A METHOD FOR AUTOMATIC WHOOSH SOUND DESCRIPTION

Eugene Cherny
Embedded Systems Laboratory
Abo Akademi University
Turku, Finland
eugene.cherny@abo.fi

Johan Lilius
Embedded Systems Laboratory
Abo Akademi University
Turku, Finland
johan.lilius@abo.fi

Dmitry Mouromtsev
ISST Laboratory,
ITMO University
St. Petersburg, Russia
mouromtsev@mail.ifmo.ru

ABSTRACT

Usually, a sound designer achieves artistic goals by editing and processing the pre-recorded sound samples. To assist navigation in the vast amount of sounds, the sound metadata is used: it provides small free-form textual descriptions of the sound file content. One can search through the keywords or phrases in the metadata to find a group of sounds that can be suitable for a task. Unfortunately, the relativity of the sound design terms complicate the search, making the search process tedious, prone to errors and by no means supportive of the creative flow. Another way to approach the sound search problem is to use sound analysis. In this paper we present a simple method for analyzing the temporal evolution of the "whoosh" sound, based on the per-band piecewise linear function approximation of the sound envelope signal. The method uses spectral centroid and fuzzy membership functions to estimate a degree to which the sound energy moves upwards or downwards in the frequency domain along the audio file. We evaluated the method on a generated dataset, consisting of white noise recordings processed with different variations of modulated band-pass filters. The method was able to correctly identify the centroid movement directions in 77% sounds from a synthetic dataset.

1. INTRODUCTION

There are two different approaches to sound design according to how a sound is generated and used [1]. A sample-oriented approach is based on the processing of the recorded sounds, which are then arranged according to a moving picture content. A procedural approach implies building up signal synthesis and processing chains and making them reactive to happenings on the screen. But the complexity of implementing DSP algorithms for artistic tasks made this approach rarely used in the industry, hence the sample-oriented method is used most in sound design. In this case the design process starts with searching for sounds in libraries.

To facilitate search and decrease the time spent searching, library manufacturers provide textual annotations for sounds that could be written in filenames, in PDF files, or encoded in spreadsheet for integration with sound metadata management software. In either case, sound search is basically a keyword-search in a corpus of text. This format makes it almost impossible to annotate many aspects of sounds, as the annotations will grow too big to be manageable by a human. So, the creators of sound libraries provide annotations that are short and concise. The quality and richness of metadata varies from company to company, but mostly all provide keywords for common sound categories, representing purpose, materials, actions and other high-level sound-related concepts.

The keywords may have synonyms, change meaning depending on a context (polysemy), or even be interpreted differently by different people (relativity) [2, 3]. These aspects lead to a poor user experience in sound search applications, when the search yields a lot of results, and a designer would need to listen through many sounds before even starting doing the creative work [4]. To narrow down the search, an experienced professional may want to exclude specific libraries from search or use boolean expressions to include synonyms, but all these actions are far from the creative work.

One way to approach this problem could be augmenting the keyword-based search with audio analysis algorithms to estimate some sound parameters that are pertinent to the workflow. There are a number of concepts in sound design which have more or less unambiguous meanings that many people agree on (like whooshes, risers, hits or pads to name a few), and we can model these sounds without doing a substantial amount of knowledge elicitation work. These models can be used in the search context to filter out the sounds which do not fit into the model parameters a user is interested in.

Of course, a comprehensive method to address the problem would include many of such models, providing a sound designer with a diverse set of content-based search filters. But in this paper we will only consider a model of a whoosh sound, as it has a relatively straightforward definition and is widely used in the production. The AudioSet ontology [5] defines whoosh as "a sibilant sound caused by a rigid object sweeping through the air." In sound design this term is usually interpreted in a more generic way, for example as "movements of air sounds that are often used as transitions, or to create the aural illusion of movement" [6]. This movement is an essential quality of a whoosh sound and can be expressed as volume or spectral movements. The definitive characteristics of this movement are relatively slow attack and decay, a prominent climax point, and often a noticeable spectral change from the higher frequencies down to the lower ones or the opposite way around. Sound of this class are widely used, e.g. in game menu navigation, trailer scene transitions, for sounds of space ships passing by, and so forth.

In this paper we describe a whoosh model and a set of audio analysis algorithms to fit an arbitrary sound file into this model. The model is used to estimate the spectral movement descriptor quantifying the degree of the whoosh transition up or down along the spectrum. The model is based on splitting the sound into frequency bands, and approximating the volume envelopes in each band as piecewise linear function. This representation is similar to the multi-stage envelopes used in sound synthesizers[1], so we can easily extract attack and decay values from it. We test our method on a synthetic dataset consisting of different variations of white noise processed with a modulated band-pass filter.

The paper is organized as follows. Section 2 provides an overview of the related work. Section 3 describes the whoosh model

[1] Also known as breakpoint function.

and its centroid metric. In Section 4 we describe procedures to fit an arbitrary sound to the model. Section 5 describes the evaluation procedures, and Section 6 reports the evaluation results. Lastly, in Section 7, we discuss the results, and then conclude the paper in Section 8 by summarizing contributions and possible applications.

2. RELATED WORK

A good introduction to the sound retrieval problem can be found in [7]. This work explored what words subjects used to describe sound, and grouped verbal descriptions into three major classes: *sound itself* (onomatopoeia, acoustic cues), *sounding situation* (what, how, where, etc.) and *sound impression* (adjectives, figurative words, etc.). These classes provide different conceptual tools to organize sound.

The *sound impression* class deals with the interpretation of how words connected to sound [8, 9, 10]. This class consists of descriptions that are usually used in informal communication, and they often do not have established interpretations. Thus, building a recognition system for them would require the elicitation of strict definitions (like the authors of [11] and [12] did).

The *sounding situation* class contains descriptions of setting in which a sound production occurred. They may include a sound source (a bird, a car), a sounding process (twittering, rumbling), a place (a forest, a garage), a time of production (morning), etc. Essentially, commercial sound libraries mostly annotated with the concepts of this class, but adding all word variations and synonyms to enrich the metadata would sacrifice readability and search simplicity. Moreover, treating linguistic relativity may require a more deep understanding of what is actually recorded in a sound. For example, a *long* hit sound in the *user interface library* could be much shorter than a *short* hit from the *trailer library*. One could address this issue by adding a *trailer* keyword to the annotation, but this is just the one example of many possible whoosh usages, and covering up them all may just be impossible to achieve with limited human resources. To some extent these problems can be addressed with natural language processing or knowledge elicitation [3, 13, 14], but treating the relativity problem with these methods would require a substantial amount of knowledge engineering to categorize and organize sound concepts. There are ontologies that provide some structured information about audio-related concepts [5, 15, 16, 17] and with some adaptations they can assist in the sound search. For example, in [12] authors describe the implementation of the timbre attribute prediction software that uses concepts from the ontology provided in [15].

The *sound itself* class describes acoustic features of the sound "as one hears it." It includes sound representation models (music theory, spectromorphology, onomatopoeia, etc.). For example, in [18] authors explore the relevance of sound vocalizations compared to the actual sound. Interestingly, in this paper authors connected vocalizations with the morphological profiles of the sound, based on Schaeffer's spectromorphology concepts. There is a substantial work done on automatic sound morphological profile recognition [19, 20, 21], but real sound designers rarely have to deal with the morphological concepts.

3. WHOOSH MODEL

According to the previous section, our work is related to the modelling sound properties (the *sound itself* class), as we want to quantify the parameters of some known class of sounds (whooshes). As was discussed in the Introduction, the main property of a whoosh sound is the *movement* which can be temporal, spectral or both at the same time. This movement is characterized by the prominent peak somewhere in the middle of the sound: the intensity rises to this peak from silence and decays down after passing it. A spectral change can also happen together with the intensity, often with prominent transitions from higher to lower part of the spectrum or other way around. We build our model with the proposition that two movements together define the whoosh sound.

We continue with the following assumptions:

1. The sound intensity movement of the whoosh sound can be described using either an *attack-decay* (AD) or an *attack-hold-decay* (AHD) envelopes. This comes directly from the proposition we mentioned just above.
2. The model will be used in the search context, where a user already knows what sounds are whooshes, and only wants to filter out some of them according to the settings. Thus, a recognition between whoosh and non-whoosh sounds is not needed.

The purpose of the model is to describe the temporal envelope and the spectral movement of a whoosh sound. The model consists of a simplified multi-band representation of the sound intensity envelope and a movement metric, which quantifies a degree to which the frequency content moved up or down in the spectrum.

The *attack-decay* envelope is an ideal representation of a whoosh movement, but in practice, it is often hard to define one single peak, especially in the long sounds. In this case an *attack-hold-decay* envelope could be a better fit. The AHD envelope can be modeled as a piecewise linear function defined somewhere at a sound with length L:

$$E_i(x) = \begin{cases} a_1 x + b_1, \text{ if } x \in [x_0^i, x_1^i) \\ a_2 x + b_2, \text{ if } x \in [x_1^i, x_2^i) \\ a_3 x + b_3, \text{ if } x \in [x_2^i, x_3^i] \\ 0, \text{ otherwise} \end{cases}, x \in [0, L] \quad (1)$$

where x is an arbitrary time point in a sound file, x_0^i and x_3^i are the beginning and the end of the envelope, and x_1^i and x_2^i are the breaking points. The AD envelope model is a special case of AHD, where $x_1^i = x_2^i$. A sound can contain several non-overlapping sound envelopes, and i denotes an index of the envelope in the sound.

For each sound we define two fuzzy membership functions for the "beginning" and the "end" linguistic variables:

$$m_{beg}(x) = \frac{x}{L} \in [0, 1] \,, \quad (2)$$

$$m_{end}(x) = 1 - m_{beg}(x) \in [0, 1] \,, \quad (3)$$

which results can be interpreted as "to what extent the time point x is located in the beginning or in the end of a sound." We use these functions to weigh an arbitrary envelope value:

$$V_l^i(x) = m_l(x) E_i(x), \quad (4)$$

where l denotes a linguistic variable, which can be either "beg" or "end".

The breaking points x_1^i and x_2^i from the Eq. 1 are considered rs representing points, where the most sound energy is located.

We are using them to calculate a *cumulative peak value* for the linguistic variable l:

$$p_l = \sum_i (V_l^i(x_1^i) + V_l^i(x_2^i)), \tag{5}$$

which value can be interpreted as "how much of sound energy lies in the beginning or in the end of a sound."

We use the piecewise functions (Eq. 1) to compress envelope signals in the each band of a sound, passed though a crossover. With a set of models in each frequency band, we can estimate the centroid of the peak values for each linguistic variable, adapting the standard spectral centroid formula [22]:

$$C_l = \frac{\sum_b b p_l^b}{\sum_b p_l^b}, \tag{6}$$

where b is a band index, and p_l^b is a cumulative peak value of the band b for the linguistic variable l.

And finally, the centroid descriptor is a difference between centroids of two linguistic variables:

$$C_d = C_{end} - C_{beg} \tag{7}$$

C_d estimates how the centroid changes over time, with negative values suggesting a decrease, and positive values — an increase.

4. FITTING A SOUND TO THE MODEL

To construct the model we need to find values for all non-input variables in the Eq. 1. Our strategy here is to first find the boundaries of the envelope (points x_0^i and x_3^i), and then to find breaking points x_1^i and x_2^i. Knowing envelope values at these points, we can calculate the linear function coefficients a and b in each piece by solving a trivial linear equation system. This approach is similar to the one used in [19], where the maximum sound envelope values are being connected to the both, beginning and end of the sound, thus forming a two-segment AD model. In our method we construct the AHD model which can often be more realistic approximation. The rest of this section will explain, how exactly all these steps has been performed.

In sound libraries a single audio recording can contain several variations of the same sound separated by silence. This is a common practice, e.g. for the sounds of weapon shooting or footsteps, etc. So, we start with looking for the long silence segments and use them as separators to split recordings into multiple sounds.

The separated sounds are then split into frequency bands with a crossover filter bank, and the RMS envelope signals are estimated for each band. Based on these signals we split each band into regions separated by silence. A simple threshold processing is used to find the initial set of regions, that may get merged if they are close to each other.

The piecewise models are then built for each resulting region. The beginning and the end points (x_0^i and x_3^i) are defined by regions' bounds, thus we need to find and connect climax points to create the linear representation. We do this by essentially the same algorithm as we used in the silence-split operation above, but with a higher threshold value relative to the maximum envelope value in the region. The new operation will yield a number of "high-energy" regions, and their maximum values are used as climax points for the multi-segment model. However, in many cases, the resulted model will contain a large number of segments due to noise and irregularities in sound. To overcome this, we discard all points between the first and the last climax-points. The two peak points left will be used as x_1^i and x_2^i, thus creating the AHD model. If there is only one peak point found after performing this operation, then we can create the AD model, where ($x_1^i = x_2^i$). The AD model can then be simplified by removing the middle piece from the piecewise model in the Eq. 1.

Having all breakpoints of the piecewise model found, determining the a and b coefficients for each linear function is as trivial, as finding the equation of the line given two points.

5. EVALUATION

We use the following method parameters in the evaluation:

- The sample rate is set to 44100 Hz.
- The RMS window size for the envelope signal estimation is set to 20 ms.
- For the separating sounds in the sound file we set the non-silence split threshold T_{ss} to 0.001 (-60dB), and non-silence merge threshold T_{sm} to 500 ms. This means, that all non-silence regions with value higher than 0.001 will be merged together if the silence gap between them is shorter than half a second.
- The crossover is implemented using a bank of 4-order Butterworth filters. It can have a different number of frequency bands, but the split frequencies should to be spread evenly on the equivalent rectangular bandwidth scale (ERB) [23]. This provides a perceptual weighting for comparing signal energies between different frequency bands.
- To find the envelope bounds in frequency bands, the T_{ss} value is set to 75 ms.
- To find the high energy regions in envelopes, the peak split threshold (T_{ps}) is set to 80% of the maximum envelope signal value between the bounds. The peak merge threshold T_{pm} is set to 5 ms. I.e. the high energy regions with the envelope value more than 80% of the maximum will be merged together into one if the gap between them is shorter than 5 ms. The individual peaks will be identified as the maximum value in the each region.

First, we demonstrate the method for a simple signal by feeding in a 1-second sine wave with frequency sweeping from 50 to 10000 Hz into it (Fig. 1).

We also test the method on a synthetic dataset generated with Csound [24]. It consists of 1-second sound files with the variations of the white noise processed by a modulated band-pass filter. The modulation linearly changes the center frequency and the bandwidth of the filter from cf_1 to cf_2, and from bw_1 to bw_2 respectively. We choose the set of center frequencies and bandwidths for modulation as follows:

- Create an list of tuples (f_l, f_h) with all possible pairwise combinations of the set $\{0, 100 \ldots 20000\}$, so that $f_l < f_h$. f_l and f_h are the bottom and the top frequencies of the pass-band respectively.
- Combine the resulted tuples into a list of $((f_{l1}, f_{h1}), (f_{l2}, f_{h2}))$. Each element of the list represents a pair of initial and target modulation parameters. The parameters are transformed into the center frequency and bandwidth pairs as follows: $bw_i = f_{hi} - f_{li}$, $cf_i = f_{li} + 0.5bw_i$, where $i \in \{1, 2\}$.

- To reduce the number of samples, we first leave out the tuples where the absolute difference between the initial and the target center frequencies are more than 5000[2], and then sampling every 200th sample from what is left.

This procedure leaves us with 365330 variations of the initial and the target filter frequency-bandwidth pairs. We pass these values into Csound [24] to generate the sound files[3]. Our method is evaluated using different number of frequency bands. The goal of this experiment is to find, in which situations our method fails to correctly predict the spectral direction of whoosh sounds.

6. RESULTS

Fig. 1 shows how the method works on a trivial example: a sine wave with frequency rising up linearly from 1000 Hz to 10000 Hz. We see how the algorithm fits the linear model to the envelope signals in each band. The red vertical lines depict peak positions. Values at the peaks are weighted for each linguistic variable as follows (Eq. 4):

$V^1_{beg} \approx 0.002 \quad V^1_{end} \approx 0$
$V^2_{beg} \approx 0.354 \quad V^2_{end} \approx 00$
$V^3_{beg} \approx 0.314 \quad V^3_{end} \approx 0.04$
$V^4_{beg} \approx 0.007 \quad V^4_{end} \approx 0.347$

Which yields the following centroids after applying Eq. 6:

$C_{beg} \approx 1.445 \quad C_{end} \approx 2.893$

The "end" centroid is higher, suggesting the sound's frequency content is moving up in the spectrum.

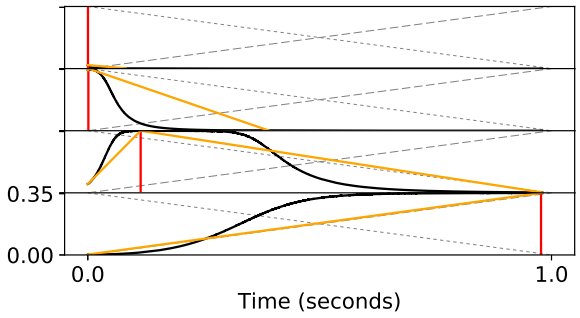
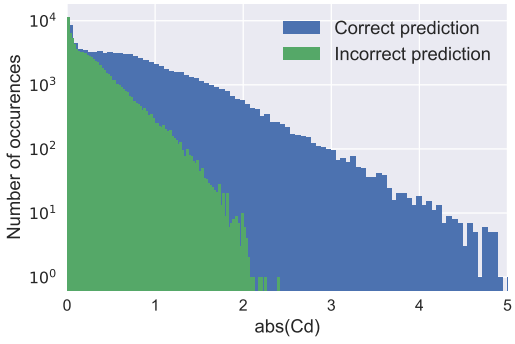


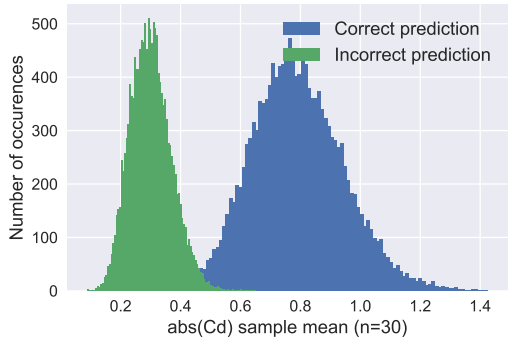
Figure 1: A sine sweep analyzed with the described method. The frequency goes up linearly from 1000 to 10000 Hz. The figure presents four graphs with signal envelopes and linear functions for each frequency band (the lowest band is at the top row). Split frequencies are 422, 1487 and 4596 Hz. Y-axis in each graph is the envelope amplitude. All graphs have identical scales, so only the bottom one's scale is annotated. Black solid lines are the envelope signals. Yellow lines are the piecewise models (Eq. 1) fitted to the envelopes. Red horizontal lines depict the envelope's peak value position. Dotted and dashed gray lines are fuzzy membership functions for the "beginning" and the "end" linguistic variables respectively. The signal in the first band is weak compared to the other bands, but it passed the threshold defined by T_{ss}.

[2]We decided not to include samples with such prominent spectral transition, as it may be easy for the algorithm to identify the whoosh direction in such cases.

[3]The second-order Butterworth filter was used, see the *butterbp* opcode, URL: `http://csound.github.io/docs/manual/butterbp.html`



(a)



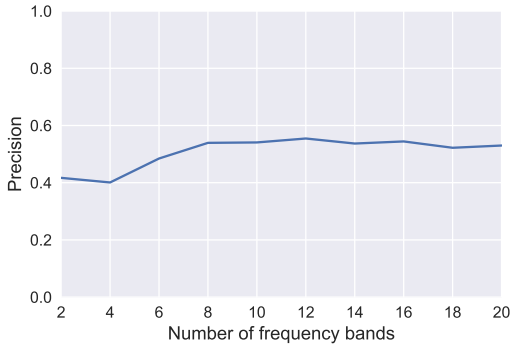
(b)

Figure 2: a) Distribution of the absolute centroid descriptor ($abs(C_d)$) for correct and incorrect predictions. For this histogram the correct predictions data has been sampled to be comparable with the amount of incorrect predictions data available. b) Sampling distribution of the mean of $abs(C_d)$ with sample size equal to 30.

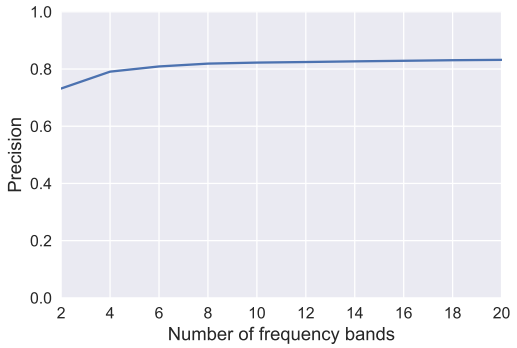
In the second experiment we test the method on a set of synthetically prepared sounds, as described in the previous section. From the 365330 variations of filter frequency sweeps of the white noise signal, the centroid movement direction of 282815 sounds (approx. 77%) were identified correctly.

Fig. 2-a shows the distribution of the absolute centroid descriptor for both correct and incorrect predictions. We see that on average, the prediction errors happen more often when the centroid movement is low (Fig. 2-b). There is a slight positive relationship between the absolute centroid descriptor and the prediction correctness ($r \approx 0.319$). Approximately the same correlation exists between the difference of the start and the end center frequencies of the band-bass filter modulation in generated sounds ($r \approx 0.317$).

We test how the algorithm performs when evaluated with different number of frequency bands. The obvious outcome of using less bands is lesser frequency domain precision and inability to recognize whoosh direction when spectral variations of a sound are completely covered by a single band. So, the ideal method may fail to recognize the direction of some whooshes when evaluated with a few bands, but it will eventually succeed after increasing the



(a) "Unstable" group



(b) "Stable" group

Figure 3: A precision of whoosh direction discrimination in two groups of sounds.

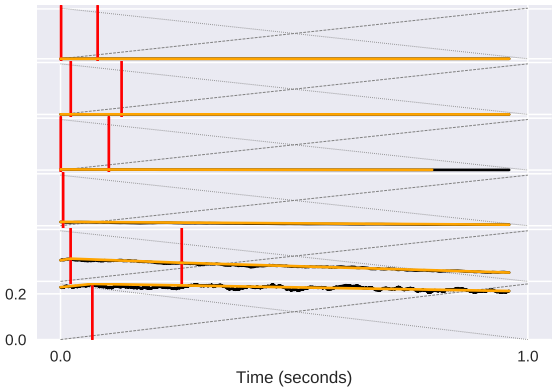
number of bands. But test results are not that simple. We observed three group of sounds:

1. Sounds where the estimated whoosh direction is the same for all numbers of frequency bands.
2. Sounds where the direction estimation is incorrect for a low number of bands, but it improves after increasing their number and does not regress anymore.
3. Sounds, where the direction estimation varies for different number of frequency bands without any apparent pattern (e.g. it estimates correctly for 4 and 6 bands, then regresses at 10 bands, then improves for 18 bands, etc.).

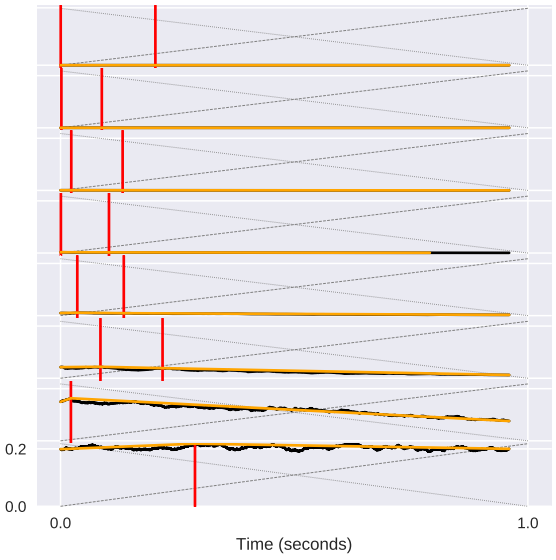
For convenience we call first two groups of sounds *stable* and the third one *unstable*. The *unstable* subset contains 9506 sounds (including both, correct and incorrect predictions), which is 26% of the test data. Fig. 3 compares the bandwise method precision in the two subsets. We see that the precisions are approximately 0.58 and 0.81 for the *unstable* and *stable* respectively. The precision grow as the number of frequency bands increase, and stabilize at approximately 8 bands in both groups.

7. DISCUSSION

In this section we will briefly discuss the method performance analysis results from the previous section.



(a) 6 bands, incorrect pred., $C_{beg} \approx 4.46$, $C_{end} \approx 4.37$



(b) 8 bands, correct pred., $C_{beg} \approx 6.06$, $C_{end} \approx 6.57$

Figure 4: An example of noisy peak detection. The graph axes are the same as in the Fig. 1. The noise sound ($cf_1 = 9292, bw_1 9697, cf_2 = 11413, bw_2 = 7071$) has been analyzed with the method two times: with 6 and 8 bands. Note the peak position in the bottom bands: although the envelope signal is essentially the same, the estimated peak position vary significantly due to noise in the signal. This lead to the incorrect direction prediction in the case (a).

First, there is an evident correlation between absolute C_d and the method precision. This descriptor is a difference between centroid values weighted for the two linguistic variables representing a temporal placement of an event. This suggests, that the method may have worse performance for the sounds with subtle spectral centroid movements. Currently, the implementation lacks notion of the "still" category, which contains sounds that does not perceivably go "up" nor "down", and many incorrect predictions might go into this class. But the implementation of this class requires to conduct a qualitative research determining, what is a perceptually significant whoosh movement. This is an important question for the content-based sound search, but it is out of scope of this

particular paper.

Second, the data clearly shows shows that in general, we can increase the number of frequency bands to get a higher precision, but at some point, adding more bands will stop affecting it. This suggests that some bands can be redundant to represent a particular sound. Thus, the piecewise representation can be improved by introducing an adaptive analysis algorithm, that would change the number of bands according to the sound's spectral content.

Lastly, the inconsistent method performance for different number of frequency bands suggests the peak-detection sensitivity noise. The white noise has equal intensity at different frequencies and relatively steady volume envelope. Filtering the noise with a bandpass filter will increase irregularities in the envelope, thus providing noise for our peak-detection algorithm. Fig. 4 shows one sound processed by the method using 6 and 8 frequency bands. We see, that peak positions in the multi-segment representation vary significantly, which affects the prediction. We can tackle the noise in both, the envelope signal by increasing its smoothness with filtering, as well as making more stable piecewise linear models by using more advanced fitting algorithm [25, 26]. Another way to improve the method robustness could be to use the integration of the product of membership and piecewise functions instead of peak values in V_l^i estimation (Eq. 5). This way the method will not be dependent on the peak detection errors caused by the sound irregularities.

Another point of improvement could be testing the method on real sounds, but we have found extremely little sounds on Freesound [27] that contain tags for both, the "whoosh" keyword and the direction. E.g. there are only 10 whooshes annotated with the tag "down". One option could be to annotate arbitrary whooshes by hand and validate the method, but many sounds are intended to be the "still" whooshes, so without proper differentiation of this class by the method the hand-annotation will be prone to errors. But manual annotation will could be prone to bias: as the method's creators, we may be annotating to make it pass the test. Thus, a third party should be involved in annotation process.

8. CONCLUSIONS

We presented a descriptor for describing spectral centroid movement in audio files, based on fuzzy membership functions and piecewise linear model representation of sound envelopes in different frequency bands. The method has been evaluated on a dataset consisting of 365330 synthetic test sounds. It predicted spectral centroid movement correctly in 77% of sounds in a dataset. We identified the probable cause of prediction errors as noisiness in the peak detection algorithm, and suggested ways to improve it.

The described method can be used in the sound search context to filter or sort whoosh sounds according to the spectral movement. The centroid descriptor C_d can not only be used for up-down whoosh differentiation, but also to estimate the degree of spectral movement in sounds, and to sort according to it. Also, knowing the break points positions, we can filter sounds based on the relative length of the attack and decay lengths. These filters can be implemented, for example, as user interface widget in the sound metadata management software to assist the search.

The source code used for this paper is available on-line: `https://github.com/ech2/papers`.

9. ACKNOWLEDGMENTS

This work was partially financially supported by the Government of the Russian Federation, Grant #074-U01. The *librosa* [28] Python library for sound analysis was used in this project.

10. REFERENCES

[1] Andy Farnell, *Designing sound*, MIT Press, 2010.

[2] Fabiano M Belém, Jussara M Almeida, and Marcos A Gonçalves, "A survey on tag recommendation methods," *Journal of the Association for Information Science and Technology*, vol. 68, no. 4, pp. 830–844, 2016.

[3] Frederic Font and Xavier Serra, "Analysis of the Folksonomy of Freesound," in *Proc. of the 2nd CompMusic Workshop*, Istanbul, Turkey, July 12, 2012, pp. 48–54.

[4] Eugene Cherny, "Email interview with Axel Rohrbach (BOOM Library GbR) about problems with metadata management in sound design," unpublished.

[5] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, USA, 2017.

[6] Tim Prebble, "SD101: Wooshes," Available at `http://www.musicofsound.co.nz/blog/wooshes-101`, 2007 (accessed on June 21, 2017).

[7] Sanae Wake and Toshiyuki Asahi, "Sound retrieval with intuitive verbal expressions," in *Proceedings of the 1998 International Conference on Auditory Display (ICAD'98)*, Swindon, UK, Nov. 01-04, 1998, pp. 30–30.

[8] Michel Bernays and Caroline Traube, "Expression of piano timbre: gestural control, perception and verbalization," in *Proceedings of CIM09: The 5th Conference on Interdisciplinary Musicology*, Paris, France, 2010.

[9] Graham Darke, "Assessment of timbre using verbal attributes," in *Proc. Conference on Interdisciplinary Musicology*, Montreal, Quebec, 2005.

[10] Asterios Zacharakis, Konstantinos Pastiadis, and Joshua D. Reiss, "An interlanguage unification of musical timbre: Bridging semantic, perceptual, and acoustic dimensions," *Music Perception: An Interdisciplinary Journal*, vol. 32, no. 4, pp. 394–412, 2015.

[11] Thomas Grill, "Constructing high-level perceptual audio descriptors for textural sounds," in *Proceedings of the 9th Sound and Music Computing Conference (SMC 2012)*, Copenhagen, Denmark, 2012, pp. 486–493.

[12] Andy Pearce, Tim Brookes, and Russell Mason, "First prototype of timbral characterisation tool for semantically annotating non-musical," Audio Commons project deliverable D5.2, Surrey, UK, 2017, available at `http://www.audiocommons.org/materials`.

[13] Frederic Font, Joan Serrà, and Xavier Serra, "Audio clip classification using social tags and the effect of tag expansion," in *AES 53rd International Conference on Semantic Audio*, London, UK, Jan. 26, 2014, pp. 157–165.

[14] Eugene Cherny, Johan Lilius, Johannes Brusila, Dmitry Mouromtsev, and Gleb Rogozinsky, "An approach for structuring sound sample libraries using ontology," in *Proc. International Conference on Knowledge Engineering and Semantic Web (KESW 2016)*, Prague, Czech Republic, Sept. 21-23, 2016, pp. 202–214.

[15] Andy Pearce, Tim Brookes, and Russell Mason, "Hierarchical ontology of timbral semantic descriptors," Audio Commons project deliverable D5.1, Surrey, UK, 2016, available at `http://www.audiocommons.org/materials`.

[16] Tohomiro Nakatani and Hiroshi G. Okuno, "Sound ontology for computational auditory scene analysis," in *Proceeding for the Fifteenth national Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, USA, July 26-30, 1998, pp. 1004–1010.

[17] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al., "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[18] Guillaume Lemaitre, Olivier Houix, Frédéric Voisin, Nicolas Misdariis, and Patrick Susini, "Vocal imitations of non-vocal sounds," *PLoS ONE*, vol. 11, no. 12, pp. 1–28, Dec., 2016.

[19] Geoffroy G. Peeters and Emmanuel Deruty, "Automatic morphological description of sounds," *The Journal of the Acoustical Society of America*, vol. 123, no. 5, pp. 3801–3801, 2008.

[20] Geoffroy Peeters and Emmanuel Deruty, "Sound indexing using morphological description," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 675–687, 2010.

[21] Lasse Thoresen and Andreas Hedman, "Spectromorphological analysis of sound objects: an adaptation of Pierre Schaeffer's typomorphology," *Organised Sound*, vol. 12, no. 2, pp. 129, 2007.

[22] Wikipedia, "Spectral centroid," accessed June 21, 2017, Available at `https://en.wikipedia.org/wiki/Spectral_centroid`.

[23] Julius O Smith and Jonathan S Abel, "Bark and ERB bilinear transforms," *IEEE Transactions on speech and Audio Processing*, vol. 7, no. 6, pp. 697–708, 1999.

[24] Victor Lazzarini, Steven Yi, John ffitch, Joachim Heintz, Øyvind Brandtsegg, and Iain McCurdy, *Csound - A Sound and Music Computing System*, Springer, 2016.

[25] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE International Conference on Data Mining (ICDM 2001)*, 2001, pp. 289–296.

[26] Vito MR Muggeo, "Segmented: an R package to fit regression models with broken-line relationships," *R news*, vol. 8, no. 1, pp. 20–25, 2008.

[27] Frederic Font, Gerard Roma, and Xavier Serra, "Freesound technical demo," in *Proceedings of the 21st ACM International Conference on Multimedia*, New York, NY, USA, 2013, MM '13, pp. 411–412, ACM.

[28] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee, "librosa 0.5.0," Feburary 2017, Available at `https://doi.org/10.5281/zenodo.293021`.