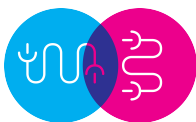


Project Acronym: **MusicBricks**  
Project Full Title: **Musical Building Blocks for Digital Makers and Content Creators**  
Grant Agreement: **N°644871**  
Project Duration: **18 months (Jan. 2015 - Dec. 2016)**

## D3.2 First release of API

Deliverable Status: **Final**  
File Name: **MusicBricks\_D3.2.pdf**  
Due Date: **30 June 2015 (M6)**  
Submission Date: **30 June 2015 (M6)**  
Dissemination Level: **Public**  
Task Leader: **Universitat Pompeu Fabra**  
Authors: **Jordi Janer (UPF), Thomas Lidy (TUW), Jakob Abeßer (IDMT), Sascha Grollmisch (IDMT), Alexander Schindler (TUW), Frederic Font (UPF).**





The MusicBricks project consortium is composed of:

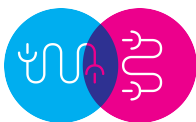
<b>SO</b>	Sigma Orionis	France
<b>STROMATOLITE</b>	Stromatolite Ltd	United Kingdom
<b>IRCAM</b>	Institut de Recherche et de Coordination Acoustique Musique	France
<b>UPF</b>	Universitat Pompeu Fabra	Spain
<b>Fraunhofer</b>	Fraunhofer-Gesellschaft zur Foerderung der Angewandten Forschung E.V	Germany
<b>TU WIEN</b>	Technische Universitaet Wien	Austria

### **Disclaimer**

*All intellectual property rights are owned by the MusicBricks consortium members and are protected by the applicable laws. Except where otherwise specified, all document contents are: “©MusicBricks Project - All rights reserved”. Reproduction is not authorised without prior written agreement.*

*The commercial use of any information contained in this document may require a license from the owner of that information.*

*All MusicBricks consortium members are also committed to publish accurate and up to date information and take the greatest care to do so. However, the MusicBricks consortium members cannot accept liability for any inaccuracies or omissions nor do they accept liability for any direct, indirect, special, consequential or other losses or damages of any kind arising out of the use of this information.*



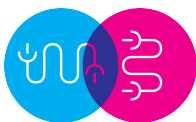
## Revision Control

Version	Author	Date	Status
0.1	Jordi Janer (UPF)	April 23, 2015	Initial Draft
0.2	Thomas Lidy (TUW)	May 8, 2015	Contributions by TUWIEN
0.3	Jakob Abeßer (IDMT)	May 21, 2015	Contributions by IDMT
0.4	Jordi Janer (UPF)	June 25, 2015	Quality Check
0.5	Marta Arniani (SIGMA)	June 29, 2015	Quality Check
0.6	Jordi Janer (UPF), Cyril Laurier (STROMATOLITE)	June 30, 2015	Final Draft reviewed
1.0	Marta Arniani (SIGMA)	June 30, 2015	Final review and Submission to the EC



## Table of Contents

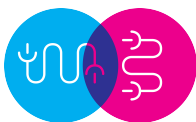
Executive summary .....	5
<b>1. Introduction .....</b>	<b>6</b>
1.1 Context.....	6
1.2 Objectives .....	6
<b>2. Preparation of a #MusicBricks Portfolio.....</b>	<b>7</b>
<b>3. Workshop Music Processing: selected #MusicBricks APIs .....</b>	<b>8</b>
3.1 Introduction.....	8
3.2 Rhythm and Timbre Analysis from Music - TU Wien .....	9
3.3 Melody Analysis - UPF .....	9
3.4 Real-time pitch detection and applications - Fraunhofer-IDMT .....	9
<b>4. First Release of #MusicBricks APIs .....</b>	<b>10</b>
4.1 Introduction.....	10
4.2 Transcriber - Melody & Bass Transcription, Beat & Key & Tempo Detection API (by Fraunhofer IDMT) .....	11
4.3 Real-time Pitch Detection - Monophonic & Polyphonic API (by Fraunhofer IDMT) .....	12
4.4 Rhythmic and Timbre - RP extract: Audio Feature Extraction (by TU Wien).....	13
4.5 Search by Sound: Music Similarity Retrieval API (by TU Wien / Spectralmind).....	14
4.6 Freesound API (MTG-UPF).....	15
4.7 Melody Extraction API (MTG-UPF).....	17
4.8 Onset Description API (MTG-UPF) .....	18
<b>5. Download and installation.....</b>	<b>19</b>
<b>6. Conclusions .....</b>	<b>20</b>



## Executive summary

The present document is a deliverable of the MusicBricks project, funded by the European Commission's Directorate-General for Communications Networks, Content & Technology (DG CONNECT), under its Horizon 2020 research and innovation programme.

This document introduces the first release of the #MusicBricks technology portfolio provided by the consortium research partners. In particular, in Work Package 3 we deployed these technologies in form of APIs for an easy integration by creative users. During the reported period since the previous deliverable (D3.1), these technologies have been presented in two creative Testbed events and successfully integrated in several projects by the creative users community. The first release of the APIs includes a comprehensive yet not exhaustive portfolio of music algorithms: Melody Extraction, rhythm and Timbre analysis, Gesture Sensors, Search Similar Sound, Freesound, Real-time Pitch Detection, Music Transcriber, and Real-time Onset Description. This portfolio will be refined and extended for the Final Release of the APIs, which shall be delivered on Month 12.



## 1. Introduction

This document is elaborated upon the previous deliverable D3.1. It provides a thorough description of the technologies included in the *First Version of the APIs*. It includes details about the current technical status of each technology in terms of readiness, documentation, installation and licensing options.

### 1.1 Context

During the reported period (April to June 2015), technology partners have jointly collaborated in structuring a portfolio of #MusicBricks technologies. These #MusicBricks were used already by hackers in two events in the scope of the project: MusicTechFest (May, Umea) and MusicHackDay (June, Barcelona). Additionally, the Consortium organized a technical Workshop about “Music Processing”, which was collocated with the MusicHackDay in Barcelona on June 17th. In the workshop, participants were trained by researchers and technologists members of the consortium. A big effort of this period is reflected in the way the technical information on #MusicBricks was made available to participants through dedicated websites.

The release of the First Version of the APIs (June 2015) will be followed by a second and Final Version of the APIs (December 2015). Intermediate improvements on the technologies APIs will be added in the next 6-months.

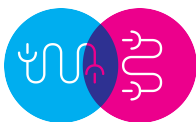
### 1.2 Objectives

As specified in the Description of Work document, the goal of this deliverable was to ensure that the provided technologies can be used in the Creative Testbeds (*First release for use in Creative Testbeds*). These objectives are principally framed in the Task 3.1 Wrapping existing Tools and technologies together. And partially in the Task3.2 Creating Specific tool Combinations for advanced features.

In the past two Creative Testbed events (MusicTechFest and MusicHackDay), participants have extensively used the provided technologies in their projects. It validates the usability of the First Version of the APIs, and therefore the results of the work carried out during the first 6 months of the #MusicBricks project.

The objectives of the document are addressed in the next sections:

1. Preparation of MusicBricks portfolio
  - a. working sessions before and during Umea event
2. Hands-on Presentation of MusicBricks portfolio
  - a. Tutorial and workshop on Music Processing sessions
3. Description of the MusicBricks portfolio
  - a. API reference, current status and licensing conditions
4. Documentation and installation
  - a. Comparative table and Links



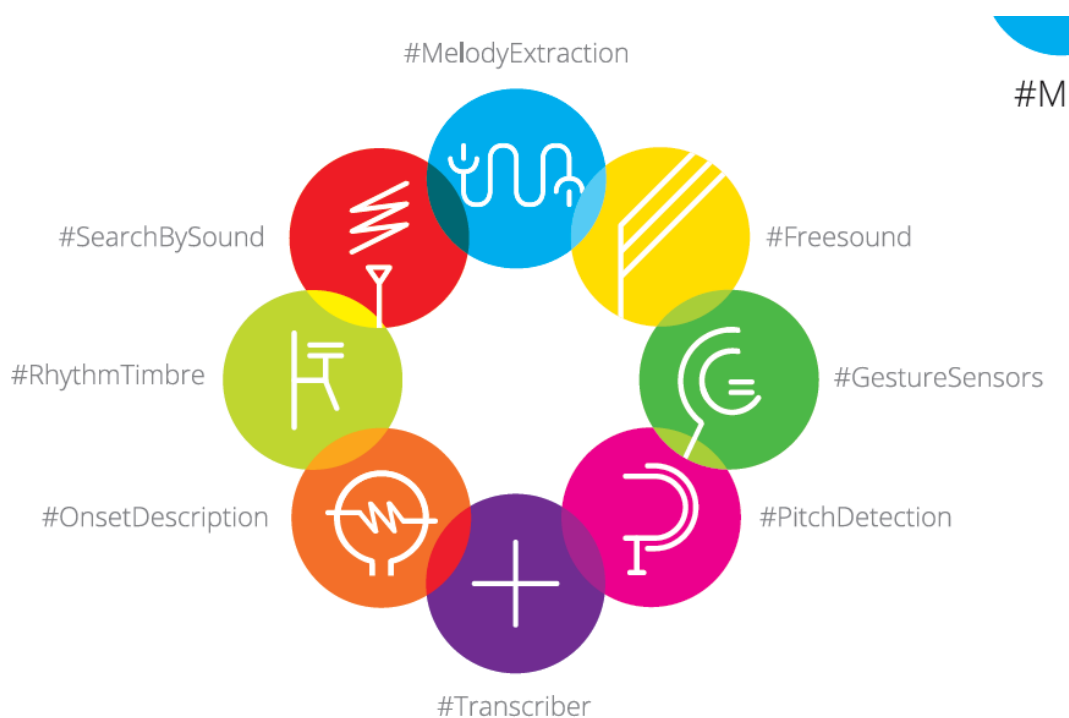
## 2. Preparation of a #MusicBricks Portfolio

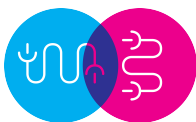
We carried out working sessions with WP3 and WP4 partners before and during 4 days in the scope of the MusicTechFest (Umea, Sweden) from May 29th to June 1st 2015. These working sessions allowed the technology partners (TU Wien, Fraunhofer IDMT, IRCAM and MTG-UPF) to refine and agree on the contents of the #MusicBricks technology portfolio. We identified components that can be more relevant to creative users (hackers), in terms technical and ease of use. For example, for certain project having a real-time implementation is a clear demand by users.

One very important aspect that we addressed was to avoid overlap between the technologies of different partners. This is especially crucial when we present #MusicBricks portfolio to general audiences, who are not aware of the research background of the Consortium. In particular TU Wien, Fraunhofer IDMT and MTG-UPF all are experts in technologies related to “Music Information Retrieval” that are integrated as part of the #MusicBricks portfolio.

In the process to build a comprehensive yet non-overlapping portfolio, we firstly listed all individual technologies according its functionality. Secondly each partner selected the tools that were more appropriate in terms of uniqueness, complementariness, novelty or performance.

Next figure shows the portfolio of #MusicBricks, as released in the First Version of the APIs.





### 3. Workshop Music Processing: selected #MusicBricks APIs

#### 3.1 Introduction

As part of the activities for preparing the *First Version of the APIs* in WP3, the technology partners TU Wien, MTG-UPF, Fraunhofer IDMT and IRCAM have participated in the MusicBricks Workshop on June 17th, collocated with the MusicHackDay in Barcelona that takes place the day after (June 18th).

This workshop is an occasion to present the MusicBricks API with more insight to the hacker community, while providing the right environment for a hands-on session with participants. Participants of the workshop are part of the MusicHackDay.

The content is structured in four sessions under the name of “Music Processing”. Each session addresses one music processing topic, giving hints about the use of #MusicBricks technologies in practical applications.

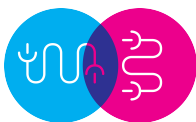
1. **Melody Analysis** (Nadine Kroher): this session presents a number of pitch tracking and melody transcription algorithms implemented in the *Essentia* library. We will demonstrate their differences in performance and give suggestions on which algorithm to use with respect to instrumentation, genre and task on a number of practical examples.
  - a. Applications: visualization of predominant melody, pitch tracking, tuning rating, source separation
  - b. Required installation: *Python (2.7)*, *iPython notebook*, *Essentia library*, *Melody extractor binaries (precompiled Linux/OSX)* and *Python example code* .
2. **Rhythm and Timbre Analysis** (Thomas Lidy): this session gives an overview and introduction on how to extract rhythmic and timbral information out of music, showing how specifically Rhythm Patterns and Statistical Spectrum Descriptors manage to capture a song’s rhythmic and timbral content for real-world applications such as music search by audio similarity.
  - a. Applications: get rhythm and timbre features from the audio to compute similarity between songs, find music with similar rhythm or timbre, detect the genre of a song, generate playlists of music of a certain style, recommend music, etc.
  - b. Required installation: *Python (2.7)*, *iPython notebook* and *several packages*, *RP\_extract*
3. **Real-time Pitch Detection** (Jakob Abesser): This session focuses on the real-time automatic transcription of monophonic and polyphonic audio input. Also we will discuss several offline analysis tools such as melody and bass line transcription as well as the estimation of key, tempo, and beat positions. We show also how music transcription algorithms are integrated in music learning applications.
  - a. Applications: Application of pitch detection for real-time performance assessment in music games, Visualization of different analysis results such as notes and beats in *SonicVisualiser*
  - b. Required installation: *IDMT tools (MusicBricksTranscriber, Pitch Detection Library)*, *Python (2.7)*, *cmake*, *(SonicVisualiser)*
4. **Gesture Sensors for Music Performance** (Emmanuel Flétly & Frédéric Bevilacqua): in this session we will provide the wireless motion sensor R-IoT that allows for low latency 9-axis data streaming (accelerometers, gyroscope, magnetometer), along with real-time analysis tools (Max/MSP).
  - a. Applications: new musical instruments, motion capture, gesture-controlled music systems, etc.
  - b. Required installation: *MaxMSP with the MuBu &co library*, *collection of gesture analysis patches*

Information for participants was available online in the URL:

<http://musichackday.upf.edu/mhd/2015/?session=hands-on-workshops>

Next we include a more detailed description of the session related to the API technologies part of the Work Package 3 by MTG-UPF, TU Wien, and Fraunhofer IDMT.





### 3.2 Rhythm and Timbre Analysis from Music - TU Wien

TU Wien members gave an overview and introduction on how to extract rhythmic and timbral information out of music to a general audience (target group: people with technical/engineering but not scientific background). The overview covered how audio is stored and processed in computers and how we use spectral representations to process it further and extract meaningful information (features) out of it.

We then covered how specifically Rhythm Patterns and Statistical Spectrum Descriptors and Rhythm Histograms manage to capture a song's rhythmic and timbral content and what is the purpose of it by giving examples of real-world applications such as music search by acoustic similarity, generating playlist of a particular mood using similar audio or using pre-trained models to detect music metadata such as music genre which can be used to query online services.

These algorithms are included in the free and open source RP\_extract library, which is available for Python, Matlab and Java. In a hands-on session we gave an introduction to using this library, using the interactive iPython Notebook on screen, where coding can be performed and followed step-by-step in real-time, and graphics visualizing the outcome of the feature analysis can be shown inline.

We have also briefly introduced and showed how to access the SMINT API which provides a similar functionality as a ready-available Web service, containing also an already pre-analyzed library of 50,000 songs from freemusicarchive.org.

In this session we also both suggested and discussed with the people ideas on what to build on top of this library.

### 3.3 Melody Analysis - UPF

UPF-MTG members presented its advances in algorithms for melody extraction both for monophonic and polyphonic inputs. The presentation covered a divulgative overview of the how melody extraction algorithms work, as well as giving insights to obtain best results in different singing styles and music genres.

These algorithms are already integrated in the Essentia Library, and are part of the “#MusicBricks Melody Extraction API”.

We highlight in section [3.7.4](#) below the relevant aspects for the appropriateness of having this technology presented as part of the workshop, and put available to hackers as part of the MusicHackDay.

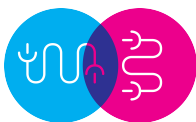
### 3.4 Real-time pitch detection and applications - Fraunhofer-IDMT

Fraunhofer IDMT members gave first an introduction on music analysis technologies with special focus on the real-time predominant melody and automatic transcription of the bass line from polyphonic audio signals.

Then, the application of music transcription algorithms for music learning and music performance assessment has been discussed as examples of how to use the available technology.

Additionally, we presented how to extract additional parameters such as key, tempo, and the beat grid and provide details about the different output formats MIDI, Music XML, and plain XML to be used for instance in Python for further processing or Sonic Visualizer for visualization purpose.

Finally, using a sample application, we demonstrated the algorithms for real-time monophonic and polyphonic pitch detection, which amongst others can be integrated into mobile applications.



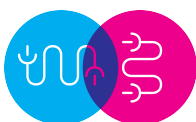
## 4. First Release of #MusicBricks APIs

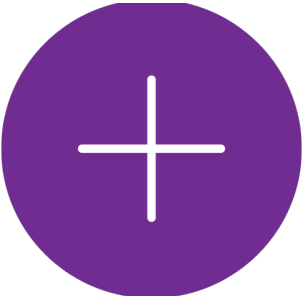

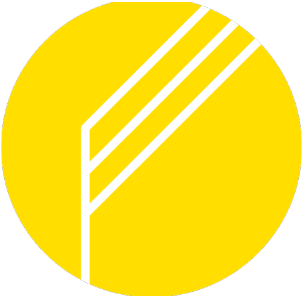
### 4.1 Introduction

In this section we describe the technologies and APIs provided by each research partner as part of the *First Version of the API*.

Lists of APIs:

	<p><b>Melody extraction</b></p> <p>This module includes a number of pitch tracking and melody transcription algorithms implemented in the Essentia library. We will demonstrate their differences in performance and give suggestions on which algorithm to use with respect to instrumentation, genre and task on a number of practical examples. Applications include visualization of predominant melody, pitch tracking, tuning rating, source separation.</p>
	<p><b>Real-time Onset description</b></p> <p>This module allows to detect onsets in real-time and provide a number of audio descriptors. It is part of <b>essentiaRT<sup>~</sup></b>, a real-time subset of <b>Essentia</b> (MTG's open-source C++ library for audio analysis) implemented as an <b>external for Pd and Max/MSP</b>. It provides a number of features to slice and provide on-the-fly descriptors for classification of audio in real-time. A number of extractors analyse instantaneous features like the onset strength, the spectral centroid and the MFCC's over a fixed-size window of 2048 points, after an onset is reported. Furthermore, <i>essentiaRT<sup>~</sup></i> is able to perform estimations on larger time-frames of user-defined lengths, and to report finer descriptions in terms of noisiness, f0, temporal centroid and loudness.</p>
	<p><b>Rhythm and Timbre Analysis</b></p> <p>This is a library that processes audio data as input and analyzes the spectral rhythmic and timbral information in the audio to describe its acoustic content. It captures rhythmic and timbral features which can be stored or directly processed to compute acoustic similarity between two audio segments, <b>find similar sounding songs (or song segments)</b>, create <b>playlists of music of a certain style</b>, <b>detect the genre</b> of a song, make <b>music recommendations</b> and much more. Depending on the needs, a range of audio features is available: Rhythm Patterns, Rhythm Histograms (i.e. a rough BPM peak histogram), Spectrum Descriptors and more. The library is available for Python, Matlab and Java.</p>
	<p><b>Search by Sound Music Similarity</b></p> <p>The Search by Sound online system is based on the Rhythm and Timbre Analysis (see above) and provides a system which can be used via a REST Web API (called SMINT API) to upload, find and match acoustically similar songs in terms of rhythm and timbre – without the need to install any prerequisite or run the analysis on your own. It can be used with your own custom music dataset or the readily available content from <a href="http://freemusicarchive.org">freemusicarchive.org</a> that was already pre-analyzed by rhythm and timbre, to find music matching a particular rhythm or timbre from that archive.</p>



	<p><b>MusicBricks Transcriber</b></p> <p>The MusicBricksTranscriber (Melody &amp; Bass Transcription + Beat &amp; Key &amp; Tempo Estimation) provided by Fraunhofer IDMT is an executable that allows to transcribe the main melody and bass line of a given audio file. Also, the beat times, the key, and the average tempo are estimated. The results can be provided as MIDI, MusicXML, or plain XML files. In addition, a Python wrapper is included to further process the analysis results.</p>
	<p><b>Real-time Pitch Detection</b></p> <p>The real-time pitch detection allows to estimate the predominant melody notes (monophonic) or multiple notes (polyphonic) from a consecutive audio sample blocks. This allows to transcribe the currently played / sung note pitches from a recorded instrument / vocal performance. The monophonic version also estimates the exact fundamental frequency values. Typical applications are music games and music learning applications. Fraunhofer IDMT provides a C++ library as well as sample projects that show how to include the functionality.</p>
	<p><b>Freesound API</b></p> <p>Freesound (<a href="http://www.freesound.org">www.freesound.org</a>) is a state-of-the-art online collaborative audio database that contains over 200K Creative Commons licensed sound samples. All these sounds are annotated with user-provided free-form tags and textual descriptions that enable text-based retrieval. Content-based audio features are also extracted from sound samples to provide sound similarity search. Users can browse, search, and retrieve information about the sounds, can find similar sounds to a given target (based on content analysis) and retrieve automatically extracted features from audio files; as well as perform advanced queries combining content analysis features and other metadata (tags, etc...). The Freesound API will provide access to a RESTful API with API Clients in Python, Javascript, Objective-C.</p>

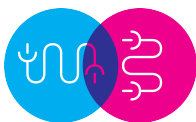
Next sections provide detailed information of the different technologies provided by research partners.

## 4.2 Transcriber - Melody & Bass Transcription, Beat & Key & Tempo Detection API (by Fraunhofer IDMT)

### 4.2.1 Description

The first binary executable allows performing different music analysis tasks for a given WAV or MP3 audio file of 15 minutes length maximum:

- **Bass Transcription**
  - This algorithm transcribes the bass line of a song as a sequence of note events
- **Melody Transcription**
  - This algorithm transcribes the predominant melody of a song as a sequence of note events
- **Key**
  - This algorithm estimates the most likely major or minor key of a given song
- **Beat Grid**



- This algorithm estimates the metric structure of a given song and returns the beat times and beat numbers
- **Tempo**
  - This algorithm estimates the average tempo of a song in beats per minute (bpm)

#### 4.2.2 Platforms

- Windows
- Mac OSX
- Linux

#### 4.2.3 Interface Description

The executable allows to select the **analysis tasks** to be performed using **commandline parameters**. The analysis results (transcription, beat grid, etc.) can be exported as

- **MIDI** (melody & bass transcription results)
- **MusicXML** (melody & bass transcription result)
- **XML** (easy-to-read format with all results)

for further use in other programs.

#### 4.2.4 Readiness level and IPR

- Stable and reliable. Tested in scientific and commercial settings throughout many years.
- Overview and documentation will be included
- Royalty-free, limited, non-transferable license

### 4.3 Real-time Pitch Detection - Monophonic & Polyphonic API (by Fraunhofer IDMT)

#### 4.3.1 Description

The realtime melody transcription technology will be provided as dynamic libraries (DLL / SO files) to be included into the user's programs.

#### 4.3.2 Platforms

- Windows, Mac OSX, Linux
  - Sample-Code in C++ will be provided
- iOS, Android
  - A sample application for both operating systems will be provided as demo

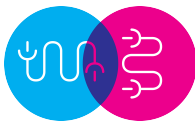
#### 4.3.3 Interface Description

The interface will provide methods to

- Switch between the pitch detection version (monophonic, polyphonic)
- Create / delete a pitch detection object
- Set reference frequencies for polyphonic pitch detection (to limit the search range)
- Get one / multiple detected fundamental frequencies for a given sample buffer

#### 4.3.4 3.3.4 Readiness level and IPR

- Stable and reliable. Tested in scientific and commercial settings throughout many years
- Overview, documentation, and sample application will be included
- Royalty-free, limited, non-transferable license



## 4.4 Rhythmic and Timbre - RP extract: Audio Feature Extraction (by TU Wien)

### 4.4.1 Description

RP\_extract is a library that processes audio data (WAV, PCM, or MP3) as input and analyzes the spectral rhythmic and timbral information in the audio to create different audio descriptors (a.k.a. features).

These audio descriptors can be used to find similar sounding songs, create automatic playlists, make music recommendations etc. Depending on the needs, a range of audio features is available:

#### Rhythmic descriptors:

- RP: captures Rhythm Patterns (see picture at the right - audible frequency vs. repetition frequency) and is able to find songs with similar rhythm
- RH: Rhythm Histogram: simplified rhythm descriptor (roughly containing rhythmic strength for different bpm values)

#### Timbral descriptors:

- SSD: Statistical Spectrum Descriptor: describing timbral aspects of the audio Sonogram; by that it is possible to find songs with similar timbral characteristics
- MVD: The Modulation Frequency Variance Descriptor measures variations over critical audible frequency bands for a specific rhythmic repetitions (derived from a rhythm pattern).

#### Temporal descriptors:

- TSSD: Timbral variations over time (based on SSD)
- TRH: Temporal Rhythm Histograms - rhythmic variations over time (based on RH)

Features can also be combined (e.g. to cover rhythmic **and** timbral aspects).

The output is a corresponding list of feature vectors for the supplied input data, in the following formats:

- Matlab: [SOMlib](#) file format (an extended CSV file format with headers)
- Java: [SOMlib](#) file format or [Weka ARFF](http://www.cs.waikato.ac.nz/ml/weka/arff.html)<http://www.cs.waikato.ac.nz/ml/weka/arff.html>
- Python: CSV or Python dictionary (with the feature abbreviations as dictionary keys and the feature vectors as values)

#### This library is available free and open source:

Python: [https://github.com/tuwien-musicir/rp\\_extract](https://github.com/tuwien-musicir/rp_extract)  
Matlab or Java: <http://ifs.tuwien.ac.at/mir/downloads.html>

Documentation:

Quick description: <http://ifs.tuwien.ac.at/mir/audiofeatureextraction.html>  
Tutorial: [http://ifs.tuwien.ac.at/mir/audiofeatureextraction/tutorial/RP\\_extract\\_Tutorial.html](http://ifs.tuwien.ac.at/mir/audiofeatureextraction/tutorial/RP_extract_Tutorial.html)  
Research paper: [1]

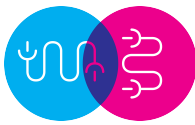
### 4.4.2 Platforms

Implementations:

- Python
- Matlab
- Java

All platforms running Python or Matlab or Java are supported:

- Windows, Mac OS X, Linux



All versions include decoding functionality for MP3 compressed audio through an external program (mpg123, ffmpeg).

#### 4.4.3 Interface Description

Function name: **rp\_extract**

- **data**: pcm signal data
- **samplerate**: signal sampling rate
- **extract\_rp**: extract Rhythm Patterns features
- **extract\_ssd**: extract Statistical Spectrum Descriptor
- **extract\_sh**: extract Statistical Histograms
- **extract\_tssd**: extract temporal Statistical Spectrum Descriptor
- **extract\_rh**: extract Rhythm Histogram features
- **extract\_trh**: extract temporal Rhythm Histogram features
- **extract\_mvd**: extract modulation variance descriptor
- **resample**: do resampling before processing: 0 = no, > 0 = resampling frequency in Hz
- **skip\_leadin\_fadeout**: how many sample windows to skip at the beginning and the end
- **step\_width**:  $\geq 1$  each  $\text{step\_width}$ 'th sample window is analyzed
- **n\_bark\_bands**: 15 or 20 or 24 (for 11, 22 and 44 kHz audio respectively)
- **mod\_ampl\_limit**:
- **spectral\_masking** use Spectral Masking
- **transform\_db**: use transformation into DeciBel
- **transform\_phon**: use transformation into Phon
- **transform\_sone**: use transformation into Sone (requires transform\_phon enabled)
- **fluctuation\_strength\_weighting**: apply Fluctuation Strength weighting curve

#### 4.4.4 Readiness level and IPR

**Java version:**

- Version 0.7 available
- Not fully implemented and supported
- Free and open source (further development is encouraged)

**Python, Matlab versions:**

- Stable and reliable. Tested in scientific settings throughout many years.
- Download code and examples from GitHub: [https://github.com/tuwien-musicir/rp\\_extract](https://github.com/tuwien-musicir/rp_extract)
- Overview and documentation provided on: <http://ifs.tuwien.ac.at/mir/musicbricks>
- Tutorial provided (also available as interactive iPython notebook)
- Free and open source

## 4.5 Search by Sound: Music Similarity Retrieval API (by TU Wien / Spectralmind)

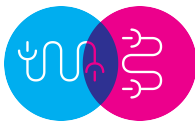
### 4.5.1 Description

The “Search by Sound” system can analyze music tracks and store their fingerprints (features) in a database. Via an API, one can query for similar sounding songs, using a combination of audio-feature based and meta-data query (e.g. “give me rhythmically similar songs from the genre ‘Electronic’”).

The system can be set up with a custom (or empty) music library where additional songs can be added via the API. We also provide a pre-analyzed library with 50,000 songs from <http://freemusicarchive.org>

“Search by Sound” consists of:

- SMAFE Audio Feature Extractor, a server side application that analyzes rhythmic and spectral content of MP3 files and stores them in a database



- SMINT Audio Similarity System which computes similarities between the different feature vectors of all songs in the database
  - SMINT API: A REST Web API to query for similar songs, as well as adding and removing songs (see Interface Description below)
  - Search by Sound Web Frontend: A web frontend to interactively search for songs in the database, retrieve and listen to similar songs and perform segment searches based on a waveform segment of a song.

#### 4.5.2 Platforms

- SMAFE and SMINT run as binary libraries on Linux systems (optionally Mac OS)
- SMINT API runs on Apache Web Server using PHP 5.3
- all this is ready hosted at: <http://musicbricks.ifs.tuwien.ac.at> [2]
- SMINT API can be queried by any application capable of accessing **REST APIs**
- Search by Sound Web Frontend is provided for testing and evaluation purposes (can be also customized / extended on request)

#### 4.5.3 Interface Description

The SMINT API is a REST API which can be queried by using HTTP requests.

The API methods are:

- **track/add ...** Add a Track by sending a URL where the track a) can be downloaded or b) providing a local file location (on the same server as the API is running).
- **track/delete/:smint\_track\_id ...** Removes a track from the system.
- **track/:smint\_track\_id ...** Returns a list of tracks that are similar to the given trackid.
- **track\_external\_key/:external\_key ...** Returns a list of tracks that are similar to the given external key.
- **version ...** Returns version information on the API.

The API returns a proper HTTP status code and an XML document.

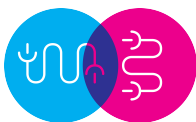
#### 4.5.4 Readiness level and IPR

- Stable and tested in industrial environments.
- Service is running as server backend on server <http://musicbricks.ifs.tuwien.ac.at>
- The API is accessible via endpoint: <http://musicbricks.ifs.tuwien.ac.at/smintapi/>
- Overview and documentation provided on: <http://ifs.tuwien.ac.at/mir/musicbricks>
- Full API documentation provided: [http://ifs.tuwien.ac.at/mir/musicbricks/SMINT/SMINT-API-Documentation\\_v1.1.1c-SbS-rel1.3.pdf](http://ifs.tuwien.ac.at/mir/musicbricks/SMINT/SMINT-API-Documentation_v1.1.1c-SbS-rel1.3.pdf)
- Binaries of backend implementation only available under commercial license (on request)
- API examples provided
- free usage of API and server time

## 4.6 Freesound API (MTG-UPF)

### 4.6.1 Description

**Freesound** ([www.freesound.org](http://www.freesound.org)) is a state-of-the-art online collaborative audio database, built utilising UPF-MTG's technologies, that contains over 200K sounds uploaded and annotated by registered web users. Freesound is the only EU academic database with an API used by commercial organizations as a resource of Creative Commons licensed sound samples. Freesound is continuously growing at a rate of ~120 new sounds and ~1000 new registered users per day. The contents of Freesound are very heterogeneous, ranging from environmental recordings to instrument samples, including voice, foley, music loops and sound effects. All these sounds are annotated with user-provided



free-form tags and textual descriptions that enable text-based retrieval. Content-based audio features are also extracted from sound samples to provide sound similarity search.

The Freesound API will provide access to:

- Collaborative repository
- CC licensed sounds +200k
- RESTful API
- API Clients:
  - Python, Javascript, Objective-C

With the Freesound API users can browse, search, and retrieve information about Freesound users, packs, and the sounds themselves of course. Users can find similar sounds to a given target (based on content analysis) and retrieve automatically extracted features from audio files, as well as perform advanced queries combining content analysis features and other metadata (tags, etc...). With the Freesound API, you can also upload, comment, rate and bookmark sounds.

#### 4.6.2 Platforms

Freesound API can be queried by any application capable of accessing **REST APIs**.

#### 4.6.3 Interface Description

The following functionalities will be supported by the defined interface:

- **Authentication:** Supports standard token (api key) authentication or OAuth2 for post requests that need to be linked to a Freesound user account.
- **Searching:** Search based on textual terms, audio features, geo-tagging information, or a combination of these.
- **Downloading sounds:** Download original quality sounds or lossy previews with different qualities in a unified format (either mp3 or ogg).
- **Uploading sounds:** Use OAuth2 authentication to link API requests to a Freesound user account and upload sounds to Freesound from a third party application using the API.

Additionally, a set of Client Libraries for Freesound API will be included:

- Python
- Javascript
- Objective-C (iOS)

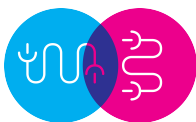
For example, using the Python client for Freesound API, one could make a query and retrieve a list of files using the example code below:

```
import freesound, sys,os
c = freesound.FreesoundClient()
c.set_token("<your_api_key>","token")
results = c.text_search(query="dubstep",fields="id,name,previews")
for sound in results:
    sound.retrieve_preview(".",sound.name+".mp3")
    print(sound.name)
```

By the end of the tasks in WP3 (M12), we shall be able to provide accompanying example applications such as:

1. Freesound Drum Machine
  - a. Web app
  - b. Assign FS samples to an online step sequencer
2. Freesound SampleBank Creation
  - a. Web app
  - b. Specify tags and build a SampleBank archive





- c. Compatibility with common sampler formats (e.g. soundfonts)

## 4.7 Melody Extraction API (MTG-UPF)

### 4.7.1 Description

The Melody Extraction API is based on the Essentia Library (<http://essentia.upf.edu>) [3,4], developed at the Music Technology Group of Universitat Pompeu Fabra in the last years.

Essentia is an open-source C++ library for audio analysis and audio-based music information retrieval released under the [Affero GPLv3 license](#). It contains an extensive collection of reusable algorithms which implement 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.

For #MusicBricks, we will provide a specific library for melody extraction, both from solo or a cappella recordings, as well as for predominant melody extraction from polyphonic audio.

Melody is the most salient and identifiable element in a musical piece. It had a prominent role in the origins of music, which relied on oral tradition or the usage of rudimentary instruments (e.g. prehistoric flutes). But still today, melodies are at the musical forefront from an educational context to the latest commercial song hit. In the scope of MusicBricks, we aim at promoting music creation and therefore we consider the use and reuse of melodies as a fundamental step. Two existing and complementary technologies for melody extraction will be integrated in MusicBricks and accessed through an API. The first technology component inputs a cappella recordings by users and the second technology component processes polyphonic mixtures with predominant vocals. As part of the Ideas Incubation stages of MusicBricks (WP6), these melodies could then control external instrument synthesizers and remixed with other musical accompaniment.

### 4.7.2 Platforms

The MelodyExtraction library is developed in C++. Python bindings are also provided in order to be able to use Essentia in an interactive development environment, and they fit naturally with the IPython/NumPy/Matplotlib environment (similar to Matlab).

The library is cross-platform and currently supports:

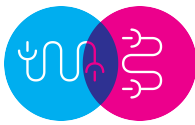
- Linux
- OSX

### 4.7.3 Interface Description

- Inputs:
  - audio filename (WAV, 44.1kHz, 16bit)
- Output:
  - YAML file with pitch values in Hz and pitch confidence

### 4.7.4 Readiness level and IPR

- Technology is ready to be tested. Download code and examples from <http://essentia.upf.edu>
- Fully documented algorithms and tutorial examples as part of Essentia library
- Stable version 2.1: library used in several projects
- Affero GPL License: source code available. Github repository



## 4.8 Onset Description API (MTG-UPF)

### 4.8.1 Description

Onset Description is part of the EssentiaRT~, and it is provided as a binary file compiled for Mac OSX, Linux and Windows. Additionally, it comes with a collection of abstractions designed to make interaction with the object easier. A number of help files and use examples complete the package. EssentiaRT~ is offered free of charge for non-commercial use only.

### 4.8.2 Platforms

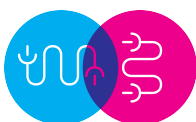
- Max OS X (10.7 or newer), a recent version of Debian/Ubuntu, Microsoft Windows 7/8 32-bit or 64-bit)
- Software
  - Pd-extended (version 0.42.5 or newer) or Max (Version 5 or newer).
  - On Mac and Windows please make sure you use 32-bit versions of Pd and Max.

Please note: examples are not provided for Pd and not Max (save for a help file). Please consult the Pd patches and adopt them for your own needs in Max.

### 4.8.3 Interface Description

### 4.8.4 Readiness level and IPR

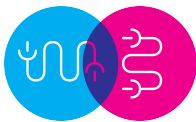
- Technology is ready to be tested. Download code and examples from <http://mtg.upf.edu/technologies/EssentiaRT~?p=Download%20and%20installation>
- Fully documented algorithms and tutorial examples as part of EssentiaRT~ library
- Development version v0.2: library used in several projects
- Affero GPL License: source code available. Github repository



## 5. Download and installation

Next table compiles the #MusicBricks included in the First Version of the APIs, including the download links, system requirements and licensing terms.

Tool	Platform	Programming Environment	Real-Time Usage	Links for Installation and Usage	Provider	License
Melody Extraction	Linux Mac OS X Windows	C++ Python Executables	No	<a href="#">OverviewDownloadTutorial (Python code and Binay executables)</a>	UPF MTG	Affero GPLv3
Onset Description	Linux Mac OS X Windows	PureData MaxMSP	Yes	<a href="#">Documentation and download</a>	UPF MTG	Open source
Rhythm and Timbre Analysis	Linux Mac OS X Windows	Python Matlab Java	Possible	<a href="#">Overview Downloads GitHubTutorial Feature Description</a>	TU Wien	Open source
Search by Sound Music Similarity Web API	any	REST API	No	<a href="#">OverviewAPI Documentation Frontend tutorial</a>	TU Wien – Spectralmind	Free usage of API (Binaries and source subject to licensing)
MusicBricksTranscriber (Melody & Bass Transcription + Beat & Key & Tempo Estimation)	Linux Mac OS X Windows	Excecutable , Python Wrapper	No	<a href="#">Documentation &amp; Download</a>	Fraunhofer IDMT	Royalty-free, limited, non-transferable license
Real-time Pitch Detection	Linux Mac OS X Windows iOS Android	C++ Library	Yes	<a href="#">Documentation &amp; Download</a>	Fraunhofer IDMT	Royalty-free, limited, non-transferable license
Gesture Sensors and Gesture Analysis	Any platform with OSCfor receiving raw data MacOS or Windows for Gesture Analysis with MaxMSP	MaxMSP	Yes	<a href="http://ismm.ircam.fr/devices/">http://ismm.ircam.fr/devices/</a>	IRCAM	Sensor should be returned after the hackathon,Max external and patches free
FreeSound API	Web API	Python, Javascript and Objective C clients	No	<a href="#">Documentation</a>	UPF MTG	<a href="https://www.freesound.org/docs/api/terms_of_use.html">Terms of use https://www.freesound.org/docs/api/terms_of_use.html</a>



## 6. Conclusions

The First Version of the APIs will be thoroughly put in practice by users in the selected projects in the Creative Testbeds in the next months. This integration work will provide valuable feedback to improve the technologies and API towards the final release (M12), as further explained in D5.1 and D6.1.

As specific next steps, in WP3 we will provide:

- an extension of the portfolio with new features or new #MusicBricks (related to Task 3.3)
- implementation of Demo Applications by combining multiple #MusicBricks (related to Task 3.2)

The results of the future steps in WP3 will be reported in the deliverable D3.3 Final Version of the APIs, which is due on Month 12.