Deliverable D4.3

First prototype tool for the automatic semantic description of music pieces

Grant agreement nr	688382
Project full title	Audio Commons: An Ecosystem for Creative Reuse of Audio Content
Project acronym	AudioCommons
Project duration	36 Months (February 2016 - January 2019)
Work package	WP4
Due date	30 April 2017 (M15)
Submission date	30 April 2017 (M15)
Deliverable type	Report ( ), Demonstrator (X), Other ( )
Report availability	Public (X), Confidential ( )
Task leader	QMUL
Authors	Johan Pauwels
Document status	Draft ( ), Final (X)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

Executive Summary

As part of the Audio Commons Ecosystem, a number of tools are provided for the automatic analysis of audio content without the need for human intervention. These tools are designed for extracting i) musical audio properties for music pieces and music samples, and ii) non-musical audio properties for any kind of sounds. Work-in-progress versions of these tools have been released in parallel for the first prototype in the Audio Commons Ecosystem.

This document presents a first overview of the tools that can be used to semantically annotate music pieces. We specifically discuss those descriptors that cannot be applied to samples, because they require more temporal context or musical complexity. As such, they complement the tools presented in D4.2 and D5.2. Examples of descriptors for music pieces are genre or mood tags . One application would be to run these tools as a service on a webserver to automatically annotate all audio in the AC ecosystem. The resulting labels can subsequently be used to match audio to user queries, which are generally formulated on a high semantic level.

In order to ease parallel development and modularity, the one “tool” presented in this deliverable actually consists of a collection of smaller tools. Most of them are either part of the Vamp framework, initiated at QMUL, or the Essentia framework, initiated at UPF-MTG. Both frameworks are mature and permit the reuse of basic components. Another advantage is that new tools can be developed in isolation from the network technologies that are used to deploy these tools in the AudioCommons ecosystem. As such, annotation tool authors don’t need to know about web technologies and vice versa, making the separation of responsibilities easy. This also means that the resulting tools can be used in other scenarios, e.g. on personal computers.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 688382

1 Description of the demonstrator/software

This demonstrator consists of a number of tools that take an audio file as input and return a high-level semantic labels (incl. possibly time-varying annotations). The different tools produce different types of semantic labels, such as instrumentation, musical key, etc., and work independently from each other.

Informed by the survey on user requirements[1] in D2.1, a first draft of the Audio Commons ontology is currently under development. As part of this ontology, a sound schema[2] is proposed to serve as the minimum requirement for a provider to support within the Audio Commons Ecosystem. This draft includes a number of high-level semantic musical properties that will need to be extracted automatically. These properties are listed in the table below. The tools that are currently being developed are either Vamp plugins or part of the Essentia framework. Both come with separate installation instructions, which comprise the rest of this document. The table below lists which property is implemented in each framework. Further details can be found in their respective section. At this point, not all musical properties have an associated tool that can generate them yet.

Musical Property	**Example**	**Feature Extraction Framework**
Genre	Rock, classical, jazz, hip-hop	Essentia (E)
Instruments	Piano, guitar, voice	Vamp (E)
Mood	Happy, dreamy, sad, aggressive	Essentia (E)
Themes	Advertising, fashion, film & tv	
Tonality	A major, Bb minor	Vamp (S), Essentia (S)
Tempo	126 BPM, adagio	Vamp (S), Essentia (S)
Chord	Emaj7, Ddim	Vamp (S), Essentia (S)

Table 1: Musical properties per framework

Some of the implementations are in a finalised state and are expected to be integrated into the Audio Commons API (WP2) without significant difficulties, whereas others are still evolving and will require more effort for the integration. In the table, this is indicated by the suffixes (S) for stable software and (E) for experimental. The sections of the documentation below also reflect this. Note that the quality of the implementation does not necessarily reflect the quality of the underlying algorithm. Stable software can in many cases still be improved in terms of the output it generates.

Naturally, the Audio Commons ontology contains more properties than the one listed above. On the one hand, some properties need to be derived from metadata, because they are not contained in the audio (e.g. title, artist). On the other hand, audio derived features also include more technical qualities

[1]Audio Commons Initiative Survey on Creative Interaction with Audio Content including responses by over 660 audio professionals connected to the Audio Commons Ecosystem via our industry partners: https://docs.google.com/forms/d/1c3iJhZcAPCqfmlUekj3aHAYPxeY6q-KW2bxZa8tUqNA/edit#responses

[2] https://docs.google.com/spreadsheets/d/19GlELgsx5AReb6d-8_6wfsVdSfUwCDqgr8k2AXLVP48

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

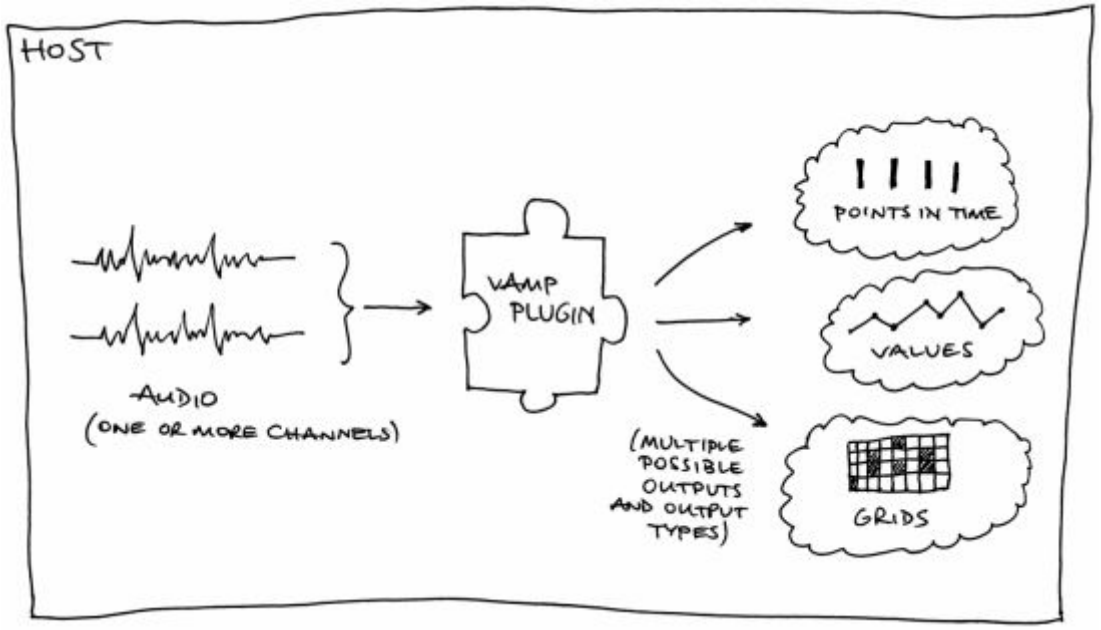
related to the encoding of audio files (e.g. bitrate, number of channels) and low-level features (see D4.1 and D4.2 for an overview of both). Finally, novel timbral descriptors are explained in D5.2.

1.1 Production-ready software

1.1.1 Vamp framework

Introduction

Vamp is an audio processing plugin system for plugins that extract descriptive information from audio data — typically referred to as *audio analysis plugins* or *audio feature extraction plugins*. Just like an audio effects plugin (such as a VST), a Vamp plugin is a binary module that can be loaded up by a host application and fed audio data. However, unlike an effects plugin, a Vamp plugin generates not processed audio output, but symbolic information related to the content of audio files. Typical things that a Vamp plugin might calculate include the locations of moments such as note onset times, visualisable representations of the audio such as spectrograms, or time-varying one dimensional data (i.e. time series) such as power or fundamental frequency.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

Installation instructions

A Vamp plugin set consists of a single dynamic library file with DLL, .dylib, or .so extension (depending on your platform) plus optionally a category file with .cat extension and an RDF description file with .ttl or .n3 extension[3].

To install a plugin set, copy the plugin's library file and any supplied category and/or RDF files into your system or personal Vamp plugin location.

The plugin file extension and the location to be used depends on your platform:

	File extension	System plugin folder	Personal plugin folder
Linux or other Unix	.so	/usr/local/lib/vamp	$HOME/vamp
OS/X	.dylib	/Library/Audio/Plug-Ins/Vamp	$HOME/Library/Audio/Plug-Ins/Vamp
Windows (32-bit)	.dll	C:\Program Files\Vamp Plugins	
Windows (64-bit)	.dll	C:\Program Files (x86)\Vamp Plugins	

You can alternatively set the VAMP_PATH environment variable to list the locations a host should look in for Vamp plugins.

VAMP_PATH should contain a semicolon-separated (on Windows) or colon-separated (OS/X, Linux) list of paths. If it is set, it will override the standard locations listed above.

Available plugins

A number of Vamp plugins that provide descriptors listed in Table 1 are listed in the table below along with their download locations.

Name	AC Descriptor(s)	Website	Source code available
BeatRoot	Tempo	https://code.soundsoftware.ac.uk/projects/beatroot-vamp	Y
Chordino	Chords	http://www.isophonics.net/nnls-chroma	Y

[3] This additional information provides mechanisms for finding plugins and categorising them in end user tools.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

INESC Porto Beat Tracker	Tempo	http://smc.inescporto.pt/technologies/ibt/	Y
Queen Mary plugin set	Tempo, Tonality	http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html	Y
Vamp Aubio plugins	Tempo	http://aubio.org/vamp-aubio-plugins/	Y

Vamp Plugin Hosts

A software tool that is capable of loading and executing a plugin is called a host. Once plugins have been installed, they can be used by running one of the available hosts. A number of host applications exist, each with their own advantages. They are listed in the table below. For large-scale deployment of a plugin as part of an API, either Sonic Annotator (a standalone executable) or VamPy Host (a Python package) would be the most suitable.

Name	**Description**	**Website**
Sonic Visualiser	Visual analysis tool	http://www.sonicvisualiser.org/
Audacity	Audio editor	http://www.audacityteam.org/
Ardour	Digital audio workstation	https://ardour.org/
Sonic Annotator	Batch extraction	http://www.vamp-plugins.org/sonic-annotator/
VamPy Host	Python integration	https://code.soundsoftware.ac.uk/projects/vampy-host

VamPy plugin for rapid prototyping

As part of the Vamp framework, a Python extension named VamPy exists which allows to rapidly create Vamp plugin prototypes in Python. It needs to be installed like any other Vamp plugin by downloading it from http://vamp-plugins.org/vampy.html. Then one can write plugins in Python simply by saving the source files to the Vamp plugin directory. A tutorial that includes a Python template can be found at http://isophonics.net/content/getting-started-vampy.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1.1.2 Essentia framework

Essentia is a open-source C++ library for audio analysis and audio-based music information retrieval. It contains an extensive collection of algorithms including audio input/output functionality, standard digital signal processing blocks, statistical characterization of data, and a large set of spectral, temporal, tonal and high-level music descriptors.

Essentia is cross-platform and it is designed with a focus on optimization in terms of robustness, computational speed and low memory usage, which makes it effective for many industrial applications. The library is also wrapped in Python and includes a number of command-line tools and third-party extensions, which facilitate its use for fast prototyping and allow setting up research experiments very rapidly.

Instructions to install Essentia on a multitude of platforms can be found at http://essentia.upf.edu/documentation/installing.html. Precompiled binaries can be found on http://essentia.upf.edu/documentation/extractors/. The Essentia framework contains a multitude of descriptors, all of which are listed at http://essentia.upf.edu/documentation/algorithms_reference.html. The ones that are particularly relevant for this deliverable are displayed in the table below

Essentia name	**AC Descriptor**	**Website**
Key	Tonality	http://essentia.upf.edu/documentation/reference/std_Key.html
KeyExtractor	Tonality	http://essentia.upf.edu/documentation/reference/std_KeyExtractor.html
PercivalBpmEstimator	Tempo	http://essentia.upf.edu/documentation/reference/std_PercivalBpmEstimator.html
RhythmExtractor	Tempo	http://essentia.upf.edu/documentation/reference/std_RhythmExtractor.html
RhythmExtractor2013	Tempo	http://essentia.upf.edu/documentation/reference/std_RhythmExtractor2013.html
ChordDetection	Chord	http://essentia.upf.edu/documentation/reference/std_ChordsDetection.html
ChordsDetectionBeats	Chord	http://essentia.upf.edu/documentation/reference/std_ChordsDetectionBeats.html

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1.2 Experimental software

1.2.1 VamPy framework

Instrument identification plugin

An instrument identification plugin prototype has been developed that can distinguish between 24 instrument categories. These classes are: Shaker, Electronic Beats, Drum Kit, Synthesizer, Female Voice, Male Voice, Violin, Flute, Harpsichord, Electric Guitar, Clarinet, Choir, Organ, Acoustic Guitar, Viola, French Horn, Piano, Cello, Harp, Conga, Synthetic Bass, Electric Piano, Acoustic Bass, Electric Bass. The plugin uses a neural network that consists of two convolutional layers and three densely connected layers. It is trained on mel-frequency spectrograms with 128 bands calculated from the first five seconds of 2400 loops (100 per class) of the Apple Logic Pro X library. More details of the network architecture can be found in Appendix 1.

Installation and usage instructions

Because the drawback of a VamPy plugin is that it is not self contained, but relies on a number of external packages, it can be challenging to deploy. To make matters easier, a Docker image[4] has been created that consists of a minimal operating system with all the necessary dependencies. The Docker configuration file and Python source code to recreate the image are available at https://github.com/AudioCommons/vampy-instrument-identification. However, the easiest way to run the image is to download it from DockerHub, as per the instructions below.

First, the Docker runtime environment needs to be installed, if it wasn't already. Installation instructions can be found at https://docs.docker.com/engine/installation/. Then start Docker such that you get the command line interface. In order to run the instrument identification plugin from a Docker container, the following command needs to be used: `docker run -it --rm --volume <path to local audio dir>:/srv jpauwels/vampy-instrument-identification`

Explanation: `docker run -it` starts an interactive terminal in a container created from the image jpauwels/vampy-instrument-identification which will be pulled in from DockerHub if you don't have it locally. --rm removes the container after you halt it (type exit on the command line), because you don't want to keep changes to the image anyway and this way you will be forced to create a new container from the the image every time you start it (such that the image can be synchronised with new updates pushed to DockerHub). Because the container has an isolated filesystem, you'll want to map some directory with audio files to the path /srv in the container with the --volume command.

Two Vamp transformation definitions (predominant-instrument.n3 and instrument-probabilities.n3) have been added to the home dir, so inside the container you can execute `sonic-annotator -t ~/predominant-instrument.n3 -w <writer> <audiofile>`. The first transform returns the label of the predominant instrument found in the file along with its probability and the second returns the probabilities of all 24 instruments.

[4] A Docker image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it. See: https://www.docker.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

For automated use when you don't want an interactive shell, you can combine both commands as `docker run --rm --volume <path to local audio dir>:/srv jpauwels/vampy-instrument-identification /usr/local/bin/sonic-annotator/sonic-annotator -t /root/predominant-instrument.n3 -w <writer> <audiofile relative to local dir>`.

1.2.1 ProbCog toolkit

A possible route for the integration of ontologies into semantic annotation tools has been explored by examining Markov and Bayesian Logic Networks (MLN/BLN). The ProbCog tool has been used for this purpose. Some example configurations for various chord estimation approaches are made available at https://github.com/jpauwels/MLN-vs-BLN.

Because the ProbCog toolkit is non-trivial to install, these examples are also packaged together with the toolkit in a Docker container. The resulting image has been uploaded to DockerHub, such that you can retrieve it from there using the tag `jpauwels/probcog:mln-vs-bln`.

Because the software includes a GUI, an X-server is required and starting Docker is slightly more complicated. On macOS, you need XQuartz with "Allow connections from network clients" enabled under the Security tab in Preferences. Then launch the image as follows:

```
ip=$(ifconfig en0 | grep inet | awk '$1=="inet" {print $2}')
xhost + $ip
docker run -e DISPLAY=$ip:0 -v /tmp/.X11-unix:/tmp/.X11-unix --rm -it jpauwels/probcog:mln-vs-bln
```

The configuration files provided in the repository are available inside the home directory of the image. If you want to run the toolkit on its own, you can use the tag `jpauwels/probcog:base`. This software is currently not expected to be integrated in the Audio Commons API, because of its high computational requirements, but serves as a demonstrator for the underlying technology.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

2 Conclusion

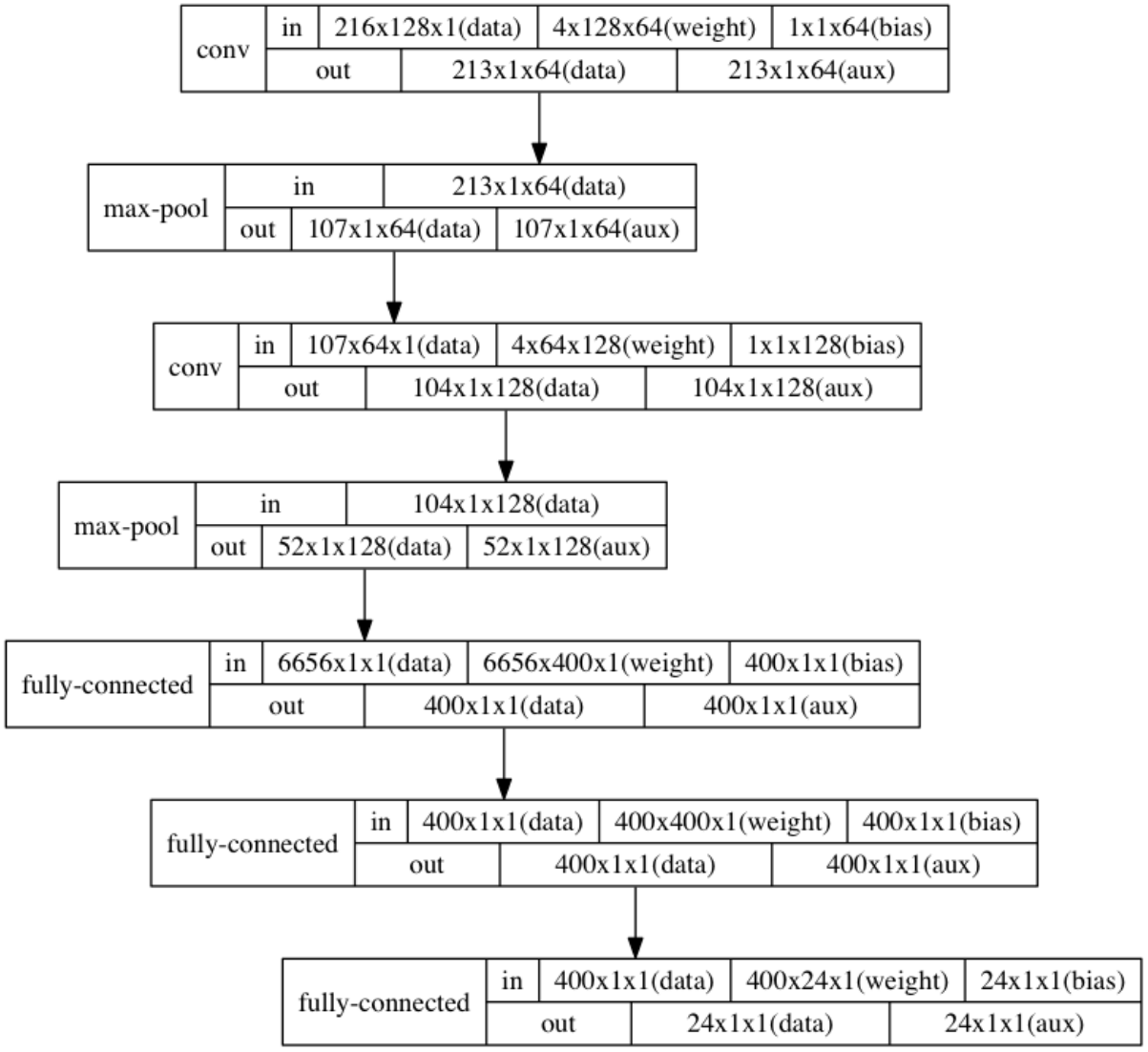
In this document, an overview was given of the software that is currently available for the automatic annotation of music pieces in the Audio Commons ecosystem. These descriptors can be generated by a collection of smaller tools, either available as Vamp plugins or as part of Essentia. This report is intended to inform WP2 about the software available for integration into an automatic annotation API.

The information contained herein is just a snapshot of the current situation, which is expected to be improved constantly over the remainder of the project. Future efforts will be directed towards developing new tools for those descriptors that have been identified as useful by WP2, but currently have no associated software, notable the description of “themes”. Also existing tools will constantly be upgraded when it comes to the quality of the generated labels and to the quality of the implementations themselves. Moreover, the separate tools are expected to be integrated into one, for easier rollout. Finally, a close integration with the tools described in D4.2 and D5.2 needs to be achieved, such that the generated descriptors complement each other.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 688382

APPENDIX 1: Neural network architecture of the VamPy instrument identification plugin

The exact parameters of the neural network as used in the VamPy instrument identification plugin can be found in the diagram below. All activations functions are Leaky ReLU's[5] with gradient 0.33, except for the last layer, where they are softmaxes.



[5] Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models, Proceedings of the International Conference on Machine Learning

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382