UE SI350 :Travaux Pratiques sur l'indexation audio

Gaël RICHARD - Miguel ALONSO

1 Détection du tempo et Mixage audio avec ajustement rythmique

1.1 Introduction

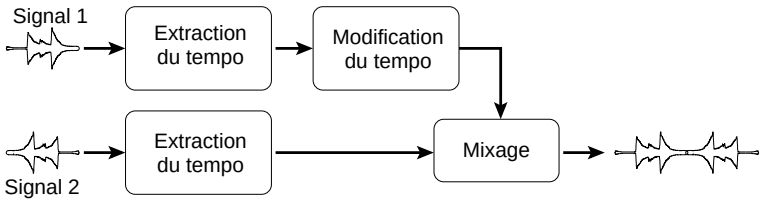

Figure 1: Mixage avec ajustement du tempo

L'objectif est de réaliser un algorithme de détection du tempo. Ceux qui auront du temps ou qui souhaiteront aller un peu plus loin pourront réaliser un module de mixage audio permettant de passer de manière souple entre deux morceaux de musique successifs en adaptant progressivement le tempo du premier morceau à celui du second morceau (voir figure 1).

Le fichier `tp_index_audio.zip` contient un ensemble de programmes Matlab utiles ainsi que quelques signaux courts (voir paragraphe 1.4).

1.2 Réalisation du module de détection du tempo

Il s'agit de réaliser un module de détection du tempo. Ce module doit être capable de retrouver l'organisation musicale interne d'un temps (l'intervalle allant d'un *beat* à l'autre), en termes des relations temporelles et de position des accents. Sous Matlab, ce module sera une fonction `function tempo=extract_tempo(sig1,Fs)` qui retournera la valeur du `tempo` en fonction du signal `sig1` et de la fréquence d'échantillonnage `Fs`. Le schema de fonctionnement de l'algorithme de détection du rythme contient les étapes suivantes (voir figure 2)

- **Fenêtrage:** on extraira une fenêtre de signal de longueur `wlen` correspondant à 10 secondes du signal. Si le signal d'entrée `sig1` est plus court que 10 secondes, la totalité du signal d'entrée sera utilisé.

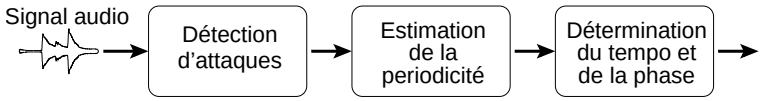

Figure 2: Schema de fonctionnement du système d'extraction du tempo

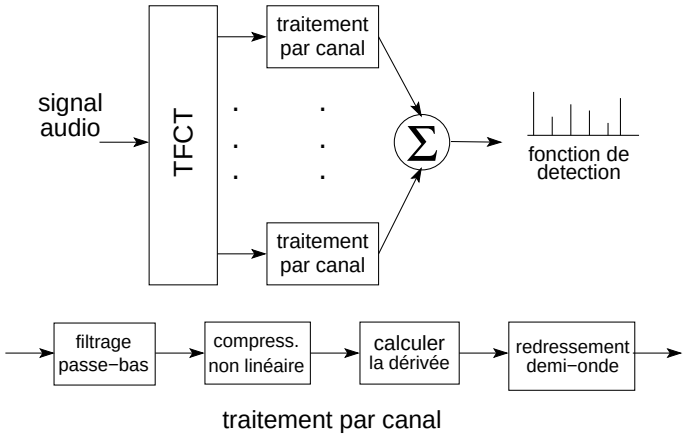

Figure 3: Principe de fonctionnement de l'extraction d'attaques

- **Détection d'attaques :** On développera ensuite un module de détection d'attaques (*onsets* en anglais). Celui-ci a pour but de fournir l'emplacement temporel des notes musicales dans le signal audio. Pour cela, on écrira une fonction `function [fonc_det,Fs2] = fonc_detection(sig1, Fs)` qui calculera la *fonction de détection* échantillonnée à la fréquence d'échantillonnage `Fs2` à partir du signal musical `sig1` échantillonné à une cadence `Fs`. Le calcul de cette fonction de détection utilise la notion de flux spectral (dérivée par rapport au temps de la transformée de Fourier à court terme). Les différentes étapes pour la détection d'attaques sont résumées sur la figure 3):

  - **Calcul du spectrogramme**: Analyser la fonction `spectro.m` qui est donnée et qui permet d'obtenir le spectrogramme du signal `sig1`. En particulier, on s'attachera à préciser:

    1. ce que représente le coefficient `kc`.
    2. la taille de fenêtre (`lfen`) et la valeur du paramètre FFTSIZE qui vous paraissent raisonnable ?
    3. ce que représente la variable `win_shift` et quel est son intérêt dans le calcul du spectrogramme?

  - **Traitement non-linéaire par canal:** Afin d'amélioration la détection des attaques, il est important d'effectuer plusieurs étapes de prétraitement dans chaque canal de TFCT (voir bas de la figure 3).

    1. *Filtrage passe-bas*: De manière générale, nous recherchons une information rythmique (tempo) qui possède une périodicité à basse-fréquence (< 10 Hz), ainsi la première étape est un filtrage passe-bas. Néanmoins, il est nécessaire, malgré ce filtrage de conserver l'impulsivité des attaques. On utilise pour cela un filtre dont la réponse impulsionnelle est l'arche descendante d'une demi-fenêtre de Hanning dont la discontinuité aide à préserver l'allure de l'attaque (voir figure 4 pour une fenêtre volontairement trop grande pour faciliter la visualisation). Quelle doit être la taille (en échantillons) de la réponse impulsionnelle pour obtenir une fréquence de coupure aux alentours de 10 Hz? Quelle est la nouvelle fréquence d'échantillonnage à l'intérieur de chaque canal de la TFCT?

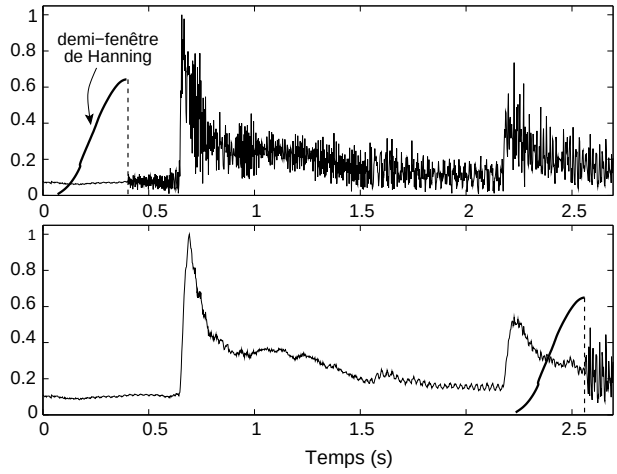

Figure 4: Filtrage passe-bas avec une demi-fenêtre de Hanning

2. *Compression dynamique*: La sensibilité du système auditif humain aux variations de l'intensité sonore peut être approximée en utilisant une loi de compression logarithmique. Ainsi, la dynamique du signal à l'intérieur de chaque canal est compressée à travers une conversion en décibels (dB) ($y^k_{\text{dB}}(t) = 10 \cdot \log_{10}(a^k(t))$ où $a^k(t)$ est l'amplitude du signal du k^{ieme} canal issu de l'analyse par TFCT).
3. *Calcul du flux spectral*: Le flux spectral est calculé en dérivant les signaux compressés de chaque canal de l'analyse par TFCT. On utilisera pour cela le filtre différentiateur dont la réponse impulsionnelle est donnée dans le fichier *fdiff.mat*. Vérifier que c'est bien un filtre différentiateur. Introduit-il un retard ?
4. *Redressement demi-onde*: En pratique, on ne souhaite détecter que les attaques de notes (partie positive de la dérivée) et non à détecter les fins de notes (partie negative de la dérivée). Le redressement demi-onde consistera ainsi à annuler les valeurs négatives du flux spectral.

La *fonction de détection* est ensuite obtenue en sommant la contribution de tous les canaux fréquentiels.

- **Détection de périodicités:**

1. *Lissage de la fonction de détection*: La première étape consiste à lisser la fonction de détection. On peut, pour cela utiliser le même filtre passe-bas (demi-fenêtre de Hanning) déjà utilisé précédemment (Voir figure 5 pour un exemple de fonction de détection après lissage).
2. *Calcul du tempo:* On écrira une fonction `tempo = calcule_periodicite(fonc_det,Fs2)` qui donnera le tempo à partir de la fonction de détection échantillonnée à $Fs2$. Le calcul de la périodicité s'effectuera en recherchant le maximum de l'autocorrélation normalisée dans un interval [`Tmin - Tmax`] que l'on déterminera. (*Variante possible pour ceux qui ont du temps: Des études dans la perception du rythme montrent que nous avons une préférence pour les tempi proches de 120 battements par minute (BPM). Il est ainsi possible de rechercher parmi tous les pics de l'autocorrélation celui qui répresente le mieux le rythme musical (selon la perception humaine). Pour cela,*

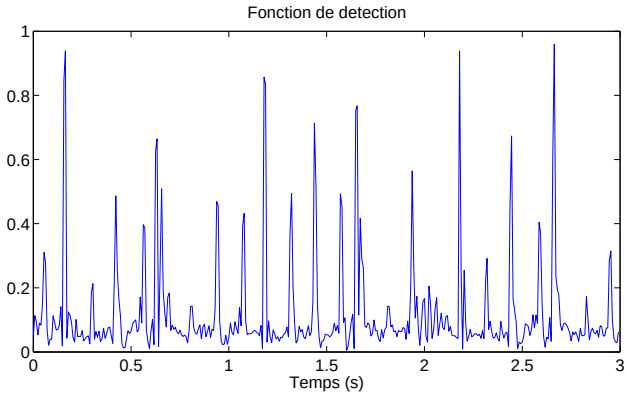

Figure 5: Aspect général de la fonction de détection. Les pics indiquent les instants où le changement du contenu fréquentiel est plus élevé.

on utilise une courbe de pondération perceptuelle que l'on pourra obtenir avec le program `ponderation_rayleigh` *fourni avec le TP. Son utilisation est :* `[poids,lags] = ponderation_rayleigh(Tmin,Tmax,Fs2)`*, où* `poids` *donne la pondération correspondante au décalage indiqué par* `lags` *(voir courbe rouge dans la partie haute de la figure 6). Le tempo est alors donné par le maximum de l'autocorrelation pondérée entre Tmin et Tmax*).

1.3 Réalisation du module de mixage (Optionnel)

Le module de mixage va consister à effectuer un passage souple entre les deux morceaux successifs. On réalisera une fonction `[mix1,sig_mod]=mixage(sig1,Tempo1,sig2,Tempo2,Fs)` qui implémentera ce module. Il est ici proposé de jouer sur 3 paramètres qui permettront d'adoucir la transition:

Mixage par OverLap and Add (OLA) Le principe consiste à additionner la fin du premier signal fenêtré par l'arche descendante d'une demi-fenêtre de Hanning au début du second signal fenêtré par l'arche montante d'une demi-fenêtre de Hanning (voir figure 7). On définira pour cela un paramètre `lmix` qui réglera la durée de la transition entre les deux signaux.

Ajustement progressif du tempo On utilisera pour cela la fonction `anasyn_psola` donnée (programme `anasyn_psola.m`) qui réalise une modification de l'échelle temporelle par l'algorithme PSOLA. Cette fonction est appelée sous la forme `sig1m=anasyn_psola(sig1, Tempo1,Tempo2,Fs);` où `Tempo1` est le tempo du premier signal, `Tempo2` le tempo du second signal et `Fs` la fréquence d'échantillonnage. `sigm1` est le signal modifié temporellement que l'on mixera avec le signal 2 à l'aide de la fonction de mixage par Overlap and Add.

Ajustement du point de jonction Afin d'améliorer le mixage, on se propose de calculer un meilleur point de raccordement de telle sorte que la rupture dans la périodicité du tempo soit moins sensible. Pour cela, on prendra les deux dernières secondes du premier son modifié (`sig1m`) et les 2 premières secondes du deuxième signal (`sig2`). On calculera leur fonction de détection (`fonc_det`) (en utilisant la méthode implémentée aux paragraphes antérieures). On calculera ensuite le meilleur point de jonction en recherchant le maximum

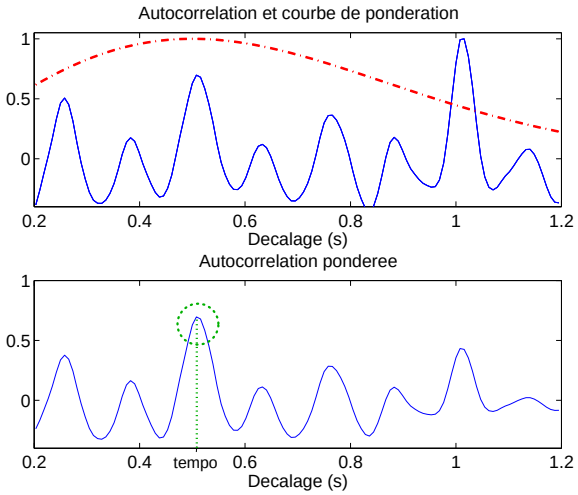

Figure 6: En haut : autocorrélation et courbe de pondération. En bas : autocorrélation pondérée.

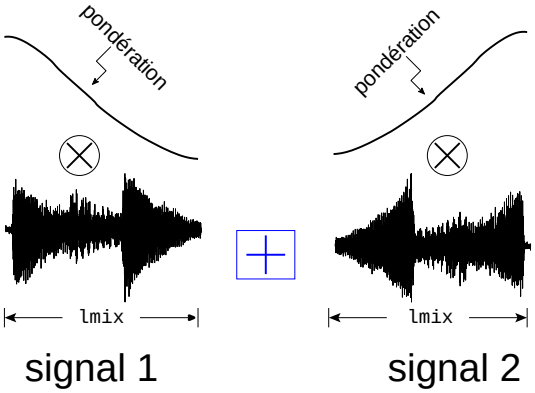

Figure 7: mixage OLA

de la corrélation entre `fonc_det1m` du premier signal modifié et `fonc_det2` du second signal. Le maximum de cette inter-correlation nous donnera finalement l'indice de début du `sig2` que l'on doit considérer.

Dans certains cas, les tempos des deux morceaux à mixer sont très différents. Que pourriez vous faire pour garantir un mixage au mieux dans ce cas ?

1.4 Signaux et programmes fournis

Le fichier `tp_index_audio.zip` contient un ensemble de programmes Matlab et quelques signaux courts.

- `sig1.wav` et `sig2.wav` le deux signaux initiaux à considérer
- `sig3.wav`, `sig4.wav`, `sig4.wav` qui sont trois autres signaux

On privilégiera l'utilisation les signaux sig1, et sig2 pour mettre au point les différents modules.