Synthesis of a polyphonic play using only one instrument

Maxime ARBISA, Prof. Dr. Stefan WEINZIERL

March 2016 - July 2016

Contents

Introduction

English

Orchestral anechoic recordings are more and more needed for binaural synthesis and orchestral simulations in acoustic environments. The concern for anechoic recordings (pure sound, with no reverberation or reflexion) is that they can be placed in every acoustic environment afterwards, and are perfect for acoustic simulations.
However, these orchestral recordings are either rares or the studios that enable them are often too small to welcome great numbers of musicians at the same time. Thus, techniques that synthesize a whole instrument section from one anechoic solo recording have to be found. Moreover, a great audio quality is required for binaural synthesis, and the audio rendering of the traditional chorus effect, that makes an instrument sound more like an ensemble, is often too poor for such purposes.
In this paper, we explore several high audio quality techniques to generate an instrument section starting from a solo anechoic recording, and compare their performances according to the instrument section (snare or violin) we want to create. These techniques stand on a phase vocoder approach and improve the Pitch-Time-Amplitude (PTA) algorithm. The time domain PSOLA method is also evaluated.
Finally, the anechoic instrument sections are placed in a virtual acoustic environment, created by room acoustical simulation and dynamic binaural synthesis (giving to the listener the impression of sitting in a real concert hall listening to a full orchestra) and are tested with subjective listening tests.

Français

Les enregistrements anéchoïques d'orchestre sont de plus en plus demandés pour des applications de synthèse binaurale et des simulations dans différents environnements acoustiques. L'avantage des enregistrements anéchoïques (son pur, sans réverbération ni reflexion du son) est qu'ils peuvent être ensuite placés dans n'importe quel environnement acoustique.
Cependant, ces enregistrements sont rares car les studios les permettant sont souvent trop petits pour accueillir un grand nombre d'instrumentistes en même temps. Ainsi, il est nécessaire de trouver des méthodes permettant de synthétiser une section complète d'instruments à partir de l'enregistrement anéchoïque d'un seul de ces instruments. De plus, la synthèse binaurale requiert des enregistrements de très bonne qualité sonore, dont l'effet de 'chorus', qui permet à un instrument de sonner comme un ensemble est incapable.
Dans ce rapport, nous étudions plusieurs techniques en comparant leurs performances suivant la section d'instruments (violons ou caisses claires) désirée. Ces méthodes, dont les résultats audios sont de haute qualité, utilisent le vocodeur de phase et améliorent sensiblement l'algorithme de base: l'algorithme Pitch-Time-Amplitude (PTA). La méthode TD-PSOLA, opérant dans le domaine temporel, est aussi étudiée.
Enfin, les enregistrement anéchoïques d'orchestre obtenus sont évalués par des tests d'écoute. Les enregistrements sont alors simulés dans différentes salles de concert (nos environnements acoustiques), et la synthèse binaurale apporte à l'auditeur la sensation d'être assis en plein milieu de la salle.

Keywords

Analysis, Anechoic, Binaural synthesis, Chorus, Instrument section, Orchestral effect, Phase locking, Phase vocoder, Pitch-Time-Amplitude (PTA), PSOLA, Synthesis, Transient detection.

Acknowledgement

Results

The code, bibliography and audio results are available in open source on GitHub:
`https://github.com/MaximeArbisa/AudioSynthesisFinal`

Many thanks

I would like to thank Prof. Dr. Stefan Weinzierl and Dmitry Grigoriev for their supervision and help during this study.
Dmitry is a student who was doing his master thesis on the same subject. Thanks to him for his support and his constant good mood.

Part I
Synthesis of a polyphonic play

1 Abstract

In this part, we propose to study and implement several algorithms that generate orchestral effects based on a solo anechoic recording.
J. Meyer [1] has noted that one of the effect of an instrument section lies in the broadening of the peaks at harmonic frequencies. With instrumental ensembles the 3 dB bandwidth of the spectral peaks deviate up to ±20 cents from the nominal frequencies. J.Pätynen and al. [2] noted that another effect was small temporal fluctuations in instruments' onsets, and that these ones were following a normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma^2 \simeq 40ms$.
Thus, it seems that the simulation of an instrument section lies in the fact that two different instruments would never play at the exact same pitch and at the exact same time.
In the first section, we'll describe already used methods and recall tools (mainly the phase vocoder) that will be useful for our simulations. The articles [3], [4] will be partly used for this theoretical part.
In the second section, we'll implement a technique developed by J. Pätynen and al. in [2], called PTA (Pitch shift - Time difference - Amplitude modulation) algorithm and using a phase vocoder approach. Then, we'll improve the results by working on *vertical phase coherence* and *transient processing*, developed respectively by J.Laroche and M. Dolson [3], and A. Röbel in [5].
Then, starting from this technique, we'll work on alternative methods and approaches using onset detection, pitch detection, and a time-domain method this time: the TD-PSOLA method (Time Domain - Pitch Synchronous OverLap and Add), developed by J. Laroche and E. Moulines in [6].
In a last part, we'll explain how our own database of recordings were conducted (violins and snares), and how we excerpted the parameters of our simulations. Finally, we discuss which simulations must be chosen depending the instrument section we want to create, and subjective listening tests were conducted to asset our implementations.

2 Latest developments and recap on the phase vocoder

2.1 Latest developments

Some techniques, such as the audio effect *chorusing* [7] already exist and are widely used in musical industry' devices when a single instrument is needed to sound more like an ensemble.
The *chorus effect* lies on the fact that in an ensemble, two instruments would never play the exact same sound, at the exact same pitch. Moreover, their pitch difference is constantly varying in time. This method is based on a delay line whose tap point is modulated over time.
Thus, this method consists in creating a signal y, that is the original signal x modulated by a time-varying delay $\tau(t)$, such as:

$$y(t) = x(t - \tau(t))$$

Thus, applied to numeric signals sampled every ΔT, and choosing for example a linear delay $\tau(t) = \alpha * t$, we'll get:

$$y_n = y(n\Delta T) = x(n\Delta T - \tau(n\Delta T)) = x(n\Delta T(1 - \alpha))$$

We can see that this modulation causes a local re-sampling of the signal, that is now sampled at $F_s' = F_s/(1 - \alpha)$ if x was first sampled at F_s. In general, for a delay $\tau(t)$, we get a local re-sampling of:

$$F_s'(t) = F_s(1 - \frac{d\tau}{dt}(t))^{-1} \quad (1)$$

Thus, when we play the generated signals at the original constant F_s, we get variation in pitch and in time for each instrument as wanted.
We implemented two different delays: a periodic delay and a random one, as used by J. Pätynen and al. [2]. The periodic delay was set as $\tau(t) = \tau_0 + \alpha sin(w_0 t)$. The local sampling frequency F_s' was then such as, following (1):

$$F_s'(t) = F_s(1 - \alpha w_0 cos(w_0 t))^{-1} \approx F_s(1 + \alpha w_0 cos(w_0 t)) \quad (2)$$

We have to set w_0 such as to get low variations in the signal. We can set a period $T_0 = 5s$ that correspond in samples to $w_0 = 1/5F_s$. α corresponds to the amplitude of the delay, and is responsible for the pitch and time modification of the signal, as not only the pitch but also the playback rate with which one we read the signal is changed. Thus, if $F_s' \leq F_s$, when played at F_s the signal will be higher pitched and read faster. As we want a pitch shift of ± 20 cents [1], corresponding to a variation of 10^{-2} around F_s in (2), that is to say $\alpha w_0 \leq 10^{-2}$, thus $\alpha \leq 5.10^{-2}.F_s$. Finally, τ_0 corresponds to the offset for every instrument and is picked following a random distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma = 30ms$.
The second delay line proposed by [2] is picked randomly following a normal distribution, with a modulation depth of 1.3ms, and low-pass filtered at 3Hz. Offsets randomly picked between 0-25ms were then added for each instrument. The computational cost of this method was however much higher than the sinusoid delay.
These two chorus effects are implemented in the function *chorus.m*.
However, the chorus effect presents numerous artifacts. The most famous, its characteristic vibrato, or beating artifact, happens according to U. Zölzer [7] when the offset between simulated instruments is null and the delay line is in between 0-3ms. Interferences between close frequencies (when $\Delta f \approx 1/100Hz$) are then often audible, and that why an offset between instruments is essential.
Moreover, the effect is really dependant to the amplitude of the delay. Lower values were considered to introduce too small differences and higher values for the modulation were heard as unnaturally 'fuzzy' audio results. This is caused by the local resampling which acts as a expansion/contraction of the frequency axis of the signal, and that can lead to an aliasing effect, especially for signals with high frequencies.
Finally, the parameters, resulting in time and pitch modifications are too dependant one from each other, reducing our freedom of action for the desired simulations.
For binaural synthesis, as we're working with separate mono-sources, every instrument simulated has to have a good audio quality. With the parameters we chose for the sinusoidal delay, the overall chorus effect was good, but taps/interferences could definitely be heard when listening to the separate sources.
Thus, although widely used in music industry, this effect is yielding fair audio quality results that are not good enough for *auralisation* and *binaural synthesis* issues. That's why we're introducing the *phase vocoder*, that can generate high-quality results that are needed in our applications.
The audio results are in the folder 'Violin', as well as all the other results in this part. The original recording from which we are doing our tests is a former anechoic recording belonging to TU and named 'violin.wav'. 'chorus_12.wav' and 'chorus_1.wav' are respectively the chorus effect of a section of 12 violins and just one of them.

2.2 Phase Vocoder, a powerful tool in signals modification and synthesis

2.2.1 Description of the approach

The *phase vocoder* approach is more and more commonly used in audio modifications thanks to its production of high quality audio results.
Its approach uses the *Short Term Fourier Transform (STFT)* and consists in a sequence of analysis - modification - resynthesis of the signal.
The analysis consists in computing the STFT of the signal to divide the signal into frames and compute a *Fast Fourier Transform (FFT)* on each one of them to get their spectral representation.

Spectral modifications are then applied to the FFT of each one of these frames, and the signal is finally re-synthesised through an inverse FFT (iFFT) and overlap-add of all of its frames.
The following diagram explains the process enhanced by the phase vocoder:

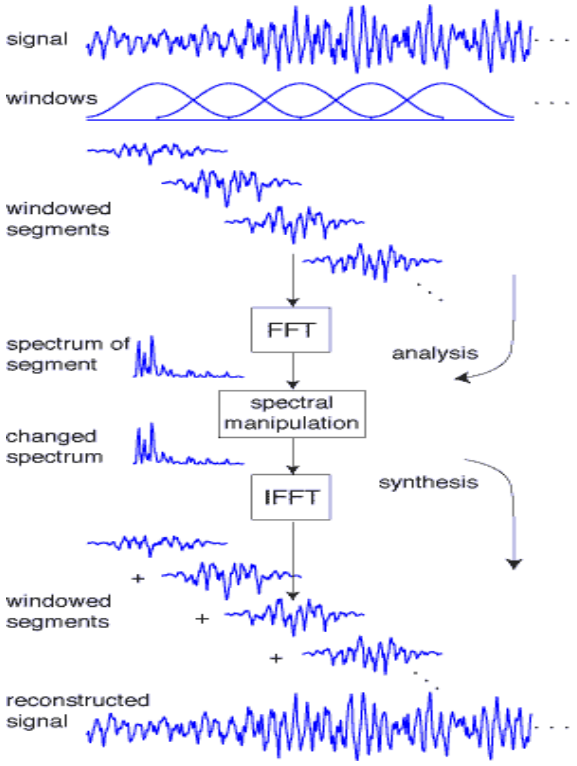


Figure 1: Phase vocoder approach

First, the analysis of the signal is done computing a *Short Term Fourier Transform* of the signal. The signal is divided into frames that are portions of the signal windowed around analysis marks t_a, usually set as $t_a^u = u.R_a$, where R_a is constant and the *analysis hop factor*. The *analysis window* w_a is a Hanning window of length N (signals' portions are thus of length N) in our case. A FFT is then performed on each one of this frame, to obtain a time-frequency representation of the signal. The Fourier Transform has a length, or a spectral precision of N_{fft}, that corresponds to the N_{fft} *vertical channels* of our phase vocoder. In order to prevent the problem of *aliasing*, we have to have $N \leq N_{fft}$. In the rest of this paper, we'll take $N = N_{fft} = 2048$, and the hop factor will be set to $R_a = \frac{N}{4}$, guaranteeing an overlap of 75%.
Below a formula for the STFT at the analysis time t_a^u, in the vocoder channel k:

$$X(t_a^u, \Omega_k) = \sum_{n \in \mathbb{Z}} x(n + t_a^u) w_a(n) e^{-jn\Omega_k} \tag{3}$$

where x is the original signal, w_a is the analysis window (in this paper a Hanning window), and this window is chosen from now on as symmetric around zero. $\Omega_k = 2\pi \frac{k}{N_{fft}}$ is the center frequency of the kth vocoder channel.
The following image shows the STFT of a piano playing a music scale:

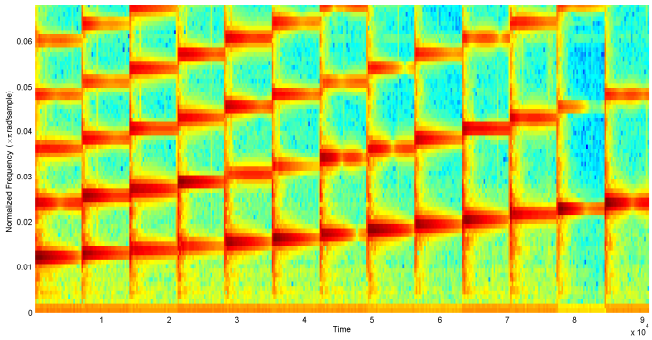


Figure 2: STFT of a music scale played by a piano

The diagram clearly exhibits a rise in the fundamental frequencies as the piano plays the music scale, and the partials are also shown for each fundamentals.
Once the STFT is computed, we obtain for each frame its FFT $X(t_a^u, \Omega_k)$. We can then compute the desired spectral modifications to get the new FFT's and resynthesize the signal on the synthesis marks t_s, usually defined as $t_s^u = u.R_s$. The specific spectral modifications we'll use will be described in detail in the next section.
After these modifications, we obtain the modified FFT's $Y(t_s^u, \Omega_k)$. The resynthesis of the signal is then obtained by inverse Fourier transforming the modified $Y(t_s^u, \Omega_k)$, that will give us for each frame a synthesised signal y_u. These signals are then windowed using a synthesis window w_s and are summed all together using the overlap-add method to get the final synthesised signal y, as shown in figure 1:

$$y(n) = \sum_{u \in \mathbb{Z}} y_u(n - t_s^u) w_s(n - t_s^u) \text{ with} \tag{4}$$

$$y_u(n) = FFT^{-1}[Y(t_s^u, \Omega_k)(n)] = \frac{1}{N_{fft}} \sum_{k=0}^{N_{fft}-1} Y(t_s^u, \Omega_k) e^{jn\Omega_k}$$

If no modification is done ($i.e\ t_a^u = t_s^u\ \forall u\ (R_a = R_s)$ and $Y(t_s^u, \Omega_k) = X(t_a^u, \Omega_k)$), then the synthesised output y is equal to the original one x if and only if:

$$\chi(n) = \sum_{u \in \mathbb{Z}} w_a(n - t_s^u) w_s(n - t_s^u) = 1 \forall n \tag{5}$$

In the following, we'll work with $w_a = w_s = hanning(N)$. χ was computed with these two windows thanks to the function *ola.m* and gives for an overlap of 75% ($R_s = N/4$):

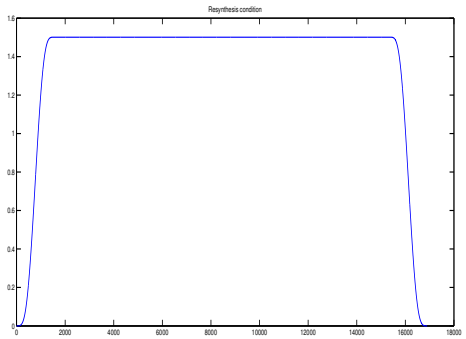


Figure 3: Perfect resynthesis condition

We can see that the $\chi = 1$ for these parameters, ignoring a multiplication factor, and apart from both sides of the signal. By setting $w_s = 1/max(\chi).w_s$, we have our condition fully respected. This setting will be used for the rest of this paper.

2.2.2 Consistency of the synthesised STFT

In the previous section, we've shown that the final signal y is synthesised from the modified STFT $Y(t_s^u, \Omega_k)$.
However, this STFT does not correspond, in practice, to any actual signal. In particular, if you compute the STFT of the synthesised signal y, you would likely not find $Y(t_s^u, \Omega_k)$.
Thus, the sequence of STFT frames must satisfy strong consistency conditions, because they correspond to *overlapping* segments, to guarantee coherence and consistency in the synthesised signal. To achieve this, a lot of techniques and conditions will be described in the following sections.
In the meantime, we present here a tool developed by J. Laroche and M. Dolson in [3] that will measure the consistency of the modified STFT and thus the quality of our synthesis:

$$D_M = \frac{\sum_{u=P}^{U-P-1} \sum_{k=0}^{N_{fft}-1} [|Z(t_s^u, \Omega_k)| - |Y(t_s^u, \Omega_k)|]^2}{\sum_{u=P}^{U-P-1} \sum_{k=0}^{N_{fft}-1} |Y(t_s^u, \Omega_k)|^2} \tag{6}$$

where $Z(t_s^u, \Omega_k)$ is the STFT of the synthesised signal and $Y(t_s^u, \Omega_k)$ is the modified STFT. U is the total number of frames, and we don't take into account the P first and last frames, to avoid errors due to missing overlapped segments in the resynthesis formula (as we saw in the extrema (first and last frames) of χ (fig. 3), where $\chi \neq 1$.
The better the consistency, the closer Y is to Z and so the closer D_M is to 0. In the rest of this article, our different techniques will use this measure to verify the consistency of the STFT generated. The implementation of this measure can be found in function *STFTConsistency.m*.
The phase vocoder offers various transformations by modifying the STFT of the signals treated. In the following sections, we'll talk about the most relevant ones that we're going to use in our study. The main actions we'll deal with in the next sections are time-scaling and pitch-shifting modifications.

2.2.3 time-scaling modification

The main theory related to this section is written using both articles written by J. Laroche and M. Dolson, and J. Laroche and E. Moulines in [3] and [4] respectively. A further explanation of the phase vocoder time-scaling modification can be found in the annex. An implementation of the time-stretching modification is proposed in the function *timestretch.m*.
The time-scaling modification consists in stretching the synthesis time-marks t_s^u, while preserving locally the spectral content.
The first step consists as usual in an analysis through the STFT of the signal. The signal is analysised with analysis frames spaced by Δt_a^u. Usually, and in our cases, $t_a^u = u.R_a$, that gives us a constant interval between analysis frames of $\Delta t_a^u = R_a$.
Then the synthesis frames will be stretched and separated by Δt_s^u, also often set such as $t_s^u = u.R_s, \Delta t_s^u = R_s$. To achieve the time-scaling modification, we'll just have to synchronize the phases of the generated STFT $Y(t_s^u, \Omega_k)$ between two consecutive synthesis frames, thanks to the so-called instantaneous frequency $\hat{\omega_k}(t_a^u)$, which is the local spectral content in the kth channel at time t_a^u.
The instantaneous frequency between two consecutive analysis frames can be computed thanks to the phase of the analysis STFT $X(t_a^u, \Omega_k)$, denoted $\angle X(t_a^u, \Omega_k)$, using the *phase unwrapping* method. Below is only the way to compute it. The origin of this method and the explanation of the hypothesis under which one it can be applied are in the annex.
The instantaneous frequency for every channel k, $\hat{\omega_k}(t_a^u)$, between two consecutive analysis frames placed at t_a^u and t_a^{u-1} is first computed using the *heterodyned phase increment*, that is unwrapped, as follows:

$$\Delta\Phi_k^u = \angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) - \Delta t_a^u \Omega_k \tag{7}$$

where $\Delta t_a^u = t_a^u - t_a^{u-1}$.
Then, we take the first determination of Δt_a^u between $[-\pi, \pi]$: $\Delta_p \Phi_k^u = \Delta\Phi_k^u - 2n\pi$, such as $|\Delta_p \Phi_k^u| < \pi$.
Finally, we get for the instantaneous frequency $\hat{\omega_k}(t_a^u)$ for every kth channel with:

$$\hat{\omega_k}(t_a^u) = \Omega_k + \frac{\Delta_p \Phi_k^u}{\Delta t_a^u} \tag{8}$$

Assuming our signal is a sum of sinusoids (M^{c}Aulay and Quatieri model):

$$x(t) = \sum_{i=1}^{I} (t) A_i(t) e^{j\Psi(t)}$$

and that only one sinusoid falls in a channel, then the instantaneous frequency in channel k is the instantaneous frequency of the sinusoid i that fell in that channel. Plus, the *heterodyned phase increment* corresponds to the small phase shift resulting from this sinusoid i in channel k (represented by $\hat{\omega_k}(t_a^u) = \omega_i(t_a^u)$) being close but not equal to Ω_k.
To achieve the time-scaling, we have to synchronize the phases of the new STFTs $Y(t_s^u, \Omega_k)$ between the synthesis frames, as suggested by the figure excerpted from [4]:

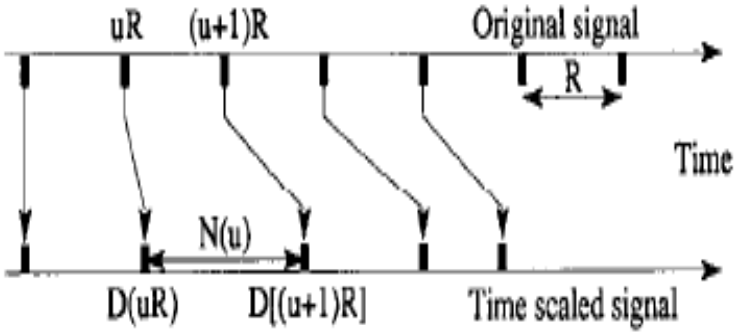


Figure 4: Time-scaling operation
R = R_a, D(uR) = t_s^u and N(u) = Δt_s^u

The amplitude of the synthesised STFT is set as the same as the analysis one:

$$|Y(t_s^u, \Omega_k)| = |X(t_a^u, \Omega_k)| \tag{9}$$

And the new phases are calculated deriving the instantaneous frequency, using $\omega(t) = \frac{d\Phi}{dt}$:

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + \hat{\omega_k}(t_a^u) \Delta t_s^u \tag{10}$$

We have a phase correction of $\hat{\omega_k}(t_a^u)\Delta t_s^u$ to synchronise the phase between synthesis frames. Plus, we suppose that the instantaneous frequency remains constant between two consecutive synthesis frames (that suppose that Δt_s^u is small enough). Finally, as $\hat{\omega_k}(t_a^u)$ remains unchanged during the entire operation, we've changed the length of the signal while keeping our spectral content.
However, this transformation stands on three hypothesis, that we'll be dealing with during the entire study, and that is explained further in the corresponding annex:

- First, that the signal is a sum of sinusoids (M^{c}Aulay and Quatieri model) whose characteristics (amplitude and frequency) vary slowly in time. That enables us in part to derive the phase on the analysis window length, as we did for example in (10). It is the *stationary hypothesis*.

- Then that the length of the analysis window $N = N_{fft}$ is big enough so that only one sinusoid of the signal can be at the same time in the same channel. For a monophonic signal (with which one we work on anechoic records), at frame t_a^u, we'll have a sound composed of a fundamental frequency f_0 and its partials $f_h = hf_0$. If we want to separate them, we have to guarantee that the cutoff frequency of the analysis window ω_h is less than half the spacing between two consecutive harmonics, $f_0/2$. This is the *narrow-band* hypothesis. Thus, for a Hanning window of length N, we want: $\omega_h = 2\pi f_h$, $f_h = 2/N \leq f_0/2$, so $N \geq 4/f_0$.
  Let's assume we detect sounds down to A2 (110Hz) for the violin, then $N \geq 4 * F_s/f_0 \simeq 1600$ samples is good with our setup $N = 2048$. Be careful though, because if N is too big, the stationary hypothesis may not be true enough on the analysis window length.
  Plus, as $N = N_{fft}$, a greater N, a greater N_{fft} and so a better spectral precision to read the frequencies.

- Finally, for the unwrapping technique to be verified, we have to ensure $\omega_h.R_a < \pi$, or phase offsets of π can appear in the adjacent channels (the explanation of this condition can be found in the annex). It is the *unwrapping* hypothesis. For a Hanning window, this can be expressed as $2\pi\frac{2}{N}R_a < \pi$, i.e $R_a < N/4$ which corresponds to an overlap of at least 75% between the analysis windows. This hypothesis meets with the one created by the sub-sampling of the STFT. Indeed, instead of computing the STFT every samples, we compute it every $\Delta t_a^u = R_a$ samples to reduce the computational cost of the STFT. This sub-sampling will create duplications of the spectrum that can create aliasing. To have a perfect reconstruction, we must have $B_w \leq F_s'/2 = F_s/(2\Delta t_a)$, with B_w bandwidth of our loss-pass analysis window. Here, as working with a Hanning window, we'll have:

  $$B_w = \frac{C_w}{N}F_s = \frac{2}{N}F_s \text{for a Hanning window}$$

  So we need $\Delta t_a \leq N/4$ that implies again an overlapping of at least 75% between the analysis frames.

2.2.4 Pitch shifting

The operation of pitch shifting can also be realized with the phase vocoder, this time by multiplying the instantaneous frequencies in every channel by the desired factor, and resynthesize the signal using a sum-of-sinusoids model (M^{c}Aulay and Quatieri model) with the amplitudes and the new frequencies.
Another method and the one we will be using in the next sections is the *time-stretch* method. It consists simply in resampling the original signal by a factor of $d \in \mathbb{Q}$ closest to the desired detune. If the signal is read at the former sampling frequency F_s, resampling will cause a pitch-shift but will also change the length of our signal. The signal will endure a compression or a dilatation of d.
From this point, we can use the time-scaling method presented above to stretch the signal without modifying the new spectral content. We set $t_a^u = uR_a$ and $t_s^u = uR_s$, with R_a and R_s constant, and such as $R_s/R_a = 1/d$.
Finally, the signal will be back to its original length, but the pitch would have changed by a factor of d. An implementation can be found in the function *pitchshift.m*.

We've explained the use as well as the hypothesis that rule the phase vocoder approach, and are now ready to use it to simulate a full section of violins out of one anechoic recording.

3 Pitch-Time-Amplitude (PTA) algorithm - method based on a phase vocoder approach

The aim of this section is to study an algorithm proposed by J.Pätynen and al. [2] and called PTA (Pitch-shifting - Time fluctuations - Amplitude modulation).
As stated in the introduction, the effect of an instrument section lies in the broadening of the peaks at harmonic frequencies up to ±20 cents from the nominal frequencies [1]. J.Pätynen and al. [2] have also noted that another effect was small temporal fluctuations in instruments onsets, and that these ones were following a normal distribution $\mathcal{N}(0, \sigma^2)$, where $\sigma \simeq 40ms$ is the standard deviation.
Thus, it seems that the simulation of an instrument section lies in the differences between the pitch, the onsets and the amplitude between the different musicians. J. Pätynen and al. [2] used these observations to generated M musicians by adding pitch, temporal and amplitude fluctuations based on the original recording.
The method is using phase vocoder techniques, and is explained in the figure shown below, and excerpted from [2]:

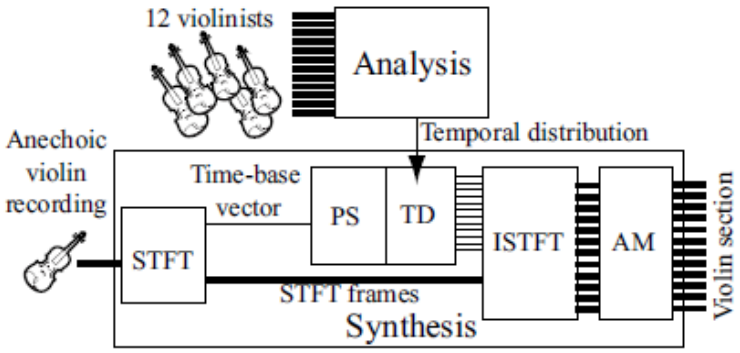


Figure 5: Pitch-Time-Amplitude (PTA) algorithm

The method is separated in three main modifications brought to the original recording to generate the musicians:

- The first one is a simple pitch-shift around the original recording that symbolizes the differences of intonation between different violinists in one orchestra.
- The second one is the use of Metropolis-Hastings sampling to add time differences/fluctuations between the violins.
- Finally, an amplitude modulation is set to simulate the varying playing dynamics between musicians and between consecutive notes.

3.1 Pitch shifting

The pitch-shift operation is done using the phase vocoder and realizing an inverse time-stretching operation. In [2] and in our study, we're using an analysis Hanning window w_a of length N = 2048 samples. The synthesis window is set as $w_s = w_a$, and $N = N_{fft}$ as stated in the previous sections, to fulfill the *no-aliasing* and *narrow-band* hypothesis. Moreover, $t_a^u = uR_a$ (constant interval between analysis frames), where $R_a = \frac{N}{4}$ to have an overlap of 75% and verify the *unwrapping* hypothesis.
In [2], pitchs are varied ±10 cents around the original one (to get the final ±20 cents observed by Meyer [1]). 100 cents correspond to the gap between two consecutive frequencies, i.e an interval of $2^{\frac{1}{12}}$ for the tempered scale.

Thus, a change of n cents will result in a pitch shift corresponding to: $2^{\frac{n}{1200}}$, so for ± 10 cents to $2^{\frac{n}{1200}} = 1.0058$, so approximately $\pm 0.5\%$ from the original pitch.
Thus, for every wanted musician m, we pick a random pitchshift factor following the normal distribution $\mathcal{N}(1, 0.005)$, and do the corresponding pitch shifting using an inverse time-stretch method.
The inverse time-stretching approach consists, as suggested in its name, in first time-stretch and then resample the signal.
Thus, for a musician m, a pitch-shift factor following the normal distribution $\mathcal{N}(1, 0.005)$ is picked, and we choose the factor $d^{(m)} \in \mathbb{Q}$ closest to the desired detune.
Then, the inverse time-stretching method consists setting the synthesis frames, such as $t_s^u = uR_s$, with $R_s = d^{(m)} R_a$, and computing the new STFT's $|Y(t_s^u, \Omega_k)|$ using the time-scaling approach:

$$\begin{cases} |Y(t_s^u, \Omega_k)| = |X(t_a^u, \Omega_k)| \\ \angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + \hat{\omega_k}(t_a^u)\Delta t_s(u) \end{cases} \tag{11}$$

where $X(t_a^u, \Omega_k)$ is the STFT of the original signal, and $\Delta t_s(u) = t_s^u - t_s^{u-1} = R_s$.
At this point, we have a signal compressed or expanded by a factor $d^{(m)}$ whose spectral content was kept unchanged.
Then, a resampling of a factor $\frac{1}{d^{(m)}}$ will change the pitch and the length of the signal, back to its original one, and with the desired detune.
The inverse time-stretching approach is wise here, as we're computing only one analysis STFT for the M instruments desired, and generate M synthesised STFT. If we have used the normal time-stretching approach, we would have to generate M analysis STFT, whose computations depend on the lengths of the M resampled signals, and then compute M synthesised STFT.

3.2 Time Differences using Metropolis-Hastings sampling

More than just a pitch-shifting, J. Pätynen and al. [2] introduce time differences on each synthesised frames to simulate time variation between the M different players. This will complete the pitch-shifting to create more "volume" in the sound to sound more as an ensemble.
The time-fluctuation is achieved by setting:

$$t_s^u = t_s^u + \delta(u) = uR_s + \delta(u) \tag{12}$$

adding a little time fluctuation in the synthesis time-instants. If $\delta(u) > 0$, the synthesis frame originally at t_s^u will be moved forward to $t_s^u + \delta(u)$, giving the impression that the musician is playing slightly behind the tempo.
The question now is which model we have to choose to simulate at best the time fluctuation.
J. Pätynen and al. in [2] chose a random Markov chain, following the Metroplis-Hastings sampling. This method, based on a Markov Chain Monte Carlo method, presents the advantage that picked samples depend on the previous state as in every Markov Chain, and can emulate the effect of a musician playing slightly behind the tempo, and catching the tempo at the next moment, or vice-versa. That is similar to the behavior of the traditional chorus effect but without its characteristic vibrato, and with more manoeuvrability in the parameters.

3.2.1 Metropolis-Hastings sampling - Theory

The explanation of the Metropolis-Hastings sampling given here is excerpted from an article of S. Chib and E. Greenberg in [8].
Classical simulation techniques usually generate independent successive samples. With Markov chains, and more specifically the Metropolis-Hastings chain here, we include dependency between observations. This dependency is defined by the so-called the *candidate-generating density*, and noted $q(x, y)$ such as $\int q(x, y)dy = 1$. This density is just saying that when a process is at a point x, the density generates a value y from $q(x, y)$. Let's define $\pi(.)$ as the density of the distribution with which one we're picking our samples. In most of the cases, we'll find for some x and y:

$$\pi(x)q(x,y) > \pi(y)q(y,x) \tag{13}$$

which means that the process of sampling moves from x to y too often and from y to x too rarely. Note that the problem can be seen the other way as it is symmetrical.
To counterbalance this phenomena and reduce the moves from x to y, we'll introduce a so-called *probability of move* $\alpha(x,y) < 1$, which is the probability that the move is made, to reduce the number of transitions from x to y. Thus, the transition from x to y will now be set by $q'(x,y) = q(x,y)\alpha(x,y) = p_{MH}(x,y)$, defined as the *Metropolis-Hastings candidate-generating density.*
What we want to have now, to balance (13) with the *probablity of move* $\alpha(.)$ is:

$$\pi(x)q(x,y)\alpha(x,y) = \pi(y)q(y,x)\alpha(y,x)$$

But as we want more transition from y to x, considering the inequality (13), we set $\alpha(y,x)$ as large as possible, so to its upper limit 1.
From the previous equations, we have:

$$\alpha(x,y) = \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}$$

Finally, we get:

$$\alpha(x,y) = \begin{cases} min[\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1] \text{ , if } \pi(x)q(x,y) > 0 \\ 1, \text{ otherwise.} \end{cases} \tag{14}$$

The Metropolis-Hastings algorithm is then defined as follows, with the arbitrary value of the first sample $x^{(0)}$ and N the length of the desired chain:

Algorithm 1 Metropolis-Hastings algorithm

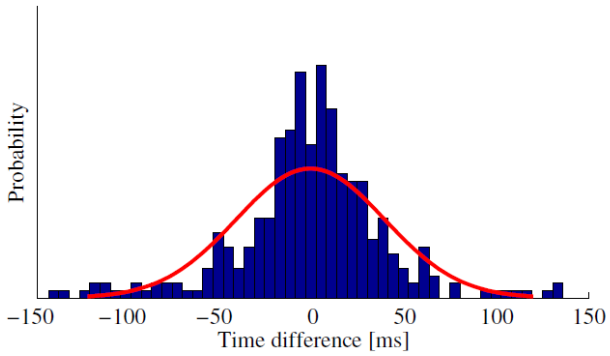
Steps

1: **for** $j = 0, 1, ..., N-2$ **do**
2: Generate y from $q(x^{(j)}, .)$ and u from $\mathcal{U}(0,1)$
3: **if** $u \leq \alpha(x^{(j)}, y)$ **then**
4: set $x^{(j+1)} = y$
5: **else**
6: set $x^{(j+1)} = x^{(j)}$
7: **end if**
8: **end for**
9: Return the values $x^{(0)}, x^{(1)}, ..., x^{(N-1)}$

Where $\mathcal{U}(0,1)$ is the *standard uniform distribution* whose probability density equals 1 in $[0,1]$.

3.2.2 Application

By setting contact microphones (to avoid noise between nearby instruments) on different violins in the same section, and picking their onsets, J. Pätynen and al. have noticed that the distribution of the onsets, thus the temporal fluctuations were following a normal distribution $\mathcal{N}(\mu, \sigma^2)$ of mean $\mu = 0$ (on average, the section is in the tempo) and with a standard deviation $\sigma \simeq 40ms$. The distribution is shown below:



σ	μ	2.5 %	25 %	50 %	75 %	97.5 %
40.35	0.00	-101.85	-18.34	0.00	19.10	84.20

Figure 6: Distribution of the onsets temporal differences, from [2]

Thus, the Metropolis-Hastings sampling algorithm can be simplified a lot. Indeed, for a normal random distribution, the *candidate-generating density* is symmetric: $q(x, y) = q(y, x)$ as samples are picked randomly, and $\pi(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} > 0$.
Thus the expression of $\alpha(x, y)$ in (14) is:

$$\alpha(x, y) = min[\frac{\pi(y)}{\pi(x)}, 1] = min[\frac{e^{-\frac{1}{2}(\frac{y}{\sigma})^2}}{e^{-\frac{1}{2}(\frac{x}{\sigma})^2}}, 1] \tag{15}$$

Finally, as it is a random normal distribution, in step 2 of the Metropolis-Hastings algorithm, y is just taken randomly in $\mathcal{N}(0, \sigma^2)$.
Below a density for a symmetric distribution (the same as our normal distribution):

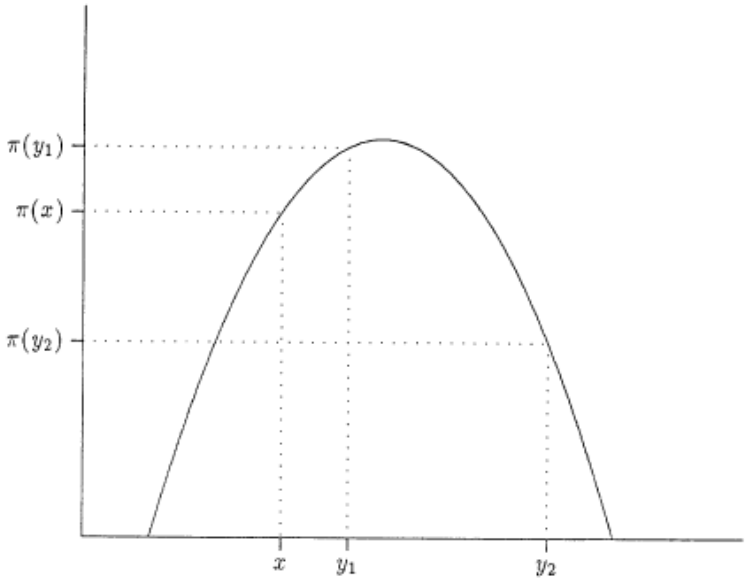


Figure 7: Probability of move, from [8]

If $\pi(y) \leq \pi(x)$ (the jump goes 'uphill') $\alpha(x, y) = 1$ and the move is always made. Otherwise, if it goes 'downhill', it is accepted with a non-zero probability.

In our case, if the musician plays slightly after or before the tempo with an offset of x, and if y, the offset at the next time, is close to 0, $\pi(y) > \pi(x)$ (as closer to 0), the move is always made, and the musician catches the tempo, and vice-versa.
Thus, we generate a different Metropolis-Hasting chain for every musician $MH^{(m)}(u) = \delta^{(m)}(u)$ of length N_a, the number of analysis frames, and we low-pass filter it with a Hanning window.
The algorithm implemented in *MetropolisHastings.m* shows the time differences generated:

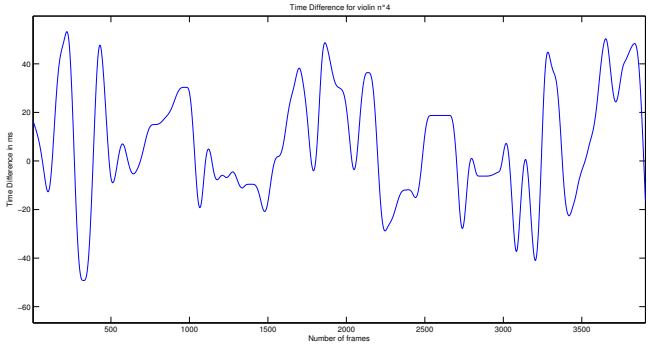


Figure 8: Time Differences for a musician m, in ms

As we can see, filtering this random chain with a low-pass filter simulates a musician 'moving' around the tempo, consecutively catching it and playing slightly behind or before the tempo at the next moment.
The Metropolis-Hastings sampling is perfect for a phase vocoder approach. As it can remain in the same state during several pickings, it guarantees at a low-frequency coherence between overlapping and consecutive synthesis frames.
As we're placing the synthesised STFT for each musician m at time-instants $t_s^u = u.R_s + \delta^{(m)}(u)$, we compute them accordingly, for each musician m:

$$\begin{cases} |Y(t_s^u, \Omega_k)| = |X(t_a^u, \Omega_k)| \\ \angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + \hat{\omega_k}(t_a^u)\Delta t_s(u) \end{cases} \tag{16}$$

where $\Delta t_s(u) = t_s^u - t_s^{u-1} = R_s + \delta^{(m)}(u) - \delta^{(m)}(u-1)$, $R_s = d^{(m)}.R_a$. In practice, $\delta^{(m)}(u) - \delta^{(m)}(u-1) = d^{(m)}(\delta^{(m)}(u) - \delta^{(m)}(u-1))$, so that the time fluctuations are following $\mathcal{N}(0, \sigma^2)$, $\sigma = 40ms$, when resampled by a factor $\frac{1}{d^{(m)}}$.
Admitting we're in a zone where the average time difference is $\delta^{(m)} \simeq 40ms$, the synthesis frames will be set locally a bit after where they should have been if they were set at the normal time-instants $t_s^u = uR_s$. The instrument is then heard as playing right after the tempo.
The spacing between two consecutive synthesis frames is not constant anymore. We have to be careful though that $\Delta^{(m)}(u) = \delta^{(m)}(u) - \delta^{(m)}(u-1)$ remains small enough, that means small variations between two consecutive values of $\delta^{(m)}$ so that the synthesised frames keeps overlapping correctly. The hypothesis of a constant instantaneous frequency $\hat{\omega_k}(t_a^u)$ that remains constant between two consecutive frames, which we use to compute the synthesised phase, also lies in the small gap between two frames.
That's why we low-pass filtered the Metropolis-Hastings sampling, to keep coherence between overlapping consecutive synthesis frames.
Moreover, E. Moulines and J. Laroche in [4] recalls that theoretically, if $f : t_a^u \mapsto t_s^u$ is not linear (in our case, f no longer equals $f(t) = d^{(m)}t$), then the synthesis window should be normalized at each synthesis frames.
Indeed, in figure 3, when we did the perfect reconstruction hypothesis and we normalized the synthesis window w_s so that $\chi = 1$ on the synthesis frames, we did it for all synthesis frames in one time because the time-stretching factor was constant. Here, as the gap between synthesis frames is varying, we should normalize the synthesis window at each frame to make them overlap

correctly.
However, in practice, this normalisation can be skipped if a sufficient overlap between consecutive synthesis frames is provided, which was fufilled with $\delta^{(m)}(u)-\delta^{(m)}(u-1)$ when we low-pass filtered the time fluctuations we generated (thee maximum of $\frac{d(\delta(u))}{du}$ was lower than 50 samples, which represents approximately 10% of the hop factor R_s.

3.3 Amplitude modulation

The last part of the PTA algorithm consists in an amplitude modulation, which simulates varying playing dynamics between musicians and between consecutive notes.
This will be done by adding a factor $\gamma(u)$ for every frame in (4) before the synthesised signals $y_u(n-t_s^u)$:

$$y(n) = \sum_{u\in\mathbb{Z}} \gamma(u) y_u(n-t_s^u) w_s(n-t_s^u) \tag{17}$$

Of course, $\gamma(u) = \gamma^{(m)}(u)$ and $t_s^u = t_s^{(m),u} = u.R_s^{(m)} + \delta^{(m)}(u)$, where $R_s^{(m)} = d^{(m)}R_a$ depends on the musician m, but the expression will be written as above to be clearer.
As for the time difference sampling, the chain γ of length N_a, the number of frames will be set as a low frequency random Metropolis-Hastings sampling. As for the tempo, it simulates a musician playing louder, then catching the average volume and below, and the low-pass filtering stands for coherence between overlapping synthesis frames.
J. Pätynen and al. [2] chose a normal distribution $\mathcal{N}(\mu,\sigma^2)$ centered in $\mu = 1$, and with a 1dB standard deviation σ, which corresponds to a standard deviation of $10^{\frac{1}{20}} \simeq 12\%$. The modulation frequency is set to 5 Hz which corresponds approximately to eight notes in moderate tempo.
As the metropolis-Hastings sampling is operating every frame, we will look for a Hanning window operating on frames whose length N_f is such as:

$$w_h = \frac{2F_e}{N_f R_a} \leq 5Hz$$

as frames are set every R_a samples, so $N_f \geq \frac{2F_e}{5R_a}$.
After obtaining the low frequency amplitude chain for each musician, we scale the sum of parallel random values to unity to change balance between different musicians without affecting the general sound level.

This technique is coded in the function *PTA.m*, and uses the functions *ola.m*, and *MetropolisHastings.m*.
It shows good audio results that are available in the repository *Violin*, and are named 'PTA_12.wav' and 'PTA_1.wav', which are respectively the results for a full section of 12 violins, and a single violin source.

4 Improvement of the phase vocoder

The method described above, standing on a classical phase vocoder approach, shows high-quality audio results as compared to the traditional chorus effect. However, even if the audio results are good, we can hear at some point a "loss of presence" in the synthesised signals. This artifact, called *phasiness*, is with *transient smearing* characteristic artifacts of the phase vocoder applications.
Phasiness (or reverberation or "loss of presence") is heard as a characteristic coloration of the signal; in particular, time-expanded audio signals, even for modification that are close to one (that is our case), can sound as if the musician is much further from the microphone than in the original recording. One has to note that this effect is independent from amplitude modulation and occurs even when there is none.
Transient smearing occurs also even with modification factors that are close to one, and is heard

as a slight loss of percussiveness in the signal. Percussive intruments (as the piano, or drums) are perceived as having less "bite."
These two artifacts will be dealt with and explained in the two following sections.

4.1 Reducing *phasiness* with *phase locking techniques*

This section is inspired by the work of J. Laroche and M. Dolson in [3] who developped two techniques to reduce considerably the effect of *phasiness*.

4.1.1 Limitations of the horizontal phase propagation

The problem of *phasiness* is known to lie in the modifications of phases in the STFT. Indeed, when applying the formula:

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + \hat{\omega_k}(t_a^u)\Delta t_s(u)$$

This formula guarantees a *horizontal phase coherence*, in other words, coherence is maintained only *within* each frequency channel over time. In the following section, we see why it is also important to have coherence *across* frequency channels for a given synthesis frame, to achieve what J. Laroche and M. Dolson name *vertical phase coherence*, and how incoherent synthesis phase computation can easily lead to the phasiness artifact.
If horizontal and vertical phase coherence are not preserved, the modified STFT won't be a valid one for the synthesis signal y, so the STFT consistency will be damaged and phasiness will be heard. Thus, the results of the following algorithms will be analysed through the consistency measure we introduced above.
Because phase propagation errors are in the center of many of the sound quality issues in the phase vocoder, it is important to understand how sinusoidal phases are altered by vocoder-based time-scale modifications.
Let's unroll the phase-propagation equation (10), assuming that $t_s^u = uR_s$, and $t_a^u = uR_a$:

$$\begin{aligned}\angle Y(t_s^u, \Omega_k) &= \angle Y(t_s^0, \Omega_k) + \sum_{i=1}^{u} R_s \hat{\omega_k}(t_a^i) \\ &= \angle Y(t_s^0, \Omega_k) + \sum_{i=1}^{u} [R_s\Omega_k + \frac{R_s}{R_a}\Delta_p\Phi_k^u]\end{aligned}$$

Replacing $\Delta_p\Phi_k^u$ by its definition, and noting $\alpha = \frac{R_s}{R_a}$, we obtain:

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^0, \Omega_k) + \alpha \sum_{i=1}^{u} [\angle X(t_a^i, \Omega_k) - \angle X(t_a^{i-1}, \Omega_k) + 2m_k^i\pi]$$

where $2m_k^i\pi = \Delta_p\Phi_k^i - \Delta\Phi_k^i$ (m_k^i is the unique integer (the *unwrapping factor*) found thanks to the *unwrapping* hypothesis $R_a\omega_h < \pi$, with ω_h cutoff frequency of the analysis window w_a, that links the STFT analysis phase with the instantaneous phase in the channel). In the previous equation, we also recover the wrapped phase determination up to an integer multiple of 2π. Finally, we get:

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^0, \Omega_k) + \alpha[\angle X(t_a^u, \Omega_k) - \angle X(t_a^0, \Omega_k)] + \alpha \sum_{i=1}^{u} 2m_k^i\pi] \tag{18}$$

This equation shows that at a given time-instant t_s^u, the synthesis phase depends only on the phase of the current analysis instant, on the initial synthesis frame, on the initial analysis frame and most importantly on the series of *phase-unwrapping factors* m_k^i.
Let's take sinusoid I, part of the original signal x following the M^{c}Aulay and Quatieri model. The

channels k influenced by this sinusoid are the ones such as $|\omega_k - \omega_I| < \omega_h$. Thus, for these channels k, we'll have, according to (35) in the annex:

$$X(t_a^u, \Omega_k) = A_I(t_a^u) e^{j\psi_I(t_a^u)} W_a(e^{j(\Omega_k - \omega_I(t_a^u))})$$

where W_a is the FFT of the analysis window w_a. If we have a symmetric window, W_a is real, thus its phase is null, and all the adjacent channels influenced by the nearby sinusoid I will have the same phase $\angle X(t_a^u, \Omega_k) = \psi_I(t_a^u) + 2n_k^u\pi$ that equals the instantaneous phase up to a multiple of 2π. Only the amplitude $A_I(t_a^u)W_a(e^{j(\Omega_k - \omega_I(t_a^u))})$ will change depending on the proximity with the nearby sinusoid.
In practice, we look for the phase difference between two consecutive analysis instants, and relate it to the instantaneous phase difference up to the unwrapping factor m_k^u:

$$\begin{aligned}\angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) &= \psi_I(t_a^u) - \psi_I(t_a^{u-1}) + 2m_k^u\pi \\ &= \omega_I(t_a^u)\Delta t_a(u) + 2m_k^u\pi\end{aligned}$$

by deriving the phase. Then we have to find the unique m_k^u so the instantaneous frequency of the sinusoid I $\omega_I(t_a^u) = \hat{\omega_k}(t_a^u)$ for the k*th* channels influenced by the sinusoid is known, and that's what the phase unwrapping method tends to do.
NB: in practice however, the analysis window is not centered and is nonzero for $0 \leq n < L$. This offset creates, a variation of π in the adjacent channels.
Thus, it seems essential that modified STFT $Y(t_s^u, \Omega_k)$ has to keep the same synthesis phases in channels around the sinusoids composing the signal. This is precisely what we called the *vertical phase coherence*.
In (18), as $\angle X(t_a^u, \Omega_k) = \psi_I(t_a^u) + 2m_k^u\pi$ for all channels k around the sinusoid, we want the sum of the unwrapping factors $\alpha \sum_{i=1}^{u} 2m_k^i\pi$ equal (modulo 2π) in nearby channels.
Let's place ourselves in a synthesis frame u, on channels k that are nearby a sinusoid I. The computation of the *unwrapping factors* m_k^u depends directly on the *unwrapping* condition:

$$|\angle X(t_a^u, \Omega_k) - \angle X(t_u^{i-1}, \Omega_k) - \Omega_k R_a - 2m_k^i\pi| = |\Delta\Phi_k^u - 2m_k^u\pi| = |(\omega_I(t_a^u) - \Omega_k)R_a| < w_h R_a < \pi$$

So that we can find a unique $2m_k^i$ that satisfies the equation and gives us the instantaneous phase and frequency. For each channel k, $\Delta\Phi_k^u$ is the *heterodyned phase* which is actually the small phase shift resulting in the sinusoid I being close, but not necessarily equal to Ω_k.
If R_a is small enough, then the unwrapping condition is fulfilled and there is no danger of phase unwrapping errors in the adjacent channels. However, we're working here with $R_a = \frac{N}{4}$ such as $w_h R_a = \pi$. As before, there won't be any problem for the closest channels of the sinusoid I.
However, the furthest channels can be influenced by noise or unrelated sinusoid, modifying slightly $\angle X(t_a^u, \Omega_k) - \angle X(t_u^{i-1}, \Omega_k)$ for these channels. This will lead to a wrong unwrapping factor m_k^u, and result in breaking the equality between the phases of the related channels.
More, we can see in (18) that phase unwrapping errors accumulate in the sum, losing the vertical coherence for ever.

In (18), we've seen that the phase at time-instant t_s^u, $\angle Y(t_s^u, \Omega_k)$ depends on the initial synthesis phase $\angle Y(t_s^0, \Omega_k)$. J.laroche and M. Dolson have shown that this initialization is important, especially when the time-expansion factor α is an integer. For this case, an initialization $\angle Y(t_s^0, \Omega_k) = \alpha\angle X(t_s^0, \Omega_k)$ is advised to reduce consistently the phasiness artifact.
We don't apply this initialisation as we're working with non-integer factors close to one, and will set $\angle Y(t_s^0, \Omega_k) = \angle X(t_a^0, \Omega_k)$ for the rest of the implementation.

4.1.2 Identity phase locking

Departing from the observations noted above, J. Laroche and M. Dolson developped a technique, called *rigid phase locking* that locks vertically the synthesis phases around the peaks in the STFT. The first step consists in locating energy peaks at time t_a^u in the analysis STFT frame. These peaks corresponds to the sinusoids i present in the original signal, and compute for the peak channels the new synthesis phase according to the classical phase-propagation formula (10). These peak channels are noted Ω_{k_i}, and are considered in our implementation as channels whose amplitude is larger than its four nearest neighbours, and larger than $\frac{max(|X(t_a^u,\Omega_k)|)}{5}$. This threshold prevents us from taking unrelated peaks only caused by noise, and decreases the computationnal cost of our implementation.
Then the phases of the nearby channels are locked to the new synthesis phases of these peaks. In our algorithm, we choose to set the upper limit of the region surrounding peak Ω_{k_i}, as the channel of lowest amplitude between the two peaks Ω_{k_i} and $\Omega_{k_{i+1}}$.
From what we explained in the section, the synthesis phases around the peak are then chosen to be related in the same way as the analysis phases: the synthesis phase difference between adjacent channels around a peak are set identical to the analysis phase difference in the analysis STFT.
Thus, once we've computed $\angle Y(t_s^u, \Omega_{k_i})$ for all peaks, for channels k in the zone of influence, we set:

$$\angle Y(t_s^u, \Omega_k) - \angle Y(t_s^u, \Omega_{k_i}) = \angle X(t_a^u, \Omega_k) - \angle X(t_a^u, \Omega_{k_i})$$

Or, in other terms:

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^u, \Omega_{k_i}) + \angle X(t_a^u, \Omega_k) - \angle X(t_a^u, \Omega_{k_i}) \tag{19}$$

There's no danger in unwrapping factors errors anymore for further channels from the sinusoid channel (peak). By relating the synthesis phases in the way as were the analysis ones, we're sure to keep the same coherence across channels from the analysis STFT to the synthesised STFT. In particular, as explained above, for nearby channels around a sinusoid I, if we have a perfect signal (no noise or unrelated sinusoid), then $\angle X(t_a^u, \Omega_k) = \angle X(t_a^u, \Omega_{k_I})$ with Ω_{k_I} the channel where the sinusoid I is. This provides $\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^u, \Omega_{k_I})$. The coherence is preserved for channels surrounding a sinusoid, independently from the unwrapping factors.
More, since unwrapping is performed only on peak channels, we're sure that the instantaneous frequency of the sinusoid I is close to the center frequency of this channel. As a result, the heterodyned phase $\Delta\Phi_{k_i}^u$ is minimum and the unwrapping condition $|\angle X(t_a^u, \Omega_k) - \angle X(t_u^{i-1}, \Omega_k) - \Omega_k R_a - 2m_k^i\pi| = |(\omega_I^u - \Omega_k)R_a| < \pi$ to find the unique unwrapping factor m_k^i is clearly fulfilled. ω_I^u is here the instantaneous frequency of the sinusoid I in the channel. The unwrapping condition $\omega_h R_a < \pi$ can then be relaxed, allowing even greater values of R_a and overlap down to 50% can be chosen without generating unwrapping errors. For our case, whose analysis Hanning window requires a normal overlap of 75%, it reduces the computational cost by a factor of two!

Finally, this technique is really cheap as concerned its computational cost. Phase unwrapping technique is only applied to peak channels. Then, phases are locked according to (19) by defining:

$$\theta = \angle Y(t_s^u, \Omega_{k_i}) - \angle X(t_a^u, \Omega_{k_i}) \tag{20}$$

and simply rotate the phase for the adjacent channels with the phasor $Z = e^{j\theta}$ (the amplitude is the same, only a phase shift is operated) as follows:

$$Y(t_s^u, \Omega_k) = ZX(t_a^u, \Omega_k) \tag{21}$$

Thus, the phase lock computation for the neighboring channels only require one complex multiply.
The algorithm is summed below:

Algorithm 2 Identity phase locking

Steps

1: **for** STFT frames **do**
2: Locate prominent peaks
3: **for** each peak **do**
4: Compute the synthesised STFT according to the phase unwrapping method:
5: $|Y(t_s^u, \Omega_{k_l})| = |X(t_a^u, \Omega_{k_l})|$
6: $\angle Y(t_s^u, \Omega_{k_l}) = \angle Y(t_s^{u-1}, \Omega_{k_l}) + \hat{\omega_{k_l}}(t_a^u)\Delta t_s(u)$
7: Compute $Z = e^{j\theta}$
8: Compute $Y(t_s^u, \Omega_k) = ZX(t_a^u, \Omega_k)$ for all channels in the zone of influence around the peak, including the peak
9: **end for**
10: **end for**

4.1.3 Computation inter-peaks and scaled phase locking

In the previous algorithm, and since the beginning, the phase propagation is computed between two consecutive synthesis frames within the same channel. However, signals and their sinusoids can slowly move across channels between consecutive frames, even when analysing a single note. A chirp signal is the perfect example to illustrate this case. Especially now that we're concentrating on peaks, a sinusoid can move slightly from channel k_0 at time t_a^{u-1} to channel k_1 at time t_a^u.
In that case, the unwrapping equation should be based for two peaks on the phase difference $\angle X(t_a^u, \Omega_{k_1}) - \angle X(t_s^{u-1}, \Omega_{k_0})$, and the phase propagation equation changed to:

$$\angle Y(t_s^u, \Omega_{k_1}) = \angle Y(t_s^{u-1}, \Omega_{k_0}) + \hat{\omega_{k_1}}(t_a^u)\Delta t_s(u) \tag{22}$$

To determine which peak in time-instant t_a^{u-1} is related to the new one in t_a^u, we'll just take the dominant peak Ω_{k_0} of the region where the peak channel Ω_{k_1} at the current time belonged at the former time-instant t_a^{u-1}.
J. Laroche and M. Dolson [3] propose also a new phase locking technique called *scaled phase locking* that sets the synthesis phases of the channels in the region of peaks as follows:

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^u, \Omega_{k_i}) + \beta(\angle X(t_a^u, \Omega_k) - \angle X(t_a^u, \Omega_{k_i})) \tag{23}$$

where β is a phase-scaling factor. Identity phase locking is simply achieved with $\beta = 1$. J. Laroche and M. Dolson found experimentaly that setting $\beta = \frac{2}{3} + \frac{\alpha}{3}$ with α as the time-stretching factor helps reducing phasiness.
In our cases, as $\alpha \simeq 1$, this case doesn't change much from the identity phase locking in theory. However, subjective listening tests with the two phase locking techniques proved that the audio results have a better quality using the scaled phase locking technique.
This technique has the disadvantage that we can not use the phasor $Z = e^{j\theta}$ to compute the synthesis phases anymore. More, we have to unwrap the phases $\angle X(t_a^u, \Omega_k)$ across channels k around the peak k_i to avoid jumps of $2\beta\pi$ in the synthesis phases. Even in our case where $\beta \simeq 1$, we want to avoid small phase shifts resulting from $2\beta\pi \simeq 2\pi$ but not necessarily equal.
This unwrapping operation and the calculations of the analysis phases inter peaks will unfortunately lead to a greater computational cost than with the identity phase locking.
The scaled phase locking algorithm is then set as defined below.

Both identity and scaled phase locking techniques were implemented in the functions *PTA_PhaseLock.m* and *PTA_ScaledPhaseLock.m* respectively. Each one can use inter or intra peaks phase unwrapping computation during the analysis stage by setting the parameter *phaselock* to *'inter'* or *'intra'*.

Algorithm 3 Scaled phase locking with computation Inter Peaks

Steps

for STFT frames **do**
  Locate prominent peaks
  for each peak **do**
    Find the associated previous peak and compute the synthesised STFT according to the phase unwrapping method:
      $|Y(t_s^u, \Omega_{k_l})| = |X(t_a^u, \Omega_{k_l})|$
      Compute $\omega_{k_1}\hat{}(t_a^u)$ with $\angle X(t_a^u, \Omega_{k_1}) - \angle X(t_a^{u-1}, \Omega_{k_0})$
      $\angle Y(t_s^u, \Omega_{k_1}) = \angle Y(t_s^{u-1}, \Omega_{k_0}) + \omega_{k_1}\hat{}(t_a^u)\Delta t_s(u)$
    Unwrap analysis phases across all channels in the region around the peak to avoid $2\beta\pi$ jumps
    Compute $Y(t_s^u, \Omega_k) = \angle Y(t_s^u, \Omega_{k_i}) + \beta(\angle X(t_a^u, \Omega_k) - \angle X(t_a^u, \Omega_{k_i}))$ for all channels in the region around the peak
  end for
end for

4.1.4 Results

The above phase locking techniques were added to the former PTA algorithm and the consistency measure for the synthesised STFT was computed, using the formula (6) by taking the time-instants with time fluctuations $t_s^u = uR_s + \delta(u)$, with $R_s = d^{(m)}R_a$. P is taken when χ reaches 1, according to the perfect reconstruction graph 3. We can see that $\chi(N_{frames} - P)$ also equals 1 on the other side.
The algorithms were compared through STFT consistency measures and computational cost evaluations. The STFT consistency measures presented below was taken each time as the mean of the STFT consistencies obtained for the M instruments generated.
The results are slightly improved, and the computational costs are good as well:

Phase Locking technique	D_m	Time
Classic PTA	-38dB	49.55s
PTA Identity PhaseLock - Intra Peaks	-39dB	50s
PTA Identity PhaseLock - Inter Peaks	-40dB	54.57s
PTA Scaled PhaseLock - Intra Peaks	-39dB	70.12s
PTA Scaled PhaseLock - Inter Peaks	-38.5dB	70.86s
PTA Identity PhaseLock - Intra Peaks - 50%	-37dB	26.77s
PTA Scaled PhaseLock - Inter Peaks - 50%	-35dB	37.64s

Table 1: STFT consistency and execution time for different phase locking techniques

- 50% defines the same algorithms run with an overlap of 50%. The STFT consistency was slightly improved with the phase-locking techniques, but remain very close, as we're working with modification factors close to one.
However, the audio results are heard to be better, and also appear to be more 'punchy': the loss of presence created by the *phasiness* artifact, as if the musician was further from the microphone, is no longer heard.
To check that the consistency in the generated STFT is indeed improved with the phase locking techniques, tests with a time-stretching factor of 1.5 were computed and the results are exposed below:

Phase Locking technique	D_m
Classic PTA	-34dB
PTA Identity PhaseLock - Intra Peaks	-46dB
PTA Identity PhaseLock - Inter Peaks	-43dB
PTA Scaled PhaseLock - Intra Peaks	-39dB
PTA Scaled PhaseLock - Inter Peaks	-38.5dB
PTA Identity PhaseLock - Intra Peaks - 50%	-32dB

Table 2: STFT consistency for a time-stretch factor of 1.5

The STFT consistency are shown to be merely improved with the new algorithms, up to a decrease of -12dB as compared with the classical PTA!
The scaled phase locking algorithms don't present as good results as the identity phase locking in general. Though, the audio listening suggested that the results were better, and less 'phasy' with the scaled phase locking, and although the STFT consistency can show strong consistency and coherence in the synthesis signals, it is not a clear indicator of phasiness.
As concerned the computational cost of the algorithms, the identity phase locking ones are as low computational as in theory and have the same cost than the classic PTA (even with peaks computation), whether with intra or inter peaks computation. Indeed, as the inter/intra peaks is done only one time in the analysis stage (compared to the M synthesis stages), the overall time execution is not far from one to another. The PTA scaled phase lock is unfortunately longer in execution, as it can't use the phasor operator to change phase in peaks's regions of influence, but is still fair and its results have better quality than the PTA and the identity phase locking.
Finally, the Scaled and Identity PTA run with an overlap of 50% (hop factor $R_a = \frac{N_w}{2}$), which reduces as supposed the number of computations, and so the time of execution by a factor of two. The STFT consistency measure is really fair with -35dB and -37dB for the identity phase locking as compared with the classical PTA (-37.5dB). This would have been a perfect solution if we wanted to have fair results in a short time. However, the synthesised sound rough and rubbling, probably because of an insufficient overlap, and even if the time fluctuations were set to zero. J. Laroche and M. Dolson [3] also noticed this phenomenon and advised to avoid the artifact to choose β close to one, that was already our case.
Thus, the phase locking techniques are solutions that can be easily added to the classical PTA, and are shown to reduce phasiness artifact as compared with the results obtained with the PTA algorithm, while keeping a fair computational cost. Every phase locking technique was heard and the scaled phase locking technique, with inter peaks computation was the one who yielded the best audio results. This is the one we will be using from now on until the end of the study. Only phase artifacts were heard with the phase locking algorithms when the musician was breathing, creating a slight 'fuzzing/laser' effect.
All the results are available in the 'Violin' folder, with *'technique_ nbViolins.wav'*.

4.2 Reducing *transient smearing* with transient detection

The study above works well for constant-frequency signals, or signals whose frequencies vary slowly (as the *chirp* signal used in [3]).
However, if the signal changes brutally (what we call a *transient attack*), the technique above won't work as the phase of the peaks between consecutive frames are not related anymore. Plus, the *stationary hypothesis* is no longer fulfilled, and time stretching attack transients (so updating the phase using an unrelated previous phase with the unwrapping method) will result in a *transient smearing* problem, or loss of percussiveness in the signal, as introduced above. In more severe cases, a complete change of the sound characteristics may take place.
Though it may not be heard clearly with a section of violins (as they are non-percussive instruments), this transient smearing artifact can create audio problems for some percussive instrument sections composing a symphonic orchestra, such as drums or horns.
Below is the recording of a snare we did in the second part:

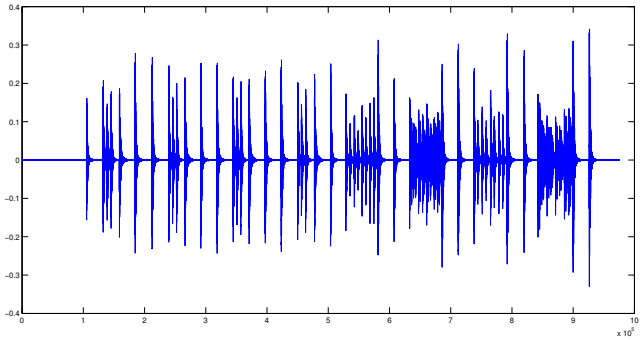


Figure 9: Snare signal

We stretched the signal by a factor of two to see the artifact. Below on the left is the zoom on the first onset for the original recording, and on the right on the same onset for the time-stretched recording.

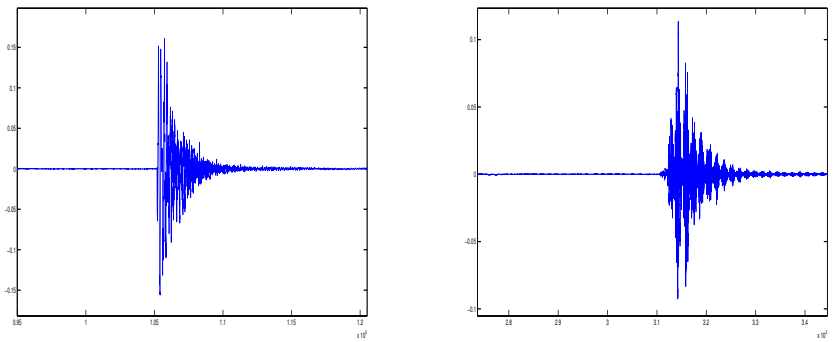


Figure 10: Transient smearing artifact

As we can see, the transient attack has disappeared, resulting in a loss of percussiveness in the sound. This artifact can be heard with the audio file *transientSmearing.wav* in the folder 'Snare'.

The study proposed by Axel Röbel in [5] consists in detecting frames where transient attacks occurs and re-initializing phases of the synthesis STFT for these frames, as the phase propagation formula in (10) is no longer valid.
To detect transients, A. Röbel computes the *Center Of Gravity* (COG) $\hat{t}$ defined in [5] as:

$$\hat{t} = \frac{\int ts(t)^2 dt}{\int s(t)^2 dt}$$

which can also be computed, at time instant t_a^u with:

$$\hat{t_a^u} = \frac{\int -\frac{\partial \phi(t_a^u,\omega)}{\partial \omega)} A(t_a^u,\omega)^2 d\omega}{\int A(t_a^u,\omega)^2 d\omega} \tag{24}$$

where $A(t_a^u,\omega)$ and $\phi(t_a^u,\omega)$ are respectively the amplitude and phase of the analysis STFT at time t_a^u: $X(t_a^u,\omega) = A(t_a^u,\omega)e^{j\phi(t_a^u,\omega)}$.
As we can see, the computation of the COG lies on a phase criteria which is particularly relevant for a phase vocoder application, as compared to energy based criteria used in former methods. Moreover, we'll see after that these two criteria, thanks to formula developed by Auder and al. [9] are immediately related.

The COG developed by A. Röbel lies on the *time-reassignment operator* $\gamma : (t, \omega) \mapsto (t', \omega')$ developped by Auder and al. [9] and shown in the figure below:

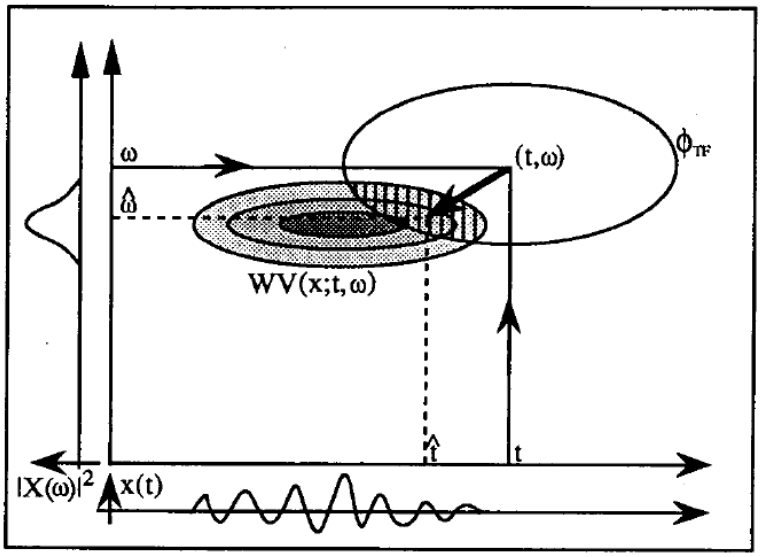


Figure 11: Time reassignment method, from [9]

The time-reassignment operator applied to time-frequency representations consists in computing (t', ω') that will "find" the energy around (t, ω) in the representation. In the example above, we can see that the energy zone is 'before' and 'below' (t, ω). As a result, we'll have $t' < t$ and $\omega' < \omega$.
The Center Of Gravity stands on a time-reassignment method, and basically indicates where the energy is in an analysis frame. Further explanation on how we find this COG are excerpted from [9] and given in the annex.
As A. Röbel is focusing only on time issues, the time will be fixed at t_a^u, and the COG obtained in (24) by integrating the time computed by the time reassignment operator on the frequency axis, and weighting it with the amplitudes $A(t_a^u, \omega)$.
The time-instant t_a^u is also assumed to be in the center of the window, without loss of generality, to get the COG formula.
Admitting we're encountering an attack transient. For stationary partials, the phase is constant, so the COG is equal to 0 and is in the center of the analysis window. If we encounter a transient, then the average phase derivative is negative, and $\hat{t_a^u} > t_a^u = 0$ is in the right of the analysis window, looking for the energy as shown in fig 11.
Then, as we get closer from the transient, the phase derivative and $\hat{t_a^u}$ will decrease slowly back to zero, as we're reaching a stationary signal.
As the COG can't go further than the window right size, when detecting a transient, it will be placed at +50% of the window size to its maximum value, and will then move back to the center of the frame at 0%.
In the article, A. Röbel [5] has tested the values of the COG for different ramp inclination (different type of transients) and found that a threshold of $C_e = 4.4\%$ of the analysis window's length ensures a minimum of 60% of transient covered by the analysis window, regardless of its type.
When we're on a transient, the synthesis phases are not related anymore with the ones in the previous frame. The phases are reinitialized and the phase propagation using the unwrapping method is going on in the next frame.

As we said, this phase vocoder improvement is particularly useful to prevent percussiveness in attack transients from being lost during the phase propagation computation. Though it may not

be heard clearly with a section of violins (as they are non-percussive instruments), this transient detection is useful for some percussive instrument sections composing a symphonic orchestra, such as drums or horns.
The COG is computed in each analysis frames of the signal and is display below for a violin and for a snare that we personally recorded anechoically. A. Röbel [5] worked on polyphonic signals and had to integrate the phase derivative on different sub-bands in (24) and then sum it all to distinguish energy brought by real transients are by partials of notes already existing. In our case, working with a solo recording removes all problems concerning polyphonic treatments, possible overlapping of partials from different notes, and we can directly integrate on the whole frequency range.
Morevover, Auder and al. [9] found a relation relating the phase derivative with the STFT of the signal windowed by w_a and $\mathcal{T}w_a(t) = t.w_a(t)$, the analysis window multiplied by a time ramp, and is more adequate for computational purposes when working with the phase vocoder and STFT:

$$-\frac{\partial\phi(t_a^u,\omega)}{\partial\omega} = -\Re\left(\frac{STFT_{\mathcal{T}w_a}(t_a^u,\omega).\overline{STFT_{w_a}(t_a^u,\omega)}}{|STFT_{w_a}(t_a^u,\omega)|^2}\right) \tag{25}$$

That's the implementation we used to compute the COG in every frame with the function *computeCOG.m*. The evolution of the COG, so its computation in every frame is found in function *COG_g.m* and is displayed below respectively for the violin we studied before and for a snare:

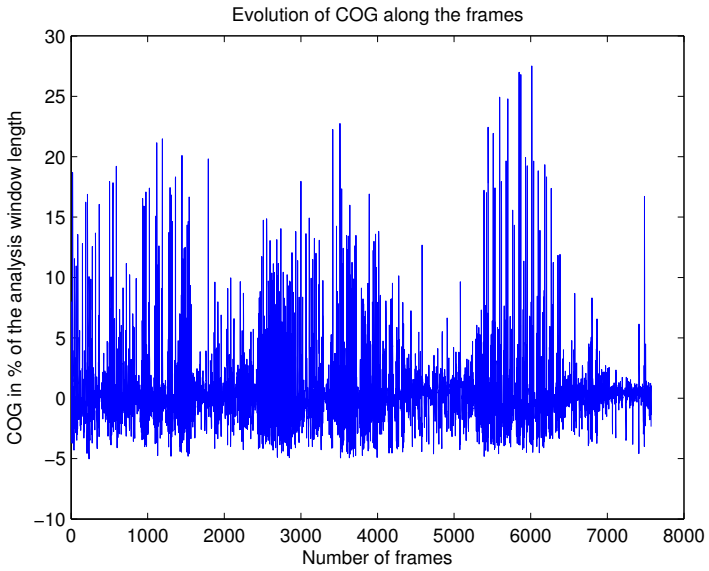


Figure 12: COG for a violin

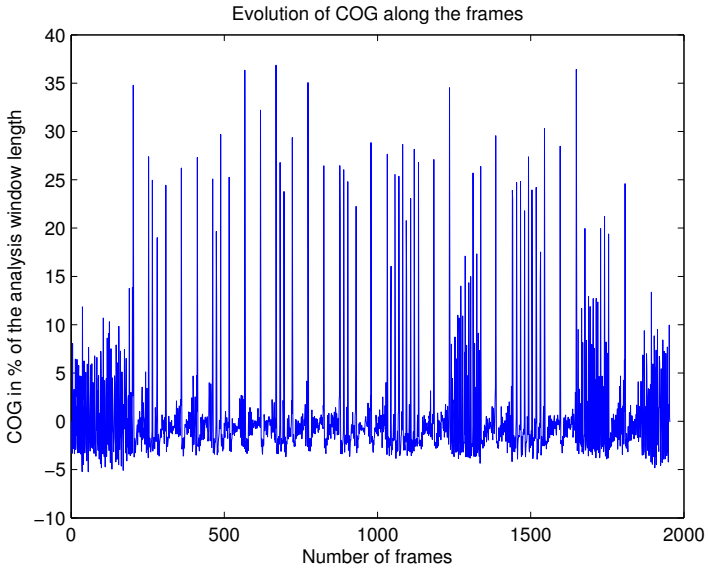


Figure 13: COG for a snare

In these two graphs, the peaks corresponds clearly to the onsets in the original recordings. As we can see, they are much more clear for the percussive snare signal, reaching until 40% of the analysis window length on the right (the max is 50% as the time origin is in the center of the window). For violin sounds, the COG reaches only 25%.
The threshold $C_s > C_e$ to detect transients will be set to $COG > C_s = max(COG)/2$. After verifying this condition, we assume that we are on the transient when $COG < C_e = 4.4\%$ in a given frame as stated by A. Röbel. In this frame, the synthesis phase is re-initialized.
The following graph shows the first onset, after time-stretching using the transient detection:

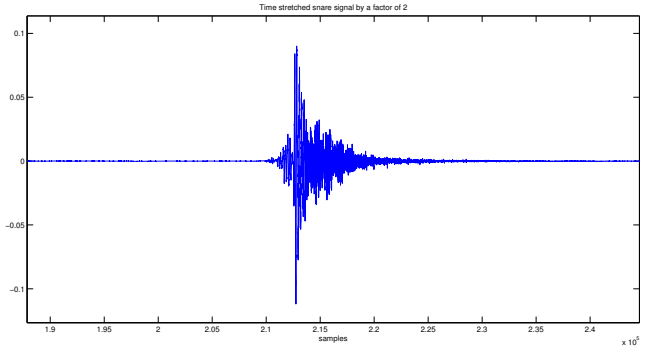


Figure 14: Restored transient attack

The transient attack has clearly been restored, and the synthesised sound has regained in percussiveness.
Thus, we add this transient detection to the *scaled phase lock PTA* algorithm with inter-peaks computation to get the *improved PTA* algorithm described below:

For the violin, the results appear to be less phasy than with the scaled inter-peaks algorithm, especially during the musician breaths where phasy artifacts were heard the most. As the phase locking algorithms already corrected the phasiness effect, making the sound more punchy, and as the violin are non-percussive signals, the regain of percussiveness was hard to detect on the onsets between the two algorithms. However, we will see in the second part that transient smearing artifact was indeed corrected when percussive snare signals are processed.

Algorithm 4 Improved PTA

Steps

for STFT frames **do**
  if transient detected **then**
    Re-initialize the synthesis phases:
      $Y(t_s^u, \Omega_k) = X(t_a^u, \Omega_k)$, so:
        $|Y(t_s^u, \Omega_k)| = |X(t_a^u, \Omega_k)|$ and
        $\angle Y(t_s^u, \Omega_k) = \angle X(t_a^u, \Omega_k)$
  else
    Scaled phase lock PTA with inter-peaks computation algorithm
  end if
end for

The audio results can be found in the same folder 'Violin', in 'improvedPTA_1.wav' and 'improvedPTA_12.wav', and the performances of the algorithm are given below:

Phase Locking technique	D_m	Time
Classic PTA	-34dB	49.55s
PTA Scaled PhaseLock - Inter Peaks	-36dB	70,86s
Inproved PTA	-39.5dB	75.62s

Table 3: Improved PTA performances

The time computation is approximately the same as the scaled phase locking algorithm, where the COG detection was added. As for the STFT consistency, the result shows improvement compared to the former algorithms.
Another advantage of the improved PTA algorithm for all instruments is that as the phase propagation formula is no longer used when detecting attack transients, we are sure that the instantaneous frequency will remain the same and constant between two consecutive synthesis frames.
As a result, a greater interval between consecutive synthesis frames is allowed (providing of course that the frames overlap in a sufficient way), and greater variations of time fluctuations are permitted without hurting the sound.
In practice, the time fluctuations can be low-pass filtered with a cutoff frequency of 5Hz (eight notes in the medium tempo), as for the amplitude modulation! This provides a fluctuation on each onset, and not a general and progressive fluctuation as in the PTA algorithm.
With a low-pass filter of 5Hz cutoff frequency, the time fluctuations given in the graph below:

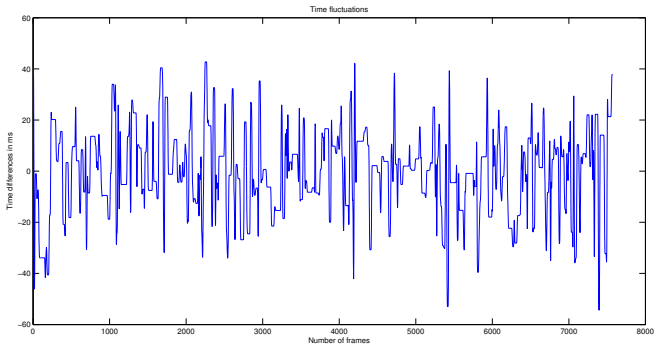


Figure 15: Time fluctuations - 5Hz cutoff frequency

were such as, on average $max(\frac{d\delta(u)}{du}) = 500 < R_a$ samples for an overlap of 75%. This can create an overlap varying between 100% and 50% at max. Thus, we can still consider that a

general normalisation during the synthesis stage can still be operated (as revealed by E. Moulines [4]), and that coherence still exists between consecutive synthesis frames. The only issue would be to have a rough sound as when we ran the algorithms with an overlap of 50%, that can't happen as this overlap is only for the extrema, and the average overlap is still 75%.

5 Phase vocoder with onset analysis

In the previous sections, we were adding time fluctuations on the signal frames, thus progressively. The changes operate smoothly, but without taking care of the onsets and the reality of the way musicians are playing, apart from our improvement when low-pass filtering the time fluctuations at 5Hz. In real life, time fluctuations can only occur when the musician is playing notes, and so on onsets.
Thus, the technique we're developing below lies on the same process and same techniques, except that we'll apply the signal modifications to the notes between two consecutive onsets, and not on every frame.
First step was to choose our way to detect onsets. Several, using techniques described by J.P. Bello and al. [10] are explored.

5.1 Onset Detection

5.1.1 Spectral difference

The first method was the *spectral difference (SD)* developed by J.P. Bello and al. in [10], and given at time-instant t_a^u by the following formula:

$$SD(t_a^u) = \sum_k [H(|X(t_a^u, \Omega_k)| - |X(t_a^{u-1}, \Omega_k)|)]^2 \tag{26}$$

where $H(x) = \frac{x+|x|}{2}$. This formula is directly related to the energy change between two consecutive analysis frames, and is particularly adapted for our phase vocoder approaches.
The spectral difference method can be found in *spectralDifference.m*.
Other techniques described by J.P. Bello were also explored. One of them computing the signal phase evolution across frames was also relevant for our approaches. However, tests run on our violin base showed results for these two methods not as good as with the two following detections.

5.1.2 Phase criteria - COG

In his article, Bello talks about phase criteria, and how they can be related with energy. His examples meet computations that resemble to the COG computation, and indeed, we showed in the annex how A. Röbel related it to an energy criteria. Due to its high precision as shown in the previous part (also for non-percussive instruments), it remains a strong choice for onset detection, and will be used for the PSOLA part.

5.1.3 Volume detection

We have the chance to work with monophonic and anechoic sounds. As a result, a simple volume detection can work. The only issue is that some notes can be missed if the musician plays notes legato for example, but this is not a major problem, resulting only in missing some onsets. The graph showing the amplitude detection is shown below:

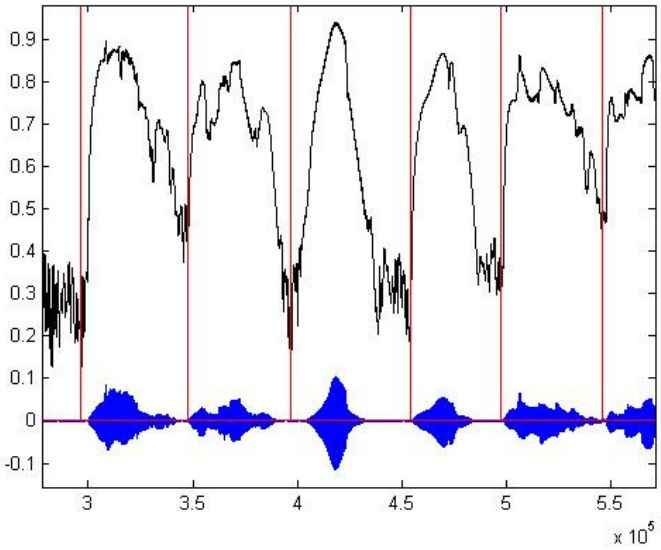


Figure 16: Onset detection using signal's amplitude

With this technique, instead of detecting onsets (where amplitude is at local maximum), we will detect the silences, thus the minimum of amplitude between notes. This choice will be discussed in the following.
Both COG and volume detection showed fair results for our test base of violins. However, the volume detection is the one that fits the best this algorithm as detecting silences instead on onsets and will be the one used for this algorithm, even if its computational cost is much bigger than with the COG detection.

5.2 Time difference model

The advantage with working with onsets is that a low pass-filtering for the delays is not required, as a coherence between successive onsets is no longer necessary, and that time difference can be computed independently.
However, the Metropolis-Hastings sampling can not be used anymore. Indeed, the advantage of the Metropolis-Hastings sampling for the phase vocoder approach was that the delay could remain the same, guaranteeing at a low frequency coherence between successive frames. Here, as we're working with a smaller number of onsets, such a sampling can create important delay between instruments in a short amount of time.
A random delay following a normal distribution $\mathcal{N}(0, \sigma^2)$ would also have the same effect, as the number of onsets is not big enough to ensure a general mean that equals 0.
Thus, we decided to create a wise random picking, such as:

$$\begin{cases} \delta(0) \text{ follows } \mathcal{N}(0, \sigma^2) \\ \delta(l) = -sign(\delta(l-1))\delta(l)| \end{cases} \tag{27}$$

Where δ follows $\mathcal{N}(0, \sigma^2)$ are the delays at each onsets and $sign(\delta(l-1))$ is the sign of the previous delay. We chose as above a standard deviation $\sigma = 40$ms. Thus, by forcing the sign of the delay, we create a musician that goes around the right tempo, playing consecutively above then below the average tempo in the next onset. This guaranteed a minimum delay between the musicians at the end of the play.

5.3 The replica algorithm

The replica algorithm follows the same steps as the PTA algorithm, with the only difference that it is working on notes.
First, an onset detection following the signal volume is done, and the signal is separated in silences (where the amplitude is the lowest). This cuts our signal in segments. Each one of these segments represent one or two tones.
For each one of the notes between two silences, we pick the time difference as given above and time-stretch the signal accordingly, using the phase vocoder approach with the scaled phase locking technique with inter peaks computation to avoid phasiness artiact. The segments are then resampled for the pitch shift, weighted for the amplitude modulation and added back together to create the new signal. As we're working with the onsets, we can not overlap-add the segments together, as in the phase vocoder approach, and must apply a fade-in/fade-out between the segments. And this is the reason why we are working with with silences instead of onsets. If we had worked directly with onsets, adding back the segments together would have resulted in a rubbling/taping sound, such as with the 50% overlap for the phase locking techniques, unusable for binaural application. That's why we used this detection instead of the COG onset detection, even if the COG is much more computational effective.
Moreover, the transient detection has also to be dealt with, as the beginning of the segment is not directly on an onset, and the attack transient is in the middle of a segment.
This method shows very good results. However, it appears to be much more computational costly than the previous algorithms, because of the onset detection that takes most of the time.

6 Time-scaling modification using time domain technique

Apart from frequency-domain techniques as the phase vocoder approach used in the previous sections, time-domain signal methods such as the TD-PSOLA (Time Domain - Pitch Synchronous OverLap and Add) method developped by E. Moulines and J. Laroche in [6] can be used.
This method modifies the pitch or duration of the signal by dividing it in small overlapping segments (defined by *analysis marks*) and replace them in the right positions.
In the following, an instrument m will be synthesised using the time-stretching approach (achieved by time-scaling the signal, then resample it to reach the desired pitch shift), but this time using the PSOLA method.

6.1 Usual time-scaling modification

With the PSOLA method, the signal is first analysed and divided in small segments, defined by the analysis marks t_a^k whch are placed on the waves composing the signal.
Then, if we want to alter the duration of the synthesised signal, we just have to duplicate or remove some segments, without changing the frequency, and collect these segments together using the overlap-add technique. The graph showing this modification is presented below:

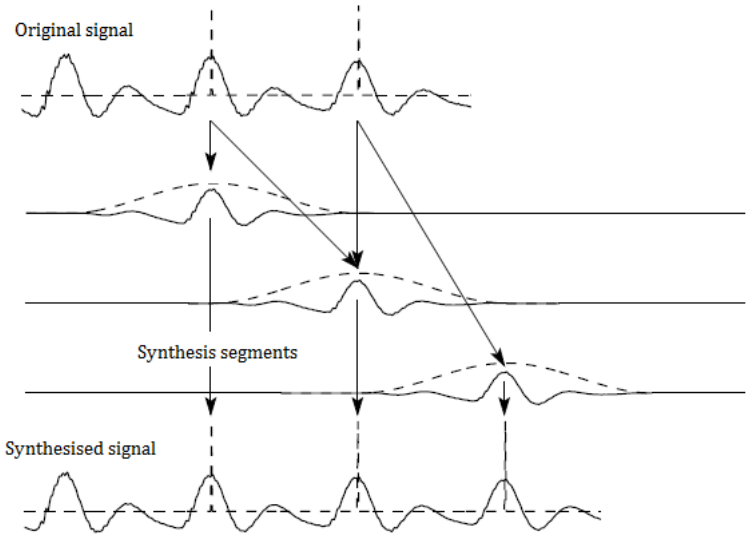


Figure 17: Duration modification with PSOLA method

The new segments will be indicated by the *synthesis marks* t_s^k.
The first step is to analyse our signal and set the analysis marks on its sinus waves, which assumes that we are knowing the signal's frequency. Once we've found the period at each analysis mark t_a^k: P_a^k, we set the next analysis mark on the next wave form, at $t_a^{k+1} = t_a^k + P_a^k$.
To find the period, we simply used the normalized auto-correlation formula:

$$\overline{r}_x(m) = \frac{\sum_{n=0}^{N-1-m} x(n)x(n+m)}{\sqrt{\sum_{n=0}^{N-1-m} x(n)^2}\sqrt{\sum_{n=0}^{N-1-m} x(n+m)^2}}$$

where x is the analysed signal.
With the normalized formula, we have $|\overline{r}_x(m)| \leq \overline{r}_x(0) = 1 \forall m$. If the signal is periodic, the auto-correlation will present peaks every m multiple of the period, and these peaks will have a value close to $max(\overline{r}_x(m)) = 1$.
In our case, the period will be computed on four periods of the signal $x(t_a^k : t_a^k + 4P_a^k)$ and the max of the auto-correlation m_f will be searched between $f_{min} = 50Hz$ (G1) and $f_{max} = 1000Hz$ (C6).
We set a threshold $\beta = 0.7$. If $\overline{r}_x(m_f) > \beta$, then the signal is considered periodic and $P_a^k = m_f$. Else, we consider that there is no harmonic signal, and P_a^k will be set to 10ms by default (100Hz), so that we can move forward faster than with a usual frequency ($f_0 > 100Hz$) without missing any note.
The auto-correlation method and f_{min} and f_{max} were chosen respectively for its low computational cost, and because higher frequency intervals yields in less precision in the results. For frequencies f_0 greater than f_{max}, we'll detect $f_0/2$ and move by two wave forms at each time, which is not a problem for high-pitch sounds.
Thus, the analysis marks will be set with the following algorithm:

The *spectral product* detection was also explored for the pitch detection, but its greater computational cost, as well as the instability of its results, depending on the spectral resolution, made us choose the auto-correlation method.
If we want now to modify the duration of the signal without changing its frequency, we will simply duplicate or remove the periods/wave forms marked during the analysis stage, and sum all of them using the overlap add technique.
As shown in 17, the original signal is separated in two segments, which are duplicated to obtain three overlapped synthesised segments. The original signal is then stretched by a factor of 1.5.

Algorithm 5 PSOLA Analysis marks

Steps

1: initialisation: $k = 0$, $t_a^0 = 1$, $P_a^0 = 10ms$
2: **while** $t_a^k < length(x) - 4P_a^k$ (the whole signal is processed) **do**
3: Increment $k = k + 1$
4: Find the period of excerpted signal $x(t_a^{k-1} : t_a^{k-1} + 4P_a^{k-1})$: P_a^k
5: Set the next analysis mark: $t_a^k = t_a^{k-1} + P_a^k$
6: **end while**

More generally, for the synthesis stage, we will define the *synthesis marks* t_s^k as the places where the wave segments will be in the synthesised signal, and a playback rate $n(k)$, which is how we are going to read our original signal.
These synthesis waves will be then summed to create the synthesised signal, using the overlap-add technique as follows:

$$y(t_s^k - P_a^{\lfloor n(k) \rfloor} : t_s^k + P_a^{\lfloor n(k) \rfloor}) = x(t_a^{\lfloor n(k) \rfloor} - P_a^{\lfloor n(k) \rfloor} : t_a^{\lfloor n(k) \rfloor} + P_a^{\lfloor n(k) \rfloor}) w_s \tag{28}$$

where w_s is the synthesis window, in our case a Hanning window of length $2P_a^{\lfloor n(k) \rfloor} + 1$, $\lfloor . \rfloor$ is the round operator, and y is the synthesised signal. The sinus waves are commonly windowed between the previous sinus wave $t_a^{\lfloor n(k) \rfloor} - P_a^{\lfloor n(k) \rfloor}$ and the next one $t_a^{\lfloor n(k) \rfloor} + P_a^{\lfloor n(k) \rfloor}$.
The next step is to define the synthesis marks t_s^k and the playback rate $n(k)$. For a time-scaling operation of a factor α, the algorithm is as follows:

Algorithm 6 PSOLA Synthesis marks

Steps

1: initialisation: $k = 0$, $n(0) = 0$, $t_s^0 = t_a^{\lfloor n(0) \rfloor} = 1$
2: **while** $t_a^{\lfloor n(k) \rfloor} < length(x) - 4P_a^{\lfloor n(k) \rfloor}$ (the whole signal is processed) **do**
3: Increment $k = k + 1$
4: Define the playback rate: $n(k) = n(k-1) + \frac{1}{\alpha}$
5: Set the next analysis mark: $t_s^k = t_s^{k-1} + P_a^{\lfloor n(k) \rfloor}$
6: **end while**

The essence of the PSOLA method lies in the way you are reading the signal with the playback rate. In the case where you are stretching the signal by a factor of $\alpha = 2$, we have $\lfloor n(k) \rfloor = \lfloor n(k+1) \rfloor$, for k odd, and the sinus waves at $t_a^{\lfloor n(k) \rfloor}$ will be duplicated twice at synthesis marks $t_s^k = t_s^k = t_s^{k-1} + P_a^{\lfloor n(k) \rfloor}$ and $t_s^{k+1} = t_s^k + P_a^{\lfloor n(k+1) \rfloor} = t_s^k + P_a^{\lfloor n(k) \rfloor}$, multiplying the signal's length by a factor of two, and keeping its spectral content, as the sinusoids are reproduced on the signal's periods.

6.2 Creation of an instrument section with PSOLA

To create a section of M instruments, we follow the same steps as for the PTA algorithm.
To simulate each instrument m, we'll do an inverse time-stretching approach, as for the PTA algorithm. The original recording will be analysed once for all instruments, following the *PSOLA analysis marks algorithm*, that will give us the analysis marks of our signal, and the every instrument will be time-stretched by a factor $\alpha = d^{(m)}$ and by adding time fluctuations with the PSOLA method.
The final signal will then be resampled by a factor $\frac{1}{d^{(m)}}$, back to its original length and with the desired pitch shift.
Adding time fluctuations with PSOLA will be quite easy, as we just have to remove or add periods in our synthesised signal to create the delays.

This method enables us to work directly on the onsets, as with the *replica algorithm* presented in the previous section, which is the best simulation, as instruments can only be delayed when they play notes, compared to a progressive and continuous way as modeled with the phase vocoder approach. We will use the same time-fluctuation model as for the *replica algorithm* in (27), with the delays δ following $\mathcal{N}(0, \sigma^2)$, where $\sigma = 40$ms.
To simulate these delays, we first did an onset detection, based on the transient detection and the COG computation, and saved the corresponding analysis marks $t_a^{k,l}$.
For each musician, during the synthesis stage, we then added or removed periods to the original recording to simulate these delays. To achieve this, we modified the playback rate $n(k)$ at each onset.

If the musician was late when we read the onset/analysis mark $t_a^{\lfloor n(k) \rfloor, l}$, $(\delta(l) > 0)$, we added $P_{nb} = \lfloor \frac{\delta(l)}{P_a^{\lfloor n(k-1) \rfloor}} \rfloor$ periods to the preceding sound (theoretically a silent sound). Moreover, as it is the first time we reach the onset, we know that $\lfloor n(k-1) \rfloor < \lfloor n(k) \rfloor$, and so that we take the period of the previous sound.
The algorithm is as follows:

Algorithm 7 Add periods

Steps

1: take period of the previous sound: $P = P_a^{\lfloor n(k-1) \rfloor}$
2: **for** $j \leq P_{nb}$ **do**
3: increment k = k+1
4: add synthesis marks: $t_s^k = t_s^{k-1} + P$
5: keep the playback rate as is: n(k) = n(k-1)
6: **end for**

Thus, we create more synthesised signal, while the playback rate n stays on the onset, ready to keep on reading the signal.
If we want to remove P_{nb} periods, we will just move backward and overwrite the P_{nb} last synthesis marks, while keeping the same playback rate. In practice, this will remove the end of the previous note or silence.

Algorithm 8 Remove periods

Steps

1: $k = k - P_{nb}$
2: $n(k) = n(k + P_{nb})$

The new synthesis marks will be overwritten, and the onset marked by the playback rate $t_a^{\lfloor n(k) \rfloor, l}$ will be played in the next loop.
When $t_a^{\lfloor n(k) \rfloor}$ is not an onset, the usual time-stretching modification is used with $\alpha = d^{(m)}$, leading to the desired instrument pitch-shift and time-fluctuations.
An amplitude modulation was also added in the overlap-add equation (28) following a low-pass filtered Metropolis-Hastings sampling at 5Hz, and whose length is the number of synthesis marks t_s.

6.3 Results

In practice, we had to take longer segments for the overlap-add synthesis, taking two periods below and above:

$$y(t_s^k - h.P_a^{\lfloor n(k) \rfloor} : t_s^k + h.P_a^{\lfloor n(k) \rfloor}) = x(t_a^{\lfloor n(k) \rfloor} - h.P_a^{\lfloor n(k) \rfloor} : t_a^{\lfloor n(k) \rfloor} + h.P_a^{\lfloor n(k) \rfloor}) w_s \qquad (29)$$

to prevent the synthesised sound from 'fuzzing' or 'rubbling'.
h was set to 2 and the synthesis window w_s was accordingly of length $2hP_a^{\lfloor n(k) \rfloor} + 1$. The only consequence was that the overall amplitude of the signal was twice higher, but a normalization by a factor $\frac{1}{h}$ made the signal back to its original volume.
The audio quality of the results were very good and the phasiness artifact, as we're working directly in time-domain on the waves of the signal, does not exist. Plus, working with an onset detection guaranteed for the transient attacks to be preserved. This would not have been the case if removing/adding periods had been done randomly on the signal, that could have been another solution, maybe deleting periods on the onsets.
But the main advantage of the PSOLA method is its computationnal cost. Time tests were done on the different algorithms, and the PSOLA method proved to be 1.5 to more than two times faster than with the phase vocoder approaches:

Algorithm	time
Classic PTA	49.55s
Improved PTA	75.62s
PSOLA	35s

Table 4: Computation duration for the different algorithms

Thus, the PSOLA method yields high quality audio results, avoids recurrent artifacts such as phasiness of transient smearing and guarantees a very low computational cost. Plus it enables, as opposed to the chorus effect, and such as the phase vocoder approaches, the time fluctuations, the pith-shifting and the amplitude modulation to be non-dependant and enables a great freedom in the choice of the parameters of our simulations. That makes him a very good candidate for high quality synthesis of polyphonic plays.
The audio results are also present in the folder 'Violin', at *PSOLA_12.wav* and *PSOLA_1.wav*. The code is present in *PSOLA_AnalysisMarks.m* and *PSOLA_SynthesisMarks.m* for the analysis and synthesis marks computation, and *PSOLA.m* for the whole algorithm.

Part II
Subjective listening tests on personal recordings

Once the algorithms implemented, the next step is to test them through subjective listening tests to see how the audio rendering is actually perceived.
The first part was to get our own recordings database, and to analyse them to get the parameters (pitch and onset distribution) of our instrument sections. The algorithms presented above were then run according to these parameters to create the audio files.
The second part was to develop a ABC-Hidden Reference (ABC-HR) listening test so that a database of listeners can evaluate the different algorithms.

1 Anechoic recordings and results of the algorithms

The first step was to get our own anechoic recordings. We decided to record violins and snare, to have percussive and non-percussive instruments, try the real effects of our algorithms and see if the artifacts (phasiness and transient smearing) were corrected.

1.1 Anechoic recordings

The first day, we recorded two violins in the anechoic chamber of the TU of Berlin. Preliminary tests were done to see what was the best position for the musicians, absorption panels and the best configuration of the microphones to minimize cross-talk, as they were recorded together.
A loudspeaker was placed in the position of the first musician emitting a white noise signal at a certain amplitude in dB. The microphone of the second musician was then placed in different positions around the area, recording the white noise signal and looking for the best position to minimise cross-talking. The best position was found to place the musicians in the north-east and north-west corners of the room, facing the conductor and with absorption walls in between them.

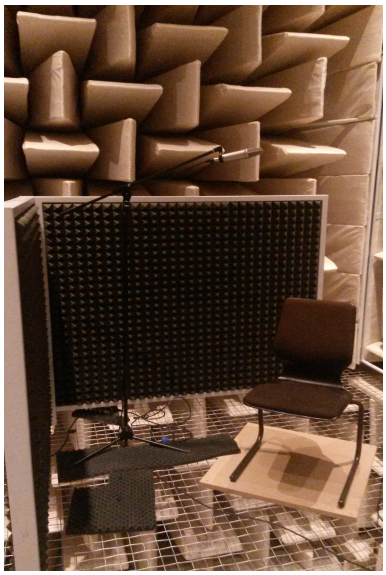


Figure 18: Configuration of the recording for a violonist

The musician were recorded with a microphone *Neuman U89*:

Figure 19: Microphone Neuman U89 used for the recording

where up to 6 different directivities could be chosen. The most used are:

- omnidirectionnal O takes the sound equally from all directions
- cardioid reduces sounds coming from behind
- 8 configuration cancels sounds coming perpendicularly to the membrane

In our case, the microphones were placed above the musicians, in the 8 configuration, so that sounds are only taken from below (the musician) and above (almost no signal from above, as most of the sound coming from the other musician is absorbed in the anechoic chamber), and sounds coming from the perpendicular (so eventually from the other musician) are annihilated due to the directivity.
Twelve mono recordings (as they are to be used for binaural synthesis), six for each musician, were performed during this session for the tests.
Furthermore four recordings of a snare drum were recorded. As there was only one musician, no preparation of the room was needed and the recordings were done directly in the anechoic chamber with a condenser microphone in 1m distance from the snare drum.

1.2 Analysis of the recordings

The next step was to analyse our recordings to excerpt the parameters of our own onset and pitch distributions.

1.2.1 Onset distribution

The onsets were marked manually for all 12 recordings, thanks to the freeware *Sonic visualiser*.
For each onset, the reference was taken as the mean of the onsets, and the distribution around these references was observed. The onset detection results were then analysed through the existing package *allfitdist.m* that detects the closest distribution.
The result is given below:

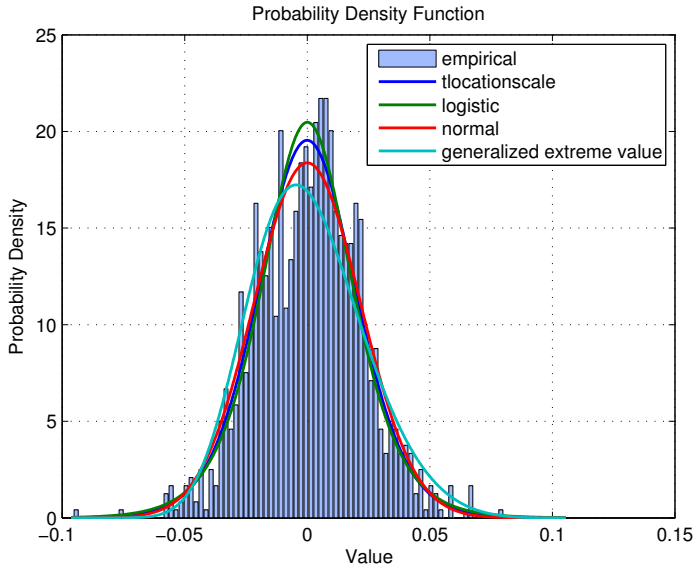


Figure 20: Onset distribution

So we see that our onset distribution follows a normal distribution $\mathcal{N}(\mu, \sigma^2)$ centered in $\mu = 0$ and of standard deviation $\sigma = 22ms$.

1.2.2 Pitch distribution

To detect the pitch distribution, we had to use a high precision fundamental frequency estimator. The auto-correlation used for the PSOLA method clearly not being precise enough.
To achieve this, we used the YIN algorithm described by Albert de Cheveigné and Hideki Kawahara in [11]. This high-precision algorithm stands on several steps that consist in using the Average Square Difference Funtion (ASDF):

$$d_t(\tau) = \sum_{j=t+1}^{t+N} (x_j - x_{j+\tau})^2$$

where N is the length of the analysed signal and normalize it, by setting $d_t(\tau) = 1$ if $\tau = 0$ and

$$d_t(\tau) = \frac{d_t(\tau)}{\frac{1}{\tau}\sum_{j=1}^{\tau} d_t(j)}$$

for the rest.
The ASD function is much less sensitive to the amplitude of the partials than the usual Auto-Correlation technique.
Then the minimum is taken according to a threshold and a parabolic interpolation is done to find the fundamental frequency with even more precision.

The package *Yin* provided by A. de Cheveigné and al. [11] was used. The algorithm was run on all the signals with a buffersize of 10000 samples, and a 48 samples hop factor. The minimum frequency f_{min} and maximum frequency f_{max} were respectively set to 30Hz (B0) and 5000Hz (E8). The algorithm also provides a normalized aperiodicity measure a_p for each frames, and a sound is defined periodic when $a_p < 0.1$ by default in the algorithm.
Thus, for all violins, we looked upon intervals/ensemble of frames where $a_p < 0.1$ (which corresponded to a note) and picked on these intervals the frame where the aperiodicity rate was at its minimum. The chosen fundamental frequency f_0 was the corresponding one, as are doing A. de Cheveigné and al. to give the 'best' frequency detected for the signal.

We then collected the pitch-shift in cents that separated these frequencies from the closest tuned note, based on the A440, and plot their histogram:

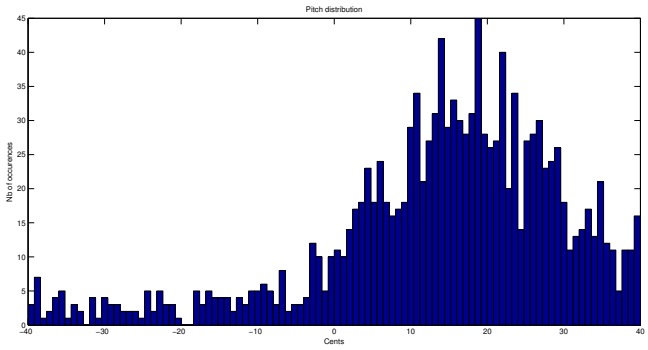


Figure 21: Pitch distribution showing a general detune

The pitch distribution clearly shows a general detune in our recordings, that are approximately 20 cents above the original tune based on the A440.
Thus, we decided that all violins were on average around 20 cents, so that the pitch shift factor was 1, and removed the average detune to each violin to fit the new distribution. Cents above 40 cents and below -40 cents were removed, as corresponding to a wrong note detection.

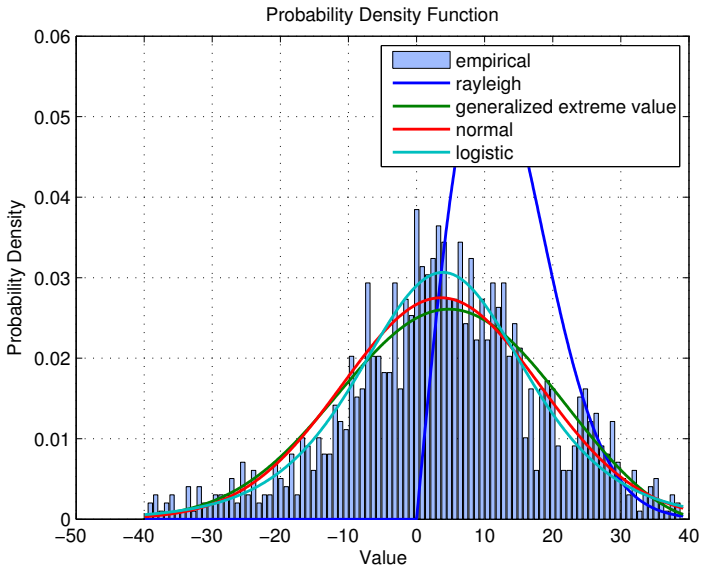


Figure 22: Pitch distribution centered

The desired algorithms to compute the pitch distribution are in the script *pitchdistribution.m* and added in the function *yin.m* given by the package.
The normal distribution fitted our distribution at the third place, with the *allfitdist.m* algorithm. Thus we chose it and the parameters were $\mu \simeq 0$ (all recordings have the same detune) and $\simeq 14.51 cents$, so a pitch shift of $2^{12.51/1200} = 1.0071$ around the original one.
Thus, the pitch shift factors were picked following $\mathcal{N}(1, \sigma^2)$ with $\sigma = 0.0071$.

1.3 Computation with the algorithms

Using these onset and pitch distributions as parameters for our time difference and pitch shift operations, we ran on our 12 violin recordings and 4 snare recordings, the algorithms that showed

best results during our study: PTA, PTA Phase locking - inter peaks, improved PTA and PSOLA. The STFT consistency and execution time are given below:

Phase Locking technique	D_m	time
Classic PTA	-37.5dB	14s
PTA Scaled PhaseLock - Inter Peaks	-33dB	21.40s
Improved PTA	-33dB	23.05s
PSOLA	/	12.30s

Table 5: STFT consistency and computational time for violin recordings

Phase Locking technique	D_m	time
Classic PTA	-30dB	11.41s
PTA Scaled PhaseLock - Inter Peaks	-39.5dB	18.36s
Improved PTA	-42dB	18.43s
PSOLA	/	7.34s

Table 6: STFT consistency - computational time for snare recordings

The results obtained are coherent with what we found during the testing parts. However, listening tests showed that PSOLA method and Improved PTA showed results are not as good as they were in the test part for the violin. In particular, the violins were much more delayed for the PSOLA method, which is not acceptable for a violin section. This was caused by a wrong onset detection. And indeed, when we look at the COG of the violin:

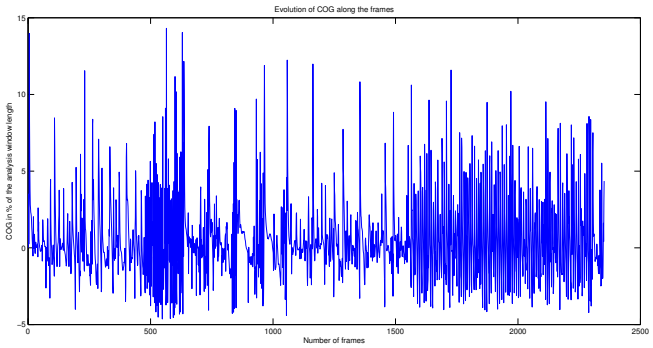


Figure 23: COG for the recorded violin

The COG of the recorded violin appears to be much more noisy than with the test one. The COG is also less identifiable (the max is around 15%, less than the 25% for the elder violin). As a result, the threshold $max(COG)/2$ does not detect enough onsets (only 22 against 120 real ones), and delays on the onsets detected can quickly lead to a complete mess between the musicians. As a result, we had to choose $\sigma = 10ms < 22ms$ to compensate the errors. Here, we see that the PSOLA method is very dependant on the onset detection and another one, such as the one used by the replica algorithm would have worked perfectly with the method and could have yielded much better results. Actually, the replica algorithm yielded results as good as the testing recordings. We did not have the time to run the tests again with this configuration, but we see that a significant improvement can be brought to PSOLA by developing a good and stable onset detection.
This bad COG detection also affected the improved PTA algorithm, creating more phasiness on the onsets, than it was actually correcting. Listening tests on the scaled phase locking algorithm proved better results. Thus, as opposed to the first part, we chose the scaled phase locking

algorithm for our recorded violins as the 'improvedPTA' algorithm.
Finally, for the snare, the STFT consistency was surprisingly greatly improved with the phase locking algorithms. When listening to the results though, the phase locking techniques created weird effects, that can be described as 'laser' effects on snare transient attacks compared to the PTA algorithm.
Indeed, the phase locking techniques try to give back coherence to the phase, concentrating on the peaks. However, it is known that a percussive snare spectrum is much more like a noise sound than an harmonic sound. Plus, the frequencies can be very close such that the frequency resolution of the FFT is often not sufficient to detect all the frequencies composing the signal. That is why we prefer to use high-resolution methods for analysis percussive instruments. Thus, a wrong peak, so frequency detection, coupled with providing a phase coherence where it does not exist can create unnatural percussive sound, making the phase locking techniques not a good choice for percussive instruments.
However, the transient detection described by A.Röbel clearly restored the attack transient for the snare sound (whose COG was, as opposed to the violin really clear, as seen in fig 13), making it sound more punchy as with the classical PTA algorithm. The 'improved PTA' for a snare was then chosen to be the classical PTA, without phase locking in which we added the transient detection.
As for the testing part, the audio results can be found respectively in the folders 'Snare' and 'Violins' and follow the same notation *'method_ nbInstruments.wav'*.

2 Subjective listening tests

Subjective listening tests were then conducted to a panel of 30 people to check how our algorithms were actually perceived.
The people were asked to assess 60 conditions according to how they resemble to two references. The two references were simply the two instrument sections composed of the 12 different violins and the 4 different snares.
The section effect was obtained by multiplying one of the violin recordings 12 times to obtain a violin section. Apart from the different performance of each algorithm, we wanted to know how the number of original recordings in a section simulation could impact the perceived effect, and from which one the effect is perceived as close to the reference. As an example, we created two other sections with 2 and 3 different violins replicated 6 and 4 times respectively, to achieve our 12 violins sections.
The 60 conditions were composed of:

- 5 algorithms: PSOLA, PTA, improvedPTA, Replica, Chorus.
- 2 rooms: the anechoic sections were placed in two different acoustic environments: a dry room and a concert hall
- 2 instruments: snare, violin.
- up to 3 different instruments for the sections: 1 instrument times 12, 2 different instruments times 6, 3 different instruments times 4.

In practice, for the snares, we only went up to 2 different recordings. A chorus effect with the low-pass random delay line developed in [2] was also implemented, and expected to sound as the anchor (the furthest simulation from the reference).

2.1 Integration in acoustic environments and binaural synthesis

For each algorithm were returned 12 mono audio results for the 12 violins generated. These violins were then placed in virtual acoustic environments and binaural room impulse responses (BRIRs) gathered by the software RAVEN developed in Institute for Technical Acoustics in Aachen. [12], and following the work of S. Wienzierl and al. (the interested reader can learn more in [13], [14]

and [15]).
The BRIR's are created by emitting a Dirac impulse signal at the position of the source and simulating the first reflections of the room with Ray-Traycing following the geometrical acoustics model and further simulating the reverberation according to statistical acoustics. For each degree of horizontal inclination of the listener (360), an impulse response for both ears is recorded using the head related transfer function of the dummy head FABIAN created at the TU Berlin, resulting in 720 impulse responses.
The different sources are situated on the stage and the listener in the auditorium, at a distance of twice the 'critical distance' (distance where the direct sound is equal to the diffuse sound) from the closest source, so the reverberation of the room is guaranteed.
The listener uses special headphones for binaural synthesis:

Figure 24: Headphones for binaural synthesis with head-tracker

The headphones are connected with a head-tracker that follows the head inclination, and the corresponding impulse responses of the BRIR (depending on the room) are convoluted with the violin or snare signal by the software SoundScape Renderer [16], simulating a full-immersion experience, as if the listener is sitting in the room listening to the instrument section.

2.2 ABC-Hidden Reference listening tests

The tests were then implemented as ABC-Hidden reference tests, and set up thanks to the WhisPER software developed in TU Berlin. This software enables you to choose between the different types of listening tests (AB, MUSHRA, ABC-HR) and helps you connect your listening conditions to a user-friendly interface that the listener will use. A pureData patch was developed to connect buttons of the WhisPER interface to markers that read our different conditions through the freeware Ardour (open source digital audio workstation).
With the ABC-Hidden Reference test, for each condition, you have 3 recordings: the reference, a hidden reference and the recording we want to assess. The interface is given below:

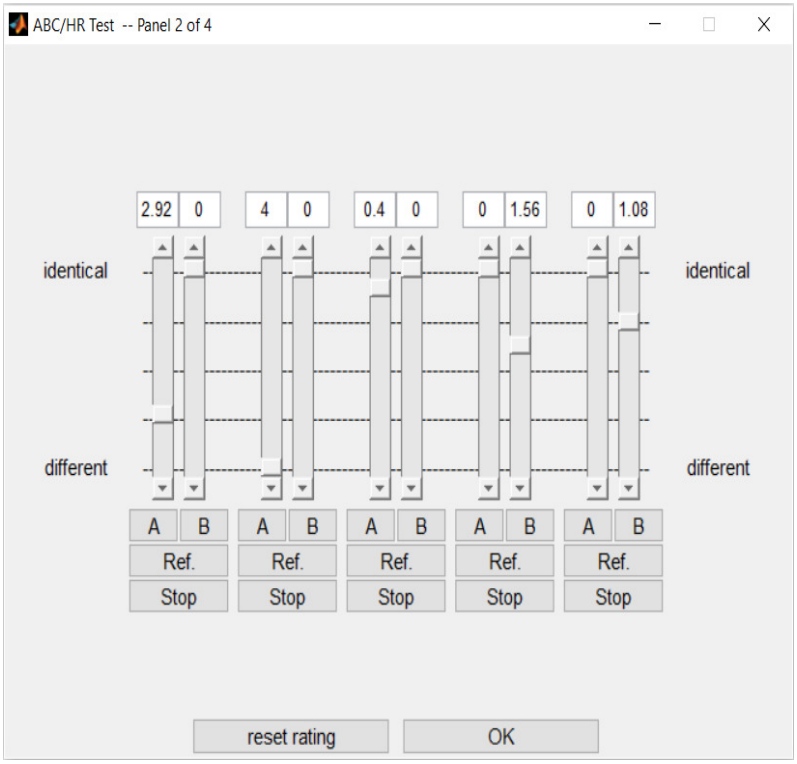


Figure 25: ABC-Hidden Reference tests

For each tests, the reference is given and the listener can listen to it whenever he wants. Then from the two recordings, he has to pick the one he thinks is the hidden reference, set the quality to the max, and rate the other one in comparison. 5 conditions are given each time on the same interface, so in addition with the comparison with the reference, the listener can compare the different algorithms in between them. Hiding the reference helps us know is the listener can actually hear a difference (whether the simulation is very good if the listener is on average right about the hidden reference, or the listener hasn't a good perception if he is on average wrong about the reference).
The listening tests are to be conducted between the first and the last week of July, so the results are unfortunately still expected.

2.3 Additional information

Finally, in the end of the tests, the listeners were asked to answer a survey, on the web address:
`https://www2.ak.tu-berlin.de/musikstudie/umfrage/index.php/survey/index/sid/996748/newtest/Y/lang/de`

This test was simply to gather basic sociological information about our listeners (age, education, gender), and to know what was the listener's relation to music (musician or involved in musical production, how often he listens to music), to assess his musical and listening expertise.

Conclusion

In this study, we started from the latest developments in terms of high audio quality section simulation with the PTA algorithm, and improved them with a better understanding of the phase vocoder. We successively got rid of some characteristic artifacts of the phase vocoder such as phasiness and transient smearing, in order to always provide better quality results for binaural and acoustic simulations.
A completely new approach for instrumental section simulation, using the time-domain PSOLA method was also revealed and shown to yield excellent audio results. The new techniques were evaluated and were shown to provide either a better audio rendering and/or a better computational cost.
Moreover, the goal of this study is eventually to create a full anechoic orchestra for full concert immersion, such as the synthesis of instrument sections must be done for all the different type of instruments composing a symphonic orchestra. By working on a percussive snare instrument, we discovered that techniques supposedly good for violin or polyphonic sounds in the literature might not be adapted for anechoic percussive instruments. Thus, the techniques must be adapted to each type of instruments and studies on all instruments composing an orchestra are still needed. For percussive instruments for example, a synthesis with high-resolution methods can be worth considering, still based on the pitch, time and amplitude modification proposed with the PTA algorithm.
Finally, we showed that for onset-based algorithms, the dependence with the original recording quality can be strong. As we're working with anechoic solo recordings, a stable and better onset detection that improves the proposed algorithms might be found.
In the end of this study, listening tests were conducted to check with the theory and if the algorithms were actually perceived as better. In the time this report is being written, the listening tests are not done yet, but the results will be submitted in the same time as the presentation in beginning of September.

Bibliography

[1] J. Meyer. *Acoustics and the performance of music*. New York, NY, USA, 2009.

[2] Jukka Pätynen, Sakari Tervo, and Tapio Lokki. "Simulation of the violin section sound based on the analysis of orchestra performance". In: *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (Oct. 2011).

[3] Jean Laroche and Mark Dolson. "Improved Phase Vocoder, Time-Scale Modification of Audio". In: *IEEE Transactions on speech and audio processing* 7.3 (May 1999), pp. 323–332.

[4] Jean Laroche and Eric Moulines. "Non parametric techniques for pitch-scale and time-scale modification of speech". In: *Speech commun* 16 (Feb. 1995), pp. 175–205.

[5] Alex Röbel. "A new approach to transient processing in the phase vocoder". In: *Proc. of the 6th Int. Conference on Digital Audio Effect (DAFx-03)* (Sept. 2003).

[6] E. Moulines and J. Laroche. *Non parametric techniques for pitch-scale and time-scale modification of speech*. Vol. 16. Feb. 1995, pp. 175–205.

[7] U. Zölzer. *DAFX: Digital Audio Effects*. Ed. by John Wiley Sons. Chichester, United Kingdom, 2001.

[8] E. Greenberg S. Chib. "Understanding the Metropolis Hastings algorithm". In: *American Statistician* 49.4 (1995), pp. 327–335.

[9] François Auger and Patrick Flandrin. "Improving the readability of time-frequency and time-scale representations by the reassignment method". In: *IEEE Transactions on speech and audio processing* 43.5 (May 1995), pp. 1068–1089.

[10] J. P. Bello et al. "A tutorial on onset detection in music signals". In: *IEEE Tr. Speech and Audio Processing* 13.5 (2005), pp. 1035–1047.

[11] A. de Cheveigné and H. Kawahara. *YIN, a fundamental frequency estimator for speech and music*. Vol. 111. Apr. 2002, pp. 1917–1930.

[12] D. Schroder. "Physically Based Real-Time Auralization of Interactive Virtual Environments". In: (2011).

[13] Stefan Weinzierl et al. "The Acoustics of Renaissance Theatres in Italy". In: 101 (2015), pp. 632–641.

[14] Clemens Büttner and Stefan Weinzierl. "The acoustics of early concert venues in Japan". In: *DAGA* (2010).

[15] Clemens Büttner et al. "Acoustical characteristics of preserved wooden style Kabuki theaters in Japan". In: (Sept. 2014).

[16] M. Geier and al. "The SoundScape Renderer: A United Spatial Audio Reproduction Framework for Arbitrary Rendering Methods". In: (2008).

Annexes

Recovering the instantaneous frequency of a sinusoid, using the *phase unwrapping* method of the phase vocoder

Our first hypothesis is that the signal follows the M^{c}Aulay and Quatieri signal model, that assume that the signal is a sum of sinusoids:

$$x(t) = \sum_{i=1}^{I(t)} A_i(t) e^{j\psi_i(t)} \tag{30}$$

with I(t) the number of sinusoids composing our signal at time t, $A_i(t)$ and $\psi_i(t)$ being respectively the amplitude and the so-called *instantaneous phase* of the ith sinusoid.
The instantaneous phase $\psi_i(t)$ is directly related, with the well-known equation to the instantaneous frequency $\omega_i(t)$, that we want to guess:

$$\omega_i(t) = \frac{d\psi_i}{dt} \tag{31}$$

The phase vocoder stands on three hypothesis:

- First, that characteristics of the sine waves (amplitude, frequency) vary slowly in time, it's the *stationary* hypothesis. Thus, on the length of the analysis window w_a, so for $n \in [\![0, N-1]\!]$, we have on the time-instants t_a^u:

$$\begin{cases} A_i(t_a^u + n) \simeq A_i(t_a^u) \\ \psi_i(t_a^u + n) \simeq \psi_i(t_a^u) + \omega_i(t_a^u).n \end{cases} \tag{32}$$

  Thus, the STFT of this signal on the analysis window will be:

$$X(t_a^u, \Omega_k) = \sum_{i=1}^{I(t_a^u)} A_i(t_a^u) e^{j\psi_i(t_a^u)} W_a(e^{j(\Omega_k - \omega_i(t_a^u))}) \tag{33}$$

  where $W_a(e^{j\omega})$ is the FFT of the analysis window w_a.

- The second hypothesis considers $N = N_{fft}$ big enough so that there's only one sinusoid I in a channel k:

$$\exists! I / |\Omega_k - \omega_I(t_a^u)| \leq \omega_h \tag{34}$$

  where ω_h is the cutoff-frequency of the analysis window w_a. This is the *narrow-band* hypothesis and stands when $N \geq \frac{4}{f_0}$, where f_0 is the normalzed fundamental frequency $\frac{F_0}{F_s}$ of the analysed sound. In the case of an instrument analysis, a good f_0 is the lowest frequency reachable by the instrument.
  In that case, we can greatly reduce equation (33):

$$X(t_a^u, \Omega_k) = A_I(t_a^u) e^{j\psi_I(t_a^u)} W_a(e^{j(\Omega_k - \omega_I(t_a^u))}) \tag{35}$$

  Here, as w_a is chosen as symmetric around 0, we know that W_a is real and thus that $\angle W_a(e^{j(\Omega_k - \omega_I(t_a^u))}) = 0$.
  Thus, the STFT phase $\angle X(t_a^u, \Omega_k)$ equals the instantaneous phase $\psi_I(t_a^u)$, up to an integer multiple of 2π for $|\Omega_k - \omega_I(t_a^u)| \leq \omega_h$. We can then compute the instantaneous frequency $\omega_I(t_a^u)$, using $\omega_I(t) = \frac{d\psi_I}{dt}$ and deriving it between two consecutive analysis frames where the stationary hypothesis is still true ($\Delta t_a^u \leq N$):

$$\angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) = \psi_I(t_a^u) - \psi_I(t_a^{u-1}) + 2n\pi = \omega_I(t_a^u)\Delta t_a(u) + 2n\pi \quad (36)$$

Here, the instantaneous frequency is available through the phases of the STFT up to an integer n. And that's this integer that we would like to determine to find the right instantaneous frequency.
To achieve this, we'll unwrap the phase, hence the name *phase unwrapping* method:

$$\angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) = \Omega_k \Delta t_a(u) + (\omega_I(t_a^u) - \Omega_k)\Delta t_a(u) + 2n\pi \quad (37)$$

As the sinusoid I falls in the channel k, we have:

$$|(\Omega_k - \omega_I(t_a^u))\Delta t_a(u)| < w_h \Delta t_a(u) \quad (38)$$

- Finally, we assume that $\Delta t_a(u)$ is such as $w_h \Delta t_a(u) < \pi$ (that is the so called *unwrapping hypothesis*), the unwrapping equation (37) leads to the *heterodyned phase* computation. That is to say that $\exists! n/$:

$$|\angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) - \Omega_k \Delta t_a(u) - 2n\pi| = |\Delta_p \Phi_k^u| < \pi \quad (39)$$

which leads to the instantaneous phase $\omega_I(t_a^u)$ once n is determined to satisfy the above equation with:

$$\omega_I(t_a^u) = \Omega_k + \frac{\Delta_p \Phi_k^u}{\Delta t_a(u)} \quad (40)$$

Thus, we can find the instantaneous frequency $\omega_I(t_a^u)$ of the sinusoid I (that is composing the signal at time-instant t_a^u) in the channel k of the vocoder, assuming $|\Omega_k - \omega_I(t_a^u)| \leq w_h$. This thanks to the *heterodyned* phase that represents the small phase shift between the center frequency Ω_k of the channel and the frequency of the nearby sinusoid I.
In practice, we compute the instantaneous frequencies for all channels k in the phase vocoder: $\hat{\omega_k}(t_a^u)$, and the instantaneous frequency in the kth channel equals the instantaneous frequency of the sinusoid I that falls into the channel.

Transient detection

The detection of the transients, as the first step of the method proposed by A. Röbel [5], stands on the *time-reassignment operator* $\gamma : (t, \omega) \mapsto (\hat{t}, \hat{\omega'})$ developped by a Auger and Flandrin in 1995 [9] for a time-frequency representation.
Auder and Flandrin (1995) define a very general technique to detect the COG that applies for all time-frequency representations. In this section, we'll just explain the case corresponding to the STFT/spectrogram time-frequency representation. The analysis window note w_a will be of type Hanning and will be used as the filter for the region around the point (t, ω). In this specific case, the reassigned values $\hat{t}(t, \omega)$ and $\hat{\omega'}(t, \omega)$ are as follows:

$$\hat{t}(t, \omega) = t - \frac{\partial \phi_{w_a}(t, \omega)}{\partial \omega} \tag{41}$$

$$\hat{\omega}(t, \omega) = \omega + \frac{\partial \phi_{w_a}(t, \omega)}{\partial t} \tag{42}$$

where $STFT_{w_a}(x, \omega) = A_{w_a}(x, \omega) e^{j \phi_{w_a}(x, \omega)}$. As we're working with Discrete Fourier Transform (DFT), Auder and al. [9] proposed an alternative formula, that computes $\hat{t}(t, \omega)$ and $\hat{\omega}(t, \omega)$ with STFT directly:

$$\hat{t}(t, \omega) = t - \Re\left(\frac{STFT_{\mathcal{T} w_a}(t, \omega) . \overline{STFT_{w_a}(t, \omega)}}{|STFT_{w_a}(t, \omega)|^2}\right) \tag{43}$$

where $\mathcal{T} w_a(t) = t . w_a(t)$ is the analysis window multiplied by a time ramp.
A. Röbel [5] takes this formula, and assume that the time origin is at the center of the frame, causing t = 0 relatively to the analysis frame. As a result, we have:

$$\hat{t}(t, \omega) = 0 - \frac{\partial \phi_{w_a}(t, \omega)}{\partial \omega} \tag{44}$$

and:

$$\hat{t}(t, \omega) = 0 - \Re\left(\frac{STFT_{\mathcal{T} w_a}(t, \omega) . \overline{STFT_{w_a}(t, \omega)}}{|STFT_{w_a}(t, \omega)|^2}\right) \tag{45}$$

Finally, as A. Röbel is concentrating only on the time assignment, the Center Of Gravity is achieved by integrating this new time on the frequency axis, and weighting by the amplitudes, and is applied to the center of each STFT frame placed at t_a^u:

$$t_{COG}^u = \frac{\int -\frac{\partial \phi(t_a^u, \omega)}{\partial \omega} A(t_a^u, \omega)^2 \mathrm{d}\omega}{\int A(t_a^u, \omega)^2 \mathrm{d}\omega} \tag{46}$$

noting $\phi_{w_a}(t_a^u, \omega) = \phi(t_a^u, \omega)$ and $A_{w_a}(t_a^u, \omega) = A(t_a^u, \omega)$, with:

$$-\frac{\partial \phi(t_a^u, \omega)}{\partial \omega} = -\Re\left(\frac{STFT_{\mathcal{T} w_a}(t_a^u, \omega) . \overline{STFT_{w_a}(t_a^u, \omega)}}{|STFT_{w_a}(t_a^u, \omega)|^2}\right) \tag{47}$$

Auder and al. [9] give us a simple way to compute the COG in practice, using the STFT at time-instant t_a^u. Moreover, using the latter formula, A. Röbel [5] shows that if we derivate the spectral energy $|STFT_{w_a}(t_a^u, \omega)|^2$ with respect to window position t_a^u and normalize by the spectral energy, we obtain:

$$-\frac{\partial |STFT_{w_a}(t_a^u, \omega)|^2}{|STFT_{w_a}(t_a^u, \omega)|^2 \partial t_a^u} = -2\Re\left(\frac{STFT_{\mathcal{D} w_a}(t_a^u, \omega) . \overline{STFT_{w_a}(t_a^u, \omega)}}{|STFT_{w_a}(t_a^u, \omega)|^2}\right) \tag{48}$$

which besides a factor of two can be derived from the previous equation by replacing the STFT of the window $\mathcal{T} w_a$ with the STFT of the derivative $\mathcal{D} w_a$ of the analysis window. As $\mathcal{T} w_a$ and

$\mathcal{D}w_a$ are relatively similar functions, the two equations above are related, proving that the phase based and the energy based criteria are related. The phase based criteria is thus a good method in the transient attack detection, especially for a phase-vocoder approach.