Audio Engineering Society

Convention Paper

Presented at the 127th Convention
2009 October 9–12 New York, NY, USA

Automated Cloning of Recorded Sounds by Software Synthesizers

Sebastian Heise[1], Michael Hlatky[1], and Jörn Loviscach[2]

[1] Hochschule Bremen (University of Applied Sciences), 28199 Bremen, Germany
Sebastian@h3e.eu, mhlatky@acm.org

[2] Fachhochschule Bielefeld (University of Applied Sciences), 33602 Bielefeld, Germany
joern.loviscach@fh-bielefeld.de

ABSTRACT

Any audio recording can be turned into a digital musical instrument by feeding it into an audio sampler. However, it is difficult to edit such a sound in musical terms or even to control it in real time with musical expression. This does not change much if a more sophisticated resynthesis method is applied. Many electronic musicians appreciate the direct and clear access to sound parameters a traditional analog synthesizer offers. Can one automatically generate a synthesizer setting that approximates a given audio recording and thus clone a given sound to be controlled with the standard functions of the particular synthesizer employed? Even though this problem seems highly complex, we demonstrate that its solution becomes feasible with today's computer systems. We compare sounds on the basis of acoustic features known from Music Information Retrieval and apply a specialized optimization strategy to adjust the settings of VST instruments, which is sped up using multi-core processors and networked computers.

1. INTRODUCTION

The automatic adjustment of music synthesizers to simulate a given recorded sample is a difficult task for three reasons: First, it is not evident how one should specify the error metric of the search process; obviously, a comparison of the original and the simulated waveform on a sample-by-sample basis would be by far too picky, as the waveforms should only sound similar, which can be achieved by very highly differing waveforms. Second, the search is non-linear and comprises a large number of dimensions, some of them not continuous but discrete. This leads to optimization algorithms exploring only fraction of the solution space and/or getting stuck in local extremums. Third, the parameters of a standard synthesizer rarely stick to a standard selection. And even when they do, the actual meaning of numeric settings of parameters such as "Shape" or "Resonance" varies from unit to unit. We have created a system that successfully addresses all three of these issues for a broad range of sounds and synthesizer units.

2. RELATED WORK

The cloning of sounds is a method used widely in popular music – musicians often try to adapt their "style" to cover an idol, using similar instrumentation and recording techniques. In a musical genre called "beatboxing." artists realistically emulate especially drum and percussion instruments only with their mouth, lips, tongue and voice. Even in nature, moths mimic sounds of bats as defense strategy [2], and the lyre bird has perfected mimicking human-produced sounds as that it can for example sing realistic-sounding camera-shutter or chain-saw tones [3].

However in computer science, the optimized cloning of arbitrary sounds has not been addressed yet, most probably due to the fact that the problem is highly complex. What has been achieved so far is the frequency-domain adjustment of sounds by means of equalization, which is often utilized in ADR matching [11]. This however does not require any optimization strategy, as precise adaption in frequency domain can be undertaken very effectively with sufficient FIR filters.

In a previous work [9], one of the authors already used a genetic optimization algorithm to find equalizer settings so that the magnitude response curve matches anchor points defined by the user. Genetic optimization methods are also known to be highly effective in filter design [4][5][6][7][8].

In further previous work [10], one of the authors searched for statistical coherence in VST synthesizer setup libraries in order to simplify the adjustment of a highly-parameterized synthesizer by means of on parameter change automatically adjusting further, intrinsically unaffected parameters.

Ultimately, in previous work we addressed the automatic adjustment of VST reverberation effects [1]. In this work we demonstrated software that uses given room impulse response recordings and optimizes the settings of standard VST reverberation plug-in in order to achieve perceived similarity of the reverberation produced by the plug-in in comparison to convolution with the impulse response. We showed that results of the tool could easily keep up with the quality of hand-crafted settings by professionals at a higher speed in the plug-in's setting up time. In this work we extend the topic to all recorded audio files as optimization target and use more generic plug-ins such as standard VST synthesizers.

3. IMPLEMENTATION

3.1. Audio Similarity

In perceived tonal audio, three features must resemble technically in order to provide an impression of "similarity" to the listener. The overall spectrum's shape (timbre), the volume curve over time (loudness), and the fundamental frequency (pitch). In studies conducted in 1977, Grey [14] identified two vital acoustic features: "spectral energy distribution" and "synchronicity in the collective attacks and decays of upper harmonics," which are required to be similar in order to allow for perceived similarity in two different sounds.

In the opinion of the authors, in order to achieve perceived similarity, especially the onset of a sound must be cloned as accurately as possible. Furthermore, the cloned sound must have the same fundamental frequency, and the loudness envelope must be similar.

Our software analyzes the audio output of a VST synthesizer plug-in, compares it to a given recording and adjust the parameters of the plug-in in order to achieve perceived similarity between the two sounds. As there is no need for real-time audio output during the optimization, the plug-ins can run at full speed and need not be throttled to a given playback samplerate. Hence, the computation can be accomplished in a fraction of the actual audio playback duration.

3.2. Similarity Measure

To capture the similarity of the original and the synthesized sound, the comparison of the synthesizer's audio data to the target recording is not based on the waveforms as such but on acoustic features extracted from them. Mel-Frequency Cepstral Coefficients (MFCCs) [12] are a first choice here. They are known to represent all noisy signals well, as they bear information about the spectrum's general shape. However, MFCCs alone do not suffice to fully represent tonal sounds, which are typical for most musical instruments.

In previous work [1], when comparing only the reverberation tail of an impulse response, MFCCs over time proved to be sufficient enough to allow for realistic-sounding results. In this work, however, when comparing by far more complex sounds, as for example recordings of piano notes, drums or violins, MFCCs over time alone did not prove to be an effective measure. As all MFCCs over time enter the comparison with equal weight, a sound could be found as “similar” by the software, which indeed resembles the bulk of the original quite accurately; the synthesized sound’s onset, however, was highly different from the original's, thus resulting in perceived dissimilarity. In informal listening tests, especially the optimized sound’s onset turned out to be very crucial for the perceived similarity. We therefore opted for a solution that allows trading off precision in the time domain (thus being able to clone the sound’s onset as well as possible) against precision in the frequency domain (thus being able to achieve the correct fundamental frequency and overall spectral envelope). To facilitate balancing the optimization between frequency domain and time domain, we form the spectrum of the MFCCs over time by applying a further discrete cosine transformation (DCT).

The system extract the 26 MFCCs from audio sampled at 44.1 kHz at a window size of 1024 samples with 50% overlapping windows. Then a 1-D DCT is taken of each MFCC over time. This results in a matrix with the first column (y-axis) containing the “offset” of each of the 26 MFCCs, and each line (x-axis) the frequency content of every MFCC’s time series. By now allowing for a weighting on the x- and y-axis of this matrix in the comparison, we can differentiate between an optimization targeting the frequency domain and one targeting the time domain. This allows for a differentiation in the optimization for more accurate sound onset, or for a more accurate volume envelope. Taking both dimensions always fully into account is rather disadvantageous, as it often exceeds the capabilities of the used synthesizer plug-in. The software actually only allows for a selection of how many values in x- and in y-domain are taken into account. By including all values in the x-domain and only, say, 5 in the y-domain, the software compares with a high resolution in time-domain, and by including all 26 values in the x-domain, however less in the y-domain, a high tonal resolution is achieved in the optimization.

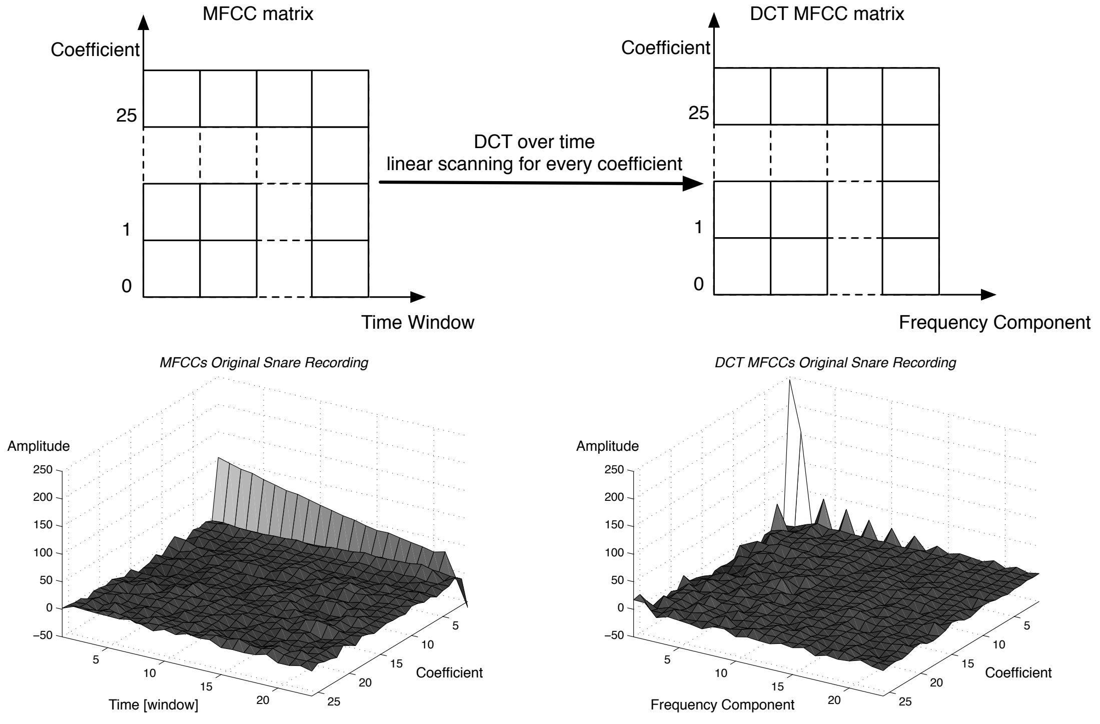


Fig. 1: The design of the MFCC matrix and the corresponding cosine-transformed DCT MFCC matrix. In the lower part of the figure, exemplary data for a snare drum recording are displayed.

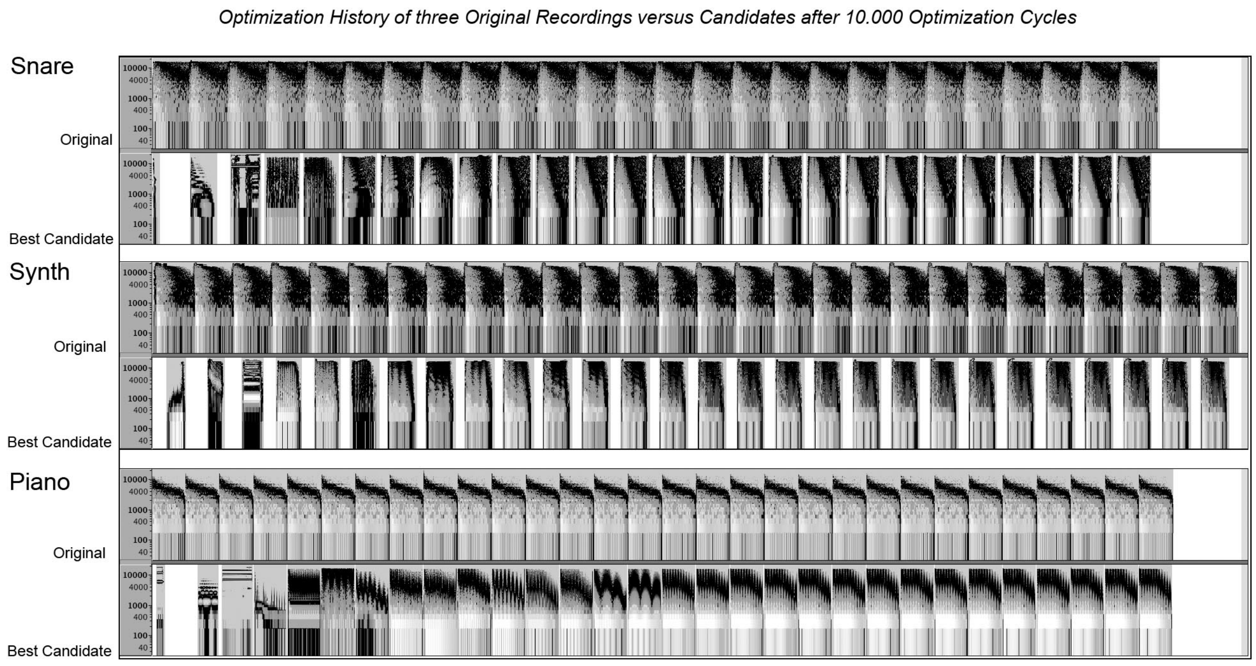


Fig. 2: Spectrograms of the original recordings versus the best candidates in the course of the optimization. In 10,000 optimization steps, each time a new best candidate was found the spectrogram is plotted in comparison to the original sound's spectrogram above. The figure is screen-captured from Audacity's spectrum display.

As examples for this work we used a dry snare drum recording, a classical synthesizer chord, and a piano recording. All three audio files were cloned automatically by our software and the free ASynth[2] VST plug-in. The plug-in is a six-voice virtual analog synthesizer with two oscillators and three circuit modeled filters. Obviously the plug-in will not produce the original sound completely realistic-sounding, as the plug-in lacks the acoustic variability. It is, however, astonishing how close the plug-in's results come to the original recordings after just a few optimization steps.

In Fig. 2, the course of the optimization is displayed. In 10,000 optimization steps, every time a new better candidate was found by the system, its spectrum over time is plotted in the timeline with the original sound's spectrum overt time as comparison above. The figure of 10,000 optimization denotes the number of comparisons between the original and the candidates.

For the comparison, the software simply computes the Euclidean difference between the DCT matrices of the original and of the candidate. A lower value here depicts higher similarity. This simple approach was favored above more complicated distance measures suggested by the MIR community because of its computational efficiency while maintaining stable results. Furthermore, Herrera et. al. [15] as well as Jensen et. al. [16] have shown that a specific distance measure is not crucial for a feature comparison's results.

3.3. Optimization

In previous work [1], we already compared four different optimization strategies to tackle this challenging problem. The optimization of a synthesizer plug-in is high-dimensional and highly non-linear. Common synthesizer plug-ins to be found in digital audio production normally have more then 50 parameters, which all have to be taken into account during the optimization process.

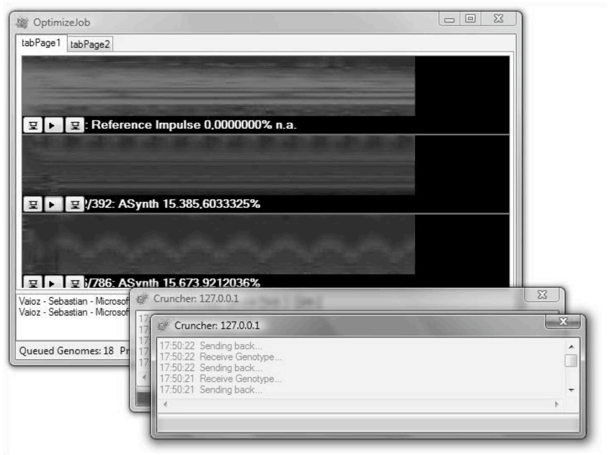
We tackle this problem with a dedicated optimization algorithm, a Particle Swarm Optimization (PSO) [13]. In our earlier work [1] this technique proved to be the most reliable. We did, however, also test an evolutionary genetic algorithm, and a simplex optimization method. However, as the results did not differ in terms of optimization accuracy and speed from our earlier research, they are not considered further here.

[2] http://antti.smartelectronix.com/

The PSO algorithm works as follows: A population of particles is initialized by the system. Each particle carries a vector containing the plug-in settings, and an offset vector, which is randomly initialized. In every update step the particles are moved through the problem space by adding the offset to the current setting. Each particle keeps track of its coordinate in problem space that has generated the best similarity measure so far: the local best setting. Furthermore, the optimizer keeps track of the population's best setting ever: the global best setting To update a particle, its position is shifted toward a random blend of the local and the global best settings. The PSO algorithm provides the VST plug-in with all parameter settings as well as with single MIDI note event. This note's number, its note-on and note-off times, and its velocity are also subject to optimization.

3.4. Software

The Software is implemented in C# using the Microsoft .Net 3.5 framework. The optimizer functions as a server application, tracking the optimization history and assigning tasks to a user-defined number of client cruncher processes, which can be spread over a network of interconnected machines. The communication between the cruncher client and the optimization server is carried out using named pipes [17]. Each cruncher client contains its own VST host.



```
: Reference Impulse 0,0000000% n.a.
/392: ASynth 15.385,6033325%
/786: ASynth 15.673,9212036%
Vaioz - Sebastian - Microsof
Vaioz - Sebastian - Microsof
Queued Genomes: 18  Pr
```

```
17:50:22  Sending back...
17:50:22  Receive Genotype...
17:50:22  Sending back...
17:50:21  Receive Genotype...
17:50:21  Sending back...
```

Fig. 4: The user interface of the prototype software. On the main screen, the MPEG 7 spectral envelopes of the sounds are displayed, the cruncher clients display job state information.

When a client crunches launches, all VST synthesizer DLLs as well as the original recording's DCT MFCC matrix are provided to the cruncher client processes.

As the optimization proceeds, the crunchers receive new VST settings from the optimization server, compute the plug-in's audio data, extract the data's features, compute the DCT MFCC matrix and then compare it to the original recording's dataset, from which they calculate the distance measure. This value is submitted back to the optimization server, again via a named pipe.

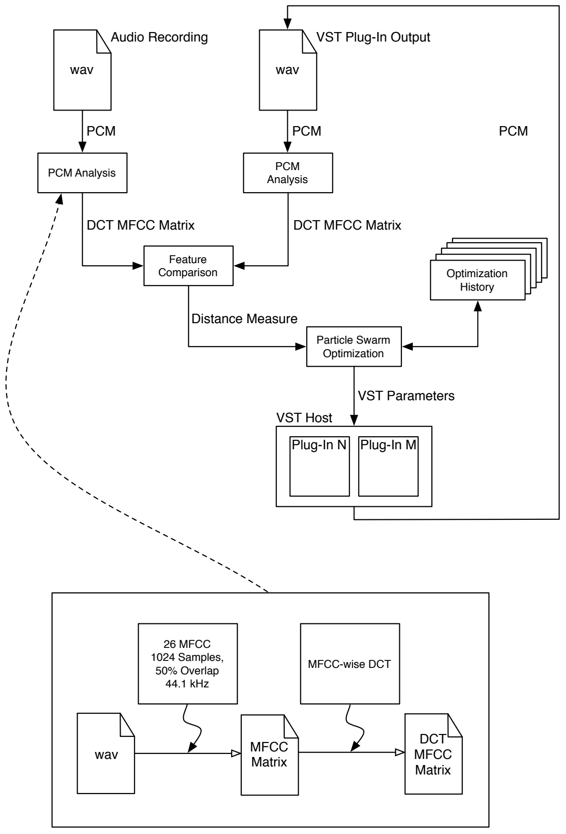


Fig. 5: Schematical overview of the software: The software prototype consists of four main parts: The PCM Analysis as described in 3.2, The Feature Comparison, the Optimization and the VST Host.

The optimization server tracks all evaluated VST settings along with the corresponding distance as similarity rating. Furthermore, for user interface purposes, the MPEG 7 spectral envelope [18] of each new candidate is extracted as well and displayed to the user. This feature is consumes most of the computational power; it can be switched off for greater performance.

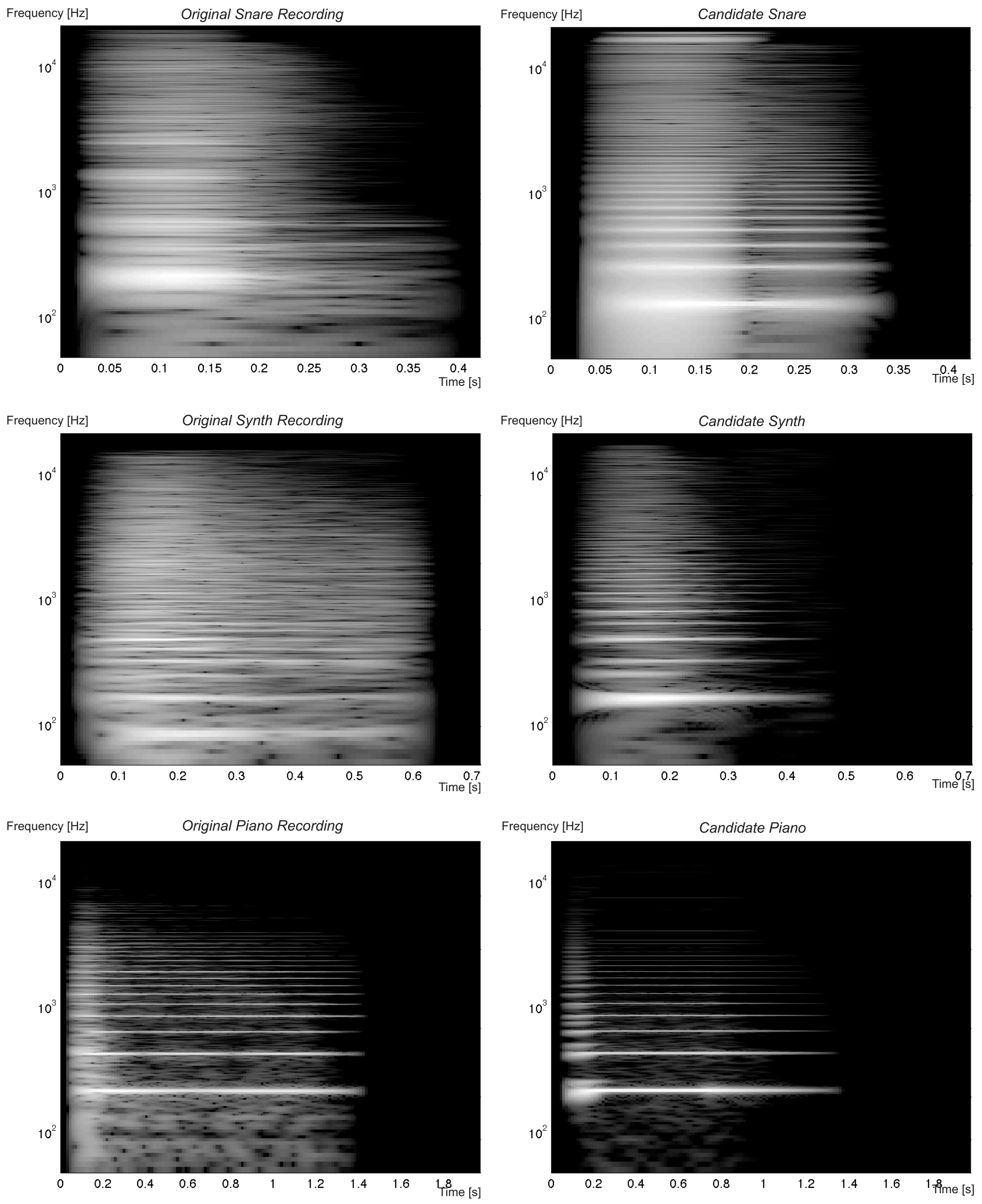


Fig. 6: The results of the optimization after 10.000 cycles. On the left-hand side of the figure, the spectrogram of the original snare drum, synthesizer patch and piano note recording are plotted. On the right-hand side, the corresponding synthesized sounds are shown. What clearly can be seen is that the fundamental frequencies of the original sounds were met, as well the on-set characteristics. The duration of the sound were not mimicked in the first two examples, the resulting audio "sounds" similar due to the similarity in the overall spectrum envelope.

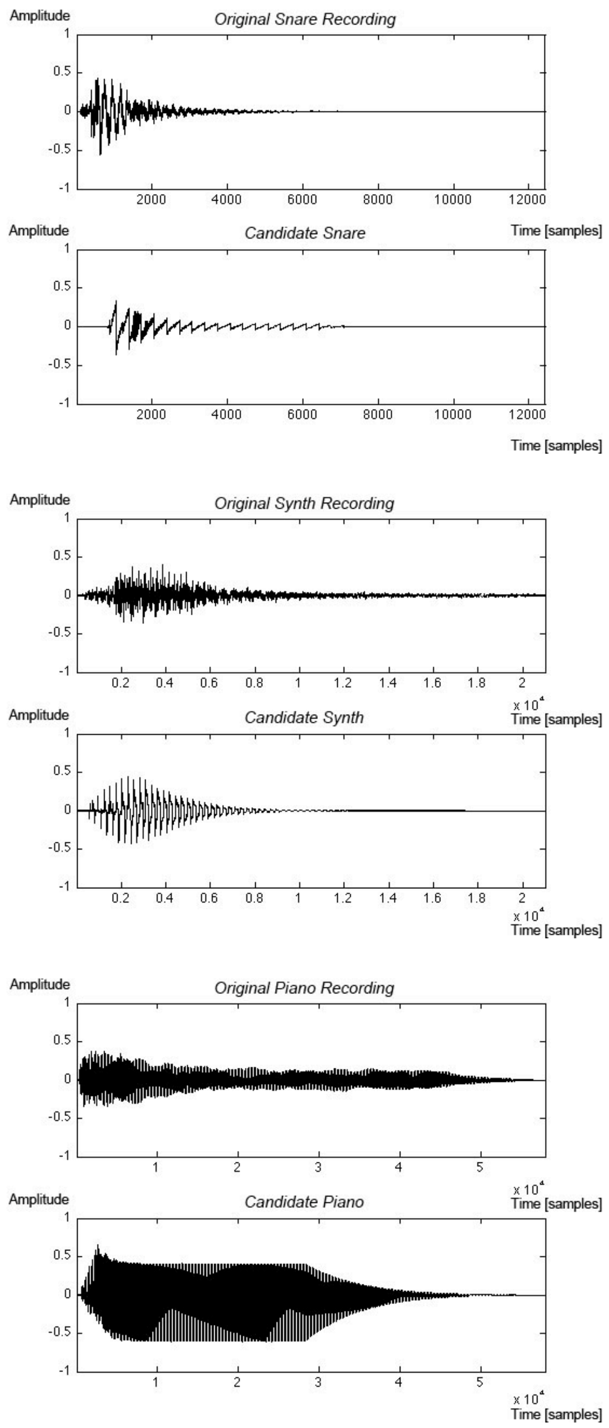


Fig. 7: Example for the optimization: A recording of a snare drum, a synthesizer patch and a piano note are given as original. Our software adjusts the parameters of the ASynth plug-in in such a way that it produced the candidate waveforms after 10,000 optimization steps.

4. RESULTS

In Fig. 6, spectrograms are displayed for each of the original and the corresponding optimized sounds after 10,000 optimization steps. For the cloning of the snare drum recording and the synthesizer patch sound, we opted for a higher tonal reproduction, therefore taking all 26 DCTs of the MFCCs into account, and 50% of the available time information. For the snare drum recording, with a 1024 sample window and 50% overlap, the software calculates 23 MFCC vectors over time. After the linear-scanning DCT for every coefficient, the first 12 frequency coefficients of each MFCC are used in the optimization process. This yields to a 12x26-value matrix in the optimization process. The synthesizer patch's matrix with the same weighting in the course of the optimization had 20x26 values. For the cloning of the piano note recording, we opted for a higher time-span reproduction, therefore taking only 50% of the MFCCs into account, but all the available information in the time domain, thus yielding to a 13x112 matrix. The fundamental frequency of the original piano note recording was still found accurately in the course of the optimization. Fig. 7 shows the corresponding waveform displays for all sounds.

From the waveforms, the similarity of the sounds is not obvious, and also from the spectrograms in Fig. 7 the similarity can not be judged completely. One can, however observe the similarity in the spectrograms in terms of fundamental frequency, spectral shape, attack and decay characteristics of the sounds as well as the overall duration.

5. CONCLUSION AND OUTLOOK

We have demonstrated a software that allows for cloning of any recorded sounds with by VST software synthesizers, insofar as the provided VST plug-in can produce a similar-sounding audio output. The plug-in's settings are obtained using a Particle Swarm Optimization strategy, the similarity measure is calculated using the difference of the coefficient-wise discrete Cosine Transform of the Mel-Frequency Cepstral Coefficients of the synthesizer's audio output and the original audio recording. Within the capabilities of the utilized plug-in, the optimization usually finds adequate settings of the optimization plug-in after only 2,000 similarity ratings, mostly already the tenth found candidate bears significant similarity to the provided audio material. The computation of 10,000 optimization cycles usually takes one hour on a standard laptop computer with two processing cores.

Future work may incorporate the use of more sophisticated synthesizer plug-in's to not be limited by a plug-in's capabilities, and listening tests to judge the quality and the similarity of the cloned sounds in comparison to the original recordings.

6. REFERENCES

[1] Heise, S., Hlatky, M., Loviscach, J.: Automatic Adjustment of Off-the-Shelf Reverberation Effects. Presented at the 126th AES Convention, (Munich, Germany, May 7-10 2009), 2009.

[2] Wake Forest University (2007, May 30). Moths Mimic Sounds To Survive. ScienceDaily. Retrieved July 13, 2009, from http://www.sciencedaily.com-/releases/2007/05/070529211003.htm

[3] http://www.youtube.com/watch?v=VjE0Kdfos4Y

[4] Rimell A. and Hawksford, M.: The application of genetic algorithms to digital audio filters. Presented at the 98th AES Convention (Paris, France, February 25–28, 1995), 1995.

[5] K. Uesaka and M. Kawamata, "Evolutionary synthesis of digital filter structures using genetic programming," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 50 (2003), no. 12, 977–783.

[6] V. Aggarwal and W. Jin, "Filter approximation using explicit time and frequency domain specifications," GECCO'06, 753–760.

[7] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, "Optimal design of digital IIR filters by using hybrid Taguchi genetic algorithm," IEEE Transactions on Industrial Electronics 53 (2006), no. 3, 867–879.

[8] Y. Yu and Y. Xinjie, "Cooperative coevolutionary genetic algorithm for digital IIR filter design," IEEE Transactions on Industrial Electronics 54 (2007), no. 3, 1311–1318.

[9] Loviscach, J.: Graphical Control of a Parametric Equalizer. Presented at the 124th AES Convention, (Amsterdam, The Netherlands, May 17-20, 2008), 2008.

[10] Loviscach, J.: Programming a Music Synthesizer through Data Mining. Presented at the 8th NIME Conference, (Genova, Italy, June 5-7, 2008), 2008.

[11] http://support.apple.com/kb/TA23246?viewlocale=en_US

[12] J. Foote. Content-based retrieval of music and audio. In C.-C. J. Kuo, editor, Multimedia Storage and Archiving Systems II, Proceedings of SPIE, pages 138–147, 1997.

[13] Kennedy, J. and Eberhart, R.: Particle swarm optimization. Presented at the IEEE International. Conference on Neural Networks, (Perth, Australia, November 27 – December 1, 1995), 1995.

[14] Grey, J. M. (1977). Multidimensional perceptual scaling of musical timbres. Journal of the Acoustical Society of America, 61(5), 1270-177.

[15] http://ismir2000.ismir.net/papers/herrera_paper.pdf

[16] http://www.ee.columbia.edu/~dpwe/pubs/JenECJ07-gmmdist.pdf

[17] http://msdn.microsoft.com/en-us/library/aa365590(VS.85).aspx

[18] Kim, H.-G., Moreau, N. and Sikora, T.: MPEG-7 Audio and Beyond - Audio Content Indexing and Retrieval. John Wiley & Sons Ltd (West Sussex, England 2005) 79, 2005. AES Convention, (San Francisco, CA, USA October 5-8. 2006), 2006.