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

SkAT-VG: Sketching Audio Technologies using Vocalizations and Gestures

D5.5.2
Informed classifiers of imitations

First Author	Geoffroy Peeters
Responsible Partner	IRCAM
Status-Version:	Final-1.0
Date:	May 13, 2016
EC Distribution:	Consortium
Project Number:	618067
Project Title:	Sketching Audio Technologies using Vocalizations and Gestures

Title of Deliverable:	Informed classifiers of imitations
Date of delivery to the EC:	05/05/2016

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

Abstract	This deliverable D.5.5.2 presents the results of the tasks 5.3 and 5.4 of the WP5 of the SkAT-VG Project. The task 5.3 "Informed Classifer" relates to the automatic recognition of vocal imitations starting from the transcription of an audio signal into vocal primitives (VPs). In D.5.5.1, KTH has proposed a system to automatically transcribe an audio signal into these VPs. In the first part of the present deliverable D.5.5.2, we study the automatic recognition of the vocal imitation categories from this transcription. The task 5.4 "Fusion Classifier" relates to the automatic recognition of the vocal imitation categories using the fusion (early or late) of the information provided by the VPs transcription and the set of audio features IRCAM has proposed in D.5.5.1. In the second part, we propose an innovative method to automatically derive time and frequency "primitives", named Audio Primitives (APs) from a dataset of audio recordings. These APs then constitute the alphabet of a transcription system. To derive these APs we propose the use of the Shift-Invariant Probabilistic Latent Component Analysis (SI-PLCA). The activations over time of the VPs then serve as input for the decoding of a set of hidden Markov models representing the various vocal/gesture imitation categories. In the third part, we present the continuation of the gesture analysis of the imitation database and the formalisation of the Gesture Primitives (GPs) initially presented in D.5.5.1. A new set of features for characterising the different "frequency" behaviours is proposed, based on the application of a Non-negative Matrix Factorisation (NMF) on the different scalograms (wavelet transform). This feature space is explored for both, finding automatic gesture primitive and creating classification systems.
Keyword List:	audio, vocal, gesture, imitation, SI-PLCA, NMF, primitives, classification, recognition, hidden Markov model, scalogram

Disclaimer:

The SkAT-VG Consortium consists of the following entities:

#	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 Deliverable D5.5.2

Version	Date	Description	Author
v0.1	11/04/2016	Import from .tex template	E.M.
v0.2	21/04/2016	Import in svn repository	E.M.
v0.3	26/04/2016	Proofreading by IRCAM SAS	G.P., E.M.
v0.4	27/04/2016	Integration of IRCAM SAS/ISSM	G.P., F.B., E.M., G.M.B.
v0.5	03/05/2016	Abstract and Executive Summary IRCAM SAS/ISSM	G.P., F.B., G.M.B.
v1.0	13/05/2016	Revision after review	G.P., F.B., E.M., D.R.

Tab. of Contents

Index of Figures

List of Acronyms and Abbreviations

DoW Description of Work

EC European Commission

PM Person Months

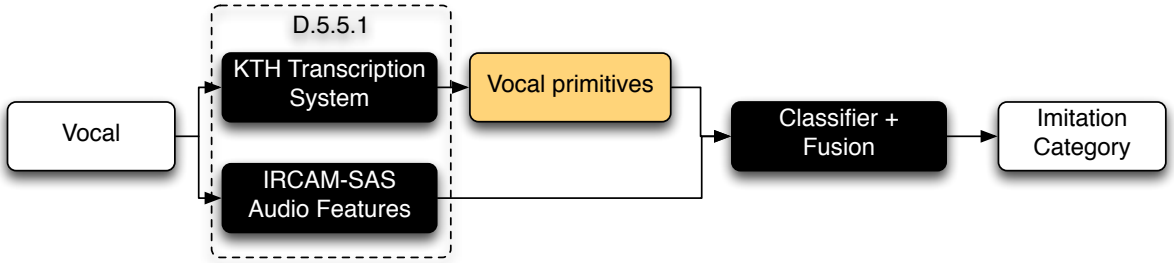
WP Work Package

1 Executive summary

The deliverable D.5.5.2 presents the results of the tasks 5.3 and 5.4 of the WP5 of the SkAT-VG Project.

The task 5.3 "Informed Classifier" relates to the automatic recognition of the imitations categories starting from the transcription of an audio signal into vocal primitives (VPs). In D.5.5.1, KTH has proposed a system to automatically transcribe an audio signal into these VPs. In the present D5.5.2 deliverable, we study the automatic recognition of the vocal imitation categories from this transcription. For each file, KTH transcription system outputs: a) the global phonetic transcription, b) the global observation probability and c) the set of global audio features used by the acoustic model. We use a, b and c as input to a soft-margin SVM-RBF classifier to recognize the categories (see Section 2.3).

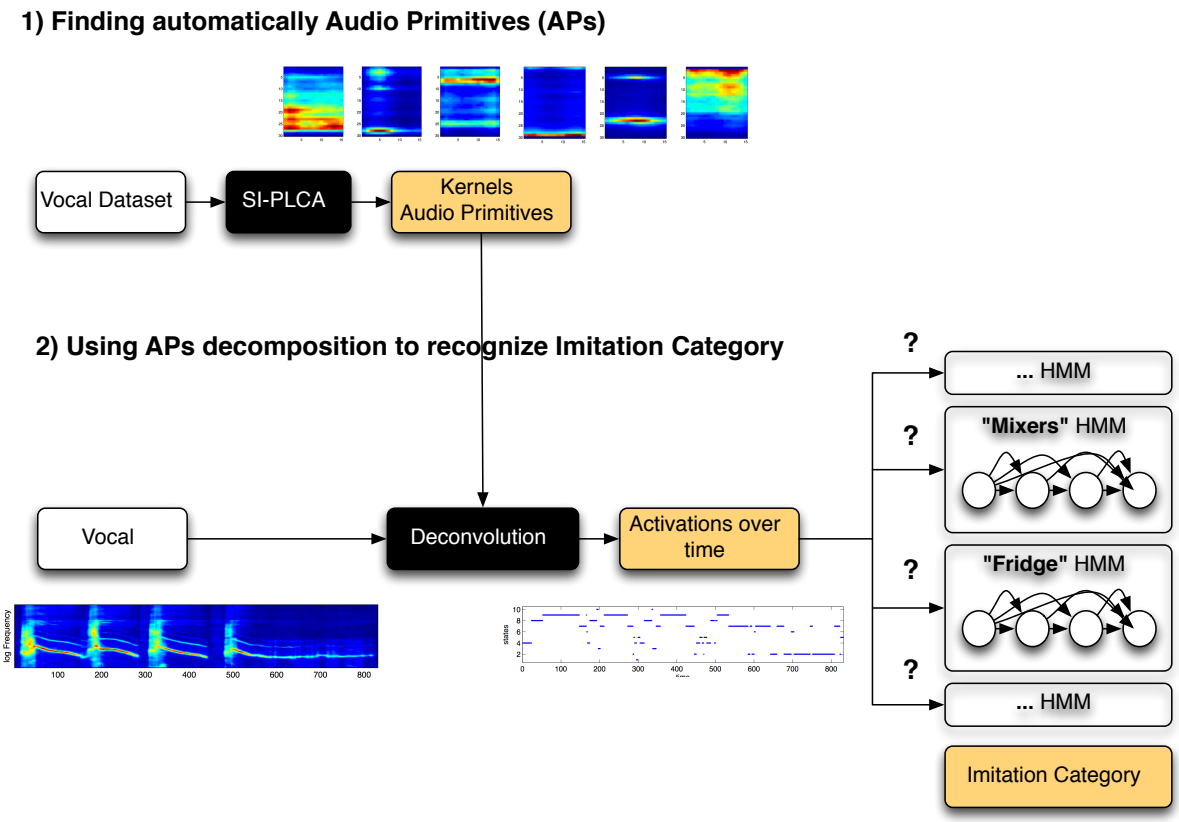
The task 5.4 "Fusion Classifier" relates to the automatic recognition of the imitations categories using the fusion (early or late) of the information provided by the VP transcription and the set of IRCAM audio features proposed in D.5.5.1. We tested the use of a, b and c in complement to IRCAM audio features to recognize the imitation categories (see Section 2.4).



The initial goal of task 5.3 was to develop a system to recognize the imitation categories inspired by speech recognition, i.e. including 1) a language model (a model that represents the set of possible sequences of phonemes used by people to imitate a given imitation category), 2) an acoustic model (a model that allows to recognize the phonemes over time from their acoustic occurrences). The idea was then to recognize the categories by decoding a set of hidden Markov models trained for each imitation category. Unfortunately, it was not possible to transcribe the vocal/gesture imitation dataset into vocal primitives. It was therefore not possible to train a language model. Also the system that transcribes an audio file into vocal primitives only performs at the file level (no transcription over time).

In order to overcome these problems, IRCAM-SAS presents in section 3 an innovative and

promising methodology that allows to automatically estimate the definition of audio primitives, to automatically get their locations in time and use them in the framework of a language/acoustic model in the form of a set of hidden Markov models. In this approach, the primitives are not manually annotated but are automatically learned using unsupervised learning algorithm from a dataset of recordings. These primitives are named "audio primitives" since they do not rely on any vocal production mechanism but only on time and frequency audio representation. For this task, we propose the use of the Shift-Invariant Probabilistic Latent Component Analysis (SI-PLCA), the kernels of which are considered as "audio primitives". SI-PLCA was chosen since it allows to fulfill the two important requirements: additivity of the primitives (so that a given time-segment can be represented as the superposition of several primitives, for example a low-frequency harmonic component with super-imposed to it a high-frequency noise sound) and shift-invariance in time and frequency (the audio signal is represented by a reduced set of primitives shifted in time and frequencies). Using a small annotated dataset, we show that such automatically derived audio primitives represent acoustically meaningful cues. We then develop a system that allows recognizing vocal imitations. For this the activations over time of the vocal primitives are considered as emissions of the hidden states of a Markov model. The succession of states is specific to each imitation category. We therefore represent each imitation category by a specific hidden Markov model.



In section 4 of this deliverable we also present the continuation of the gesture analysis of the imitation database and the formalisation of the Gesture Primitives (GPs) initially presented in

D.5.5.1. The different gestural strategies can be associated to frequency components. A new set of features for characterising gesture strategies is proposed, based on the decomposition of scalograms obtained with the wavelet transform. The decomposition is derived from a Non-negative Matrix Factorisation (NMF). Precisely, NMF allows for modeling complex gestural scalograms as a mixture of prototypical basic shapes, consistent across participants. It allow for reducing dimensionality by creating a compact and robust component space. As a result, a gesture primitive can be described by a distribution of the different components. We performed clustering analysis on the components, K-means and Gaussian Mixture Models, and found that the K-means better fits our criteria. We also evaluated classification using K-nearest neighbour and Gaussian Mixture Models.

2 Informed and Fusion classifiers

2.1 KTH phonetic transcription system outputs

In deliverable D5.5.1, KTH has proposed a system to automatically describe a vocal signal in terms of vocal primitives (VPs). These VPs represent specific vocal mechanism describing high-level phonetic content used in (European) languages. KTH has focused on three VPs: *phonation*, *slow myoelastic vibrations* and *turbulence*. Given an input vocal audio signal, KTH has proposed a system that allows to detect if one (or several) of the three VPs has been used to produce the sound. For this, KTH has used supervised learning methods applied to a dataset annotated into these three VPs. It should be noted that this dataset is different from the ones developed by IRCAM and which is annotated into vocal/gesture imitation categories.

The goal of IRCAM-SAS in task 5.2 was to develop a classifier into vocal imitations (6 categories for within the abstract family, 10 within the interaction family and 10 within the machine family) from scratch, i.e. without considering the specificities of the vocal sounds and the specific mechanisms of of the vocal productions. We therefore named it "blind" classifier. This has been described in deliverable D5.5.1.

In the opposite, the goal is here (task 5.3) to take advantage of the fine description of the vocal mechanisms used to produce the vocal sounds. This description is provided thanks to the KTH system. We therefore name it "informed" classifier in the sens that the classifier is "informed" by a higher-level description specific to the vocal signal.

Finally, we combine both "blind" and "informed" classifier into a single system which named "fusion" classifier.

2.2 KTH descriptors definitions

KTH system for Vocal Primitives (VPs) recognition has been thoroughly described in deliverable D5.5.1. We do not describe it again but only focus on the exploitation of its outputs to recognize the vocal imitations categories on the SkAT-VG dataset.

KTH actually proposes three independent binary (presence/non-presence) classifier for each of the three VPs considered *phonation*, *slow myoelastic vibrations* and *turbulence*.

For a given audio file, the system outputs three different information:

L: Low-level. A large set of descriptors (62) which are used by KTH as observation to predict the VPs. It should be pointed out that these descriptors are shared by the three classifiers.

S: Scalar. Three scalar values which represent the output of the three trained Partial Least Square (PLC) models. They can be considered as the likelihood of the three VPs classes.

T: Thresholded. Three boolean values which report about the detection or not of each of the three vocal primitives: *phonation*/*no-phonation*, *vibrations*/*no-vibrations*, and *turbulence*/*no-turbulence*. They are obtained by thresholding the previous Scalar values.

In order to develop the "Informed" classifiers, Ircam-SAS considered the three different kinds of outputs (L,S,T). KTH system was first run on the whole vocal/gesture imitation dataset. The resulting outputs have been used to train prediction models for each imitation categories.

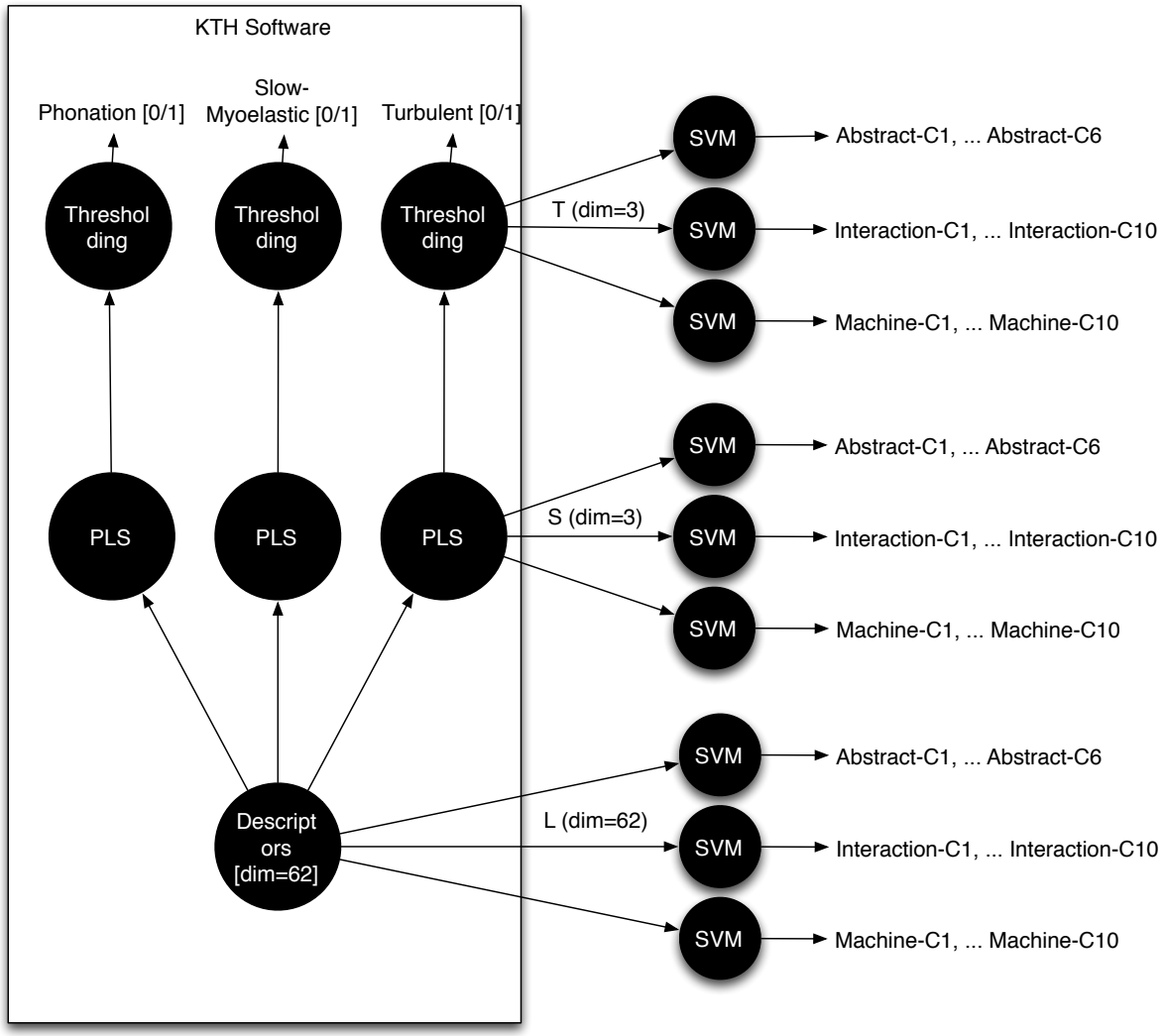


Figure 1: Ensemble of the KTH descriptors (**T**, **S** and **L**) along with the methodology used by Ircam SAS to build the Informed Classifiers (Sec. 2.3).

In Figure 1, we illustrate how the three outputs of the KTH system has been used to train the "Informed" classifiers.

2.3 Informed Classifier

Dataset. For each audio file of the vocal/gesture imitation dataset, we have computed the **L**, **S** and **T** descriptors. For this we have used the full dataset collection, i.e. we consider all the recordings of all subjects, both using Voice Only and Voice plus Gesture experiments (see D5.5.1 for details). This results in a collection of about 7300 audio files.

Evaluation scenario. The dataset is divided into three families: abstract, interaction and machine. Each family is further sub-divided into 6 categories within the abstract family, 10 within the interaction family and 10 within the machine family. The goal is to predict the

categories independently within each family. A distinct recognition task is therefore performed for each of the three families. This is performed using 10-fold cross-validation. It should be noted that we performed a subject-filtering: all imitations from a given subject are either in the test or in the train set, but never in both. Recognition performances are measured using the well-established Accuracy, Recall and Precision values. Recall and Precision are given for each category and averaged over the 10 folds.

Recognition system. Considering that none of the three types of outputs of KTH software is providing time-varying information (the information already represents the whole file duration, the evolution of the signal is already embedded into the descriptors themselves), it was not possible to test Dynamic Time Warping or Hidden Markov Models classification strategies as we did in deliverable D5.5.1. We therefore relied on Support Vector Machines (SVM) classifier.

We first applied a z-norm normalization to the descriptors (T, S or L) before giving them to the SVM training algorithm.

We used a soft-margin SVM with a RBF kernel. The parameters σ (of the RBF) and C (wrong classification) of the soft-margin SVM have been tuned using grid-search. For this, within each of the 10-folds, the training fold is further subdivided into 3 folds for parameter optimization. The parameters leading to the best average results (*mean recall*) over the 3 sub-folds is then chosen as the fold parameters. We explored the following parameters grid:

$$\begin{aligned} C &= \{0.00001, 0.001, 0.1, 1, 10, 1000, 100000\} \\ \sigma &= \{0.5, 0.8, 1.0, 1.2, 1.5, 2, 5\} \end{aligned}$$

This grid has been determined after a first set of early experiments, in which we have found effective to limit the search for σ around 1.

Results. In Table 1 we report the corresponding recognition results for each the three families (Abstract, Interaction and Machine) using each of the three descriptor sets (T,S and L). As one can see, the results obtained with the binary prediction of the vocal primitives (T) are lower than the ones obtained with the "likelihood" of the vocal primiteves (S) which itself is lower than the results obtained using the low-level descriptors (L). These results are disappointing since they somehow show that using the higher-level phonetic information provided by the vocal primitives (T or S) decrease the results.

We give a set of possible explanations:

1. There is no relationship between vocal primitives and vocal/gesture imitation categories; which is to say that people may use a large variety of vocal mechanisms to imitate the same category. Unfortunately, we cannot check this, given that we don't have the annotations into vocal primitives of the vocal/gesture imitation dataset. We can only rely on the estimation by KTH system of the vocal primitives.

2. There is a relationship between vocal primitives and vocal/gesture imitation categories; but the current KTH system failed to detect the vocal primitives on vocal/gesture imitation dataset. For the same reason as above we cannot check this assumption. However one potential limitation of the current system is to provide a single estimation for the whole file while the audio file may contain a succession over time of various vocal primitives.

3. There is a relationship between vocal primitives and vocal/gesture imitation categories and the current KTH system performs correctly; however the number of vocal primitives is not large enough to represent all the vocal mechanisms used in the vocal/gesture imitation dataset. This is possible. From a machine learning point of view, the low number of dimensions of T and S (only 3) gives a small amount of freedom to find the most discriminative directions to separate the 6, 10 and 10 categories. The fact that L has 62 dimensions will therefore favor a larger recognition score.

As a comparison, we also provide in Table 1 the equivalent results obtained using the GenMorpho descriptors with an SVM classifier, as presented in D5.5.1. While the L descriptors have been tuned to highlight the vocal primitives, the GenMorpho descriptors have been tuned to highlight directly the vocal/gesture imitation categories. This seems beneficial.

Family	#Categories	Descriptors	**T**	**S**	**L**	**GenMorpho**
Abstract	6	Accuracy	29.57	46.75	76.66	84.35
		Mean Recall	31.67	48.33	**77.47**	85.12
		Mean Precision	29.03	47.81	78.15	85.47
Interaction	10	Accuracy	22.03	41.49	64.31	71.58
		Mean Recall	22.39	41.65	**64.41**	71.69
		Mean Precision	21.79	41.91	65.04	72.39
Machine	10	Accuracy	25.96	38.24	66.84	69.06
		Mean Recall	24.67	37.35	**65.93**	68.15
		Mean Precision	17.71	37.84	67.68	69.90

Table 1: Results of 10-folds crossvalidation for the three imitation families using the **T**, **S** and **L** descriptors.

2.4 Fusion Classifier

The aim of Task 5.4 is to perform the "fusion" between the Blind classifier (Task 5.2) and the Informed classifier (Task 5.3). Exploiting well-known machine learning approaches, it is possible to merge the results of different classifiers, applied on the same task, in order to improve the overall recognition performances. In principle, the fused systems should compensate for each other weaknesses and provide better results. Since the Blind and Informed Classifiers are built according to different perspectives, we expect them to carry complementary information: the fusion should then work better than both classifiers alone.

Two main fusion paradigms exist: early and late-fusion. For a given classification system we denote by $\{x\}_i$ the input descriptor set, by f_i the classification algorithms and by $f_i(\{x\}_i)$ the predicted classes for an input $\{x\}_i$.

The **early fusion** of two systems i and j consists in merging their input $\{x\}_i$ and $\{x\}_j$ into a single input. The classification algorithm is therefore the same and is shared for/between the two inputs: $f_k(\{x\}_i, \{x\}_j)$. The dimensionality of the new inputs is of course larger.

The **late fusion** of two systems i and j consists in training a top-classifier based on the outputs of the individual outputs of bottom-classifiers: $f_k(f_i(\{x\}_i), f_j(\{x\}_j))$. To train it,

the two (or more) bottom-classifiers are first trained independently. The top-level classifier is trained based on the outputs of the two-bottom classifiers. For this, the two bottom-classifiers are evaluated over the training (or validation) set. The computed values from each classifier $f_i(\{x\}_i)$ and $f_j(\{x\}_j)$ (affinities, log-likelihoods, etc.) are then concatenated, and used as descriptors to train the fusion statistical model.

As a reminder, in deliverable D5.5.1 (Blind classifiers) Ircam-SAS has provided two different recognition systems. We choose to use the GenMorpho (GM) descriptors to test the effectiveness of the fusion schemes. Without giving here details about GenMorpho computation, we recall that they consist of a vector of 13 values per audio file, with embedded temporal evolution (no time-series). The recognition results of the GenMorpho descriptors on the vocal/gesture imitation dataset, recalled from D5.5.1, are reported in the last column of Table 2 and 3. We point out that the data folding scheme used for those has been applied for all the results in this Section, so results are directly comparable.

2.4.1 Early fusion

For the early fusion paradigm, we tested the concatenation of the GenMorpho descriptors and, in turn, **T**, **S** or **L** vocal primitives descriptors. 10-fold cross-validation with subject-filtering has been applied, and SVM parameters have been tuned by sub-cross-validation according to the same grid search described in part 2.3. The recognition results for each of the three families (abstract, interaction and machine) are given in Table 2.

For both Abstract and Interaction families the best results come from the fusion with the **S** descriptors. Unfortunately, in both cases the fusion mean recall values are actually lower than their counterpart from GenMorpho descriptors alone.

For the Machine family, instead, the best mean recall of 69.15% is found by the fusion between GenMorpho and **L** descriptors. In this case fusion is effective, because this value is slightly better than the mean recalls of the two classifiers alone.

Family	Descriptors	**GM+T**	**GM+S**	**GM+L**	**GM alone**
Abstract	Accuracy	82.11	83.68	83.00	84.35
	Mean Recall	82.94	**84.41**	83.84	85.12
	Mean Precision	83.35	84.67	84.20	85.47
Interaction	Accuracy	70.63	71.43	70.96	71.58
	Mean Recall	70.72	**71.60**	71.02	71.69
	Mean Precision	71.31	72.23	71.35	72.39
Machine	Accuracy	66.04	67.61	70.36	69.06
	Mean Recall	64.80	66.59	**69.15**	68.15
	Mean Precision	66.45	68.25	70.80	69.90

Table 2: Results of early fusion recognition on the three Families using GenMorpho (GM) and **T**, **S** and **L** descriptors fused with GenMorpho ones. 10-fold crossvalidation is applied, along with sub-cross-validation to tune the SVM parameters.

2.4.2 Late fusion

Given the fact that early fusion has proven effective only when applied to the **L** descriptors (for the Machine family), we have decided to apply a late fusion strategy only between GenMorpho and **L** descriptors (disregarding **T** and **S**). As introduced before, late fusion approach uses a two-level paradigm:

- first train the bottom classifiers $\{x\}_i \rightarrow f_i(\{x\}_i)$ and $\{x\}_j \rightarrow f_j(\{x\}_j)$,
- then train the top-level classifier $\{f_i(\{x\}_i), f_j(\{x\}_j)\} \rightarrow f_k(\{f_i(\{x\}_i), f_j(\{x\}_j)\})$.

In principle the three classifiers f_i, f_j and f_k could be totally different, but we have chosen to rely on SVM in all cases. This means that we need to tune the parameters for the three SVMs.

The detailed approach is:

1. GenMorpho and **L** descriptors are computed for the whole dataset.
2. The dataset is folded in $F = 10$ parts (with subject filtering): 9 folds are used as train set D_{tr}^f and the remaining one as test D_{te}^f (folds $f \in [1, F]$).
3. Bottom-classifiers training:
   - The two bottom classifiers are trained on D_{tr}^f: one with GenMorpho and the other with **L** descriptors:
   - Bottom-level classifier optimization (*): For each bottom classifier, the best SVM parameters are found using sub-fold-cross-validation (dividing D_{tr}^f in three folds and searching using grid-search the best C and σ parameters).
4. Bottom-classifiers evaluation on training data: the two resulting models are then applied on D_{tr}^f to collect the output affinities $A_{tr}^{\{f,j\}}$ where $j \in [1, 2]$ indicates the two models.
5. Top-level classifier training:
   - $A_{tr}^{\{f,1\}}$ and $A_{tr}^{\{f,2\}}$ are concatenated, and used to train the top-level classifier
   - Top-level classifier optimization (**): The best SVM parameters are found using sub-fold-cross-validation: 3-folds sub-cross-validation is applied to tune the parameters C and σ of the Late model.
6. Top-level classifier testing on test-data D_{te}^f: the category recognition is done according to the maximum late affinity.
7. Points 2. to 6. are repeated for each fold $f \in [1, 10]$, and results are averaged.

We report in Tab. 3 the results of the described procedure.

- In column LateFusion(*) we report the results using only the bottom-level classifier optimization (we did not use the top-level classifier optimization (**).

Family	Subcrossvalidation:	LateFusion(*)	LateFusion(**)	LateFusion(*)+(**)	GM alone
Abstract	Accuracy	80.21	81.24	80.32	84.35
	Mean Recall	80.36	**81.58**	80.65	85.12
	Mean Precision	84.00	84.11	83.59	85.47
Interaction	Accuracy	63.97	67.86	67.06	71.58
	Mean Recall	63.97	**67.90**	67.12	71.69
	Mean Precision	70.88	68.48	68.02	72.39
Machine	Accuracy	59.12	61.90	62.85	69.06
	Mean Recall	58.75	60.75	**61.76**	68.15
	Mean Precision	64.26	62.30	62.89	69.90

Table 3: Recognition results of late fusion on the three Families using **L** and GenMorpho descriptors. 10-fold crossvalidation is applied, along with subcrossvalidation with various SVM parameter optimization.

- In column LateFusion(**) we report the results using only the top-level classifier optimization (we did not use the bottom-level classifier optimization (*).
- In column LateFusion(*)+(**): we report the results using both optimization.

Despite our efforts, the mean recalls shown in Tab. 3 are, for all Families, weaker than the ones obtained using the early fusion and/or GenMorpho (GM) descriptors alone. Arguably, the affinities provided by the bottom-classifiers are not meaningful if used to train the Late models.

3 Automatic estimation of Audio Primitives and recognition of vocal/gesture imitation using audio primitives

Despite the disappointing results obtained using vocal primitives estimation to recognize the imitation categories, there was still a strong scientific interest in the SkAT-VG project to have a higher semantic interpretation of what people use to imitate a given sound category. Are all people using the same kind of sounds to imitate a printer-fax? Is there a sub-set of possible sounds to imitate a printer-fax? Does the type of sounds used to imitate a printer-fax depend more on the subject than on the category?

Rather than manually annotating the whole vocal/gesture imitation test-set into primitives, we decided (given also the FET nature of the project and the encouragement of the reviewers to take risks and be innovative) to develop a new approach to automatically derive the set of primitives that allow describing a given dataset.

We name them "audio primitives" by opposition with the "vocal primitives" developed by KTH. The audio primitives do not assume any production mechanisms related to the voice. They only rely on time and frequency audio representation. To find the audio primitives of a given dataset, we have chosen, among the possible methods for unsupervised learning, the Shift-Invariant Probabilistic Latent Component Analysis (SI-PLCA).

In order to validate the fact that these audio primitives (automatically found by SI-PLCA) actually bring a semantic, we have created a small dataset manually annotated into audio primitives. This small dataset allows to compare manual and automatically estimated primitives.

Given the set of audio primitives, a given sound can be automatically decomposed into these primitives. The resulting temporal activations can then be considered as emissions of hidden states belonging to a sequence model specific to each imitation category. We therefore trained a hidden Markov model for each category. Each state of a model represents the use of the audio primitives.

We believe this research direction, apart from the fulfillment of the requirements of the SkAT-VG DOW, can also have an even more far-reaching and innovative content.

In part 3.1, we first present the small dataset manually annotated into audio primitives that will be used to validate the results obtained by SI-PLCA. We then present in part 3.2, the SI-PLCA algorithm and its use to obtain audio primitives. We also present a distributed version of SI-PLCA which allows to partly solve the computational issues of this method. We finally present in part 3.2.5 how the activation of the audio primitives of a given unknown signal can be used to perform automatic recognition of imitation categories given a set of HMMs.

3.1 Ircam dataset of manual annotation into audio primitives

In part 3.2 we will present an unsupervised learning tools (SI-PLCA) that allows finding automatically audio primitives. In order to assess the performances of it, we have manually annotated a small test-set into "acoustic cues". By *cues* we mean salient behaviors of the audio signal in its temporal and spectral dimensions: we thus relax the requirement for a detailed phonetic transcription.

3.1.1 Acoustic cues definition

We introduce here the dictionary of **acoustic cues** used for the manual annotation. Every audio signal is observed by means of its short-term spectrogram. We then mark the salient regions in the spectrogram (both in time and frequency) by means of the following labels:

Group	Label	Definition
Content	**H**	Harmonic: clear presence of several harmonic lines
	N	Noise: region has a clear energy content, but without evident harmonics
	R	Roughness: similar to noise, but when listened there is some low-frequency repetitiveness
Resonance	**F**	Presence of clearly noticeable formants/resonances
Evolution	**S**	Static: the spectral energy distribution remains stable for more than 0.5s
	QS	Quasi-static: long-term changes, similar to Static, but with very slow rise/fall
	V	Variable: sequence of random short-term variations in spectral energy
	L	Slope: short-term, rather fast (less than 2s) and monotonic change in spectral distribution
	P	Pulse: very fast phenomena, with short attack and decay (clicks and similar)
Repetition	n	A number is present if a certain pattern is clearly repeated; identical numbers identify identical patterns.

We point out that no difference is made between rising and falling profiles for the L Evolution. Moreover P Evolution is bound to loudness/energy, while S, QS and L are more linked to spectral cues.

To ease the automatic parsing of the annotated dataset, we follow some labeling rules. Labels from the the groups "Resonance", "Evolution" and "Repetition" can be omitted, but if used have always to be in order; HFL and HL are both valid labels, while HLF is not. The labels from the group "Contents" can be combined together, but always respecting the given order: HNL is a valid label, while NHL is not.

In Fig. 2, we provide an example of manual annotation. The input audio file has been splitted in several regions according to the aforementioned dictionary and rules.

Of course, the annotation is not completely objective and personal annotator perception can influence the labeling (especially for the Content). Moreover, we rely on only one annotator. Despite these weaknesses, we think that the annotated dataset provides a good starting point to assess SI-PLCA.

3.1.2 Annotated dataset description

Ircam-SAS has annotated 115 audio files chosen from the vocal/gesture imitation dataset. The data have been chosen from the Voice Only portion of the dataset (disregarding the Voice plus Gesture part). We have only chosen data from the Interaction and Machine family

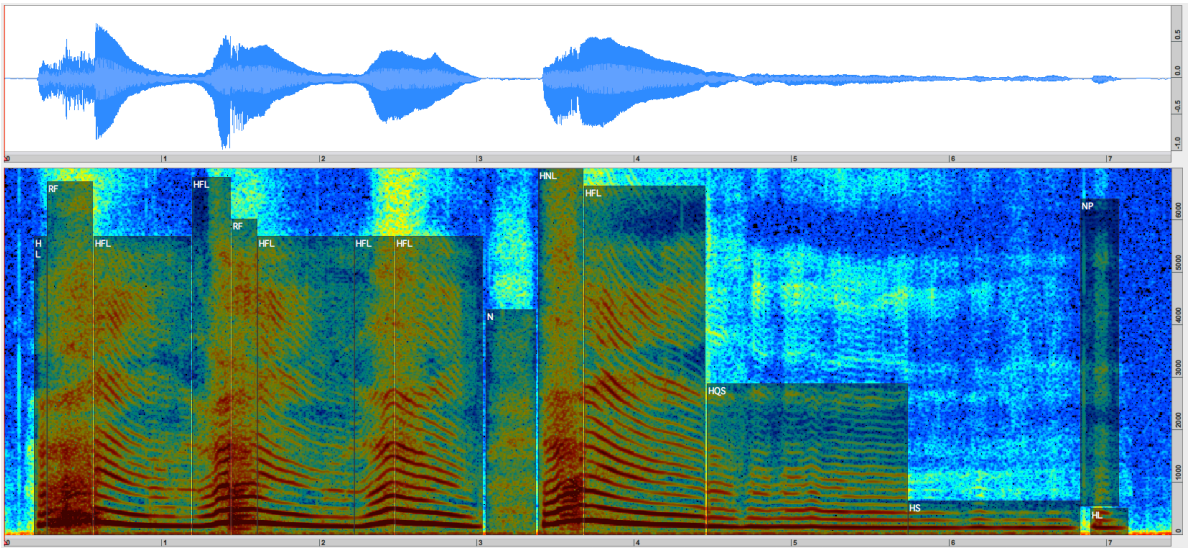


Figure 2: Example of annotations on the SkAT-VG dataset. The following sequence of labels has been manually annotated over time: HL, RF, HFL, HFL, RF, HFL, HFL, HFL, N, HNL, HFL, HQS, HS, HL+NP.

because their acoustic content presents a higher variability than those of the Abstract family.

Twelve imitation Categories have been chosen:

Family	Categories
Interaction	blowing, shooting, crumpling, rubbing, hitting, dripping, filling
Machine	filing, fridge, mixers, vehicleext, vehicleint

We have only used the recordings of the first five subjects and for each imitation only used the last trial (which is supposed to be the best one).

Using this selection, we have selected and manually annotated 115 audio files (total recording duration of 486.38s).

3.1.3 Distribution of the annotated labels

Using this manual annotations, we propose here an analysis of the vocal/gesture imitation dataset.

We define an annotation label as a specific combination of "acoustic cues". It should be noted that, in order to reduce the number of possible combinations between the "acoustic cues", we decided to omit the "Resonance" and "Repetition" groups. This reduces the number of combinations to 18.

We first shows the distribution the annotation labels for a) the whole dataset, b) each imitation category and c) each subject.

3.1.4 Global distribution.

In Figure 3 we represent the distribution of the annotation labels for the whole dataset. The figure simply counts the number of occurrences of each unique annotation label over the whole

dataset.

Some clarifications are needed to explain the strong un-balanceness of the histogram. The X label appears to be among the most used, and deserves an explanation. The annotated regions almost never "cover" the entire surface of a spectrogram (see Figure 2), because low-energy zones are deemed as uninteresting: they do not contain any relevant acoustic cue. We have thus decided to introduce a fictitious X label, which is automatically added during the annotation parsing to account for the non-labeled areas of the signal. This hence enables for the analysis of these regions too. Since for every imitation the X label is automatically introduced once and only once, in the histogram the corresponding value is 115 (the number of audio recordings).

The NP label is by far the most used. Noise pulses, in fact, appear rather often in the imitations (clicks, stops, etc.), and are sometimes produced in groups (a remarkable example is the `dripping` category). The labels N, HL and NV are then the most used, followed by R and NS.

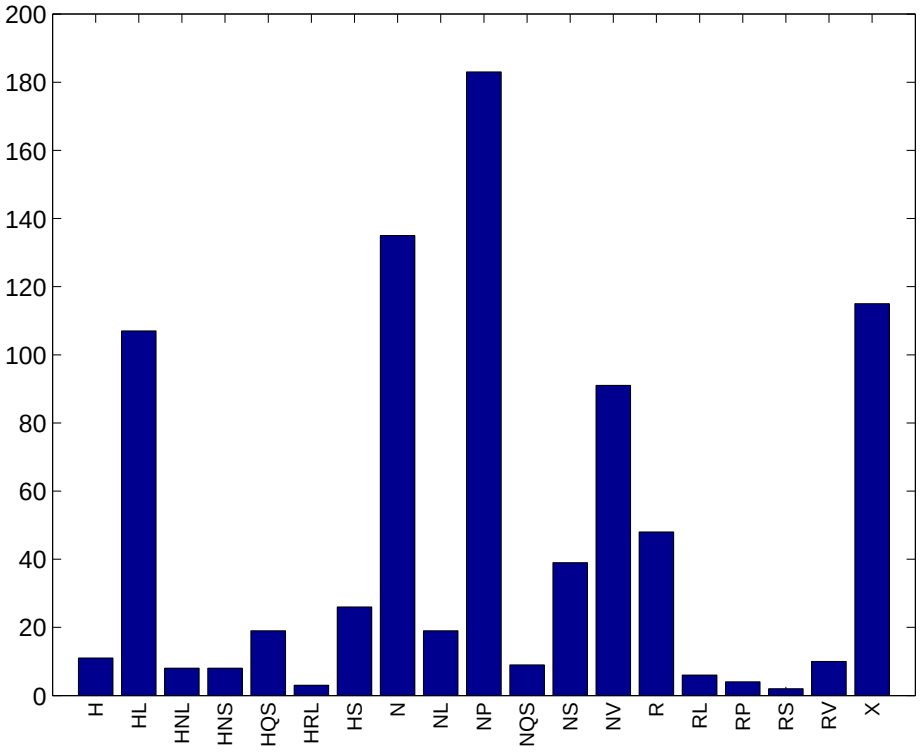


Figure 3: Bar graph of annotations labels usage over the whole dataset.

3.1.5 Distribution by imitation category.

In Figure 4(left) we show the usage of annotation labels by imitation category. This distribution has been normalized by the distribution presented in Figure 3, to marginalize the effect of labels

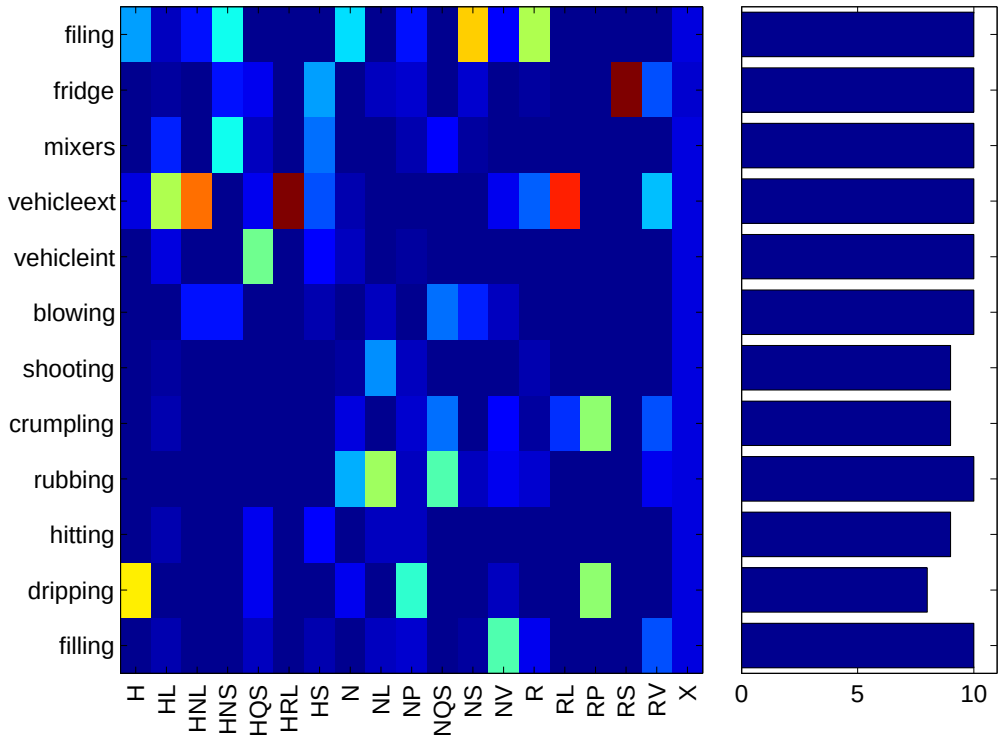


Figure 4: Left panel: Per-category distribution of annotation labels. Right panel: Count of imitations per imitation category in the dataset.

usage and enhance each category content: each column is thus a distribution.

Some intuitive associations are confirmed. `Fridge` imitations are strongly linked to a static roughness label (RS). The `vehicleext` category is correlated to several different label, but all share the slope (L) evolution: the original reference sounds are in effect cars revving up. Fig. 4(right) show a the balanced categories content of the dataset.

3.1.6 Distribution by subjects.

In Figure 5 we show the usage of annotation labels by subject. In this case the distribution are sorted by subject. We can see that the majority of RS labels are assigned to imitations from the first subject, while HRL are mostly used by the fifth subject.

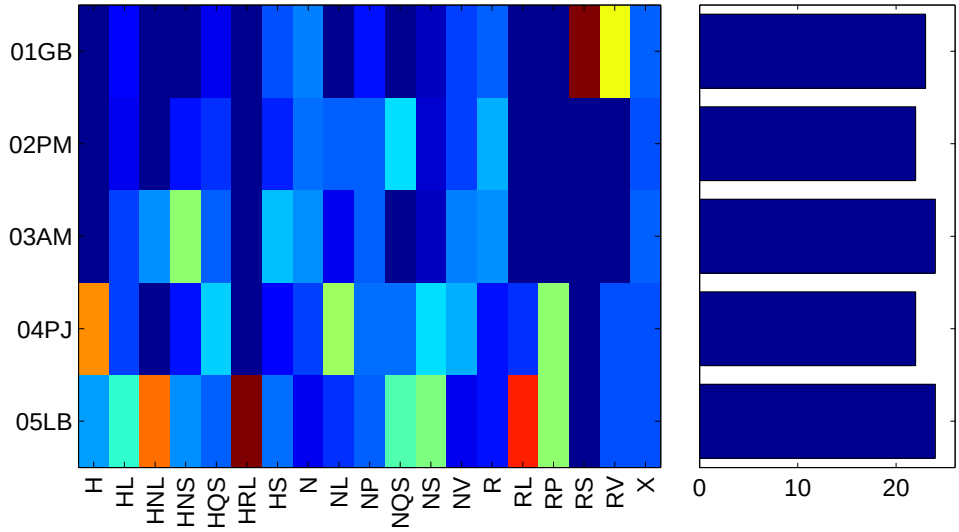


Figure 5: Left panel: Per-subject distribution of the annotation labels. Right panel: Count of imitations per subject in the dataset.

3.2 Automatic derivation of audio primitives using SI-PLCA

As mentioned above, our goal is to develop a method that allows to automatically derive a set of primitives, which we name "audio primitives" from the analysis of a dataset of audio recordings. The method used to derive these primitives is the Shift-Invariant Probabilistic Latent Component Analysis (SI-PLCA). In part 3.2.1 we first motivate the choice of this method (among the various possible unsupervised learning methods). In part 3.2.2 we then present this method, apply it to our problem and propose a distributed version of the training algorithm. In part 3.2.4 we assess the use of the SI-PLCA to derive acoustically meaningful primitives by comparing these audio primitives to the "acoustic cues" annotated in the previous part. Finally, in part 3.2.5, we propose a system for recognizing the vocal/gesture imitations based on the activations over time of the audio primitives. This system relies on modeling each imitation as an hidden Markov model the states of which emit audio primitives.

3.2.1 Choice of unsupervised learning approach

Our goal is to automatically define audio *primitives* from a dataset of recordings. By "primitives" we mean the relevant acoustic cues that arise observing a time/frequency representation of the audio signal.

The interest of this problem lays in the absence of information about the audio content: the automatic tool is supposed to both define and detect the primitives by itself. Intuitively, we need to rely on unsupervised learning techniques. There are some peculiarities in the nature of the audio primitives we are interested in:

- they can appear at the same time and/or frequency location (eg. low-frequency harmonics superimposed to high-frequency noise);
- they are repeated in shifted time and/or frequency positions.

Among the various unsupervised learning methods, we focused on the ones allowing for additivity and shift-invariance of the primitives. It is the case for the Shift-Invariant Probabilistic Latent Component Analysis (SI-PLCA).

To introduce SI-PLCA, we begin with a well-known mathematical tool for the unsupervised decomposition of matrices: the Non-negative Matrix Factorization (NMF). NMF is a tool to factorize a non-negative matrix V into two non-negative terms [5, 4]:

$$V \approx WH$$

The H and W matrices are usually indicated as the *bases* and the *mixture weights*: in this modeling the data vectors (columns of V) are expressed as linear combinations of the bases in H mixed by the weights in the columns of W. The NMF approach is algebrical, with no probabilistic extensions. It has a well-founded background theory and has proven successful in a number of applications. Despite this, there are cases in which NMF is not easily applicable: for example, if the input data has more than two dimensions or some constraints are needed on the shape of the bases H, then it is not straightforward to extend NMF beyond the given formulation.

NMF can be reformulated in a probabilistic framework as the Probabilistic Latent Component Analysis (PLCA) [7]. PLCA is a technique which, among many others, belongs to

the *latent class models* (well-known others are Probabilistic Latent Semantic Analysis, Latent Dirichlet Allocation, etc.). All these have a pair of points in common:

- they are conceived to explain the input data by means of *latent* classes;
- they are not applied directly to experimental data, but rather on their histograms.

More precisely, to apply PLCA we need to observe one realization of a random multidimensional variable $\mathbf{x}$. We then compute its empirical distribution function $P(\mathbf{x})$. PLCA explains $P(\mathbf{x})$ as a mixture of latent distributions, which are made explicit along with their mixing weights.

In the following sections we introduce and use a *shift-invariant* formulation of the PLCA.

3.2.2 SI-PLCA methods

3.2.2.1. Mathematical background

Shift-Invariant Probabilistic Latent Component Analysis (SI-PLCA) has been introduced [9] as an extension to the PLCA [7].

Let us begin by focusing on the PLCA model, which can be stated as:

$$P(\mathbf{x}) = \sum_{z=1}^{K} P(z)P(\mathbf{x}|z) = \sum_{z=1}^{K} \left[P(z) \prod_{j=1}^{N} P(x_j|z) \right] \tag{1}$$

where $P(\mathbf{x})$ is the N-dimensional distribution of the random variable $\mathbf{x}$. The second term of Eq. 1 expresses clearly that we are describing $P(\mathbf{x})$ as the weighted sum of several $P(\mathbf{x}|z)$. Hence we suppose that the global *behavior* of $P(\mathbf{x})$ is explained by several other, distinct distributions $P(\mathbf{x}|z)$: the *latent classes*. These are combined together by means of the latent variable z, which assumes discrete values in $\{1, K\}$ and whose distribution $P(z)$ expresses the mixing weights among the latent classes. The model is fully expanded in the last term of Eq. 1, where $P(\mathbf{x}|z)$ is decomposed into its single-dimensional parts $P(x_j|z)$. By the *local independence principle*, expressing z renders independent the distributions of $\mathbf{x}$ along each of its dimensions (which otherwise could be dependent).

In order to estimate the latent components $P(\mathbf{x}|z)$ and the distribution $P(z)$ it is possible to apply a standard Expectation-Maximization (EM) procedure. As it is common with this approach, we begin by estimating the contribution to $P(\mathbf{x})$ for each value assumed by z:

$$R(\mathbf{x}, z) = \frac{P(z) \prod_{j=1}^{N} P(x_j|z)}{\sum_{z'} P(z') \prod_{j=1}^{N} P(x_j|z')} \tag{2}$$

We can then estimate the distribution of z by marginalization over the whole space of $\mathbf{x}$:

$$P(z) = \int P(\mathbf{x}) R(\mathbf{x}, z) d\mathbf{x}$$

Similarly, for each value of z, we can compute the $P(x_j|z)$ distributions by marginalizing over all the other dimensions of $\mathbf{x}$:

$$P(x_j|z) = \frac{\int \cdots \int P(\mathbf{x}) R(\mathbf{x}, z) dx_k, \forall k \neq j, k \in \{1, N\}}{P(z)}$$

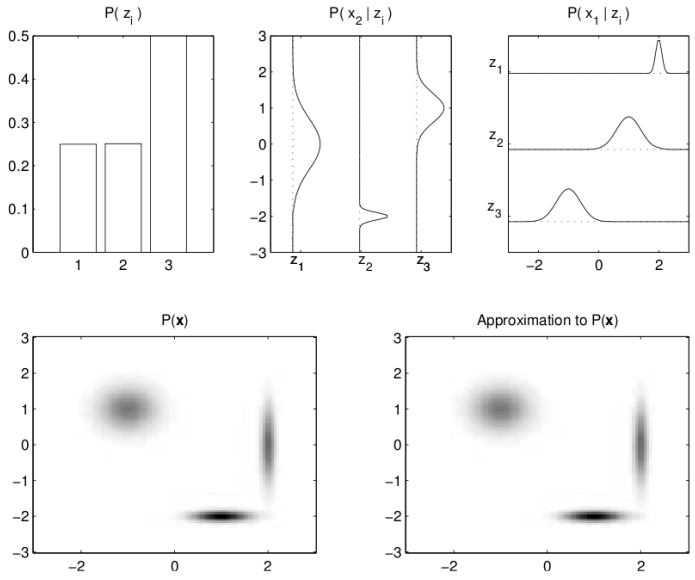


Figure 6: Example of PLCA decomposition over a bidimensional random variable (from [9]).

In Fig. 6 there is an example of PLCA. The input random distributions is bidimensional (lower left panel), and is clearly composed by three distinct "components". We can then apply the PLCA iterative procedure, having $K = 3$ and $z \in \{1, 2, 3\}$. In the top left panel we can see that the PLCA estimated the distribution $P(z)$ as equally probable for the first two values of z, and almost twice as probable for the third value. In fact, looking at the input, there is one component which is stronger (remark: the association between the input components and the values of z is aleatory and can change between different runs of the EM procedure). Coming to the top center and right panels, here we have the estimated $P(x_j|z)$ distributions: they are two because they represent the two dimensions of $\mathbf{x}$, and each of them is actually made by three distinct curves, because they are conditioned to the values of z. It is easy to verify that the three input components are approximated by the outer vector product of $P(x_1|z)$ and $P(x_2|z)$ for each fixed z. The complete approximation of the input (lower right panel of Fig. 6) has been found, according to Eq. 1, by weighting the outer products by $P(z)$.

Let us now introduce the SI-PLCA. Still looking at Fig. 6, it is easy to figure out that, if one of the three components would have been replicated in a shifted position, this would have been incorporated into the output estimates:

- setting $K = 4$ would result in the fourth component being clearly isolated into its own $P(x_j|z)$ distributions;
- letting $K = 3$ would return still a three term decomposition, which would have been estimated in order to maximize the approximation of the input (but would probably have badly approximated all the three-plus-one input components).

What is missing from the current formulation is the so-called *shift invariance*, that is the

capability of the procedure to recognize a given component despite having it being moved around.

It has then been proposed [9] to update the Eq. 1 to allow for shift invariance by using a convolutive model, which has two distinct parts to decompose the input. We call **kernels** (or bases) the "elementary" distributions which are latent in the input, and **activations** the distributions of the shifts which, applied to the kernels, reproduce the input $P(x, y)$.

The model becomes:

$$P(\mathbf{x}) = \sum_z \left(P(z) \int P(\boldsymbol{\tau}|z) P(\mathbf{x} - \boldsymbol{\tau}|z) d\boldsymbol{\tau} \right) \tag{3}$$

The term $P(\boldsymbol{\tau}|z)P(\mathbf{x} - \boldsymbol{\tau}|z)$ replaces what is $P(\mathbf{x}|z)$ in Eq. 1. $P(\boldsymbol{\tau}|z)$ represents the K kernels distributions, dependent on z. A new variable $\boldsymbol{\tau}$ has been introduced: it denotes the dimensions of the kernels. The term $P(\mathbf{x} - \boldsymbol{\tau}|z)$ describes the activations of the kernels along the space of the input $\mathbf{x}$.

Without loss of generality, we can fix the input dimensionality to $N = 2$ and enumerate the input dimensions as x and y. Eq. 3 is then restated as:

$$P(x, y) = \sum_z P_Z(z) \iint P_K(\tau_x, \tau_y|z) P_I(x - \tau_x, y - \tau_y|z) d\tau_x \, d\tau_y \tag{4}$$

The estimations of $P_Z(z)$, $P_K(\tau_x, \tau_y|z)$ and $P_I(x - \tau_x, y - \tau_y|z)$ can be done with the same EM approach introduced above, but the formulas have to take into account the shifts. The R auxiliary variable (Eq. 2) is used to compute the contributions for all the possible values of z over the space of $\mathbf{x}$. It is now updated to find the contributions due to each possible (τ_x, τ_y) shift over (x, y) space:

$$R(x, y, \tau_x, \tau_y, z) = \frac{P_Z(z) P_K(\tau_x, \tau_y|z) P_I(x - \tau_x, y - \tau_y|z)}{\sum_z P_Z(z) \iint P_K(\tau_x, \tau_y|z) P_I(x - \tau_x, y - \tau_y|z) d\tau_x \, d\tau_y} \tag{5}$$

In the maximization step there are now three equations, one for each term to be estimated:

$$P_Z(z) = \iiiint P(x, y) R(x, y, \tau_x, \tau_y, z) dx \, dy \, d\tau_x \, d\tau_y \tag{6}$$

$$P_K(\tau_x, \tau_y|z) = \frac{\iint P(x, y) R(x, y, \tau_x, \tau_y, z) dx \, dy}{P_Z(z)} \tag{7}$$

$$P_I(\tau_x, \tau_y|z) = \frac{\iint P(x + \tau_x, y + \tau_y) R(x + \tau_x, y + \tau_y, \tau_x, \tau_y, z) dx \, dy}{\iiiint P(x + \tau_x, y + \tau_y) R(x + \tau_x, y + \tau_y, \tau_x, \tau_y, z) dx \, dy \, d\tau_x \, d\tau_y} \tag{8}$$

Eq. 6 is simply the marginalization of $P \cdot R$ product by all the variables except z: all the contributions of all the variables are thus counted together. To estimate the kernels it is necessary to marginalize over the space (x, y), to leave only the contributions of z and of the shifts τ_x and τ_y (Eq. 7). For each point in the bidimensional space of the kernels (plus dimension z), are cumulated the contributions from all the points in (x, y). In the last equation 8 the aim is opposed: for each point in input space (x, y) we cumulate the contributions from all the possible shifts of the kernel z.

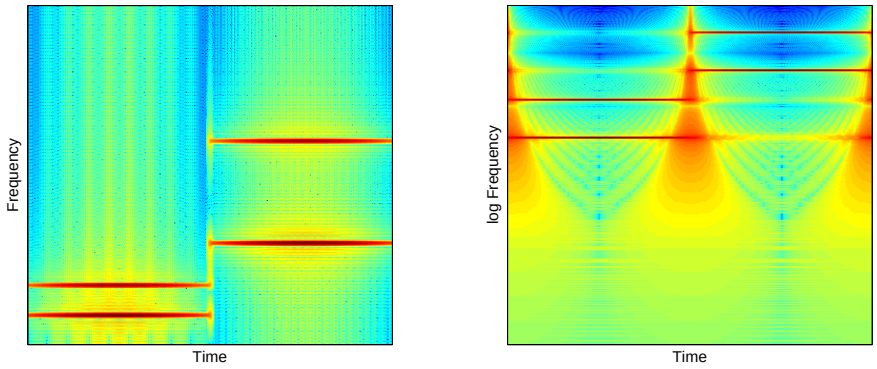


Figure 7: Comparison between FFT (left panel) and CQT (right panel) performed on the same signal (440Hz and then 1500Hz tones with one harmonic). CQT has higher low-frequency resolution and is shift-invariant also along frequency.

3.2.2.2. Application to audio data

As already introduced, the SI-PLCA works on N-dimensional histograms of random variables. Let us consider the spectrogram of an audio signal (that is, its frame by frame short-time Fourier transform): by means of a simple normalization it fits the definition of distribution of probability, over time and frequency dimensions. We can then associate the spectrogram to the probability distribution of a bidimensional random variable, therefore enabling the use of the PLCA.

In the context of SkAT-VG , we are looking for the automatic detection of acoustic primitives, and this motivates our interest into the shift-invariance property of the SI-PLCA. In order to properly exploit the SI-PLCA characteristics, also the spectral representation of the input audio signal has to guarantee the shift-invariance. Using well known FFT-based spectrogram, the invariance is easily verified over the time dimension: if the same event is repeated over time, also its spectral representation will be repeated and identical. In the frequency domain things are less straightforward, and FFT is not suitable. We would like to exploit the shift-invariance to recognize pitch-shifted events as the same acoustical primitive. The notes of musical instruments, and every common *pitched* sound, are characterized by *harmonics*: a note with fundamental frequency at f_0 Hz is actually the superposition of pure tones at frequency kf_0 Hz, with $k \in \mathcal{N}$. Let us suppose to have an acoustic event with only one harmonic, as the one depicted in Fig. 7 (left panel). At fundamental frequency 440Hz, its harmonic is at 880Hz. Pitch-shifting the event to 1500Hz, the harmonic moves up to 3000Hz. Being the frequency scale linear, the spacing between the "spectral lines" in the first and in the second case are different: the representation is thus not shift-invariant.

In order to exploit the SI-PLCA it is needed a spectral representation which preserves the time/frequency structure of harmonic events when these are pitch-shifted. Given the multiplicative nature of the harmonic sounds, a solution to this issue is to use spectral representations with *logarithmic* frequency scale. The Constant-Q Transform (CQT) is a spectral representation alternative to the FFT: the input audio signal is analyzed by means of a bank

of filters with constant quality factor Q [1, 10]. The center frequencies f_k of the K analysis filters in the CQT are defined according to a geometrical series, with the formula:

$$f_k = f_0 2^{k/b}, \;\; k \in \{1, 2, \ldots, K\} \tag{9}$$

being f_0 the lowest analyzed frequency and b the number of frequency bins per octave ($b = 36$ in the present document). Since Q is defined as the ratio between center frequency and bandwidth of a filter, the uniform coverage of the analyzed spectrum is ensured. When applying the CQT to a pitch-shifted harmonic sound, as in Fig. 7 (right panel), its representations remains pretty identical, except for the frequency shift.

A CQT spectrogram is thus suitable to be used with the SI-PLCA for acoustical primitives recognition. Since SI-PLCA works on distributions we normalize the CQT such that all its values sum to one.

CQT analysis has been carried out between 70 and 5000Hz, using 36 bins per octave. The toolkit which has been used for the CQT [6] allows the tuning of a parameter γ, to increase time resolution on lower frequencies. Recalling Eq. 9, the bandwidths B_k of the filters are found as:

$$B_k = \frac{f_k}{Q} + \gamma$$

Following the default settings of the toolkit we set $\gamma = 20$, and eventually the ratios between f_k and B_k are actually *not* constant: this is known as Variable Q Transform (VQT). For the analysis of the dataset we use VQT: we have verified that the spectrograms remain suitable for shift-invariant analyses, while the increase in temporal resolution improves the analysis effectiveness. However, in the following sections we still use the acronym CQT because it better conveys the key idea.

3.2.2.3. Implementation details

The formulation of SI-PLCA given in Eqs. 5-8 is intended for random variables with continuous domains (except z which is discrete). We need to apply the SI-PLCA to a CQT spectrogram, which is discrete in both time and frequency.

Eqs. 5- 8 have thus been reformulated in a way suitable for translation in software. To clarify the exposition, we define the operator $I_x\{f(x)\} = \sum_x f(x)$. The SI-PLCA has been implemented in source code by the following formulas [9]:

$$R(x, y, \tau_1, \tau_2, z) = \frac{P_Z^{(n)}(z) P_K^{(n)}(\tau_1, \tau_2|z) P_I^{(n)}(x - \tau_1, y - \tau_2|z)}{\sum_{z'} P_Z(z') I_{\tau_1', \tau_2'}\{P_K(\tau_1', \tau_2'|z) P_I(x - \tau_1', y - \tau_2'|z)\}} \tag{10}$$

$$P_Z^{(n+1)}(z) = I_{x,y,\tau_1,\tau_2}\{P(x, y) R(x, y, \tau_1, \tau_2, z)\} \tag{11}$$

$$P_K^{(n+1)}(\tau_1, \tau_2|z) = \frac{I_{x,y}\{P(x, y) R(x, y, \tau_1, \tau_2, z)\}}{P_Z^{(n+1)}(z)} \tag{12}$$

$$P_I^{(n+1)}(x, y|z) = \frac{I_{\tau_1,\tau_2}\{P(x + \tau_1, y + \tau_2) R(x + \tau_1, y + \tau_2, \tau_1, \tau_2, z)\}}{I_{x',y',\tau_1,\tau_2}\{P(x' + \tau_1, y' + \tau_2) R(x' + \tau_1, y' + \tau_2, \tau_1, \tau_2, z)\}} \tag{13}$$

In the formulas, n represents the iteration number; we let at most 130 iterations to have the algorithm converge.

An initial random guess is used for P_Z, P_K and P_I when $n = 1$. In particular, all the three distributions are initialized according to a uniform distribution. P_Z and $P_I(x, y|z)$ are filled with equal values, normalized to sum up to one. The kernel distributions $P_K(\tau_1, \tau_2|z)$ are instead initialized with the realization of a uniform random variable: the numeric "noise" of P_K is needed to let the algorithm convergence begin.

One of the interesting characteristics of the (SI-)PLCA techniques is the straightforward way of imposing constraints on the "shapes" of P_K and/or P_I. One would like to have P_K describing the "dictionary" of the input data (the elementary cues of the histogram), and P_I representing the activations of the kernels: ideally, P_I should only contains a certain number of delayed deltas. In other terms, it is advisable to impose some *sparsity constraints* on the estimated distributions. Several proposed strategies [8, 3] are founded on a full-fledged statistical background (*entropic prior*). We however applied a different approach, suggested in [9], following the well-known "simulated annealing" paradigm. At each iteration n the distributions P_K and P_I are updated by means of Eqs. 12 and 13 respectively. Right after each update, the following formulas are applied:

$$P_K(\tau_1, \tau_2|z) \leftarrow c_1 \cdot P_K(\tau_1, \tau_2|z)^{\alpha(n)}, \quad \alpha(n) > 0 \tag{14}$$

$$P_I(x, y|z) \leftarrow c_2 \cdot P_I(x, y|z)^{\beta(n)}, \quad \beta(n) > 0 \tag{15}$$

In Eq. 14 the updated P_K is again updated with an exponentiation by α; the value c_1 is simply a normalization to ensure P_K still remaining a valid probability distribution. Similarly, Eq. 15 is used to re-update P_I using β. The key point lies in the correct choice of α and β values, which are evolving along the iterations. $\alpha(1)$ is fixed to a value < 1, such as $[0.8, 0.9]$, and slowly rose to 1, which is reached at the end of the iterations. For P_I the β is initially fixed to 1 and then slowly rose, such to reach about $[1.05, 1.1]$ at the end of the algorithm (or even before). By applying to P_K an exponent with value < 1 we obtain the effect of "flattening" the distributions: at the begin we are thus keeping the kernels rather uniform, avoiding early "locally optimal" choices. As suggested by [9], this equals to seek for high entropy in the kernels: entropy will then be lower on the distributions P_I, thus hopefully enforcing sparseness on these. Eq. 15 operates with identical aim, but conversely since it is applied on P_I: the effect is to enhance peaks in P_I, once again resulting in sparseness.

Fig. 8 shows the results of SI-PLCA when applied on a toy problem; two distinct distributions are repeated over the domain of the problem, by means of shifts along both the axes. The SI-PLCA figures out the shapes of the two distributions (the kernels) and at the same time locates them over by the corresponding activations. The mixing probabilities P_Z explain that one of the kernels appears more used than the other.

3.2.2.4 Distributed algorithm

The computational requirements of the SI-PLCA algorithm are rather costly. Eqs. 10-12 have to be computed much times (between 60 and 100 iterations in real-world problems), and each of them implies several nested loops. As a rule of thumb, the computational cost of the SI-PLCA depends on the product of: the input sizes, the kernel/shift sizes, the number of kernels which we are looking for, and the allowed iterations. Despite this, the challenging part of this computation is due to its memory requirements, which are leaded by the matrix R: its

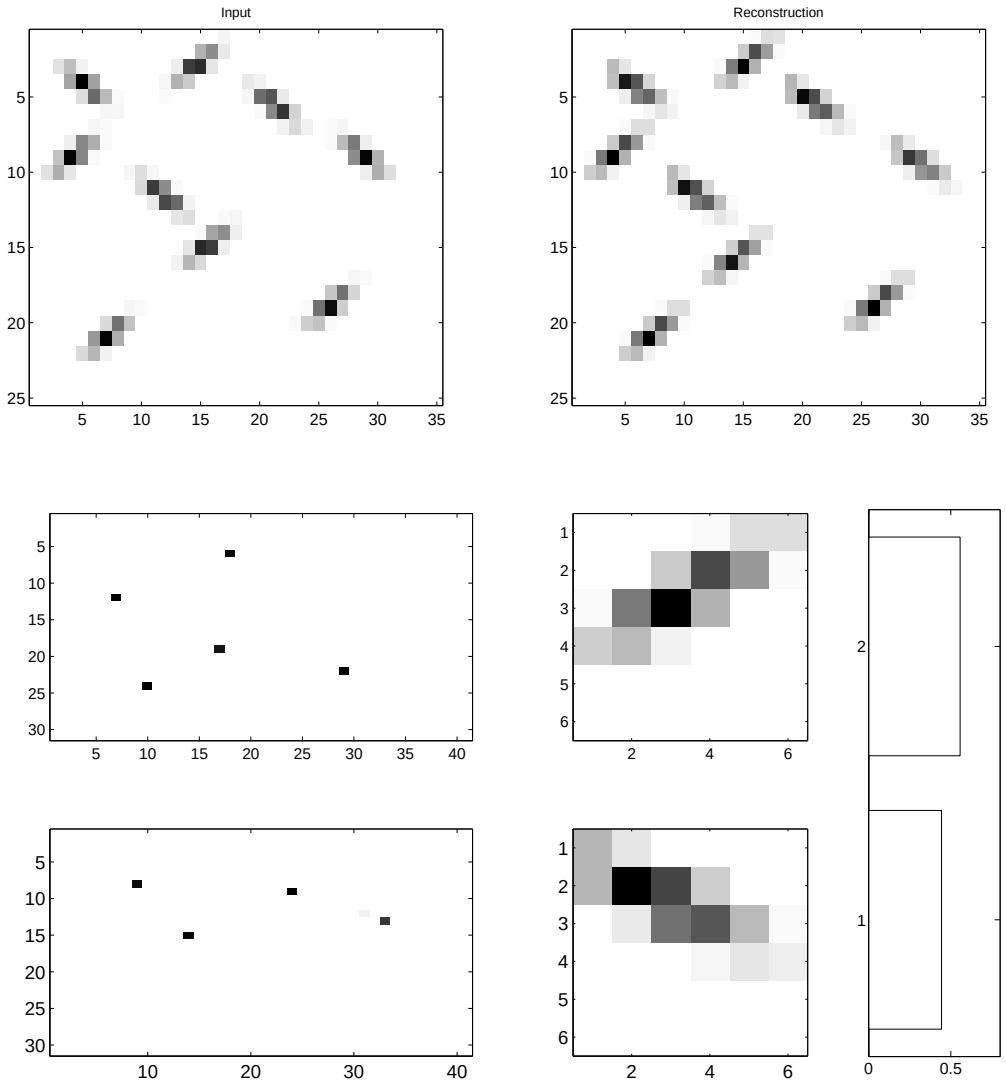


Figure 8: Example of SI-PLCA on toy problem. Input distribution (top left), reconstructed distribution (top right), activations (lower left), kernels (lower center) and mix probabilities (lower right).

size depends on the product of the input and kernels sizes, and the number of kernels. More in detail, given the "deconvolution" operated by the SI-PLCA, we need to zero-pad the input (both in head and in tail, and in both dimensions) and P_I by the size of the kernels/shifts. Being the sizes of the input X and Y, the shifts $\mathcal{T}_x$ and $\mathcal{T}_y$, and K the number of kernels:

- The zero-padded input matrix has size $(X + 2 * \mathcal{T}_x, Y + 2 * \mathcal{T}_y)$;
- the P_I matrix has size $(X + \mathcal{T}_x, Y + \mathcal{T}_y, K)$;
- the P_K matrix has size $(\mathcal{T}_x, \mathcal{T}_y, K)$;
- the P_Z vector has size K;
- the R matrix has size $(X + 2 * \mathcal{T}_x, Y + 2 * \mathcal{T}_y, \mathcal{T}_x, \mathcal{T}_y, K)$.

We have applied the SI-PLCA on our annotated dataset (Sec. 3.1.2). To do this we extract the CQT from each imitation, and then we stack the CQTs along their time dimension (more details in Sec. 3.2.3). The resulting input CQT/histogram of the SI-PLCA has size $(X, Y) = (111, 34037)$. In one of our biggest analysis, we sought for $K = 8$ kernels of size $(\mathcal{T}_x, \mathcal{T}_y) = (80, 5)$: this results in a matrix R which has 2.9526×10^{10} elements, that is about 219.98GiB (using double precision floating point). Further data storage is needed for the P_I distributions, which in our implementation and for the given example have 52.016×10^6 elements: about 396.85MiB.

In order to fulfill the memory requirements and limiting the computing time, we have developed a distributed version of the SI-PLCA algorithm. This has the advantage of running on several machines, thus splitting the needed size of memory and parallelizing the computations. The key idea underlying the design of the distributed algorithm is that **only the kernel distributions need to be shared** among the computing nodes. The input is splitted along the time dimension, and each node receives a portion of it. Then, each node computes and updates the P_I distributions relative to its portion of the input, while the kernels are updated jointly by all the nodes.

This is the workflow of the distributed algorithm:

1. CQT of each input recording is computed.
2. The SI-PLCA input CQT is obtained by juxtaposing CQT along time dimension.
3. The whole CQT is splitted by the number of available computing cores.
4. Each node zeros-pads and normalizes its input.
5. Given the sizes of the input, each node initializes its own P_I, and receives the same initial guesses for P_K and P_Z.
6. Each core computes its own R matrix (Eq. 10).
7. Exploiting R and the input, each core computes P_Z and P_K (Eqs. 11 and 12).
8. P_Z and P_K are sent to the master node; here they are averaged (such to converge to the same set of kernels on all the nodes) and sent back to the nodes.

9. Each node updates its P_I (Eq. 13).
10. Steps 6-9 are repeated until convergence.
11. All the local P_I are convolved with P_K (for each value of z) and results are sent to the master node.
12. If needed, the reconstruction of the input is computed inverting the initial split operation (step 3).

The algorithm has been developed in Matlab, with the core computing routines (each of the Eqs. 10-13) being translated in C. The distribution among several machines is done using the Matlab Parallelization Toolbox [2], which is based on the well-known Message Passing Interface (MPI) paradigm.

Some points in the described algorithm are critical. The step 3 has been implemented by an Overlap And Add (OLA) approach: the boundary regions of the CQT chunks are shared among neighbor nodes, and windowing is applied to avoid abrupt discontinuities in the inputs (this is motivated by the need for a smooth input reconstruction with bi-dimensional convolution, rather than by the SI-PLCA itself). Steps 11-12 are designed correspondingly: the OLA windowing over time has to be precisely inverted. To accomplish this, each node performs the bidimensional convolutions between the local $P_I(x, y|z)$ and $P_K(\tau_x, \tau_y|z)$ (one convolution per value of z), then the results are averaged and weighted by P_Z. Moreover, the nodes delete head and tail zero paddings along the time axis, and invert the normalization applied in step 4. The master node then receives the partial reconstructions and apply OLA to recover the full input reconstruction (inverting step 3).

In step 10 is determined if the algorithm can be terminated. Being an iterative procedure, we need a measure of convergence. The most effective one is the direct comparison between input data and current input reconstruction, by means of a divergence measure. However, input reconstruction at each iteration can be slow. To evaluate the convergence of the algorithm we thus rely on a measure of the rate of change of P_K between iterations: when P_K is (almost) not evolving, so are doing P_I, P_Z and the input reconstruction. The algorithm can thus be stopped.

3.2.3 Application of SI-PLCA to automatically find audio primitives

The distributed algorithm has been exploited to apply the SI-PLCA on the whole dataset introduced in Sec. 3.1.2. As described in Sec. 3.2.1, our aim is to let SI-PLCA automatically discover the audio primitives which characterize the dataset. What we are looking for, in other words, is a link between the manual annotations of the vocal primitives and the kernels. The kernels found by the SI-PLCA play in fact the role of a "dictionary" of primitives found automatically by the learning technique.

In order to apply the SI-PLCA to the dataset, all the audio recordings (about 486s) have been chained together in a longer signal; the CQT transform has then been computed over it. The resulting spectrogram is a very large matrix. Being the analysis range between 70 and 5000Hz, we have around 6 octaves, hence $6 \times 36 = 216$ bins. On the time axis, the chosen toolkit [6] uses about 210 frames per second, thus we have $\approx 102 \times 10^3$ frames. The actual

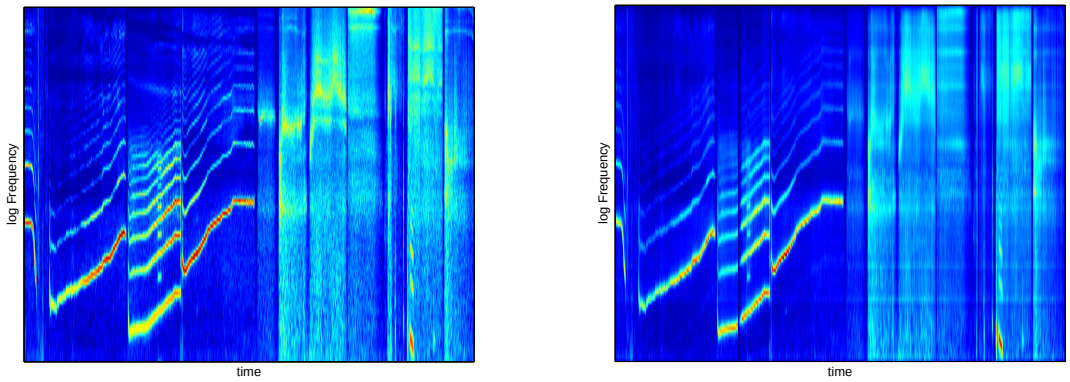


Figure 9: Portion of input CQT: excerpt of about 20s from the dataset (left panel). Reconstruction of the input excerpt by means of 4 kernels (right panel).

CQT matrix size is (222×102111). Since this is not tractable by our computers, we resample the CQT to a lower resolution. Given the bidimensionality of the CQT, we rely on Matlab internal `imresize` routine (image resampling with antialiasing) to downsample by a factor of 2 on frequency axis and of 3 on time axis. The final input matrix for the SI-PLCA has size (111×34037) (recall that we have already taken these as reference sizes when talking about the SI-PLCA implementation in Sec. 3.2.2).

In Fig. 9 is shown an example of the SI-PLCA applied on the dataset; for clarity, the figure shows only a portion (about 20s) of the CQT spectrogram. On the right panel we can see that the reconstruction is effectively describing the main cues of the signal. In this example we used only 4 kernels of size (60×6), shown in the right part of Fig. 10. Each kernel has its own individual contribution to the reconstruction of the input portion (Fig. 10 left panels). First and third kernels are clearly related to the different harmonic contents of the dataset, with the first one also used to enhance noisy parts. Second and fourth kernels are less structured, and are activated mostly in noisy parts.

In Fig. 11 and Fig.12 are shown a part of the results found by SI-PLCA, applied on the whole dataset, with different parameters. Fig. 11 we found 6 kernels of size (30×15); while first (left top) and last (right bottom) are rather unstructured, the remaining are all oriented toward the harmonic content of the dataset. Interestingly, since the time dimension of the kernels is bigger than in Fig. 10, it is clear how the second (and the fifth) kernel is well-suited to describe impulsive content: in fact, its time domain appears lumped. In our CQT spectrograms harmonics are spaced by 18 bins (36 bins per octave downsampled by 2), so the kernels size in frequency has to be at least 18 to exploit the shift-invariance. In the light of this limit we present in Fig. 12 yet another set of kernels: 8 with size (25×25). The fourth kernel correctly catches the harmonic structure, but the extended time axis lets the kernels show more complex primitives.

Due to the memory requirements of our algorithm we can not compute larger families of kernels (bigger cardinality and/or bigger kernels). Moreover, increasing the kernels size means increasing their specificity, such that an even larger number of kernels is required to accurately

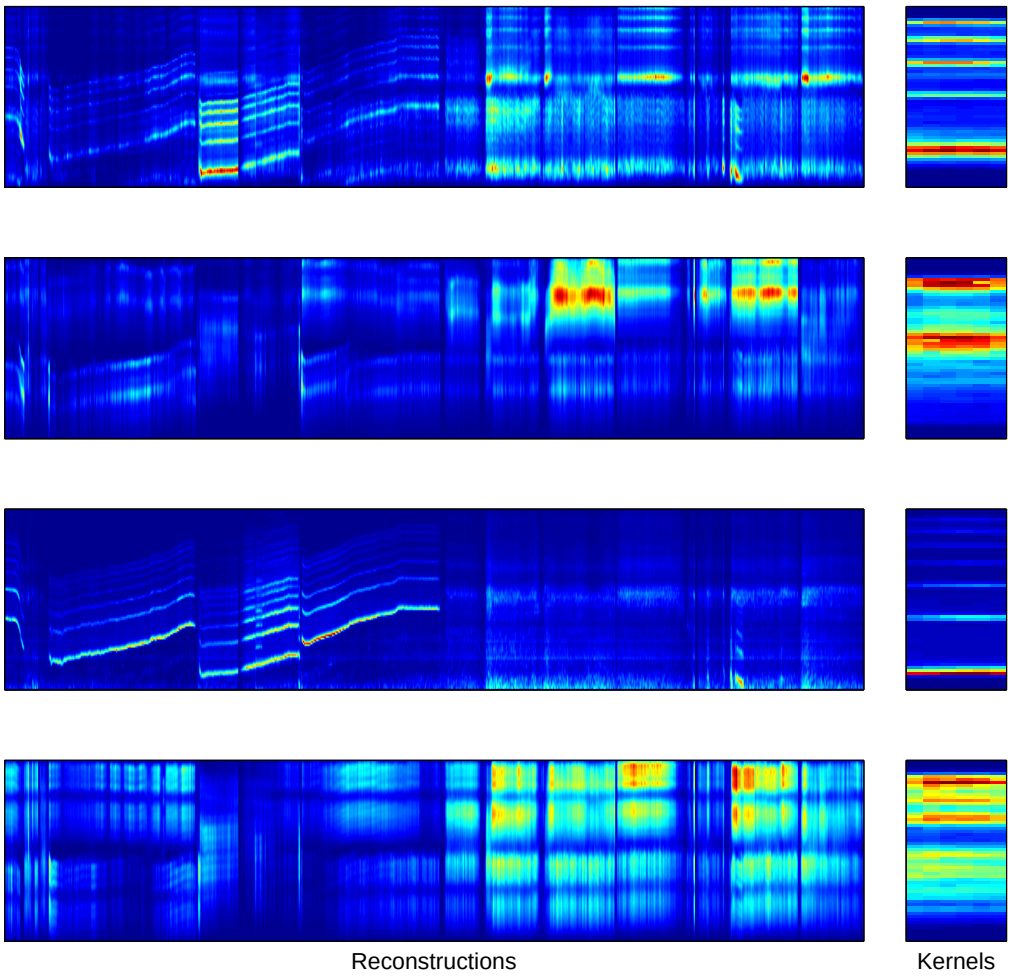


Figure 10: Contribution of the kernels to the reconstruction shown in Fig. 9 (left panels). The corresponding found kernels of size (60×6) (right panels). We remark that the frequency axes in left and right panels are different.

describe the input. A more refined, memory-optimized implementation of our algorithm, based on the computation of *sufficient statistics* instead of the matrix R, will allow for the aforementioned experiments.

3.2.4 Comparison between manually annotated and automatically found by SI-PLCA audio primitives

The aim of computing the SI-PLCA on the dataset is, as stated in Sec. 3.2.1, to automatically find the acoustical primitives eventually present in the recordings.

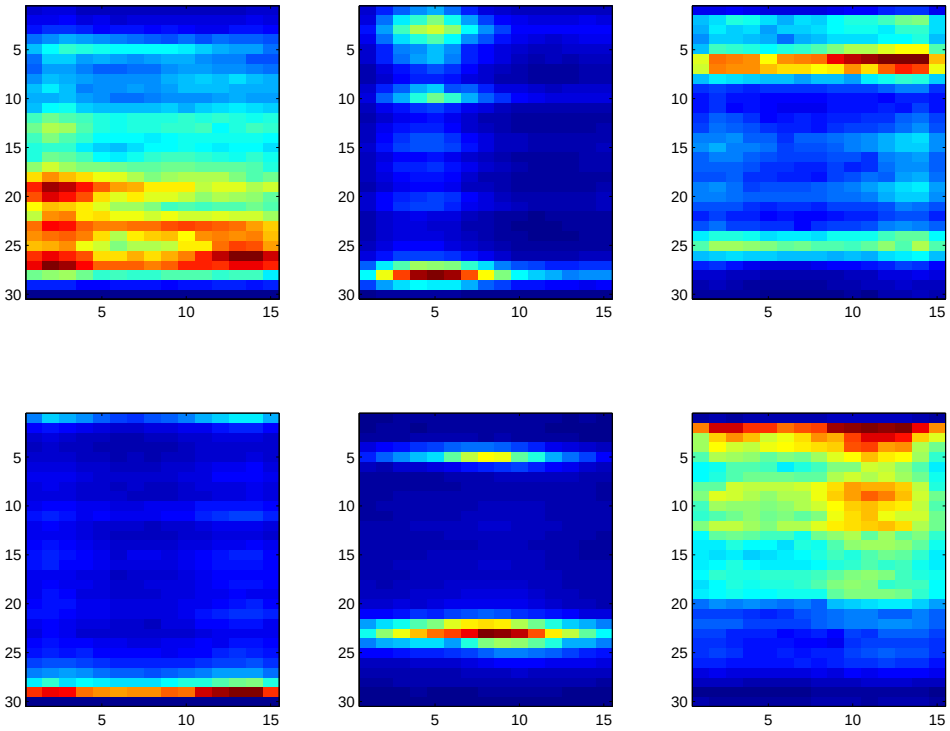


Figure 11: Six kernels of size (30×15) found by SI-PLCA over the whole dataset. Kernels with lumped support in time (pulses) or frequency (harmonics) emerge. In text, the numeration is from the left, top to the right, bottom row by row.

In order to verify the effectiveness of SI-PLCA decomposition on this task, we exploit the manual annotations that have been introduced in Sec. 3.1.2. In principle, we would like to find some kind of correspondence between the annotated labels and the kernels. If a kernel k is often activated to describe regions which have a certain label l, then we conclude that k and l are somewhat linked: the kernel k successfully describes the acoustical cue of the labeled primitive.

As a first step in this evaluation, we match the annotated regions of each spectrogram (Fig. 2) to the reconstruction contributions given by each kernel (Fig. 10, left column). For each region of each audio file, we compute the per-kernel energy of the reconstruction. We then normalize by the size of the region, and compute a mean over same-labeled regions. The result is a matrix $\mathcal{R}_K$ with a row for each label and a column for each kernel, displaying for each label the average kernel reconstruction contribution.

Taking as a first experiment the six kernels depicted in Fig. 11, each with size (30×15), we see in Fig. 13 (left panel) the corresponding matrix $\mathcal{R}_K$. At a first glance, the fifth column of the matrix seems to be sorted (increasing values from top to bottom): in fact, the fifth kernel has an harmonic structure, and at the bottom of the matrix there are labels with harmonic content: HL, HS, HQS.

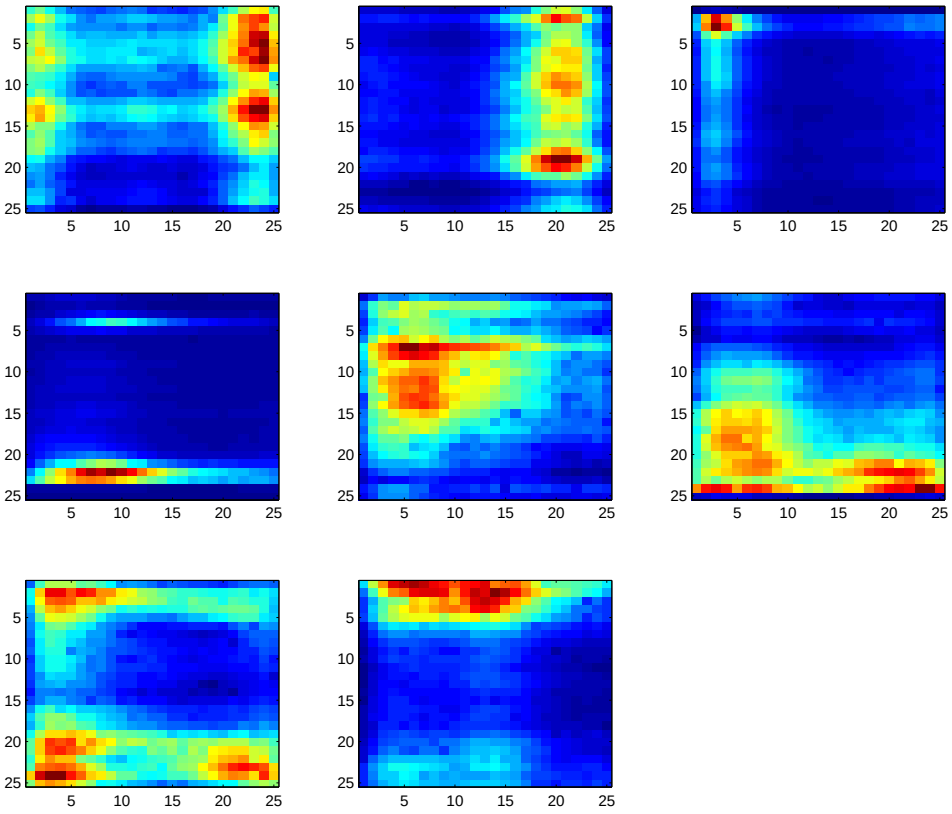


Figure 12: Eight kernels of size (25×25) found by SI-PLCA over the whole dataset. Kernels with lumped support (pulses and harmonics) are found, along with more complex structures. In text, the numeration is from the left, top to the right, bottom row by row.

To confirm this intuition, we compute the matrix $\mathcal{C}_K$ of the correlation coefficients between each pair of rows of $\mathcal{R}_K$. Each element of $\mathcal{C}_K$ in position (i, j) is found as the corresponding covariance between rows i and j in $\mathcal{R}_K$, normalized by the geometrical average of two rows variances. Furthermore, we sort the rows and columns of $\mathcal{C}_K$ by computing a dendrogram on the complete linkage of $\mathcal{C}_K$ rows. Fig. 13 central and right panels show, respectively, $\mathcal{C}_K$ and the dendrogram.

By looking at the figure, grouping between labels with similar content emerges. We can see the groups R{V,S}, N{S,QS,V,N,_}, H{RL,NS,NL,L,S,QS}. In other cases the groups are less clear (it is the case for R{L,_}, or H); we point out that these labels come from a manual annotation, which is likely to be not always correct and objective.

In Fig. 14 there is the evaluation of another SI-PLCA decomposition: in this case we used 8 kernels of size (25×25), the same shown in Fig. 12. For the sake of graphic representation, in this case the correlation coefficients have been exponentiated to let the grouping emerge a bit more. Even if different from the previous example, and less effective, the clustering among

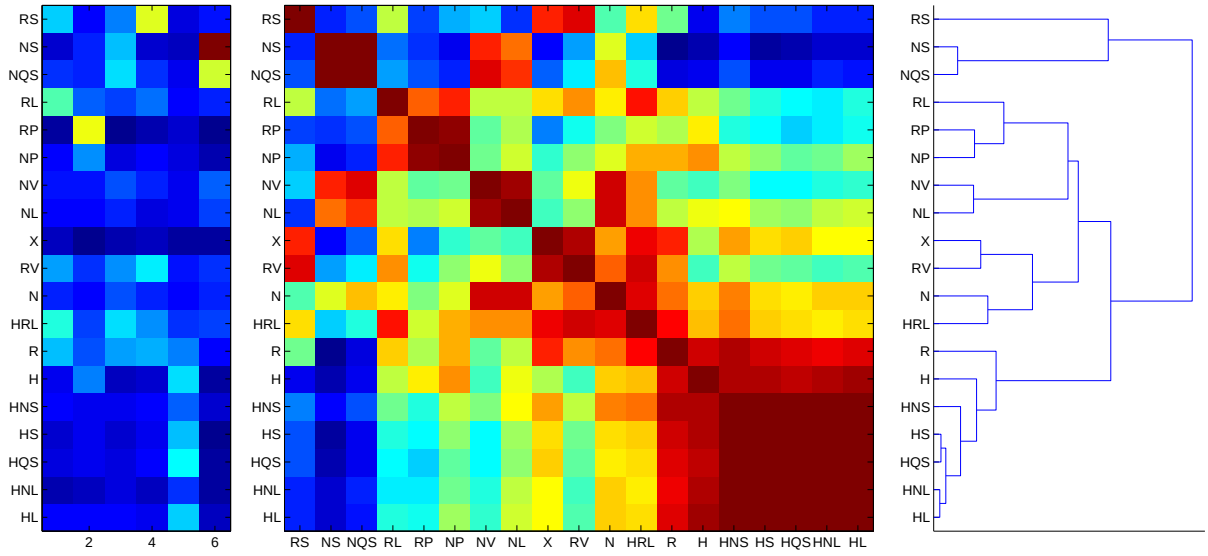


Figure 13: Labels correlations by kernel usage on 6 kernels of size (30×15). From left to right: kernel activations for each label, labels correlations computed on the kernel activations, resulting dendrogram for the labels. Grouping between similar labels confirms that kernels are associate to audio primitives.

Noise, Roughness and Harmonic primitives is recognizable (especially from the dendrogram).

3.2.5 Recognition of imitation categories using automatically found audio primitives

In previous Sec. 3.2.4 we established a correlation between the kernels/bases found by SI-PLCA decompositions and the acoustical primitives which have been manually annotated on the dataset. Recalling one of the aims of our work (Sec. 1), we would like to test if the kernels can be used to train temporal models of the imitations. The interest in this approach is twofold:

- support the effectiveness of SI-PLCA in finding meaningful acoustic bases, without manual intervention;
- assess a recognition by means of automatically designed audio features.

To test the automatic classification we proceed in the following way. We begin by computing the SI-PLCA decomposition of the full dataset using K kernels, shared among the whole dataset. For each of the $N = 115$ audio files we then find K reconstructions $\mathcal{S}_{n,k}$, due to the individual kernels: these are the spectrograms obtained by convolving each kernel k with its corresponding activations for the given input recording n. An example of these per-kernel contributions has already been discussed (Fig. 10, left panels).

We now need to convert the reconstructions into descriptors suitable for a statistical modeling. To this aim, each of the $\mathcal{S}_{n,k}$ spectrograms is summed over its (log)frequency bins: this

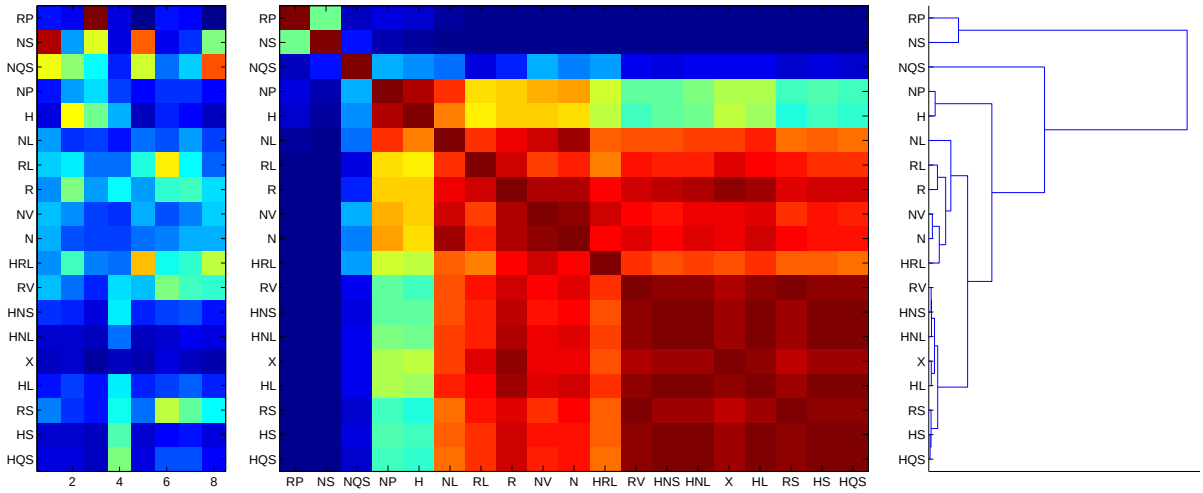


Figure 14: Labels correlations by kernel usage on 8 kernels of size (25×25). From left to right: kernel activations for each label, labels exponentiated correlations computed on the kernel activations, resulting dendrogram for the labels.

results in a vector $\mathbf{d}_{n,k}(t)$ which evolves only along the time dimension. In this way the values of $\mathbf{d}_{n,k}(t)$ are proportional to the energy of the k-th reconstruction at each time t.

For each input audio file n we end up with K vectors $\mathbf{d}_{n,k}(t)$: these are then stacked together in a matrix $\mathcal{D}_n(k,t)$ and normalized such to have, for each time frame, values in the range $[0,1]$. The matrices $\mathcal{D}_n$ (one per audio file) are then used as descriptors for the classification.

To carry out the recognition we need to model the temporal evolution of each descriptor, hence we chose to use hidden Markov models (HMM). To model each category we train an HMM with s hidden states (which has to be enough even for repetitive categories). The emission probability of each state is modeled by a gaussian mixture model (GMM) with g gaussians, which has as many dimensions as the descriptors (that is, the number of SI-PLCA kernels).

Given the reduced amount of examples in the dataset (115), we use only three folds for the crossvalidation of the results (filtered by subject). In Tab. 4 are shown the figures obtained by the automatic recognition with the aforementioned method. The results refer to the two kernel configurations which have been previously discussed in Sec. 3.2.3: 6 kernels of size (30×15) and 8 kernels of size (25×25). The shown figures have been obtained by tuning the HMM parameters. For the configuration with 6 kernels, we set $s = 30$, $g = 1$ and the underlying GMMs use a diagonal covariance matrix. With the 8 kernels configuration, we set $s = 35$, $g = 1$ and full covariance matrix.

The found results are compared with a baseline system. It is exactly the same already presented in Deliverable D5.5.1 (Sec. 2.7), based on DTW alignment of time series; the classification is done by a k-Nearest Neighbor (k-NN) algorithm. Despite the baseline system is performing better than the proposed method based on SI-PLCA, we can confirm that the latter is working as expected.

In Fig. 15 is shown an example of HMM decoding of one imitation; in the image we use the 6 kernels configuration (please refer to Fig. 11 for the kernels contents). In the topmost panel we see that the input is made by an alternance of noisy and harmonic content. In the harmonic parts, the decoded state is the number 9 and, looking at the bottom panel, this state has a strong emphasis on kernel number 5 (Fig. 11). The fifth kernel is in fact the most suitable to describe harmonic contents. Similarly, noisy parts are decoded by the state number 8: this one is mostly associated to kernel number 6, that is a "noisy" one. The last part of the input is mostly decoded as state number 2, which is a mixture of kernels 4 and, to a lesser extent, 2 and 5: the harmonic content is thus confirmed.

Given the recognition results and the example of decoding, we can conclude that the computed descriptors are correctly summarizing the content of the imitations. In other words, the proposed unsupervised learning procedure successfully finds a meaningful set of acoustical primitives: the aforementioned kernels.

Decompositions:	6 ker. (30×15)		8 ker. (25×25)		DTW baseline	
Categories	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
blowing	41.67	51.85	41.67	83.33	66.67	77.78
vehicleext	11.11	11.11	0.00	0.00	22.22	66.67
vehicleint	8.33	33.33	25.00	22.22	83.33	41.67
rubbing	100.00	48.15	91.67	59.17	91.67	69.44
filing	52.78	40.00	61.11	50.00	0.00	0.00
dripping	41.67	66.67	36.11	38.89	55.56	72.22
filling	25.00	50.00	75.00	83.33	83.33	48.15
mixers	8.33	33.33	16.67	33.33	83.33	72.22
fridge	58.33	88.89	66.67	66.67	16.67	16.67
crumpling	83.33	74.60	50.00	47.22	83.33	69.84
shooting	100.00	82.22	100.00	77.78	91.67	93.33
hitting	50.00	21.67	33.33	15.87	50.00	64.44
Mean val.	48.38	50.15	49.77	48.15	60.65	57.70
Accuracy	49.55		50.77		61.01	

Table 4: Automatic recognition results obtained using descriptors found by SI-PLCA decomposition. Two configurations and a baseline system are compared. See text for details.

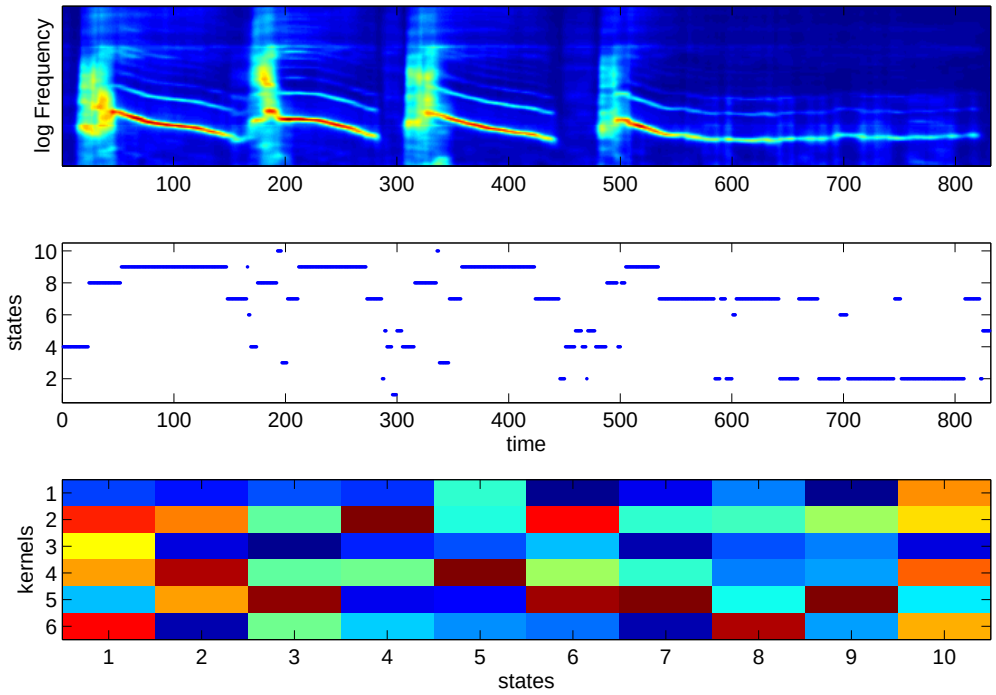


Figure 15: Example of HMM decoding. Reconstructed CQT by SI-PLCA kernels (topmost panel), decoding of input imitation by the HMM, HMM states centroids for the depicted decoding.

4 Gestures Recognition

We describe here an extension of the gesture analysis presented in D5.5.1. The dataset is composed of recorded gestures performed concurrently to a vocal imitation. We found from qualitative observations that different gestural strategies can be associated to frequency components of the movement, which seems related to metaphoric representation of sound characteristics. Therefore, we proposed in D5.5.1 to use a continuous wavelet transform which allows to characterize typical oscillatory movements (Low frequency range from 0.2–0.5Hz to 50Hz, at framerates about 100 to 500Hz)[1]. Wavelet transform [16] provides a multi-resolution solution that derives a high localization both in time and frequency.

We focused here on two main tasks:

- The development of a classification system for identifying primitives (created manually or automatically). This implies deriving gesture-level descriptors.
- A clustering system for defining automatic primitives that group similar gestures in the dataset.

4.1 Database

For computing the features we use the database created in the early stage of the project. After filtering (removing errors, e.g. participants that did not follow instructions, etc.) 1576 gestures were kept from 35 different participants.

The database is manually segmented and annotated according to the taxonomy presented in D5.5.1. and recalled below. The segmentation allows for isolating active regions, where the actual gesture occurred, and for removing parts not directly linked to the vocalization (such as the movement performed to reach the mouse after the gesture) [2].

Manual labeling of primitives Six different gesture primitives where proposed from qualitative observation. They were defined defined by their frequency components:

- *Steady:* gestures that practically do not change during time. It ranges from purely steady gesture to really slow evolution.
- *Smooth:* fluid and gradual movements
- *Dynamic:* abrupt, energetic and rapid actions.
- *Impulse:* single and sudden excitations.
- *Periodic:* encloses all motions that have a periodicity in time.
- *Shaky:* gestures that involves hand shaking.

The dataset has been annotated by one annotator, but we plan to complete this annotation task with a larger numbers of annotators.

[1] as described in D5.5.1, such frequency ranges are not well resolved using fixed resolution analysis transforms such as FFT

[2] participants had to click a mouse for ending the experiment which introduces a constant artifact in all the recordings

Dynamic	Impulse	Periodic	Shaky	Smooth	Steady
301	244	340	103	308	280

Table 5: Gestures distributions according to manual labeling

4.2 Features extraction.

The first step is to obtain a set of gesture features that characterize the different 'frequency' behaviors, so that gesture primitives can be properly described. These descriptors are then used for both the *clustering* system that defines the automatic primitives and the *classification* step that labels gestures with respect to primitives (manual or automatic).

4.2.1 Sensors.

Gestures are tracked with inertial measurement units with 3D accelerometer and 3-axis gyroscope (Modular Musical Objects (MO)[14]). Each participant has two, one per wrist.

4.2.2 Wavelet.

As described in details in D5.5.1, the frequency analysis is carried out with the wavelet transform. The offline mode is used. Each accelerometer axis is analyzed independently so that the final gesture scalogram is the weighted sum of the scalograms on each axis as shown in Figure 16.

The transformation is calculated using as *parameters:* base wavelet = Morlet, samplerate = 200., frequency range = 0.2Hz–50Hz, omega0 = 5.0, bands per octave = 8, delay ratio = 1.5. As a result, 64 different frequency scales are computed.

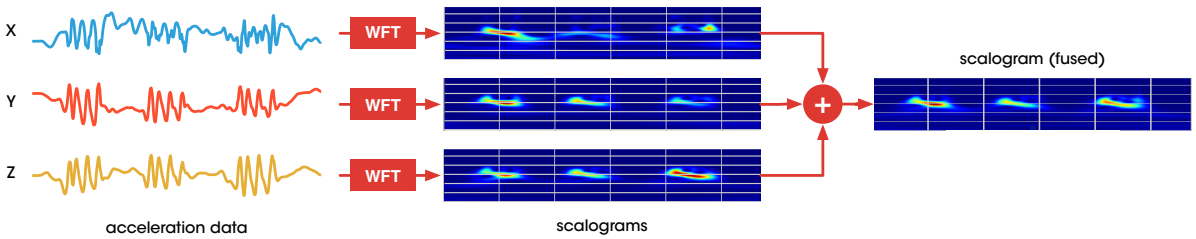


Figure 16: Overview of the wavelet analysis process.

4.2.3 Nonnegatice Matrix Factorization (NMF).

In analyzing the different scalograms, we found that gestures within the same category seem to be characterized by similar combinations of basic 'shapes' defined time-scale space. In order to confirm this hypothesis, and thus quantitatively characterize such shapes, we proposed to use Non-negative matrix factorization (NMF). In this technique a non-negative matrix is described

as a product of two also non-negative matrices:

$$V_{f,t} \approx (W \times H)_{f,t} = \sum_{i=1}^{k} W_{f,k} H_{k,t}. \tag{16}$$

where *k* is the number of basis components wanted to be found, in our case $k = 25$. $V \in R^{f \times t}$ is the original non-negative data, $W \in R^{f \times k}$ the basis vector decomposition and $H \in R^{k \times t}$ the weight matrix. Each column of *V* represents a F-dimensional data sample and each row a data feature. In *W*, each column is referred to as a basis vector. Rows of *H* stand for the gain of corresponding basis vector. *k* is chosen so that $(f + t) \times k < (f \times t)$. The product $W \times H$ presents a compressed form of the original data V.

In our context, NMF can provide the basic scalogram shapes of the dataset. These basic shapes can be then combined to describe the full scalogram associated to each gesture primitive.

Preprocessing. Following the methodology presented at [15] the scalgoram is treated as an image. While the amplitude of each scalogram is normalized between 0 to 255[3] (float values), time is compressed or expanded to a fix resolution (500). This means for examples that two gestures with different length and amplitude ranges are transformed into a 500 length scalogram with amplitude values that range from 0 to 255. Therefore, total time and amplitude information of a given gesture are lost. We will describe in section 4.2.4 how we will recover such information for clustering and classification.

The V matrix. This matrix is the result of the concatenation of each gesture in the dataset after flattening its information[4]. This decomposition and concatenation permits the NMF technique to find coherent scalogram motifs in different locations and with complex forms.

Basic components. *W* and *H* are calculated resolving the equation:

$$\begin{aligned} &\min_{W,H} f(W,H) \equiv \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} (V_{i,j} - (WH)_{i,j})^2 \\ &\text{subject to } W_{i,a} \geq 0, H_{b,j} \geq 0, \forall i, a, b, j. \end{aligned} \tag{17}$$

There are several approaches to solve (17). We choose the 'Alternating Nonnegative Least Squares Matrix Factorization Using Projected Gradient or LSNMF' [11]. It converges fast and returns small approximation errors in our case. LSNMF solves bound constrained subproblems using the projected gradient method.

[3] as an analogy with images.

[4] the original size of a scalgoram matrix after preprocessing, is 64 scale bands $\times$ 500. Flattening is the process of transforming this matrix into a vector of 32000 length. This process is performed reading the information from left to right and top to bottom.

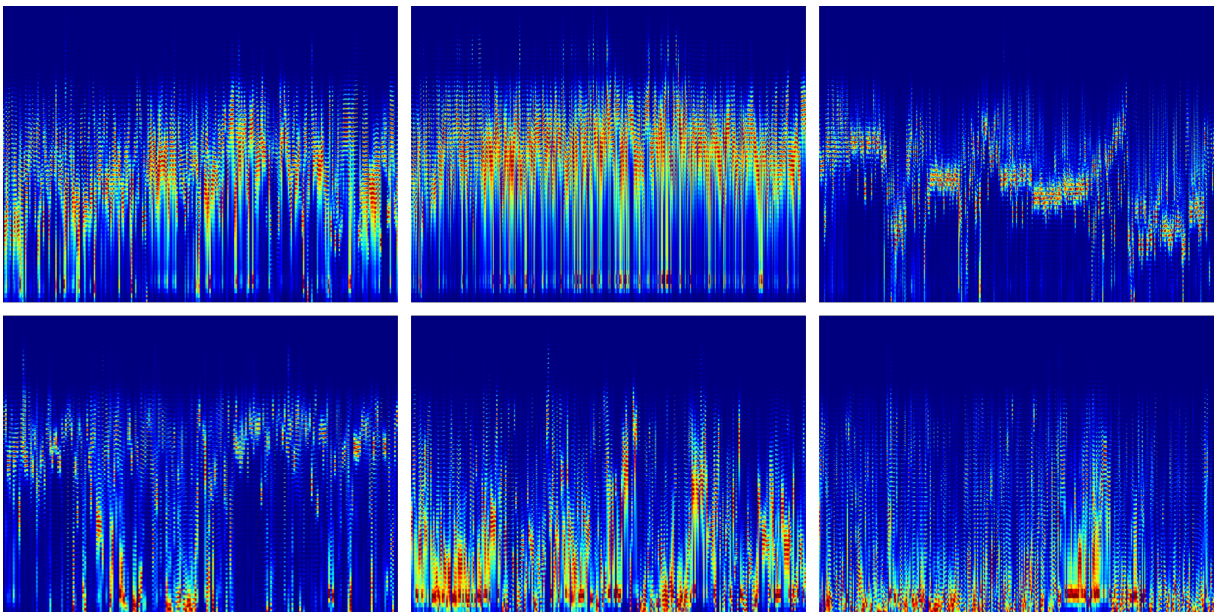


Figure 17: V matrix: Scalogram flattened and concatenated. Top: Dynamic, Impulse, Periodic. Bottom: Shaky, Smooth, Steady.

New gestures. Each gesture scalogram (new or already known) is described as a linear combination of the *k* basis vectors:

$$\arg\min_{x} = \|Ax - b\|^2 \text{ for } x >= 0 \tag{18}$$

where *x* is the weight of each component to be calculated, *A* the vector of components *W* and *b* the original scalogram. Only additive combinations (negative values are not permitted) are allowed. Any given scalogram does not have to use all the available components.

4.2.4 Complementary features.

Two other descriptors are added to recover from the information in the normalization process, such as *time* (duration in seconds or samples) or *energy* (total energy of the original scalogram)[5].

4.2.5 Normalization.

We found that the values of each descriptor follow a exponential distribution. For this reason we applied $ln(x+1)$ transformation. Finally, the dataset is standardized centering and scaling each feature independently (zero mean and unit variance distribution).

[5]Other features could be also includes such as the one derived from the multiple-mode frequency tracking presented in D.5.5.

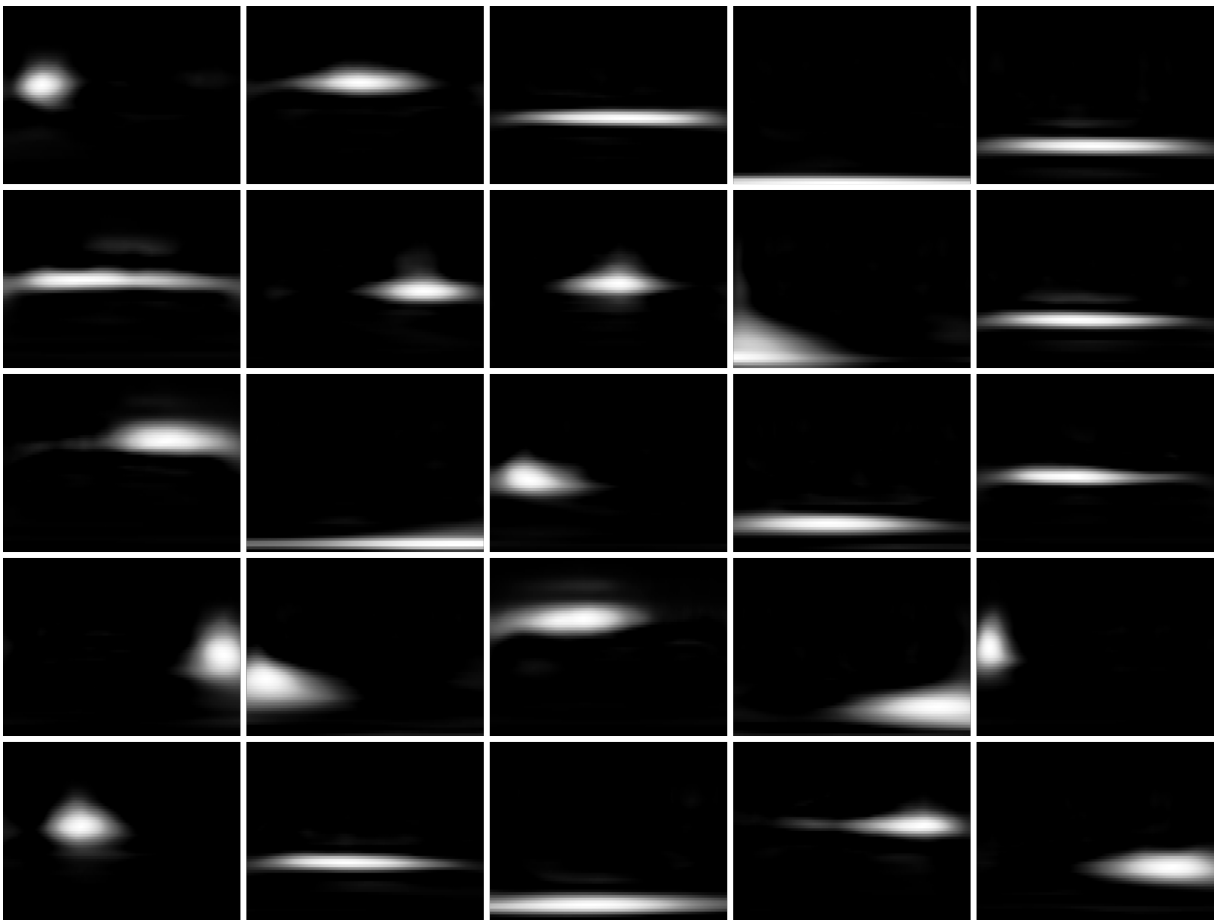


Figure 18: The 25 basic components of our database (after reshaping them). Each one represents a basis shape spatially localized (time and amplitude) in our gesture scalogram collection. They can be seen as a collection of descriptors.

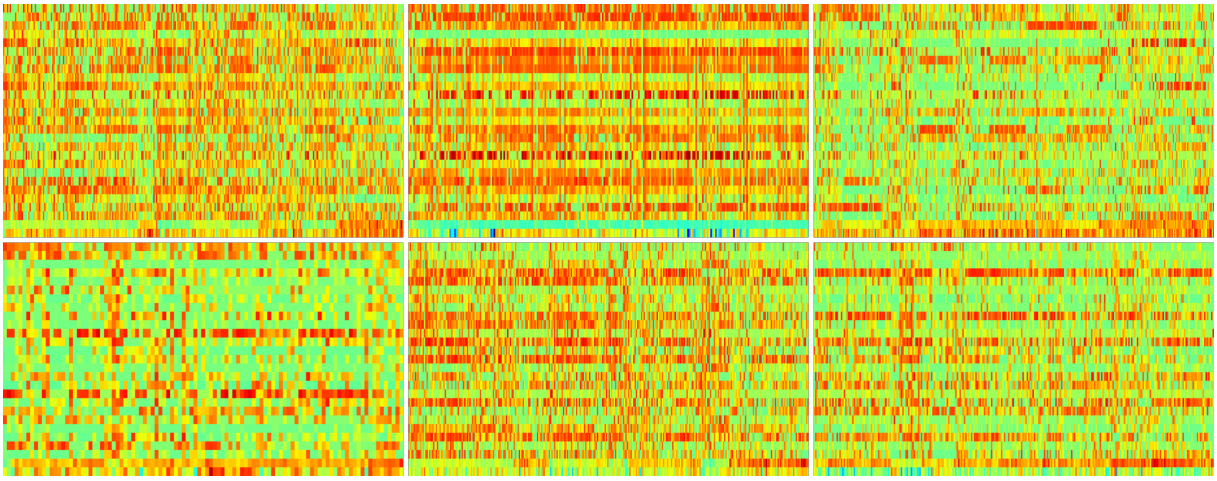


Figure 19: The database described as feature values. Rows represent features and columns represent gestures. Top: Dynamic, Impulse, Periodic. Bottom: Shaky, Smooth, Steady.

4.2.6 Discussion.

The use of the NMF decomposition has been successful for modeling complex scalogram as a mixture of prototypical basic shapes, consistent across participants. It provides thus a compact and reduced representation of the gesture scalogram. In the following sections, we will show how clustering can be operated using these components and the recognition of gesture primitives.

4.3 Clustering

In order to discover significant groups of gestures, the 27 dimensions[6] feature space previously described is clustered. This is achieved running a K-Means algorithm [12] on all gestures in the dataset.

We also evaluated clustering using GMMs, where each Gaussian distribution corresponds to each class [7]. We found that this method was less effective and less consistent compared to K-Means. Hence, we are not presenting these results.

Finding an appropriate number of cluster that optimally partitions the data set is critical. The use of prior knowledge on the context can allow one to set a number of clusters. In our case, we set the number of cluster to six, to compare the automatic clustering to the six primitives defined manually.

4.3.1 Clustering evaluation

The automatic gesture primitives found are analyzed with respect to both, quantitative and qualitative criteria.

Qualitative: The visualization of the different discovered clusters allow to evaluate qualitatively the validity of the results. The gestures space was reduced into a 3D space using Kernel[8] Principal component analysis (KPCA). With this technique the 48% of the variance of the original space remains. In this space we can manually evaluate the distribution of the different clusters.

Clusters cover different areas of our feature space. Analyzing the distribution of points shows that there is some overlap in the border of the clusters.

Clusters can be also analyzed according to the distribution of manual primitives, as presented in Figure 21. It is shown that cluster number one is the combination of dynamic and impulse gestures, number two includes mostly steady and smooth gestures, three a different type of impulse, four is a mix between periodic and dynamic, five periodic and shaky and six a combination of smooth and dynamic. These results are consistent with the problems found during the annotations process where the threshold between some labels is ambiguous.

The center of each cluster can be described as a scalogram, using the values of each of their NMF components as presented in Table 6 and Figure 22. These scalograms give

[6] the 25 basic vectors of the NMF + global time and energy.

[7] several configurations were used: adding extra Gaussian for modeling outliers and different covariance matrix.

[8] kernel = 'sigmoid'

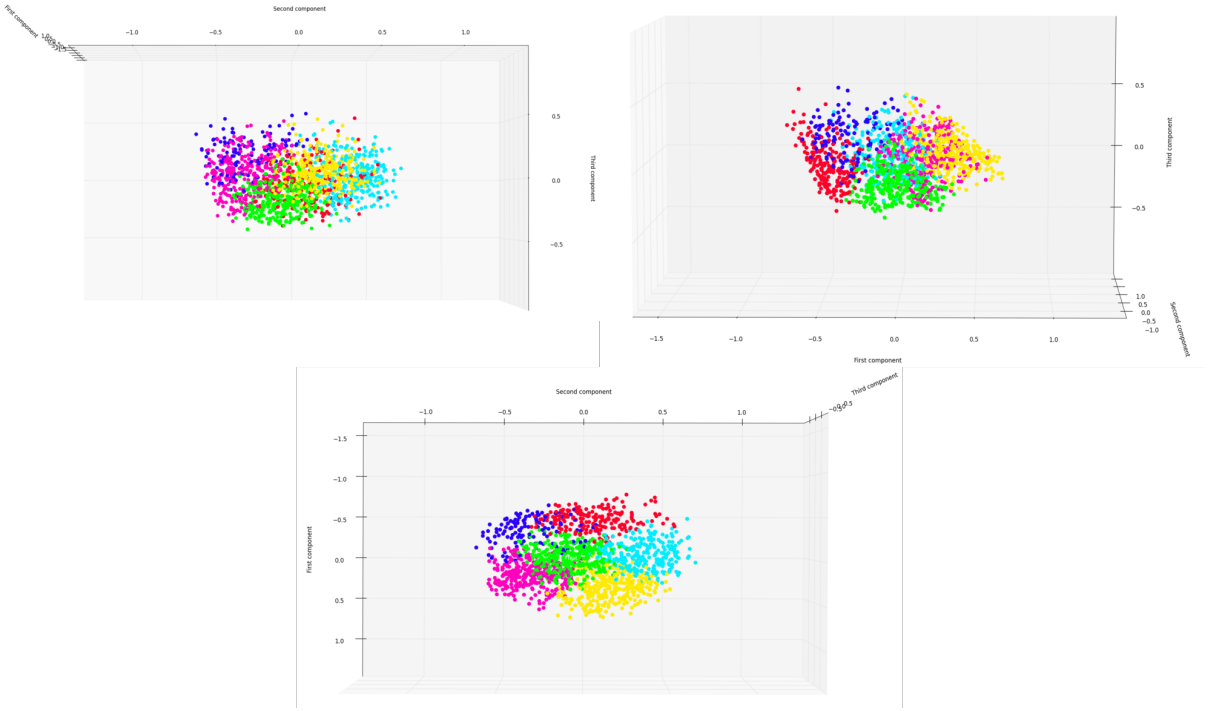


Figure 20: Dataset distribution with respect to the automatic primitives. Space has been reduce to three dimensions using PCA. The three figures corresponds to three different views of the same 3D space built from the first 3 principal components

significant information about the different clusters in the frequency domain. For instance, the two clusters that comprise impulse gestures respond to different frequency areas.

	0	1	2	3	4	5
Time (s)	0.86	5.87	0.67	4.70	4.09	2.93
Energy	1.86×10^9	1.18×10^9	1.00×10^9	8.33×10^9	6.01×10^9	1.73×10^9

Table 6: Time and energy values for cluster centers

The final part of the quantitative analysis consists on retrieving the *10*-closest gesture to the centers. With the visualization of these videos, we ascertain that results are consistent. This opens interesting perspectives for future psycho-acoustic evaluations where participants will compare different primitives and evaluate the coherence of each group.

Quantitative: Analysis of different standard scores that measure the consistency of the clustering process. Two main techniques: *supervised* that rely on a ground-truth and *unsupervised* which studies the shape and characteristics of the clusters themselves.

Supervised methods: They allow for comparing the similarity between two label sets. In our case, they can be used to compare the automatic clustering with manual primitive labeling. The results of the principal measurements can be found in Table 7. The main issue of such an evaluation is the possible inconsistencies in the manual annotations.

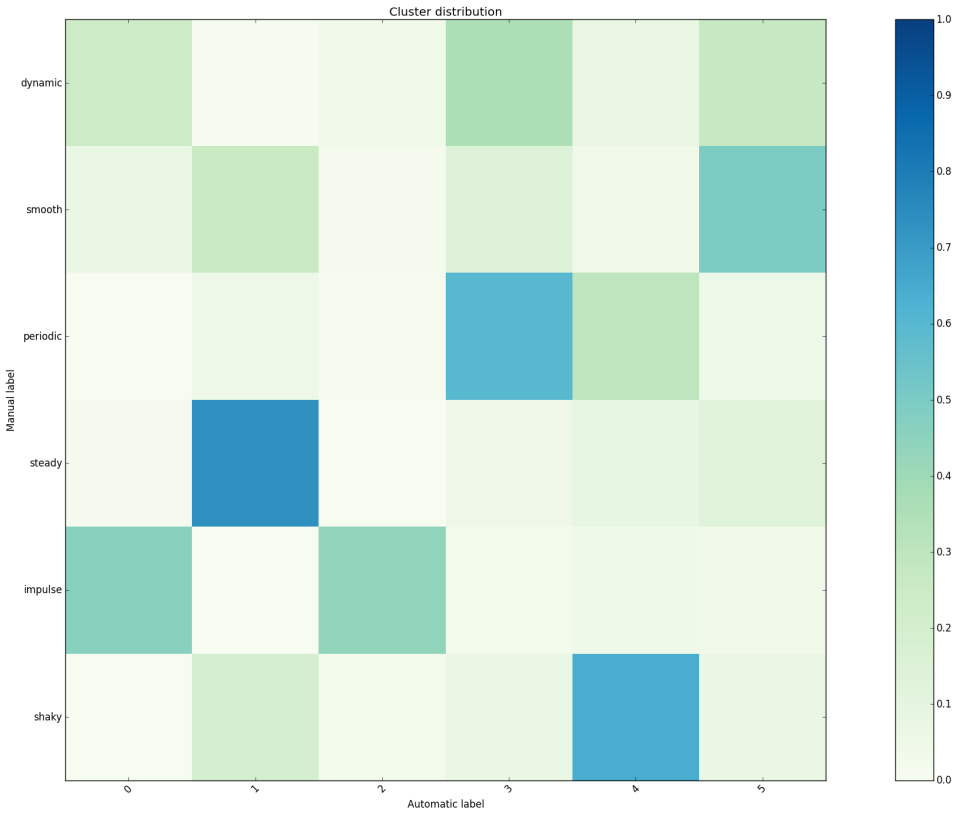


Figure 21: Cluster distribution

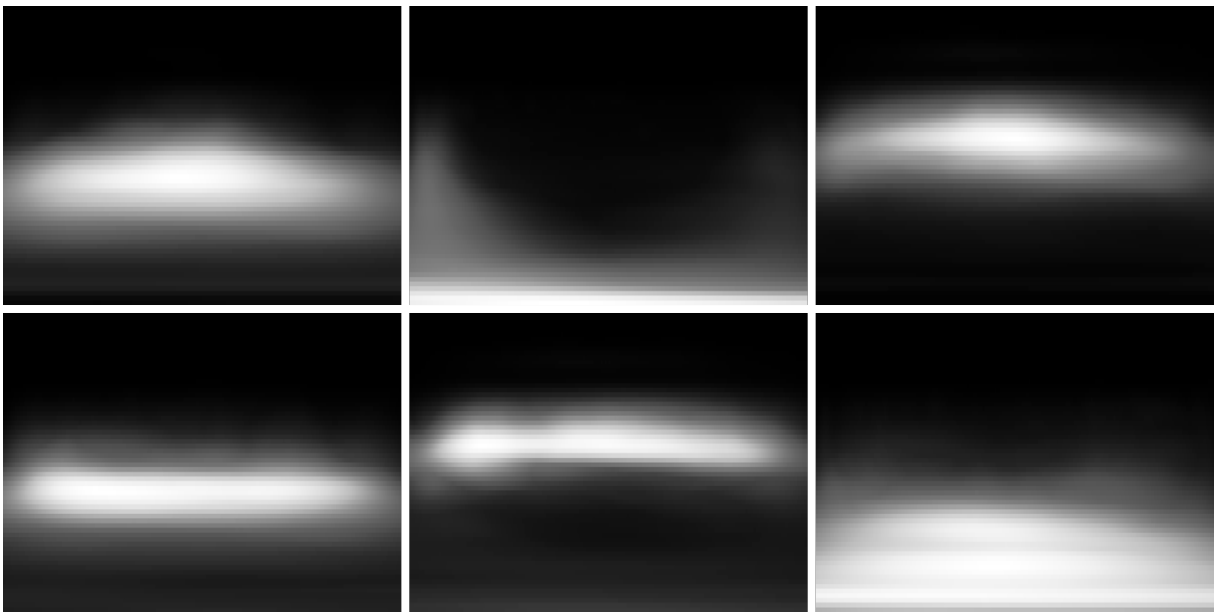


Figure 22: Centers of each cluster represented as scalograms reconstructed from the NMF descriptors. Top: classes 0, 1 and 2 Bottom: classes 3, 4 and 5.

Unsupervised methods: They measure the 'quality' of the model itself. Two main

AMI	Rand Index	V measure	Completeness	Homogeneity
0.3373	0.2641	0.3405	0.3403	0.3407

Table 7: Supervised scores for clustering evaluation. See appendix 4.4 for definitions.

aspects are considered.

- **Compactness:** points inside a cluster should be as close to each other as possible.
- **Separation:** clusters should be widely spaced. Three possible distances between clusters: closest members, most distant and centers of the clusters.

Dunn	DB	Silhouette	CH	SS	PB	PBM
0.3414	2.2879	0.1038	0.819	29.30	741.695	0.674

Table 8: Unsupervised scores for clustering evaluation. See appendix 4.4 for definitions.

4.3.2 Classification

Two standard techniques are used for the recognition of gesture primitives: K-nearest neighborhoods (KNN) and Gaussian Mixture Models (GMM)[13].

4.3.3 K-nearest neighborhoods

For KNN, k is equal to the 4 closest neighborhoods. The final classification is assigned to the most predominant class in an uniform way without weighting with respect to their closeness. Several metric distances [9] where used, obtaining the best results in both, manual and automatic classes, with Minkowski order 2 distances which is equal to Euclidean distances.

	Accuracy	Precision	Recall	F1 score
Manual primitives	0.6586	0.6472	0.6314	0.6366
Automatic primitives	0.8743	0.8776	0.8721	0.8724

Table 9: K-nearest neighborhoods classification metrics

As expected the classification of the automatic primitives is highly precise. K-means clustering follows an orthogonality division of the space as well as the KNN search due to the Euclidean distance used.

Analyzing the classification of manual primitives, we can see how the confusions are similar to the problems found in the manual labeling. The main misclassification is between the classes dynamic and smooth. The boundary between these two categories is diffuse and complex to establish. The worse classification category is shaky which is a special case of periodic, related with the shaking hands. This movement has a high frequency and normally a low amplitude that explain the confusion with the steady and periodic classes.

[9] Euclidean, Manhattan, Chebyshev, Minkowski with order 3 and 4.

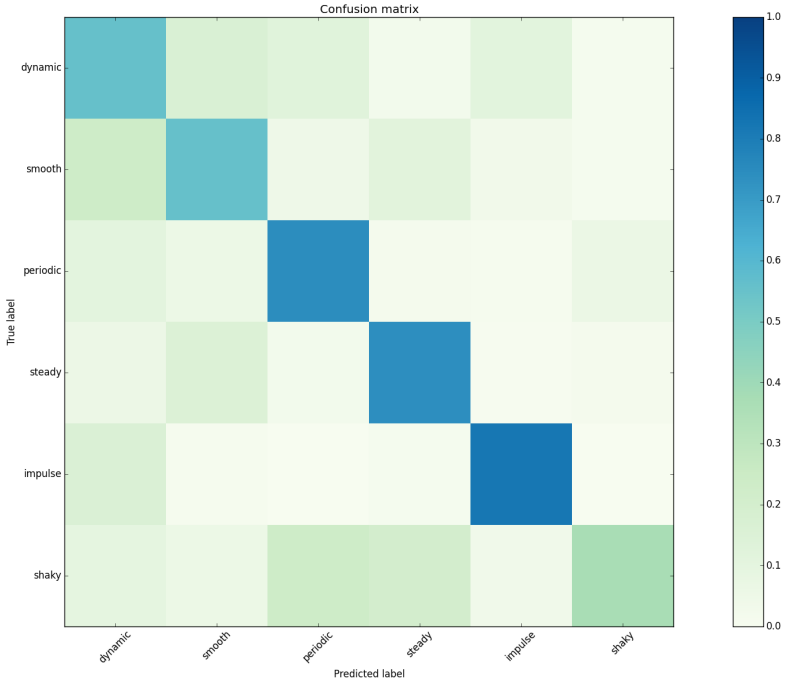


Figure 23: Confusion matrix for the KNN classification of manual primitives.

4.3.4 Gaussian Mixture Model

The methodology follows a standard 10-fold cross-validation using the whole database, so that, at each fold the 90% of gestures of each category is used for training and 10% for testing. The training set is employed to create a GMM where the data points of each label are modeled as one o more Gaussians.

In this context, the classification of a data point is the results of finding the label of the Gaussian it mostly probably belong to. Several configuration have been used such as, modeling each class with one or more Gaussians [10] and different covariance matrices[11]. Results are presented in Table 10. The optimal configuration is with two Gaussians per class and diagonal as a covariance matrix.

	Accuracy	Precision	Recall	F1 score
Manual primitives	0.624 ± 0.049	0.635 ± 0.048	0.648 ± 0.058	0.617 ± 0.047
Automatic primitives	0.893 ± 0.049	0.90 ± 0.047	0.891 ± 0.051	0.891 ± 0.052

Table 10: GMM mean and std of the 10-fold cross-validation classification.

In Figure 24 we can see the confusion matrix for the manual labeling. While the shaky category is better recognized with the GMM, the rest of the categories are worse classified. Specially, it is interested is to observe that the 'smooth' category. With the GMM method, the confusion between 'smooth' and 'dynamic' is lower than with KNN method, but the confusion between smooth and steady is increased.

[10] 1, 2, 3, 4 and 5.
[11] full, spherical, tied, diagonal.

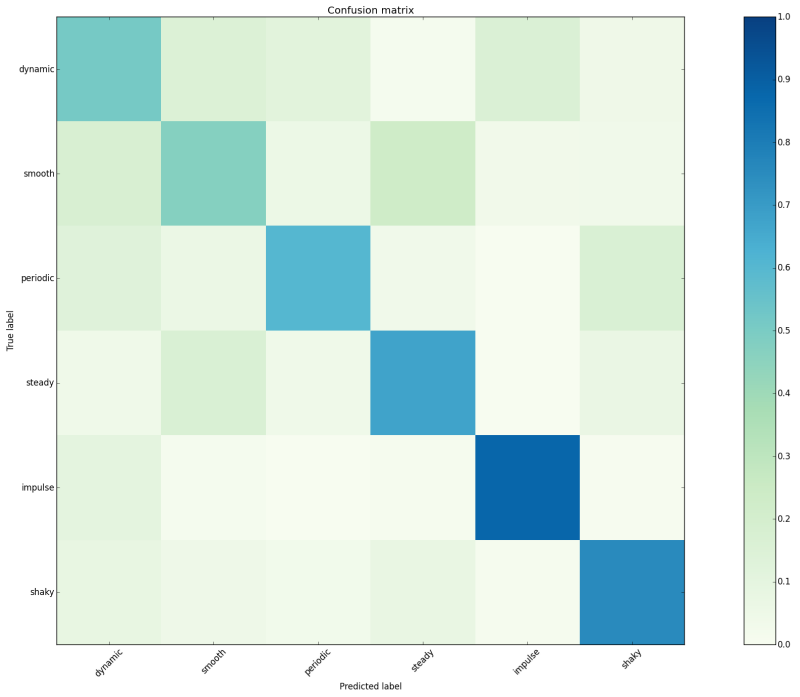


Figure 24: Confusion matrix for the GMM classification of manual primitives.

4.4 Appendix

4.4.1 Supervised methods definitions

- *Mutual info score (MI):* This is equal to the Kullback-Leibler divergence of the joint distribution with the product distribution of the marginals. It is independent of permutation of the class or cluster label values.

- *Adjusted mutual info score (AMI):* This is a modified version of MI where the information shared between classes is taken into account. The AMI returns a value of 1 when the two partitions are identical and 0.0 for random labeling.

- *Adjusted rand score:* it considers all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. Similarity score between -1.0 and 1.0. 0.0 value for random labeling and exactly 1.0 when the clusterings are identical.

- *V measure score or Normalized Mutual Information (NMI):* normalization of MI. Results are between 0 (no mutual information) and 1 (perfect correlation).

- *Completeness score:* if all the data points that are members of a given class are elements of the same cluster. Score between 0.0 and 1.0.

- *Homogeneity score:* if all of its clusters contain only data points which are members of a single class. Score between 0.0 and 1.0.

4.4.2 Unsupervised methods definitions

- *Dunn:* It is computed as the square root of the minimum distance between any two clusters (distance between the two closest points) divided by the square root of the maximum distance between any two points in the same cluster. It relates separation/overlapping with compactness. As it only measures the worse scenario it is highly susceptible to influence from noise, outliers, or two clusters that happen to be close together. The higher the value, the better the clustering will be.

- *Davies-Bouldin:* It compares each cluster to every other cluster. Similarity in measure for each pair of clusters as the sum of the average distances of each point in the two clusters to its respective center is divided by the distance between the two cluster centers. The maximum values of this function for each cluster are averaged. A lower score will be the result of less dispersion within clusters and more distance between clusters.

- *Silhouette:* It is the mean of every point score. Each individual value is define as the difference between the average distance between that point and every other point in its cluster (intra-cluster distance) and the minimum average distance between that point and the other points in each other cluster (inter-cluster distance). It measure relates separation to compactness by subtraction rather than division. As clustering improves, the score will approach 1.

- *Calinski-Harabasz (CH) or variance ratio criterion (VRC):* Well-defined clusters have a large compactness (inter-cluster distance or between-cluster variance BCSM) and a small separation (intra-cluster distance or within-cluster variance (WCSM). CH relates BCM to WCSM. The larger the VRC ratio, the better the data partition. A high score is deserved.

- *Sum-of-Squares (SS):* The measure reverses the relationship between separation and compactness. Since SS divides compactness by separation, a lower score indicates better clusterings.

- *Point biserial (PB):* It finds the difference between the average intra-cluster distance and the average inter-cluster distance. This measure is like Silhouette, except that it measures separation from all non-cluster sharing points, rather than only those of the closest cluster. A high score is deserved.

- *PBM:* It relates compactness with separation. While compactness is the sum of the distances between each point and its cluster centroid, separation is the maximum distance between any two cluster centers. Separation is normalized over a measure of dispersion, calculated as the sum of the distances between all points.

References

[1] Judith C. Brown. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[2] Einar Heiberg. Matlab parallelization toolkit. v. 1.20, Nov. 2003.

[3] Matthew D. Hoffman. Approximate Maximum A Posteriori inference with entropic priors. *The Computing Research Repository*, abs/1009.5761, 2010.

[4] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[5] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[6] Christian Schörkhuber, Anssi Klapuri, Nicki Holighaus, and Monika Dörfler. A Matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*, Jan 2014.

[7] Madhusudana Shashanka, Bhiksha Raj, and Paris Smaragdis. Probabilistic latent variable models as nonnegative factorizations. *Computational Intelligence and Neuroscience*, 2008.

[8] Madhusudana Shashanka, Bhiksha Raj, and Paris Smaragdis. Sparse overcomplete latent variable decomposition of counts data. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1313–1320. Curran Associates, Inc., 2008.

[9] Paris Smaragdis and Bhiksha Raj. Shift-invariant probabilistic latent component analysis. Technical report, MERL, 2007.

[10] Gino Angelo Velasco, Nicki Holighaus, Monika Dörfler, and Thomas Grill. Constructing an invertible constant-Q transform with non-stationary Gabor frames. In *Proceedings of Int.l Digital Audio Effects Conference (DAFX)*, pages 93–99, Paris, 2011.

[11] Chin J. Lin., Projected gradient methods for nonnegative matrix factorization. Neural Computation, 19(10): 2756–2779, 2007.

[12] Arthur, D. and Vassilvitskii, S., K-means++: the advantages of careful seeding, Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035, 2007

[13] Reynolds, D. A., Gaussian Mixture Models, Encyclopedia of Biometric Recognition, Springer, Journal Article, February 2008.

[14] F. Bevilacqua, N. Schnell, N. Rasamimanana, J. Bloit, E. Flety, B. Caramiaux, J. Françoise, E. Boyer. De-MO: designing action-sound relationships with the MO interfaces. In CHI '13 Extended Abstracts on Human Factors in Computing Systems (CHI EA '13). ACM, New York, NY, USA, 2907-2910, 2013.

[15] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755): 788-791, 1999.

[16] Christopher Torrence and Gilbert P Compo. A practical guide to wavelet analysis. Bulletin of the American Meteorological society, 79(1):6178, 1998.