and choices have been made to port the classifier in Max.

The classification of vocal imitations depends on two different types of processing:

- Computing sound descriptors: transforming the input audio in a specific feature space.
- Classification: assigning a given vocal imitation to one or several categories.

Both of them have required specific developments. Specifically, we developed a group of max externals that allow users to compute specific audio descriptors (defined in Task5.1 and 5.2) and to perform audio classification based on the learning on the audio database developed in Task 4. Please note that it is straightforward to complement the proposed implementation with gesture analysis and recognition since this part is also already existing in Max using the MuBu library.

2 Processing and algorithms

2.1 Audio descriptors

The optimised set of audio descriptors proposed in D.5.5.1. are: Loudness, Inharmonicity, TotalEnergy, Noisiness, Spectral Centroid, Spectral Spread, Zero Crossing, Lpc Formant, Pitch Strength, Pitch and Spectral Peak Min.

2.2 Classifier

We have chosen to adapt the audio classifier based on Dynamic Time Wrapping (DTW) that was evaluated in Matlab. The method is based on the alignment of sound descriptors, one by one, with all the templates of an database. In order to facilitate its integration in real-time scenario, we proposed to implement a similar approach based an alignement of templates, but performed using the forward procedure, as used in the formalisation of Hidden Markov Models ([Rab89]). Specifically, we used the method proposed recently by Jules Françoise, ([Fra15, FCB12]) and implemented in the XMM library.

3 Implementation

The software we implemented makes use of *MuBu & friends* Max library. *MuBu* is a multi-buffer container for sound and motion data[1] ([SRS$^+$09]). The hierarchical memory structure of MuBu includes *tracks* and *buffers*, which perfectly matches our need. The audio descriptors

[1]MuBu project presentation, IRCAM, October 2010. `http://imtr.ircam.fr/imtr/images/MuBu-project-presentation-2010-10-20.pdf`

are stored in separate *tracks*. A track is basically an aligned contiguous array of matrices with additional data, its memory is preallocated to a given maximum size. The different audio samples are stored in *buffers*, which all share the same track structure (i.e. identical set of audio descriptors). Nevertheless, the actual memory size, and other parameters can vary between *buffers*.

MuBu also includes processing plugins called PiPo[2]. We then implemented the computing of the different sound descriptors as PiPo plugins.

Finally, MuBu includes classifier algorithm, such as KNN, GMM and Hierarchical HMM. In particular we modify the HMM implementation of XMM library[3], *mubu.hhmm* to perform the vocal imitation classification ([FSBB14]).

3.1 Sound Descriptors

The classifier uses the set of feature proposed in the D5.5.1. These PiPo modules are implemented in C++ and compiled to be used in Max. They are implemented as two different sets of PiPo plugins. The final set of descriptors is the union of both sets, which is performed in MuBu using the merge functionality.

- Specific features: LpcFormant, SpcPeakMin, PitchStrength and Pitch. For this set of descriptors, a single specific PiPo has been created (*pipo.skatdesc.mxo*). This PiPo computes each feature in parallel dealing with one signal frame at time. Two remarks:
  - The computation of the LPC formants involves the resolution of a linear equation of a polynomial of order N. In the Matlab version, roots are found using a Eigen Vector decomposition. In the PiPo version, the decomposition is carried out by first finding quadratic factors using Bairstow's method, then extracting roots from quadratics.
  - The pitch algorithm used in the Matlab version is the Swipep algorithm. In PiPo we use Yin. As a result the computation of Pitch Strength is different.
- Standard features from the Ircam Descriptors: Loudness, Inharmonicity, TotalEnergy, Noisiness, SpcCentroid, SpcSpread and ZeroCross. A specific PiPo with the tuned parameters for the SkAT-VG project has been derived from the original one (*pipo.skatircamdesc.mxo*)

There are some minor deviation from the descriptors computed in Matlab. First, as already said, the pitch and pitch strength algorithms are different. Second, there also some differences in configuration due to the fact that the Max implementation is less flexible than Matlab to adapt window size and hop size separately for each descriptor. Nevertheless, these differences produce a deviation of 4% in the accuracy of results as compared to the original classifier (DTW version), which is considered as acceptable in our context.

[2] `http://ismm.ircam.fr/pipo/`*PiPo* is a c++ API for modules processing streams of multi-dimensional data.

[3]urlhttps://github.com/Ircam-RnD/xmm

Classifier

Following the same approach as the one proposed with DTW, we create a distinct HMM model for each audio imitation of the dataset. For a new imitation O, and a particular HMM λ we can compute $\Pr(O \mid \lambda)$, which reflects the probability of belonging to that model. As the hidden states are modeled as Gaussian Mixture Models (GMM) and trained with just one signal (an imitation in our dataset) the value of $\Pr(O \mid \lambda)$ give us likelihood value that compares the input signal to the template. This measure depends on the number of states of the HMM and the covariance values of the GMM. Instead of comparing each feature individually, the whole space is embedded in the GMM that model each state. At the end, the comparison is done in a probabilistic way highly dependent on the configuration parameters.

In order to optimise these parameters, we conducted a complete set of experiments in python (using the C++ compiled version of the algorithm). The best configuration has been found using 10 hidden states per HMM, a left-right configuration and a value for the initialization of the covariance matrix of 0.5 for relative and 0.1 for absolute.

Considering the normalisation of audio descriptors, after testing different Hierarchical HMM configurations, the best result has been achieved with standardized features (zero mean and std equals 1) and their differentiation, which produces a final space with twenty-two dimensions. The standardization has been added internally in C++ but the second one (differentiation) is performed directly in max.

The complete trained model is exported as a JSON file that can be imported either in *mubu.hhmm* Max external or in the XMM library (C++).

3.2 Downloading and installing

3.2.1 System requirements

The system requires the installation of the standard MuBu and PiPo libraries v1.9 for Max 7 (available in August 2016). They can be found at:

- `http://forumnet.ircam.fr/product/mubu-en/`

3.2.2 Package content

The package can be found at `https://nuage.ircam.fr/index.php/s/kupiOfWiaK4Tu2f` and it contains the following files:

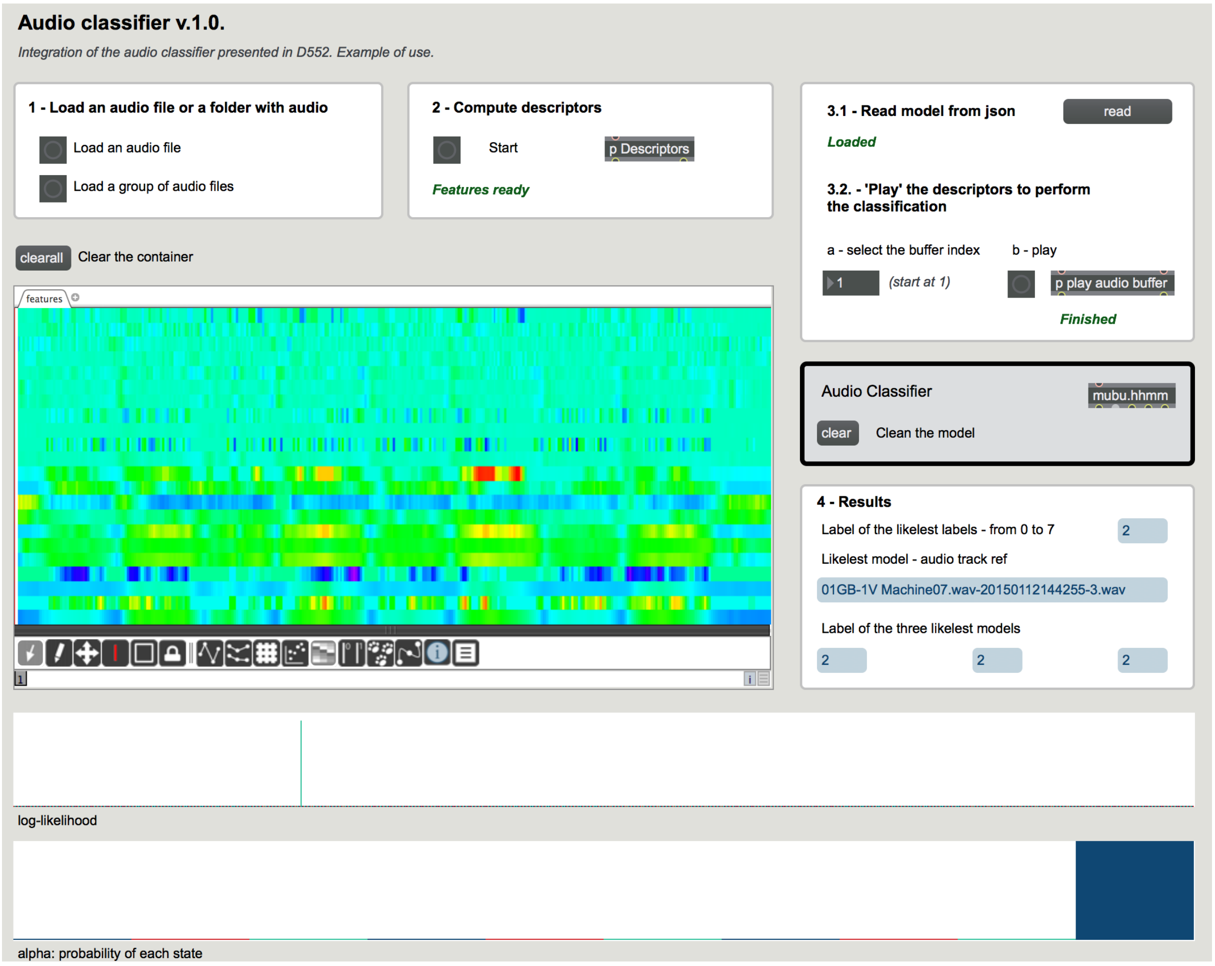
- *audio_classifier_v1.maxpat* : example patch demonstrating how to use the audio classifier.
- *hhmm.json* : HMM models trained on the SkAT-VG imitation database.
- *mubu.hhmm.mxo* : HHMM max object for performing the classification. It can import the JSON file. This version is a special adaptation for SkAT-VG project of the *mubu.hhmm.mxo* of the current MuBu release (v1.9) that uses a new API not available to the public currently (but will be in forthcoming release).
- *pipo.skatircamdesc.mxo* : PiPo for computing Ircam descriptors.

- *pipo.skatdesc.mxo* : PiPo for calculating the other specific audio descriptors.

Remark: The *mubu.hhmm.mxo*, *pipo.ircamdesc.mxo* and *pipo.skatvg.mxo* must be in your MuBu path or in the same folder than *audio_classifier_v1.maxpat*.

3.2.3 Procedure

Follow the different steps indicated on the main GUI of the *audio_classifier_v1.maxpat*.



Audio classifier v.1.0.
Integration of the audio classifier presented in D552. Example of use.

1 - Load an audio file or a folder with audio
Load an audio file
Load a group of audio files

2 - Compute descriptors
Start
p Descriptors
Features ready

3.1 - Read model from json
read
Loaded

3.2. - 'Play' the descriptors to perform the classification
a - select the buffer index
1 *(start at 1)*
b - play
p play audio buffer
Finished

clearall Clear the container

Audio Classifier
mubu.hhmm
clear Clean the model

4 - Results
Label of the likelest labels - from 0 to 7: 2
Likelest model - audio track ref: 01GB-1V Machine07.wav-20150112144255-3.wav
Label of the three likelest models: 2 2 2

Figure 1: Patch overview

It has three parts:

1. Load the audio that you want to classify in the MuBu container.
2. Compute the needed features for the classifier.
3. Perform the classification.
    - Load the *hhmm.json*. It might take a few minutes.
    - Select the buffer (audio track) from the MuBu container you want to classify.

- Play the track.

4. As the track is being played the classification is displayed in real time in box 4. The final classification is obtained once the track is completely played. The patch shows also the likelihood of all the models as well as the alpha (states in the HMM) of the likeliest model.

References

[FCB12] Jules Françoise, Baptiste Caramiaux, and Frédéric Bevilacqua. A Hierarchical Approach for the Design of Gesture-to-Sound Mappings. In *Proceedings of the International Conference on Sound and Music Computing*, Copenhagen, Denmark, 2012.

[Fra15] Jules Françoise. *Motion-sound Mapping By Demonstration*. Theses, Université Pierre et Marie Curie - Paris VI, March 2015.

[FSBB14] Jules Françoise, Norbert Schnell, Riccardo Borghesi, and Frédéric Bevilacqua. Probabilistic Models for Designing Motion and Sound Relationships. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME'14)*, London, UK, 2014.

[Rab89] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[SRS$^+$09] Norbert Schnell, Axel Röbel, Diemo Schwarz, Geoffroy Peeters, and Riccardo Borghesi. MuBu & Friends - Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP. In *Proceedings of the International Computer Music Conference (ICMC)*, 2009.