UE MU4RBI08-audio
Compte-rendu de TP 3 : reconstruction du signal audio : TFCT inverse

---

NEHMAR Anis,TOUZARI Lisa
Master Automatique,Robotique, Parcours: ISI

Introduction

L'objectif de ce tp est d'implémenter l'algorithme de Transformée de Fourier à Court-Terme Inverse **itfct**. La fonction **itfct** doit prendre comme paramètre la **tfct** du signal pas d'avancement de la fenêtre d'analyseNhop, ainsi que lenombre de points fréquentiels utilisés dans l'algorithme TFCTN f f tet la fréquence d'échantillonnage F_s du signal initial. On verra par la suite une application de la **itfct** pour le débruitage par soustraction spectrale

Partie 1 : Implémentation de la TFCT inverse par Overlap-Add (OLA)

1. Préallocation de mémoire pour le signal reconstruit y.

   Pour faire une allocation de mémoire pour le signal reconstruit y :
   - On détermine le nombre de trames du signal d'entrée qui est égal au nombre de colonnes de sa matrice tfct.
   - En exploitant la valeur de y_l, on peut alors déterminer le nombre d'échantillons du signal d'entrée.

2. Reconstruction de chaque trame y_l à partir de sa TFD

   A partir de la matrice tfct du signal d'entrée, on reconstruit le signal y_l. Chaque ligne de la matrice y_l reçoit une colonne de la matrice tfct du signal d'entrée décalée avec l*y_l, l étant le numéro de la ligne de la matrice y_l.

3. Somme des trames reconstruites y_l à la bonne position de y.

   Pour cette étape, il suffira de faire une somme de toutes les trames y_l décalées.

4. Normalisation de y

   Pour le calcul de la tfct du signal d'entrée, on a utilisé une fenêtre de Hamming de largeur supposée égale à N_{fft}. Pour normaliser notre signal reconstruit, on doit calculer la constante k qui est égale à la somme des échantillons de la fenêtre divisée par son pas d'avancement N_{fft} . Il suffira alors de diviser le signal reconstruit y par k.

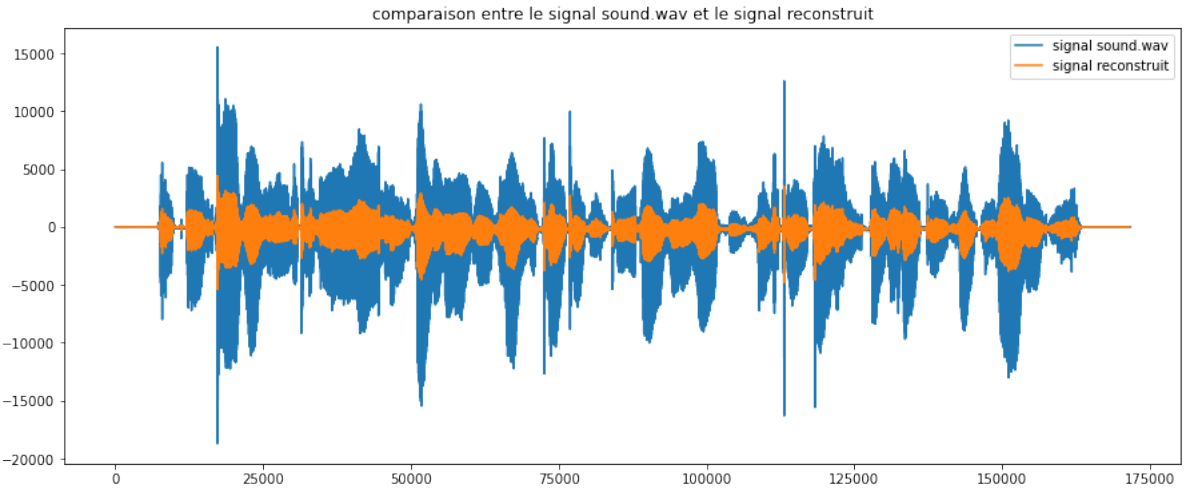
5. Test de l'algorithme sur la matrice x_{mat} calculée par l'algorithme **tfct.py**, appliqué aufichier audio **sound.wav**

   (a) Calcul de la matrice tfct du signal **sound.wav**

   Pour le calcul de la tfct, on choisi un N_{fft}=N_{win}= 1600, et un N_{hop}= 400, on a choisi la valeur de N_{hop} petite par rapport à N_{win}, car plus cette valeur est petite, plus on diminue l'effet de bord et le son reconstruit sera de meilleure qualité.

   (b) reconstruction du signal **sound.wav** avec la fonction **itfct**

On reconstruit le signal **sound.wav** en appliquant la itfct sur la matrice tfct calculé précedement, et on écoute le signal résultant. On remarque alors qu'on a le même son.



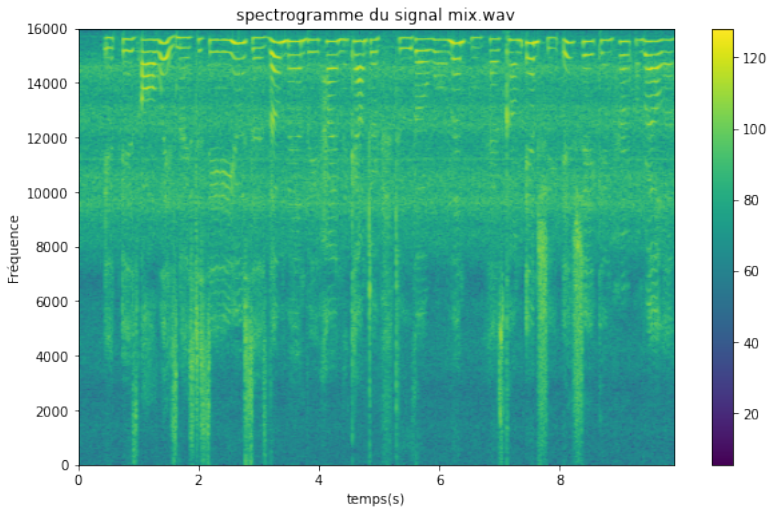
(c) calcul de l'erreur quadratique

On normalise les valeurs du signal d'entrée et celui reconstruit (on soustrait la moyenne et on divise par l'écart type du signal chaque échantillon), on obtient une erreur égale à 46%.
On remarque que malgré une erreur assez grande, le son écouté est le même que le signal d'origine.

Partie 2 : Application au débruitage pour soustraction spectrale

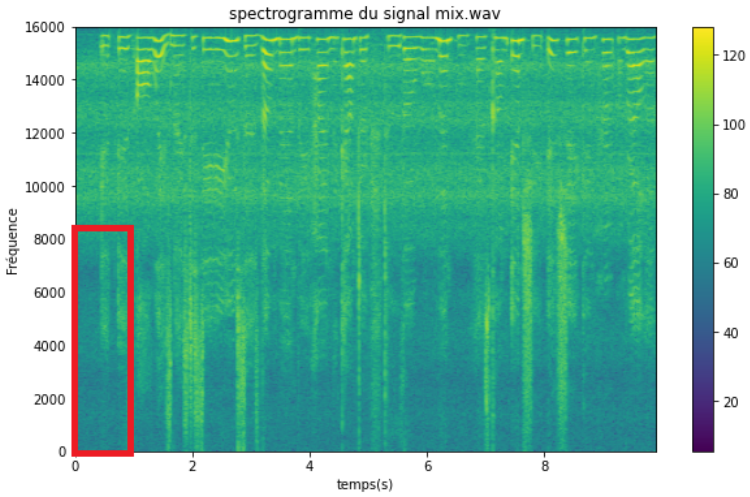
1. Calcul de la tfct du signal audio bruité **mix.wav**

On applique la fonction **tfct** sur le signal **mix.wav**, et on visualise le spectrogramme.

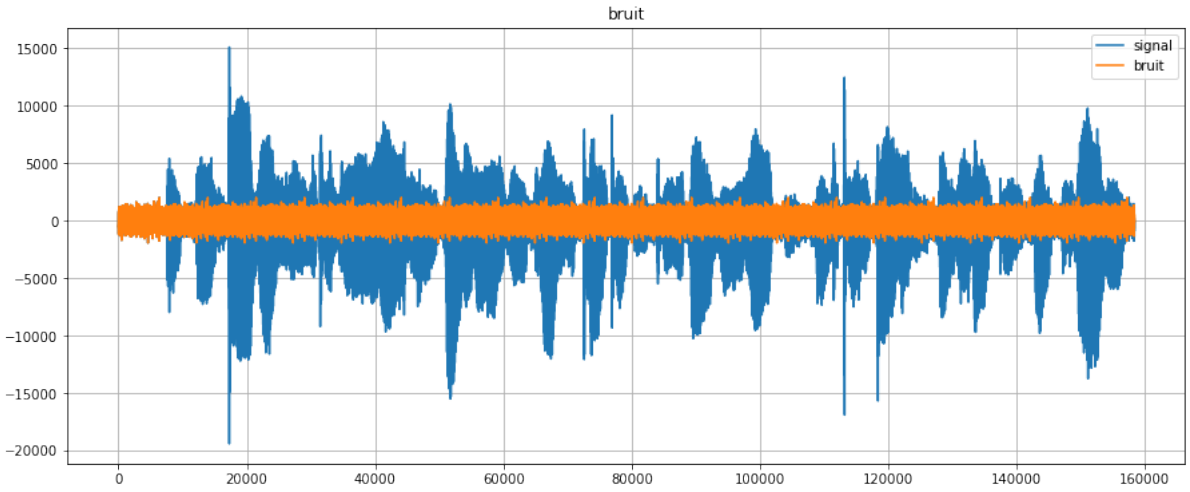


2. Tracer et commenter le spectre du bruit

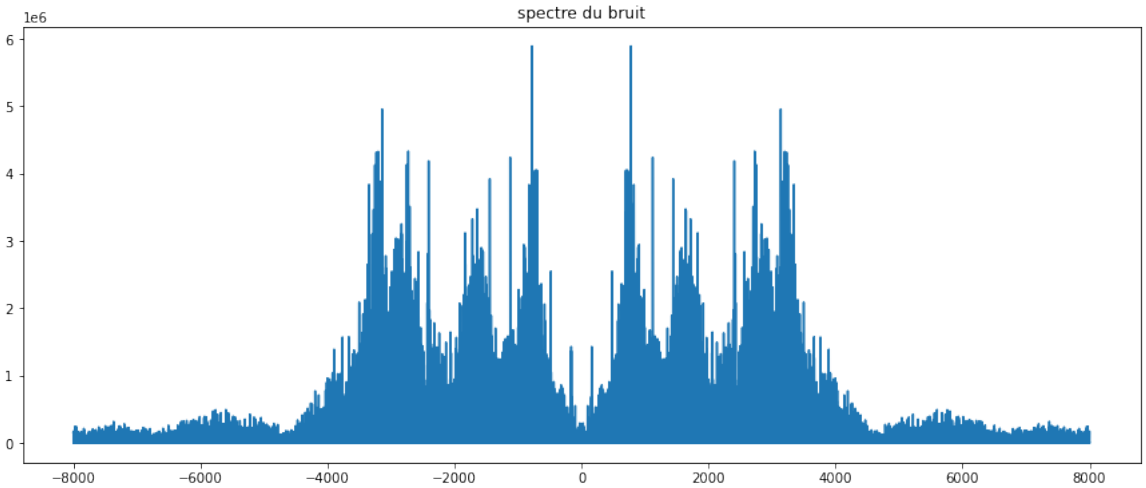
On peut voir sur le spectrogramme du signal **mix.wav** plusieurs zones de bruit, et notamment au début du signal.



En identifiant la zone de bruit, on extrait cette partie du signal et on construit le signal bruit qui a la même longuer, c'est à dire, le même nombre d'échantillons que le signal **mix.wav**.



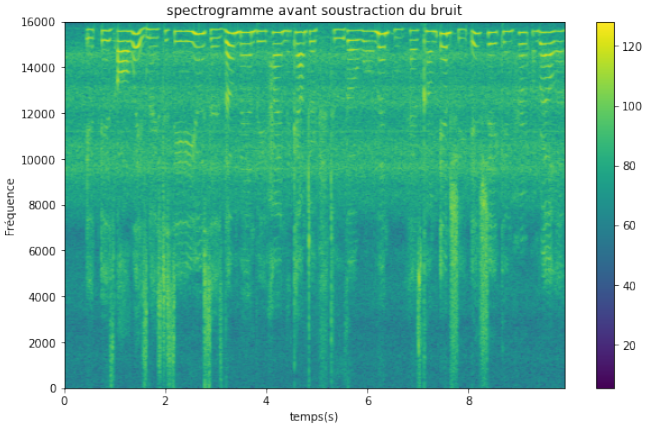
On trace alors le spectre du bruit.

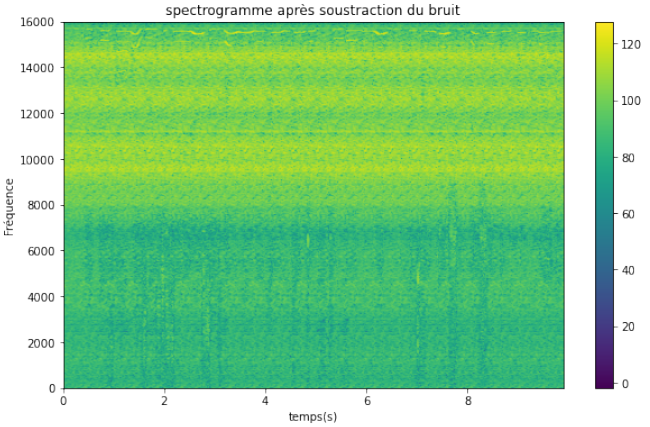


D'après le spectre obtenu, on voit que le bruit est constitué de plusieurs fréquences d'amplitudes différentes, donc on peut en déduire que notre bruit n'est pas un bruit blanc (la densité spectrale de puissance n'est la même pour toutes les fréquences de la bande

passante).

3. Estimation du spectrogramme du signal débruité $|X_{clean}|$





On remarque qu'après la soustraction du spectre du bruit, les zones de silence (zones bleues) sur le spectrogramme ont disparues.

4. le redressement du signal débruité

Il faut veiller à redresser les valeurs négatives du signal debruité à zéro, pour traiter le cas où le module du bruit est supérieur au module du signal, afin d'éliminer complétement le bruit.

5. Reconstruction du signal audio débruité à partir de $|X_{clean}|$

On reconstruit le signal à partir de $|x_{clean}|$, on constate que le signal reconstruit n'est pas le même que le signal d'origine.
Pour la reconstruction du signal, et plus exactement le calcul des transformées inverses, il nous faut l'information complète du signal (module et phase), par contre pour le calcul de $|x_{clean}|$, on a recupéré que le module du spectre du signal d'entrée et du bruit en négligeant la phase (perte importante d'information).

6. Reconstruire le signal audio débruité à partir du spectre complexe

Après reconstruction du signal à partir du spectre complexe du signal débruité, on remarque que le bruit de fond a été éliminé. Mais la qualité du son reconstruit n'est pas parfaite, la voix entendue est différente de celle d'origine.
En exploitant le module ainsi que la phase du spectre de **mix.wav**, on a pu en effet reconstruire le son mais avec une mauvaise qualité.

7. Etude de la qualité du signal audio débruité

Après l'écoute du signal debruité, on constate que la qualité du son n'est pas bonne, donc on peut conclure que le débruitage n'est pas parfait. L'algorithme de reconstruction **itfct** introduit des erreurs lors du calcul des transformées inverses ce qui implique une perte d'information.