Effet autotune en temps réel

Alexis DAVID, Aymane KHATTABI

January 2024

Introduction

Dans ce projet nous allons mettre en oeuvre un effet d'autotune par synthèse additive. Nous voulons une implémentation temps réel. Pour cela nous allons travailler en C++ avec l'API RTaudio.

1 Analyse du signal

1.1 Extraction de la fréquence fondamentale

Nous voulons tout d'abord connaître la fréquence fondamentale du signal car c'est elle qui donne la hauteur de la note et elle nous donnera une information sur la position des harmoniques à extraire par la suite.

Nous savons qu'un signal périodique a une auto-corrélation périodique et que la période est donnée par l'écartement entre deux maxima de l'auto-corrélation. De plus le premier maximum est en zéro car c'est le signal multiplié par lui même sans décalage. Il nous suffira donc de connaître la position du deuxième maximum pour avoir la période et donc la fréquence.

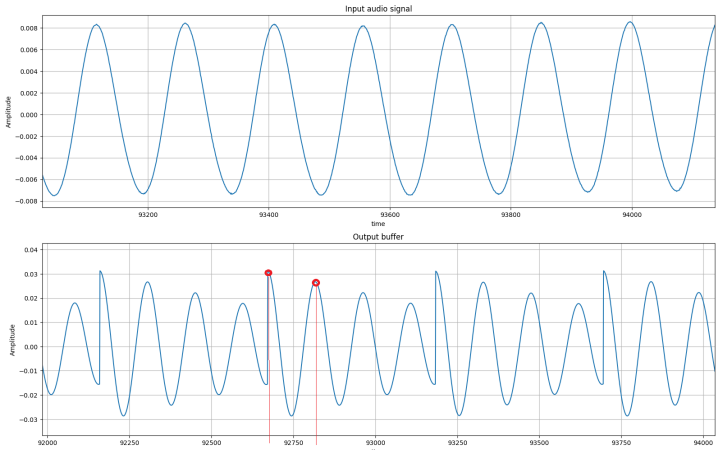


Figure 1: Signal temporel (haut) et auto-corrélation

Sur la courbe du haut de la figure 1 on voit le signal d'entrée à 300 Hz que nous avons envoyé dans le micro de l'ordinateur. Au dessous, on voit l'auto-corrélation qui passe par un maximum à chaque cassure car il s'agit du début d'un batch. Nous mesurons l'écart entre ce maximum et le suivant. En divisant la fréquence d'échantillonnage par cet écart on obtient : $\frac{f_s}{\delta_t} = \frac{44100}{145} = 305Hz$

Sur la figure 2 on affiche la fréquence détectée au cours du temps. Nous avons envoyé un signal à 300 Hz devant le micro et on peut voir que la fondamentale est bien détectée.

La fréquence minimale que l'on peut détecter sans buffer circulaire est :
$f_{min} = \frac{fs}{2*BufferFrames} = \frac{44100}{2*512} = 43Hz.$

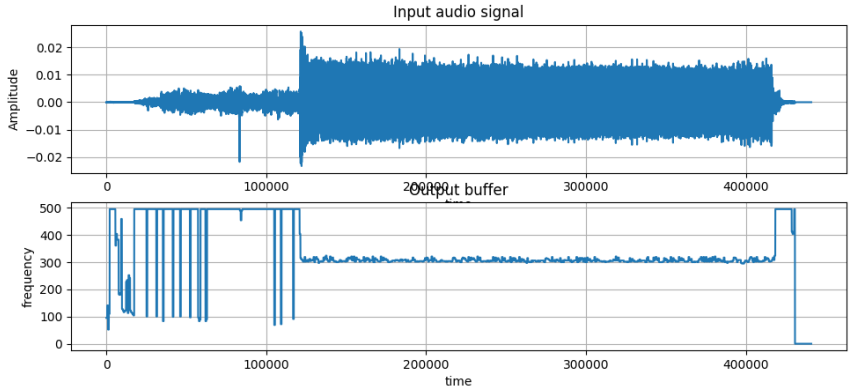


Figure 2: Détection de fréquence à 300 Hz (bas) et signal temporel en entrée (haut)

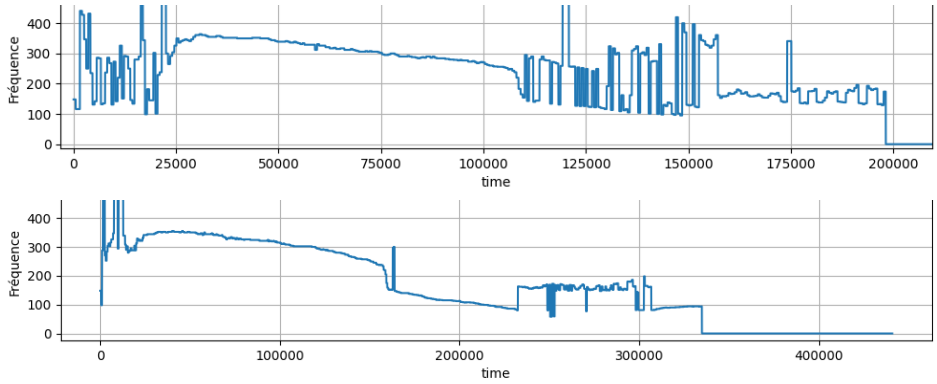


Figure 3: Détection de f_0 avec un buffer simple (haut) et un buffer circulaire de taille 4*512 (bas)

Pour faire des analyses plus fines nous avons implémenté un buffer circulaire. Cela permet de faire une auto-corrélation sur un signal plus long donc qui peut contenir de plus grandes périodes. Cependant, comme nous nous intéressons à des voix humaines il nous a semblé raisonnable de ne considérer que les fréquence allant de 50 Hz à 500 Hz. On voit sur la figure 3 une comparaison entre une analyse avec un buffer de taille 512 et un buffer circulaire de taille 4*512. Pour les hautes fréquences il n'y a pas de grandes différence. En revanche pour les basses fréquences la détection de f_0 est un peu améliorée. On peut noter que la voix d'Alexis passe d'une voix de "tête" à une voix de "poitrine" autour de 200 Hz. Et nous avons remarqué que c'est à cause des vibrations de la voix de poitrine que l'analyse devient plus difficile.

1.2 Extraction des harmoniques

Pour la synthèse du son nous aurons besoin des harmoniques. Ce sont elles qui donneront le timbre de la voix. Pour les obtenir nous devons extraire les valeurs des amplitudes situé tous les multiples de la fréquence fondamentale f_0.
Ainsi pour la k-ième harmonique il faudra prendre le point $n_k = \frac{f_0 * N}{f_s}$
où N est le nombre de points pour la transformée de fourrier discrète, f_0 la fréquence fondamentale et f_s la fréquence d'échantillonnage. Pour N nous avons pris une la valeur correspondant à la taille du buffer circulaire.

La correction d'amplitude à apporter est un facteur multiplicatif de $1/N_{fft}$. Où N_{fft} est notre taille de fft.

2 Synthèse du signal

2.1 Première synthèse sans la phase

Dans cette première synthèse on dispose des amplitudes a_k des 10 harmoniques considérées, de la fréquence fondamentale f_0 et de la fréquence d'échantillonnage f_s. Le signal S reconstruit est :

$$S(i) = \sum_k a_k cos(2\pi f_0 i/f_s) \tag{1}$$

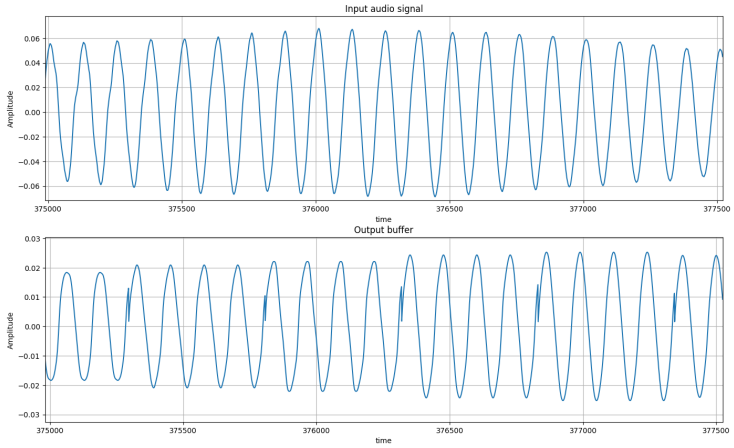


Figure 4: Signal original (haut) et reconstruction sans la phase (bas)

En moyennant sur 18 périodes et en sachant que la fréquence d'échantillonnage f_s est 44100 Hz. On détermine sur la figure 5:

- Fréquence fondamentale du signal d'origine : $\frac{f_s}{377385-375132} * 18 = 352Hz$
- Fréquence fondamentale perçue du signal reconstruit : $\frac{f_s}{377372-375061} * 18 = 344Hz$

La fréquence perçue est inférieure car la reconstruction trame par trame n'est pas toujours en phase. On voit en effet des cassures régulières dans le signal temporel reconstruit. Il y a donc a chaque buffer un décalage qui allonge la durée moyenne des périodes et donc diminue la fréquence fondamentale.

2.2 Ajout de la phase

Pour éviter l'effet évoqué au dessus et pour diminuer la distorsion, nous allons prendre en compte la phase de la synthèse. Ainsi le signal S reconstruit est :

$$S(i) = \sum_k a_k cos(2\pi f_0 i/f_s + \phi(k)) \tag{2}$$

Pour mettre à jour les valeurs de phase correspondant à chaque harmonique nous devons utiliser la fréquence fondamentale du buffer précédant, noté f_{prev}. Le décalage en phase correspondant à un buffer de taille n est donc ajouté comme ceci :

$$\phi(k) = \phi(k) + 2\pi k f_{prev} n/fs \tag{3}$$

On trouve cette fois-ci des valeurs beaucoup plus proches pour les fréquences fondamentales :

- Signal originale : $f_0 = 226$

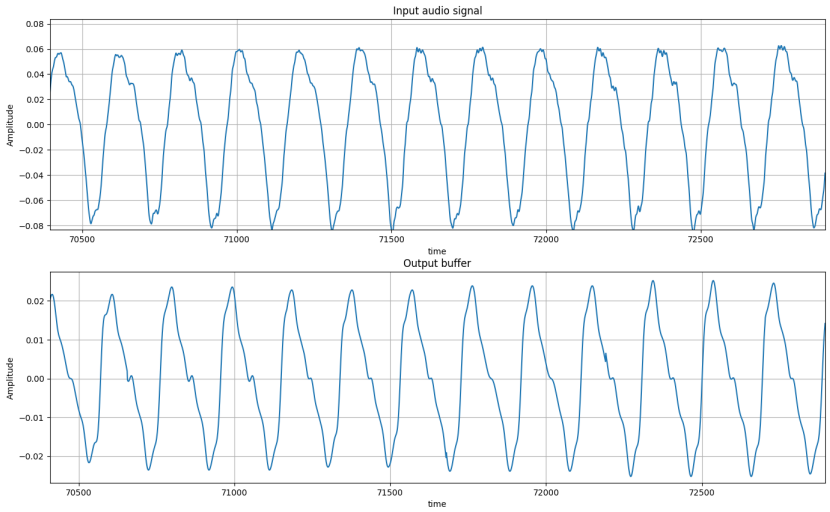


Figure 5: Signal original (haut) et reconstruction avec la phase (bas)

- Signal reconstruit avec la phase : $f_0 = 228$

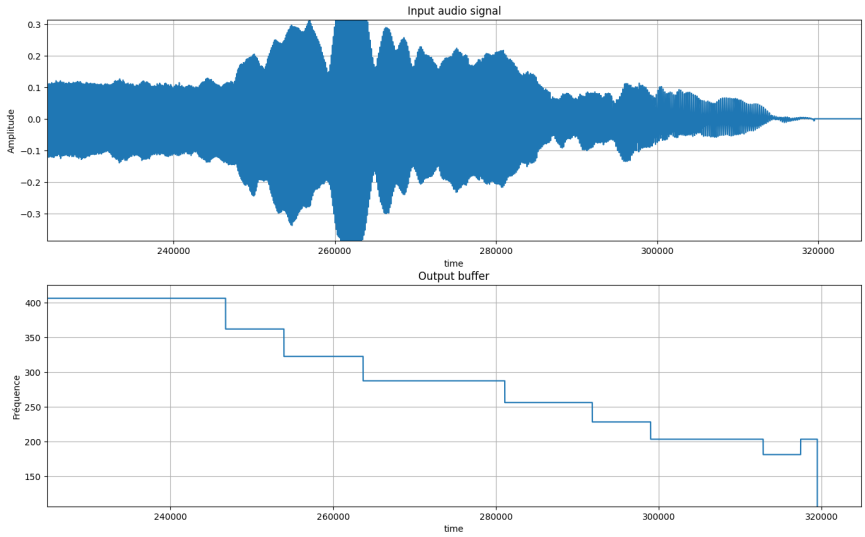
La fréquence perçue avec le signal reconstruit sur 11 périodes est maintenant légèrement plus haute que l'originale mais ceci est due aux imprécisions de mesure.

On note aussi que cette fois-ci les transition entre les trames sont bien meilleures même si parfois perceptibles en zoomant.

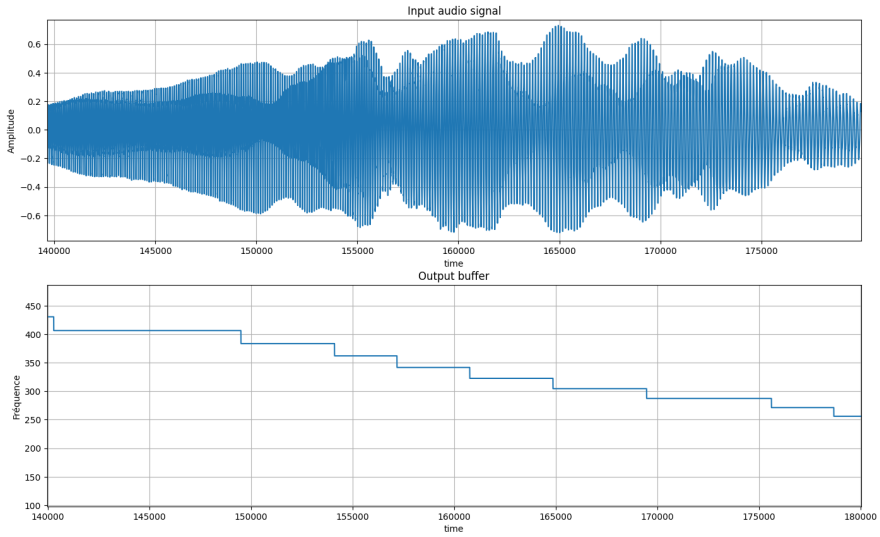
3 Effet auto-tune

Maintenant, nous allons modifier notre f_0 juste avant la synthèse pour changer la hauteur des harmoniques mais par leurs amplitudes qui ont été extraites avec le 'vrai' f_0.

Le principe de faire chanter juste est de changer f_0 pour lui donner une valeur arrondie correspondant à une note présente dans la gamme chromatique. Pour cela on divise la fréquence en 12 demi-tons en échelle logarithmique. Nous essayons aussi avec 4 et 6 intervalles pour avoir un effet plus perceptible à l'écoute. On voit sur la figure 6 que la fréquence fondamentale ne prend que certaines valeurs et il y a 2 fois plus de paliers dans la descente de fréquence avec 12 intervalles que avec 6 où les sauts sont plus grands.



(a) 6 intervalles



(b) 12 intervalles

Figure 6: Fréquences de signaux autotunés pour deux nombres d'intervalles