Vocodeur

Marc Heinrich
École normale supérieure

Baptiste Lefebvre
École normale supérieure

30 mai 2013

1 Introduction

Le principe du vocodeur est de représenter un signal par sa transformée de fourrier à fenêtre, afin de rendre plus facile certaines modifications (sur les fréquences présentes par exemple).
Le but de ce projet était d'implémenter les trois phases du vocodage :

- Analyse, calcul de la FFT à fenêtre
- Modification éventuelle la représentation fréquentielle
- Synthèse, reconstitution du signal à partir de la FFT à fenêtre

Nous avons implémenté la première et la troisième phases à partir de l'algorithme décrit dans *Implementation of the Digital Phase Vocoder Using the Fast Fourier Transform*, M. R. Portnoff.
Pour la modification du son, nous avons implémenté un programme permettant de modifier le signal aussi bien en temps qu'en fréquence.

2 Algorithme

2.1 Analyse

L'algorithme naïf pour réaliser la phase d'analyse consiste à calculer, pour chaque instant t, la convolée du signal d'entrée avec un filtre passe bas, décalé de t (pour conserver les valeurs temporelles du signal d'entrée proches de t). Puis de considérer sa transformée de Fourier.
Nous avons pris comme filtre un sinus cardinal, restreint à une plage finie, comme suggéré dans l'article. Cette méthode demande beaucoup d'opérations, et n'est donc pas très rapide. Il y a deux idées pour améliorer la complexité. Si on note N le nombre de bande de fréquences que l'on veut obtenir dans notre transformée de Fourier à fenêtre

- Une astuce permet de faire la transformée de Fourier sur un signal de longueur N, au lieu de la taille du support du filtre (typiquement $2QN + 1$).

– Comme les fréquences contenues dans chacune des bandes de fréquences (vues comme des signaux temporels), sont limitées, on peut reconstruire le signal initial avec seulement 1 échantillon sur N.

En pratique, pour le deuxième point, on prend un peu plus que 1 échantillon sur N, du fait que le filtre qu'on utilise n'est pas parfait.

Code pour la phase d'analyse :

```
% h : filtre utilisé
% samples : les valeurs initiales
% N : nombre de canaux de fréquence
% R : pas d'échantillonnage pour la DFT à fenêtre
% X : résultat de la transformée de Fourier à fenêtre


% On pré calcule les filtre décalé, ainsi que les samples décalés.
for m = 1:N ;
    hi(m,:) = [h, zeros(1,N)]((-M:N:M) +M +1 +m) ;
    xshift(m,:) = [samples(m:size_samples) , zeros(1,m-1+2*M+1)] ;
end ;

% xt : valeurs avant la fft

for i = (1:R:number_of_samples) ;
    xt((i-1)/R +1,:) = sum(xshift(:, i + M+1 +(-M:N:M)) .* hi,2) ;
    xt((i-1)/R +1,:) = shift(xt((i-1)/R +1,:),mod(i,N)) ;
    end ;

% Finalement on fait la fft sur chaque ligne pour avoir le résultat.
X = fft(xt,[],2);
```

2.2 Synthèse

Pour reconstruire le signal à partir de la transformée de Fourrier à fenêtre, on calcul la FFT inverse pour chacun des échantillons (espacés de R), et on ré-obtient le signal initial, en prenant le coefficient de la n^{eme} bande au temps n, par interpolation à partir de ceux dont l'on dispose. En procédant ainsi, on peut calculer les valeurs par bloc de R à la fois (ils ne dépendent que des même coefficients de la FFT à fenêtre).

Code pour la phase de synthèse :

```
% Calcul du filtre d'interpolation (interpolation linéaire)
f = [ ((R-1):(-1):0) /R ; (0:(R-1)) /R ] ;
```

```
% on calcul les résultats par blocs de R à la fois
for n = 1:size(X0,1)
    % On calcule la fft sur chaque ligne
    s = ifft(X0(n:(n+1),:),[],2) ;
    % shift puis sélection pour avoir les bons coefficients
    s = shift(s,mod(-n*R,N),2) ;
    s = s(:,1:R) ;
    x((R*(n-1)+1):(R*n) ) = real(sum(f .* s,1)) ;
end ;
```

2.3 Modification

Pour appliquer des modifications à notre son, on procède de la façon suivante :
On commence par accélérer le son en ré-échantillonnant le signal initial (on prend une valeur sur deux pour accélérer le son deux fois).
Ensuite, on modifie le spectre (via le vocodage), en écrasant le spectre vers les hautes fréquences (pour une accélération, on dilate le spectre pour un ralentissement).

3 Resultats expérimentaux

Pour l'implémentation, on a réussi à faire un programme qui fait la phase d'analyse puis de synthèse, sans aucune modification du signal. Le signal obtenu en sortie ressemble beaucoup au signal initial lorsqu'on prend un pas d'échantillonnage suffisamment petit (pour $R = \frac{N}{8}$, on n'entend pas la différence entre les deux signaux). Par contre, lorsque R devient trop grand, le son commence à être déformé. Pour $R = \frac{N}{4}$, et $N = 2^8$, on peut entendre des bruits de fond graves lorsque le signal possède. On observe un phénomène d'aliasing : les trop hautes fréquences sont repliées sur les basses fréquences. Pour des sons d'instruments de musique, cela donne une modification des notes que l'on entend.

Pour ce qui est de l'accélération ou du ralentissement du son, les résultats qu'on a obtenus n'étaient pas très probants.
La parole (sans musique, instruments...) est ce qui a donné les meilleurs résultats. Cependant, quand on essaye de ralentir, on a une oscillation basse fréquence qui se rajoute sur les sons graves, donnant un peu un effet de voix chevrotante. Lorsqu'on l'accélère, on a aussi un phénomène qui pourrait ressembler à des battements.
Pour tenter de trouver pourquoi notre programme donne ces effets sonores, on a essayé avec une sinusoïde et du bruits blancs.
Lorsqu'on regarde le spectre de la sinusoïde après passage dans le vocodeur, on

observe que des nouvelles fréquences sont apparues, s'étalant autour de la fréquence initiale et des ses multiples. Cela pourrait effectivement correspondre à des phénomène de battement qui nous font entendre des bruits de plus basse fréquence.

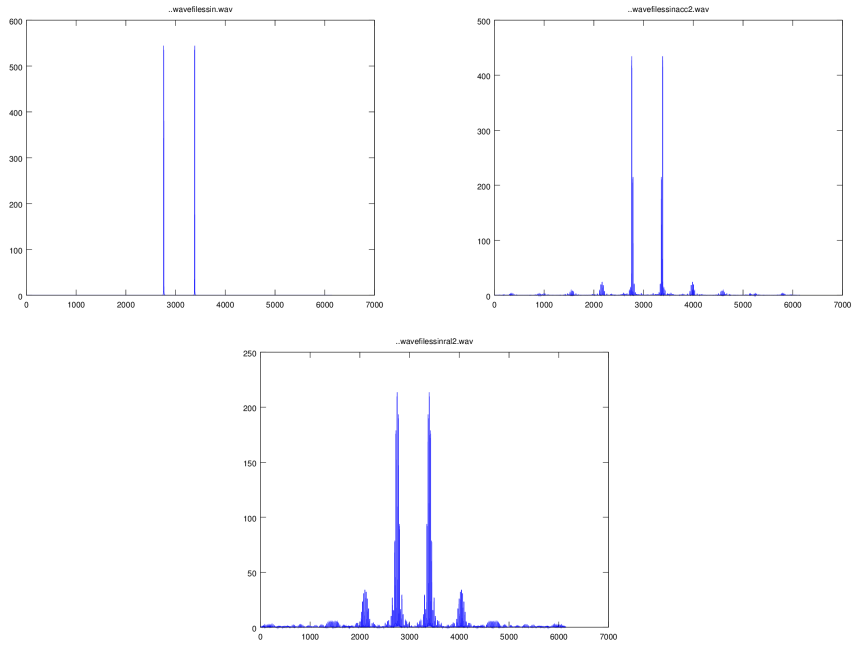
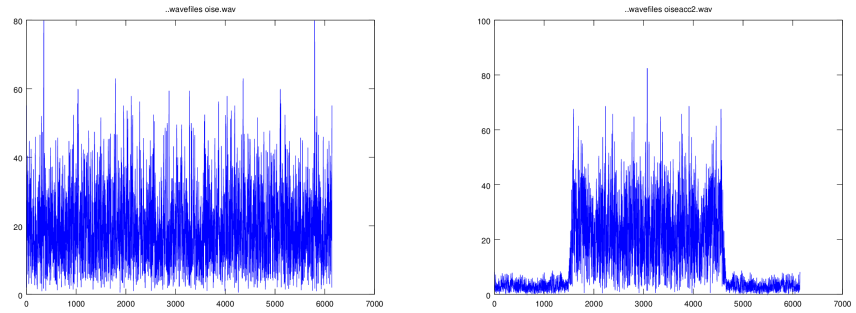


Figure : Spectre sur une fenêtre d'une sinusoïde, (a) avant vocodage, (b) après accélération (x2) (c) après ralentissement (x0.5)

Le bruit blanc accéléré ou ralenti, donne aussi cette impression de son basse fréquence assez régulier qui se rajoute au bruit initial. On observe aussi une atténuation des hautes fréquence (cas de l'accélération), ce qui est normal, puisque lorsqu'on modifie le spectre, on le contracte vers les basses fréquences, en l'étendant au delà avec des 0. Par contre, il semble qu'il y ait aussi une légère atténuation des basses fréquences qu'on ne sait pas justifier.



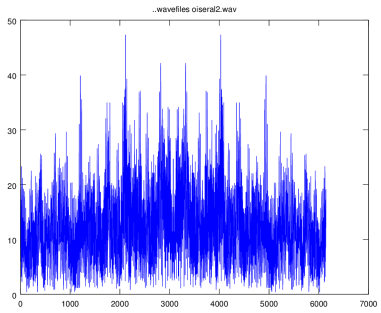


Figure : Spectre sur une fenêtre d'un bruit blanc, (a) avant vocodage, (b) après accélération (x2) (c) après ralentissement (x0.5)