UNIVERSITÀ DI PADOVA FACOLTÀ DI INGEGNERIA

TESI DI LAUREA

A TRANSIENT-PRESERVING AUDIO TIME-STRETCHING ALGORITHM AND A REAL-TIME REALIZATION FOR A COMMERCIAL MUSIC PRODUCT

Laureando: Alessandro Altoè

Relatore: Prof. Federico Avanzini

Corso di Laurea Magistrale in Ingegneria Informatica

11 Dicembre 2012

Anno Accademico 2012/2013

A mio padre e mia madre.
Grazie per avermi fatto crescere in una casa piena di libri e di stereo.

Abstract

Time scaling an audio signal without altering the signal pitch and the perceived sound quality is a classical research topic, that has been studied from the ’50s [48]. Techniques to perform this kind of time scaling go under the name of time stretching, since the signal is “stretched” (expanded or compressed) keeping the original perceptual attributes.

In literature many time stretching techniques and variants have been presented, but, since the intrinsic difficulty of the problem, time-stretching is still subject of research. The choice of which technique to use is strongly application dependent, since different algorithms targets different kind of sound sources (e.g. monophonic or polyphonic instruments), and each one shows intrinsic pros and cons.

The purpose of this work is to integrate a time stretching functionality into the software of an electronic musical instrument with a tabletop Tangible User Interface named Reactable, produced by Reactable Systems (Barcelona, Spain). This is particularly challenging, since the time stretching routine should target any kind of sound source, and must be carefully implemented in order to work real time within all Reactable tasks without introducing latency.

With this in mind, the phase vocoder [37] has been selected as the best fitting time stretching technique to be integrated. Phase vocoder is a time-scaling technique that is based on the *Short Time Fourier Transform* (STFT) representation of the signal, and it can produce high quality time stretching for any sound source.

Unfortunately the phase vocoder suffers of two major drawbacks: phase dispersion [1] and transient smearing [41].

Phase dispersion indicates that the phase relationship between different STFT bins are not preserved, causing an annoying reverberant artifact called phasiness. Phasiness can be dramatically reduced using a real-time inexpensive technique named rigid phase locking [26], whose patent is owned by Creative Technology Ltd. [13]. In order to not infringe any intellectual property, a phase locking technique based on a sinusoidal model [46] of the signal has been developed for Reactable.

Transient smearing is a much more complex problem to solve real-time. Transient smearing is an artifact caused by the time-scale and spectral modification of attack transients; the procedure to avoid this artifact is called transient preservation. Before applying some transient preservation technique, obviously, attack transients must be detected and located both in time and frequency domain (*transient detection*).

Transient detection, even if it is at the bottom of many audio DSP algorithm, is a non-trivial problem. It can be performed directly in the STFT or more accurately through wavelet analysis,

and often requires to analyze in advance relatively large portions of the input signal. For this reason in many real-time time stretching implementations transient detection is performed on the whole audio signal off-line, and results are stored in order to perform real-time the transient-preserving time stretching algorithm[5] [40] [19].

Once transients are detected, it is important to apply a preservation procedure that, at the same time, accurately reconstructs the original attack transients and does not impair the stable regions of the signal.

The core of this work is a sub-band transient detection/preservation scheme based on the *complex domain transient detection* [3] [53], and inspired by Röbel's work [42]. This proposed technique can be integrated in a real-time phase vocoder analysis/synthesis scheme without introducing latency at relatively low computational costs.

Additional real-time techniques to improve the phase vocoder quality will be discussed, as stereo image preservation and window envelope compensation.

The resulting time stretching algorithm is formally tested against two of the most renowned real-time commercial library, achieving the best score.

The thesis is organized as follows. Chapter 1 introduces the Reactable and the design requirements for the time stretching functionality. Chapter 2 presents the fundamental time scaling techniques, in both time and time-frequency domains. Chapter 3 discusses in detail the phase vocoder and the state-of-the-art of phase vocoder improvements: phase locking, transient detection and preservation techniques. The main original contributions of this work are presented in chapter 4. Here, the proposed phase locking and the transient detection/preservation scheme are discussed in detail, along with a proposed variant of *rigid phase locking* and other phase vocoder improvements. Furthermore chapter 4 discusses a real-time zero-latency sample-accurate realization of the resulting algorithm, and formal listening test results.

Sommario

Modificare la durata di un segnale audio senza alterarne l'intonazione o gli attributi percettibili è un argomento di ricerca studiato fin dagli anni '50 [48]. Per ottenere questo tipo di modificazione temporale si utilizzano tecniche di *time stretching*, attraverso le quali il segnale viene "stirato" (espanso o compresso), senza che ne vengano alterati gli attributi originali.

In letteratura sono presenti molte tecniche di time stretching, ma, data la difficoltà intrinseca del problema, esse sono ancora oggetto di studio. Scegliere quale utilizzare dipende fortemente dall'applicazione, in quanto differenti algoritmi hanno come target differenti sorgenti sonore (per esempio strumenti monofonici o polifonici), ed ognuno presenta pregi e difetti.

Lo scopo di questo lavoro è stata l'integrazione della funzionalità di time stretching nel software di uno strumento musicale elettronico con interfaccia tangibile, che si chiama Reactable ed è prodotto da Reactable Systems (Barcellona, Spagna). L'integrazione di tale funzionalità ha richiesto parecchio sforzo dato che deve garantire buoni risultati con ogni tipo di sorgente sonora, e che la routine di time stretching deve svolgersi in tempo reale assieme a tutti i processi di Reactable senza introdurre latenza.

Tenendo conto di questo, è stato scelto il phase vocoder [37] come il metodo più adatto per essere implementato. Esso è una tecnica che si basa sulla Short Time Fourier Transform (STFT) del segnale e può produrre stretching di ottima qualità per qualsiasi tipo di sorgente sonora.

Sfortunatamente il phase vocoder soffre di due problemi: la dispersione di fase [1] e il *transient smearing* [41].

La dispersione di fase si riferisce al fatto che il phase vocoder non mantiene le relazioni tra le fasi di differenti bin della STFT, causando un artefatto chiamato phasiness. La phasiness può essere fortemente ridotta usando una tecnica chiamata *rigid phase locking* [26], il cui brevetto è proprietà di Creative Technolgy Ltd. [13].

Per non infrangere nessuna proprietà intellettuale, ho sviluppato e integrato nel software di Reactable una tecnica di phase locking basata su modelli sinusoidali del segnale.

Il transient smearing è un problema piú complesso da risolvere in tempo-reale. Esso è l'artefatto causato dalla modificazione spettrale e temporale dei transitori d'attacco. La procedura per evitare che ciò accada si chiama *transient preservation*. Ovviamente, prima di applicare qualsiasi procedura di transient preservation, è necessario localizzare i transitori d'attacco in tempo-frequenza tramite una tecnica di *transient detection*. Localizzare i transitori, nonostante sia alla base di molti algoritmi di DSP, è tutt'altro che un problema semplice. La transient detection può svolgersi analizzando la STFT del segnale o in modo più accurato tramite l'analisi *wavelet*. Spesso essa richiede di analizzare in anticipo porzioni relativamente estese del segnale d'ingresso. Per questo motivo in molte realizzazione di time-stretching la transient detection viene svolta off-line sull'intero segnale, memorizzandone i risultati, per poi poter effettuare il time-stretching con transient preservation in tempo reale [5] [40] [19].

Una volta che tutti i transitori sono stati localizzati è importante applicare una procedura di transient preservation che ricostruisca accuratamente i transitori di attacco senza degradare la qualità delle regioni stazionarie del segnale.

Il nocciolo di questo lavoro è uno schema a sottobande di transient detection/preservation basato sulla localizzazione dei transitori nel dominio complesso [3] [53] e ispirato al lavoro di

Röbel [42]. Tale schema, proposto nella presente tesi, può essere integrato nel phase vocoder senza introdurre latenza ad un costo computazionale relativamente basso.

Altre tecniche per migliorare la qualità del phase vocoder, come la compensazione dell'effetto delle finestre e la preservazione della stereofonia, verranno presentate e discusse nel corso di questa tesi.

L'algoritmo di time stretching risultante è stato infine testato formalmente con due delle più note librerie real-time di time stretching, ottenendo ottimi risultati.

Questa tesi è organizzata come segue:

- il capitolo 1 introduce Reactable e i requisiti di progettazione per la funzionalità di time stretching;
- il capitolo 2 presenta le principali tecniche di time stretching nel domino del tempo e in tempo-frequenza;
- il capitolo 3 tratta dettagliatamente lo stato dell'arte del phase vocoder e delle tecniche per migliorarne i risultati: phase locking, transient detection e preservation;
- il capitolo 4 discute i contributi originali di questo lavoro. Qui le tecniche di phase locking proposte, lo schema di transient/detection preservation ed altri miglioramenti del phase vocoder sono trattati nel dettaglio. Vengono inoltre presentati una realizzazione real-time dell'algoritmo proposto e i risultati dei test d'ascolto.

Acknowledgments

I would like to thank Gunter Geiger and Marcos Alonso for giving me the great opportunity to work at Reactable Systems. I wish also to express my gratitude to Professor Federico Avanzini for the possibility to pursue my thesis work in Barcelona, and for his precious help during the writing of this essay.

A special thanks goes also to Alex, Nadine, Carlos, Bego for the really nice time spent together in the office. I am particularly grateful to Andres, Mathieu, John and Thodoris for all the brilliant suggestions and constant feedbacks.

I wish to thank Jordi Bonada for the feedbacks and for the stereo image preservation tips, and Fabio Tesser for his help with the listening tests and for all the suggestions.

I am extremely grateful to my family for the economical and moral support, without which I could have never accomplished this work. In particular, a special thanks goes to Laura and Gabriele for their invaluable help with my English prose. Last but not least, I would like to thank Roberta and Aikia for all the funny moments together.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work took place in the context of Erasmus LLP european project, from april to september 2012. As intern student in Reactable Systems S.L. my task was to integrate into *Reactable Live!* software a time stretching algorithm. A time stretching algorithm is an algorithm designed to alter the length of an audio signal preserving the pitch and the timbre of the original sound source.

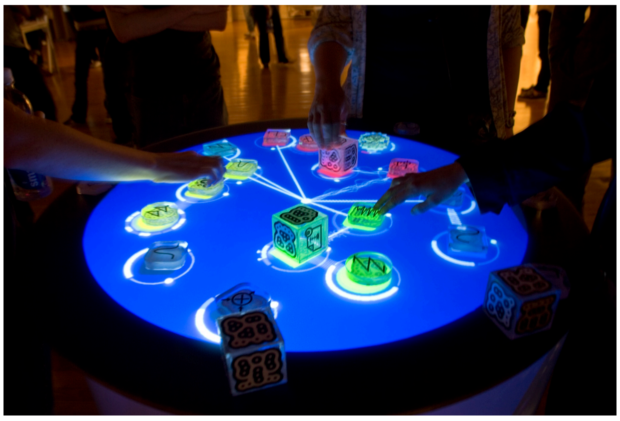


Figure 1.1: *Reactable Live!*

1.1 Reactable

The Reactable is an electronic musical instrument with a tabletop Tangible User Interface that has been developed within the Music Technology Group at the Universitat Pompeu Fabra in Barcelona, Spain by Sergi Jordà, Marcos Alonso, Martin Kaltenbrunner and Günter Geiger.

The Reactable is a round translucent table, used in a darkened room, and appears as a backlit display. By placing blocks called *tangibles* on the table, and interfacing with the visual display via the tangibles or fingertips, creating music or sound effects.

Reactable Systems produces three different Reactable models:

Reactable Experience : a limited feature but very solid version of Reactable targeting museums and educational exhibitions;

Reactable Live! : full feature portable version of Reactable targeting live music performances;

Reactable by Mos : luxury version of Reactable Live!

A virtual Reactable application for Android and Ios smartphone/tablet is produced too.

Basically Reactable's functional scheme (figure 1.2) is the following: a camera and a projector work as input/output interface, sending and receiving data to and from a computer, which sends to the audio intefrace the digital sound signal.

The camera (input interface) recognizes tangible objects and users control gestures. Through projection performance feedbacks, virtual setting panels and visual effects appear on the translucent surface.

The computer elaborates data from the camera and sends feedback data to the projector.

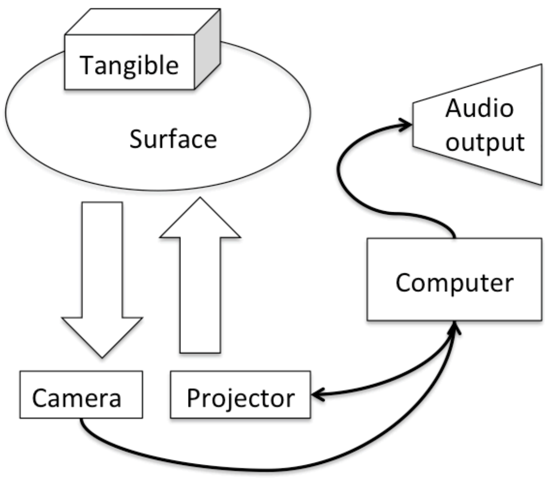


Figure 1.2: *Reactable's functional scheme.*

1.1.1 Tangibles

A tangible object is basically a polyhedron or a cylinder, with one or more faces painted in a way that the computer, through a GPL computer vision library called *reacTIVision*, can recognize not only the position of object on the table, but also its rotation.

Each tangible represents a module, which can be a sound generator, an audio effect or a controller. User can connect together different tangibles by placing them in an appropriate way on the surface, and he can control modules parameters both rotating the tangibles or with a virtual touch interface achieved by projection and fingers detection.

There are four categories of tangible objects:

Generators Generators generate sound. A generator could be an oscillator, a loop player or a synthesizer objects as well as sound input through a soundcard. The generators have in common that they generate sound by themselves and do not need to be connected to any other object in order to be heard. When put on the table they will connect automatically to the output or to effects and filters.

Effects Effects objects need sound input from a generator in order to apply some audio effect (e.g. reverb, delay, filter, distortion etc...) to the input sound.

Controllers Controllers send control data to effects and generator. There are two type of controller: the *Sequencer* and the *Low Frequency Oscillator* (LFO). While sequencers emit note events and trigger the envelopes of other objects, LFOs emit a continuous stream of control data and allow to modulate parameters continuously.

Global Controllers Global controllers do not connect to any other object. They change the behavior of the whole Reactable application, like global tempo, volume or tonality.

1.1.2 Temporal parameters

Tangible temporal parameters depends on the global parameter, which are controlled by the global controllers.

The global time parameter is the *tempo*, that is the conventional way to represent pace in Western music. Tempo is expressed in beats per minute *bpm*, that is defined as number of the basic time units (beats) in a minute. The beat is indicated by the time signature of the music, and usually is the quarter note (♩). The bar (or measure) is a segment of time defined by a given number of beats. Through the *Global Tempo* user can change the tempo and the bar-length of Reactable application.

Most of tangible temporal parameters are expressed as note-length, so, when user change the tempo through the Global Tempo, they are re-computed real-time in order to achieve an automatic synchronism.

Modify the temporal parameters of an event sequence or of an audio effect such a delay time, is straightforward. Problems arise when changing the temporal parameters of a pre-recorded piece of audio.

1.1.3 The loop object

The loop object is a loop player, i.e. an object that plays in loop a pre-recorded piece of audio.

Its tangible is a cube, and it is a 5 banks $\times$ 4 loops. Five of the six faces of the cube map the loop banks, while the rotation angle maps the loop position. User can switch between loops in the same bank rotating the cube on the surface, and can choose which bank to use by placing the cube on the table with the right face down. For each loop position user can load a sound file through a virtual touch panel.

Once a loop is loaded its tempo is computed through a simple algorithm that basically segments the input signal in beats in order to make the loop-length fit as much as possible to some integer multiple of the bar-length. If the loop's tempo is not equal to the global tempo, because the length of the loop is not a multiple integer of the global bar length or because the user change the global tempo, the loop need to be time-scaled in order to be played synchronously.

1.1.4 Motivation for the integration of a time-stretching functionality

Before the work described in this thesis was implemented into Reactable software, loops time-scaling was achieved by variable speed replay that is basically a mathematical time-scaling of the sound signal (i.e. playing the loop faster or slower), but as we will see in the next chapter this leads to alter the pitch and the timbre of the original sound.

Reactable provides a nice and easy synchronization mechanism that allows users not to worry about keeping relationships between global parameters and tangibles parameter except loops. In fact, users must be very careful when playing loops:

- if a user wants to play at the same time two loops that have different lengths, the result can be an annoying detuning effect;
- if a user wants to play a loop with a different tempo from the one which was recorded, the pitch and the timbre are going to be altered.

For this reasons many Reactable users were demanding for a mechanism that can keep the time synchronization without altering the loops pitch.

This can be achieved trough a time-stretching technique, which is basically a technique that computes an output track from an input audio track, where the same notes start in time-scaled positions, with durations proportional to the scaling factor.

1.2 Requirements for Reactable time-stretching implementation

Many different time-stretching algorithms and implementations are present both in market and in literature, each one with different computational complexity, quality and targeting. Some of them work fully real-time, some need a pre-processing routine in order to work on-line, and others work off-line.

In this section design requirements for the time-stretching implementation into Reactable will be analyzed.

1.2.1 Interaction requirements

Since Reactable strength is the intuitive and easy-to-use interface, time stretching functionality needs to be easy to use too. For this reason:

- it has to work for any source without needing user settable parameters;

- when a user loads a loop it has to be reproduced immediately without waiting some pre-processing routine;
- whenever a user changes the tempo of Reactable application, or switch between different sound files, the loop has to be reproduced synchronously to global timing parameters.

1.2.2 Real-time requirements

Time-stretching must work for six stereo loops playing, along with all Reactable tasks. Because we didn't want to introduce much audio latency, computational peaks must be avoided.

In fact time-stretching has to work with a sample rate of 48000 Hz and a latency buffer shorter than 5 ms.

1.2.3 Quality

Time-stretching should have a good live-quality. Since Reactable is a powerful live instrument and not a production software, some quality needs to be sacrificed in order to avoid jitters, glitches and application freezings with a certain safe margin.

Trade-off computational complexity/quality should be a user settable parameter in order to:

- perform real-time time-stretching without any problem on slower and older computers;
- achieve higher quality time-stretching on newer machines.

The quality should be proportional to the stretching factor: when the stretching factor is close to one the original sound and the stretched one should be ideally indistinguishable, while for larger stretching factors it is important not to introduce annoying artifacts.

1.2.4 Targeting

Reactable targets electronic music, and since electronic music exploits audio taken from any source, the time-stretching functionality should target any sound source. However we can target the quality for different stretching factors and different sound sources.

In fact Reactable users compose music by playing together different loops that could be mixes of different sound sources but rarely a complete produced song. Furthermore the detuning effect caused by variable speed replay is very annoying for pitched portions of sound while less for un-pitched ones.

In particular we can classify different kind of loops in four sets:

Textures : non-percussive sounds. Textures can be recordings of polyphonic or monophonic instrument (e.g. a synth, a bass guitar, or human voice) , or a mix of two or more non-percussive sounds (e.g. a saxophone and a trumpet, a choir).

Beats : percussive sounds. Beats can be recordings of drums or pitched percussive instrument (e.g. a glockenspiel).

Simple mixes : mixes of beats and textures, with two or three different sources that are clearly distinguishable (e.g. a drum and a bass, a hi-hat and a saxophone) and do not heavily influence each other behavior in the mix signal.

Complex mixes : mixes of beats and textures, with more than two sound sources. Complex mixes peculiarity is that, due to production processing, each source heavily influence each other behavior in the mix signal. For instance in a complete song, due to dynamical processing, a sound source energy can modulate the volume of other sound sources.

This partitioning is clearly not operational, since the distinction between two different sets is subtle, fuzzy and sometime subjective. However in the design process it was sufficient to targeting the quality of the time-stretching.

In particular:

- good quality is required for complex mixes for stretching factors in the interval $-10/+10\%$;
- high quality is required for textures, beats and simple mixes in the interval $-15/+15\%$;
- for large stretching factor textures and beats should be privileged ;
- extreme stretching inevitably introduces artifacts that should not be annoying for textures and beats, providing a sort of nice “creative” effect when slowing down the tempo (*graceful degradation*).

1.3 Real-time commercial time-stretching libraries

The first step in the integration of the time-stretching functionality into Reactable was to test and compare some commercial time-stretching algorithms, in order to decide if it was more convenient to develop and implement a new algorithm, or just integrate some commercial library.

In the market there are many high quality time-stretching libraries as Serrato’s *Pitch’n’Time*, Izotope’s*Radius*, but since they are not designed to work real-time they have not been taken into account. There are also real-time GPL library as *Rubber Band* and *Sound Touch*, but their quality was considered too low.

This section presents the two real-time time-stretching library that have been really taken into account before deciding to implement a new algorithm for Reactable.

1.3.1 Zplane’s *Elastique*

Elastique is probably the most in-fashion time-stretching/pitch-shifting library. It is used in renowned audio software as Native Instrument *Traktor*© and in Ableton *Live!*©. Elastique library offers three different algorithms:

Soloist : low-computational complexity algorithm targeting monophonic sound sources as voice.

Efficient : low-computational complexity algorithm targeting polyphonic sound sources.

Pro : higher computational complexity algorithm, targeting any sound source.

Elastique is a high-quality real-time time-stretching library for relatively small stretching factors, however for larger stretching it shows evident artifacts as eco, phasiness and transient smearing.[1]

1.3.2 DSP Dimension's *DIRAC*

DIRAC is integrated in many high quality waves editor, as Steinberg Wave Lab© and Prosoniq Time Factory©. It targets every sound source and, in its real-time version, the only user settable parameter is the quality. It works real-time and the computational workload is, as expected, proportional to quality.

1.4 Motivation for the development of a new time stretching algorithm

During the preliminary evaluation on commercial time stretching library DIRAC was considered the best choice, in fact DIRAC fits all Reactable quality and targeting requirements.

Unfortunately DIRAC time-stretching requires to analyze the input about 100 ms in advance before computing the output. This causes a computational peak when skipping through different files or different file positions, causing possible glitches and jitters.

Since Reactable users are free to skip between files in any moment, and the synchronization methods requires skipping through different file positions in a very small interval (1.5 ms in the worst case), it's integration was difficult and dangerous.

The only way to integrate safely DIRAC library into Reactable software was to impose a larger audio latency buffer to Reactable application, in order to "smear" the computational peak in the time, but it was not considered appropriate since the latency buffer should be small for interaction issues.

The only Elastique algorithm that could have been integrated in Reactable is Elastique *Pro*, since it targets any sound source. Elastique Pro works very well for most of audio sources, but for some sources it introduces minor artifacts also for relatively small stretching factors. In particular, for very dynamic sound sources it tends to attenuate sharp transients, and makes them sound unnatural.

After a preliminary and simple implementation of the algorithm proposed in this thesis, it was decided to keep developing it since the results, both from quality and computational point of view, were promising.

The goal was not to develop a time-stretching that outperforms DIRAC and Elastique, but to achieve a similar quality with an ad hoc implementation that fits all Reactable requirements.

[1]Phasiness and transient smearing will be exhaustively discussed in chapter 3.

Chapter 2

Time Scaling Techniques

Imagine a music producer who wants to mix together different audio tracks: unless the tracks have the same *tempo*, he or she has to *time scale* them, i.e. change their duration.

This could be achieved very easily by time scaling the original audio signals but, from a musical point view, a time scaled version of an audio track is not a mathematical time scaled version of the original audio signal.

Formally, an α time scaled version of an audio track is a track where the same notes start in time scaled position $t' = t\alpha$, with durations proportional to α. We will refer to α as *stretching factor*.

In order to achieve this result several *time-stretching* techniques have been developed since 1950's [16] in both analog and digital domain, but they have become popular only with the advent of digital audio era. In this chapter basic time stretching techniques will be introduced.

2.1 Variable speed replay

Variable speed replay is the easiest way to time scale an audio track and it simply consists in the time scaling of the original audio signal.

Suppose we want to obtain signal y as the α time scaled version of the signal x. In the analog domain this could be simply achieved by:

$$y(t) = x(t/\alpha). \tag{2.1}$$

The notes starting position and duration are perfectly scaled, however their pitch is scaled too of a factor $1/\alpha$. As a matter of fact, in the Fourier Transform:

$$Y(f) = \int x(t/\alpha)e^{(-2\pi itf)}dt = \alpha \int x(t')e^{(-2\pi it'\alpha)}dt' = \alpha X(\alpha f). \tag{2.2}$$

In the digital domain, the relationship between input and output signal is more complex and corresponds to sample rate conversion, however the frequency relationship is very similar to the one in the analog domain.

In literature many different sample rate conversion techniques exist, and the quality of variable speed reply depends very much on the used one [6]. An overview of them can be found in [33].

2.2 Time segment based algorithms

The basic idea behind time domain time stretching algorithms is to divide the input in small overlapping segments, and then rearrange them to fit the desired sound length.

Time domain time stretching algorithms have been used since 1950's in the analog domain and they have been implemented trough modified tape recorders [16]. The most famous example of an analog device who implements time stretching is the *Phonogène Universal*, developed by Pierre Schaffer and Jacques Poullin between 1959 and 1963 [48].

2.2.1 Synchronous overlap and add algorithm

The first time segment domain algorithms is the Synchronous Overlap and Add (SOLA). The idea is to segment the input in overlapping frames with a certain analysis hop size, and then overlap and add them with a different synthesis hop size.

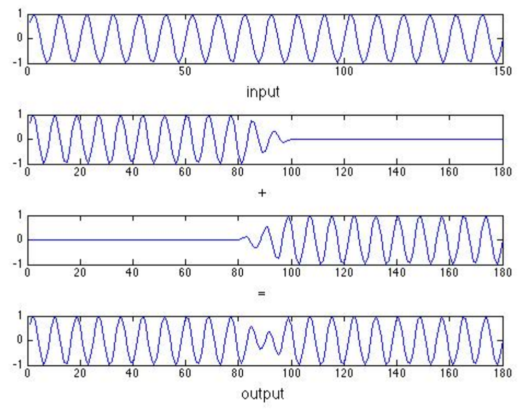


Figure 2.1: *Example of phase discontinuities with basic overlap add.*

The basic overlap and add scheme is the following:

- Segment the input in n length frames, with a hop size hop_a;
- compute the synthesis hop size hop_s as $hop_s = hop_a \times \alpha$, where α is the time stretching factor;
- overlap the segments with hop_s hop size, fade in and out them in order to achieve constant envelope;

- compute the output signal summing the faded overlapping portions of frames.

This very low complexity method leads to some unpleasant phase discontinuity artifacts. An example of phase discontinuity is shown in figure 2.1.

To avoid the phase discontinuities of overlap and add, a common approach is to choose the relative overlapping position of two frames (in the input or output) in order to sum portions of signal that are "similar". This can be achieved by shifting one frame of a time lag k that maximize a similarity function, and the most common one is the normalized cross correlation [16] [51].

The cross correlation function, in a L length interval is defined as:

$$r_{x,y}(k) = \begin{cases} \frac{\sum_{i=0}^{L-k-1} x[i]y[i+k]}{\sum_{i=0}^{L} x^2[i] \sum_{i=0}^{L} y[i+k]} & \text{if } 0 < k \leq L-1 \\ r_{y,x}(-k) & -L+1 < k < 0 \end{cases} \tag{2.3}$$

In SOLA algorithm (figure 2.2), just before fading and overlapping the input frame to the already computed output, the overlap position of the input frame is shifted by the k time lag that maximize the cross correlation between the frame and the output in a δ length interval around the overlap and add position.

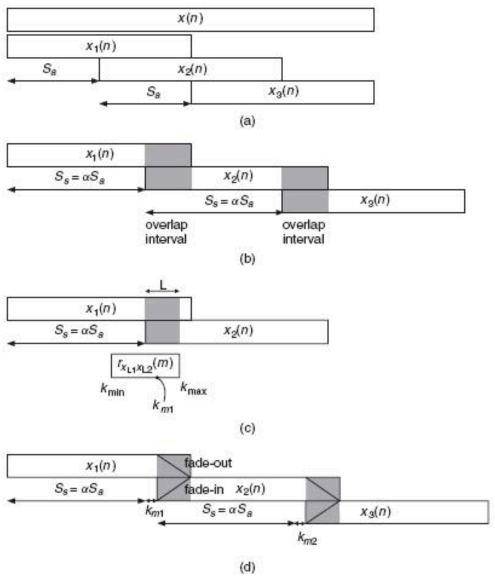


Figure 2.2: *SOLA algortihm.*

Even though this method avoids phase discontinuities, it has a major draw-back: in case of pitched sounds it can misalign pitch peaks causing pitch discontinuities artifacts that can be reduced using rather long windows (between 40 and 80 ms [14] [51]).

2.2.1.1 WSOLA algorithm

WSOLA is a variant of SOLA algorithm that usse wave similarity measures to segment the input. In fact instead of changing the synthesis hop size to avoid phase discontinuities, it uses a constant synthesis hop size, while the analysis hop size is adjusted by a k time lag in order to maximize the wave similarity between the time scaled signal and the original input in the corresponding time scaled position. This approach leads to better results with lower computational complexity than traditional SOLA [51].

2.2.1.2 Other SOLA variants

In the literature different time stretching algorithms based on correlation methods exist, that can be seen as improvement of SOLA algorithm.

SOLAFS [23] chooses the input frames in order to maximize their cross correlation with the already computed output. Dorran and Lawlor in [14] present an algorithm called VSOLA, that is a sub-band version of the SOLA algorithm.

2.2.2 Pitch Synchronous Overlap and Add

The aim of Pitch Synchronous Overlap and Add (*PSOLA*)[8] [21] is to time scale the signal by repeating/discarding some input frame, using the knowledge of pitch to correctly synchronize time segments.

PSOLA algorithms is based on the hypothesis that the sound is characterized by a pitch, as monophonic instruments or human voice.

If the local pitch of signal at time t is $P_{input}(t)$, and we want to time scale the input by a factor α, then at time $t' = t/\alpha$ the time scaled output should have $P_{output}(t') = P_{input}(t)$. To achieve that result PSOLA is composed of two phases: the analysis and the synthesis phase. Very briefly:

Analysis Determination of the pitch period $P(t)$ of the input signal and of time instants (pitch marks) t_i correspondents to maximum amplitude of signal at pitch-synchronous rate.
Extraction of segment centered at t_i using a smoothing window of length dependent on $P(t)$.

Synthesis Overlap and add the extracted frames. If the time stretching factor $\alpha > 1$ some frame will be repeated, if $\alpha < 1$ some frame will be discarded.

The determination of the pitch period is a non-trivial problem and can be performed in both frequency or time domain [50]. An efficient and widely used pitch tracking algorithm that works in time domain is the *Yin* algorithm [10].

PSOLA can lead to very good results for pitched portions of sounds, but additional procedures to treat un-pitched and fast changing regions of the signal [16] are needed in order to achieve high-quality time scaling.

2.3 Time-frequency domain algorithms

In this section we will introduce basics of time stretching algorithm that rely on the time-frequency representation of the signal, i.e. on the short-time Fourier Transform.

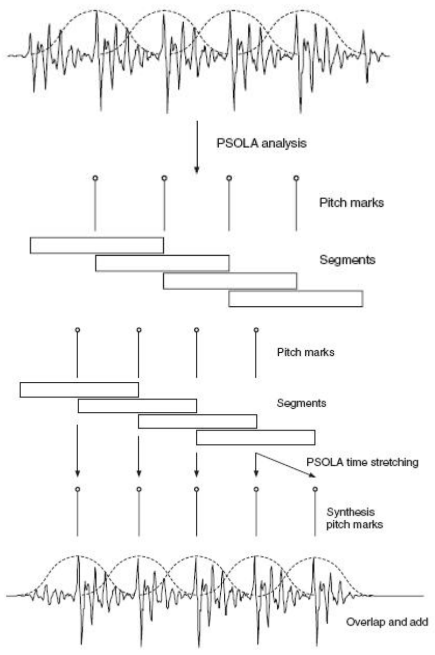


Figure 2.3: *PSOLA Analysis/Synthesis scheme*

2.3.1 Short-Time Fourier Transform

The Short-Time Fourier Transform (*STFT*) is a well known concept, widely described in literature and one of the basic concept in the field of digital signal processing. For these reasons we will not spend much time on it (a detailed description of STFT technique can be found in [33]).

STFT is basically performed computing the Discrete Fourier Transform (DFT) of windowed portions (frames) of the input signal starting at different time positions. For the sake of clarity, the DFT of the signal is computed with an algorithm called Fast Fourier Transform (FFT), however in literature FFT is widely used also to indicate the DFT. Here DFT and FFT will be used to indicate respectively the transform and the algorithm.

Formally, the STFT of a signal $x(n)$ is:

$$X(n,k) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)e^{-j\frac{2\pi mk}{N}}, k = 0, 1, ..., N-1 \tag{2.4}$$

where $X(n,k)$ is the time-varying spectrum with the frequency bin k, at the time index n. At each time index x is weighted by a N-length window $h(n)$. $X(n,k)$ is a complex number, and can be expressed as magnitude $|X(n,k)|$ and phase $\angle X(n,k)$. If $x(n)$ is a real signal, its STFT at each time index is *Hermitian Symmetric* to $N/2$, i.e. the real part is even, while the imaginary part is odd.

We will call the spectrum at one time instant $[X(n,0), X(n,1)..., X(n,N)]$ as *STFT frame*.

The STFT suffers of two major problems. The first one is *spectral leakage*: multiplication in time domain correspond to convolution in the frequency domain, so it's useful to choose a smoothing window as $h(n)$ in order to reduce ripples in the STFT frames. However, weighting the input with a smoothing window inevitably brings to loose some frequency resolution. A good way to reduce the ripples in the STFT without loosing too much frequency resolution is to zero-pad each portions of input signal before computing the STFT.

The second problem is about resolution. Briefly, if we choose for N a large value we will achieve a good resolution in frequency domain, but, while performing some analysis on the STFT we wont be able to map event in the frequency domain to time domain with good resolution and vice-versa. In order to reduce this problem the STFT is computed on overlapping frames.

At the same time, roughly speaking, a sinusoid with frequency corresponding to the first STFT's bin has a period of exact N samples, while a sinusoid corresponding to the $N/2$ bin has a period of only two samples. It follows that to have a good trade off between time and frequency resolution we need large value of N for low frequencies, while smaller for higher frequencies.

2.3.2 Phase vocoder

The *phase vocoder* (figure 2.4) is a very old technique to process signals in the time-frequency domain, and it was invented in the *Bell Labs* in 1966[18]. However, thanks to the increase in computational power (Moore's law) and to many improvements to it, the Phase Vocoder has become a popular and attractive technique to perform time-scale modifications only in the last two decades.

The phase vocoder is not a specific algorithm to time-scale audio signals, it can be viewed as a time-frequency analysis/synthesis framework. It finds application in filtering, pitch shifting, time scaling and many other "exotic" [27] effects.

The phase vocoder was invented only one year after the publication of Cooley-Tukey algorithm [9], and in its first formulation time-frequency analysis/synthesis was performed by banks of filter and oscillators (*filter bank summation model*).

Ten years after [37] phase vocoder was implemented directly in the time-frequency domain using the FFT algorithm, and this formulation is called *block-by-block analysis/synthesis model*.

The later formulation is much flexible and computationally efficient, so we will treat only the block-by-block analysis/synthesis model.

2.3.2.1 Phase vocoder basics

The phase vocoder, as mentioned before is a time-frequency domain analysis/synthesis framework. Basically the analysis is performed computing the STFT of the input with a certain analysis hop size; the synthesis is performed, after some spectral modification, by overlapping and adding the Inverse Discrete Fourier Transform (IDFT) of the modified STFT frames with a certain synthesis hop size.

The analysis gives us:

$$X(sR_a,k) = \sum_{m=-\infty}^{\infty} x(m)h(sR_a - m)e^{-j\frac{2\pi mk}{N}} = |X(sR_a,k)|e^{\angle X(sR_a,k)}. \tag{2.5}$$

where R_a is the analysis hop size, s the time index and h is the analysis window. After some modification of STFT frames, we obtain the frames $Y(sR_s, k)$, where R_s is the synthesis hop size. The inverse discrete Fourier transform (IDFT) of a modified FFT frame is given by:

$$y_s(n) = \frac{1}{N} \sum_{k=0}^{k=N-1} e^{j\frac{2\pi(n+sR_s)k}{N}} Y(sR_s, k) \tag{2.6}$$

and then the output $y(n)$ is obtained by overlapping and adding the inverse Fourier transform:

$$y(n) = \sum_{-\infty}^{\infty} f(n - sR_s) y_s(n - R_s), \tag{2.7}$$

where f is the synthesis window.

To achieve a zero-phase analysis and synthesis regarding the center of the window, it is necessary to perform a circular shift of the segment before computing the FFT of the input frame [11]. The effect of this circular shift must be taken into account during the synthesis phase, performing another circular shift after the computation of the IFFT of modified STFT frame.

Modifying STFT frames could lead to discontinuities artifacts during the synthesis phase, and furthermore there is some time aliasing due to the circular aspect of convolution. For this reason is necessary to apply a synthesis window before the overlap and add (window *tapering*).

2.3.2.2 Time scale modification using the phase vocoder

Time scale modification using the phase vocoder is performed through phase interpolation.

The idea behind is very intuitive and simple: if we calculate the STFT of the input signal with an analysis hop size R_a, we can perform time scaling using a different synthesis hop size $R_s = R_a \times \alpha$, where alpha is the stretching factor, and modifying the phase in order in such a way that the instantaneous frequencies are preserved.

Consider a stable sinusoid (constant amplitude, constant frequency) at time index t with frequency ω and phase $\phi(t)$. At time index $(t + \delta)$, under the stability hypothesis, the phase is $\phi(t + \delta) = \phi(t) + \omega \times \delta$.

It follows that to time scale a stable sinusoid, if we know the initial phase $\phi(0)$, at time index sR_s, the only spectral transformation we have to do is to assign to the corresponding STFT bin the phase $\varphi(sR_s) = \phi(0) + \omega \times sR_s$.

Unfortunately, in real audio signal the stability hypothesis is not valid, but we can take in account amplitude and phase deviation.

Consider the k-th STFT bin at time index sR_a, with phase $\phi_k(sR_a)$, and at time $(s + 1)R_a$ with phase $\phi_k((s + 1)R_a)$. The phase deviates from the one we should expect in case of stable sinusoid of:

$$\Delta_{\phi_k}(s + 1) = \phi_k((s + 1)R_a) - (\phi_k(sR_a) + \omega_k \times R_a). \tag{2.8}$$

where $\Delta_{\phi_k}(s+1)$ is called phase deviation and ω_k is the frequency correspondent to the k-th bin:

$$\omega_k = \frac{2\pi k}{n} \tag{2.9}$$

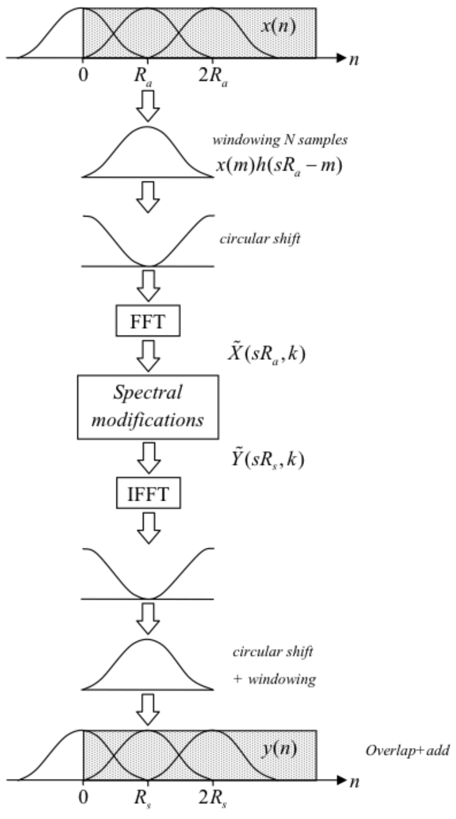


Figure 2.4: *Block-by-block phase vocoder using FFT/IFFT*

with n the FFT length.

However, considering that the phase computed from the FFT assumes values in $(-\pi, \pi]$, we need to put the phase deviation in that interval. This is achieved through the princarg function, defined:

$$\text{princarg}(x + 2\pi m) = x, \quad -\pi < x \leq \pi, m \in Z. \tag{2.10}$$

We can now define the *unwrapped phase difference* as:

$$\omega_k R_a + \text{princarg}(\Delta_{\phi_k}(s+1)). \tag{2.11}$$

Now consider the time scaled version of the signal. If the modified STFT bin at time index sR_s has phase $\varphi_k(sR_s)$, we can compute $\varphi_k((s+1)R_s)$ as the sum of the target phase (the phase the bin would assume in the case of a stable sinusoid) plus a deviation term:

$$\varphi_k((s+1)R_s) = \varphi_k(sR_s) + \omega_k R_s + \text{princarg}(\Delta_{\phi_k}(s+1))\frac{R_s}{R_a}. \tag{2.12}$$

φ is called *unwrapped phase*, and the process to compute it is called *phase uwrapping*.

this ensure that:

$$\Delta_{\phi_k(sR_a)} = \Delta_{\varphi_k(sR_s)} \frac{R_s}{R_a} \tag{2.13}$$

i.e. the instantaneous frequency is preserved.

From this results follows that time scaling with phase vocoder is achieved computing the phase of each synthesis STFT bin through equation 2.12, while keeping the same magnitude value for analysis and synthesis stage.

2.3.2.3 Phase vocoder problems

Phase vocoder suffers of two major problems that will be discussed in detail in the next chapter.

The first major drawback is called *phase dispersion*. The phase unwrapping gives a phase equal to the measured phase plus a term that is a multiple of 2π that is not the same for all bins. Once multiplied for a non integer time stretching factor, this produces phase dispersion i.e. the phase difference between bins in the synthesis phase it's different from the measured one.

The artifact caused by phase dispersion is called *phasiness*, and basically it is due to the fact that phase dispersion introduces beatings[1] between adjacent channels causing a sort of reverberant effect.

The second problem is called *transient smearing*. Phase unwrapping is valid under the assumption that signal is nearly stationary. If the amplitude of a sinusoid change abruptly (a situation normally denoted as attack transient) the nearly stationarity hypothesis is no more valid. Time stretching attack transients with the phase vocoder results not only in softening the perceived attack, but in a complete change in the sound characteristics.

Several techniques to reduce phasiness and transient smearing have been introduced in literature, and they will be presented in the next chapter.

2.3.3 Sinusoidal modelling

Sinusoidal models, originally proposed in [31], extend the STFT framework by presenting higher-level modeling of the spectral data obtained with it.

A sinusoidal model explicitly represents the sound signal $s(t)$ as a sum of slow varying sinusoids:

$$s(t) = \sum_{r=1}^{R} A_r(t) cos[\Phi_r(t)] \tag{2.14}$$

where $A_r(t)$ and $\Phi_r(t)$ are the instantaneous amplitude and phase of the rth sinusoid, and R is the number of sinusoids.

A sinusoidal model of the sound signal is achieved through iterative analysis of the STFT, and synthesis is performed by additive synthesis. Time scaling with a sinusoidal model simply consist in time scaling the parameter of the built model.

[1]In acoustics, a beat is an interference between two sounds of slightly different frequencies, perceived as periodic variations in amplitude whose rate is the difference between the two frequencies

Sinusoidal models are a very powerful tool to build an high level model of a sound source but, because a sound signal is not only composed by slow varying sinusoids, they need to be extended in order to achieve an high synthesis quality. In fact sound is composed by partials and residual. Partials are sinusoidal components that usually correspond to modes of vibration of the producing sound system, while residual is the difference between the signal and the estimated partials.

To take into account both partials and residuals, in 1989 Xavier Serra [46] introduced a signal model in which the sound is decomposed in Deterministic (partials) and Stochastic (residual) components. In this model, the analysis consists in the identification in the STFT of partials and residual. The synthesis can than be performed through additive synthesis, representing partials, plus filtered white noise that approximates the residual.

Another improvement was introduced by Verma et al. in 1998 [52], adding to Serra's work an additional explicit model for attack transients.

2.4 Conclusion

In this chapter time domain and time-frequency domain time scaling techniques have been analyzed. Time segment based stretching algorithms are particularly attractive because have low computational complexity.

As already stated, PSOLA does not work with polyphonic signal, and SOLA also is less adequate for polyphonic music. This is due to the fact that the necessary synchronization between overlapping frames can be achieved ensuring proper phase relation in case of a monophonic source, while not with a polyphonic one, especially if the source does not exhibit strong quasi periodic elements [19].

The most promising time segment based algorithm is WSOLA. It produces really high quality stretching for factor close to one, otherwise it causes unpleasant reverberation artifacts because treats in the same way both stationary and non stationary portions of sound. Special measure to treat differently non stationary portions of the sound from stationary are needed, like in [19] and [36], at the cost of performing pre-analysis or to reach a real-time computational complexity similar to the phase vocoder one.

Time frequency domain algorithms exhibit higher computational complexity, but are more adequate for polyphonic music.

Time-scaling with sinusoidal models leads to very good results for extreme stretching factors, however, the synthesized signal sounds "different" from the original one also for stretching factor one (no time scaling), and the real-time construction of the model is computationally too heavy to be efficiently implemented in Reactable software.

Phase vocoder was considered the best choice, since it is the most adequate technique for polyphonic signals, and from the phase vocoder analysis data it is possible, at relatively low computational costs, to detect real-time local features of the sound signal in order to improve results. Furthermore, in phase vocoder analysis/synthesis scheme it is possible to include useful results from sinusoidal models, without needing to build an accurate model of the sound signal to be time-scaled.

Chapter 3

Phase Vocoder Improvements

3.1 Introduction

In the previous chapter the phase vocoder was briefly introduced, along with phasiness and transient smearing. Unfortunately phase vocoder suffers also of some other problems, that will be discussed below.

In the following chapter we will analyze in detail all phase vocoder issues, along with the state of art of techniques to treat them.

3.2 Phasiness

As introduced in the previous chapter, time scaling with phase vocoder is achieved using a synthesis hop size R_s equal to the analysis hop size times the stretching factor ($R_s = R_a \times \alpha$), and performing phase unwrapping:

$$\begin{aligned}\angle Y((s+1)R_s, k) &= \\ = \angle Y(sR_s, k) + \omega_k R_s + \text{princarg}(\angle X((s+1)R_a, k) - \angle X(sR_a, k) - \omega_k \times R_a)\frac{R_s}{R_a} &= \\ = \angle Y(sR_s, k) + [\omega_k R_a + \text{princarg}(\angle X((s+1)R_a, k) - \angle X(sR_a, k) - \omega_k \times R_a)]\frac{R_s}{R_a}&. \quad (3.1)\end{aligned}$$

where X and Y denote the STFT of the input and the output.

Phase unwrapping guarantees phase consistency from frame to frame (*horizontal coherence*), but not within a given frame between neighboring bins (*vertical coherence*).

In fact phase unwrapping returns a phase that is equal to the measured phase plus a term that is a multiple of 2π. As far as the second term is not the same for every STFT bin, there is phase dispersion if the stretching factor $\frac{R_s}{R_a}$ is not integer, i.e. the phase distance between synthesis STFT bin is different from the analysis.

Unfortunately, for integer stretching ratio vertical phase coherence can be lost too [25]. Consider a sinusoid starting at time 0 that migrates from bin k_0 to $k_1 = k + r$. The phase equation can be re-written:

$$\varphi(sR_s, k_0) = \varphi(0, k_0) + [\phi_k((s)R_a, k_0) - \phi_k(0, k_0)]\alpha \tag{3.2}$$

where the unwrapping can been discarded because the $2l\pi$ term is multiplied by an integer factor.

When the sinusoid falls in the bin k_1 at time t, the synthesis phase becomes:

$$\varphi(tR_s, k_1) = \varphi(0, k_1) + [\phi_k((s)R_a, k_1) - \phi_k(0, k_1)]\alpha \tag{3.3}$$

$\phi_k(0, k_1)$ and $\phi_k(0, k_0)$ can hold different values, because the initial phase value of bin k_1 can be influenced by another sinusoid, or simply because they are distant.

So, when the sinusoid migrates from bin k_0 to bin k_1 theres a jump in the phase of $\theta_{k_1} - \theta_{k_0}$, where:

$$\theta_k = \varphi(0, k) - \alpha\phi(0, k).$$

However, if $\theta_k = C, \forall k \in N$, the synthesis phase become:

$$\varphi(tRs, k) = C + \alpha \times \phi(tR_a, k) \tag{3.4}$$

that is consistent with the ideal synthesis phase.

Phase dispersion introduces artificial beatings between adjacent bins and alter the phase distance between partials of the same note, causing an annoying sort of reverberant artifact named phasiness, that is the major drawback of phase vocoder.

3.3 Phase-locked phase vocoder

In order to reduce the phasiness several techniques have been proposed, but it is still subject of research [6]. The most common one consists in "locking" (constraining) the phase between adjacent bins (*phase-locking*) in order to reduce artificial beatings.

3.3.1 Loose phase-locking

The concept of phase-locked phase vocoder was introduced in 1995 by M. Puckette [38].

Recognizing that a constant-amplitude, constant- frequency sinusoid in channel should have identical analysis phases in all nearby bins, Puckette proposed a new simple and computational efficient way to lock synthesis phase.

If $Y(t, k)$ is the k-th bin of traditional phase vocoder synthesis bin at time index t, rather than using that value for synthesis loose phase-locked vocoder use:

$$Y'(t, k) = Y(t, k) + Y(t, k-1) + Y(t, k+1). \tag{3.5}$$

As a result, the phase of bin k is basically unchanged if bin k is a local maximum of the STFT frame, because $Y(t, k-1)$ and $Y(t, k+1)$ have much lower amplitude; but if k is a left or right maximum, its phase will be close respectively to the one of $Y(t, k-1)$ or $Y(t, k+1)$.

This technique is very attractive because it requires only a few additional operation per bin, and because Puckette's phase locked vocoder can be implemented without using any trigonometric function. Unfortunately reduction in phasiness is signal dependent and never dramatic [26].

3.3.2 Rigid phase-locking

In 1997 Laroche and Dolson [25] proposed two new techniques to lock the synthesis phase of phase vocoder: *identity phase locking* and *scaled-phase locking*. Both those techniques, inspired by loose phase locking [26], rely on the same idea: locking the phase around the peaks of the STFT frame. This choice can been seen as a step closer to the sinusoidal model, where the sinusoids are represented by STFT peaks.

Rigid phase locking begins with a simple peak-picking stage, where a bin is marked as peak if its amplitude is larger than its four neighbor. The series of peaks subdivides the frequency axis into *regions of influence* located around each peak.

The idea is to compute the unwrapped phase only for peaks bin, and for the remaining bins in the same region *locking* the phase to the phase of peak bin.

In [26] the proposed way to compute the upper limit of region of influence is to set it to the middle frequency between the peak and the next one, or in correspondence to the lowest amplitude bin between two peaks.

3.3.3 Identity phase locking

Assuming that a STFT frame peak is the trace of a constant amplitude constant frequency sinusoid, the synthesis phases are constrained around the peak in order to be related in the same way as in the analysis: the synthesis phase differences between non peak and peak bin in the same region of influence are made identical in the synthesis STFT to correspondent phase difference in the analysis STFT.

For non-peak bin k, lying in the region of influence of peak l the synthesis phase formula becomes:

$$\angle Y(sR_s, k) = \angle Y(sR_s, l) + \angle X(sR_a, k) - \angle X(sR_a, l) \tag{3.6}$$

Identity phase locking dramatically reduces the phasiness, with only one complex multiplication per non peak bin. In fact identity phase locking requires trigonometric calculations only for peak bins: once computed the synthesis phase for a peak bin it is possible to calculate the angle θ required to rotate $X(sR_a, k)$ into $Y(sR_s, k)$ as:

$$\theta = \angle Y(sR_s, l) - \angle X(sR_a, l). \tag{3.7}$$

and obtain the phasor $Z = e^{j\theta}$ to rotate the non peak bins.

The synthesis equation for non peak bins then becomes:

$$Y(sR_s, k) = ZX(sR_a, k) \tag{3.8}$$

This technique really improves the quality of the traditional phase vocoder saving some computations, but it does not take into‘ account peak migration from one bin to another.

3.3.4 Scaled phase locking

Scaled phase locking improves identity phase locking recognizing that if a peak migrates from bin l at time index s to channel u at time index $s+1$, the unwrapping equation 3.1 should be based on $X((s+1)R_a, u) - X(sR_a, l)$ instead of$X((s+1)R_a, u) - X(sR_a, u)$.

Likewise phase propagation should be:

$$\angle Y((s+1)R_s, u) = \angle Y(sR_s, l) + [\omega_u R_a + \text{princarg}(\angle X((s+1)R_a, u) - \angle X(sR_a, l) - \omega_u R_u)]\frac{R_s}{R_a}. \quad (3.9)$$

The simplest way to determine which peak at time index s corresponds to u at time index $s+1$ is picking the peak at s lying in the same region of influence.

Then it is possible to lock the phases of non-peak bins, in a similar way as identity phase locking. Generalizing equation 3.6:

$$\angle Y(sR_s, k) = \angle Y(sR_s, k) + \beta[\angle X(sR_a, k) - \angle X(sR_a, u)] \quad (3.10)$$

where β is a phase scaling factor ($\beta = 1$ in identity phase locking).

Laroche and Dolson listening tests [26] show that setting β to an intermediate value between 1 and α (where α is the stretching factor) leads to better results in scaled phase locking. There is no strong theoretical background for this choice, but the reason is probably that phases of modified signals should not be linearly related to those in the original one.

Scaled phase locking can be summarized as follows:

- Find peaks in the actual STFT frame, and split the frequency axis in regions of influence;
- Connect peaks with previous peaks in the same region of influence;
- Propagate the phase for peak bins with (3.9);
- Lock the phase of non-peak bins to the phase of peak bins in the same region of influence with (3.10).

Scaled phase locking outperforms loose and identity phase locking, however some effects of phase dispersion are still present. In fact locking the phase between adjacent STFT channels dramatically reduces the introduction of beatings, but some phasiness remains due to the lack of phase locking between peaks:

1. during onset partial mis-synchronization results in softening the attack transient and altering source timbre (transient smearing);
2. some minor reverberant artifacts can be still heard due to the lack of “shape invariance”, i.e. lack of phase synchronization between harmonics of the same note;
3. floating and bass smearing can occur at low frequencies, due to the fact that two partials could be closer than two STFT peaks.

Achieving shape invariant time scaling of a signal requires to identify bins that compose a single note, and this is still subject of research. Some examples of shape invariant time scaling can be found in literature, as in [43] and [32], targeting monophonic sounds as speech.

3.4 Resolution problems

STFT suffers of the well known time-frequency resolution problem. If we consider an N-point STFT frame obtained without zero padding, the frequency resolution is $\frac{Fs}{N}$ where Fs is the sampling rate. It follows that frequency resolution increases linearly with N, but using rather long analysis window ends up in a less accurate mapping between spectral events in time domain. At the same time, roughly speaking, the period of a sinusoid with frequency $\frac{Fs}{N}$ is N samples, while the one of a sinusoid with $Fs/2$ is only two samples.

Human hearing perception is not linear in the frequency axis: hearing frequency resolution decreases exponentially with the frequency. This mean that using a multi resolution approach (i.e. using different time-frequency resolution in different sub-bands) is indeed a good idea. A multi resolution phase vocoder can be implemented through *wavelet transform* [22] instead of STFT as in [20] or with parallel windowing [5]. Hoak and Marcus proposed in [29] to convolve the spectrum with a variable kernel function in order to get a variable windowed spectrum.

Unfortunately all these techniques increase the computational load of phase vocoder, since they require more than a FFT for analysis frame or a complex convolution.

The first problem that can arise is bass/high smearing. If we choose a short analysis window, because the frequency resolution is low, two partials at low frequencies can fall in neighboring bins in STFT frames, and, due to spectral aliasing (Gibbs phenomenon) during synthesis phase there could be distortion (bass smearing). Vice-versa if we choose a long window, fast events at high frequencies are smeared on many frames, lowering the crispness and the presence of the sound (high smearing).

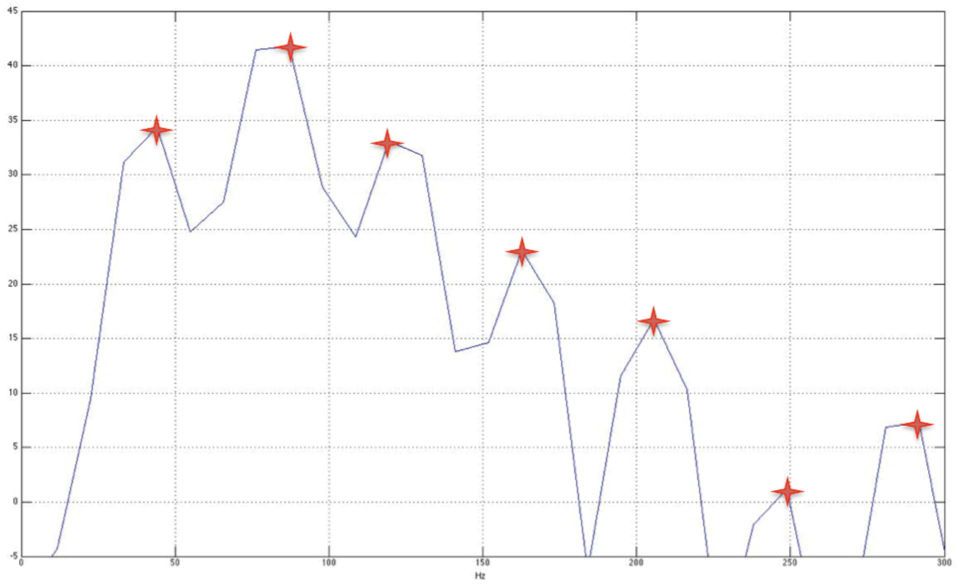
Unfortunately this problem is too intrinsic of STFT and cannot be completely avoided without using a multi resolution analysis framework. Zero padding can slightly improve frequency resolution but its effect is not dramatic (see figure 3.1).

Scaled phase locking can be improved in order to reduce resolution problems. Basically peaks can be picked non-uniformly in the DFT: having smaller region of influence at low frequencies helps to track close partials, while locking the phase in larger region on higher spectrum keeps residual and partials synchronized, reducing the loss of presence typical of phase vocoder.

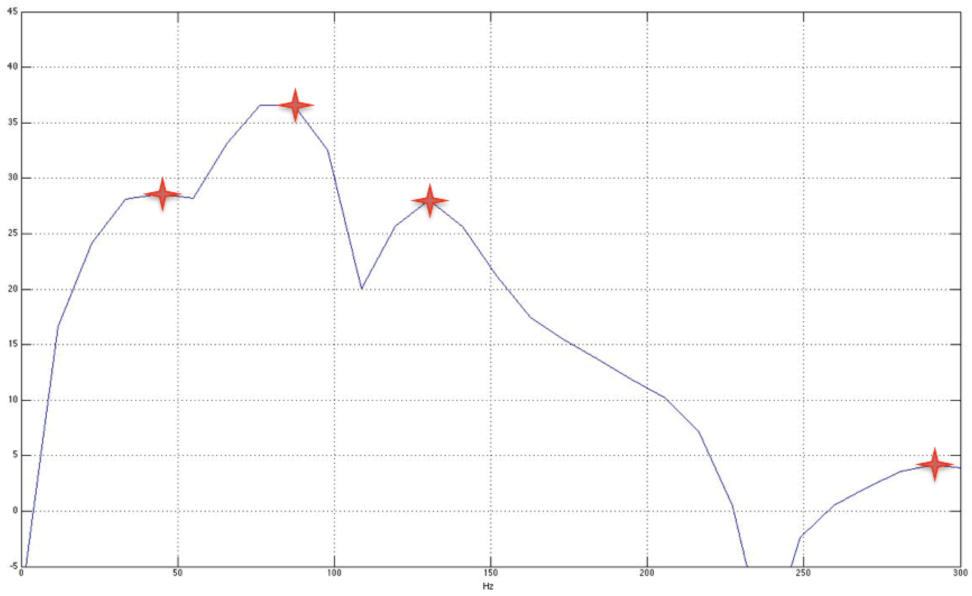
Consider again figure 3.1. Using the smaller window 3.1(b) leads to lock bins around 200 Hz to the peak at about 120 Hz, while from the DFT computed through the larger window 3.1(a), we can see that there is a peak at about 200 Hz that it is a partial (5th harmonic since the fundamental frequency of low E is located at about 40 Hz).

Now consider figure 3.2. Larger analysis window leads to lock bins in smaller region, while many peaks are spurious and represent residual. Vice-versa, smaller window helps keeping the residual phases synchronized.

Considering this, Karrer et al. [24] propose a non-uniform peak picking in the Fourier transform, with adequate heuristic to connect peaks between frames. Sakurai et al. [45] propose to divide spectrum in critical bands according to a psycho-acoustical scale, and then, in each critical band, to lock the bins phases to the most prominent peak in that band.

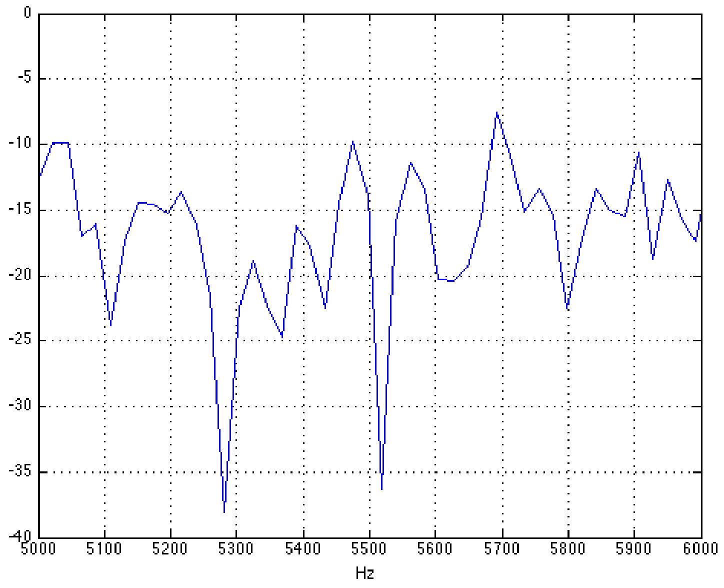


(a)

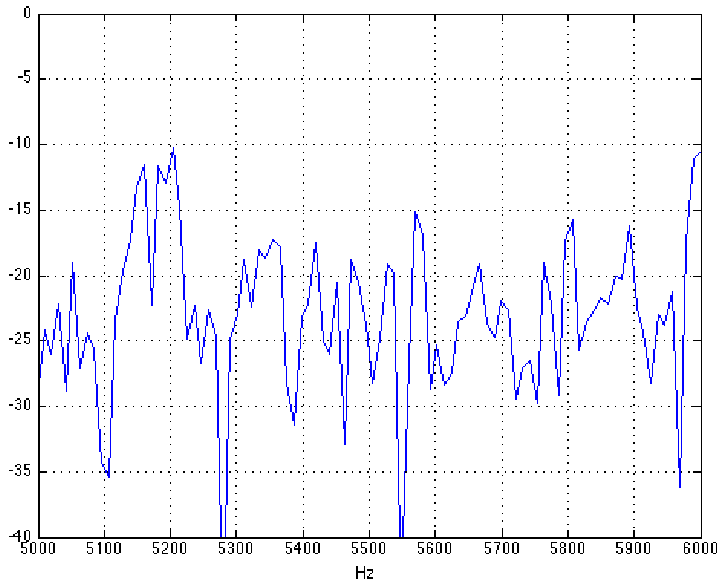


(b)

Figure 3.1: *Zoom of lower DFT of a double bass low E sampled at 44.1 KHz, computed on 4096 samples analysis window without zero padding 3.1(a) and 2048 samples with zero padding factor of two 3.1(b)*



(a)



(b)

Figure 3.2: *Zoom of the DFT of a hi-hat release sampled at 44.1 KHz, computed on 1024 samples analysis window 3.2(a) and 2048 samples 3.2(b) .*

3.5 Windowing

As mentioned in the previous chapter, a smoothing analysis window is necessary to reduce spectral leakage, and a synthesis smoothing window is needed too in order to avoid discontinuities between frames edge during overlap and add procedure, and to make the effect of time aliasing under the level of perception.

3.5.1 Analysis window

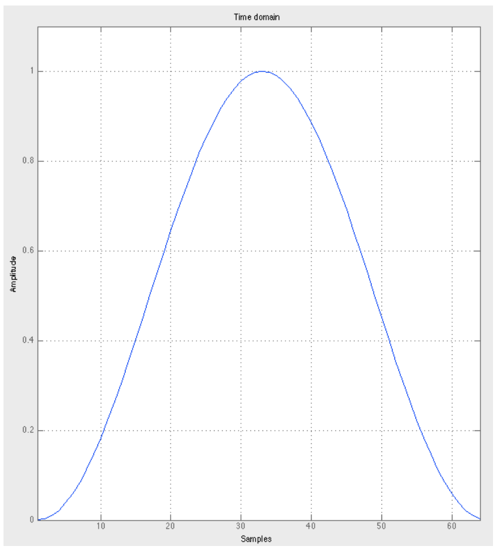
Briefly, multiplication in time domain corresponds to convolution in transform domain, so using a rectangular window in the analysis phase (i.e. using no window at all) introduces too much leakage, since the Fourier transform of a rect function is a complex sinc function.

Using an analysis smoothing window reduces leakage but affects resolution.

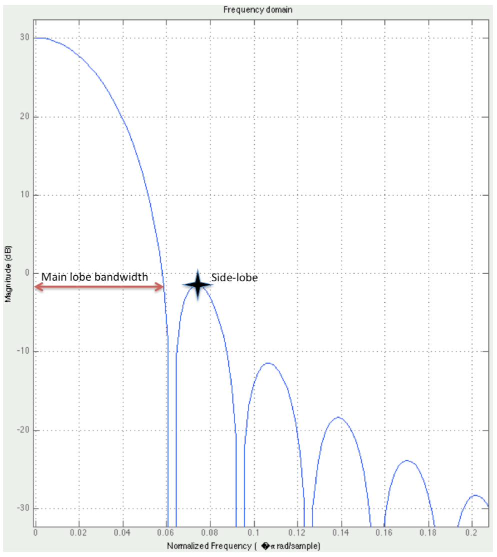
If we consider the Fourier transform of a smoothing window (figure 3.3), we can define the main-lobe as the central maximum, and side-lobe as a local maximum different from main lobe. We define the side-lobe level as the magnitude of a side-lobe, and main-lobe bandwidth as the minimum distance from the center such that the magnitude does not exceed the maximum side-lobe level.

Small main-lobe bandwidth is needed to preserve resolution, while small side-lobe levels are necessary in order to reduce the leakage. Unfortunately this two requirements are in contrast, so the choice of the window is a non trivial problem that depends on the application. Zero padding can help to not loose much resolution, at the cost of performing larger FFT.

An extensive comparison between different windows can be found in [47].



(a)



(b)

Figure 3.3: *Hanning window in time and frequency domain.*

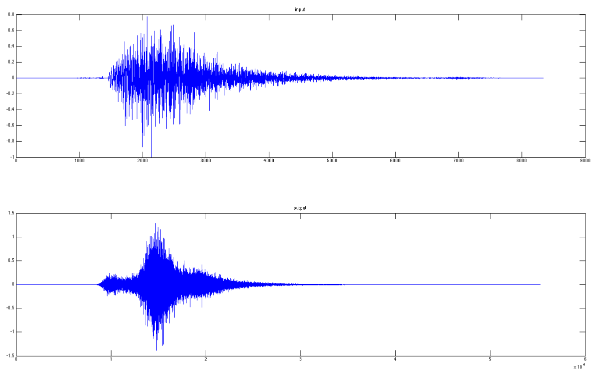


Figure 3.4: *Transient smearing of an hi-hat shot.*

3.5.2 Synthesis window

Right before the overlap and add procedure, the reconstructed signal frames need to be windowed again to ensure that there is no phase discontinuities at the frames edge, and furthermore phase modification corresponds to filter the signal through an all-pass filter, so there is some time aliasing due to circular convolution.

It is important to ensure that during the synthesis phase the sum of the windows remains constant, i.e. there is no amplitude modulation due to windowing.

Obviously window envelope depend not only on the synthesis window, but also on the analysis one. The effect of the overlap and add can be compensated through a sample per sample division, or can be dramatically reduced choosing an appropriate synthesis window such the truncated gaussian window [47] or asymmetric triangular window [40].

3.6 Transient smearing

As mentioned in the previous chapter, phase-vocoder phase interpolation mechanism works under the hypothesis that the audio signal is composed mainly by quasi-stationary sinusoids. However during an attack transient the quasi-stationary hypothesis is violated and results are poor (figure 3.4).

This happens for two reasons:

- phase dispersion: during attack transient partial phases mis-synchronization result in softening the perceived attack and in an alteration of the sound timbre.
- time scaling fast changing regions of signal inevitably leads to alter the perceived attributes of a sound source.

Rigid phase locking slightly reduces transient smearing because it guarantees phase synchronization between bins in the same region of influence. In fact phase synchronization between

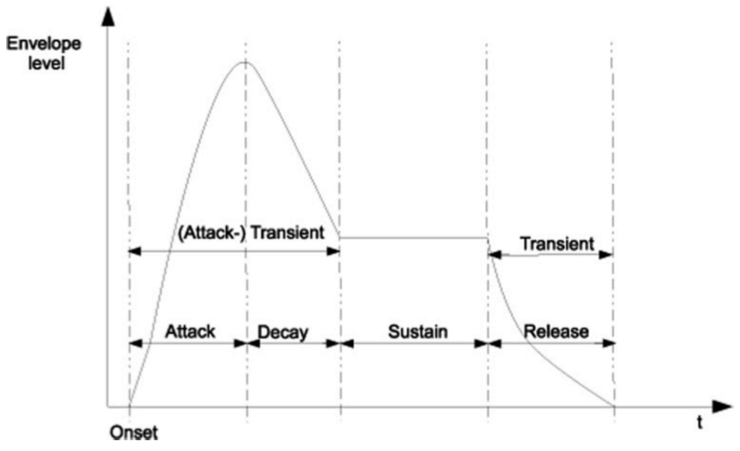


Figure 3.5: *Segmentation of a note.*

neighboring bins was originally proposed in [39] as a way to reduce smearing for integer factor time-scaling. However special techniques to treat attack transients are needed in order to achieve high quality time scaling.

In the previous chapter an attack transient was defined as "a situation in which amplitude change abruptly"; even if this definition is pretty straight-forward we need a more formal one.

There is not an unique definition of attack transient, and we will use one close to the one given in [53]. A single note can be segmented in 5 time intervals: *attack,decay,sustain,release* according to figure 3.5. Attack and decay are time intervals in which the amplitude of the note increase and decrease abruptly to reach the sustained (steady-state) portions of the note. The release is the "tail" of the note in which the notes fade out.

It follows that an attack transient is the portion of signal between the onset[1] and the sustained portions of a note (i.e. transients is the portions of the signal contained in attack and decay time). Notice that this definition is not always valid because, e.g. electronic synthesizer sound signals, attack and decay could be very long and exhibit quasi-stationary properties.

3.6.1 Transient detection

Before applying procedure to recover attack transients in the phase vocoder, it's obviously necessary to locate them in the sound signal. As we will see later, transient should be located on both time and frequency axis.

Even if transient detection is at the bottom of many DSP algorithms, it is a very complex subject, still matter of research. Good transient detection algorithm are frequently quite heavy and works along with different analysis framework as STFT, wavelets, cosine discrete transform, Sinusoidal plus noise signal decomposition, linear prediction etc. For this reason in time scaling algorithms, as in [40], [5] and [51], transient detection is frequently performed off-line and only

[1]Onset is the time index at which the note start.

time scaling is performed real time.

For the purpose of this work, we will only treat transient detection techniques that work in the time-frequency domain, since for design issues a real-time transient detection was needed.

Basically transient detection algorithms working in the STFT domain search for energy and phase changing; according to [3] they can be divided into three groups:

- based on energy information;
- based on phase information;
- based on phase and energy information (sometimes called *hybrid*).

3.6.1.1 Detection based on energy information

There are several techniques based on energy information, and most of them rely on measures of energy difference between consecutive STFT frames. In their simplest implementations, transient detection is performed by checking when a *transient function* exceeds a certain threshold. A very simple transient detection function could be spectral energy difference:

$$S(n,B) = \sum_{k=l}^{u} [X(n,k)^2 - X(n-1,k)^2]^{\frac{1}{2}} \tag{3.11}$$

or the spectral distance:

$$E(n,B) = \sum_{k=l}^{u} [|X(n,k)| - |X(n-1,k)|]^2 \tag{3.12}$$

where l and u are the lower and upper bounds of the band B considered.

Refinements can be done as in [30] by observing that energy content migrates toward high frequencies during transients and then weighting the transient function differently for each sub-band considered.

3.6.1.2 Detection based on phase information

Another common approach to transient detection rely on the fact that for a stable sinusoid the phase difference between two frames is approximately constant, i.e.:

$$\phi(n,k) - \phi(n-1,k) \cong \phi(n-1,k) - \phi(n-2,k) \tag{3.13}$$

where $\phi(n,k)$ is the measured phase of the STFT bin k at time index n. It follows that the phase deviation

$$\Delta_\phi(n,k) = |\text{princarg}(\phi(n,k) - 2\phi(n-1,k) + \phi(n-2,k))|. \tag{3.14}$$

will tend to 0 if the phase value is accurately predicted, which implies that the analyzed signal at this time instance consists of stable parts. On the contrary, during transients, Δ_ϕ will assume larger values since the instantaneous frequencies are not well defined for these parts.

One simple detection function based on this assumption could be:

$$T_p(n,B) = \frac{\sum |\text{princarg}[\phi(n,k) - 2\phi(n-1,k) + \phi(n-2,k)]|}{|B|}. \tag{3.15}$$

Furthermore Bello and Sandler [4] propose a method for transient/steady state separation that uses phase deviation along with statistical analysis. In particular, they notice that for STFT peaks in steady-state portions of signal, the phase deviation has a variance close to 0, while during a transient it assumes large values.

3.6.1.3 Detection based on phase and energy information

Energy-based algorithms are usually fast and easy to implement, however their effectiveness decreases when transients of signal are not pronounced and when energy bursts of different events overlap in polyphonic mixtures. On the other hand, detection based on phase information is sensible to noise and phase distortion. Better detector can be implemented using both phase and energy information.

3.6.1.4 Complex domain detection

The first technique here illustrated was originally proposed in [3], and imply a fully combined approach where energy and phase information is simultaneously analyzed. For locally stated regions in audio signals, it can be assumed that frequency and magnitude values remain approximately constant, so by predicting values in the complex domain, the effect of both phase and energy can be considered. Consider the polar form of the predicted value of the k^{th} bin of STFT frame at time index m:

$$\bar{X}(k,m) = |\bar{X}(k,m)|e^{j\bar{\phi}(k,m)} \tag{3.16}$$

where the predicted magnitude $|\bar{X}(k,m)|$ corresponds to the magnitude of the previous frame $|X(k,m-1)|$, and the target phase can be computed through equation 3.13 as:

$$\bar{\phi}(k,m) = \text{princarg}[2\phi(k,m-1) - \phi(k,m-2)|. \tag{3.17}$$

Now consider the measured value in the complex domain from the STFT:

$$X(k,m) = |X(k,m)|e^{j\phi(k,m)}. \tag{3.18}$$

By measuring the euclidean distance between the predicted and the actual value of the bin in the complex space we can quantify the stationarity of the k^{th} bin as:

$$\Gamma(k,m) = |X(k,m) - \bar{X}(k,m)|. \tag{3.19}$$

Summing these stationarity measures across all bins, it is possible to construct the detection function:

$$\gamma(m) = \sum_k \Gamma(k,m). \tag{3.20}$$

Equation (3.19) can be written as:

$$\Gamma(k,m) = |X(k,m) - \bar{X}(k,m)| =$$
$$= \left\{ \{Re[X(k,m)] - Re[\bar{X}(k,m)]\}^2 + \{Im[X(k,m)] - Im[\bar{X}(k,m)]\}^2 \right\}^{1/2} =$$
$$= \Big\{ \{|X(k,m)|\cos[\phi(k,m)] - |\bar{X}(k,m)|\cos[\bar{\phi}(k,m)]\}^2 +$$
$$+ \{|X(k,m)|\sin[\phi(k,m)] - |\bar{X}(k,m)|\sin[\bar{\phi}(k,m)]\}^2 \Big\}^{1/2} . \quad (3.21)$$

Rotating the axis:

$$\Gamma(k,m) = |X(k,m) - \bar{X}(k,m)| =$$
$$= \Big\{ \{|X(k,m)|\cos[\phi(k,m) - \bar{\phi}(k,m)] - |\bar{X}(m,k)|\}^2 +$$
$$\{|X(k,m)|\sin[\phi(k,m) - \bar{\phi}(k,m)]\}^2 \Big\}^{1/2} . \quad (3.22)$$

It follows that, if the phase prediction is "good", only the energy difference is taken into account; if amplitude is constant, then Γ is proportional to phase deviation equation (3.14) weighted by the bin magnitude.

In steady state portions of signal, $\gamma(m)$ will assume low values, while during transients it will rise up to much higher values. It is possible to bound the function $\gamma(m)$ in the interval $[0, 1]$ using a simple peak follower:

$$\gamma_0(m) = \sum_k \Gamma(k,m)$$
$$\gamma(m) = \frac{\gamma_0(m)}{\max[\gamma_0(0), \gamma_0(1) ..., \gamma_0(m)} \quad (3.23)$$

so $\gamma(m)$ will assume values close to 0 during steady state portions of signal, and values close to one during transients.

Dealing with different sound signals requires different thresholds, but also for a single signal an optimal fixed threshold can perform poorly: in general, especially in complex audio mixes, in a same audio track there are different kind of events, each one with its own characteristics in time-frequency domain.

Let's consider for example an audio track made of a constant amplitude noise burst followed by a violin solo. If we put an high threshold, since violin attack transient are "soft", most of the violin attacks are not going be detected. Vice-versa, if we put a low threshold in order to detect correctly all the violin transients, during the noise burst the many not existing attack transient are going to be detected.

To avoid this kind of problems a *dynamic threshold* δ_t is required. A simple but effective dynamic threshold, originally proposed in the context of impulse noise detection [28] and adapted to complex domain attack transient detection in [3], is the weighted *median*[2] of an H-length section of the detection function around the corresponding frame:

$$\delta_t(m) = C \ \text{median}[\gamma(k_m)], \quad k_m \in [m - \frac{H}{2}, m + \frac{H}{2}] \tag{3.24}$$

where $C > 1$ is a predefined weighting value.

The idea behind the use of this adaptive threshold is rather simple: as soon as an attack transient frame is analyzed values of γ assume larger values than "average". However median gives us a more robust threshold than arithmetic average: it's more robust to periodic events happening in the analyzed frames, and thanks to look-ahead it avoid to detect as transient a non transient event that is happening after a long portion of signal where γ is close to zero.

To avoid the threshold from rising in the position of a peak, the interval length H has to be set longer than the assumed duration of the peak in the transient function.[3]

As anticipated this method leads to a very accurate transient detection, with relatively low computational costs. In order to localize transient not only in the time axis, but also in the frequency axis, complex domain transient detections can be performed with a sub-band approach as in [53] and [40].

3.6.1.5 Center of gravity based detector

In the context of phase vocoder, Axel Röbel ([42] [41]) proposed a peak per peak approach to transient detection, in order to be able to perform different operations on stable bins from bins representing an attack transient.

The idea behind Center of Gravity (*COG*) transient detection is very simple and intuitive. Consider a stable sinusoid: in every analysis frame, the COG of the instantaneous energy is located in the center of the window. Now consider a sinusoid multiplied by a linear ramp with saturation (a simple attack model). Until a portion of a ramp is inside the analysis window the temporal COG of the signal is located to the right of the window center (see figure 3.6).

So it follows that, if the COG of an analysis frame is located to the right of the window center, in the frame is happening an attack transient situation. In order to separate the steady-state spectral component in the STFT, the COG is computed for each peak as explained below.

The COG of the instantaneous energy of a signal $s(t)$ is defined as:

$$\bar{t} = \frac{\int ts(t)dt}{\int s(t)dt} \tag{3.25}$$

and can be computed in the Fourier domain as:

$$\bar{t} = \frac{\int -\frac{\partial\phi(\omega)}{\partial\omega} S(\omega)d\omega}{\int S(\omega)d\omega} \tag{3.26}$$

[2]The median of a set is defined as the middlemost value of an ordered table of the set values.

[3]In [28] H was originally set shorter than the duration of a transient in order to detect impulses different from transients.

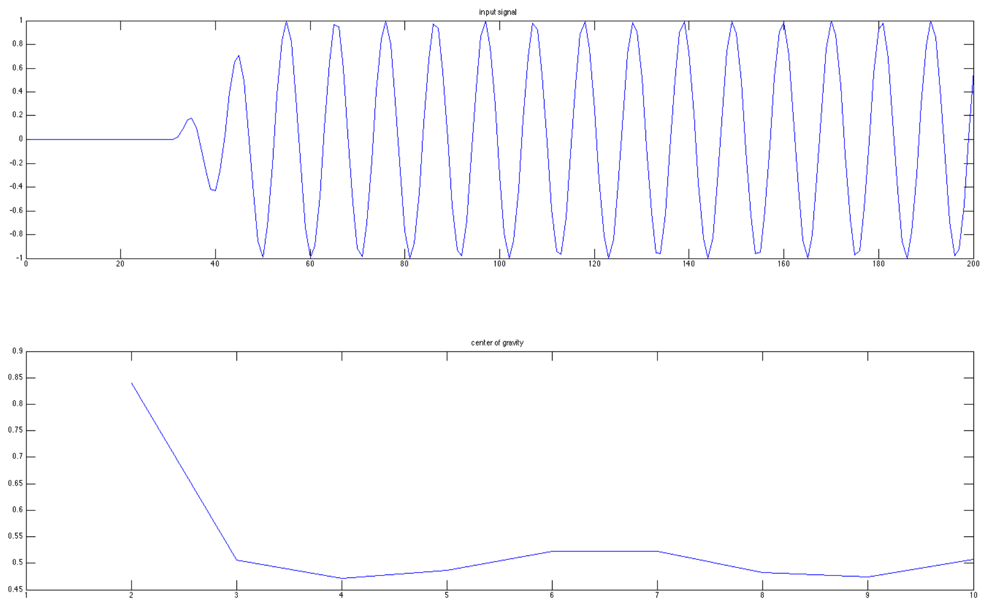


Figure 3.6: *COG over time of a ramped sinsuoid*

where $-\frac{\partial\phi(\omega)}{\partial\omega}$ is the group delay.

The COG can be computed in the DFT replacing the integral with a summation and the derivative with the differentiation in the properly interpolated DFT spectrum. The group delay can be also computed efficiently by means of a DFT of the signal with a modified analysis window [2].

In order to be able to compute the COG for a single peak at frequency ω_0 in a frame at time index m, Röbel proposed to modify equation (3.26) as:

$$\bar{t}_m(\omega_1) = \frac{\int_{\omega_l}^{\omega_u} -\frac{\partial\phi(\omega)}{\partial\omega} S(\omega) d\omega}{\int_{\omega_l}^{\omega_u} S(\omega) d\omega} \tag{3.27}$$

where ω_l and ω_u are the frequency of the amplitude minima below and above the current maximum.

Considering that COG transient detection is straightforward and can be summarized as:

- Compute the COG for every peak in the current STFT frame;
- If for a peak the COG is above a fixed threshold C_s mark the peak as transient peak;
- If the number of the peak marked as transient is above a dynamic threshold an attack transient is detected;

- Until the end of the attack transient event collect all the peaks of consecutive STFT frames that have COG above the threshold C_s in a non-contracting set.

- The attack transient is finished [4] when the spectral energy of transient bins having a COG above a second threshold $C_e < C_s$ in the current frame is smaller than half the spectral energy contained in the transient bin set.

In particular, the dynamic threshold proposed by Röbel is very simple and relys on a binomial statistical model using short history of F_h frames. The spectrum is partitioned in non-overlapping sub-band and a transient is detected when the probability of transient for at least one band in the current frame is higher than the probability of non-transient. If N is the number of independent events of the statistical process (i.e. the number of peaks that can be detected in the current band), and p the probability of a transient peak, the variance of the binomial distribution is:

$$\sigma^2 = p(1-p)N. \tag{3.28}$$

So, if the number of peaks marked in transient in the current band is n, we want to select the transient probability such that it is consistent with n within the range of G times the standard deviation:

$$n = pN \pm G\sigma, \tag{3.29}$$

where the plus and minus sign are used to determine respectively the transient probability p_c for the current frame and for the frame history p_h.

An attack transient is detected if the current frame transient probability is larger than the frame history transient probability.

This method leads to very good results, because it separates steady-state component from transients inside the frame. Unfortunately it requires an additional FFT per frame in order to compute the group delay, or alternately requires spectral interpolation.[5]

3.6.2 Transient preservation

Once an attack transient is detected, it is important to take special measures to preserve it, without creating much discontinuities in the stable spectrum. Basically, a transient is preserved by locking in some way the transient bins phases to the measured phases, but, anyhow some smearing can still remains due to the different synthesis hop size from the analysis hop size. Below we present three techniques, that can be considered the state of art of transient preservation in the phase vocoder.

[4]As explained in section 3.6.2.3 this is not the frame in which the attack transient finish, but in which it is located in the center of the window.

[5]Spectral interpolation is computationally a more expensive solution, since requires interpolation on a FFT obtained with zero-padding. See [46] pg. 45.

3.6.2.1 Non-linear time scaling

The first and simpler technique, originally proposed in [17] and implemented real-time in [40], consists in not time scaling transients and locking the synthesis phase to the analysis phase.

In a pre-analysis stage, onsets are located using a fixed resolution sub-band analysis and the complex detection function (see section 3.6.1.4). A constant-Q filter bank splits the signal into four sub-bands, and the complex function is calculated separately for each band. Once detected, onsets are combined in order to keep a maximum of one onset for a fixed-length window. As higher frequency bands have better time resolution, a higher band always takes precedence over a lower one.

Tonal onsets are differentiated from percussive onsets. This is performed by looking to the transient time length and bandwidth. Detected onsets segment the signal in notes, and an attack transient is then assigned to the first part of the note (from 10 to 50 ms depending on the note length).

After having detected attack transients and relative durations, the stretching factor is adjusted in order to keep the global stretching ratio. If α is the global time stretching factor, the adjusted one is:

$$\bar{\alpha} = \frac{W_t}{W_{ss}}(\alpha - 1) + \alpha \tag{3.30}$$

where W_t and W_{ss} are the length respectively of transient regions and of steady-state regions .

In phase vocoder synthesis transient are not time scaled, the synthesis phase is locked to the analysis phase in the temporally masked region of percussive transients. In particular, for the first synthesis frame of a percussive attack transient region the previous synthesis phase is replaced with the analysis phase.

3.6.2.2 Constant frame rate time scaling

In the context of near-lossless time-scale modification of audio Bonada [5] proposed a different approach to phase vocoder time scaling. Instead of changing the synthesis hop size, the hop size used is the same both in analysis and synthesis. Time scaling is achieved discarding ($\alpha < 1$) or repeating some frame ($\alpha > 1$) as in figure 3.7.

To avoid smearing, transients are not scaled (no frame is repeated and discarded during transient), and the synthesis phase is locked to the analysis phase. In particular the measured spectrum phase is used (phase re-initialization) above a specified frequency cut F_c; for frequencies lower than F_c the phase is continued for quasi-stable peak while the original phase is used for non stable peak.

Bonada in his work classifies attack transients in different types, and for each type defines a different value of F_c.

3.6.2.3 Röbel's transient preservation

Röbel investigated the problem of phase re-initialization of phase vocoder. Since during transient the signal does not have a predictable relation with the previous frames, the phase re-initialization is inevitable if the shape of the signal shall be recreated. However, deciding in which position to

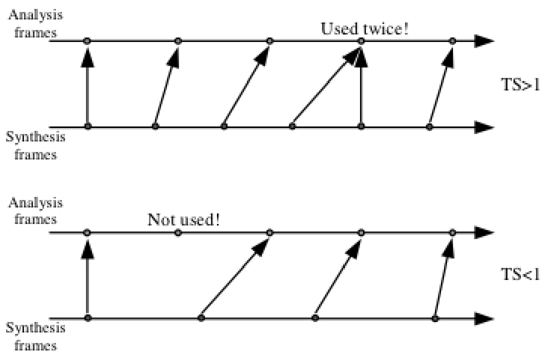


Figure 3.7: *Constant frame rate time scaling*

re-initialize the phase is not a trivial problem. In fact the phase should be re-initialized when the transient is happening about in the middle of the window for two main reasons:

1. the transient is reproduced without any error only in the frame that gets the phase reinitialized, so the reinitialization should take place when the impact of the reconstructed transient on the output signal is larger.

2. reinitialization of the phases will produce the transient at the very same position where it was located in the analysis window, and during synthesis the frame is repositioned to its new location using the frame center as time reference; it follows that to avoid the need to relocate the transient, reinitialization should take place when the transient is in the center of the window.

Since COG is linearly related[6] to transient position in the window, and since the COG based transient detection function stops detecting the transient when the COG of transient bins is located about the center of the window, it follows that the right position to perform the phase re-initialization of transient bins should take place in the frame for which the detection function finishes the detection of the transient.

Non transient bins phases should be continued with the phase vocoder algorithm, in order to not introduce phase discontinuity artifacts on steady-state regions of the signal.

Two other measures has to be taken in order to avoid artifacts:

1. during the transient and before phase re-initialization, transient bins magnitude and phase should be kept equal to the one of the last non transient frame to avoid pre-echo artifacts;

2. transient bins magnitude should be slightly amplified in the frame in which phase re-initialization takes place to compensate the lack of contribution from previous frames.

[6]In fact COG is linearly related to the transient position only when the transient is happening about in the center of the window.

3.7 Conclusions

In this chapter several phase vocoder issues and improvements have been analyzed. In literature exist many other improvements/variants to the standard phase vocoder, as for example methods to achieve shape invariance in fixed time intervals [34], that here are not treated because are not fundamental for the real-time implementation.

In the next chapter, many of the results presented here will be used to build a full real-time transient preserving phase vocoder, that provides a good quality time stretching.

Chapter 4

Proposed Time Stretching Technique

In this chapter we will present a new and fast transient preservation algorithm for the traditional phase vocoder. This technique is based on most of the transient preservation and detection techniques described in the previous chapter, but it works full real-time with very low latency. Other techniques to improve the audio quality of time scaling output will be introduced along with two implementations of phase locking. Furthermore, a zero-latency real-time time stretching implementation including the proposed techniques will be presented here, along with formal and informal listening tests results.

4.1 Basic idea

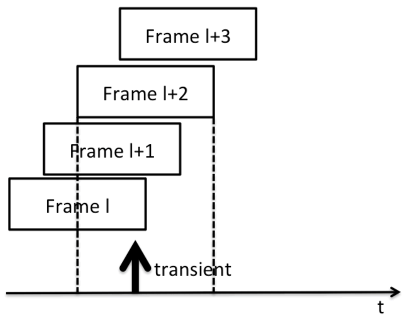


Figure 4.1: *Basic idea to find the frame in which the transient is occurring in the center of the analysis window.*

As pointed out in section 3.6.2.3, when time-scaling with constant time-stretching factor, if a transient is occurring, the synthesis phase should be equal to the analysis frame when the transient is occurring at about the center of the window.

In particular Röbel states in [41] that for a step transient the normalized root mean squared error (NRMSE) between the signal frame after relocation with the correct signal is below 25%

for transient positions in the center of the window and if the frame relocation offset is smaller than an 8th part of the window size.

Since for real-time and computational issues the COG transient detection was considered too time-consuming (see section 3.6.2.3), a workaround was needed.

The trick to find the frame in which the transient is occurring in the center of the window is rather simple: if it is possible to detect the transient as soon as it appears in the analysis window all we need is a counter!

Consider figure 4.1. Here a transient is detected as soon as it enters in the analysis window, in the analysis frame with time index l. If we know the window length and the analysis hop size (or equally the analysis overlap factor), to find in which analysis frame the transient is occurring about the center of the analysis window is straightforward: the time position of the transient does not change, what is changing is the time index of the analysis frames.

Formally, assuming that a transient can be detected as soon as it appears in the analysis window, if the detection occur at time index l the frame in which it will appear in the middlemost position is the frame with time index $l + o/2$, where o is the overlap factor:

$$o = \frac{n}{R_a} \tag{4.1}$$

where n is the window length and R_a the analysis hop size.

Even if the idea behind this algorithm is simple, there are several techniques that we need to develop:

- an efficient detection function that triggers a transient event as soon as it appears in the analysis window;
- a synchronization mechanism between different transients happening in the same window;
- a method to avoid discontinuity artifacts on stable spectral region of signal while reconstructing the transient;
- a technique to treat frames in which the transient is occurring, prior to the frame in which the transient is occurring in the center of the window;
- a method to avoid post-eco when the stretching factor is large.

4.2 Time scaling model and parameters

The transient preserving technique is designed to work real-time within the traditional phase vocoder model, along with different phase locking technique.

Time scaling is achieved through using a synthesis hop size proportional to the stretching factor:

$$R_s = R_a * \alpha. \tag{4.2}$$

Since a multi-resolution approach was considered too expensive for a fast real-time implementation, a good trade-off between high and bass frequencies resolution was achieved imposing an analysis window length of 2048 samples and an overlap factor of 4 with a sample-rate of 44.1kHz, that in the time domain correspond to a window length of 46 ms, with an hop size of 11.6 ms.

The chosen analysis/synthesis window is a periodic Hanning window since it exhibits a good trade-off between side-lobe level and main-lobe width, and assume zero value on the edges, reducing the time aliasing introduced by the circular aspects of convolution. The analysis is performed through zero-phase analysis but, in order to save computation, without zero-padding.

4.3 Transient handling

4.3.1 Detection scheme

The detection function used is the complex domain transient detection explained in section 3.6.1.4.

In order to not introduce latency, the dynamic threshold δ_t is the median filter without lookahead:

$$\delta_t(m) = C \ \text{median}[\gamma(k_m)], \quad k_m \in [m - H, 0] \tag{4.3}$$

where γ is the value of the detection function, and $C > 1$ the scaling factor.

Using the median filter without lookahead makes the detection scheme more sensitive and, at the same time, less robust than using the lookahead as shown in [53].

The fact that the detection is more sensitive is positive because the transient is detected as soon as it appears in the window, but the lack of robustness causes a significative number of false detection, leading to unpleasant artifacts in the synthesis stage.

For this reason a triple treshold scheme was introduced (see figure4.2).

In first place the spectrum is partitioned in a certain number of non-overlapping sub-bands, in order to localize the transients both in time and spectral region.

For each band the complex detection function is then computed, along with the energy deviation between the current frame and the previous one.

If both the complex detection function is above its dynamic threshold and there is some significant increase in the energy of the considered sub-band, a transient event in the relative sub-band is detected.

Finally, if the number of bands marked as “in transient” is above a fixed threshold than a transient event is detected.

The reasons behind this triple threshold scheme are multiple and intuitive.

The first one is to avoid false detections that can cause disastrous effects on the synthesis stage, without affecting the time localization of transients. In fact, the simpler technique to avoid false detections should be increasing C in the dynamic threshold equation. However this action can cause several problems:

- bad time localization of transients;

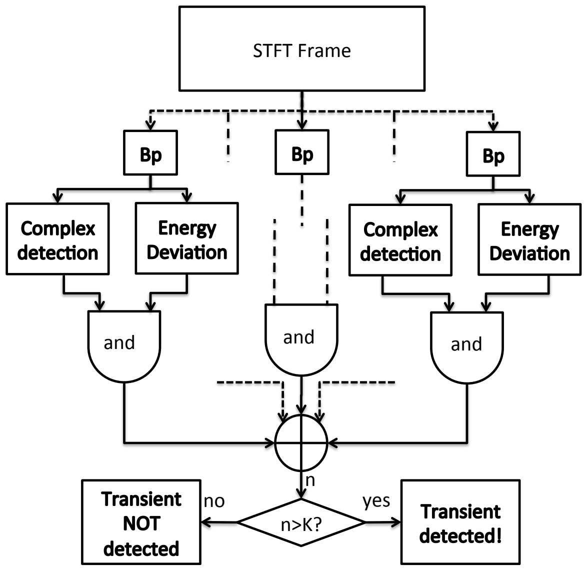


Figure 4.2: *Proposed transient detection scheme.*

- high probability of false negatives;
- false positives can still be detected, e.g. if some frequency modulation is applied to a stable sinusoid.

It follows that we need a mechanism to avoid false positives, and at the same time to allow false negatives only for those transients that can be "smeared" without causing perceptually significant artifacts.

The bandwidth and the energy deviation gives us a good indication of the "hardness" of the transient and of its perceptual level.

Furthermore thresholding the number of bands in transient after evaluating the energy deviation is a good idea. In fact a false detection with the complex detection function can occur in two bands if some spectral energy moves from one band to another for some reason (e.g. portamento or frequency modulation). Thresholding the positive energy deviation causes the detection function to detect the transient only in one band, and so triggering the number of bands marked as in transient avoid this kind of false detections.

4.3.2 Synchronization

Once a transient is detected, a non-contracting set of transient bands is created. Then the detection is performed on a certain number of consecutive frames, adding to the transient bands set every band for which a transient situation is triggered.

Since in the higher spectrum more accurate time resolution is needed but at the same time events above a certain frequency F_p are perceptually not very important, the transient "starting" position corresponds to the frame for which the higher band below F_p is first marked as in transient.

4.3.3 Transient preservation

Transient preservation is achieved basically by using the analysis measured phases in the synthesis of the frame occurring half window length after the transient starting position, that will be mentioned as *reset position*, since basically we are "resetting" the synthesis phase of phase vocoder.

As pointed out in [6], during a transient it is better to use the original phases as much as possible, but to not impair stable spectral region, the spectrum must be partitioned in order to use the original phases only around the peaks in which the transient is occurring. At the same time, the impairment produced by using the original phases on stable spectral region above a certain cut frequency F_c is below the level of perception.

So, the transient preservation is performed using the original phases for all the bins with frequency above F_c. For the sub-bands below F_c the preservation scheme is applied only if the considered band is marked as "in transient".

This approach is clearly sub-optimal since in the same band there could be both stable and un-stable bins, but, as stated in [53], a sub-band approach to transient processing can lead, perceptually to better results than a bin per bin approach.

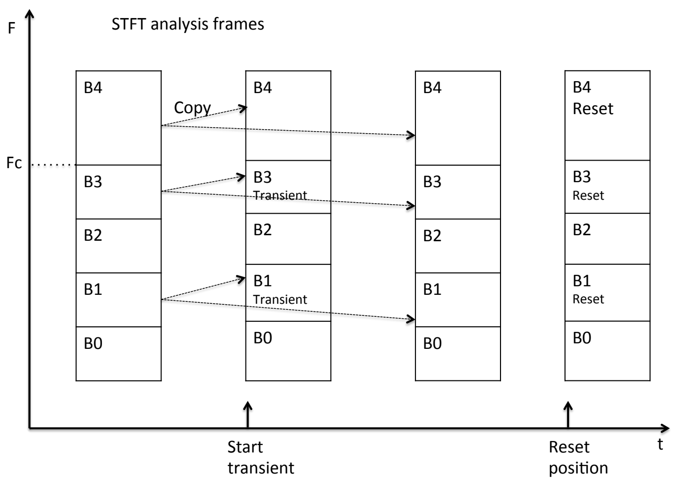


Figure 4.3: *Transient preservation scheme.*

To completely avoid smearing and pre-eco artifacts, for transient frames before the reset position some action needs to be taken. Similarly to Röbel's scheme, what is proposed in this work is to use the magnitude and phase values of the last frame marked as non in transient as phase vocoder analysis data for the bands in which the transient is occurring (figure 4.3). Basically it corresponds to repeat some spectral portions of the signal for some synthesis frames.

For relatively small stretching factors this technique does not introduce perceivable artifacts, while for large expansions of the signal it can cause annoying reverberant artifacts. A way to avoid this artifact is to anticipate of one frame the reset position, and then impose $\alpha = 1$ for two consecutive frames while keep using the original phase in the transient bands.

Imposing $\alpha = 1$ for some synthesis frames helps to avoid eco artifacts when the stretching factor is large and also for smaller stretching leads to a more accurate transient reconstruction.

Obviously the effect of this temporary change of stretching factor must be compensated, and very quickly in order to keep the rhythm in case of closed transient in time (e.g. a drumroll).

4.3.4 Implementation

The preservation scheme clearly runs with the same temporal parameters of the main algorithm, with a 46 ms window and an overlap of 4.

The detection is performed in 12 non-overlapping bands each one corresponding to two adjacent bands of the bark scale [54], and the dynamic threshold is computed on an history of 7 frames (about 80 ms) with the scaling factor C empirically set to 1.5.

The bark scale is a psycho-acoustical scale that approximate the critical bands of hearing, for more information see [55]. The bark-scale is theoretically a good choice in order to partition the spectrum in transient and stable bands, but, because the analysis window is too small, there is no enough resolution to perform the transient detection on the full scale.

Zaunschirm, Reiss and Klapuri in [53] use a 6 band logarithmic scale to partition the spectrum, but this partitioning scheme causes some discontinuities artifacts for stable portions of signal while performing the transient preservation technique, and furthermore sometime leads to smear some spectral region of a transient.

Using an “half” bark-scale gives us a good trade-off between accuracy and resolution.

The energy deviation for the m-th frame and the k-th band is computed through:

$$\Delta E_k(m) = \frac{E_k(m) - E_k(m-1)}{E_k(m)}, \tag{4.4}$$

where E_k is the spectral band energy. The energy difference is divided by the band energy in order to bound the energy deviation in the interval $(0, 1]$ in case of positive energy deviation. Notice that, differently from the energy based detection function described in the previous chapter here we are not considering the absolute value of energy deviation, since we are only interested in energy increases.

Empirically a good trade-off between smearing some small transients and avoid false positive detection was achieved putting a threshold of about 0.4 on the ΔE function, while a transient is detected as soon as at least three bands are marked as in transient. Thresholding the energy causes a delay of one frame in the detection since the samples near the window edge are heavily attenuated by the analysis window, so, during the synchronization step this effect must be taken in account.

The synchronization is performed on 3 consecutive frames (about 33 ms), and the upper frequency bound for synchronization (F_p) is about 6kHz.

When a transient is detected, all bands located at frequencies above $F_c = 2$kHz are considered as in transient, while for bands located below F_c the preservation procedure take place only if a transient situation is detected in the considered band.

For the frame in which the phase re-initialization takes place the magnitude of the bins belonging to the bands marked in transient is slightly amplified for two reason: the first is to compensate the lack of contribution from the previous frames, the second one is to mask the effect of phase re-initialization for stable spectral region included in transient bands. Empirically a good gain factor value of 1.1 was found. In figure 4.4(c) is shown a comparison of drum shots time stretched with the standard phase vocoder and the proposed algorithm.

4.4 Phase locking

As stated in the previous chapter rigid phase locking techniques reduce dramatically the phasiness in the phase vocoder synthesis. Unfortunately both identity and scaled phase locking are patented so, for the Reactable implementation a phase locking technique based on sinusoidal models was used, while for testing the quality of the transient preservation algorithm a rigid phase locking technique was implemented.

4.4.1 Rigid phase locking variant

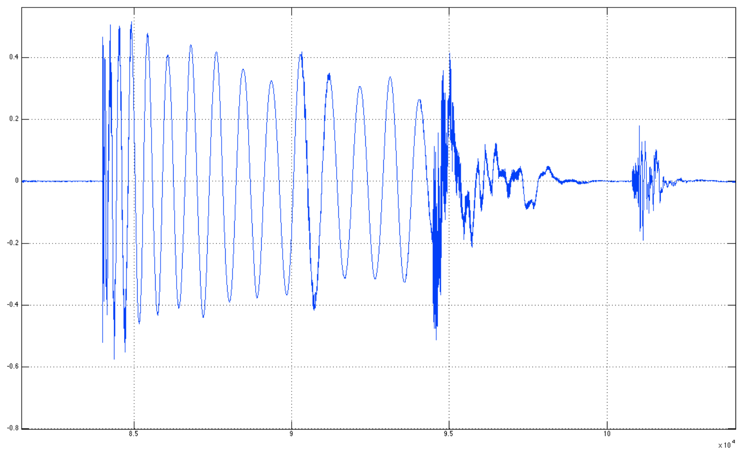
The rigid phase locking scheme proposed here relies very much on scaled phase locking (see section 3.3.4). As in the scaled-phase locking the STFT peaks divides the spectrum in region of influence, delimited by the frequency index with minimum magnitude between two peaks. Once computed the synthesis phase for the peaks, for all other bins the synthesis phase is computed in order to keep the analysis phase distance between bins and the peak.

The differences from the standard scaled-phase locking are two.

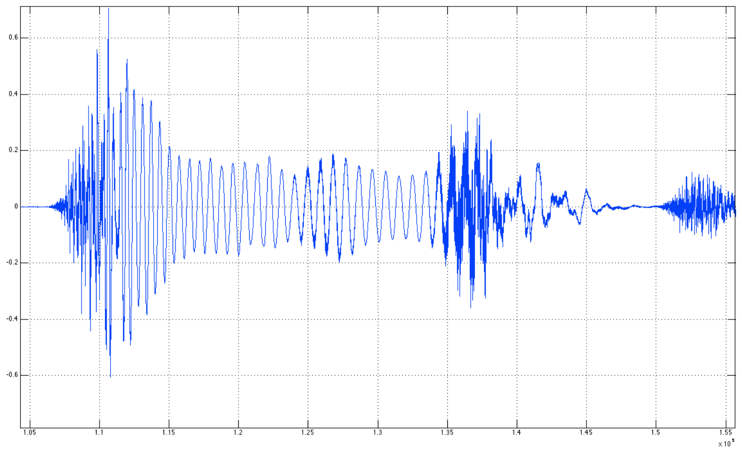
The first is that, for every frame the lower 6 bins are marked as peak (0-130 Hz). The reason behind this choice is that, as pointed out in [24], because the lack of resolution closed partials located at very low frequency can be marked as non-peak, causing an annoying distortion artifact since the instantaneous frequency is not preserved.

The second is in the connection between different peaks in different frames. Laroche and Dolson [25] propose to simply connect the peak in the current frame with the previous frame peak closed in frequency. This can lead to connecting peak representing partials with spurious peak and vice-versa, since the peak picking is performed simply searching local maximum. Furthermore, the poor resolution in the lower spectrum can lead to connect peaks representing different partials, causing minor artifacts.

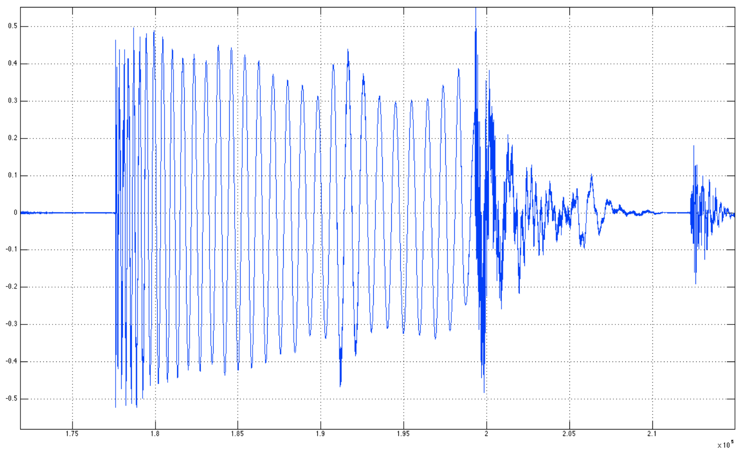
To avoid this problem, the assumption is that one peak can move of $\pm k$ positions from one frame to another. If in the current frame a peak at frequency index l is found, if in the previous



(a)



(b)



(c)

Figure 4.4: *Comparison of the original drum shots (fig. 4.4(a)) with the time stretched version ($\alpha = 2$) obtained with the standard phase vocoder (fig. 4.4(b)) and with the proposed algorithm (fig. 4.4(c)).*

frame is present a peak in $[l - k, l + k]$, the two peak are connected, and the unwrapped phase is computed accordingly to equation (3.9). If in the previous frame there are no peaks in $[l-k, l+k]$, the previous bin considered as the previous peak in equation (3.9) is chosen by taking the bin in the interval $[l - k, l + k]$ that most likely represent the same partial in the previous frame.

In [12] a probability function is defined and used to connect peak in the context of sinusoidal modeling. This function relies on spectral difference (magnitude and phase) and frequency index difference between two consecutive peaks. What is proposed here is to use only the magnitude and frequency index information to connect the current peak with the previous not peak bin, choosing the bin closest in magnitude in the interval $[l - k, l + k]$.

Preliminary tests show that this peak connecting heuristic slightly reduce minor artifacts caused by loose peak connection. The value $k = 1$ was considered a good choice since it bound the maximum frequency deviation for a partial to about ± 21 Hz, that is a reasonable value for low frequency bins, while for higher a wrong peak connection does not introduce noticeable artifacts.

4.4.2 Phase locking based on sinusoidal models

As pointed out in [46], the instantaneous frequency, the phase, and amplitude of a sinusoid do not depend only on the measured values of the peak closer to the sinusoid frequency, but they depends in a complex way on the values of a certain number of neighboring bins. The relationships between sinusoid parameters and bins surrounding the peak, largely depend on the frequency response of the analysis window, and on the real position of the sinusoid between two frequency indexes.

Sinusoidal model analysis is performed estimating accurately frequency, phase and amplitude of detected sinusoids, and then synthesis is performed through a bank of oscillators (additive synthesis). To speed up the calculation, Rodet and Depalle [44] proposed to perform the additive synthesis in the STFT, by summing directly in the synthesis STFT frames the contribution of each tracked sinusoid.

Without needing to go deeply into details, the contribution of a sinusoid is given by its convolution in the domain transform with the analysis window. To perform this efficiently, its contribution is computed through a discrete approximation of the continuos time convolution of the sinusoid transform (once accurately estimated the parameters) with the main lobe of the analysis window.

Returning to the problem of phase dispersion, rigid phase locking preserve the complex relationships between bins surrounding a peaks, while standard phase vocoder algorithm and loose phase locking do not.

The scheme proposed here to preserve the relationship, is an admittedly primitive but computationally efficient approximation of the additive synthesis in the FFT. The steps are:

- compute the DFT X of the current frame, and search for peaks (local maximums);
- obtain X' subtracting from X the complex convolution between peaks and the analysis window main lobe;

- compute the phase unwrapping with the standard phase vocoder algorithm for X';
- compute the phase unwrapping for the peaks, with the peak connection heuristic explained in the previous section;
- compute the peaks contribution through the complex convolution between peaks (with updated phases) and the analysis window main lobe;
- obtain Y by adding to X' the peaks contribution;
- perform the IDFT on Y, and overlap and add the obtained synthesis frame.

Since the sinusoidal component is approximated with an impulse in the DFT, the complex convolution requires a number of complex multiplication equal to the main lobe width.

A more elegant way to compute the contribution of one sinusoid would be an approximation of the continuos convolution through look-up tables and interpolation.

This method leads to slightly worse results than scaled phase locking for small stretching factor, since no phase synchronization is guaranteed between partials and residual. For larger stretching factor, however, the quality seems higher than the one of scaled phase locking, since thanks to the additive synthesis no distortion occurs at low frequency due to bass smearing. This technique increases the workload of the time stretching algorithm, but not in a critical way since the FFT/IFFT computations dominate the phase vocoder complexity.

4.5 Stereo image preservation

In the previous chapter the concepts of vertical and horizontal coherence have been introduced. Unfortunately the phase vocoder suffers of lack of stereo coherence too. In fact there could be phase dispersion between bin placed in the same position of the time-frequency axis but with a different stereo position (left and right), causing an annoying floating effect also for very small stretching factors.

One action that must be taken in order to not introduce too much phase dispersion between channels is to perform the phase reset on both channels when preserving a transient. This action reduces the floating but its effect is non-dramatic since not every transient is detected, and floating can still occur in stable region of the sound signal.

Ravelli et al. [40] proposed to time-shift one of the two channel on every detected transient in order to maximize the cross correlation between channels, thus reducing the rapid panning caused by stereo phase dispersion. This method eliminates the rapid panning artifacts, but we would like to achieve a stereo image preservation also for stable regions. Furthermore this approach can lead to mask some stereo production effect, e.g. the doubling of a source whit a short delay between different channels.

Bonada [6] proposes a more robust and effective approach to stereo preservation. His idea is very simple: the synthesis phases should be adjusted bin per bin in order to keep the same stereo phase distance as in the analysis phases. This is achieved by adding to each bin phase a correction factor identical in both channels.

This approach is cheap and effective, and works perfectly within the constant frame rate model, but for standard phase vocoder it introduces minor artifacts in case of independent channels and large stretching factors. In fact, in some spectral regions of the signal the energy can be located mostly on one channel, and so it would be better to adjust the phase more on the channel where the energy is lower in order to make the effect of not preserving the instantaneous frequencies unappreciable.

Consider the following situation: for two consecutive STFT frame most of the signal energy is in the left channel and the right channel energy is below the level of perception. It follows that to keep the same stereo phase distance between analysis and synthesis frame without any impairment should be better to modify the phases only of the right channel. Furthermore, in a complex mix, for a bin whose magnitude is much higher on one channel, the phase adjustment should be performed more on the channel in which the magnitude is lower.

From this arguments it follows that phase adjustment should be performed by weighting the phase adjustment on the left-right bin magnitude ratio. Unfortunately this approach can lead to phase dispersion in both channels, and is less efficient than the method proposed in [6].

A simple and efficient way to keep the left-right channel phases relationship without introducing artifacts is performing the phase vocoder algorithm on a mono mix of both channels and then, during the synthesis, preserving bin per bin the analysis phase distance between the left/right channel and the mono mix (figure 4.5).

The k-th bin analysis phase distance between the left/right channel for the and the mono mix is:

$$\begin{aligned}\Delta\phi(k)_{left} &= \text{princarg}(\phi(k)_{left} - \phi(k)_{mono})\\ \Delta\phi(k)_{right} &= \text{princarg}(\phi(k)_{right} - \phi(k)_{mono}),\end{aligned} \tag{4.5}$$

where $\phi(k)$ is the analysis phase for the k-th bin. If ψ is the phase vocoder synthesis phase, the phase distance is preserved by adding back the analysis phase distance to the mono synthesis phase on each channel:

$$\begin{aligned}\psi(k)_{left} &= \psi(k)_{mono} + \Delta\phi(k)_{left}\\ \psi(k)_{right} &= \psi(k)_{mono} + \Delta\phi(k)_{right}\end{aligned} \tag{4.6}$$

Since the Fourier Transform is a linear operator, the mono mix STFT can be computed efficiently by summing the STFTs of the left and right channel. This technique reduces the complexity of the algorithm in case of stereo signal since:

- the transient detection function can be computed only in the mono STFT;
- the phase vocoder phase unwrapping is performed only on the mono STFT (keeping the same phase distance between bins is much cheaper than phase unwrapping).

Phase locking should be performed directly on the mono mix, because it can lead to loose both the stereo and horizontal phase coherence if performed independently on the two channels.

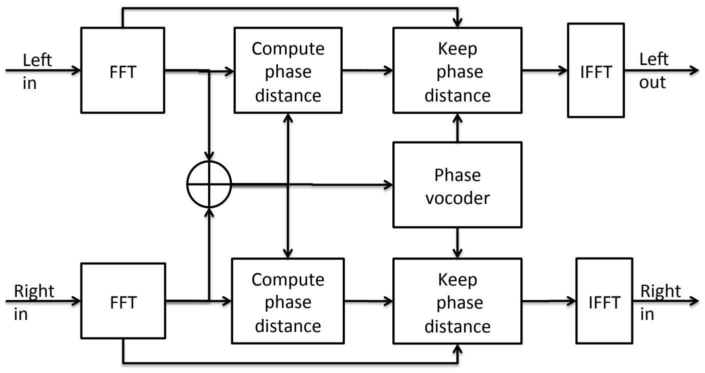


Figure 4.5: *Stereo image preservation.*

4.6 Real-time sample accuracy

One key feature in time-scaling context is definitely timing accuracy. A real-time procedure to reach sample accuracy [1] is needed because the synthesis hop size is an integer number while αR_a is not, and during transient preservation $R_s = R_a$ is imposed for some synthesis frame.

The simplest way to achieve this time accuracy is dynamically changing the synthesis hop size in order to compensate the timing error E. If the fractional synthesis hop size is:

$$R_i = \alpha Ra, \tag{4.7}$$

and the integer synthesis hop size is:

$$R_s = \lfloor R_i + 0.5 \rfloor \tag{4.8}$$

it follows that for each synthesis step a local error $e = R_i - R_s$ is generated (when $R_s = R_a$ is imposed the local error becomes $e = R_i - R_a$). Then for each synthesis step it is possible to accumulate the local error by adding it to E ($E = E + e$).

When $|E| > 1$ the error can be compensated by shifting the output frame by $\lfloor |E| \rfloor$ samples, imposing $R_s = R_s + \text{sign}(E)\lfloor |E| \rfloor$. In fact, the maximum shifting must be bounded in order to guarantee a minimum overlap between adjacent synthesis frames.

[1]Sample accuracy means that if the input has a length of n samples, the time-scaled output must have a length of $\alpha n \pm 1$ samples.

4.7 Window envelope compensation

The overlap and add procedure causes an annoying amplitude modulation effect if the overlap and add of the product between the synthesis and analysis window is not a constant.[2] In the case of the Hanning window, no window ripple are present if the overlap and add hop size is a constant integer multiple of $n/4$. In case of truncated gaussian window the effect of modulation is generally considered negligible for constant hop size.

In any case, because in this phase vocoder model the synthesis hop size is not constant, the window envelope must be compensated real-time. The cost of a direct compensation of the window envelope it is negligible, since in any case the output needs to be normalized.

An easy way to perform this compensation real-time is storing real-time a dummy signal that represents the window envelope, and, just before output the time-scaled signal divide it sample per sample by the window envelope.

4.8 Real-time implementation

As frequently stated, the fundamental motivation behind the implementation of a time stretching algorithm is the need of a time stretching procedure that does not introduce latency or computational peaks when it starts.

Without the need of going deeply into boring buffering/implementation detail, we can assume that the output produced by the time stretching is read in block of B samples and the input is fed with n samples (n analysis window length) as soon as are needed. Another assumption is that the time stretching function can be split in four equally time-consuming sub-routine :

- compute the FFTs of one input frame;
- perform the transient detection;
- compute the phase-vocoder phase increment along with transient preservation and phase locking;
- compute the IFFTs of one frame and output it.

When the time-stretching *function* is called only one of this sub-routine is performed.

Under this assumptions, it follows that, if R_s is the synthesis hop size, the time stretching routine has to be performed every $\frac{R_s}{B}$ samples,[3] and that means the function has to be called every $\frac{R_s}{4B}$ samples.

[2]In fact the shape of analysis window is modified by the phase vocoder algorithm, as pointed out in [38]. To perfectly recover the effect of the analysis window special procedure needs to be added in the compensation model, but here we will assume that the modification of the analysis window shape is negligible. For further information see [35].

[3]For sake of simplicity we can assume $Fs = 1$ and measure the time in system samples.

The time stretching routine, in order to be able to detect and preserve transient, needs to perform the analysis in advance of four[4] frames: three for the transient synchronization, one for the delayed detection (see section 4.3.2).

To perform this analysis in advance without increasing the workload of the system, the initial synthesis hop size doubled. Then, for the first five frames, only the input FFT is computed, so the workload is 7/4 larger than the normal workload of the time stretching routine (5 sub-routine for the analysis in advance plus 2 for the second frame synthesis). Furthermore, because the synthesis hop size of the first frame is doubled, there is absolutely no need to call the time stretching function more time than in a "normal" situation, ideally avoiding the introduction of a computational peak. In practice the assumption that the four sub-routine are equally time consuming is violated since FFTs and IFFTs sub-routine are heavier than the other two, but the computational peak is very low.

The effect of doubling the initial hop size is compensated real-time by the technique described in section 4.6, and transient detection starts with a delay of 11 frames (7 transient function value are needed to compute the dynamic threshold).

In case the input/output sample rate is different from 44.1kHz, in order to not increase the workload[5] of the time stretching routine, it is adapted by a very fast sample rate converter (figure 4.6).

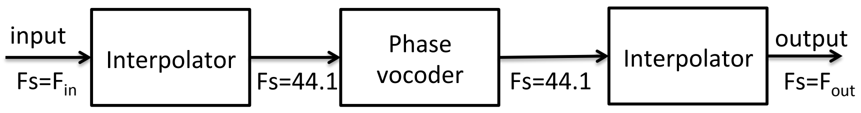


Figure 4.6: *Sample rate conversion.*

4.8.1 Performances

The algorithm has been implemented in C++, and extensively tested within the Reactable Software, with different computers. Since for the moment Reactable Live! only support Mac Osx operating system and Intel processors, the code has been optimized for the target systems.

The implementation is quite performant: it runs approximatively at a minimum of $23\times$ real-time when the stretching factor is 0.8 [6] on a 2007 MacBook Pro (3.1 model), equipped with a 2.2 GHz Intel Core Duo processor [7].

[4]When the overlap is four the temporal synchronization parameters described in 4.3.2 are match, in other cases some synchronization accuracy is sacrificed in order to not introduce computational peaks in the system.

[5]It is very unlikely in the context of electronic music to use sample rates below 44.1kHz.

[6]Real-time performances are measured with respect to the output, so it follows performances are inversely proportional to the stretching factor. Since performances are also inversely proportional to the overlap factor, to save computation the analysis hop size increase when the stretching factor is lower than 0.8.

[7]The performances are measured performing the time stretching on one core

The FFTs/IFFTs consume the larger part of the demanded computational time, and are performed by a sub-optimal MIT licensed library. Performances could be increased by choosing a more performant FFT library, but for the Reactable implementation it was highly sufficient.

4.9 Subjective listening tests

4.9.1 Formal listening test

To assign a quality to the proposed algorithm, and also to compare it with two commercial products, a formal subjective listening test was performed. In particular the proposed algorithm (with the rigid phase locking variant) was tested against Elastique Pro and Dirac[8]. The test performed is the MOS-LQS (Mean Opinion Score Listening Quality Subjective [49]).

A total of 16 subjects were asked to listen carefully to 12 short audio files and three relatives stretched versions obtained with the three stretching algorithms, and then to express an opinion on the quality of each time-stretched version. In particular listeners had to mark a score accordingly to the listening quality scale (table 4.1)[9]. The score for each algorithm is determined by the arithmetical mean of the opinions.

Table 4.1: *Listening quality scale*

Score	Quality
9-10	Excellent
7-8	Good
5-6	Fair
3-4	Poor
1-2	Bad

The 12 source file set and relative stretching factors were chosen accordingly to the quality requirements stated in section 1.2.3. In particular 3 of the 12 sources are textures, 3 drum beats, 3 simple mixes and 3 complex mixes. For each kind of source, one stretched version was produced with a small stretching factor (110/115%), one with a moderate stretching factor (125/130%) and one with a large stretching factor (150/200%). Only stretching factor greater than 1 were used for the test, since artifacts are more annoying and perceivable in case of time-expansion.

Results can be clustered on two features: one on the stretching factor (three sub-sets) and one on the kind of source (four sub-sets).

The listener set can be partitioned in two sub-sets: expert listeners and non expert listeners. In fact it was asked to listeners to answer some questions about their knowledge and experience

[8]Dirac time stretching were performed with the Prosoniq's Time Factory 2 default settings, except the quality one that was set to the higher value (quality best)

[9]In fact in the standard [49] the scale goes from 1 to 5, but it can be modified in order to allow intermediate values

in musical field and digital audio processing, in order to partition listener sets in different groups. The only statistically relevant difference in the scores was found for listeners with a large accumulated experience in the field of digital audio processing. The scores are listed in tables 4.2 and 4.3.

Table 4.2: *Listening test results (all listeners)*

Category	Proposed algorithm	Elastique Pro	DIRAC
All	**7.84** ± **1.46**	7.75 ± 1.35	7.14 ± 1.74
Small stretching	**7.94** ± **1.5**	7**.94** ± **1.29**	7.5 ± 1.67
Medium stretching	7.8 ± 1.43	**7.95** ± **1.14**	6.7 ± 1.72
Large stretching	**7.8** ± **1.46**	7.37 ± 1.53	7.15 ± 1.73
Textures	7.75 ± 1.41	**7.77** ± **1.34**	6.9 ± 1.77
Beats	7.85 ± 1.61	**8.08** ± **1.37**	6.98 ± 1.91
Simple mixes	**7.88** ± **1.47**	7.79 ±1.35	7.33 ± 1.71
Complex mixes	**7.90** ± **1.36**	7.37 ± 1.25	7.37 ± 1.49

Table 4.3: *Listening test results (experts only)*

Category	Proposed algorithm	Elastique Pro	DIRAC
All	**7.65** ± **1.45**	7.13 ± 1.32	6.08 ± 2.12
Small stretching	**8.06** ± **1.25**	7.63 ± 1.05	6.63 ± 2.06
Medium stretching	**7.69** ± **1.36**	7.31 ± 1.16	5.75 ± 2.02
Large stretching	**7.8** ± **1.59**	7.38 ± 1.41	5.88 ± 2.18
Textures	**7.58** ± **1.44**	7.33 ± 1.34	6.08 ± 1.77
Beats	7.08 ± 2.06	**7.17** ± **1.62**	5.33 ± 2.17
Simple mixes	**8.08** ± **0.64**	7.33 ± 1.03	6.25 ± 1.71
Complex mixes	**7.83** ± **1.36**	6.67 ± 1.25	6.67 ± 1.49

4.9.2 Results discussion

Listening test results are very promising. In fact the proposed time-stretching algorithm scored better than two of the most used commercial time stretching libraries.

Surprisingly, both from experts and non-experts the proposed algorithm seems to outperform Dirac and Elastique on mixes, while for texture and beats the score is about the same; furthermore the score is higher for mixes than for textures and beats. This could be explained by the fact that the sub-band partition of the spectrum for transient preservation gives optimal results in case of polyphonic music, but when there is only one source playing it would be better to apply the procedure on the whole spectrum.

We can state that the perceived stretching quality on most sources is similar to the one of Elastique, and the score differences can be considered within statistical noise.[10] Dirac scores are lower.[11] In fact Dirac is a great time stretching algorithm, which does not introduce any phasiness, but it creates some annoying artifacts when reconstructing some attack transient.

Score differences between the algorithms are, as expected, higher for experts than for non-expert. What is really interesting is that the score variance is very high for both experts and non-expert. This proof that the "annoyance" of introduced artifacts is very subjective and source dependent. For example, as will be discussed later, the proposed algorithm tends to smear transients located at high frequencies (e.g. hi-hat shot). Elastique, on the contrary preserve very well high frequencies attack transient, but introduce a perceivable pre-eco artifact. What emerged from the test results is that some user did not find particularly annoying neither the smearing neither the pre-eco, some of them found the smearing annoying while not the pre-eco and vice-versa.

4.9.3 Informal listening tests

Along with the formal listening test, some time stretching examples have been submitted to audio-DSP experts in order to have some feedback. What emerged is that, as expected, the weakness of the proposed technique is resolution. In fact, for medium/large stretching factor there is still some smearing for transients located at high frequencies; for large stretching factor the lack of resolution at low frequencies ends up in a poor quality synthesis for some sources (in particular for acoustical instrument in which low frequencies partials are very close, e.g. a double bass).

Anyway, the transient preservation was reviewed positively by the large part of reviews and work-in-progress comparison tests submitted to Reactable colleagues showed that the effort to develop a time stretching library was worthy.

4.10 Conclusion and next steps

In this work a simply and efficient transient preservation technique for the phase vocoder has been presented. This technique can be implemented without introducing latency or computational peaks in a real time system, at a relatively low computational cost.

A real-time implementation of this technique was tested against two of the most famous and widely used time-stretching libraries achieving optimal results, while another is now included in the *Reactable Live!* software.

The proposed technique can be included in different phase vocoder models, or can be tested with different transient detection techniques. However, since the idea behind this work is the

[10]Student's two-tailed t-test $p = 0.384$.
[11]Student's two-tailed t-test $p = 1.8e^{-6}$.

development of a fast and efficient time stretching algorithm, the future work should go in the direction of keeping the workload low and to not introduce latency or pre-processing routines.

For this purpose, the first issue that should be studied is the development of a multi-resolution mechanism that does not rise too much the workload of the time-stretching. This could be achieved by the scheme proposed in figure 4.7:

- the input signal is split in high frequencies and low frequencies components by two linear phase filters;
- the low frequency component is down-sampled of a factor two, in order to gain some resolution without increasing the size of the analysis/synthesis window;
- the high frequency component analysis/synthesis is performed through a smaller windows in order to improve the timing resolution and, at the same time, save computation.
- just before the overlap and add procedure the low frequencies phase vocoder synthesis must be up-sampled by a factor two and low-pass filtered.

This scheme can be implemented efficiently with a polyphase realization of the linear phase filters, at the cost of four more half-size FFT for frame (since the complexity of FFT is logarithmic and practical factors, as memory access and caching, play an important role for the FFT workload, the increase in cost should be "low").

What needs to be explored in this scheme is:

- the optimal cut frequency F_c for the low-pass and high-pass filters;
- a real-time transient synchronization technique between high frequencies and low frequencies components.

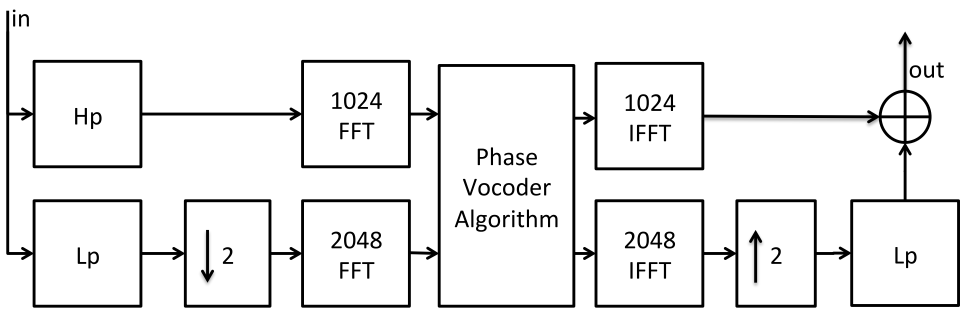


Figure 4.7: *Proposed multi-resolution phase vocoder.*

Preliminary test showed that applying some dynamic reduction to the time-stretched signal helps in reducing the perceived phasiness and transient smearing for large stretching factor. In fact the loss of vertical coherence between two close partials can still produce beating, modulating the perceived volume in a determined spectral region (see figure 4.8). Dynamic reduction (compression/exapansion, for further information see [15]) seems to reduce this problem.

Furthermore, applying some compression to the stretched signal leads to slightly lower the volume of signal portions just after prominent transients, partially masking the effect of time-scaling non stable portions of signals. Before including dynamic reduction in the time stretching technique what should be investigated is:

- what is the best technique to include (parallel compression, multi-band compression, envelope follower, multi-band expander etc...);
- what signal should modulate the dynamic reduction (the input or the output);
- what is the right amount of dynamic reduction to introduce in the system in order to reduce the phasiness without impairing too much the dynamic of the sound.

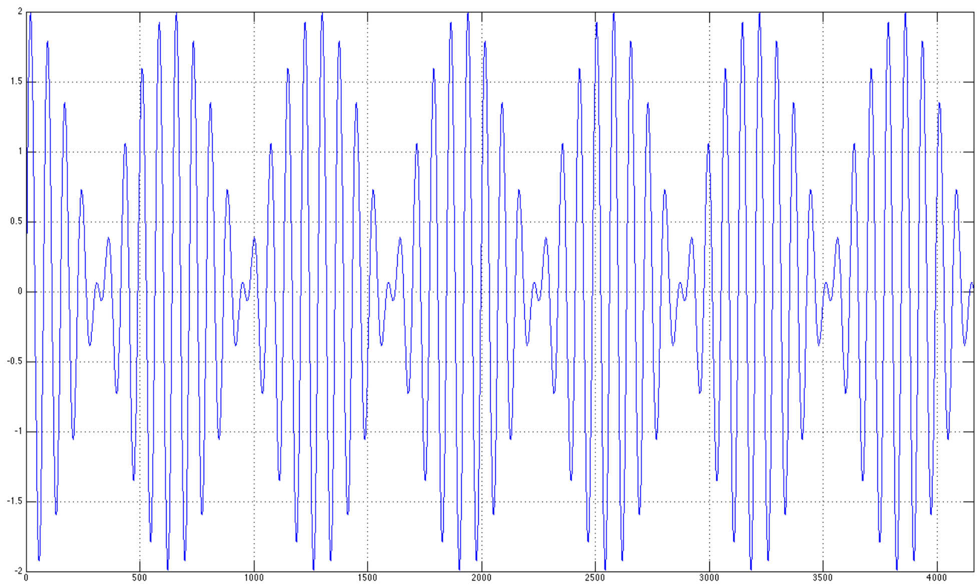


Figure 4.8: *Beating produced by the sum of two sinusoids at* 400 *and* 460 *Hz.*

Another procedure to reduce the beatings problem, that from a preliminary analysis seems promising, is to introduce some randomness in the phase unwrapping equation. Beating is a periodic event, and by introducing some randomness in the sinusoids phases it can be turn into a non-periodic event, avoiding its recognition by the auditory system. What needs to be investigated is:

- what is the maximum randomness that can be added to phase unwrapping without causing perceivable artifacts;
- what is the relationship between frequency index and maximum allowed randomness;
- whether it is better to introduce the randomness for all bins or only for peaks.

From an applicative point of view, it would be interesting and useful to integrate an high quality pitch shifter in this scheme. In fact, a rudimental pitch shifter is already implemented in the Reactable software, that basically consists in applying the variable speed replay to the output of the time stretching.

This method leads to good results for small pitch shifting, while for larger it introduces the famous *mickey mouse* [7] artifact, because not only the partials representing the fundamental frequency of the sources are equally pitch shifted, but also the formants.[12] Keeping the same magnitude for the time-scaled formants leads to alter the timbre of the source, since the frequency response of the original source is not preserved.

A good approach to be tested is to directly shift the bins in the STFT ([27]), taking into account that formants magnitude should be multiplied by some transfer function that represent the frequency response of the original source. This can be achieved with some effort in the case of monophonic sources, because the frequency response of the original source can be evaluated by analyzing the spectral envelope of the input. In case of polyphonic/multi-source signal the problem is definitely much more complex, and would probably require to go deeply into sinusoidal models.

[12]A formant in this case indicates the sinusoidal components of a sound produced by the acoustic resonance of the instrument.

Bibliography

[1] D. Arfib, F. Keiler, U. Zölzer, V. Verfaille, and J. Bonada. *DAFX: Digital Audio Effects*, chapter 7. John Wiley and Sons, 2011.

[2] F. Auger and P. Flandrin. Improving the readability of time-frequency and time-scale representations by the reassignment method. *IEEE Transactions on Signal Processing*, 43, 1995.

[3] J. P. Bello, C. Duxbury, and M. Davies. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 11(6), 2004.

[4] P. Bello and M. Sandler. Phase-based note onset detection for music signals. *Proceedings of IEEE International Conference on Audio, Speech and Signal Processing (ICASSP)*, 2003.

[5] J. Bonada. Automatic technique in frequency domain for near-lossless time-scale modification of audio. *Proceedings of International Computer Music Conference (ICMC)*, 2000.

[6] J. Bonada. Audio time-scale modification in the context of professional audio post-production. *Research work for PhD Program*, 2002.

[7] J. Bonada, X. Serra, X. Amatriain, and A. Loscos. *DAFX: Digital Audio Effects*, chapter 10. John Wiley and Sons, 2011.

[8] F. J. Charpentier and M. Stella. Diphone synthesis using an overlap-addtechnique for speech waveforms concatenation. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1986.

[9] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 1965.

[10] A. de Cheveigne and H. Kawahara. Yin, a fundamental frequency estimator for speech and musica. *Journal of the Acoustical Society of America*, 2002.

[11] A. De Götzen, N. Bernardini, and D. Arfib. Traditional (?) implementations of a phase-vocoder: the tricks of the trade. *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, 2000.

[12] P. Depale, X. Rodet, and G. Gracia. Tracking of partials for additive sound synthesis using hidden markov models. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1993.

[13] M. Dolson. System for fourier transform modification of audio. *Creative Technology Ltd.*, US 6112169, 2000.

[14] D. Dorran and R. Lawlor. An efficient audio time-scale modification algorithm for use in a subband implementation. In *Proceedings of the 6th Int. Conference on Digital Audio Effects (DAFX-03)*, 2003.

[15] P. Dutilleux, K. Dempwolf, M. Holters, and U. Zölzer. *DAFX: Digital Audio Effects*, chapter 4. John Wiley and Sons, 2011.

[16] P. Dutilleux, G. D. Poli, A. von dem Knesebeck, and U. Zölzer. *DAFX: Digital Audio Effects*, chapter 6. John Wiley and Sons, 2nd edition, 2011.

[17] C. Duxbury. *Signal Models for Polyphonic Music*. PhD thesis, 2004.

[18] J. L. Flanagan and R. M. Golden. Phase vocoder. *Bell System Technical Journal*, 1966.

[19] S. Grofit and Y. Lavner. Time-scale modification of audio signals using enhanced wsola with management of transients. *IEEE Transactions On Audio, Speech, and Language Processing*, 16(1), January 2008.

[20] K. N. Hamdy, A. H. Tewfik, T. Chen, and S. Takagi. Time-scale modification of audio signals with combined harmonic and wavelet representations. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1997.

[21] C. Hamon, B. Moulines, and F. Charpentier. A diphone synthesis system based on time domain prosodic modifications of speech. *Speech Communication*, 9, 1990.

[22] A. Harti. Discrete multi-resolutionanalysis and generalized wavelets. *Applied Numerical Mathematics*, 12, 1993.

[23] D. Hejna and B. R. Musicus. The solafs time-scale modification algorithm. *Technical Report, BBN*, 1991.

[24] T. Karrer, E. Lee, and J. Borchers. Phavorit: A phase vocoder for real-time interactive time-stretching. *Proceedings of International Computer Music Conference (ICMC)*, 2006.

[25] J. Laroche and M. Dolson. Phase-vocoder: about this phasiness business. *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (ASSP)*, 1997.

[26] J. Laroche and M. Dolson. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing*, 7(3), 1999.

[27] J. Laroche and M. Dolson. New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects. *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (ASSP)*, 1999.

[28] lsmo Kauppinen. Methods for detecting impulsive noise in speech and audio signals. *Proceedings of International Conference on Digital Signal Processing*, 2002.

[29] S. Marcus and J. Hoek. Method and apparatus for signal processing for time-scale and/or pitch modification of audio signals. *Sigma Audio Research Limited*, US 6266003, 1999.

[30] P. Masri and A. Bateman. Improved modelling of attack transients in music analysis-resynthesis. *Proceedings of the International Computer Music Conference (ICMC)*, 2000.

[31] R. J. McAulay and T. F. Quatieri. Magnitude-only reconstruction using a sinusoidal speech model. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1984.

[32] R. J. McAulay and T. F. Quatieri. Shape invariant time-scale and pitch modification of speech. *IEEE Transactions on Signal Processing*, 40, 1992.

[33] S. K. Mitra. *Digital Signal Processing: A Computer-Based Approach*. McGraw Hill, 4th edition, 2011.

[34] A. Moinet and T. Dutoit. Pvsola: A phase vocoder with synchronized overlap-add. *Proceedings of the 14th Int. Conference on Digital Audio Effects (DAFx-11)*, 2011.

[35] E. Moulines and J. Laroche. Non-parametric techniques for pitch-scale and time-scale modification of speech. *Speech Communication*, 1995.

[36] G. Peeters and X. Rodet. Sinola: A new analysis/synthesis method using spectrum peak shape distortion, phase and reassigned spectrum. *Proceedings of International Computer Music Conference*, 1999.

[37] M. R. Portnoff. Implementation of the digital phase vocoder using the fast fourier transform. *IEEE Transactions on Audio, Speech, and Language Processing*, 1976.

[38] M. Puckette. Phase-locked vocoder. *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (ASSP)*, 1995.

[39] T. F. Quatieri, R. B. Dunn, and T. E. Hanna. A subband approach to time-scale expansion of complex acoustic signals. *IEEE Transactions on Speech and Audio Processing*, 3(6), 1995.

[40] E. Ravelli, M. Sandler, and J. P. Bello. Fast implementation for non-linear time-scaling of stereo signals. In *Proceedings of the 8th Int. Conference on Digital Audio Effects (DAFx-05)*, 2005.

[41] A. Röbel. A new approach to transient processing in the phase vocoder. *Proceedings of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, 2003.

[42] A. Röbel. Transient detection and preservation in the phase vocoder. *Proceedings of International Computer Music Conference (ICMC)*, 2003.

[43] A. Röbel. A shape-invariant phase vocoder for speech transformation. *Proceedings of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, 2010.

[44] X. Rodet and P. Depalle. Spectral envelopes and inverse fft synthesis. *AES conference*, 1992.

[45] A. Sakurai and S. Trautman. Phase locking method for frequency domain time scale modification based on a bark-scale spectral partition. *Texas Instruments Incorporated*, US 8019598 B2, 2011.

[46] X. Serra. *A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition*. PhD thesis, Stanford University, 1989.

[47] J. O. Smith. *Spectral Audio Signal Processing*. W3K publishing, 2011.

[48] D. Terruggi. Technology and musique concre'te: the technical developments of the groupe de recherches musicales and their implication in musical composition. *Organised Sound*, 12, 2007.

[49] I. T. Union. Mean opinion score (mos) terminology. *ITU-T Rec. P.800.1*, 2003.

[50] V. Verfaille, D. Arfib, F. Keiler, A. von dem Knesebeck, and U. Zölzer. *DAFX: Digital Audio Effects*, chapter 9. John Wiley and Sons, 2011.

[51] W. Verhelstand and M. Roelands. An overlap-add technique based on waveform similarity (wsola) for high quality time-scalemodification of speech. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1993.

[52] T. S. Verma and T. H. Meng. An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998.

[53] M. Zaunschirm, J. D. Reiss, and A. Klapuri. A sub-band approach to modification of musical transients. *Computer Music Journal*, 2012.

[54] E. Zwicker. Subdivision of the audible frequency range into critical bands. *Journal of the Acoustical Society of America*, 3(2):248–248, 1961.

[55] E. Zwicker and H. Fasti. *Psycho-acoustics*, chapter 6. Springer, 2nd edition, 1999.