

# Time stretching en temps réel pour le live coding

Abdeslam El-Haman Abdeslam<sup>1</sup>

Superviseur : Bernard Fortz

<sup>1</sup>Université Libre de Bruxelles

aelhaman@ulb.ac.be

## Abstract

Overtone est une librairie en Clojure qui est utilisée pour faire du Live Coding (l'art de programmer en « vif »). Une des techniques les plus utilisées dans le domaine de la musique synthétisée est le time-stretching, qui consiste à rallonger ou rétrécir une pièce musicale sans changer sa tonalité. Le time-stretching est intéressant dans le live-coding lorsqu'on peut modifier les paramètres de celui-ci en temps réel. Dans cet article 2 méthodes de time-stretching avec des approches différentes seront analysées, comparées et utilisées en temps réel dans Overtone.

## Introduction

Depuis le débuts de la musique électronique, le “resampling” de pièces musicales (c'est à dire, la manipulation de celles-ci) a un rôle important pour pouvoir manipuler des pièces musicales (rajouter des filtres, des enveloppes, etc ...) [rajouter citation ici].

Un son est un signal qui est défini par sa durée, sa fréquence et son amplitude. La fréquence définit son “pitch” ou tonalité. La tonalité d'un son est plus grave si la fréquence est plus petite (donc une période plus grande).

La manipulation d'un son dans le temps est utilisée très souvent lors du mixage et DJing. Ce genre de manipulations aident à rétrécir ou prolonger cette pièce pour, par exemple, mettre un son à la même vitesse que le “tempo” d'une chanson.

Cette manipulation aura comme résultat aussi un effet secondaire. La prolongation du signal provoquera aussi que la fréquence de ce signal soit modifiée, et avec ça, un changement de tonalité non désiré se produira.

Le time-stretching est une technique pour éradiquer ce problème : changer le tempo d'un son sans changer sa tonalité.

Plusieurs algorithmes existent pour implementer cette technique. Dans cet article l'état de l'art du time-stretching sera étudié et plusieurs de ces algorithmes seront implémentés pour l'application en temps réel de cette technique sur Overtone, une librairie pour qui a été conçue pour le traitement et synthèse de son.

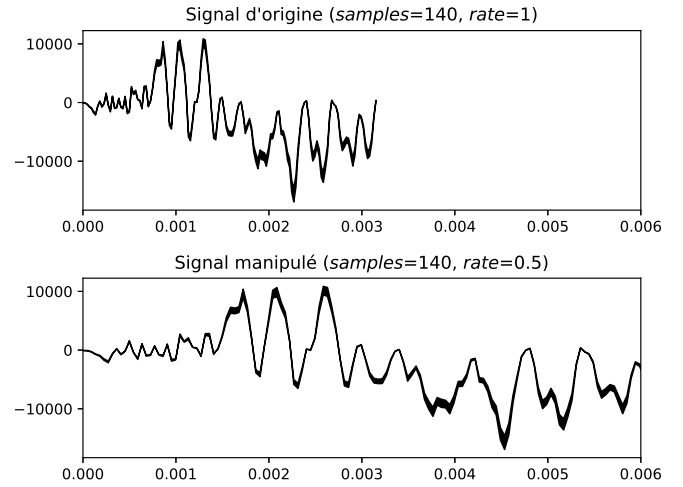


FIGURE 1 – Sample d'une guitare lu à des vitesses différents

## État de l'art

### Méthode OLA

En général la méthode utilisée dans les algorithmes TSM se résume en plusieurs étapes distinctes.

**Décomposer le signal d'entrée en grains** Le signal d'entrée est décomposé en *grains* [citer article] d'une taille relativement petite et fixe. Chacun de ces grains a un décalage *hopsiz*  $H_a$  (hopsiz d'analyse). On peut représenter ça tel que :

$$x_m(r) = x(r + mH_a) : r \in [-N/2 : N/2 - 1]$$

où  $x$  est notre signal d'entrée en format discret de taille  $L \in \mathbb{Z}_{\geq 0}$ .  $N$  est la taille du grain  $x_m$ .

La taille de ces grains doit être généralement d'une taille entre 50ms et 100ms. Ceci est important pour trouver le *pitch local* dans l'intervalle  $[mH_a - N/2 : mH_a + N/2 + 1]$ . Si l'intervalle est grand, plusieurs tonalités apparaissent dans cet intervalle, donc ce grain n'est pas représentatif.

**Modifier le hopsize** Une fois le signal décomposé, avec l'information de pitch dans chaque grain, on va définir un nouveau hopsize  $H_s$  qui va servir à récomposer ces grains. Ce  $H_s$  est défini tel que :

$$H_s = \alpha H_a$$

Dans lequel  $\alpha$  représente le *facteur d'échelle*. L'effet du TSM est fixé par le *facteur d'échelle*. Pour un effet de rétrécissement :

$$\alpha < 1$$

Au contraire, pour un effet d'élargissement :

$$\alpha > 1$$

**Superposition de grains** Avec  $H_s$  défini, on peut reconstruire le signal résultat de cette modification, en récomposant le nouveau signal avec les grains avec un décalage de  $H_s$  au lieu de  $H_a$ .

## Méthode W/SOLA

### Méthode vocodeur de phases

### TD-PSOLA

### Traitement des fréquences

### Traitement du temps et des fréquences

Des techniques SOLA peuvent déjà donner des

## Méthodes

Les 2 méthodes qui ont été développées sont la granulation et le vocodeur de phases. Ces 2 méthodes ont été utilisées en utilisant la fonction le système de génération de synthétiseurs "defsynth" d'Overtone.

## Langages fonctionnels

### Overtone

### AWESOME OVERTONE TWEAK THINGS

### SOLA approach / granulator

[Expliquer ce que c'est un granulator]

Un granulator a été implémenté sur Overtone en utilisant les UGENs suivants de SuperCollider :

### Vocodeur de phases

## Résultats

## Remerciements

Je remercie mon promoteur, Bernard Fortz, de m'avoir fait découvrir les yeux au monde du live-coding et la programmation fonctionnelle.

Je tiens aussi à remercier l'UrLab pour m'avoir accueilli au SmartMonday pour partager avec eux tout ce que j'ai appris lors de mes recherches pour ce mémoire.

[Remerciements à Antoine]

## References

Driedger, J. and Müller, M. (2016). A review of time-scale modification of music signals. *Applied Sciences*, 6(2).