TP11 – Reconnaissance musicale

Empreinte sonore d'un signal acoustique

La réalisation d'un système de reconnaissance musicale nécessite de définir une « empreinte » qui puisse caractériser chaque enregistrement musical sans ambiguïté. Cette empreinte doit être :

- suffisamment **spécifique** pour que quelques secondes d'un enregistrement suffisent à l'identifier ;
- **robuste** à des transformations simples, comme par exemple l'ajout d'un bruit parasite ;
- de **taille réduite**, afin de faciliter le stockage ;
- **facile à calculer et à comparer** pour pouvoir « passer à l'échelle ».

Comme les données brutes (onde acoustique) et les sonagrammes ne respectent pas l'ensemble de ces critères, il nous faut trouver une empreinte plus adaptée.

L'idée sur laquelle se fonde `Shazam` est d'utiliser les *pics spectraux* (maxima locaux du spectrogramme) pour créer une *empreinte sonore* propre à chaque enregistrement musical. Concrètement, soit y un signal musical et Y sa TFCT. Un point (m_0, k_0) est considéré comme un pic spectral si, $\forall m \in [m_0 - \tau, m_0 + \tau]$, $\forall k \in [k_0 - \phi, k_0 + \phi]$:

$$|Y(m_0, k_0)| \geq \max\{|Y(m, k)|, \epsilon\} \tag{1}$$

où les paramètres τ et ϕ permettent de définir la largeur temporelle et la largeur fréquentielle de la fenêtre utilisée, et donc de contrôler la densité de pics souhaitée. Quant au paramètre ϵ, il permet de définir un seuil en dessous duquel les maxima ne sont plus considérés comme des pics.

Il s'avère que les pics spectraux les plus robustes se situent dans les basses fréquences. Pour cette raison, dans `Shazam`, les enregistrements sont ré-échantillonnés à $f_e = 8\ kHz$, afin d'alléger la suite de l'algorithme sans perdre en robustesse.

Exercice 1 : calcul des pics spectraux

Faites une copie de la fonction `TFCT` du TP10, puis lancez le script `verif_TFCT` en guise de validation. Écrivez ensuite la fonction `pics_spectraux`, appelée par le script `exercice_1`, qui est censée calculer les pics spectraux d'un sonagramme. L'affichage produit par ce script doit être identique à celui de la figure 1. Le calcul de la TFCT est effectué avec une fenêtre de Hann de taille 512 (qui détermine la résolution fréquentielle) et un recouvrement de 256 (qui détermine la résolution temporelle). Ceci permet d'obtenir des indices fréquentiels compris entre 1 et 257, qui pourront être stockés sur 8 bits (à une valeur près).

Il est conseillé de tenir compte des indications suivantes :

- La fonction `imdilate` de Matlab peut être utilisée pour détecter les maxima locaux (`doc imdilate`).
- Les valeurs $\tau = 30$ et $\phi = 30$ correspondent, après conversion, à $\tau \approx 1\,s$ et $\phi \approx 500\,Hz$.
- La valeur de ϵ est fixée à $1\,dB$.

Exercice 2 : appariement des pics spectraux

Les pics spectraux obtenus ainsi ne sont pas assez spécifiques pour permettre une bonne indexation, et donc une recherche suffisamment rapide. Pour pallier ce problème, `Shazam` détecte les *paires de pics spectraux voisins* : chaque pic est apparié avec les n_v pics les plus proches temporellement et fréquentiellement. Concrètement, pour que deux pics (m_i, k_i) et (m_j, k_j) soient appariés, il faut qu'ils vérifient les critères suivants :

- $0 < m_j - m_i \leq \delta_t$
- $|k_i - k_j| \leq \delta_f$

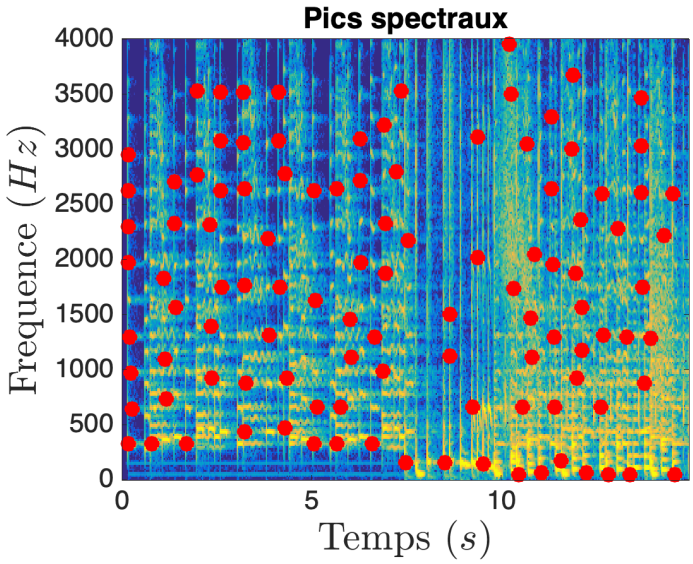

FIGURE 1 – Pics spectraux détectés dans les 15 premières secondes d'un morceau de musique.

Écrivez la fonction `appariement`, appelée par le script `exercice_2`, permettant de former des paires de pics spectraux voisins. Cette fonction doit retourner un tableau dont chaque ligne, composée du quadruplet (k_i, k_j, m_i, m_j), correspond à une paire.

Il est conseillé de tenir compte des indications suivantes :
- Les paramètres δ_t et δ_f sont fixés à 90, ce qui correspond à $3\,\tau = 3\,\phi$, puisque $\tau = \phi = 30$.
- Le paramètre n_v est fixé à 5, qui constitue un bon compromis entre robustesse et place mémoire.

Indexation des paires de pics spectraux

Une fois ces appariements effectués, il est nécessaire de trouver une *indexation* permettant une recherche rapide. Il serait possible de choisir comme *identifiant* le couple (k_i, k_j) des deux fréquences formant une paire. Si les extraits à reconnaître ne sont pas soumis à des dilatations temporelles, la valeur $(m_j - m_i)$ est elle aussi caractéristique d'une paire de pics spectraux.

Voilà pourquoi, dans `Shazam`, chaque paire (k_i, k_j, m_i, m_j) est indexée par le triplet $(k_i, k_j, m_j - m_i)$. Les valeurs k_i et k_j sont stockées sur 8 bits, tandis que la valeur $m_j - m_i$ est stockée sur 16 bits. L'ensemble, qui occupe donc 32 bits, constitue l'identifiant de la base de données. À cet identifiant sont associés la valeur m_i représentant l'instant du premier pic de la paire, relativement au début du morceau, et le numéro du morceau, qui permet de récupérer son nom, son interprète, etc. (cf. table 1).

$(k_i, k_j, m_j - m_i)$	m_i	num
$(23, 45, 52)$	6	1
⋮	⋮	⋮
$(174, 154, 84)$	498	73

TABLE 1 – Structure de la base de données utilisée pour la reconnaissance musicale.

La fonction `indexation`, qui vous est fournie, permet de calculer les identifiants associés aux paires de pics spectraux voisins. Cette fonction retourne un tableau dont chaque ligne contient l'identifiant d'une paire de pics spectraux. Avec une fenêtre de taille 512, les indices des fréquences vont de 1 à 257. Par conséquent, si l'indice 257 est « confondu » avec l'indice 1, il est possible de stocker l'indice d'une fréquence sur seulement 8 bits. Par ailleurs, l'utilisation de la fonction `uint32` vient de ce qu'un identifiant est une chaîne de 32 bits.

Reconnaissance musicale

Il est maintenant possible de construire une base de données : lancez le script `creation_bdd`, qui crée un fichier de nom `bdd.mat` contenant les empreintes d'une centaine de morceaux de musique de format WAV, calculées et rangées selon la procédure décrite ci-dessus. Pour reconnaître un extrait, il suffit de calculer son empreinte et de la comparer à celles de la base de données.

Plus précisément, pour chaque paire de pics $((m_i, k_i), (m_j, k_j))$ de l'extrait, toutes les entrées indexées par la chaîne $(k_i, k_j, m_j - m_i)$ sont recherchées dans la base de données, ce qui revient à chercher pour quels morceaux et à quels instants on passe d'une fréquence k_i à une fréquence k_j en $m_j - m_i$ secondes.

Une version simplifiée de cette recherche consiste à identifier le morceau de la base ayant le plus de paires en commun avec l'extrait analysé, mais cette façon de procéder ne prend pas en compte les instants d'apparition des paires. Une recherche plus rigoureuse est proposée dans l'exercice facultatif.

Exercice 3 : reconnaissance musicale simplifiée

Écrivez la fonction `recherche_simplifiee`, appelée par le script `exercice_3`, qui reçoit en entrée la liste des paires de pics de l'extrait et renvoie, pour chaque occurrence dans la base de données de chacune de ces paires, le numéro du morceau associé. Le script `exercice_3` affiche le nom du morceau le plus souvent cité.

Il est conseillé de tenir compte des indications suivantes :

- La fonction `recherche_simplifiee` reçoit deux arguments en entrée : la liste des *identifiants* des paires de pics de l'extrait, et la base de données.
- La base de données est une structure de données de type *containers.Map*, propre à Matlab, dont chaque clé est un identifiant, associé à un tableau répertoriant les différentes occurrences de cet identifiant (`doc containers.Map`).

Une fois cette fonction validée, lancez le script `statistiques`, qui calcule le pourcentage de bonnes reconnaissances. Ce pourcentage doit avoisiner 92 %, ce qui peut paraître décevant, en comparaison des performances spectaculaires de l'application `Shazam`. En revanche, testez la robustesse de ce système de reconnaissance musicale à un bruit additif (blanc ou de parole) : vous observez que le pourcentage de bonnes reconnaissances reste stable jusqu'à un « rapport signal sur bruit » (SNR, pour *signal-to-noise ratio*) égal à 10, voire en deçà.

Exercice 4 : reconnaissance musicale avancée (facultatif)

L'algorithme de reconnaissance musicale simplifiée ne tient pas compte du fait qu'il doit exister une cohérence entre les instants d'apparition des paires de pics de l'extrait et ceux du morceau présent dans la base. En effet, si un extrait débute à l'instant t, relativement au début du morceau, alors tout pic (t_i, f_i) de l'extrait doit apparaître dans le morceau à l'instant $t_i + t$.

Pour augmenter le pourcentage de bonnes reconnaissances, il faut imposer cette cohérence en effectuant la même recherche que précédemment, mais en veillant à stocker, pour chaque résultat, non seulement le numéro du morceau, mais également la différence temporelle entre l'apparition de la paire dans le morceau intégral et son apparition dans l'extrait. Cette différence temporelle constitue une origine potentielle de l'extrait, relativement au début du morceau intégral.

Faites une copie du script `exercice_3`, de nom `exercice_4`, dans laquelle vous remplacerez l'appel à la fonction `recherche_simplifiee` par un appel à la fonction `recherche_avancee`, qu'il vous faut écrire. Cette fonction doit recevoir trois arguments en entrée : la liste des *identifiants* des paires de pics de l'extrait, les instants d'apparition de ces paires dans l'extrait et la base de données.

Une fois ce script mis au point, vous devriez obtenir un meilleur pourcentage de bonnes reconnaissances que précédemment. Libre à vous de modifier les paramètres pour tenter d'obtenir 100 % de bonnes reconnaissances !