A polyphonic pitch tracking embedded system for rapid instrument augmentation

Rodrigo Schramm
Department of Music
Universidade Federal do Rio Grande do Sul / Brazil
rschramm@ufrgs.br

Federico Visi
Institute of Systematic Musicology
Universität Hamburg / Germany
federico.visi@uni-hamburg.de

André Brasil
Department of Music
Universidade Federal do Rio Grande do Sul / Brazil
andrebrasilguitar@gmail.com

Marcelo Johann
Institute of Informatics
Universidade Federal do Rio Grande do Sul / Brazil
johann@inf.ufrgs.br

ABSTRACT

This paper presents a system for easily augmenting polyphonic pitched instruments. The entire system is designed to run on a low-cost embedded computer, suitable for live performance and easy to customise for different use cases. The core of the system implements real-time spectrum factorisation, decomposing polyphonic audio input signals into music note activations. New instruments can be easily added to the system with the help of custom spectral template dictionaries. Instrument augmentation is achieved by replacing or mixing the instrument's original sounds with a large variety of synthetic or sampled sounds, which follow the polyphonic pitch activations.

Author Keywords

multi-pitch activations, polyphonic augmented instruments, embedded Linux computers

CCS Concepts

•**Applied computing** → **Sound and music computing;** •**Hardware** → **Digital signal processing;** •**Human-centered computing** → *Mixed / augmented reality;*

1. INTRODUCTION

Designing a Digital Music Instrument (DMI) is a very complex task. Many aspects must be planned to achieve a successful development as revealed by a recent study [11]. Morreale and McPherson analysed many of these aspects through an extensive survey with 70 DMI expert makers. Among the makers' answers are concerns about the functionality, aesthetic and craftsmanship aspects, familiarity, the simplicity of interaction, set-up time, portability, low latency, modularity and ownership.

Augmented instruments can be a bridge between traditionally established musical instruments and novel digital interfaces. They can naturally address some of the above aspects so that original characteristics might be preserved (e.g. instrument shape, modes of interaction, aesthetic, etc.) while others are extended. The augmentation may focus on the interaction between the performer and the instrument, on how sound is generated, or both.

Many hardware devices such as tiny computers, microcontrollers, and a variety of sensors and transducers have become more accessible. In addition, plenty of software and source libraries to process and generate sounds [12] are freely available. All these resources create a powerful toolset for building DMIs.

Instrument makers can manipulate audio with the aid of many digital signal processing algorithms and libraries [15, 5, 7, 16, 14]. These programming tools can operate on single (or multiple) audio channels, performing tasks such as pitch shifting, pitch detection, distortion, delay, reverb, EQ, etc. However, detection and decomposition of polyphonic audio in real time is still a challenging task, especially on low-cost embedded systems. This imposes some limitations to the design of new DMIs, requiring complex signal processing techniques, and/or the use of multi-channel transducers (e.g. hexaphonic guitar pickups) placed on the body of the musical instrument for polyphonic source separation.

We present a method for rapid and easy replacement of electric-acoustic sounds on augmented polyphonic instruments[1]. The proposed technique implements a machine learning method based on spectrogram factorisation [18], which can detect the multi-pitch activations from the polyphonic content. It runs in real-time on low-cost embedded Linux computers[2], with low latency, using a single mono input channel. This means that the technique is suitable for the design of new instruments that require only one transducer, or it can be applied to augmenting traditional instruments with mono channel output (e.g. electric-acoustic guitar). Moreover, the solution is portable and has low setup time.

Several approaches implement some kind of cross-correlation or template matching in order to estimate the similarity between the input audio signal and a pre-learned pattern [19, 13]. These methods might work well to estimate the activation of pre-learned timbre features, however, they cannot distinguish the concurrent multi-pitch content present in the audio signal.

Pre-learned templates have been explored in the Music

NIME'18, June 3-6, 2018, Blacksburg, Virginia, USA.

[1] The term polyphonic, in this paper, means the presence of multiple concurrent pitched note activations at same time.

[2] In this work we have tested our system on a Raspberry Pi-3 Model B (`https://www.raspberrypi.org`)

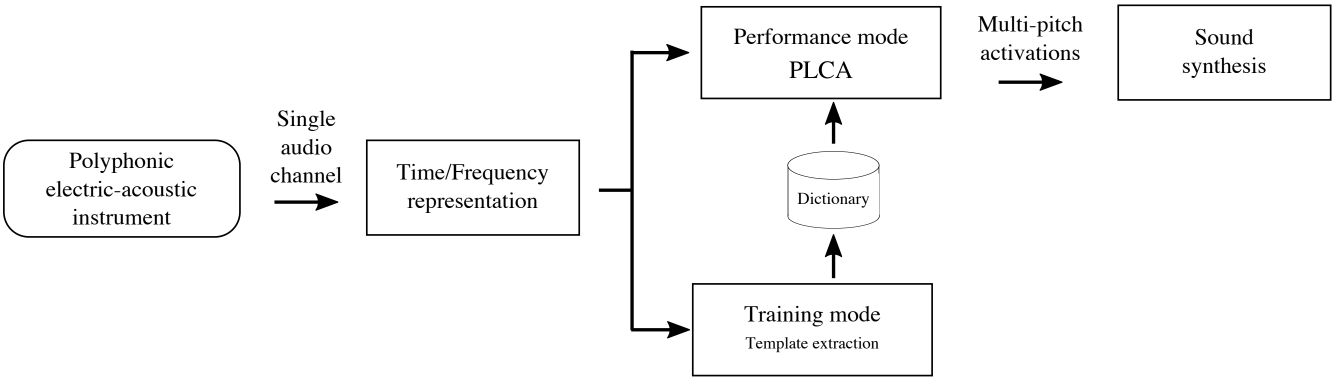

Figure 1: System pipeline.

Information Retrieval (MIR) field for the task of automatic music transcription based on spectrogram factorisation. Improvements have been achieved in the recent years using techniques based mainly on Non-Negative Matrix Factorisation (NMF) [17, 10] and Probabilistic Latent Component Analysis (PLCA) [1, 2]. Additionally, these methods allow the integration of music language models that can guide the factorisation convergency to meaningful results. A drawback of these solutions is the high computational cost involved to process the algorithm. Usually, automatic music transcription algorithms are designed to achieve better generalisation and ideally perform well with many kinds (or classes) of musical instruments. In this paper, we go the opposite way. Since the target is only one particular musical instrument, we propose a simplification in the factorisation process in favour of the usability of the system by the instrument maker.

The spectrogram factorisation in our approach is guided by a timbre dictionary, which is built from audio samples of the target musical instrument. The method allows a rapid customisation and extension by adding new dictionaries.

It is worth to note that this approach does not aim at creating a new audio to MIDI conversion technique. Multi-pitch detection is still an open problem in MIR [3] research and we do expect many false positiveand false negativepitch activations. This is the reason why this approach implements a dedicated sound generation stage based on table-based Numerically Controlled Oscillators (NCO)[3], which, in fact, explores the multi-pitch activation to create new sounds with polyphonic content, embracing spurious pitches or harmonic estimates. Figure 1 illustrates the system pipeline. Details of this system are explained in Section 2 and experiments are described in Section 3. Conclusions and future work are drawn in Section 4.

2. SYSTEM DESIGN

The core of this work is based on multi-pitch detection of the polyphonic content generated by the musical instrument. In particular, our system looks for multi-pitch activations estimated by a spectrogram factorisation algorithm based on the PLCA technique. We follow the implementation proposed by Benetos and Weide [4], which uses a fixed dictionary with spectral templates. Analogously, in our approach the input spectrogram is factorised into pitch activations, pitch deviations, and instrument parts (the system focus on only one instrument, but we might want to distinguish identical pitches originated from distinct parts of the instrument).

2.1 Polyphonic Pitch Activations

Given an audio input in the frequency domain representation, we want to decompose the spectrum $X_{\omega,t} \in \mathbb{R}^{\Omega \times T}$ where ω denotes log-frequency and t time. In this model, $X_{\omega,t}$ is approximated by a bivariate probability distribution $P(\omega, t)$, which is in turn decomposed as:

$$P(\omega, t) = P(t) \sum_{s,f,p} P(\omega|s, f, p) P_t(s|p) P_t(f|p) P_t(p) \qquad (1)$$

where $P(t)$ is the spectrogram energy. Similar to [4], the variable $p \in \{40, ..., 108\}$ denotes pitch in linear MIDI scale, $f \in \{1, ..., 5\}$ denotes the pitch deviation from 12-tone equal temperament in 20 cent resolution. The range of the possible values of f depends on the spectrogram resolution. Variable $s \in \{1, ...S\}$ denotes a specific part of the instrument body. For example, the factorisation applied to the spectrogram of an electric guitar with six strings would ideally use $S = 6$. The unknown model parameters $P_t(s|p)$, $P_t(f|p)$, and $P_t(p)$ are estimated through the iterative expectation-maximization (EM) algorithm [9]. The model parameters are randomly initialised, and the EM algorithm iterates over 30 iterations. Details of how the expectation and maximization steps are computed can be found in [4].

At the end of the spectrogram factorisation, $P_t(p)$ gives the multi-pitch activations (amplitudes of pitch components) in semitone scale, while $P_t(f|p)$ estimates the respective pitch deviations, being useful to detect possible small tune deviations or pitch bends. $P_t(s|p)$ gives the contribution of each instrument part s for each pitch activation p.

2.1.1 *Pitch Resolution* × *Computational Cost*

This approach uses the constant Q-transform for the frequency representation. The log-frequency resolution (bins per octave) of the constant Q-transform can be tuned in order to control the computational need of the algorithm, regulating the tradeoff between low computation and pitch precision. This means that if there is not enough computational power, we can reduce the number of frequency bins per octave (e.g. one template per semitone, $f \in \{1\}$). On the other hand, if we want more precision on the fundamental frequency estimate of each pitch detection and there is sufficient computational power in the embedded computer, then we can increase the number of frequency bins (e.g. 5 bins per octave give us a 20 cent resolution, $f \in \{1, ..., 5\}$).

Another important factor that might affect the real-time performance is the total extent that variable p can span. As a design decision, the instrument maker can constrain an excessively wide pitch range, breaking it into multiple and smaller pitch ranges, and consequently, reducing the processing time. Of course, this decision will affect the multi-

[3] These NCOs can be extended to implement additive, FM, sample-based and other synthesis methods.

pitch detection performance, since the system will not be able to detect pitch activations that are not indexed by p.

2.1.2 Template Dictionary

In order to generate the dictionary of timbre templates, the system implements a *training mode*. In this mode, the instrument maker selects a single part of the instrument's body and plays its respective possible monophonic sounds. For example, if the instrument is an electric guitar, each string could be considered as a "body part". During this *training mode*, the system scans the audio generated by the instrument[4], tracks the fundamental frequency of each played note, and stores the respective spectrum template. Each final spectrum template is obtained by the median over all extracted spectral vectors of each individual detected pitch and body part (see Figure 2c).

The resulting dictionary is a sparse matrix containing gaps for pitches that were not played or detected. In the constant-Q transform, there is a constant relative distance between the harmonic components and their fundamental frequency, allowing the spectrum approximation at missing pitch locations by simply shifting existing templates along the log-frequency axis. Therefore, in this work we apply a linear replication approach [8], filling the missing parts along the pitch scale. Figure 2 illustrates the dictionary creation process for an electric guitar used in our experiments. The dense dictionary after the linear replication is shown in Figure 2d.

2.1.3 Temporal coherence and Sparsity

Temporal continuity is promoted in this approach by applying a median filter to the $P_t(p)$ estimates across time. Since the system is causal, large filter span will inevitably increase latency. Conversely, very low filter span might generate too noisy multi-pitch estimates. We have set a filter span of 3 processing frames as default value.

Besides the temporal continuity, the system also allows the use of sparsity constraints on $P_t(p)$ and $P_t(s|p)$. The sparsity constraints control the polyphony level and the instrument part contribution in the resulting transcription. Both filter span and sparsity level can be controlled in real time by the performer.

2.1.4 Activation Threshold

We employ a thresholding scheme based on hysteresis to discard very low pitch activations. A high threshold T_h is used to detect pitch activations and a low threshold T_l is used to detect the deactivations. In our experiments we have set $T_h = 0.005$ and $T_l = 0.002$ as default values. Thus, before starting the audio synthesis, $P_t(p)$ is set to zero if p is not active.

We can combine $P_t(p)$ and $P_t(f|p)$ to get the pitch shift tensor $P_t(p, f)$. By stacking together slices of $P_t(f, p)$ for all values of p, we can create a $\frac{100}{f_b}$ resolution time-pitch representation:

$$P_t(f') = [P_t(f, p_{min} \ mod \ f_b), ..., P_t(f, p_{max} \ mod \ f_b)], \quad (2)$$

where f_b is the number of steps between two consecutive semitones, p_{min} and p_{max} define the pitch range in $\frac{100}{f_b}$ resolution, and *mod* is the modulo operation.

2.2 Sound Generation

For sound generation, many synthesis techniques can be employed, the most basic ones being additive, subtractive,

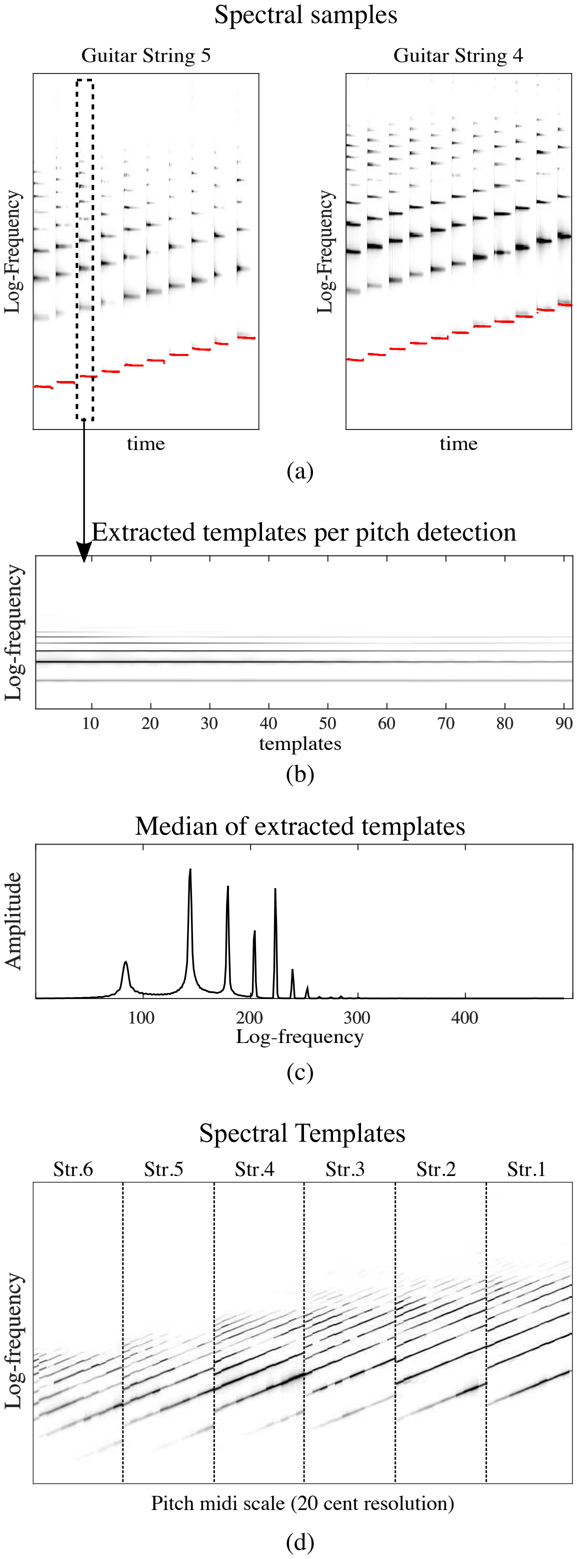

Figure 2: Dictionary construction: a) CQT representation of sampled audio from guitar strings 5 and 4 (the red curve indicates pitch); b) Extracted spectral templates from a specific pitch from string 5; c) Median estimate of templates; d) Final dense dictionary containing spectral templates of 6 guitar strings across the MIDI pitch scale using 20 cent resolution.

[4] In this work we address musical instruments that can be amplified using electric transducers (contact microphones, guitar pickups, etc.).

frequency modulation and sample-playback. Subtractive and sample-based synthesis are more complex. They require generating and manipulating waves with wide harmonic content in the digital domain, using more computational power and requiring special treatment to avoid aliasing problems. On the other hand, additive synthesis and frequency modulation can be implemented with just sine wave oscillators, but they require many oscillators per note, ranging from a minimum of two up to more than ten.

A Numerically Controlled Oscillator (NCO) consists of a phase accumulator (PA) followed by a phase to amplitude converter (PAC) [13]. The accumulator PA is a simple counter that counts forward progressing in steps computed from the desired frequency and sampling rate. The most efficient implementation of the PAC converter to generate a sine wave uses a look-up table containing a quarter or a full wave sample. Given that a wave table is already used, it can also store alternatively more complex waveforms. It can also be enlarged so as to hold a small recording of another instrument, thus implementing sample-based synthesis, all this subject to the same issues of other complex oscillators.

In this work, we have implemented a single NCO for each pitch activation $P_t(f')$ and currently use it as a primitive form of sample-based synthesis. So, for each frame t, the sound generation is computed by the following equations:

$$\begin{aligned} \phi_{f'} &= \phi_{f'} + k_{f'}, \\ x[t] &= \sum_{f'} P_t(f') W[\phi_{f'} \ mod \ S], \end{aligned} \tag{3}$$

where mod means the modulo operation, W is the wavetable and S is the wavetable size. $k_{f'} = S\frac{\theta_{f'}}{r}$ is the phase step, where r is the sampling rate, and $\theta_{f'}$ is the fundamental frequency in Hz of pitch f'.

2.3 System Prototype

A system prototype was developed using a single board computer Raspberry PI 3 (model B). This tiny Linux Computer has a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB RAM, a dedicated 400Mhz GPU, 4 USB, WIFI, bluetooth and ethernet connection. This device has quite small dimensions (3.370 in × 2.224 in × 0.67 in), making it suitable to be embedded into augmented instruments. In addition, the prototype uses a generic low cost USB 2.0 audio device of small size (1.72 in × 0.91 in × 0.47 in). This acts as the interface between the electro-acoustic instrument, the embedded computer, and the speakers. It features one monophonic mic input channel and one stereo line output channel. The audio capturing and sound synthesis has a sample rate of 44.1KHz and 16-bit resolution.

All project implementation is coded in C/C++ and runs on Raspbian Linux distribution[5]. Access to the audio buffers is done directly via ALSA[6] to avoid additional latency or processing overhead. The audio input is converted to 32 bits float point precision and all heavy processing is done using this data type. The constant-Q Transform is implemented following the method in [6], which takes full advantage of the fast Fourier transform computational efficiency. All digital signal processing is implemented with particular care to sequential memory accesses optimisation. Moreover, the fast Fourier transform computation is accelerated though GPU processing.

During the audio processing, the implementation uses a frame buffer of 4096 samples, and a hop size of 512 samples. We tested three distinct log-frequency resolution in the constant-Q Transform: 12, 36 and 60 bins per octave. The setup with 12 bins and p varying from 30 to 70 runs in real-time, consuming 70% of one single core of the CPU.

For the set of experiments reported in this paper, the user interaction was done through Linux terminal, using a USB keyboard. This user interface allows to switch between training and performance modes. In training mode, the user can add or remove instrument parts, as well as record new timbre samples from the target instrument. In performance mode, the system extracts the multi-pitch activations from the input signal (through the learned dictionary) and synthesises the output sound.

3. EXPERIMENTS

In this evaluation, we aim at unveiling operational characteristics and limitations of the proposed system. Since the system is customised for each particular instrument, there is no straightforward way to compare our results with other approaches to external multi-pitch detection. On top of that, our goal is not a system that explicitly implements automatic music transcription, thus usual comparison measures as the f-measure are not appropriate here. Moreover, this work aims at opening new possibilities for sound replacement in new augmented musical instruments. Thus, any potential false positives or false negatives would be intrinsic characteristics of the new instrument that the performer must learn how to manage and play with.

3.1 Three use cases

We have tested the proposed system in three distinct use cases. The first experiment consisted in the augmentation of a new instrument named as *Arco-Barco*. This instrument has a low polyphonic degree, its timbre contains many inharmonic components, and its audio signal is strongly corrupted by noise. A second experiment was done using crystal glasses. This complemented the first experiment since the timbre of crystal glasses is composed almost exclusively by the fundamental frequency with very few week harmonic components. Finally, a third and more extensive experiment was conducted applying our technique to the augmentation of an electric guitar. This evaluation focused on aspects such as latency, playability, reliability of multi-pitch activations and quality of sound generation. Video excerpts from these experiments can be found in `http://inf.ufrgs.br/~rschramm/projects/music/nime2018/`.

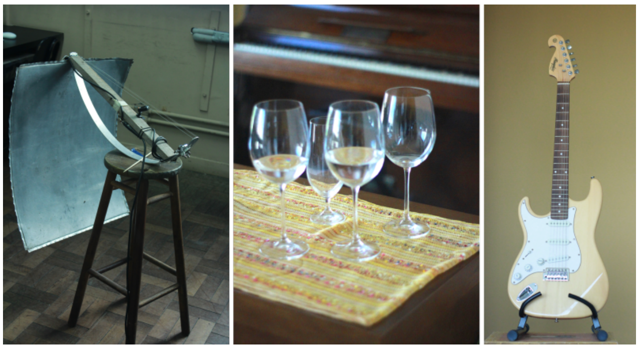

Figure 3: Musical instruments used in the experiments (from left to right): *Arco-Barco*; Crystal glasses; Electric guitar.

3.1.1 Augmented Arco-Barco

The *Arco-Barco* is a musical instrument made with pieces from waste. Its design shares similarities with the Berimbau

[5] https://www.raspbian.org/

[6] https://alsa-project.org/

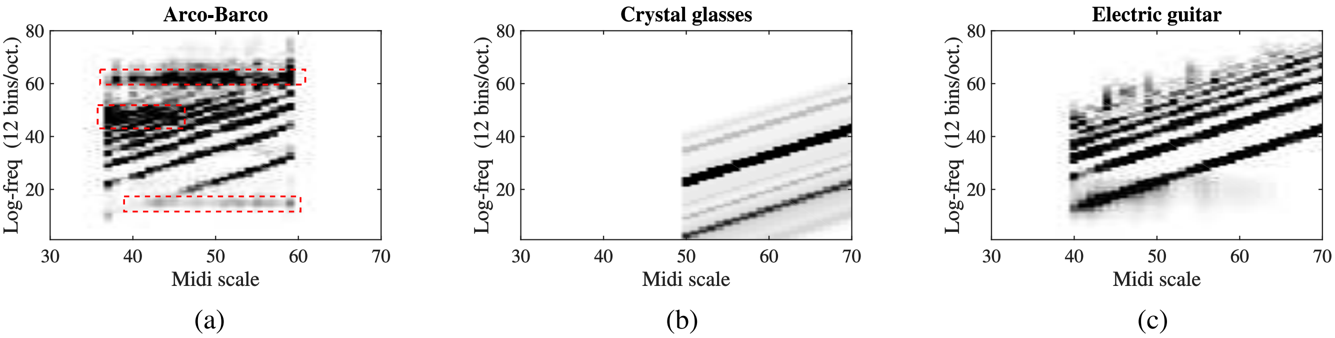

Figure 4: Trained template dictionaries for pitch resolution of 12 bins per octave.

although the *Arco-Barco* has three strings and the resonance body is built using a metal sheet. The sound is generated by rubbing the strings with a bow or by playing them with a drumstick. The pitch of this instrument is not well defined and the audio output contains many inharmonic content across the pitch range, as shown in the template dictionary in Figure 4a (dashed red areas).

Our system was not able to achieve a proper spectrum factorisation of the *Arco-Barco* timbre, generating many pitch error detections. This compromised the instrument playability.

3.1.2 Augmented Crystal Glasses

In order to check the capability of our system to decompose the audio spectrogram of instruments with clear (lower amplitudes in the higher frequency content) and harmonic sounds, we performed a proof of concept test. In this test, sounds from four crystal glasses with distinct pitches were processed by our algorithm. The sound of the crystal glasses was generated by hitting them with drumsticks, and the audio signal was captured by a condenser microphone. In this situation, the proposed system was able to detect the individual pitches and break down the audio polyphonic content, replacing the original sounds with synthetic waveforms.

3.1.3 Augmented electric guitar

We invited three professional guitarists to use the proposed system. The goal of this experiment was to check if the system would be suitable for use with a more harmonic instrument with wide pitch range. The tests with the system happened in distinct situations, with two different guitars (Jackson js-22 7 strings and acoustic Ibanez ag75). We have trained a specific dictionary for each guitar using a pitch range from E2 (82.4 Hz) to D5 (587.3 Hz). During each experiment session, the guitarists improvised with the instrument, combining the use of scales (monophonic) and chords (polyphonic).

3.2 Latency

During these experiments, we configured the system to use an analysis window of 2048 samples and three distinct hop sizes (the number of samples the system advances the analysis time origin from frame to frame): 256, 512 and 1024. The implementation using the Raspberry PI 3 could not run using a hop size of 256 samples. For this hop size, we have used a laptop computer with 2,4 GHz Intel Core i7 processor. We have set the median filter span to 3 frames in order to enforce temporal coherence.

Figure 5 shows the total latency between the input signal and the synthesised output signal, measured with the system implementation built in the Raspberry PI 3. For hop sizes of 512 and 1024 samples, the output signal presents delay of approximately 45ms and 70ms, respectively. The guitarists did not feel the latency when using 256 samples as hop size, but they notice a slight latency when using 512, although the playability was not compromised. The 1024 option was not acceptable for them. Similar latency behaviour was found for the experiment with crystal glasses.

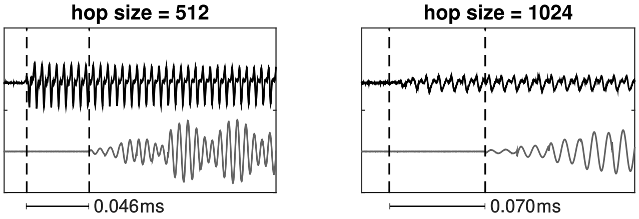

Figure 5: Latency between instrument input signal (black) and synthesised output signal (grey).

3.3 Accuracy

In order to verify the accuracy of the multi-pitch activations, the performances during the experiment with electric guitars were conducted with varying tempo, from slow to very fast. The system performed very well with melodic excerpts. However, the performers could notice sporadic missing notes when approaching very fast tempi (sextuplet notes at 140 BPM). They also registered that the perception of these missing notes varies depending of the chosen sound to replace the guitar timbre. For excerpts with chords, various harmonies were tested, including seventh chords.

Two main issues on multi-pitch activations were detected by the guitarists : 1) for some notes belonging to the chord, the system correctly detected the note but at one octave below; 2) when playing low pitches (between E2 and A2), the system sometimes added false positives related to the first two harmonics of the fundamental frequency.

3.4 Sound Generation

The system was able to generate synthetic sounds throughout the sample playback stage. We have used wavetables from `https://www.adventurekid.se/akrt/` in these experiments. Since at this stage the system does not implement any gain control of the additive partial sounds, sometimes the resulting sound is distorted by the clipping of the waveform. The guitarists also have noticed the presence of sound glitches in different degrees depending of the chosen timbre that is used for sound synthesis. These glitches are caused mainly by oscillations of activation amplitudes that vary between the thresholds T_h and T_l. This causes a frequent switch between *note on* and *note off* of the respective pitch components. The discontinuities in the resultant waveform, generated by the introduction of each new partial wave component, create these undesirable artefacts.

4. CONCLUSIONS AND FUTURE WORK

This paper described a new approach for augmenting polyphonic pitched instruments. Our method uses a spectrogram factorisation technique based on Probabilistic Latent Component Analysis to detect the multi-pitch content of electric-acoustic instruments. The multi-pitch activations are fed into a sound generation stage that implements sample-based playback.

We have tested this approach in three use cases, using distinct instruments. Based on our results, the technique seems not suitable to be applied to instruments with dominantly inharmonic timbre or strong presence of noise that is shared across the pitch range. However, the multi-pitch detection is stable if the harmonic content predominates in the instrument timbre.

The system runs in low-cost embedded devices with acceptable latency. An extensive test with electric guitars have shown that the system is able to detect melodic notes and polyphonic chords even during very fast performances.

This work is open to improvements, especially in the multi-pitch detection and in the sound generation stages. We plan to include music language models through Markov models in order to enforce reasonable constraints based on the music knowledge of chord progressions. We also plan to find better ways to balance the amplitude of the multi-pitch activations across the instrument pitch range. Further experiments are intended to evaluate the system multi-pitch detection capabilities and to measure the audio latency in distinct system configurations.

5. REFERENCES

[1] E. Benetos and S. Dixon. A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, 36(4):81–94, 2012.

[2] E. Benetos and S. Dixon. Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model. *The Journal of the Acoustical Society of America*, 133(3):1727–1741, 2013.

[3] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, December 2013.

[4] E. Benetos and T. Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 701–707, 2015.

[5] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra. Essentia: an audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference (ISMIR'13)*, pages 493–498, Curitiba, Brazil, 04/11/2013 2013.

[6] J. C. Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[7] C. Cannam, M. O. Jewell, C. Rhodes, M. Sandler, and M. d'Inverno. Linked data and you: Bringing music research software into the semantic web. *Journal of New Music Research*, 39(4):313–325, 2010.

[8] C. de Andrade Scatolini, G. Richard, and B. Fuentes. Multipitch estimation using a plca-based model: Impact of partial user annotation. In *ICASSP*, pages 186–190, April 2015.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society*, 39(1):1–38, 1977.

[10] A. Dessein, A. Cont, and G. Lemaitre. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 489–494, Utrecht, Netherlands, August 2010.

[11] F. Morreale and A. McPherson. Design for longevity: Ongoing use of instruments from nime 2010-14. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 192–197, Copenhagen, Denmark, 2017. Aalborg University Copenhagen.

[12] M. Mulshine and J. Snyder. Oops: An audio synthesis library in c for embedded (and other) applications. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 460–463, Copenhagen, Denmark, 2017. Aalborg University Copenhagen.

[13] M. Ono, B. Shizuki, and J. Tanaka. Sensing touch force using active acoustic sensing. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '15, pages 355–358, New York, NY, USA, 2015. ACM.

[14] O. Parviainen. Soundtouch audio processing library, July 2017.

[15] M. Puckette. Pure data: another integrated computer music environment. In *in Proceedings, International Computer Music Conference*, pages 37–41, 1996.

[16] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, and R. Borghesi. Mubu and friends - assembling tools for content based real-time interactive audio processing in max/msp. In *Proceedings of the 2009 International Computer Music Conference, ICMC 2009, Montreal, Quebec, Canada, August 16-21, 2009*. Michigan Publishing, 2009.

[17] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pages 177–180, Oct 2003.

[18] P. Smaragdis, B. Raj, and M. Shashanka. A probabilistic latent variable model for acoustic modeling. In *In Workshop on Advances in Models for Acoustic Processing at NIPS*, 2006.

[19] B. Zamborlin. Mogees. https://www.mogees.co.uk, January 2018.