Traitement de l'audio

OIM - TD3

2014

1 Codage de l'information

Soit un signal audio stéréo codé en 16 bits et échantillonné à 44100 Hz.

1. Combien d'échantillons y-a-t-il dans une seconde de signal ?
2. Quelle est la valeur de la fréquence maximale possible dans ce type d'enregistrement ?
3. Quelles sont les valeurs minimales et maximales possibles pour ces échantillons ?
4. Calculer la taile qu'occuperait en mémoire une heure de ce type d'enregistrement.

2 Echantillonnage et quantification

2.1 Sous-échantillonnage

On considère que nous disposons d'un signal audio monophonique, numérisé et stocké sous la forme d'un tableau d'entiers. Écrire une fonction C qui sous échantillonne ce signal en le moyennant par fenêtre glissantes.

2.2 Quantification

On considère que nous disposons d'un signal audio monophonique, numérisé et stocké sous la forme d'un tableau d'entiers. Écrire une fonction C qui quantifie ce signal en fonction du nombre de bits N désiré.

3 Calcul des moments d'un signal audio

Nous disposons d'un enregistrement audio monophonique de 5 secondes échantillonné à 22 kHz avec 8 bits par échantillon. Nous souhaitons développer un opérateur en langage C efficace pour calculer les moments d'ordre k de ce signal, par fenêtre glissante de 30 ms avec un recouvrement entre les fenêtres de 15 ms. Nous rappelons que le moment d'ordre k (M_k) d'un vecteur de N éléments est définit par :

$$M_k = \frac{1}{N} \sum_{i=0}^{N} x_i^k$$

1. Donner la taille du vecteur initial permettant de stocker le signal brut.
2. Donner la taille et le type du vecteur final permettant de stocker les moments sans perte de précision.
3. Écrire le code C de la fonction permettant de calculer le moment d'ordre k d'une fenêtre de longueur t ms.
4. Écrire le code C de la fonction permettant de calculer le vecteur des moments d'ordre 4 sur l'ensemble du signal avec une taille de fenêtre de 30 ms et un recouvrement de 15 ms.
5. Afin de minimiser les temps de calcul, nous souhaitons supprimer les calculs redondants des fonctions précédentes. Modifier vos fonctions précédentes afin qu'elles calculent le plus efficacement possible le vecteur des moments.

6. Préciser quelles sont les contraintes sur l'opérateur appliqué sur la fenêtre permettant d'utiliser ce type d'optimisation. Par exemple, aurions-nous pu les utiliser avec un opérateur qui calcule la médiane sur une fenêtre ?

4 Construction d'un spectrogramme à partir d'un signal audio

Un spectrogramme est une représentation par un diagramme temps-fréquence d'un signal. En abscisse de ce diagramme, on retrouve l'affichage temporel, et en ordonnée sont représentées les fréquences composant le signal. Un spectrogramme est très utile pour visualiser l'évolution des fréquences au cours du temps. Pour construire un spectrogramme, il faut extraire une portion de signal, appliquer un fenêtrage (par Hamming par exemple) sur cette portion puis calculer la transformée de Fourier. Vous avez à votre disposition les fonctions suivantes :

— `double * wavread(char * nom_fichier_wav, int *taille);` charge un fichier .wav et qui retourne un vecteur contenant les valeurs du signal ainsi que sa taille.
— `double * hamming(double *signal, int taille) };` réalise le fenêtrage de Hamming du signal passé en paramètre. La taille du vecteur renvoyé est identique au vecteur signal passé en paramètre.
— `double * rfft(double *vecteur_signal, int taille);` renvoi le module du spectre. La taille du vecteur renvoyé est identique au vecteur signal passé en paramètre.
Rappel : la fonction produit un résultat symétrique, seule la moitié du vecteur résultant sera à utiliser pour le spectrogramme.
— `void affiche_matrice(double * matrice, int largeur, int hauteur)` : affiche une matrice en niveau de gris. Les entiers `largeur` et `hauteur` spécifient la dimension de la matrice.

1. Si votre signal fait 36296 échantillons de long et que vous utilisez une fenêtre de 1024 échantillons de long, avec un recouvrement entre chaque fenêtre de 512 échantillons, combien de fenêtres allez vous pouvoir extraire du signal ? (Si une fenêtre ne peut être extraite en entier, nous ne la considèrerons pas).
2. Écrire la fonction
`double * spectrogramme(double *signal, int longueur_fenetre, int recouvrement)` qui calcule le spectrogramme du signal passé en argument. Cette fonction retourne une matrice dont chaque colonne correspond au calcul d'un spectre sur une fenêtre d'analyse. La longueur de la fenêtre de signal à utiliser est spécifiée par l'entier `longueur_fenetre`. Le recouvrement entre chaque fenêtre est indiqué par l'entier `recouvrement`.
3. A l'aide des fonctions précédentes, écrire un programme principal qui va charger le signal, calculer le spectrogramme et l'afficher.