ENSEA 3em SyM
Traitement du signal audio:
TP Modifications du signal audio par TFCT inverse

Geoffroy.Peeters@ircam.fr
UMR SMTS 9912 (IRCAM CNRS UPMC)
16 novembre 2016

Table des matières

1 Introduction

L'objectif de ce TP est de se familiariser avec les traitements du signal audio dans le domaine fréquentiel, en particulier à l'aide la Transformée de Fourier à Court Terme (TFCT.

Le TP s'effectuera sous Matlab. On visera à programmer proprement. Ceci inclut

- la subdivision du problème principal à résoudre en sous-problèmes, chacun sous forme de fonction (en particulier lorsque ces fonctions doivent être utilisées plusieurs fois),

- l'utilisation de noms de variables et de noms de fonctions indiquant clairement leur contenu ou leur fonctionnalité,
- la documentation du code écrit.

L'objectif du TP est d'effectuer la chaine de traitement complète : TFCT + débruitage + dilatation temporelle sans changer la fréquence + TFCT inverse.

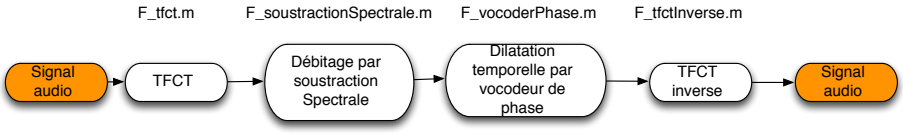

Figure :

Matériel nécessaire pour ce TP

- support du cours :
  `http://recherche.ircam.fr/anasyn/peeters/pub/cours/20161025_Peeters_20162017_ENSEA_3SYM_Cours_Transformation.pdf`
- fonction principale `C_TP.m`
- fichier audio `speech_noise.wav`

On lira le signal audio avec la fonction `audioread` de Matlab et on écrira le signal audio débruité et dilaté temporellement avec la fonction `audiowrite`. Si le signal est stéréo, on utilisera la moyenne des deux canaux pour le traitement à l'aide de la fonction `mean(audio_v,2)`.

2 Re-synthèse du signal audio original par TFCT inverse

L'ensemble des traitements suivants reposant sur l'analyse par TFCT et la re-synthèse audio par TFCT inverse, on commencera par implémenter ces deux fonctions

- `F_tfct.m` : une fonction calculant la TFCT à partir du signal audio
- `F_tfctInverse.m` : une fonction calculant le signal audio résultant de la TFCT inverse de la matrice contenant les spectres complexes à chaque instant d'analyse.

Il est conseillé d'effectuer le calcul de la TFCT soi-même (i.e. sans utiliser la fonction toute faite `specgram`). Ceci afin de se familiariser avec l'algorithme et de mieux comprendre la manière de réaliser la TFCT inverse.

On utilisera

- une fenêtre de type hanning de longueur `L_sec=60` ms
  - Question : comment calcule-t-on la taille de la fenêtre en échantillons `L_n` ?
- un pas d'avancement `STEP_sec` égale à 1/3 de la longueur de la fenêtre : `STEP_sec=20` ms
- un nombre de point `N` de la DFT égale à la première puissance de 2 supérieure à `L_n`

- on utilisera la fonction `nextpow2`

2.1 Etapes pour le calcul de la TFCT

Fonction à écrire : `F_tfct.m`

Allocation

- Calcul du nombre de trame `nbFrame`
  - Ce nombre dépend de la longueur du signal audio en échantillons `LT_n`, de la taille de la fenêtre en échantillons `L_n` et du pas d'avancement en échantillons `STEP_n`
    - Question : Quelle est cette formule ?
- Allocation de la matrice `X_im` de taille `(N/2+1, nbFrame)` dans laquelle seront stoqués les vecteurs de spectre complexe à chaque trame
  - Remarque : on ne stoquera que le demi-axe positif de la DFT (de 1 à `N/2+1`).
    - Tout les traitements se feront sur le demi-axe positif
    - Pour la resynthèse, on reconstruira l'axe négative en utilisant la règle de symétrie du spectre (complexe conjugué)
    - en Matlab, on pourra reconstruire la partie négative en utilisant `X(N:-1:N/2+2)) = conj(X(2:N/2))`

Boucle

- On crée une boucle allant de 1 à `nbFrame` avançant par pas de `STEP_n` à travers le signal audio
  - on prend une durée de signal audio égale à `L_n`
  - on multiplie par la fenêtre d'analyse
  - on calcul sa DFT de taille `N` à l'aide de la fonction `fft`
  - on ne garde que le demi-axe positif de la DFT
  - on le stoque dans une matrice `X_im` de taille `(N/2+1, nbFrame)`

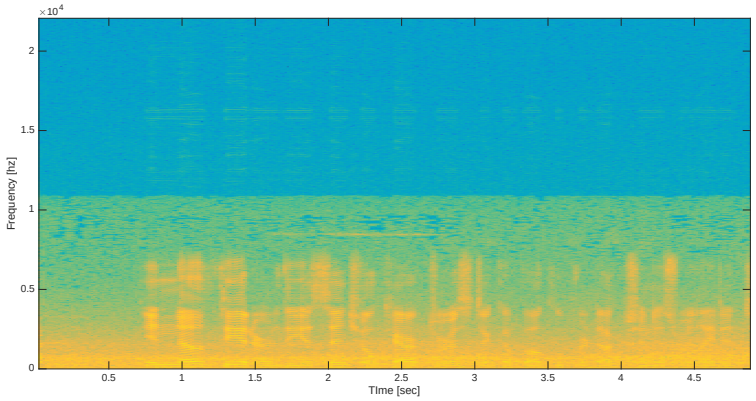

Figure : Logarithme du module de la TFCT en temps et fréquence du signal audio

2.2 Etapes pour le calcul de la TFCT inverse

Fonction à écrire : `F_tfctInverse.m`

Allocation

- On alloue le vecteur temporel dans lequel seront stoqués les additions/recouvrement progressive des trames audio `hataudio_v`
- On alloue le vecteur temporel dans lequel seront stoqués les additions/recouvrement progressive de la fenêtre de synthèse `hatfenetre_v`

Boucle

- On crée une boucle à travers les colonnes (les trames) de la matrice `X_im`
  - Pour chaque colonne on calcul sa DFT inverse et on garde sa partie réel (`real(ifft())`)
  - On tronque le signal résultant à la longueur `L_n`
  - On opère l'opération d'addition/recouvrement du signal audio
    - A quelle position doit-on additionner le signal ?
  - On opère l'opération d'addition/recouvrement de la fenêtre de synthèse

Normalisation

- Une fois l'ensemble du signal audio obtenu par addition/recouvrement, on le normalise par l'addition/recouvrement des fenêtres de synthèse
  - On évitera la division par 0 en ajoutant la valeur `eps`

On vérifiera que l'ensemble de la chaine TFCT d'un signal audio + TFCT inverse de `X_im` redonne bien le même signal audio. Pour cela, on peut par exemple superposé les deux formes d'onde en Matlab :
`plot(audio_v), hold on, plot(hataudio_v, 'r--'), hold off`
Elle doivent se superposer parfaitement.

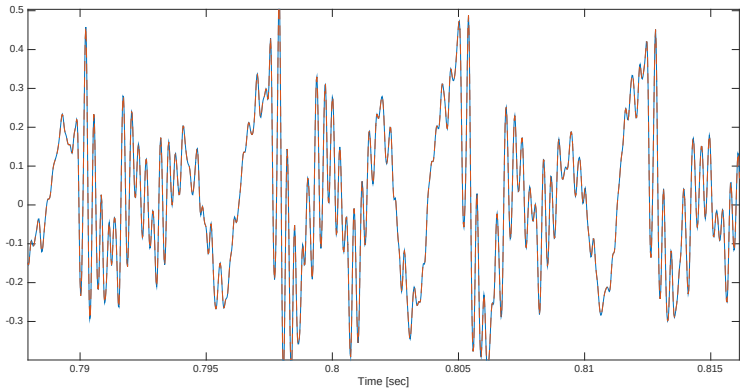

Figure : Zoom temporel sur le signal audio reconstruit par TFCT inverse (rouge) superposé au signal original (bleu)

3 Débruitage par soustraction spectrale

Fonction à écrire : `F_soustractionSpectrale.m`

Le signal audio `speech_noise.wav` contient un signal de voix par dessus un bruit de foule dans un stade. On cherche à débruiter ce signal en lui retirant le bruit de foule. On utilisera pour cela l'algorithme de soustraction spectrale. On utilisera le spectrogramme complexe `X_im` calculé à l'aide de la fonction `F_tfct.m`.

- Le bruit (signal de foule dans un stade) est présent de manière isolée entre les temps 0.1 et 0.7 s.
  - Question : comment trouver la position des trames de `X_im` correspondantes ?
- On utilisera ces trames de `X_im` pour calculer l'emprunte du bruit à retirer : $\mu(e^{j\omega}) = E\{N(e^{j\omega})\}$
- On calculera ensuite le filtre $H(e^{j\omega}) = 1 - \frac{e^{j\omega}}{|X(e^{jw})}$
  - Note : on peut grandement accélerer le calcul est utilisant les fonctionnalités matricielles de Matlab, en particulier la division terme à terme `./` ou la multiplication terme à terme.
- Pour éviter les valeurs négatives du filtre H, on appliquera la rectification demi-onde sur $H(e^{j\omega})$: $H_R(e^{j\omega}) = \frac{H(e^{j\omega})+|H(e^{j\omega})|}{2}$

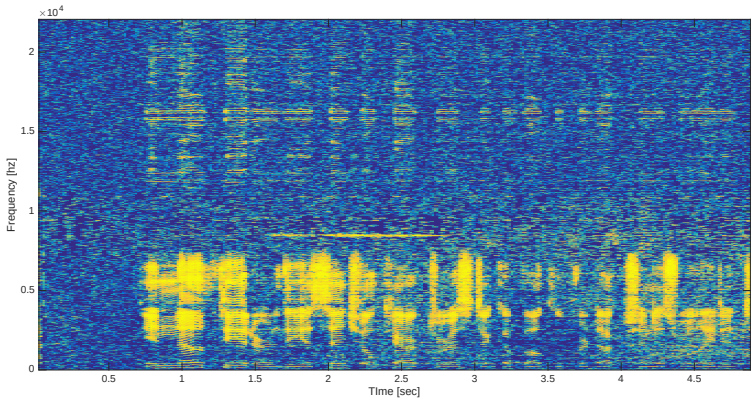

Figure : Matrice de filtrage H_R après rectification demi-one

- On reconstruira le spectrogramme complexe `hatX_im` par simple multiplication terme à terme matricielle : `hatX_im = H_m .* X_im`

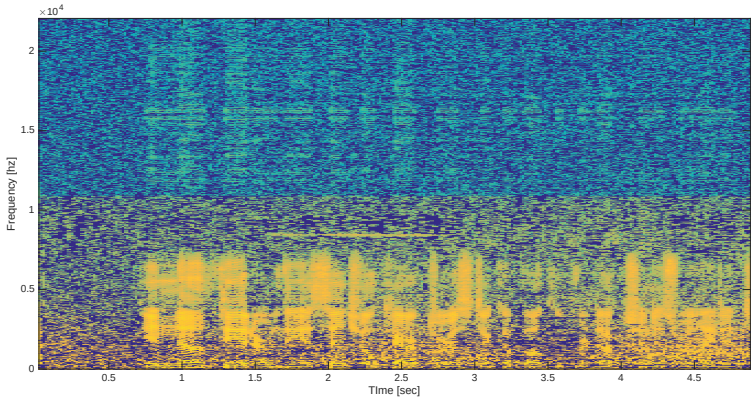

Figure : Logarithme du module de la TFCT du signal débruité

- On reconstruira le signal audio correspondant avec l'algorithme de TFCT inverse développé précédemment : `F_tfctInverse.m`.

4 Dilatation temporelle sans changement de hauteur par vocodeur de phase

Fonction à écrire : `F_vocoderPhase.m`

On utilisera l'algorithme du vocodeur de phase pour ralentir le signal débruité d'une valeur `alpha=0.6`.

Allocation

- Si on note `R` le nombre de trames d'analyse, on calcul le nombre de trames de synthèse `Rp = round(R/alpha)` nécessaire à la dilation `alpha` demandée
- On allouera ensuite la matrice qui va contenir les spectres complexes modifiés à chaque trame de synthèse `hatX_im`. Elle est de dimension `(N/2+1, Rp)`.

Boucle

- On créera ensuite une boucle pour parcourir toutes les trames de synthèse $rp \in [1, Rp]$.
  - Pour une trame de synthèse donnée rp, on calculera la trame d'analyse correspondante selon la règle de trois : `r = (rp-1)*(R-1)/(Rp-1) + 1`.

4.1 Calcul de l'amplitude

- On calculera le spectre d'amplitude résultant de l'interpolation entre `r1=floor(r)` et `r2=r1+1` .

4.2 Calcul de la phase

4.2.1 Solution 1 : la phase est gardée constante

- On gardera la phase nulle pour la reconstruction `cumulPhi=0`
- On recréera le spectre complexe de synthèse à la trame `rp` en utilisant le spectre d'amplitude interpolé et le spectre de phase ainsi crée
  - `hatX(:,rp) = amFft .*  exp(j * cumulPhi)`.
- On reconstruira le signal audio correspondant avec l'algorithme de TFCT inverse développé précédemment : `F_tfctInverse.m`.
- Remarque : en écoutant le signal `soundsc(hataudio, sr)` on entend le signal robotisé
  - Question : Comment changer la hauteur à laquelle parle le robot ?
    - Piste : essayer de refaire tourner le programme en changeant le pas d'avancement `STEP_sec` de la TFCT

4.2.2 Solution 2 : la phase est incrémentée selon un modèle théorique

- On calculera l'avancement théorique de phase `deltaPhi` pour chaque fréquence du spectre et le pas d'avancement `STEP_n` utilisé. Ceci se fait selon la règle $deltaPhi = 2\pi f_k \Delta t$
  - Question : comment calcule-t'on les fréquences f_k en Hz de la DFT ?
  - Question : comment calcule-t'on Δt à partir de `STEP_n`
- Les phases seront incrémentées progressivement au cours des trames de synthèse. Ceci de manière à garantir la continuité de l'évolution de la phase pour chaque fréquence
  - `cumulPhi = cumulPhi + deltaPhi`

  - On choisira comme phase initiale de synthèse (pour `rp=1`) celle du spectre complexe initiale d'analyse (en `r=1`)
- On recréera le spectre complexe de synthèse à la trame `rp` en utilisant le spectre d'amplitude interpolé et le spectre de phase ainsi crée
  - `hatX(:,rp) = amFft .* exp(j * cumulPhi)`.
- Remarque : en écoutant le signal `soundsc(hataudio, sr)` on entend un signal modulé en temps
  - Question : A quoi cela est-il du ?

4.2.3 Solution 3 : la phase est incrémentée en utilisant la fréquence instantannée

- Afin de prendre en compte la fréquence réellement observée dans chaque canal k de la DFT, on calculera sa fréquence instantannée f_0 en utilisant les phases observées en r1 et en r2.
- Pour résoudre l'indétermination de n (voir cours), on cherchera la fréquence instantannée f_0 la plus proche de f_k.
- Ceci conduit à

```
decalPhi = angle(X(:,r2)) - angle(X(:,r1)) - deltaPhi
decalPhi = deltaPhi - 2 * pi * round(decalPhi / (2 \* pi))
cumulPhi = cumulPhi + deltaPhi + decal_phi_v
```

- Question : expliquer en quoi ceci est équivalent à trouver la fréquence instantannée f_0 la plus proche de f_k ?

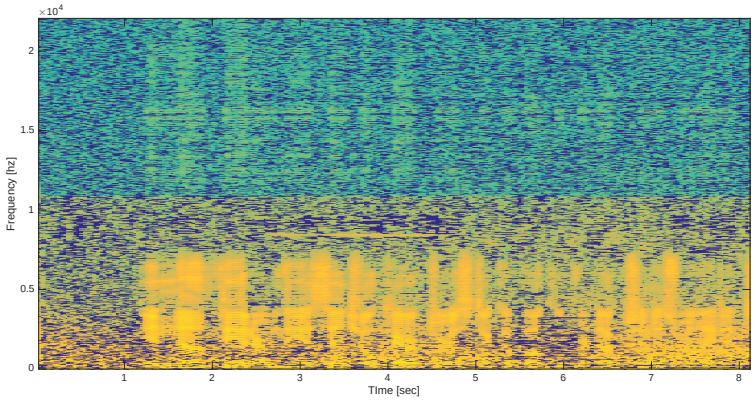

Figure : Logarithme du module de la TFCT du signal débruité et étiré d'un facteur $\alpha = 0.6$

- On reconstruira le signal audio correspondant avec l'algorithme de TFCT inverse développé précédemment : `F_tfctInverse.m`.