Time stretching en temps réel dans le cadre du live coding

Abdeselam El-Haman Abdeselam[1]
Superviseur : Bernard Fortz

[1]Université Libre de Bruxelles
aelhaman@ulb.ac.be

Abstract

Overtone est une libraire 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 paramétres 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 éléctronique, 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 envelopes, 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 sécondaire. La prolongation du signal provoquera aussi que la fréquence de ce signal soit modifié, 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.

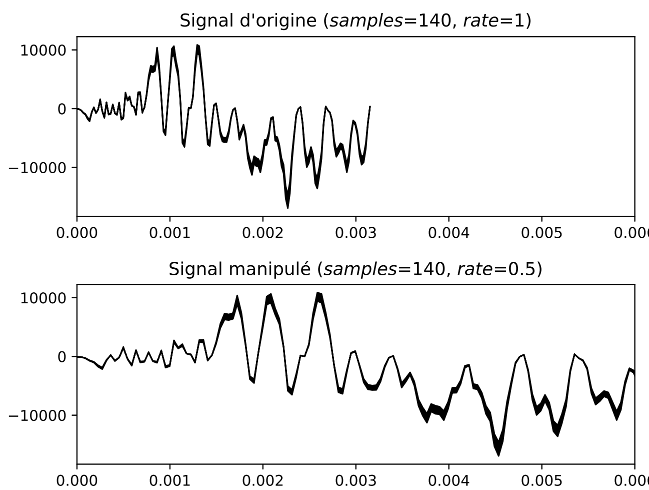

FIGURE 1 – Sample d'une guitare lu à des vitesses différents

État de l'art

Méthode OLA

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

Décomposer le signal d'entrée en grains Le signal d'entrée est décomposé en *grains* [citer article] d'une taille rélativement petite et fixe. Chacun de ces ces grains sa un décalage *hopsize* H_a (hopsize 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 doît est généralement d'une taille entre 50ms et 100ms. Ceci est important pour trouver le *pitch local* dans l'intervale $[mH_a - N/2 : mH_a + N/2 + 1]$. Si l'intervale est grand, plusieures tonalités apparaissent dans cet intervale, 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 α 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 réconstruire 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. Alors ces grains sont superposés avec une différence de H_s et additionés. C'est à dire :

$$y(t) = \sum_{m \in \mathbb{Z}} x_m(t - mH_s)$$

où $y(t)$ est le signal de sortie par rapport au temps t. u

Fenêtrage Hormis le fait que cette méthode est utilisée comme base des algorithmes qui seront illustrés plus tard dans ce document, elle crée beaucoup d'*artéfacts* à cause d'un déphasage entre les grains qui provoque des discontinuités qui se traduit par des sons lourds et courts qui n'étaient pas dans le signal initial. Ceci est dû à cause d'un déphasage important entre les grains puisque $H_s \neq H_a$.

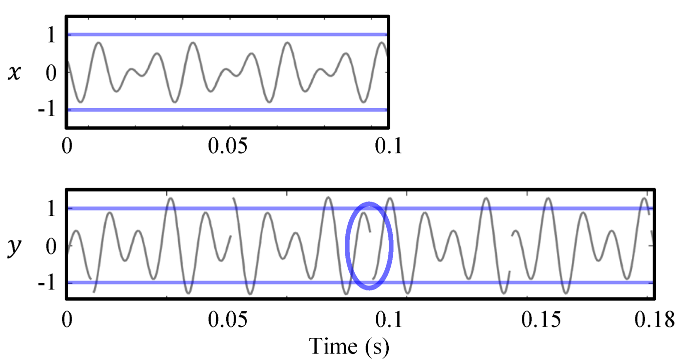

FIGURE 2 – Exemple d'artefact qui se produit quand le même grain d'analyse est utilisé pour reconstruire le signal avec le tempo modifié (Driedger and Müller (2016)).

L'algorithme *OLA* se base sur cette méthode basique et rajoute des fénêtres pour avoir une fluidité dans la transition d'un grain à un autre. Les grains x_m sont donc appliqués à une fonction de fênetrage w qui est définie :

$$w(r) = \frac{1}{2}\left(1 - cos\left(\frac{2\pi(r + N/2)}{N-1}\right)\right)$$

Où w est la *fenêtre de Hanning*.
Cette fenêtre a la propriété que

$$\sum_{n \in \mathbb{Z}} w\left(r - n\frac{N}{2}\right) = 1$$

qui rassure la continuité d'amplitude dans le signal résultant.

Méthode WSOLA

Le problème de la méthode OLA est que, étant le hopsize H_a fixe, chaque séparation entre grains est le même, peu importe la différence de phase quand les grains sont décalés et réassemblés par rapport au hopsize H_s.

L'algorithme *WSOLA* (Verhelst and Roelands (1993)) régle ce problème en rajoutant un marge de décalage Δ_m au hopsize H_a afin qu'entre les grains x_m et x_{m+1} il y ait un minimum de discontinuités.

Ce décalage Δ_m est défini étant la position dans laquelle les parties superposées des grains x_m et x_{m+1} se ressemblent au plus. Pour ceci nous devons définit Δ_{max} étant le maximum que la valeur de Δ_m doit avoir. Alors $\Delta_m \in [-\Delta_{max} : \Delta_{max}]$. Nous avons alors le nouveau grain de synthèse

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

qui va être superposé en suivant le procédé *OLA* pour réassembler les grains décrit avant.

Choix du décalage Δ_m est choisi en trouvant la meilleure resultat si on applique une fonction de corrélation entre les parties des grains qui se superposent. Plusieures formules peuvent être choisies pour trouver une corrélation entre 2 variables (une regréssion linéaire, etc ...). On peut utiliser le coefficient corrélation croisée à ce fin :

$$c_c(m,\delta) = \sum_{n=0}^{N-1} (n+\tau^{-1}((m-1)N)+\Delta_{m-1}+N).x(n+\tau^{-1}(mx_m)+\delta)$$

avec $max(m,\delta) \rightarrow \delta = \Delta_m$ et τ notre fonction de TSM.

Méthode vocodeur de phases

Méthode développée

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

OLA approach / granulator

[Expliquer ce que c'est un granulator]

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

Résultats

Remerciements

Je remercie mon prometeur, 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).

Verhelst, W. and Roelands, M. (1993). An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech. In *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing : Speech Processing - Volume II*, ICASSP'93, pages 554–557, Washington, DC, USA. IEEE Computer Society.