11451 BOURDIN Yann

Algorithmes de production et d'identification d'empreintes de sons musicaux

Passionné par la théorie et le traitement du signal, mon projet est la réalisation de deux algorithmes utilisant deux outils analytiques différents afin de les comparer. Ce sujet mêle les mathématiques et l'informatique et me permet d'adopter une démarche très personnelle.
Les interfaces sont d'une part physiques : le signal est acquis numériquement ; mais aussi mathématiques : ce sont les transformées, avec lesquelles on étudie les signaux dans d'autres "milieux" tels que le domaine temps-fréquence (Fourier) ou le domaine temps-échelle (ondelettes). Enfin, les interactions peuvent correspondre aux requêtes adressées à la base de données.

Positionnement thématique (phase 2)

INFORMATIQUE (Informatique pratique), MATHEMATIQUES (Analyse), SCIENCES INDUSTRIELLES (Traitement du Signal).

Mots-clés (phase 2)

Mots-Clés (en français)	**Mots-Clés** (en anglais)
Empreinte acoustique	*Acoustic fingerprint*
Spectrogramme	*Spectrogram*
Ondelettes	*Wavelets*
Base de données	*Database*
Analyse multirésolution	*Multiresolution analysis*

Bibliographie commentée

Un signal sonore est une fonction du temps : il correspond à la pression acoustique mesurée en un point en fonction du temps. Mais représenter le son sous sa forme d'onde nous masque beaucoup d'informations car elles sont étalées sur toute la durée du signal et peuvent ne présenter à première vue aucun forme régulière. L'analyse du signal consiste à en extraire l'information pertinente, qui sert à notre but. Dans le cas présent, cette information pertinente est l'ensemble des constituants de la musique analysée : ce sont ses notes de musique, jouées à certains instants, par certains instruments, formant sa partition. Ces constituants finissent mélangés, brouillés dans le signal ; il nous est nécessaire de passer à d'autres modes de représentation des signaux, de les décomposer, afin de récupérer l'information liée aux constituants (tels qu'ils étaient avant d'avoir été mélangés dans le signal final). [1,6]
Mathématiquement, appliquer une transformation à un signal, c'est lui associer une nouvelle fonction qui dépend d'une ou plusieurs « nouvelles » variables. L'outil le plus classique pour analyser les signaux acoustiques et qui s'est révélé particulièrement performant avec l'arrivée des ordinateurs est la transformation de Fourier : celle-ci décompose les signaux en sommes continues d'ondes sinusoïdales. La fonction transformée d'un signal nous renseigne ainsi sur le contenu fréquentiel du signal : on peut déduire du spectre du signal (diagramme amplitude-fréquence) quelles notes sont jouées dans ce dernier. Cependant, on masque cette fois l'information temporelle

: il n'apparaît plus à quel instant les notes sont jouées. Pour y remédier, on définit la transformation de Fourier à fenêtre glissante, qui consiste à décomposer un signal en fréquences (comme précédemment), mais intervalle par intervalle, à l'aide d'une fonction de fenêtrage. Notre signal est alors transformé en une fonction du temps et de la fréquence, il est alors possible de tracer le spectrogramme du signal. [1,3,6]

La transformation en ondelettes décompose les signaux en sommes continues de fonctions élémentaires correspondant aux dilatées-translatées d'une fonction appelée ondelette mère, dont les propriétés sont propices à l'étude de certains types de signaux ; ainsi, cette transformation peut s'avérer plus intéressante que la transformation de Fourier. De la même façon, on peut alors tracer le scalogramme du signal. [1,3,6]

Mon travail consiste en la réalisation d'un programme Python permettant de reconnaître le titre d'une musique à partir d'un extrait (de qualité variable). Dans un premier temps, je me suis intéressé aux travaux réalisés par le programmeur Roy van Rijn [5], qui a lui même utilisé le document *An Industrial-Strength Audio Search Algorithm* [2] issu de Shazam Entertainment, qui est l'entreprise à l'origine de Shazam, une application de reconnaissance musicale sur smartphone. Le processus détaillé dans ce rapport utilise la transformée de Fourier (discrète) à fenêtre glissante afin de produire des couples temps-fréquence (des points) correspondant à des pics d'intensité dans le spectrogramme du signal analysé : c'est l'information utile. A partir de ces points sont ensuite produites des empreintes : ce sont des chaînes de caractères obtenues, via une fonction de hachage, en associant les points d'une certaine façon entre eux. Cependant, ces méthodes utilisées par Shazam sont particulièrement complexes ; l'article [5] présente lui un processus plus simple, qui reste efficace et qui me servira de base pour mon algorithme. Dans un second temps, je m'intéresserai à l'algorithme Hokua [4], qui utilise la transformée en ondelettes (discrète), et plus précisément effectue une analyse multirésolution par ondelettes afin d'extraire l'information utile des musiques analysées. Communément aux deux algorithmes, il est nécessaire ensuite de produire des empreintes acoustiques à partir de cette information et de les stocker dans une base de données ; tout comme il faut pouvoir reconnaître une musique dont les empreintes sont dans une base de données à partir des empreintes d'un extrait. [2,5]

J'essaierai donc d'implémenter l'algorithme Hokua sous Python, en effectuant des simplifications si nécessaire, et comparerai ses performances avec le premier algorithme.

Problématique retenue

Il s'agit de réaliser un algorithme rapide et fiable : il doit pouvoir identifier une musique même si l'extrait analysé est de mauvaise qualité et ce en un temps relativement court (de l'ordre de quelques secondes), même avec une grande base de données de chansons.

Objectifs du TIPE

1. Étudier deux outils analytiques : la transformée de Fourier à fenêtre glissante, l'analyse multirésolution par ondelettes.
2. Déduire comment extraire l'information utile contenue dans les musiques analysées.
3. Chercher à compacter cette information intelligemment afin de minimiser le temps de calcul et le poids de l'information à stocker : produire des empreintes acoustiques.

4. Permettre l'enregistrement puis la recherche de cette information dans une base de données.

Abstract

In order to recognize a fragment of a song, we must have produced fingerprints of the song, saved in a database. The fingerprints must exist both in the original signal and in degraded versions: when it is recorded with a phone, for example. This information is obtained by representing our signals in the time-frequency domain (as a spectrogram) or in the time-scale domain (as a multiresolution analysis). By comparing the fingerprints of the fragment with those stored in our database through specific queries, we can determine which song is the most likely to match the fragment.

Références bibliographiques (phase 2)

[1] Barbara Burke Hubbard : Ondes et ondelettes : la saga d'un outil mathématique. : *Pour la science. 1995.*

[2] Wang Avery : An Industrial Strength Audio Search Algorithm. : *p7-13. Ismir. 2003.*

[3] B. PESQUET-POPESCU, J-C. PESQUET : Ondelettes et applications : *Techniques de l'ingénieur. 2001.*

[4] Steven S. Lutz : Hokua – A Wavelet Method for Audio Fingerprinting : *All Theses and Dissertations. Brigham Young University. 2009.*

[5] Roy van Rijn : Creating Shazam in Java. : *Site consulté régulièrement depuis juin 2017. http://royvanrijn.com/blog/2010/06/creating-shazam-in-java/*

[6] Yves Meyer, Stéphane Jaffard, Olivier Rioul : L'analyse par ondelettes : *Pour la science. Septembre 1987.*

DOT

[1] *Etude de la théorie de Fourier et de la théorie des ondelettes. Recherches autour des modules Python qui seront nécessaires à la réalisation de nos programmes. Sur Python : calcul du spectrogramme d'un signal, représentation graphique.*

[2] *Première approche. Ecriture des fonctions réalisant les tâches suivantes : découpe du spectrogramme en bandes, recherche de maxima dans chaque bande, création de nuages de pics que l'on stocke dans une base de données, similarité entre le nuage de pics d'un extrait et ceux des musiques stockées dans la base de données. Bons résultats, mais identification beaucoup trop longue et base de données trop lourde : nécessité d'une meilleure méthode.*

[3] *Etude de la méthode employée par l'application Shazam. Recherches autour de la méthode "Locality sensitive hashing".*

[4] *Deuxième approche. Ecriture des fonctions réalisant les tâches suivantes : recherche de maxima locaux dans une zone limitée en fréquence du spectrogramme, création de paires de maxima (de pics), hachage des paires que l'on stocke dans la base de données, comparaison entre les paires d'un extrait et celles des musiques stockées dans la base de données via une requête SQL. Essai de différentes formulations de la requête.*

[5] *Mauvais résultats : identification incorrecte. Essai d'autres façons de rechercher les pics et de*

les appairer. Essai d'un hachage moins strict en quantifiant les valeurs. Modifications légères de la fonction calculant le spectrogramme. Toujours des résultats non concluants. Retour en arrière à lorsque les valeurs n'étaient pas quantifiées, retour à un hachage relativement strict, essai d'une nouvelle façon d'appairer les pics.

[6] *Après des tests variés, la méthode fonctionne notamment lors d'un test n'utilisant pas la base de données. Découverte et résolution d'un bogue dans la requête SQL utilisée pour l'identification. Réussite du programme : résultats corrects, identification beaucoup plus rapide que lors de la première approche et base de données beaucoup plus légère. Réalisation d'améliorations diverses, par exemple : filtrage des pics, ajout d'une probabilité de conserver une paire, essai de différentes valeurs de certaines constantes...*

[7] *Réétude de la théorie des ondelettes, étude de l'algorithme Hokua. Recherches autour du module Python pyWavelets permettant de faire des calculs de transformées en ondelettes.*

[8] *Troisième approche. Ecriture des fonctions réalisant les tâches suivantes : prétraitement, calcul d'une analyse multirésolution (AMR) et représentation graphique, AMR "par blocs" et représentation graphique, recherche de maxima dans une AMR par blocs. Echec lors de la représentation graphique des maxima.*