Deliverable D5.2

First prototype of timbral characterisation tools for semantically annotating non-musical content

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	WP5
Due date	28 April 2017 (M15)
Submission date	28 April 2017 (M15)
Report availability	Public (X), Confidential ( )
Deliverable type	Report ( ), Demonstrator (X), Other ( )
Task leader	SURREY
Authors	Andy Pearce, Tim Brookes, Russell Mason
Document status	Final

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

This report describes the implementation and use of the demonstrator software: six perceptual models that can predict timbral attributes of an audio file: *hardness*, *depth*, *brightness*, *metallic-nature*, *roughness*, and *reverb*. These models can be used to automatically generate metadata describing the timbral properties of sound effects, which can, in turn, be implemented into a search function, enabling users to filter search results based on the timbral properties.

Each model was developed based on either existing models or literature pertaining to the acoustic correlates of timbral attributes. The performance of the models described in this document will be evaluated in the next stage of the project.

The developed models are implemented using the Python programming language and have been made available in a public source code repository[1].

[1] https://github.com/AudioCommons/timbral_models

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 models

In deliverable D5.1, the timbral attributes that were the most searched on *freesound.org* were identified. A literature review was conducted into the 40 most-searched attributes, identifying, for each attribute, any existing models and/or features that could be extracted from an audio signal which relate to the attribute. The most commonly searched timbral attributes for which suitable information was available were then modelled: *hardness*, *depth*, *brightness*, *metallic-nature*, *roughness*, and *reverb*.

Each following subsection describes an attribute, its likely acoustic correlates, and the implementation and use of the timbral model: section 1.1 *hardness*; section 1.2 *depth*; section 1.3 *brightness*; section 1.4 *metallic-nature*; section 1.5 *roughness*; and section 1.6 *reverb*.

1.1 Timbral Hardness

The *hardness* timbral attribute was the most frequently searched [Pearce et al., 2016]. Research by Williams [2010] suggested that the onset portion of a sound determines the perception of hardness. Additionally, research by Freed [1990] presented a model of mallet hardness perception for single percussive sounds with respect to four acoustic correlates: 1) spectral mean level (a form of long term average spectrum, LTAS); 2) spectral level slope (similar to cepstrum analysis); 3) spectral centroid mean (mean spectral centroid over time, measured on the bark scale); and 4) spectral centroid TWA (time weighted mean of the spectral centroid).

Research by Solomon [1959] also identified the hardness/softness perceptual dimension as a key psychological dimension of timbre. This work hypothesised that this attribute relates to a rhythmic difference between stimuli, but no metrics for measuring this were proposed.

Although no explicit model of hardness exists in the literature, there is an indication that the attack portion of a sonic event determines the apparent hardness, along with the spectral content of the attack. Therefore, a model of hardness was developed which employs three metrics: (i) attack time; (ii) attack gradient; and (iii) spectral centroid of attack. A linear regression model was then used to estimate the apparent hardness from these parameters.

Section 1.1.1 describes the implementation of the model and explains how the model extracts the relevant features. Section 1.1.2 describes the linear regression modelling, and Section 1.1.3 describes how the model can be used.

1.1.1 Hardness implementation

The implemented model of hardness first loads the specified audio file using the soundfile python library. Currently, only the left channel of the audio signal is analysed, as multi-channel signals may alter the measurable attack time. This will be addressed in a future update.

The audio is then zero-padded before and after the signal with 512 samples to allow for onsets to be detected at the beginning and end of the audio file. The model then identifies event onsets using the `onset_detect` function from the LibROSA python package [McFee, B. et al., 2015]. This analysis is conducted with the default frame size of 512 samples.

The envelope of the audio signal is then estimated by applying a causal function to the absolute values of the time domain signal. It is common to track a signal's amplitude envelope using a

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

low-pass filter. However, this approach places a limit on the fastest rise-time that can be accurately tracked. The approach taken here is therefore to track absolute signal amplitude exactly while it is rising, and to ignore amplitude falls unless they are maintained for at least 10ms. Amplitude falls are smoothed with a linear decay function, with the slope of the decay calculated so that falling from maximum level to zero would take 200ms. An example of the output of this causal function is shown in Figure 1.

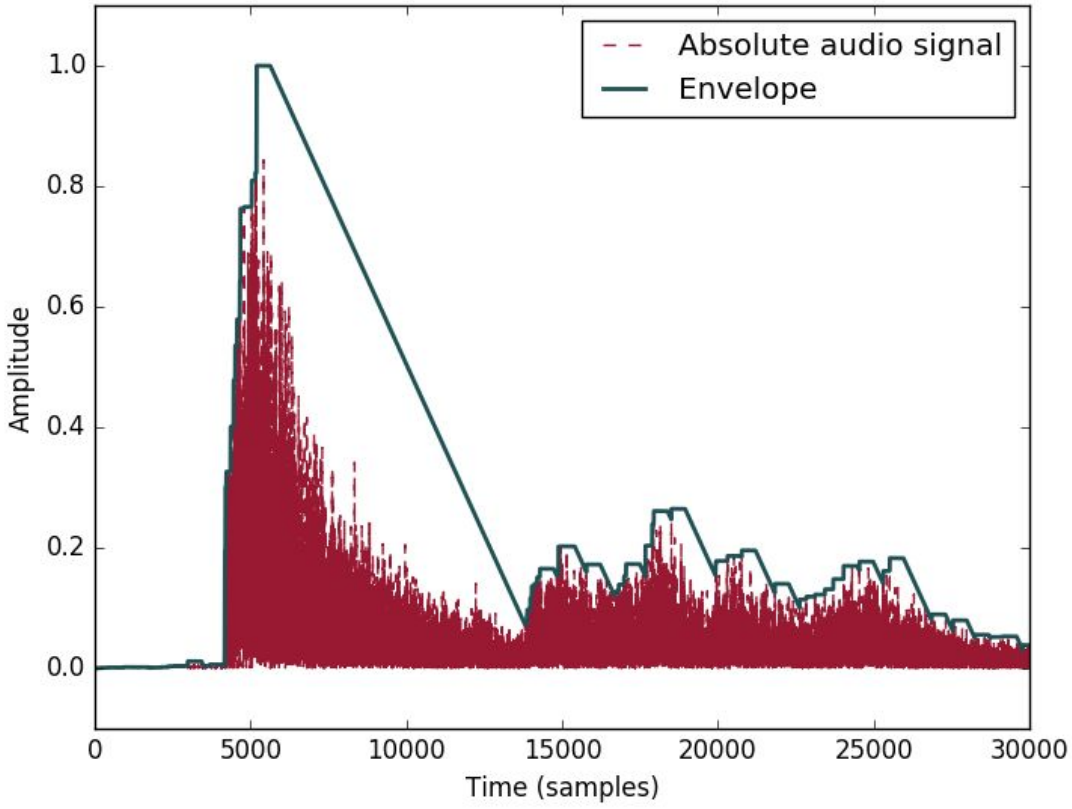

Figure 1 - Result of causal function to estimate signal envelope.

The model then calculates more precisely the times at which onsets occur using the estimated envelope, since the LibROSA onset detection function only identified onsets to the nearest 512 samples. An iterative function is called which looks at the previous 10ms of samples, identifies any values smaller than the current onset, and moves the onset back to this minimum sample. This is shown in the code snippet below, where a simplified version of this function is implemented, where 10ms relates to 441 samples (at 44.1kHz sample rate).

```
while found_onset = False:
        current_sample = envelope[onset]  # value of the current sample
        previous_samples = envelope[onset-441 : onset-1]  # previous 10ms of samples
        if min(previous_samples) < current_sample:  # check for smaller value
                onset = argmin(previous_samples) # set onset to the index of minimum value
        else:
                found_onset = True
```

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

Once an onset has been found in this manner, the function then checks the previous 200ms to allow for small amplitude deviations longer than 10ms. The function checks the dynamic range of samples between the current onset and closest smaller amplitude. If this dynamic range is less than 5% of the total dynamic range, the process of looking backwards 10ms is restarted from this closest smaller sample. A simplified version is shown in the code snippet below, with 200ms represented by 8820 samples (at 44.1kHz sample rate).

```
current_sample = envelope[onset]
previous_samples = envelope[onset-8820:onset-1]
if min(previous_samples) < current_sample:
        # closest sample less than current sample
        last_min_idx = where(previous_samples < current_sample)[-1]

        # get the dynamic range of segment
        dynamic_range=max(envelope[last_min_idx:onset])-min(envelope[last_min_idx:onset])
        if dynamic_range < (0.05 * overall_dynamic_range):
                # dynamic range is small enough to be considered part of the same attack
                onset = argmin(envelope[last_min_idx:onset])
                return  # return to previous iterative function with new onset
```

Any duplicate onsets (caused by the look back function) are removed. Each onset is then compared against the overall dynamic range. The dynamic range between the onset and the peak after the onset is calculated. If this is less than 10% of the overall dynamic range, the onset is considered to be noise within the signal and therefore removed from analysis.

For each retained onset, the attack time is calculated using the adaptive threshold method described by Peeters [2004]. The times where the envelope first crosses the 10, 20, 30, 40, 50, 60, 70, 80, and 90% thresholds are identified, th_i where i is the threshold number. This is indicated in Figure 2. For successive thresholds, the effort value, w_i, is calculated as the time between adjacent thresholds, $w_i = t_{(i+1)} - t_i$. The average effort value, w, is then computed.

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

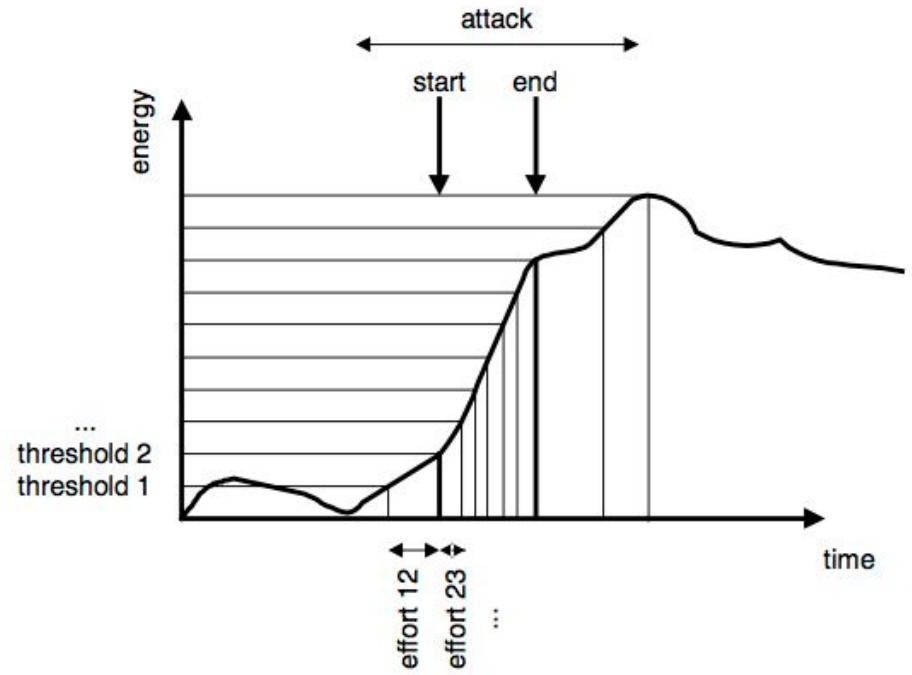

Figure 2 - Least weight method of calculating appropriate start and stop times for estimating attack time, [Peeters, 2004].

The best threshold for the start of the attack, th_{start}, is computed as the first threshold for which the effort value goes below *M*w*. In a similar way, the end of the attack, th_{end}, is determined as the first threshold for which the effort value goes above *M*w*. Peeters, [2004] used a value of *M* = 3. The attack time (metric 1) is then calculated as the logarithm of the difference, in seconds, between the start and stop time:

$$\text{attack time} = log_{10}(th_{end} - th_{start})$$

For instances where th_{start} is the same as th_{end}, the attack time is set to 1/fs (sample rate). The mean attack time is then calculated across all onsets.

The attack gradient (metric 2) is calculated as the difference between the amplitudes of the attack start and end levels divided by the linear attack time. For instances where the the attack starts and stops on the same sample, the start level is taken as being the level of the previous sample or set to zero if the previous sample was the start of the audio file. The mean attack gradient is then calculated across all onsets.

The attack spectral centroid (metric 3) is calculated over the first 125ms after the attack start, or until the next onset time if this happens before 125ms. The mean attack spectral centroid is then calculated across all onsets.

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

1.1.2 Linear regression modelling

The hardness of the audio file is represented by a single value calculated from a linear regression. The three authors participated in a short informal listening test, rating the apparent hardness of each of 15 audio samples on a scale of 0 to 100. A linear regression was then calculated from the three metrics (and all possible interactions) to the listening test result means. Using these metrics and interactions, the model can predict the results with a correlation of 0.90 and an RMSE of 11.81, shown in Figure 3.

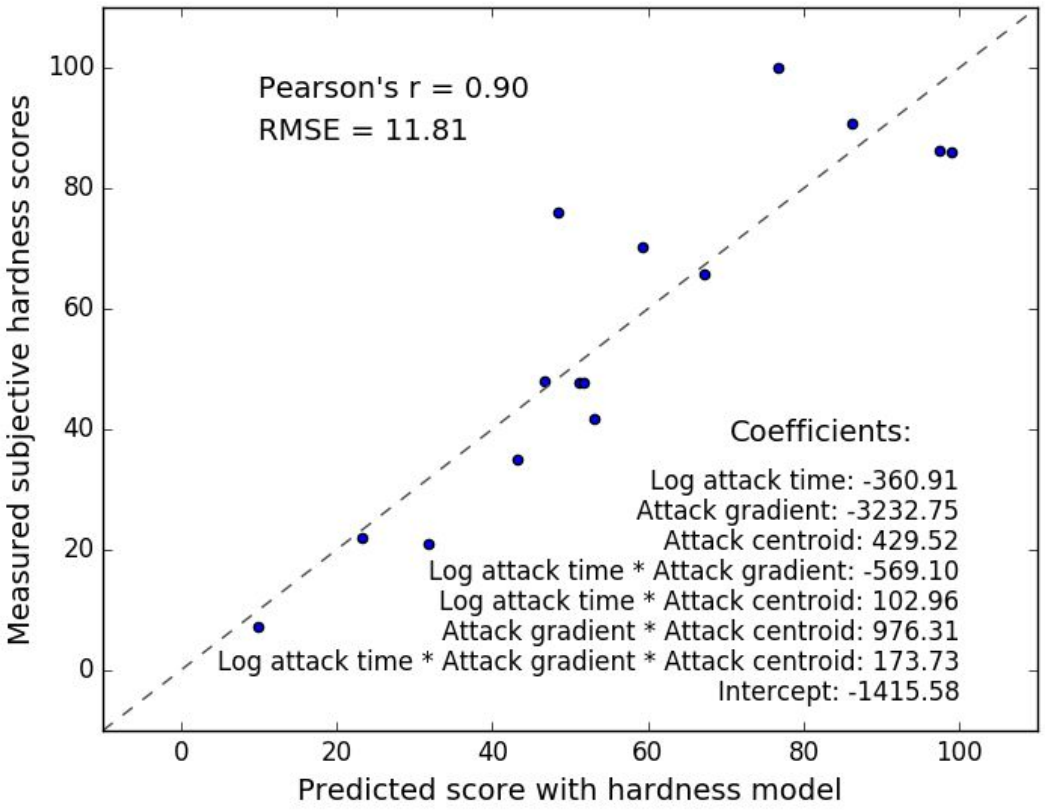

Figure 3 – Mean subjective hardness ratings against predicted scores from hardness model.

1.1.3 Using the hardness model

The hardness model relies on the numpy, LibROSA, and soundfile python packages. These can all be installed using the pip tool, e.g. `pip install numpy`

The model is a python module written in python 2.7. The function `timbral_hardness` can then be called and given a string that is the path and filename of an audio file. For example:

```
import Timbral_Hardness
fname = '/Documents/Music/TestAudio.wav'
hardness = Timbral_Hardness.timbral_hardness(fname)
```

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

This will return the hardness of the audio file as a single floating point value. Although the model was trained on data ranging from 0 to 100, the values output can be beyond these due to the nature of linear regression.

1.2 Timbral Depth

Depth can be a timbral or a spatial attribute. In this project, it is considered only in its timbral sense.

Whilst the attribute of depth is mentioned in several academic papers, no model and no suggested acoustic correlates have been reported. However, an online experiment by Cartwright and Pardo [2013], called Social-EQ, asked subjects to submit a timbral descriptor together with an appropriate setting on a 40-band graphic equaliser that demonstrates that descriptor. Six subjects chose to submit the term *deep*. The 40-band equalisation treatment submitted by each subject is shown in Figure 4. The mean equalisation of all subjects, and 95% confidence intervals, are shown in the thicker black line.

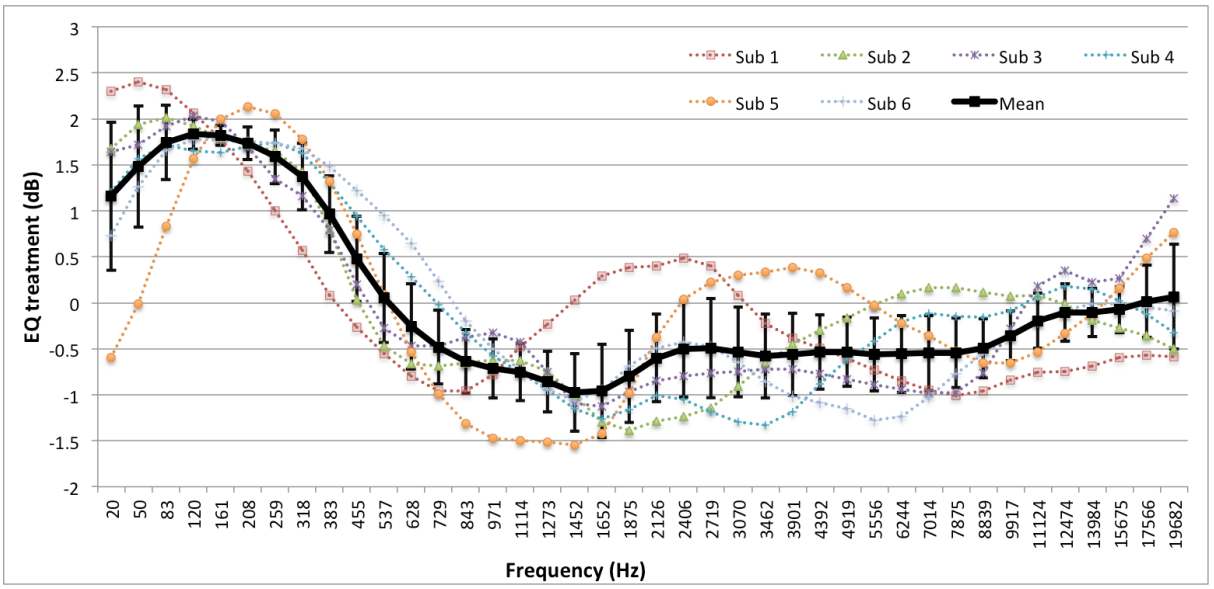

Figure 4 - Social-EQ graphic equaliser settings representing the timbral descriptor *deep*.

There is a clear trend in Figure 4 that all subjects' EQ treatments emphasised the low frequency content of the signal. Since there is a large degree of commonality in these EQ treatments, it is likely that timbral depth is related to having emphasised low frequency content. Therefore, it is suggested that a suitable model for depth would be to analyse: 1) the spectral centroid of the lower frequencies (energy pulling towards the low-end); 2) the proportion of low frequency energy; and/or 3) the low-frequency limit of the audio extract (the low frequency roll-on).

As with hardness, no existing model of depth exists. Therefore, a short listening experiment followed by a linear regression analysis was conducted to model it.

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

1.2.1 Depth implementation

The implemented model of depth first loads the specified audio file. Currently, only the left channel of the audio signal is analysed. This will be addressed in future updates. A spectrogram function is used to get the frequency domain representation of the audio split into time frames. This uses a Hawindow of length 4096, ana step size of 1024. For example, frame 1 contains samples 1 to 4096, frame 2 contains samples 1025 to 5120, frame 4 contains samples 2049 to 6144, etc.

For each frame, the model calculates the lower spectral centroid as the spectral centroid of the frequencies between 30 Hz and 200 Hz (frequencies chosen by informal listening):

$$\text{Lower spectral centroid} = \frac{\sum_{n(30Hz)}^{n(200Hz)} f(n)\,x(n)}{\sum_{n(30Hz)}^{n(200Hz)} x(n)}$$

where $n(\omega)$ is the bin number relating to frequency ω, $f(n)$ is the frequency of the n^{th} bin, and $x(n)$ is the magnitude of the n^{th} bin. The mean lower spectral centroid is then calculated across all frames.

The model also calculates the lower ratio for each frame (the ratio of energy between 30Hz and 200Hz, compared to all energy between 30 Hz and the Nyquist frequency):

$$\text{Lower ratio} = \frac{\sum_{n(30Hz)}^{n(200Hz)} x(n)}{\sum_{n(30Hz)}^{n(Nyquist)} x(n)}$$

where n(Nyquist) is the frequency bin relating to the Nyquist frequency. The mean lower ratio is then calculated across all frames.

The last metric calculated by the model is the low-frequency limit. This is calculated similarly to the '*spectral rolloff*' metric from the IRCAM timbre toolbox [Peeters, 2004]. The low-frequency limit, for this model, is defined as the frequency that 95% of the spectral energy lies above:

$$\sum_{n(30Hz)}^{n(Low\text{–}frequency\ limit)} x(n)^2 = \frac{\sum_{n(30Hz)}^{n(200Hz)} x(n)}{\sum_{n(30Hz)}^{n(Nyquist)} x(n)}$$

The mean low-frequency limit is then calculated across all frames.

1.2.2 Linear regression modelling

As with the hardness model, the three authors participated in an informal listening test on 15 stimuli, rating the subjective depth of each. A linear regression between the three metrics, and all

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

interactions, to the mean listening test results was conducted. The resulting model fits the data (Figure 5) with a correlation of r = 0.96 and an RMSE of 7.69.

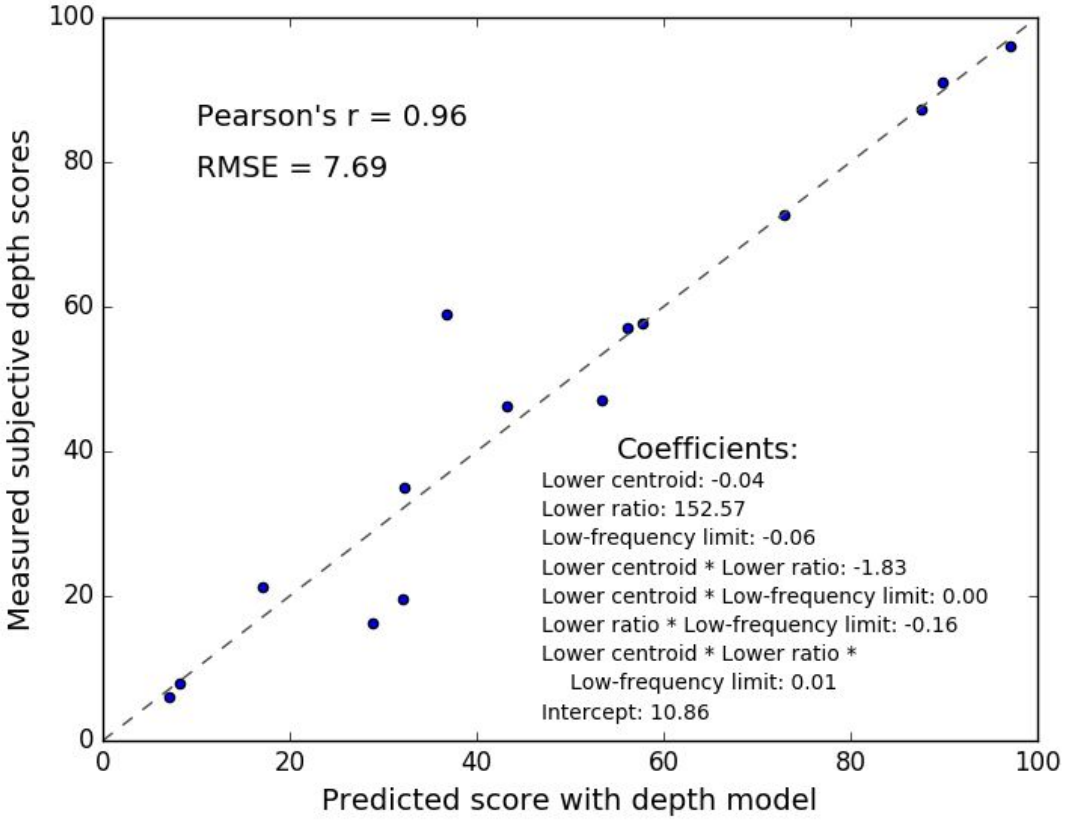

Figure 5 - Mean subjective depth ratings against predicted scores from depth model.

1.2.3 Using the depth model

The depth model relies on the numpy, scipy, and soundfile packages. These can all be installed using the pip tool, e.g. `pip install numpy`

The model is a python module written in python 2.7. The function `timbral_depth` can then be called and given a string that is the path and filename of an audio file.

```
import Timbral_Depth
fname = '/Documents/Music/TestAudio.wav'
depth = Timbral_Depth.timbral_depth(fname)
```

This will return the depth of the audio file as a single floating point value. Although the model was trained on data ranging from 0 to 100, the values output can be beyond these due to the nature of linear regression.

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

1.3 Timbral brightness

Brightness is a timbral attribute that has been studied in some detail. It has been shown in some studies that the spectral centroid is a measure that correlates with perceived brightness [Schubert and Wolfe, 2006; Schubert et al., 2004; Poirson et al., 2005; Grey and Gordon, 1978]. However, some research also suggests that the ratio of high frequencies to the sum of all energy is a better predictor [Greenwood, 2015; Omori, 2011; Lartillot and Toiviainen, 2007; Juslin, 2000; Laukka et al., 2005]. In recent work, Pearce [2017] surveyed existing models and developed a new model of brightness incorporating both a spectral centroid variant and a spectral energy ratio. This model has now been coded in a suitable format for this project.

1.3.1 Brightness implementation

The implemented model of brightness first loads the specified audio file. Currently, only the left channel of the audio signal is analysed. The audio is split into frames using a Hamming window of length 2048, with a step size of 1024 (terms defined as in Section 1.2.1). An FFT is then conducted on each frame.

The magnitude frequency response for each frame of audio is calculated and smoothed with a sample-by-sample half-octave smoothing algorithm. The smoothed response is then used to calculate two metrics, the frequency-limited spectral centroid (over 3 kHz).

$$\text{Frequency-limited spectral centroid} = \frac{\sum_{n(3kHz)}^{n(Nyquist)} f(n)\,x(n)}{\sum_{n(3kHz)}^{n(Nyquist)} x(n)}$$

and the ratio of energy over 3kHz to the sum of all energy between 20Hz and the Nyquist frequency

$$\text{Ratio} = \frac{\sum_{n(3kHz)}^{n(Nyquist)} x(n)}{\sum_{n(20Hz)}^{n(Nyquist)} x(n)}$$

where $n(\omega)$ is the bin number relating to frequency ω, $f(n)$ is the frequency of the n^{th} bin, $x(n)$ is the magnitude of the n^{th} bin, and n(Nyquist) is the bin number relating to the Nyquist frequency. The mean frequency-limited spectral centroid and ratio are calculated over all frames.

A linear regression was used by Pearce [2017] to obtain results of brightness. The coefficients from this regression were applied as:

$$\text{Brightness} = -25.8699 + 64.0127((log_{10}(Ratio) + 0.44log_{10}(SC_{3kHz}))$$

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

1.3.2 Using the brightness model

The brightness model relies on the numpy and soundfile packages. These can all be installed using the pip tool, e.g. `pip install numpy`

The model is a python module written in python 2.7. The function `timbral_brightness` can then be called and given a string that is the path and filename of an audio file.

```
import Timbral_Brightness
fname = '/Documents/Music/TestAudio.wav'
brightness = Timbral_Brightness.timbral_brightness(fname)
```

This will return the depth of the audio file as a single floating point value. Although the model was trained on data ranging from 0 to 100, the values output can be beyond these due to the nature of linear regression.

1.4 Timbral metallic-nature

In past studies, researchers have attempted to develop models that can predict, from an audio recording of an object being struck, whether the struck object is metallic [Aramaki et al., 2011; Hjortjær and McAdams, 2016]. In the work of Aramaki et al. [2011], a model incorporating four parameters was used to categorise impacting sounds as metallic, wooden, or glass. Metrics for the four parameters are coded here and used to construct a new logistic regression model specifically to predict the metallic-nature of an input signal.

1.4.1 Metallic-nature implementation

The implemented model first reads in the specified audio file. Currently, only the left channel of the audio signal is analysed. This will be addressed in a future update. The metallic-nature model calculates four parameters: the attack time, spectral spread, normalised decay time, and roughness.

The attack time of the signal is calculated in a similar manner to the attack time calculation in the hardness model (Section 1.1.2). The same causal function is applied to a zero padded signal; the LibROSA `onset_detect` function is used to estimate onset times, and the same approach is taken to identify more precise onset times.

For the metallic model, the actual attack time is calculated as the time taken, for each onset, for the signal to increase to 10% to 90% of the dynamic range, as suggested by Aramaki et al. [2011]. This is exemplified in Figure 6.

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

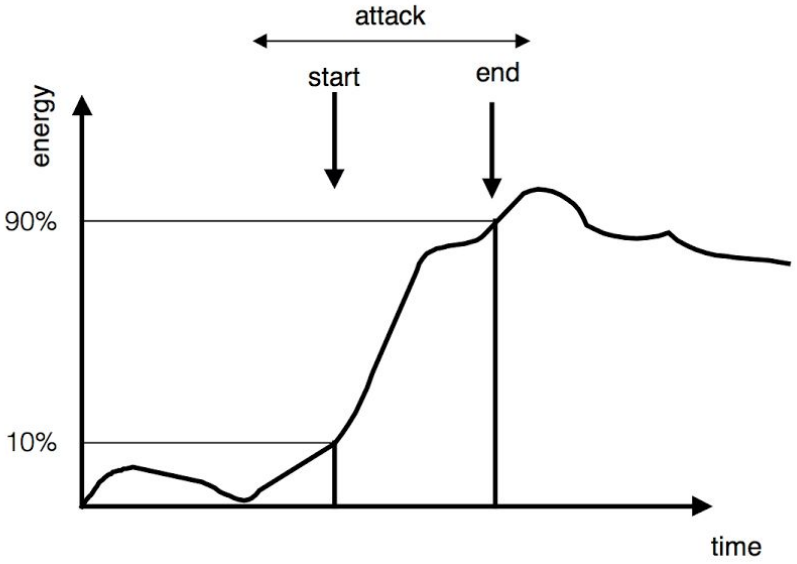

Figure 6 - Fixed threshold method calculating attack time [Peeters, 2004].

The spectral spread is calculated for each onset with the equation:

$$\text{Spectral spread} = \sqrt{\frac{\sum_{n(20Hz)}^{n(Nyquist)} (f(n)-SC)^2 x(n)}{\sum_{n(20Hz)}^{n(Nyquist)} x(n)}}$$

where $n(\omega)$ is the bin number relating to frequency ω, n(Nyquist) is the bin number relating to the Nyquist frequency, $f(n)$ is the frequency of the n^{th} bin, $x(n)$ is the magnitude of the n^{th} bin, and SC is the spectral centroid, calculated with the equation:

$$\text{Spectral centroid (SC)} = \frac{\sum_{n(20Hz)}^{n(Nyquist)} f(n)\, x(n)}{\sum_{n(20Hz)}^{n(Nyquist)} x(n)}$$

The mean spread is then calculated over all onsets.

The normalised decay time was calculated as suggested by Amaraki et al. [2011]. The envelope of the signal is estimated by first taking the absolute of the Hilbert transform of the audio signal. This is then low pass filtered with a second-order low-pass Butterworth filter with a cut-off frequency of 50Hz. The logarithm of this is then taken. This is expressed in the equation:

$$\text{Envelope} = log_{10}(F(|H(x)|))$$

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

where *x* is the audio signal, ***H***(*x*) is the Hilbert transform of *x*, and ***F***(*x*) represents filtering of the signal.

For each onset, the decay is evaluated as the time between the peak and the next onset. For every 100ms of the decay, the algorithm conducts a linear regression, calculating the mean square error (MSE) and Pearson's r. The most linear decay portion is then taken by finding lowest MSE with a negative Pearson's r. The linear regression of this 'most linear' part is then taken, and the level decayed in 100ms is then estimated from the linear regression.

The roughness is calculated as set out in Section 1.5.

1.4.2 Logistic regression modelling

Formal listening tests were conducted by a final year undergraduate student on the Tonmeister course, at the University of Surrey [Martin, 2017]. The test comprised 80 stimuli, asking subjects to rate each stimulus as 'metallic' or not. Results from all 20 subjects who participated in the test were used to calibrate the logistic regression model using the parameters of normalised decay time, attack time, spectral spread, and roughness. Coefficients of the variables were used in an inverse Logit function to obtain the probability of a signal being metallic. This is calculated as:

$$p = \frac{e^y}{1 + e^y}$$

where *p* is the probability of the signal being metallic, and *y* is the regression equation calculated as:

y = (normalised decay * 528.4) + (attack time * −0.2371) + (spectral spread * 0.0001187) + (roughness * 0.001700) − 1.084

Coefficients are presented here to four significant figures.

1.4.3 Using the model

The *metallic-nature* model relies on the numpy, scipy, LibROSA and soundfile packages. These can all be installed using the pip tool, e.g. `pip install numpy`

The model also required the `Timbral_Roughness` model to be installed.

The model is a python module written in python 2.7. The function `timbral_metallic` can then be called and given a string that is the path and filename of an audio file.

```
import Timbral_Metallic
fname = '/Documents/Music/TestAudio.wav'
metallic = Timbral_Metallic.timbral_metallic(fname)
```

This will return the probability that the audio file sounds metallic, between 0.0 and 1.0.

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

1.5 Timbral roughness

Although there are several methods to calculate the apparent roughness of an audio signal, the model of metallic-nature relies on the Vassilakis model of timbral roughness [Vassilakis, 2007] and so that same model is implemented here. This models the interaction of peaks within the frequency spectrum, where peaks of similar amplitude and frequency result in a 'rough' sensation.

1.5.1 Roughness implementation

The MIR toolbox's implementation of the roughness model [Lartillot and Toiviainen, 2007] divides the audio into 50ms frames. Each frame is then windowed with a Hanning window and subsequently zero-padded, after the frame, to the next nearest power of two. An FFT is then made of each frame, and the frequency-specific magnitudes of all frames are normalized, so that the maximum magnitude across all frequencies and frames is 1.0.

A peak-picking algorithm is then used on each frame to identify peaks within the frequency spectrum. For each frame, the algorithm identifies all peaks in the spectrum where: (*i*) the magnitude of the frequency bin is greater than 0.01; (*ii*) the magnitude of the previous and next frequency bins are less than the current bin; and (*iii*) in the frequency range between successive peaks the magnitude drops at least 0.01 below the magnitude of the lower peak. For each pair of peaks within a frame, the roughness is calculated with the equation:

$$r = 0.5X^{0.1}Y^{3.11}Z$$

with:

$$X = A_{min} * A_{max}$$

$$Y = \frac{2A_{min}}{A_{min}+A_{max}}$$

$$Z = e^{(-3.5s(f_{max}-f_{min}))} - e^{(-5.75s(f_{max}-f_{min}))}$$

$$s = \frac{0.24}{0.0207f_{min}+18.96}$$

Where r is the roughness, A_{max} and A_{min} are the maximum and minimum magnitudes of the pair of peaks, and f_{max} and f_{min} are the maximum and minimum frequencies of the two peaks respectively [Vassilakis, 2007].

The overall roughness of a frame is the sum of all roughness pairs. The overall roughness of an audio file is then calculated as the mean of all frames' roughness values.

1.5.2 Using the model

The roughness model relies on the numpy and soundfile packages. These can all be installed using the pip tool, e.g. `pip install numpy`

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

The model is a python module written in python 2.7. The function `timbral_roughness` can then be called and given a string that is the path and filename of an audio file.

```
import Timbral_Roughness
fname = '/Documents/Music/TestAudio.wav'
roughness = Timbral_Roughness.timbral_roughness(fname)
```

This will return the roughness of an audio file as a single floating point value. There is no upper limit on the maximum roughness value.

1.6 Reverb

Reverberation is a term describing the acoustic decay of an audio signal and has several common metrics. The most common metric is the RT60, a measure of the decay time. This measure is commonly taken of concert halls and acoustic spaces, yet the task of estimating the RT60 from a recording is difficult. The IEEE's ACE challenge was conducted in 2015 as an attempt to blind-estimate the RT60 and direct-to-reverberant ratio of recorded speech signals [Eaton et al. 2016]. The best performing algorithm, as suggested by Prego et al. [2015], was implemented for this project.

1.6.1 Reverb implementation

The reverb algorithm was implemented as described by Prego et al. [2012]. A power spectrogram of the signal is calculated. No frame length or window function is specified, so a Hamming window of length 2048 is chosen arbitrarily, with a frame overlap of 512 samples. Analysis is then conducted on frequencies between 20 Hz and 4 kHz. This frequency region contains the majority of information in speech signals, and also covers the frequency range normally employed for the measurement and specification of RT60 in building acoustics (averaged over 500Hz, 1kHz, and 2kHz octave bands).

Analysis is conducted per frequency bin (sub-band) in the spectrogram. For each sub-band, the algorithm searches for sub-band free decay regions (SFDR)—areas in the sub-band where the magnitude continuously drops for at least 500ms (e.g. 14 frames at 44.1kHz sample rate). If no SFDR are identified, the required number of successive decreasing frames is iteratively reduced from the initial number of frames until either at least one SFDR is found, or the required number of successive frames reaches 3.

For each identified SFDR, the Schroeder integration is calculated with the equation:

$$c(k,l,n) = 10log_{10}\left(\frac{\sum_{a=n}^{a=L} E(k,a)}{\sum_{a=n}^{a=L} E(k,a)}\right)$$

where $c(k,l,n)$ is the n^{th} frame in the l^{th} SFDR in the k^{th} sub-band, L is the total number of frames within the SFDR, $E(k,a)$ is the energy in the k^{th} sub-band at frame number a, and n is the current frame being analysed.

The Schroeder integral for each SFDR is then analysed in order to estimate the SFDR RT60. Firstly, the start of the analysis interval is selected as the first frame within the SFDR where the Schroeder

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

integral is less than −5dB. A linear regression is then calculated between all valid start and end frames within the SFDR. The most linear segment of the SFDR is then identified as the start and end frames that produce the minimum mean square error (MSE).

If this most linear segment of the Schroeder integral has a dynamic range less than 10dB, the most linear segment is selected that has a dynamic range of at least 60dB. If a 60dB dynamic range is not available for any segment, the algorithm searches for the most linear segment with at least 40dB of dynamic range, with this criteria reduced to 20dB and then 10dB if no suitable segment is identified.

A linear regression is then conducted on the selected segment of the SFDR. The SFDR RT60 is then calculated from the regression coefficients as the time it would take for this linear regression line to decay by 60dB.

The sub-band RT60 is then calculated as the median across all SFDRs within the sub-band. The overall RT60 is then estimated as the median of all sub-band RT60s. An arbitrary correction is applied by dividing the calculated RT60 by 3 in order to better estimate the actual RT60.

1.6.2 Using the model

The *reverb* model relies on the numpy, scipy, and soundfile packages. These can all be installed using the pip tool, e.g. `pip install numpy`

The model is a python module written in python 2.7. The function `timbral_reverb` can then be called and given a string that is the path and filename of an audio file.

```
import Timbral_Reverb
fname = '/Documents/Music/TestAudio.wav'
RT60 = Timbral_Reverb.timbral_reverb(fname)
```

This will return the estimated RT60 in milliseconds of the environment recorded in the audio file. If an RT60 cannot be estimated, the model returns 0.

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 deliverable, timbral models of hardness, depth, brightness, metallic-nature, roughness, and reverb were implemented. These models will be incorporated into the AudioCommons ecosystem as a means of automatically generating metadata that can be used to supplement searches, making it easier for users to identify suitable sound effects.

The models presented will be evaluated in future research as part of WP5. This will involve collecting subjective ratings for each timbral attribute, with a suitable corpus of audio, and comparing those ratings against the predicted values from the timbral models. The models will then be improved and refined as necessary.

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

3 References

Aramaki, M., Besson, M., Kronland-Martinet, R., and Ystad, S., 2011: 'Controlling the perceived material in an impact sound synthesizer', in IEEE Transactions on Audio, Speech, and Language processing, Vol. 19, No. 2, pp. 301 – 314.

Cartwright, M. and Pardo, B., 2013: 'Social-EQ: Crowdsourcing an equalisation descriptor map', in International Society for Music Information Retrieval.

Eaton, J., Gaubitch, N., Moore, A., and Naylor, P., 2016: 'Estimation of room acoustic parameters: The ACE challenge', IEEE Transactions on Audio, Speech and Language Processing, Vol. 24, No. 10, pp. 1681–1693.

Freed, D., 1990: 'Auditory correlates of perceived mallet hardness for a set of recorded percussive sound events', The Journal of the Acoustical Society of America, Vol. 87, No. 1, pp. 311–322.

Greenwood, T., 2015: Defining an Objective Measure of Brightness in Relation to Attributes of Perceived Brightness, Undergraduate technical project, Institute of Sound Recording, University of Surrey, Guildford, Surrey, UK.

Grey, J. and Gordon, G., 1978: 'Perceptual effects of spectral modifications on musical timbres', The Journal of the Acoustical Society of America, Vol. 63, No. 5, pp. 1493– 1500.

Hjortjær, J., and McAdams, S., 2016: 'Spectral and temporal cues for perception of material and action categories in impacted sound sources' in The Journal of the Acoustical Society of America, Vol. 140, No. 1, pp. 409 – 420.

Juslin, P., 2000: 'Cue utilization in communication of emotion in music performance: relating performance to perception', Journal of Experimental Psychology: Human Perception and Performance, Vol. 26, No. 6, pp. 1797 – 1813.

Lartillot, O. and Toiviainen, P., 2007: 'MIR in Matlab (II): A toolbox for musical feature extraction from audio', in International Conference on Music Information Retrieval.

Laukka, P., Juslin, P., and Bresin, R., 2005: 'A dimensional approach to vocal expression of emotion', Cognition and Emotion, Vol. 19, No. 5, pp. 633 – 653.

Martin, C., 2017: Timbral Tagging: Towards the Development of a model for Metallic-natured sounds, Undergraduate technical project, Institute of Sound Recording, University of Surrey, Guildford, Surrey, UK.

McFee, B. et al., 2015: 'librosa: Audio and music signal analysis in python', in Proceedings of the 14th python in science conference.

Omori, H., 2011: Determining the nature of the difference between perceived brightness and brightness predicted by spectral centroid, Undergraduate technical project, Institute of Sound Recording, University of Surrey, Guildford, Surrey, UK.

Pearce, A., Brookes, T., and Mason, R., 2016: 'Deliverable D5.1: hierarchical ontology of timbral semantic descriptors', technical report, available: http://www.audiocommons.org/materials/

Pearce, A., 2017: Perceived differences between microphones, Ph.D. thesis, University of Surrey.

Peeters, G., 2004: A large set of audio features for sound description (similarity and classification) in the CUIDADO project, Tech. rep., Institut de recherche et coordination acoustique / musique.

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

Poirson, E., Petiot, J., and Gilbert, J., 2005: 'Study of the brightness of trumpet tones', The Journal of the Acoustical Society of America, Vol. 118, No. 4, pp. 2656–2666.

Prego, T., Lima, A., Netto, S., Lee, B., Said, A., Schafer, R., and Kalker T., 2012: 'A blind algorithm for reverberation-time estimation using subband decomposition of speech signals', The Journal of the Acoustical Society of America, Vol. 131, No. 4, pp. 2811–2816.

Prego, T., Lima, A., Zambrano-Lopez, R., and Netto, S., 2015: 'Blind estimators for reverberation time and direct-to-reverberant ratio using subband speech decomposition', in IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, October 18-21, New Paltz, NY.

Schubert, E. and Wolfe, J., 2006: 'Does timbral brightness scale with frequency and spectral centroid?', Acta Acoustica united with Acoustica, Vol. 92, pp. 820 – 825.

Schubert, E., Wolfe, J., and Tarnopolsky, A., 2004: 'Spectral centroid and timbre in complex, multiple instrumental textures', in Proceedings of the 8th International Conference on Music Perception and Cognition, pp. 654 – 657.

Solomon, L. 1959: 'Search for physical correlates of psychological dimensions of sounds', The Journal of the Acoustical Society of America, Vol. 31, No. 4, pp. 492–497.

Vassilakis, P., 2007: 'Sra: A web-based research tool for spectral and roughness analysis of sound signals', in Proceedings of 4th Sound Music Computing (SMC), pp. 319–325.

Williams, D., 2010: Towards a Timbre Morpher, PhD thesis, University of Surrey, Department of Music & Sound Recording.

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