RHYTHMIC EXPRESSIVENESS TRANSFORMATIONS OF AUDIO RECORDINGS: SWING MODIFICATIONS

Fabien Gouyon, Lars Fabig, Jordi Bonada

Music Technology Group - IUA
Universitat Pompeu Fabra, Barcelona, Spain
{fgouyon,lfabig,jbonada}@iua.upf.es

ABSTRACT

In this paper, we propose a computer software for modifying rhythmic performances of polyphonic musical audio signals. It first describes the rhythmic content of an audio signal (i.e. determination of tempi and beat indexes at the quarter-note and eighth-note levels, as well as estimation of the swing ratio). Then, the signal is transformed in real-time using a time-stretch algorithm. We present basic techniques provided by commercial products for swing modification and compare these to our system.

1. INTRODUCTION

Music is never performed as it is written on a score. Building blocks of music (durations, pitches, timbres) are usually represented on scores respecting rules specific to Western music. Thus, if a piece of music were to be played precisely as it is written, for instance, any A would be played with the exact same (and stable) pitch as the others, all quarter-notes would last exactly the same duration, etc. Musicians always deviate from pure mechanical performance in specific and systematic ways, which make the music lively.

Focusing solely on the temporal dimension, one can identify different sorts of timing deviations. For instance, some note durations can differ from what is written on the score, the speed of execution can vary while performing (i.e. tempo changes), there can be slight temporal "shifts" of isolated notes with respect to their theoretical locations [1]. Music performance research tells us that timing deviations do not occur at any point in time, but they are rather tightly linked to the metrical structure of rhythm [7]. The metrical structure provides “anchor points” for performing systematic timing deviations. A slowing down is typical at the end of phrases in Romantic music. In Funk music, drummers use to play quarter-notes slightly “behind the beat”. Another example is the “swing”, which originates in jazz music. The term refers to a particular metrical level: eighth-notes. For [4], one characteristic aspect of the swing is that “consecutive eighth-notes are performed as long-short patterns.” [8] defines it as a “slight delay of the second and fourth quarter-beats” (in his article, “beat” refers to half-note). See figure 1 for an illustration.

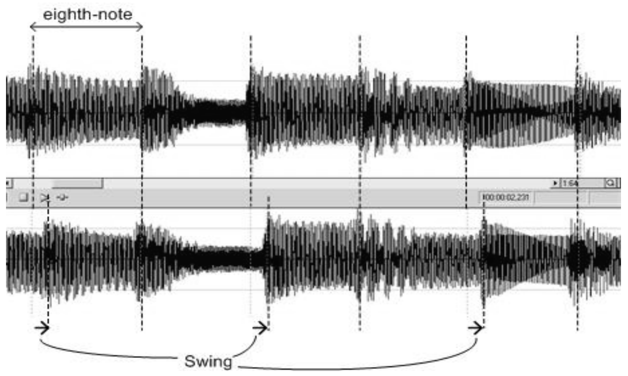

Figure 1: Illustration of an adding of swing on a polyphonic audio file.

In the remainder of this paper, we refer to two distinct terms: the *swing ratio* and the *swing factor*. The swing *ratio* refers to a mathematical expression: the duration of an eighth-note divided by that of the consecutive one. However, not any ratio is possible: as reported by [4], in actual performances, the swing ratio is function of the tempo (from 3.5:1 at very slow tempi, to 1:1, i.e. no swing, at very fast tempi; the well-known "triple-feel", 2:1 ratio, i.e. first eighth-note lasting twice the second, is present only in a specific tempo range). For a given tempo, a small subset of ratios is "natural". Audio pieces transformed without taking this into account may sound unrealistic. Therefore, we will use the term swing *factor* when referring to the continuum between "no swing" to "maximum swing"; a tempo-dependent mapping exists between the swing factor values and actual swing ratios.

2. ON SWING MODIFICATIONS

2.1. Applications for Swing Modification

Modern music production makes use of different kinds of sample libraries:

- Isolated instrument samples: Single hits of monophonic sounds that may cover the entire tessitura of an instrument, they can be played back with a sequencer, via MIDI (e.g. hits of a drum kit, bass guitar notes, etc.).
- Loops: There is no restriction regarding the polyphonic complexity and the number of notes, e.g. mix of phrases of several monophonic instruments, drum patterns. Most are (precisely) one- or two-measures long.

Tightly linked to the use of such libraries, in modern music production, there exists a need (and some tools) for modifying note timings of MIDI and audio signals. Quantization permits to adjust note rhythmic placements so that they would match precisely a score, or a template. Rhythmic expressiveness can also be modified so as to change the "groove". "Groove" resists precise definitions, but, as the "feel", it usually refers to a rhythmic phenomenon, resulting from the conflict between a fixed pulse and various timing accents played against it; or resulting from the "musician moving in non-metronomical ways" [9]. The swing (as defined above) is a particular case of groove. In music production, changing the swing is needed in the two following cases.

2.1.1. Recording post-processings

Post-processing a recording could avoid extensive and expensive recording sessions. Instead of repeating the recording asking for a slightly different feel (e.g. "yeah, that was great... but why don't you *all* go ahead and play that again, more 'swingy', you know...") one could just change it afterwards, turning a knob.

2.1.2. Swing matching

Nowadays, to save time and costs, most music producers use frequently sample libraries instead of recording all instruments with professional musicians. The typical problem a producer is confronted with appears when he wants to combine samples that contain rhythmic patterns, e.g. a drum pattern with a percussion pattern. Mixing both will sound sloppy if they have been recorded with slightly different swings.
In this context, when using isolated instrument samples, MIDI is used to control the playback of a sound. Wanting to change the swing, one just modifies the MIDI note timings to fit to a certain rhythmic template (see figure 2). Wanting to combine two different MIDI scores, one can do swing matching between both, applying the same rhythmic template to both the scores.

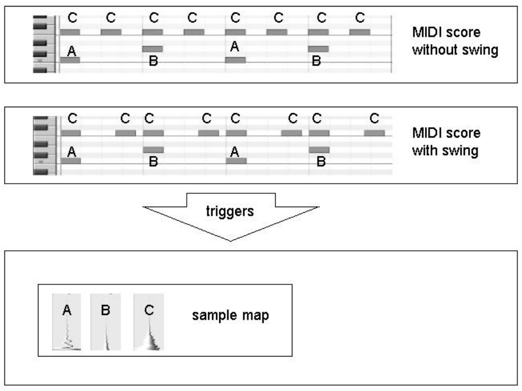

Figure 2: Example for MIDI notes triggering monophonic sounds in a sample map

On the other hand, when one wants to combine polyphonic recordings or loops, it can not be done by quantizing MIDI notes, because one has neither the MIDI score, nor the constituting isolated instrument samples. Solutions to this issue are two techniques called "Time-expansion/-compression" or "Audio Slicing". The next section explains these techniques in more detail.

2.2. Usual swing modification methods

Two techniques are commonly used: "audio slicing" (MIDI score mapping and sample sequencing) and "time-compression/-expansion". (The following explanations are based on a drum loop example.)

2.2.1. Audio slicing

This technique is based on slicing an audio recording into regions whose playback is controlled by a corresponding MIDI score (that can be edited with a sequencer):

- Segmenting: Onset detection is performed on the whole audio file to determine meaningful slice boundaries. These segments are identified by timing markers (see figure 3)
- Slicing: When the user agrees with the proposed segmentation, the audio is then chopped up into slices using the marker information.
- Instrument sample map generation: All slices are converted into a sampler compatible format, mapping the slices to MIDI notes.
- Sequencer triggering: Corresponding to time position of each slice within the original audio a MIDI score is generated that contains the time information when each slice has to be played back and triggers the sample map.

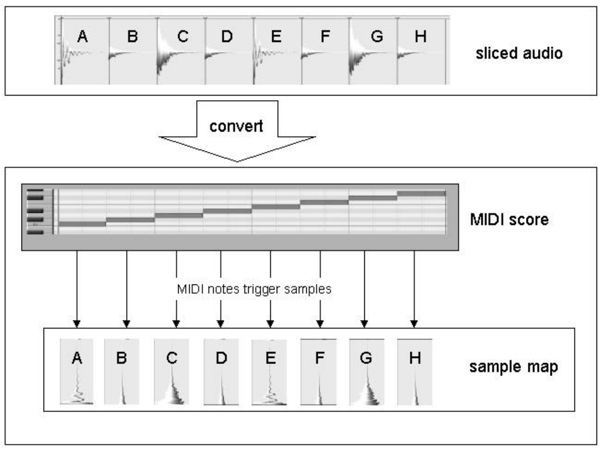

Figure 3: MIDI notes triggering polyphonic audio slices in a sample map

The swing of the original audio can then controlled by modifying the MIDI score timings (quantizing it to a different rhythmic template). Whenever a slice is moved away from its original position, problems may arise: it may overlap with others, or at certain positions where there were sound, just silence is ought to remain. Silence will cause a very unrealistic perception, especially for reverberant sounds. To avoid this critical issue, a "tail", derived from the original signal (e.g. using a reverberation technique), is usually added to each slice. That way, the decay phase of the original slice is lengthened.

A representative implementation of this method is *Propellerhead*'s *Recycle*. The user can control each stage of the algorithm. The software is aimed to work for any audio file.
A similar method is used by *Spectrasonics*. Their system "*Groove Control*" permits to change the rhythmic expressiveness of audio files shipped into loop libraries.

2.2.2. *Time-compression/-expansion*

Here, swing transformations are done as follows:

- Onset detection is performed, as in the "Audio Slicing" method.
- The onsets are used to segment the signal according to a regular grid: eighth-note positions. The quarter-note positions are also estimated.
- Portions of audio corresponding to eighth-notes can be either shortened or lengthened by a time-compression/-expansion algorithm.

A critical issue with this method is the determination of the eighth-note and quarter-note positions. The quality and rapidity of the time-compression/-expansion algorithm is also of first importance. This technique is implemented for example in *Emagic*'s *Logic Audio's "Groove machine"*. More details about this technique are given below.

2.3. Usage and sound quality compared

In our opinion, among the mentioned commercial systems, *Spectrasonics*'s *"Groove Control*" provides the best sound quality, and it permits to reach important swing factors. But it is also the less flexible: solely proprietary sample libraries can be modified. Indeed, the phases of segmentation, slicing, instrument sample map generation, sequencer triggering and sample tail generation are achieved prior to the sound library shipping (probably in semi-automatic manners). One buys both the audio loops, the isolated samples that make them up and the metadata attached to them (region boundaries, sample map, etc.).
The remaining systems, that work with any audio recording, show in general a poor sound quality, above all when used on polyphonic sounds more complex than drum loops, and at important swing ratio modifications. The main reasons are signal distortions caused by the time-scaling algorithm or unnatural sounding tails of audio slices.
A disadvantage of the abovementioned systems is that the audio signal must start on a quarter-note; otherwise, the wrong eighth-note might be shortened (i.e. the second instead of the first, which is not at all the same). Also, as swing transformation are often performed jointly with a looping of the audio, another restriction is that its length must be an integer multiple of the measure length. Therefore a sound editor has to be used to adjust the audio file boundaries. Audio Slicing systems require an additional hardware or software sampler. Finally, these systems require quite a lot of manual editing (by the user or the sample library manufacturer) to get reasonable results (e.g. in onset detection, segment the signal according to eighth-note positions).
In sum, there does not seem to exist any fully automatic software featuring professional sound quality that would permit to transform the swing of unrestricted polyphonic audio signals.

2.4. Our proposal: the Swing Transformer

The Swing Transformer is a fully automatic system and does not require neither manual editing nor a software or hardware sampler. It is based on the time-compression/-expansion technique and provides better sound quality than the standard applications.
The system consists in an offline pre-analysis stage. It does onset detection and rhythmic analysis (determination of tempi and beat indexes at the quarter-note and eighth-note levels, as well as estimation of the swing ratio, if there is any). For later use in the time-scaling algorithm, transient information is also extracted from the audio (see figure 4). Here, the distinction "onset" vs. "transient" is as follows: as detailed in [5], the rhythmic analysis input must consist in reliable note onsets. On the other hand, as explained in [2], the time-scaling algorithm apply different processings on stable and transient regions, there, the detection of non-stationarities does not have to be restricted to note onsets. In one case (rhythmic analysis) the detection of non-stationarities should be rather oriented towards "no false-alarms", in the other case (time-scaling) it should rather be oriented towards "no missed".
The transformation stage consists in analysis, time-scaling and synthesis of the audio in real-time. The time-scaling is controlled by the User Swing Ratio.

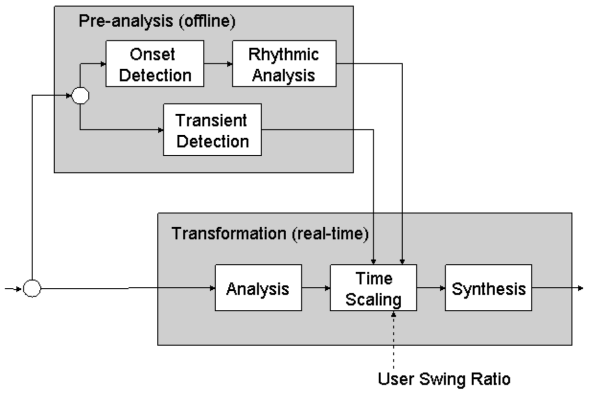

Figure 4: Block diagram of the Swing Transformer

While playing back the audio file (in a loop), the user can continuously adjust the swing ratio in real-time: one can either increase or decrease the swing.
The system provides very good sound quality for monophonic signals or polyphonic stereo mixes, it could be also extended to handle multi-channel signals.
The user can apply swing transformation in two different ways:

- *Swing Ratio modification*: the duration of consecutive eighth-notes is linearly modified as specified by the new ratio.
- *Swing Factor modification*: the user controls a "very swinging" to "not swinging" slider, and an automatic tempo-dependent mapping to corresponding "natural" swing ratios is achieved prior to the actual processing.

3. RHYTHMIC DESCRIPTION

The first step is a rhythmic analysis of the audio signal. Focusing on the swing of a musical excerpt requires the determination of two distinct metrical levels, a fast and a slow one. As swing is applied on eighth-notes, it is necessary to recognize which elements in the musical flow are eighth-notes. But this is not sufficient, one must also describe the excerpt at a higher (slower) metrical level. That is, determine the eighth-note "phase": in a group of two eighth-notes, determine which is the *first* one. Indeed, it is not at all the same to perform a long-short pattern as a short-long pattern. The existing swing ratio (if there is any) must also be estimated. (The algorithm proposed below is an extension of that detailed in [5].)

3.1. Onset detection

The audio is segmented in 11ms-long frames (hop size = frame size). Frames that have an energy superior to the average of the energy of a fixed number of previous frames (e.g. 8) are considered as frames in which an onset occurs. It is assumed that there are at least 60 ms in between two onsets. This algorithm is tuned towards no false-alarms. The transient detection for time-scaling implements a similar rationale in frequency subbands [2] and is rather tuned towards no missed. In future work, we envisage using more reliable algorithms, as e.g. [3].

3.2. IOI histogram computation

Inter-Onset Intervals (IOIs) are computed, taking into account the time differences between all onset pairs (not just successive onsets). Discrete IOI values are accumulated in a histogram which is then smoothed by convolution with a Gaussian window. That way, IOIs that are slightly different participate to the raising of the same peak in the histogram. (Indeed, it seems natural to consider that musicians cannot perform exact same time intervals, but rather perform this task with some degree of approximation.) As detailed below, the window standard deviation value is an important parameter.
Then, peak positions and heights are detected in the histogram with an N-point running window method: local maxima are detected at indexes whose values in the histogram are higher than that of their direct neighbors (N/2 on the left and N/2 on the right).

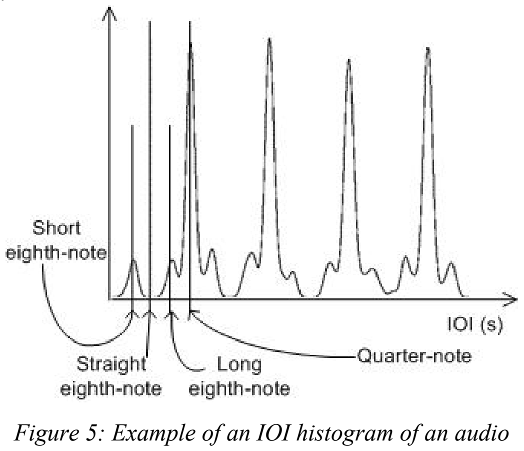

Figure 5: Example of an IOI histogram of an audio signal with a 2.7:1 swing ratio

Let us provide illustrations of the IOI histograms (see figure 5 and 6). One can verify the intuitive idea that peaks corresponding to shortened and lengthened eighth-notes are closer to the straight eighth-note length for small swing ratio than for bigger ones. The estimation of the swing ratio relies on that observation.

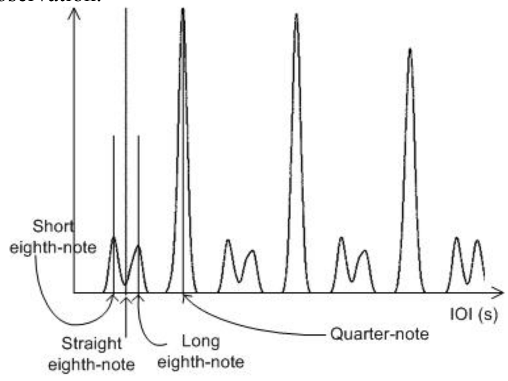

Figure 6:Example of an IOI histogram of an audio signal with a 1.5:1 swing ratio

3.3. Tick period estimation

The tick computation implements the assumption that music shows approximate *integer timing ratios between metrical pulses*. IOI histograms should show peaks at approximately harmonic positions. Therefore, the tick is computed as the *gap of the IOI histogram peak harmonic series*. To that purpose, a fundamental frequency detection method is used: the two-way mismatch error function is computed between histogram peaks and harmonic comb grid templates (see [5]). This purposely filters out the deviations from exact integer ratios between pulses that occur e.g. in the case of music that swings (as is exemplified in figures 5 and 6).
In this stage, the Gaussian standard variation is set to a medium value (e.g. 25 ms) not to be led astray by small IOIs.

3.4. Eighth-note and quarter-note period determination

The swing concerns the smallest metrical level present in the signal. As can be seen in figure 5 and 6, the amount of swing has a direct influence on the tick estimation. Let's consider the following cases:

- If the audio has a swing ratio of 1:1 (i.e. "no swing"), the computation of the tick yields the actual smallest level. That is, the tick is the eighth-note.
- If the swing ratio is 2:1 ("ternary feel"), the IOI corresponding to the straight eighth-note is not present in the signal (nor in the histogram), there are solely shortened eighth-notes (whose durations are 1/3 of that of a quarter-note) and lengthened eighth-notes (whose durations are 2/3 of that of a quarter-note). There, the tick computation yields 1/3 of the quarter-note length.
- If the swing ratio is > 2:1 (above ternary feel), the IOI corresponding to the straight eighth-note is not present in the signal, there are solely shortened eighth-notes with durations < to 1/3 of that of a straight quarter-note, and lengthened eighth-notes with durations > to 2/3 of that of a

straight quarter-note. There, as it is *restricted to integer ratios*, the computation of the tick yields either 1/3 or 1/4 of the quarter-note length.

Based on these observations, the quarter-note period is computed posterior to the tick as the maximum peak in the IOI histogram among four candidates: the tick, twice the tick, three times the tick and four times the tick. (With additional boundary restrictions: the quarter-note minimum tempo is set to 50 BPM, and the maximum to 270 BPM.)

Then the eighth-note period is computed as half the quarter-note.

3.5. Swing ratio estimation

Two different implementations of the swing ratio estimation are still under tests. They are both based on the computation of a second IOI histogram, with a Gaussian standard deviation smaller than in the previous step (e.g. 10 ms), in order to account for more peaks and also a better time precision in the peak positions.

3.5.1. Implementation 1

It is based on the computation of deviations between all peaks in the IOI histogram and integer multiples of the eight-note length. An important observation is that the deviation distribution is bimodal: one mode is around 0 (it corresponds to deviations w.r.t. quarter-note positions) and the second mode does correspond to relevant deviations. (see figure 7)

The central tendency of the second mode deviations is computed (either as the mean, the median or the mode). Eventually, the swing ratio is computed as:

$$SR = \frac{eighthnote + centraltendency}{eighthnote - centraltendency} \qquad (1)$$

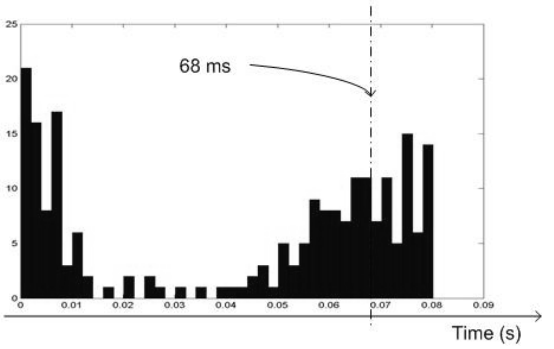

Figure 7: Distribution of the deviations {IOI histogram peaks / integer multiples of the eight-note length}. In this example, the straight eighth-note length is 161ms, the deviation central tendency of the second mode is 68 ms; this results in a 2.45:1 swing ratio.

3.5.2. Implementation 2

The basic concept in this implementation is the seeking of the best matching between IOI histogram peaks and swing templates, it resembles somehow [8]'s method. As can be seen in figure 8, templates are build as harmonic comb grids with a gap equal to the quarter-note length and with varying offsets: between the eighth-note length for the first template and e.g. 3/4 of the quarter-note length for the last one (this is directly related to the maximum boundary the user can set for swing ratio seeking). For each template, the two-way mismatch error function is computed between the grid elements and the IOI histogram peaks (see [5]), then the best template is chosen as that which yields the smallest error (i.e. which best matches the peaks).

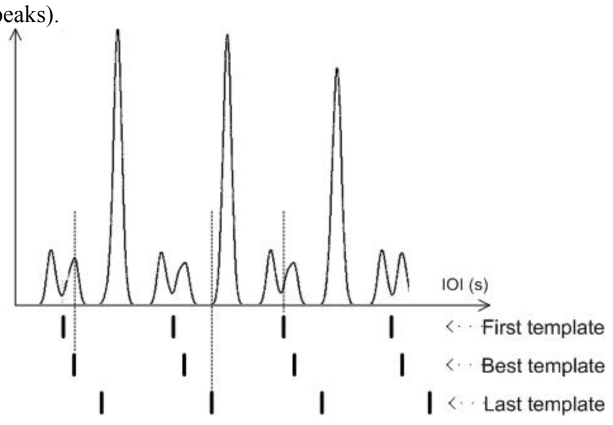

Figure 8:Illustration of the template-matching approach

3.6. Quarter-note and eighth-note position estimation

Given the quarter-note period and the onsets, the quarter-note positions are sought (and quarter-note period is slightly adjusted) so as to match to the best the onset positions (see *"Tick adjustment"* in [5]).

Logically, each quarter-note position is also an eighth-note position. The remaining eighth-note positions are simply determined as positions in between each pair of quarter-notes: half-way between two quarter-notes, adjusted with respect to the detected swing ratio.

4. SWING TRANSFORMATION USING TIME-SCALING

4.1. Swing transformation

Concretely, modifying the swing means moving onsets corresponding to eighth-notes from their original positions to different ones. In figure 9, an example is shown for an audio file that has no swing (signal in the upper half of the figure 9). Quarter-notes are depicted by a simple number ('1', '2', '3' on the figure top). The eighth-notes are indexed with i (i=1 means "in a subdivision of a quarter-note in two eighth-notes, this is the first eighth-note", and i=2 means "in a subdivision of a quarter-note in two eighth-notes, this is the second eighth-note") and their corresponding sample positions n_i. The detected Swing Ratio (SR) in the example is 1:1 (i.e. "no swing"). When the user chooses a different swing ratio (for example 2.6:1), the regions between indexes $n_{i=1}$ and $n_{i=2}$ are expanded with the time-scale factor TS_{EXP} while the regions between $n_{i=2}$ and $n_{i=1}$ are compressed with TS_{COMP}. The scaling factors for expansion (EXP) and compression (COMP) are calculated as follows:

$$TS_{EXP} = \frac{SR_{User} + SR_{User} \times SR_{Detected}}{SR_{Detected} + SR_{User} \times SR_{Detected}} \qquad (2)$$

$$TS_{COMP} = 1 + SR_{Detected} \times (1 - TS_{EXP}) \quad (3)$$

(TS>1 means signal expansion, and TS<1, signal compression).

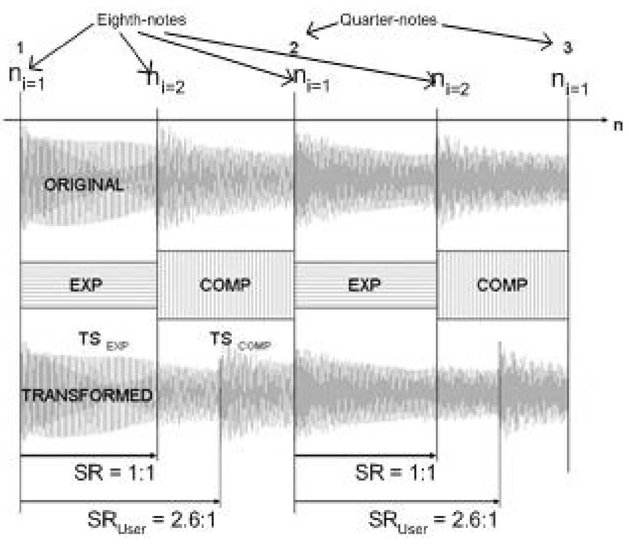

Figure 9: Swing Transformation using time-scaling of audio without swing

When the original audio signal has already swing, the processing is slightly different because the regions of expansion and compression are not equal-sized anymore. The real onset positions of eighth-notes indexed by i=2 deviate from the straight eighth-note grid. Regions to be expanded and to be compressed are adapted consequently.

4.2. The Time-Scale Algorithm

Our system uses a near-lossless frequency domain time-scale algorithm that can deal with polyphonic music [2]. It is based on the well known phase-locked vocoder. However, several improvements have been done to the basic system, starting from the fact that both analysis and synthesis frame rates are equal, thus the time scale is achieved by repeating or dropping frames. Transients are detected and not modified by the system, as well as the auditory image, which is preserved synchronizing the transients between different channels and keeping the original channel spectral phase relation bin by bin. The human auditory system response is emulated applying a discrete convolution with a variable kernel function to the frequency domain complex spectrum [6], which successfully minimizes pre-echoes and artifacts typically found in multi-band approaches

5. DISCUSSION AND FUTURE WORK

Improvements are needed on the time-scale algorithm to reduce phasing and flanging for time-scale factors above 1.3 (note that this is an important scaling factor). If we want to apply very drastic swing ratios, scaling factors superior to 1.3 are very often exceeded and sound quality may decrease. Though at large time-scale factors an analysis frame is repeated many times until the next frame is chosen, the resulting signal possibly sounds quite metallic. This can be improved by interpolation of the magnitude spectrum between subsequent analysis frames.

Another area of improvement is the handling of transients lying precisely on eighth-note positions. We assume that when the scaling factor is switched from expansion to compression (e.g. from 1.3 to 0.7) in a small group of frames belonging to a transient, this may cause a doubling of the transient (this is problematic especially for drum sounds).

Finally, it is our belief that the analysis of the deviation distribution should be further pursued. Indeed, the number of modes, the mode variances and higher moments (skewness and kurtosis) are probably representative of important information regarding diverse systematic timing deviations.

6. CONCLUSION

We proposed a system for modifying rhythmic performances of polyphonic musical audio signals: the Swing Transformer. Its implementation entailed offline rhythmic content description and real-time time-scaling. The application and source code will soon be downloadable (under GPL) through the MTG C++ Library for Audio and Music (CLAM). During the conference, the oral presentation will feature sound examples and a demonstration of the Swing Transformer.

7. ACKNOWLEDGEMENTS

We would like to thank people who admitted they couldn’t hear the difference, they helped improving the algorithms. Thanks to the CLAM team. The work reported in this paper has been partially funded by the IST European project CUIDADO.

8. REFERENCES

[1] Bilmes J. “Techniques to Foster Drum Machine Expressivity.” In Proc. ICMC, 1993.

[2] Bonada J. “Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio.” In Proc. ICMC, 2000.

[3] Duxbury C., Bello J., Davies M., and Sandler M. “Complex domain onset detection for musical signals.” In Proc. DAFX, 2003.

[4] Friberg A. and Sundström J. “Swing ratios and ensemble timing in Jazz performances: Evidence for a common rhythmic pattern.” Music Perception 19(3), 2002.

[5] Gouyon F., Herrera P., and Cano P. “Pulse-dependent analyses of percussive music.” In Proc. AES 22nd Int. Conf. on Virtual, Synthetic and Entertainment Audio, 2002.

[6] Hoek S. “Method and apparatus for signal processing for time-scale and/or pitch modification of audio signals.” Sigma Audio Research Limited. USA Patent 6266003, 2001.

[7] Honing H. “From time to time: The representation of timing and tempo.” Computer Music Journal 35(3), 2001.

[8] Laroche J. “Estimating tempo, swing and beat locations in audio recordings.” In Proc. IEEE WASPAA, 2001.

[9] Waadeland C. “‘It don't mean a thing if it ain't got that swing’ - Simulating expressive timing by modulated movements.” Journal of New Music Research 30(1), 2001.