Grenoble INP

Projet Traitement du Signal en temps réel

Author :
Mona DHIFLAOUI
Paul AIMÉ

Supervisor :
M. THOMAS HUEBER

15 janvier 2020

1 Introduction

1.1 Présentation

On se propose dans ce projet d'étudier les problématiques du traitement du signal en temps réel à partir du cas d'étude de l'implémentation en `C++` d'un effet *reverb* à convolution, avec usage de la technique de l'*overlap-add* donc. L'idée est d'appliquer au signal audio acquis en temps réel un filtre d'ambiance d'une salle créant de la réverbération. Cela est réalisé par la convolution du signal avec la réponse impulsionnelle de la salle, visible sur la figure 1.

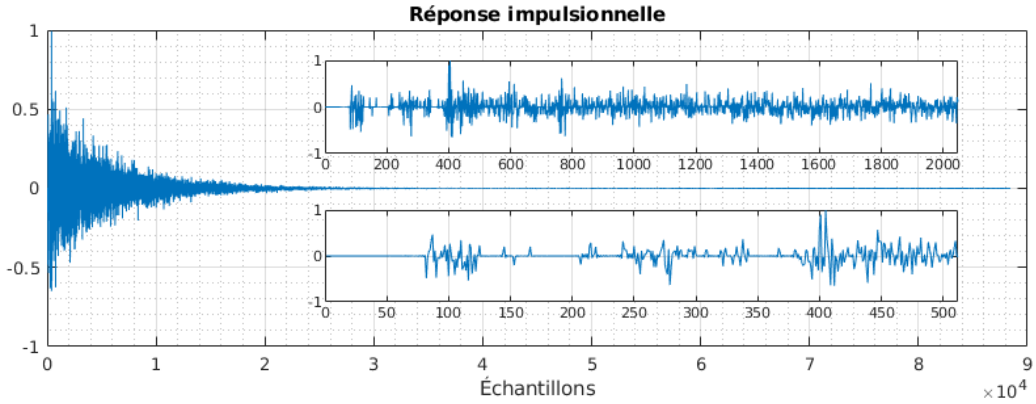

FIGURE 1 – Réponse impulsionnelle du filtre utilisé. 88431 échantillons au total. Zooms sur l'intervalle $[0, 2048]$ (*en haut*) et $[0, 512]$ (*en bas*).

On observe que la réponse impulsionnelle a une forme exponentielle décroissante, avec des échantillons d'amplitude quasiment nuls à partir de l'échantillon 30000, relativement à l'amplitude du début de la réponse impulsionnelle. On en suppose alors que les premiers échantillons ont plus d'influence que les suivants. On notera M la taille de la partie utilisée de la réponse impulsionnelle.

1.2 Caractéristiques machine

Nous avons réalisé tous nos essais sous un système Linux (Ubuntu), avec un CPU intel i5, en utilisant l'API RtAudio avec Alsa. Le nombre de buffers de la carte son est géré automatiquement par l'API et est de 4, sauf à partir d'une taille de buffer de 4096 (2^{12}) et au delà, où le nombre de buffer passe à 2. La fréquence d'échantillonnnage utilisée est de 44100 Hz.

2 Implémentation

Nous avons implémenté la convolution en temps et en fréquence, et pour les deux, nous utilisons de la même manière deux buffers pour réaliser l'*overlapp-add*, avec un buffer de taille L+M-1 pour stocker le résultat de la convolution avec le buffer d'entrée courant (taille L), et un autre buffer de taille M-1 pour enregistrer la partie de cette convolution qui se situe temporellement en dehors du buffer de sortie (également taille L) et qui doit donc être ajoutée aux résultats des convolutions avec les futurs buffer d'entrée.

Pour la mesure du temps d'exécution, on enregistre dans une structure l'horloge du CPU à chaque fin d'appel de la fonction *callback*. Cette mesure prend donc en compte le temps d'exécution de *callback* mais aussi le temps entre la fin du *callback* et l'entrée suivante dans le *callback* : notamment la durée du chargement du buffer d'entrée (communication entre la carte son et la machine via les drivers). La

fonction de mesure du temps de traitement rajoute deux appels de fonctions, et peut donc légèrement modifier le temps qu'elle mesure.

2.1 Convolution en temps

Pour la convolution en temps, on se sert uniquement des deux buffers mentionnés précédemment, puisqu'on peut mettre le résultat de la convolution courante directement dans le buffer de taille L+M-1 prévu à cet effet. On a confirmé son implémentation en comparant sa sortie à celle de la fonction implémentée d'origine dans MATLAB.

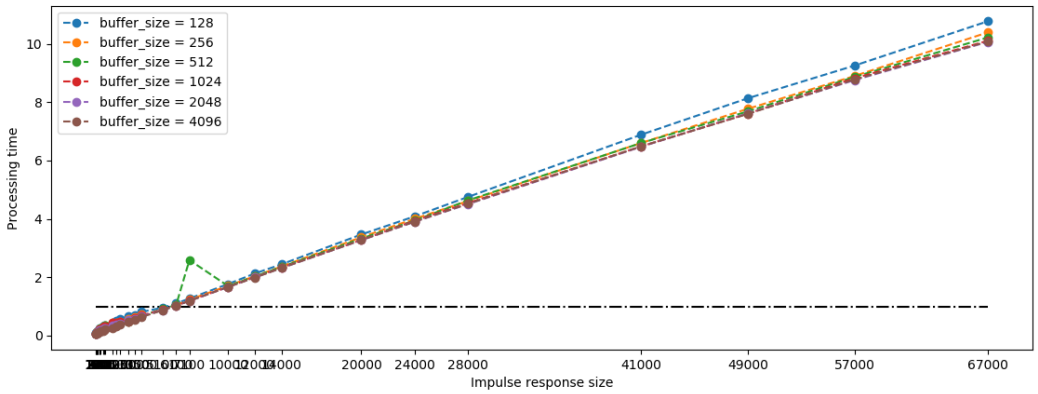

FIGURE 2 – Évolution du temps de traitement normalisé d'un buffer, en fonction des paramètres de taille du buffer audio et de taille utilisée de la réponse impulsionnelle, pour la convolution implémentée en temps.

Pour assurer un traitement du signal en temps rééel, il faut que la durée de traitement d'un buffer soit inférieure à la durée que représente le buffer – dans le cas contraire on parle d'*overrun*. Pour quantifier l'impact sur l'*overrun* des deux paramètres principaux que sont la taille du buffer d'entrée (L) et la taille utilisée de la réponse impulsionnelle (M), on trace l'évolution du temps de traitement moyen d'un buffer en fonction des ces paramètres (figure 2). Pour que les mesures soient comparables, on divise le temps de traitement mesuré par la durée représentée par un buffer. Il y a donc *overrun* lorsque cette valeur dépasse 1, représentée en pointillés noirs sur la figure 2.

Pour la convolution en temps, on obtient déjà un effet correct pour des petites valeurs de M (à partir de 100). Cependant, on observe une saturation du signal de sortie avant même l'atteinte de la limite théorique d'*overrun*, alors même que la réponse impulsionnelle a été normalisée pour éviter cela. Par conséquent, la ligne d'*overrun* présente sur la figure 2 est au delà de la limite acceptable d'utilisation. De plus la limite théorique mesurée de l'*overrun* arrive assez bas, vers M=5000, et la taille du buffer audio n'a que peu d'effet. Cependant, à l'écoute on observe que dans une situation temps réel (pas d'*overrun* ressenti), l'effet de *reverb* est d'autant plus agréable que la taille du buffer augmente.

2.2 Convolution en fréquence

Pour la convolution en fréquence, nous rajoutons 6 buffers de la taille de la fenêtre de FFT (soit la puissance de 2 suivant le nombre L+M-1). 2 sont juste remplis une seule fois car ils sont dédiés au stockage de la partie réelle et de la partie imaginaire de la FFT de la réponse impulsionnelle. 2 sont dédiés au stockage du calcul de la partie réelle et imaginaire du buffer d'entrée. Finalement, les 2 derniers sont dédiés au stockage de la partie réelle (Yr) et imaginaire (Yi) du résultat de la multiplication des

deux transformées de Fourier (TF). Le dernier buffer Yr est réutilisé pour stocker la TF inverse de $(Yr + j \times Yi)$, avant de n'en copier que la taille souhaitée (L+M-1) dans le buffer de stockage de la convolution en cours.

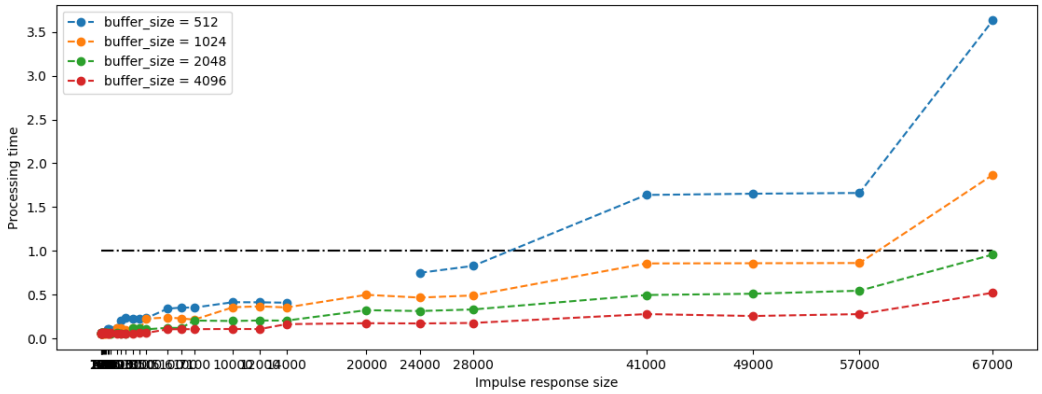

FIGURE 3 – Figure équivalente à la figure 2, pour la convolution implémentée dans le domaine des fréquences.

On répète la mesure décrite dans la section 2.1, et on la présente dans la figure 3. On observe que contrairement au cas de la convolution en temps, cette fois l'évolution du temps de traitement n'est pas linéaire mais présente des paliers. Cela est dû au fait qu'en passant dans le domaine de Fourier par la FFT on doit réaliser les calculs sur des vecteurs de taille puissances de 2. Ces paliers sont donc situés de part et d'autre de ces puissances (3 points ont été placés dans chaque intervalle). On se rend alors compte que diminuer M n'est alors ici pas toujours suffisant, et donc pas nécessaire, pour augmenter la vitesse de traitement.

À l'écoute, l'effet n'est pas agréable pour des M trop petits ($M < 5000$), mais contrairement à l'implémentation en temps il n'y a pas de saturation avec l'augmentation de M. Également, puisque cette implémentation est beaucoup plus rapide, elle autorise d'utiliser beaucoup plus d'échantillons pour la réponse impulsionnelle. De plus, contrairement au cas en temps, ici la taille du buffer audio a une grande influence sur la durée normalisée de traitement : augmenter la taille du buffer audio permet d'utiliser plus d'échantillons de la réponse impulsionelle sans atteindre la limite d'*overrun*.

3 Conclusion

Dans ce projet nous avons pu étudier, grâce à l'implémentation de la convolution en temps réel, les problématiques du traitement du signal en temps réel, et en identifier les paramètres majeurs en l'état de la taille du buffer de la carte audio et le temps d'exécution de la fonction appelée sur les données de ce buffer.

On a pu constater que la connaissance de son implémentation est fondamentale pour maîtriser le temps d'exécution, puisque suivant celle-ci, les paramètres étudiés n'ont pas la même influence sur le temps d'exécution. On a pu le voir notamment avec le fait que la taille du buffer audio a plus d'influence dans le cas de l'implémentation en fréquence qu'en temps, et également avec le fait que dans l'implémentation en fréquence le paramètre de taille de la réponse impulsionnelle n'a pas forcément une influence intuitive, ce qui pourrait entraîner une simple perte de performance en pensant réaliser une optimisation par la diminution M.

Note : Le code du projet est disponible sur GitHub, et est laissé à la libre utilisation.