A SIMILARITY SEARCH ALGORITHM FOR DRUM SOUNDS

Tobias Gutenkunst *Christoph Hold* *Fabian Seipel*

TU Berlin
Audiokommunikation und -Technologie
SoSe 2016

1. ABSTRACT

Common research in instrument recognition often concentrates on classifying instruments. This paper develops a method for finding perceptually similar alternatives to a drum sample within a dataset, but without the need of previous categorization or labeling. The proposed similarity search algorithm analyzes and weights both temporal and spectral features. A listening test with 18 subjects assesses its performance. The results confirms that the search algorithm produces perceptually similar outcomes in most cases.

2. INTRODUCTION

Most research in the field of instrument recognition has been carried out to develop pure classification algorithms which are focused on the task of labeling audio content to distinct and pre-defined instrument categories. The similarity approach presented in this paper however, is based on comparisons of drum sounds. Results are therefore bound to a definition of similarity and the combination of selected audio material in the dataset. The same input sound file will consequently provide different outcomes in case of alternate sample collections. As opposed to classification techniques, this approach does not rely on the existence of a given ground truth but needs to be evaluated by subjective listening tests to verify results in terms of the notion of similarity. Possible applications for this algorithm may include features add-ons in audio production environments to facilitate the workflow and enhance user experience.

Even though classification approaches may differ in application purpose, outcome and evaluation, the proceedings of analysis rely on the same principles to characterize sound structures. Sets for audio descriptors are mainly differentiated in temporal and spectral features [1]. Christophersen et al. [2] have proposed modeling the temporal structure via ADSR envelopes to built time-domain features. In another study, Herrera et al. [3] have examined feature selections in the case of drum sounds by first dividing the temporal envelope in parts of attack and decay state for separate spectral analysis additional to relative energy descriptors and mel frequency cepstrum coefficients. The latter has been investigated in depth by Eronen and Klapuri [4] for general musical instrument recognition, especially by evaluating various statistical measures of the same feature in different temporal states. Eronen [5] further compared given features on different frequency scales and representations for individual instruments as well as instrument families. Hierarchical structures have also been proposed as a method for successively divide instruments into groups [4] [6].

Machine learning approaches have established as pattern recognition techniques for instrument identification, whereas the individual assortment tasks need to be selected carefully from a plethora of possibilities [1]. K-Nearest Neighbours algorithms have been proven as one reliable method for this purpose which can also perform in the case of a non-existent ground truth [6] [7].

3. METHODS

To perform the similarity search, a feature selection was extracted from both the given dataset as well as for the query in a first step. Regarding the comparison task, above mentioned descriptors were combined in an n-dimensional weighted feature space and described by a distance measure to calculate results. This approach is inspired by the k-Nearest-Neighbours technique, but instead of predicting class labels on ground truth data by a defined number of k, the algorithm delivers above

mentioned distances solely on minimizing distances from sample features in query compared to the sample features from the database.

3.1. Dataset

The final set of audio to analyze, taken from the Native Instruments Maschine Expansion Libraries, consisted of each 500 snare, kick and hihat waveform sound files with nearly normal distribution in terms of file size in between 10 and 100kB and and length of 500 to 20000 samples at sampling frequency 44100 Hz. Individual sound files were also chosen with regard to temporal shape, timbral characteristics, playing techniques, processing effects, sound generation and recording context (e.g. acoustic, electronic or digital) to guarantee a high amount of diversity throughout the database. In contrast to this, some very slight variations of sounds, e.g. versions of 909 analog drum sounds, were intentionally included in the dataset in regard to test for known perceptual similarities among those. During the training phase the database was reduced to 50 samples each for performance reasons.

3.2. Feature Selection

As the similarity search should represent human sound perception and semantic meaning, the algorithm input needs to consist of a meaningful and representative feature extraction from the sound samples. Since the principal idea is to provide the user with *perceptually* similar alternatives, the choice is based on features that reflect certain perceptual attributes. These should cover the description of both the temporal domain and also the spectral domain of every sound sample as these two domains represent each sound in an interpretable, easy to handle, yet comprehensive way. In a user feedback loop including the following search algorithm, almost every feature from Lerch [8] was subjectively examined, whether it improved the search performance and suited the described concept. Ultimately, the choice came down to *Attack* and *Decay* for the temporal domain as well as *Spectral Centroid* for first moment, *Spectral Spread* for second moment and *Spectral Roll-off* plus *Spectral Skewness* for third moment spectral description. Higher moments (kurtosis) had no significant positive influence. Interestingly, the perceptually motivated Mel Frequency Cepstral Coefficients (MFCC) proved not well suited.
Both temporal values were calculated by detecting a certain threshold from the smoothed root mean square curve of the audio signals as well as the cumulated energy over time. Those thresholds were found iteratively as the first maximum of the smoothed rms curve for the attack state and for the decay state as the last sample point when the rms curve has dropped below one percent of its maximum values, given the fact that the rms function has already reached all its peaks, which was defined by a minimum of 90 percent of its maximum cumulated energy over time.
Figure 1 demonstrates the feature ability to distinguish between the sample categories, although the aim is not to categorize samples, but to search for perceptually similar samples, thus standard accuracy measures cannot be applied. For example, a high pitched analog snare drum may sound very similar to a hihat, but their labels are different.

3.3. Search Algorithm

The search algorithm's task is finding similar sounding samples. It is supposed to find the closest k alternatives around an input sample, wherefore a k-Nearest-Neighbor (KNN) framework was applied. Since the KNN searches is based on distances to the input, an appropriate, perception reflecting measure is needed. Perceived distance is neither Euclidean, nor any other basic variation of it.
Standardized Euclidean distance from one to another sample feature value is given by:

$$D_s = \frac{1}{s} \cdot \sqrt{x_{test}^2 - x_{database}^2} \tag{1}$$

whereby the previously applied Z-Normalization sets the standard deviation to $s = 1$ by definition. Thus the freely assignable parameter s can also scale the distance for this dimension, which results in an impact weighting of the corresponding feature.
The algorithm also includes a method to customize those weights for every user individually. First, a default set of weighting coefficients is provided. In order to develop those default values, a test set including very little variations of the same sound was processed (e.g. variations of analog 909 bass drums/ snares/ hihats). Assuming these samples are perceptually very similar, all feature weights were adapted to find the variations among each other. In a second step, every user is then allowed to manipulate these weights according to his or her expectations and demands for similarity to individualize the algorithm. This

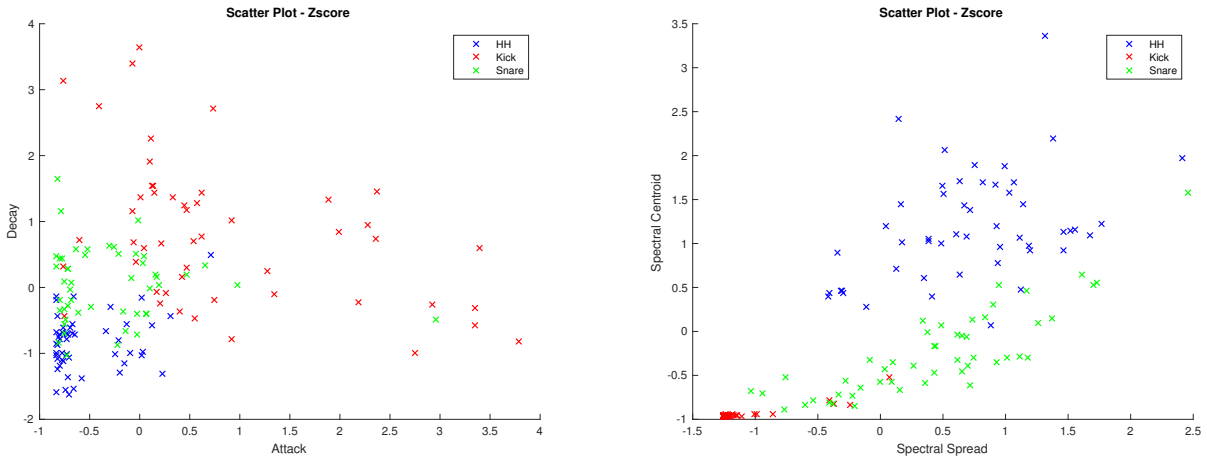

Fig. 1: Z-Normalized time based features "Attack" and "Decay", Z-Normalized spectral features "Spectral Spread" and "Spectral Centroid"

process included a very challenging small test dataset, since good results on this set promises even better results for more samples, since the feature space usually gets more compact and distance measures should perform better. It was further tested if the algorithm is able to find a duplicate of the input. A subsequent Principal Component Analysis (PCA) did not improve the results; On the contrary, it detaches the associated perceptually related dimensions, making it really difficult to adjust the weighting factors manually.

3.4. Evaluation

Since there is no objective ground truth for perceptive similarity of audio samples, evaluation of the algorithm took place via listening tests.

Those used the above mentioned drum sample dataseta and consisted of 10 different subsequent test runs in which subjects were asked to listen to four different samples. One of those was a randomly chosen input sample, which they had to compare to the other three samples. The second one was the result from the similarity search algorithm for the input sample. Both remaining two samples were randomly chosen from the remaining dataset samples under the condition that similarity according to the algorithm calculation is less than 80% in comparison to the input sample. This contraint was implemented to prevent cases where the listener was presented with multiple samples which were all classified as similar. Finally, subjects had to choose, which of the last three audio samples was most similar to the input stimulus. 18 participants took part in this study, adding up to a total of 180 comparisons.

Our hypothesis is that the algorithm finds perceptually similar alternatives to the input sample in the given dataset including 1500 drum sounds altogether. Additionally, the code should execute in a time frame which allows to implement the algorithm in user interactive environments.

4. RESULTS

4.1. Listening Test Results

As a result, listeners rated the sample which was chosen by the algorithm to be most similar with a mean of $M = 77.2\%$ times, compared to the guessing probability of 33.3%. Figure 2 shows the histogram of mean positive test outcome for all subjects, the highest test score being 100%, the lowest 60%.

A χ^2-test against random probability of $p_r = \frac{1}{3}$ returned a significant $p(\chi^2) < 0.05$, at one degree of freedom. Thus the null hypothesis—the observations are caused by randomness—can be rejected.

4.2. Execution Speed

Considering execution speed of the algorithm, there are two different relevant calculation steps. The initial feature extraction, also denoted as training phase, took about one minute per 100 samples, while the actual inquiry for the similarity search takes about 0.02 seconds. Calculating the dataset features scales linearly with the amount

of samples, consequently the feature extraction in this evaluation took 15 minutes for 1500 samples with the machine using only a single thread.

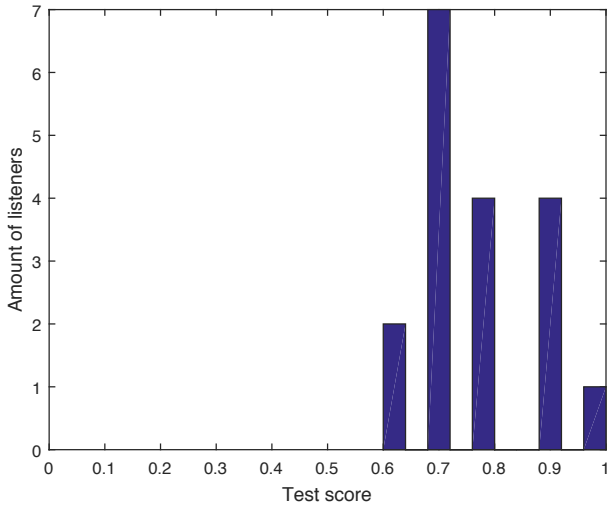

Fig. 2: Histogram showing mean test outcome of every listening test participant

5. DISCUSSION

The results from the listening test validate the algorithm's ability to find perceptually similar samples in most cases. This is further supported by the fact that searches for a sample which has multiple variations in the dataset also returned the belonging samples. For example, a 909 base drum input returned other 909 base drum samples.
In some cases, however, the output is not perceptually similar to the input sample. One reason for this might be that the input sample constitutes an outlier within the dataset. In this case even the best match is not perceived as being similar yet. Thus the similarity search works better for bigger and uniformly dense distributed datasets than for small varied and therefore scattered ones.
While the initial feature extraction takes about one minute per 100 samples, the search algorithm itself calculates almost immediately. Hence, the execution of the similarity search is fast enough for user interaction in a production environment. The feature extraction speed is not as important, because this needs to be carried out only once while importing a sample library.

5.1. Outlook

Future work might include implementing a maximum distance for the nearest neighbor in the feature space. This would help identifying outliers as such, in order to prevent perceptually wrong output samples.
The main goal of this work is to use the search algorithm inside an audio production environment. In order to do so, translating and compiling the source code, possibly including a graphic user interface, would be necesarry to implement the similarity search as an add on feature inside a regular audio production work flow.
Furthermore this may include the option of realizing a dynamic feature weighting by user input to improve and especially personalize the overall system. A potential user could for example control the impact of each feature on the outcome via interface possibilities and in this way change the outcome to a more spectral or temporal similar result.

6. SUMMARY

The aim of this paper was to develop and evaluate a similarity search algorithm for finding perceptually similar samples inside a dataset compared to an input sample. The available dataset consisted of 1500 drum sounds (each 500 bass drums, snares and hi-hats). A weighted distance measure approach inspired by kNN techniques proved best results for finding similar samples using both temporal (*Attack, Decay*) and spectral features (*Centroid, Spread, Roll-off and Skewness*). Listening tests were conducted to evaluate the functionality and perceptive reliability of the algorithm. It turned out, the algorithm does find perceptually similar samples in most cases. Code execution speed is fast enough to possibility implement the algorithm in audio production environments.

7. REFERENCES

[1] Perfecto Herrera-Boyer, Geoffroy Peeters, and Shlomo Dubnov, "Automatic Classification of Musical Instrument Sounds," *Journal of New Music Research*, vol. 32, no. 1, pp. 3–21, Mar. 2003.

[2] Morten Christophersen, Mark Aarup Mikaelsen, Jens Martin Oddershede, Thomas Deleuran Rasmussen, Steffen Stengaard Villadsen, Karen Reuter, and Per Rubak, "Automated Recognition of Drum Types," .

[3] Perfecto Herrera, Alexandre Yeterian, and Fabien Gouyon, "Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques," in *Music and Artificial Intelligence*, pp. 69–80. Springer, 2002.

[4] Antti Eronen and Anssi Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. 2000, vol. 2, pp. II753–II756, IEEE.

[5] Antti Eronen, "Comparison of features for musical instrument recognition," in *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*. 2001, pp. 19–22, IEEE.

[6] Keith D Martin and Youngmoo E Kim, "Musical instrument identification: A pattern-recognition approach," *The Journal of the Acoustical Society of America*, vol. 104, no. 3, pp. 1768–1768, 1998.

[7] Giulio Agostini, Maurizio Longari, Emanuele Pollastri, et al., "Content-based classification of musical instrument timbres," in *international workshop on content-based Multimedia indexing*, 2001.

[8] Alexander Lerch, *An introduction to audio content analysis: Applications in signal processing and music informatics*, John Wiley & Sons, 2012.