L3 MI / Systèmes de Communications

TP2 codage de source

Téléchargez l'archive dans moodle, puis dézippez-la dans votre répertoire de travail.

Dans ce TP, les fichiers audio créés peuvent être lus simplement dans une fenêtre terminal, avec la commande `play`. Attention à bien régler les paramètres audio de votre machine au préalable : en cas de doute sur le niveau sonore du fichier que vous allez écouter, mettre le volume au minimum pour commencer. Dans chaque exercice, garder le même règlage du son de manière à ne pas fausser les comparaisons lors des écoutes.

1 Codage par modélisation AR de la source

NB : vous n'avez pas besoin de comprendre le code interne des fonctions utilisées, mais les en-tête des fonctions (commentaires au début des fichiers) sont utiles pour savoir comment utiliser ces fonctions.

Ouvrir et lancer *test_codecAR.sce*. Ce programme charge un fichier audio (échantillons quantifiés sur 16 bits) dans un vecteur s, affiche s, le quantifie sur 12 bits et enregistre le signal quantifié s_q.

Ecoutez et comparez s et s_q. Baissez le nombre de bits de quantification de s_q et relancez jusqu'à la limite de perception du bruit de quantification [1]. Soit k_s^{min} ce nombre. Testez différents fichiers de voix parlée pour vérifier votre valeur.

Un signal vocal s peut être découpé en tranches de 10 à 30 ms et représenté, sur chaque tranche, par un modèle dit auto-régressif d'ordre p :

$$s(n) = \sigma_e e(n) - \sum_{i=1}^{p} a_i s(n-i)$$

C'est-à-dire que chaque échantillon $s(n)$ est une combinaison linéaire des précédents, plus un terme d'innovation (appelé aussi résidu de prédiction) $\sigma_e e(n)$, tel que la puissance de e vaut 1. Nous allons nous appuyer sur cette redondance pour comprimer presque sans perte le signal s. Au lieu de transmettre les échantillons $s(n)$ quantifiés, le codeur transmet pour chaque tranche de 20 ms les coefficients a_1 à a_{10}, σ_e et la suite des $e(n)$ (160 échantillons pour un échantillonnage à 8 kHz)

Décommenter la fin du programme. Le programme code le signal s en un flux *coded* (qui contient les a_i, σ_e et $e(n)$ successifs), décode celui-ci en un signal sonore s_{codec} et affiche e. Dans cette version, aucune valeur n'est quantifiée. Ecoutez et comparez s et s_{codec}.

A présent nous allons quantifier e sur un nombre de bits minimal. Dans l'appel de la fonction *cod_AR*, réglez les paramètres pour une quantification de e sur k_s^{min} bits (la valeur trouvée précédemment). Lancez le programme, écoutez et comparez s et s_{codec}. Baissez le nombre de bits de quantification de e jusqu'à la limite de perception du bruit de quantification. Conclusion ?

Si on alloue 50 bits par trame aux coefficients a_i et à σ_e, quel est le gain de codage, *i.e.* le rapport entre le débit binaire initial et le débit binaire avec codage ?

1. Il n'est pas nécessaire de relancer tout le programme, vous pouvez juste surligner les lignes à relancer (quantification et écriture du fichier) et taper `ctrl-e`.

2 Codage perceptif

Ouvrez le fichier *test_percept.sce*. Ce programme crée un vecteur x qui contient les échantillons d'un fichier son de fréquence d'échantillonnage 44,1 kHz.

Il réalise ensuite une analyse temps-fréquence, qui consiste à calculer le spectre du signal par tranches de 1024 échantillons se recouvrant à 50%. Pour chaque tranche analysée, il calcule la transformée en cosinus discret modifiée (MDCT), définie par 512 échantillons fréquentiels et dont la valeur absolue correspond au spectre d'amplitude sur les fréquences positives. Ainsi, un signal x de $N_{mdct}.N_{blocs}$ échantillons est représenté (sans perte d'information) par son spectrogramme XX qui est une matrice de dimensions $N_{mdct} \times N_{blocs}$.

a) Le spectrogramme d'un son a généralement la particularité de contenir de nombreux zéros. Une première idée de compression est donc de ne coder que les valeurs non nulles du spectrogramme. Exécutez le programme et notez la valeur du taux initial de zéros (en pratique, de valeurs proches de zéros) dans XX. Quel est alors le taux de compression possible ?

b) Nous allons mettre en œuvre un codeur perceptif sur le principe suivant : on codera le spectrogramme, en mettant à zéro (et en ne codant pas) toutes les valeurs du spectre sous le seuil de masquage.

Décommentez la ligne 26 et exécutez-la. Cette fonction affiche, pour chaque tranche de 1024 échantillons du signal, le spectre d'amplitude (en bleu) et le seuil de masquage (en rouge). Le dernier paramètre de la fonction règle le délai (en s) entre 2 figures successives. D'après cette visualisation, à quel taux de compression peut-on s'attendre ?

c) Remettez en commentaire la ligne 26 et décommentez les lignes suivantes. Le programme met à zéro les éléments de XX sous le seuil de masquage, créant un spectrogramme YY. Un nouveau signal, y, est reconstruit à partir de YY, puis sauvegardé dans le fichier *src6z.wav*. Enfin, le programme affiche le nouveau taux de zéros dans le spectrogramme.

Calculez le taux de compression, en considérant que :

— sans codage chaque échantillon temporel est codé sur 16 bits ;

— avec codage, chaque échantillon fréquentiel non-nul est codé sur 16 bits, mais il faut aussi ajouter une matrice binaire de dimensions $N_{mdct} \times N_{blocs}$ indiquant quels échantillons sont codés.

Ecoutez *src6z.wav* et *src6.wav*. Qu'entendez-vous ? Si la qualité des deux sons est différente, vous pouvez changer le dernier paramètre de la fonction *miseAzero*, qui indique de combien de dB abaisser le seuil de masquage pour sélectionner les coefficients du spectrogramme. Règlez ce paramètre de manière à ne plus entendre de différence entre les deux sons. Recalculez alors le taux de compression.

d) Refaites l'expérience avec différents fichiers sons (fichiers *src*.wav*).