TP - Codage audio

Adrien Llave

21 octobre 2018

- Télécharger le dossier contenant les sources du TP à l'adresse suivante : `https://github.com/a-llave/SERI-BE-transform_coding`
- Les scripts se trouvent dans le dossier **src**
- Les fichiers audio se trouvent dans le dossier **resources**. Ils sont échantillonnées à 16 kHz et quantifiés à 16 bits
- Ouvrir Matlab

L'objectif de cette séance de travaux pratiques est de mettre en œuvre un codage par transformée appliqué à un signal audio.

Tout d'abord, vous prendrez en main le quantificateur scalaire uniforme et étudierez son comportement sur un signal test et sur les extraits audio fournis. Puis, il vous faudra bien comprendre le traitement par bloc WOLA et vous familiariser avec la nature des signaux que vous manipulez. Alors, vous mettrez en œuvre le codage par transformée grâce aux connaissances acquises dans les 2 exercices préliminaires (et vos notes de cours, et l'internet). Enfin, vous comparerez les performances du codage par transformée et ses différentes variantes avec la quantification scalaire uniforme dans le domaine temporel.

Vous disposez d'un script Matlab associé à chaque exercice, fonctionnel en l'état, qui doit vous servir de base. Vous n'aurez qu'à compléter les trous dans l'algorithme, afficher les données de manière pertinente et surtout n'oubliez pas d'écouter le résultat des traitements que vous appliquez au son.

N.B. L'aide de Matlab est souvent d'un grand recours.

1 Exercice 1 : Quantification scalaire uniforme
20 min

- Ouvrir le fichier **src/uni_scal_quantif_base.m**

1) Décrire les méthodes d'arrondis possibles avec la fonction **myQuantize2**.

2) Afficher l'erreur de quantification pour chaque méthode. En déduire quel est l'avantage de chacune d'entre elles ?

3) Il est possible d'avoir un état représentant le zéro ou pas. A votre avis, quel conséquence cela peut-il avoir sur le signal quantifié ?
- Charger le signal **parole_16k.wav** à l'aide de la fonction **audioread**
- Sous-quantifier la parole grâce à la fonction **myQuantize2**

– Écouter les signaux audio à l'aide de la fonction **sound**

4) A partir d'une sous-quantification de combien de bits le bruit de quantification devient-il audible ?

2 Exercice 2 : Weighted overlap-add (WOLA) 20 min

– Ouvrir le fichier **src/WOLA_base.m**

Ce script met en œuvre un traitement par trame avec un recouvrement de 50 %.

– Implémenter le coefficient d'aplatissement spectral

1) Faire varier la taille de la trame. Quel est l'impact de la taille de la trame sur le coefficient d'aplatissement spectral pour le signal **sweep_16k.wav** ? Pourquoi ?

– Ouvrir le fichier audio **parole_16k.m**

2) Connaissant la fréquence d'échantillonnage, quelle taille de la trame en nombre d'échantillon choisissez-vous ? Justifier.

3) Dans quel contexte est-il nécessaire d'utiliser le traitement par bloc avec recouvrement et pourquoi ?

4) Quel est l'inconvénient du traitement par trame avec recouvrement lorsqu'on veut faire de la compression de données ? Comment minimiser cet inconvénient ?

3 Exercice 3 : Codage par transformée 2 h

– Ouvrir le fichier **src/TC_base.m**

– Exécuter le script, vous devez observer 2 figures cote à cote, l'une est le signal temporel sur la trame considérée, l'autre est son spectre d'amplitude.

Ce script effectue une sous-quantification d'un signal audio dans le domaine temporel et dans le domaine fréquentiel à l'aide d'une chaine de traitement WOLA.

1) A partir de ce que vous observez, la DFT (discret Fourier transform) vous semble être une transformation adaptée au codage par transformée ? Justifier.

2) Comparer visuellement les signaux quantifiés dans le domaine temporel et dans le domaine fréquentiel. Comparer leur SNR et écouter les. Commenter.

– Dans la section du code commentée *FREQUENCY DOMAIN QUANTIZING - OPTIMAL ALLOCATION FLOORED*, implémenter la répartition optimale du nombre de bits par coefficient fréquence et la quantification associée. Arrondir à l'entier inférieur la solution optimale d'allocation de bits.

Rappel :

$$R_k = R_0 + \frac{1}{2} log_2 \left(\frac{\sigma^2_{y(k)}}{\left(\prod_k^N \sigma^2_{y(k)} \right)^{\frac{1}{N}}} \right)$$

3) Reste-t-il des bits non-alloués ? Si oui, en moyenne combien et avec quel écart-type ?
- L'algorithme de Huang-Schulteiss modifié (voir annexe B) permet utiliser l'ensemble de la ressource binaire en répartissant les bits non-alloués. Mettre en œuvre cet algorithme.

4) Cet algorithme est-il viable dans une implémentation temps-réel et pourquoi ?
- Un algorithme glouton permet d'atteindre un optimum local en général satisfaisant pour un problème d'optimisation où la solution n'est pas atteignable dans un temps polynomial. Mettre en œuvre l'algorithme glouton proposé en annexe C.

5) Sur quelle hypothèse se repose l'algorithme glouton ?

6) Comparer les SNR correspondant à chaque algorithme. Sur la base de ce critère, quel algorithme est le plus performant selon vous et pourquoi ?

7) Écouter les signaux audio issus de chaque algorithme. Quel algorithme est le plus performant selon vous et pourquoi ?

8) Le SNR vous semble-t-il être un critère objectif adéquate pour quantifier le rapport signal sur bruit que vous percevez ?

9) Comparer la moyenne et l'écart type du temps d'éxecution pour chaque algorithme grâce aux fonctions **tic** et **toc**. Sur la base de ce critère, quel algorithme est le plus performant selon vous et pourquoi ?

A Norme de codage

Afin de faciliter la lecture du code, les scripts que vous allez manipuler suivent une convention de nommage des variables. Le nom des variables suivent quelques règles simple de sorte à avoir une idée en un coup d'œil du type de données qu'elles contiennent. De manière générale, je vous encourage à en suivre une dans vos projets personnels mais surtout lors de projets collaboratifs, vous gagnerez beaucoup de temps.

Type	Extension	Exemple
string	_s	myString_s
integer	_n	myInt_n
float	_f	myFloat_f
vector	_v	myVector_v
matrix	_m	myMat_m
structure	_S	myStruct_S
cell	_C	myString_C

B Algorithme de Huang-Schulteiss modifié

1) Calculer les R_k optimaux

2) Si certains R_k sont négatifs : les forcer à 0, recommencer l'étape **1** en retirant les coefficients k concernés

3) Répéter l'étape **2** jusqu'à ne plus avoir de R_k négatifs

4) Tronquer [1] les R_k

5) Allouer les bits restant aux coefficients avec l'erreur de quantification maximum

C Algorithme glouton

1) Initialisation
- $R_k = 0 \ \forall k$
- $D_k = \sigma_k^2 \ \forall k$

2) Tant que $\sum_k^N R_k \leqslant N.R_0$
- $l = \text{argmax}_k D_k$
- $R_l \leftarrow R_l + 1$
- $D_l \leftarrow D_l/4$

1. Arrondis à l'entier inférieur