BE1 – Introduction au traitement de la parole

SDI

2021 – 2022

Stéphane Rossignol – stephane.rossignol@centralesupelec.fr

1 PSOLA

1.1 Introduction

La méthode PSOLA, pour Pitch Synchronous Overlap and Add, a été inventée chez Orange (plus spécifiquement, chez l'ancien France-Télécom, CNET). Pour plus de détails sur son fonctionnement, voir vos notes de cours (slide 270 de la deuxième partie, quand vous l'aurez), ou encore par exemple :

HTTP ://EN.WIKIPEDIA.ORG/WIKI/PSOLA

(et voir aussi la thèse de doctorat finlandaise donnée en référence ou en external links sur le site de WIKIPEDIA, à partir de la page 34 (ou 43 du pdf), section 5.3.1.).

L'intérêt de PSOLA vient du fait que la méthode essaie de préserver le *timbre* de la voix originale (ou de l'instrument de musique, si nous nous intéressons à la manipulation des sons musicaux). Cette préservation est extrêmement importante pour la (re-)synthèse de la parole. En effet, nous avons vu en cours qu'il était parfois nécessaire de modifier le pitch (l'une des composantes de la **prosodie**) d'un son de parole donné, notamment pour des applications téléphoniques :

nous pouvons difficilement admettre que qui que ce soit du grand public accepte un jour de ne pas reconnaître la voix de la personne, a priori proche, qui s'adresse à lui ! De plus, bien évidemment, quand nous serons en mesure d'interagir convenablement via la parole avec nos machines (nos robots), il n'est pas envisageable que nous acceptions de parler avec des « machines sonnant par trop l'électronique ».

PSOLA est une méthode très efficace pour modifier la **prosodie** des sons de parole sans dégrader leur *timbre* (c'est sans doute la meilleure à l'heure actuelle). L'une des difficultés qui limitent sa popularité vient de ce qu'elle n'est pas facilement complètement automatisable. C'est une voie de recherche, entre beaucoup d'autres, qui ouvre les plus vastes perspectives !

1.2 Sujet

Sur mon site, vous pouvez trouver un script octave qui modifie quelques signaux artificiels grâce à la technique PSOLA :

http ://www.metz.supelec.fr/metz/personnel/rossignol/psola2.m

Il s'agit que vous l'adaptiez pour faire varier de ±10 % le pitch des parties voisées par exemple du fichier **effacer.wav**, que vous trouvez au même endroit.

Note : une méthode brutale pour faire ça serait de faire jouer à octave/matlab le son à une fréquence de restitution différente de celle d'enregistrement (par exemple : *sound(xx,fe*1.1)*) ; vous remarquerez alors la différence de qualité par rapport à ce que vous obtenez avec PSOLA.

1.3 Manipulation de l'existant – Attention

- Pour commencer, examinez le code et familiarisez-vous avec lui
- Ce code est fait pour tourner sous octave ; par principe, il n'a pas été testé sous matlab (de ce fait, il va peut être requérir quelques adaptations)
- Faites varier NPER : pour mieux voir les marques, utilisez alors NPER petit ; ou pour mieux entendre les modifications, utilisez NPER grand
- Notez les difficultés que posent les transitions (silence-signal puis signal-silence) ; même à la main, il n'est pas facile de choisir où poser les marques PSOLA ; nous avons une mesure objective de la qualité du positionnement des marques grâce au SNR mesuré après reconstitution du signal (voir la ligne de code *snr = 10*log10(sum(xx.ˆ 2)/sum(diff.ˆ 2))* ;)
- La modification du pitch obtenue est environ ce qui est désiré, mais en règle générale ne peut pas être exacte car *(ll1/2+ll2/2)/2*-0.1* (voir le code) n'est pas entier (d'où l'utilisation du *round*)

1.4 Suggestions

- Il faudrait placer les marques de façon automatique ; c'est un problème pas tout à fait résolu, alors commencez par placer vos marques à la main

(la suite du BE consistera à essayer de développer des méthodes automatiques)
  - ceci peut être long et difficile, alors traitez d'abord une petite partie du son (le premier é)
  - quelques petits outils de mon code sont peut être adaptables pour vous

— N'oubliez pas : donnez à chaque marque placée une valeur de voisement (0 : trame de son non-voisée ; ou 1 : trame de son voisée)
  - une façon simple de collecter les données nécessaires à PSOLA est de faire un fichier texte (.txt) de la forme :

position_de_la_marque_1_en_numéro_d'échantillons	voisement_1
position_de_la_marque_2_en_numéro_d'échantillons	voisement_2
...	...

soit pour les signaux artificiels du code fourni :

240	0
480	0
720	0
960	0
1200	0
1347	1
...	...

  - ces fichiers s'écrivent et se lisent facilement sous octave/matlab grâce aux commandes *save* et *load*

— N'oubliez pas : les parties non-voisées ne sont pas modifiées ; seules les parties voisées le sont

— N'oubliez pas : la taille des fenêtres est fixe sur les parties non-voisées ; elle est fonction du pitch sur les parties voisées

— Les fenêtres peuvent être asymétriques (voir le code) ; ce qui compte, c'est que le début de la fenêtre courante corresponde au milieu de la fenêtre précédente et que la fin de la fenêtre courante corresponde au milieu de la fenêtre suivante

— En règle générale, nous choisissons de positionner une marque PSOLA sur un maximum (ou un minimum) du signal temporel (en respectant la période instantanée de ce signal), ce bien sûr parce que nous multiplions par une fenêtre de pondération (Hanning, Hamming... ; elles présentent quasi toutes un maximum en leur milieu) et que nous voulons garder le maximum d'énergie du signal original
  - comme le signal de parole est hautement non-stationnaire, il n'est pas toujours aisé de respecter toutes les contraintes : coller au mieux à la période tout en s'assurant qu'elle ne varie pas trop brutalement, et positionner le milieu de la fenêtre sur un maximum

⇒ parfois, il faut glisser, d'une fenêtre à la suivante, d'un extrémum à

l'extrémum le plus proche
⇒ etc.

— L'usage des fenêtres de pondération et la manière dont elles sont utilisées viennent, bien sûr de ce qu'il faut être capable de reconstruire le signal original à partir du découpage fait par PSOLA (c'est-à-dire obtenir le SNR de reconstruction présenté ci-dessus le plus bas possible)

2 Mesure de voisement

2.1 Introduction

Nous venons de voir que notamment la mesure du degré local de voisement est nécessaire pour PSOLA (entre autres). Des méthodes existent dans la littérature qui permettent de décider si une trame de son est voisée ou pas. Parmi les mesures de voisement/non-voisement de l'état de l'art, l'énergie, le taux de passage par zéro (ou ZCR), l'analyse des coefficients d'autocorrélation, l'analyse du spectre, etc. sont très répandues. Elles requièrent de faire l'hypothèse de stationnarité du signal sur quelques dizaines de millisecondes, et donc requièrent un processus de découpage en trames. La méthode présentée ici est basée sur le signal analytique (AS). Elle utilise des trames très courtes – quelques millisecondes, possiblement. L'un des objectifs est d'obtenir une grande précision dans les positions des transitions voisé/non-voisé et non-voisé/voisé, précision forcément bénéfique au PSOLA utilisé après coup.

2.2 Sujet

Implantez une ou plusieurs méthodes simples (énergie et taux de passage par zéro). Puis implantez la méthode et testez-la sur quelques sons (ceux donnés, ou éventuellement d'autres que vous enregistrez).

Comparez les résultats obtenus avec les différentes méthodes.

2.3 La méthode

L'analyse du son est effectué en quatre étapes (figure 1).

Premièrement, le signal audio est filtré par un filtre passe-bande, avec une décroissance en $1 - \sqrt{f/f_s}$ dans la bande, et avec $f_1 = 50Hz$ et $f_2 = 300Hz$. Le but de ce filtrage est, dans les parties voisées du son, de rendre le premier partiel aussi prédominent que possible en amplitude. Donc, le signal :

$$s(n) \simeq a_1 \cos\left(2\pi f_0 n/f_s + \phi_1\right)$$

est obtenu, où f_0 est la fréquence fondamentale et f_s la fréquence d'échantillonnage. Dans les parties non-voisées du son, un bruit est obtenu.

Deuxièmement, le signal analytique est déterminé en utilisant le filtrage de Hilbert. Dans les parties voisées du son, le signal :

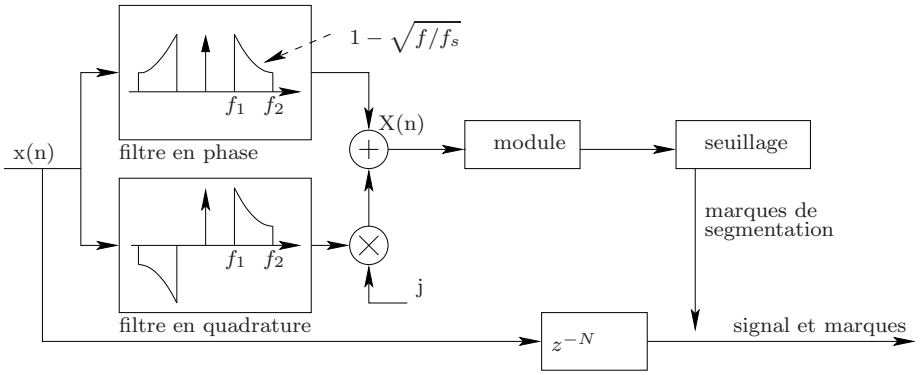

FIGURE 1 – Synoptique de la méthode

$$X(n) \simeq a_1 \exp\left(2\pi j f_0 n/f_s + j\phi_1\right)$$

est obtenu. Notons que, sur la figure 1, le filtrage passe-bande et le filtrage de Hilbert sont effectués conjointement.

Troisièmement, le module $A = |X(n)|$ du signal analytique est estimé. Dans les parties voisées du son, il faut remarquer que ce module fournit une estimation approximative de l'amplitude du premier partiel ($A \simeq a_1$). Dans les parties non-voisées du son, seulement du bruit est obtenu. Il est donc fait l'hypothèse que A a des valeurs plus grandes dans les segments voisés que dans les segments non-voisés du son.

Quatrièmement, le module A est seuillé (idéalement, automatiquement : voir les méthodes venant du traitement des images, comme Otsu, etc.).

3 Technique de positionnement (semi-)automatique des marques PSOLA

3.1 Introduction

Il serait intéressant de pouvoir positionner automatiquement les marques PSOLA. Note : en règle générale, nous utilisons pour commencer une méthode automatique, puis nous faisons une deuxième passe à l'œil et à la main, permettant de replacer les marques qui semblent ne pas tout à fait tomber où il faut, permettant aussi d'enlever les fausses alarmes, et permettant finalement d'ajouter les détections (qui semblent) manquantes.

3.2 Sujet

Mettez en place une méthode de votre cru permettant d'effectuer ceci, et testez-la sur quelques sons (ceux donnés en cours, ou éventuellement d'autres, enregistrés par vous). Vous êtes libres : soyez inventifs.

4 Conclusion – Suppléments

Allez aussi loin que possible dans le traitement de l'un (ou de plusieurs) de ces suppléments. Vous pouvez valider votre code sur des signaux simulés, puis essayer de l'appliquer au(x) fichier(s) **.wav** que vous avez choisi(s) de traiter.

— Maintenant, vous avez tout ce qu'il vous faut pour faire un PSOLA complètement automatique. Mettez les morceaux bout à bout, et essayez tout ça sur un son/des sons que vous enregistrez.

— Notez de plus que la méthode de mesure voisement/non-voisement est facilement adaptable pour devenir un pitch-tracker. Essayez de le faire.

— Il est possible aussi d'allonger ou de raccourcir chacun des phones, ce en manipulant les trames PSOLA. Notamment, pour allonger le son, il s'agit de dupliquer certaines marques PSOLA. Bien sûr, la partie du code où les trames PSOLA sont ajoutées doit être modifiée en conséquence. Pour le moment, c'est simple : implicitement il est fait l'hypothèse que les marques se suivent dans l'ordre temporel : c'est-à-dire que pour générer la trame PSOLA correspondant à la marque i nous avons besoin des marques $i-1$ et $i+1$, auxquelles nous avons accès directement dans le vecteur des marques : il s'agit de la marque précédant et de celle suivant la marque courante i. Maintenant, vu que l'ordre temporel n'est plus respecté, il faut explicitement calculer les débuts et fin de chaque trame PSOLA dans le fichier résultat (la variable DEC ne suffit plus), et il faut donner dans le fichier des marques le début et la fin de chaque trame dans le fichier original. Par exemple, nous pouvons considérer que nous ayons ça dans notre fichier des marques :

i	$i-1$	$i+1$	
...	...	...	...
m_4	m_3	m_5	1
m_5	m_4	m_6	1
m_4	m_3	m_5	1
m_6	m_5	m_7	1
...	...	...	...

(Note : la trame PSOLA 4 apparaît deux fois). La deuxième trame 4 se met à la place de la trame 6 originale : dans le fichier résultat, sa position centrale est m_6, elle débute en $m_6-(m_4-m_3)$ et finit en $m_6+(m_5-m_4)$. Du coup la position centrale de la marque 6 originale est repoussée disons en p_7 (note : le numéro d'ordre de la trame 6 originale est maintenant 7). Nous avons alors : $p_7 = m_6+(m_5-m_4)$; elle débute ici : $p_7-(m_6-m_5)$, elle finit là : $p_7+(m_7-m_6)$. Etc.