Mémoire de master 2 ATIAM

Identification automatique de titres musicaux par *fingerprinting*

Agnès Pedone

Directeur de stage : Raphaël Blouet
Yacast, 4 rue Paul Valery 75016 Paris

Mars - Septembre 2012

Remerciement

Je voudrais remercier en premier lieu Raphaël Blouet, mon maître de stage chez Yacast, qui m'a guidée pendant ces 2 mois et demi sur ce sujet particulièrement intéressant.

Je souhaite également remercier Béatrice et Eva qui ont contribué à la relecture de ce rapport.

Resumé

L'identification automatique de titres musicaux fait l'objet de nombreuses recherches, en particulier dans le cadre de l'indexation de larges bases de données et du monitoring de flux de broadcast. La piste explorée dans ce rapport est celle du marquage des signaux audio par *fingerprinting* suivi de la mise en correspondance des marques d'un signal inconnu avec celles d'une large base de données, afin d'identifier le contenu en termes de titres musicaux. Le présent travail se place dans le cadre du contrôle de flux de broadcast : sur des diffusions continues généralement dégradées, on ne connaît ni l'identité des titres diffusés, ni leurs instants, ni leurs durées de diffusion.

Nous cherchons à comparer deux systèmes d'identification, différenciés principalement par leur technique de *fingerprinting*. Le premier se base sur l'extraction de pics spectraux qui sont associés par paires, ce qui permet de construire une constellation pour chaque signal. Il s'agit d'une proposition d'implémentation de Dan Ellis, basée sur ce que propose Wang dans [Wan03]. Le second est le coeur de notre contribution : il se base sur l'apprentissage d'un modèle de Markov caché, ce qui permet de représenter le signal de manière symbolique. La séquence de symboles est ensuite organisée sous forme de clés qui rassemblent trois symboles consécutifs afin d'introduire une information temporelle dans la marque elle-même. Cette seconde représentation a l'avantage de représenter le signal de manière plus informée et donc plus compacte.

Pour finir, nous évaluons les performances des deux systèmes d'identification, en ce qui concerne la robustesse au *time shifting* dans le cadre pour une petite base de données de signaux. Les résultats obtenus pour le système proposé sont très encourageants, et en particulier, ils sont très supérieurs à ceux du système de référence. Néanmoins, on remarque que les résultats sur les données réelles n'atteignent pas les résultats des extraits auxquels nous avons appliqué nous-même le *time shifting*. Enfin, nous présentons les résultats de performances pour le système de référence, basés sur le système d'évaluation PyAFE de H. Bredin, proposé dans le cadre du consortium européen Quaero pour les systèmes d'identification audio automatiques. Même si le temps imparti ne nous a pas permis d'évaluer le second système sur des données réelles, les tests unitaires réalisés du point de vue du *time shifting* nous encouragent à poursuivre dans la voie du système élaboré.

Note : le travail présenté n'a été réalisé que sur une période de 2 mois ½ au lieu de 5 pour des raisons de santé, c'est pourquoi nous n'avons pas pu réaliser tous les tests expérimentaux prévus.

Mots clés : indexation audio, recherche d'information (IR), analyse sémantique, modèle de Markov caché (HMM), fingerprinting audio, contrôle de flux de broadcast, apprentissage automatique, segmentation audio

Abstract

Automatic audio identification is an important research subject, particularly in the field of semantic indexing of large databases. In this report, we explore the solution of audio fingerprinting and matching of fingerprints of an unknown signal with a large database of audio excerpts, in order to identify the appearing audio titles. The present work is in the perspective of radio broadcast monitoring : on continuous broadcasting generally damaged, we know neither identity nor instant or duration of musical titles.

We aim to compare two identification systems, which principally differ by the fingerprinting technic. The first one is based on spectral peaks extraction which are associated by pairs in order to build a constellation for each signal to mark. It consists in the proposition of Dan Ellis to implement Wang's system, described in [Wan03]. The second one is our contribution : we rely on machine learning so as to obtain a statistical model of the audio signal, using Hidden Markov Model in order to represent the signal in a symbolic way. The sequence of symbols is then organized in keys which bring together 3 consecutive symbols, in order to introduce a temporal information in the fingerprint itself. This second solution has the advantage of representing the signal with a knowledge-based method which is more compact.

At last, we evaluate the compared performances of the two identification systems in terms of robustness to time shifting for a small database of signals. The obtained results for the proposed system are very encouraging because they are significantly higher comparing to those of the reference system. Nevertheless, we notice that when applying to real data do not reach those of the extract on which we apply ourself time shifting. Finally, we present the performance results for the reference system, based on the evaluation toolkit PyAFE, proposed by H. Bredin for the european research consortium Quaero, specifically for the automatic audio identification systems. Even if the given time did not allow us to evaluate the second system on real data, the unitary tests concerning time shifting encourage to continue in the direction of the proposed system.

Note : the work was achieved in 2 months 1/2 instead of 5, that is why we could not realize all expected experimental tests.

Keywords : audio indexing, Information Retrieval (IR), semantic analysis, Hidden Markov Model (HMM), audio fingerprinting, broadcast monitoring, machine learning, audio segmentation

Table des matières

Introduction

Yacast : présentation de l'entreprise

Yacast est une entreprise spécialisée dans le contrôle de flux multimédias. Elle offre un service de veille des programmes multimédias sur tous les supports de diffusion : radio, télévision, internet, discothèque, etc. Entreprise moyenne, elle développe en interne des logiciels et applications pour ses activités.

Les principaux produits de Yacast sont Muzicast, pour la veille des diffusions musicales, Advercast pour la veille du marché publicitaire, Talk pour la mesure du temps de parole politique, Club Monitoring pour la reconnaissance de la musique diffusée en discothèque.

Enfin, Yacast est un partenaire du consortium de recherche européen Quaero [1] qui rassemble différents industriels et laboratoires publics. Ce consortium a pour objectif de rassembler la recherche sur les technologies d'analyse automatique de contenu multimédias, ainsi que sur leur classification et sur leur utilisation.

Présentation du sujet de stage

La grande quantité de données disponibles et diffusées de manière continue sur tous les médias (radio, internet, télévision) pose le problème de l'exploitation efficace des contenus et du contrôle de sa diffusion. Dans le cadre du contrôle de flux multimédia, on cherche entres autres à identifier de manière robuste la donnée diffusée. Cette identification peut servir au contrôle des droits d'auteur, à la production de statistiques publicitaires, etc.

L'identification audio a pour but d'assigner son titre à une chanson diffusée. Dans le cadre de ce rapport, on s'intéresse à une identification par *fingerprinting* et par mise en correspondance d'une requête audio avec un élément d'une large base de données. L'identification par *fingerprinting* est basée sur l'extraction de caractéristiques qui décrivent de manière concise, unique et robuste les différents titres musicaux [2]. Cette représentation est basée sur des propriétés acoustiques.

1. http ://www.quaero.org/
2. En revanche, la technique du *watermarking* utilise des méta-données embarquées dans la donnée audio de manière transparente et robuste aux distorsions, afin d'identifier un titre musical. Aucune analyse sémantique n'intervient.

Applications et enjeux

Les contextes applicatifs de l'identification automatique de titres musicaux ont chacun des contraintes propres :

1. l'identification des titres dans les clubs. On dispose à la fois de l'entrée ligne du système du didjé et de la captation du signal diffusé en sortie. Le didjé peut modifier le son d'entrée (typiquement étirement temporel, décalage en fréquence, ajout d'écho, etc.), et la reconnaissance doit être robuste à ces distorsions.
2. l'identification des titres diffusés sur les médias classiques (radio, télévision). On dispose des enregistrements broadcast par tranches de 5 min. Il est nécessaire d'identifier le titre diffusé quelle que soit la durée de son extrait (on doit pouvoir reconnaître les citations), sa qualité, son niveau relatif à d'autres sources parasites (sources de parole par exemple), ses distorsions appliquées volontairement ou non de la part des diffuseurs. C'est dans ce contexte en particulier que les différents systèmes étudiés dans ce travail sont conçus et testés.
3. l'identification de titres musicaux dans le cadre d'application smart-phone : la tâche doit pouvoir être effectuée dans des milieux très perturbés sur des enregistrements de qualité médiocre, avec des échantillons de quelques secondes, en temps réel, avec peu de ressources computationnelles.

Les enjeux de l'identification sont de plusieurs types :

Robustesse[3]

- **au bruit** : reconnaissance en milieu bruité, on considère ici le bruit additif : les autres sources s'ajoutent linéairement au signal cible. Ce bruit peut simplement s'ajouter de manière "transparente", ou faire occlusion au signal. Typiquement, un signal de parole par-dessus la musique sera considéré comme un bruit additif
- **aux distorsions non linéaires** : reconnaissance de signaux auxquels on a appliqué des distorsions temporelles (étirement ou compression temporelle) ou fréquentielles (transposition en fréquence), une modulation de volume, un filtrage passe-bande, un ajout d'écho, une compression avec pertes, une modulation de l'égalisation fréquentielle, etc.
- **à la désynchronisation** : reconnaissance de citations. Globalement, on se trouve face au problème de la localisation de l'extrait dans le signal complet, et localement, un décalage temporel ne doit pas gêner la reconnaissance.

Ressources computationelles

- **mémoire et temps de calcul** : la reconnaissance s'effectue grâce à une large base de données, le temps de calcul et la mémoire doivent donc être minimisés afin d'éviter des temps de traitement aberrants, ainsi que des occupations en mémoire trop importantes.

3. Pour le cas spécifique des radios, et donc pour notre travail, la chaîne de traitements exacts appliqués au signal original est inconnue pour nous, nous savons seulement que les radios appliquent couramment les distorsions évoquées, comme en parlent [CBMN02, RFB+11]. Nous n'évaluons donc pas la robustesse à une distorsion particulière, mais à un ensemble de distorsions indissociées.

Présentation du développement

Dans le présent rapport, la première partie présente de manière approfondie le sujet avec la présentation de l'état de l'art des différentes techniques de *fingerprinting*. La deuxième partie expose le système que nous prenons comme référence pour évaluer l'apport du système, que nous proposons en troisième partie, système qui constitue le coeur de notre contribution. Enfin, la dernière partie présente les résultats de performance des deux systèmes.

Chapitre 1

État de l'art sur l'identification de titres musicaux par *fingerprinting*

On établit dans cette partie un bref état de l'art sur l'identification automatique de titres musicaux. On présente la technique du *fingerprinting* qui permet de marquer les signaux audios, et la mise en correspondance du signal "requête" avec un élément de la base de donnée.

1.1 Fonctionnement général

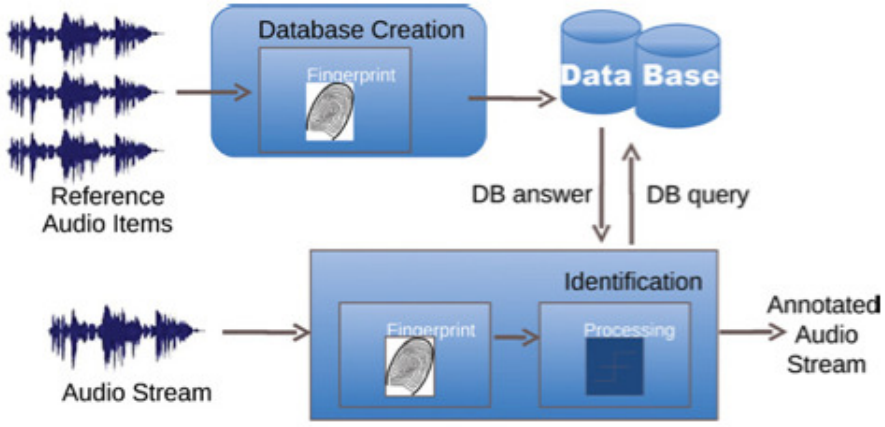


FIGURE 1.1 – Schéma du processus de l'identification audio par fingerprinting, extrait de [RFB+11].

L'identification audio se déroule en plusieurs phases (voir figure 1.1) :

1. à partir d'une base de données audio, on extrait les *fingerprints* associés. Différentes marques peuvent être générées, mais elle sont souvent basées sur une analyse temps-fréquence du signal (spectrogramme).
2. on construit une table de hachage avec tous les *fingerprints* générés pour toute la base de données audio. Il s'agit d'organiser les nombreux *fingerprints* calculés dans une table.

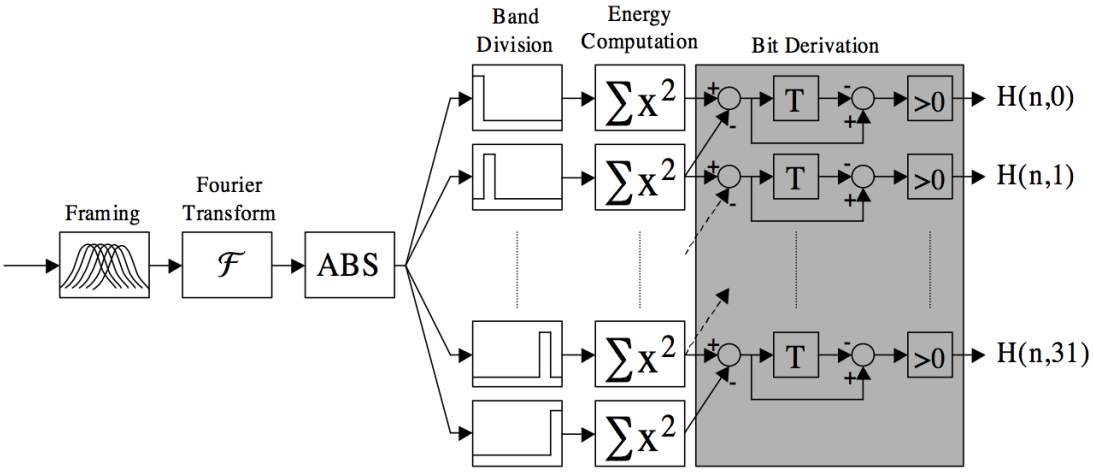


FIGURE 1.2 – Schéma du système de Philips [HKO01]

3. pour un signal "requête", on calcule ses *fingerprints* de la même manière que pour les éléments de la base de données.
4. on cherche les *fingerprints* de la base de données qui correspondent le mieux à ceux de la requête. Les algorithmes de mise en correspondance sont basés sur une recherche soit exacte, soit au plus proche voisin, soit statistique.
5. on identifie les titres qui apparaissent dans le signal-requête.

1.2 Marquage des signaux audio

On dispose du signal temporel brut, à annoter. On ne travaille pas directement sur cette représentation du signal car elle est peu robuste au bruit et aux distorsions. On passe donc dans le domaine fréquentiel, grâce à une STFT (*Short-Time Fourrier Transform*, transformée de Fourrier à court terme). La STFT analyse le signal par fenêtres temporelles, décomposant le spectre localement sur un certain nombre de bandes fréquentielles.

L'utilisation de cette représentation pour comparer les signaux est, d'une part, très coûteuse en mémoire et en complexité, et, d'autre part, peu robuste à une désynchronisation et à un étirement temporel, car elle ne représente aucune évolution temporelle. C'est pourquoi on va chercher une représentation plus compacte et symbolique. Voici les différentes représentations présentes dans la littérature :

Différence d'énergie des bandes fréquentielles

Le système de Philips (Haitsma et al.) s'appuie sur le calcul de l'énergie de chaque bande de fréquence. Afin de produire une clé binaire, il calcule ensuite la différence d'énergie entre les bandes de fréquence adjacentes. Il assigne un bit à 1 si cette différence croît entre 2 trames, et le bit est mis à 0 si la différence décroît ou stagne par rapport à la trame précédente (voir figure 1.2).

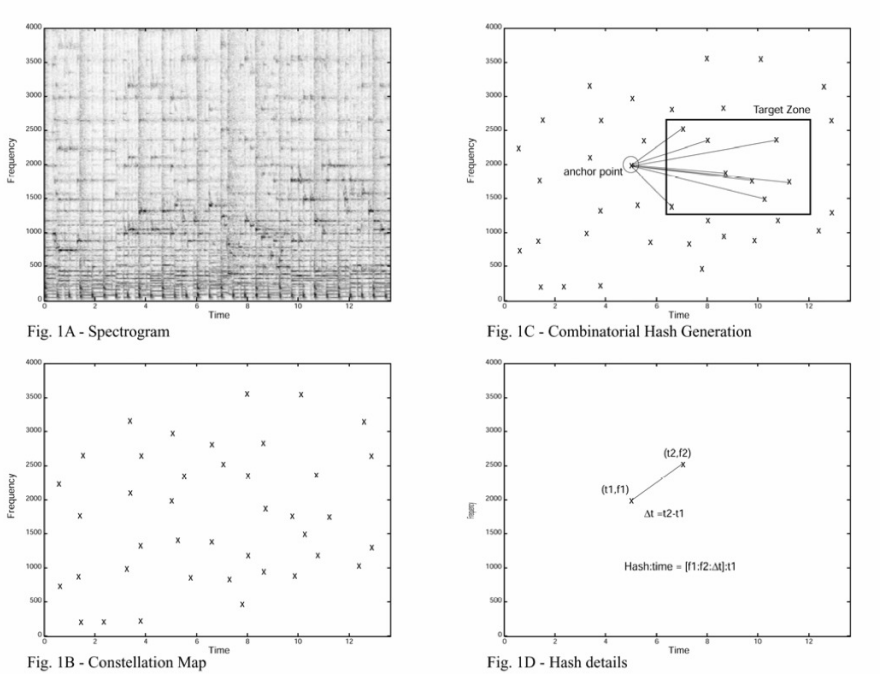


FIGURE 1.3 – Extraction des pics fréquentiels dans shazam [Wan03]

Extraction des pics spectraux

Wang, pour l'application Shazam [Wan03], propose une méthode empirique pour représenter le signal : au lieu de traiter le spectrogramme de manière linéaire temporellement, il se base sur une constellation de pics spectraux associés par paires. Ainsi un signal est-il représenté par un ensemble de paires indépendantes du point de vue temporel.

Pour construire cette constellation, il extrait d'abord des pics spectraux candidats qui sont les maxima de régions à forte densité spectrale, ce qui permet de garantir la continuité temporelle et fréquentielle. Le pic spectral de plus forte énergie est ensuite choisi dans des régions temps-fréquences définies. Seule la position temps-fréquence du pic est conservée, sans tenir compte de l'amplitude, afin d'obtenir directement une représentation binaire.

Pour construire les paires, on choisit certains points de la constellation, les *anchor points*, pour les associer à des zones cibles, les *target zones* (voir figure 1.3). À chaque anchor point est associée sa position temporelle par rapport au début du fichier, et chaque paire {anchor point,target point} est codée par 3 valeurs : 2 valeurs de fréquence absolues et 1 valeur de différence temporelle.

Cette représentation a l'avantage d'être très robuste à la quantification vectorielle. Elle est également robuste au bruit additif comme à l'occlusion.

C'est ce système qui sert de référence à notre évaluation. Il est donc plus détaillé en partie 2.

Modélisation statistique des signaux

Cano et al., dans [CBMN02], utilise une modélisation statistique des signaux. À partir de segments audio représentatifs, il extrait des MFCC à partir desquels il

FIGURE 1.4 – Ensemble de filtres candidats [KHS05] à appliquer au spectrogramme en tant qu'image. Les filtres sont choisis parmi ceux définis par Viola et Jones [VJ03] pour le traitement d'image.

apprend un dictionnaire de symboles. Il modélise acoustiquement les symboles par l'apprentissage d'un modèle de Markov caché (HMM, *Hidden Markov Models*). C'est l'algorithme de Viterbi qui permet ensuite d'aligner les signaux avec la séquence de symboles la plus probable. Nous choisissons cette représentation des signaux dans le système que nous proposons en 3.

Filtrage et transformations inspirés du traitement d'image

Dans [KHS05], Ke et al. utilisent des filtres appliqués au spectrogramme, s'inspirant des méthodes de traitement d'image : ils traitent le spectrogramme comme une image dont il faut détecter des "objets" caractéristiques. La banque de filtre qu'ils utilisent est basée sur les ondelettes de Haar, et la binarisation se fait par seuillage (cf. figure 1.4). Les filtres sont choisis afin de minimiser l'erreur de classification au cours de l'apprentissage, et, de la même manière, les seuils à appliquer sont appris de telle manière à obtenir une représentation binaire. Ke utilise un set de 32 filtres.

Cette représentation a l'avantage de conserver la propriété de similarité entre deux signaux.

Une autre solution est d'extraire des descripteurs de l'image du spectrogramme, invariants au changement d'échelle et à la rotation, comme le propose Zhu dans [Zhu10]. Il s'agit des descripteurs SIFT (*Scale Invariant Feature Transform*) détaillés par Lowe et al. [Low04].

1.3 Mise en correspondance

Pour établir une correspondance entre le signal requête et un élément de la base de donnée (*matching*), la méthode de base est de comparer leurs clés binaires respectives en calculant la distance de Hamming [1].

Néanmoins, si on a affaire à une large base de données, le calcul de la distance de Hamming avec tous les candidats de la base (i.e. tous les extraits potentiels de tous les éléments de la base de données) peut devenir beaucoup trop complexe. C'est pourquoi on cherche des stratégies de hachage moins coûteuses et plus efficaces.

De même, la recherche d'une correspondance exacte des clés est idéaliste si on admet une dégradation du signal. Les méthodes du plus proche voisin et les méthodes statistiques permettent de relâcher la condition de correspondance exacte.

Les différentes stratégies de mise en correspondance de la littérature sont les suivantes :

1. La distance de Hamming est définie comme le nombre de bits différents entre deux mots binaires.

Table de hachage et tolérance d'erreur

Afin de réduire la complexité du problème, Philips propose de construire une table de hachage sous la forme d'une *lookup table (LUT)* qui recense toutes les occurrences d'une clé dans les éléments de la base de données. Pour une valeur de clé, on a donc une liste d'extraits candidats (identification de la chanson, et localisation dans la chanson elle-même). Ensuite, c'est l'extrait qui a le plus faible taux d'erreur binaire (BER, Bit Error Rate) inférieur à un seuil ($alpha = 0.25$) qui est choisi.

Afin de considérer des erreurs de 1, 2 ou 3 bits dans la clé, [KHS05] propose de calculer successivement la distance de Hamming avec la clé entière des éléments de la base de données, puis toute les possibilités d'occlusion d'1 bit et de 2 bits. À chaque fois, on ne conserve que les candidats qui satisfont une distance de Hamming de resp. 0, 1 et 2.

Méthode du plus proche voisin (LSH)

La méthode du plus proche voisin est efficace dans le cas d'une faible dimension, mais quand la dimension augmente, ses performances deviennent similaires à celles d'une méthode linéaire exhaustive. Ce problème est appelé la course à la dimensionnalité (*curse of dimentionality*). En effet, le temps de recherche du plus proche voisin augmente exponentiellement avec la dimension des données.

On peut alors choisir une méthode approximative de recherche du plus proche voisin : on veut, dans ce cas, conserver la propriété de proximité, c'est à dire que lorsque deux extraits sonores sont similaires, on veut que leurs clés soient similaires, et ainsi la probabilité de collision est-elle plus grande pour deux extraits similaires. Pour cela, on utilise une famille de fonctions LSH (Locality-Sensitive Hashing) qui transforment les données dans un autre espace, en conservant la propriété de similarité.

Méthode statistique

Afin de prendre en compte la séquentialité temporelle des clés pour un signal, Ke et al. [KHS05] proposent d'assimiler l'enchaînement des clés à des objets, et d'utiliser la méthode RANSAC [FB81] utilisée en traitement d'image pour la reconnaissance d'objet. Cette méthode applique l'algorithme EM (Expectation Maximization) sur les différents alignements des candidats, pour apprendre le plus vraisemblable. Deux modèles d'alignement peuvent ensuite être adoptés : soit on considère que la donnée du décalage temporel de l'extrait suffit à aligner la requête avec l'original, soit on suppose que l'extrait peut avoir subi un *time shirting*[2].

Dans la même optique, Cano et al., dans [CBMN02], utilisent un algorithme d'alignement de séquences de caractères utilisés pour les alignements de chaînes d'ADN. Cet algorithme, FASTA [Gus97], opère d'abord un alignement local et exact de courtes sous-séquences, puis aligne globalement en utilisant des méthodes de mise en correspondance approximatives.

2. Dans ce dernier cas, on a besoin également de connaître le rapport d'étirement ou de compression en plus de l'offset temporel.

1.4 Conclusion

Nous avons fait ici un bref état de l'art en ce qui concerne le *fingerprinting* audio ainsi que les méthodes de mise en correspondance, dans le but de reconnaître des titres musicaux, dans des contextes peu favorables de bruit et de distorsion. Dans la première phase d'extraction des marques, on cherche finalement à décrire le signal de manière pertinente afin d'obtenir des marques à la fois compactes et robustes. Idéalement, la représentation du signal doit être assez spécifique pour limiter les collisions, mais aussi générique pour reconnaître un signal détérioré. Ensuite, des algorithmes de mise en correspondance permettent d'éviter une recherche linéaire et exhaustive, inefficace pour une large base de données. On a évoqué différentes stratégies qui permettent de limiter la complexité de la recherche.

Le système que nous proposons en partie 3 repose sur la description des signaux par une suite de symboles, comme présenté dans [CBMN02], car cette représentation a les propriétés intéressantes de robustesse et de concision. On combine cette représentation avec la formation de clés, comme le propose [HKO01], afin de coder une information temporelle dans la marque elle-même. Ces clés sont stockées ensuite dans une table de hachage. Avant de présenter ce système, nous présentons dans la partie suivante le système pris comme référence.

Chapitre 2

Système de référence

Dans ce chapitre, nous présentons le système qui nous sert de référence pour évaluer notre système. Nous avons utilisé le code fourni par Ellis [1], qui est une implémentation de ce que Wang propose dans [Wan03], déjà présenté brièvement en section 1.2, dans le paragraphe sur l'*Extraction des pics spectraux*.

2.1 Marquage des signaux audio

Afin de générer des *fingerprints* pour chaque signal (signature de la base de données ou flux à annoter), on cherche à créer une constellation de pics spectraux (voir 1.3). Pour cela, on procède comme suit :

1. **Extraction des maxima énergétiques locaux** dans l'espace temps fréquence. Après avoir calculé le spectrogramme, on calcule la magnitude logarithmique. Puis on applique une courbe de masquage aux maxima locaux et les pics énergétiques (*onsets*) sont relevés en amplitude par application d'un filtre passe-haut, ce qui permet d'éliminer les maxima locaux qui varient lentement. Seule la position temps-fréquence du pic est conservée, sans que l'amplitude soit pris en compte, afin d'obtenir directement une représentation binaire. L'analyse fréquentielle se fait sur des fenêtres de 64ms, avec un recouvrement de moitié.

   Cette phase dépend de trois paramètres principaux : la largeur et la décroissance de la courbe de masquage [2] (*masking skirt*), et le filtre passe-haut appliqué à la magnitude logarithmique.

2. **Construction des paires** : les repères ou *landmarks*. On associe à chacun des pics extraits (les points ancres, *anchor points*) une zone cible (*target region*). La taille de la zone cible influe directement sur le nombre de paires résultant. On notera que la densité de *landmarks* est fixée à 10 par seconde.

3. **Conversion des repères en valeurs de hachage**. Les repères sont sous la forme de quadruplets < t1 f1 f2 dt>, t1 étant l'instant d'apparition du premier pic, f1 et f2, les deux valeurs de fréquences, et dt la différence temporelle entre les deux

1. disponible sur `http://labrosa.ee.columbia.edu/matlab/fingerprint/`.

2. La courbe de masquage, ou courbe d'accord psychoacoustique, reproduit un effet psychoacoustique : un pic fréquentiel a un effet de masquage sur une zone fréquentielle proche. Les profils des masques dépendent de la hauteur fréquentielle et du niveau sonore.

éléments de la paire. Les valeurs de hachage sont sous la forme d'un triplet <songID, time, hash>, songID étant l'identifiant du titre musical, time étant égal à t1 (durée depuis le début du morceau). Hash est la valeur de hachage de 20 bits : 8 bits pour f1, 6 bits pour la différence fréquentielle entre les deux pics, et 6 bits pour dt. La représentation en valeur de hachage plutôt qu'en *landmarks* est plus compacte et adaptée à une organisation en table.

4. **Enregistrement des valeurs de hachage** dans la table de hachage. La table de hachage est ordonnée par valeurs de hachage. Le nombre de références vers lesquelles pointe une valeur de hachage est limitée à 20.

2.2 Mise en correspondance

Tout d'abord, l'algorithme de Wang met en correspondance un signal-requête avec l'un des éléments de la table de hachage. Voici les différentes étapes :

1. **Extraction des *landmarks***. On extrait les repères en construisant des paires de pics spectraux de la même manière que précédemment, pour la construction de la table de hachage. On notera que la densité (nombre de *landmarks* par seconde) est supérieure dans ce cas, et égale à 20 par seconde.
2. **Sélection des candidats**. On trie les candidats de la base de données qui contiennent le plus de valeurs de hachage en commun avec le signal-requête, en considérant la séquentialité temporelle.
3. **Identification du titre**. Le candidat qui a le plus de correspondances avec une signature est sélectionné grâce à un seuil : ce candidat est choisi si la portion du flux étudiée contient au moins n% des éléments de la signature du candidat. On se place pour cela sur des éléments temporels de 1min40, en se décalant de moitié à chaque itération. On fait donc l'hypothèse que les titres sont séparés d'au moins 50s. On a choisi la valeur de 1min40 de manière empirique, ayant observé que 2 chansons au plus apparaissent sur les segments de 5min disponibles.

Chapitre 3

Système proposé

3.1 Présentation du système

La solution de génération de *fingerprints* que nous étudions dans ce chapitre s'inspire de celle de Cano [CBMN02]. Plus précisément, dans l'optique de diminuer la taille des *fingerprints* nécessaire à une reconnaissance robuste, on cherche à apprendre un dictionnaire de symboles qui pourra décrire de manière compacte et pertinente les signaux audio. La table est construite à partir de la formation de clés qui sont le regroupement de 3 symboles consécutifs. L'algorithme de mise en correspondance repose sur un alignement local, puis global de clés.

Grâce à une large base de données, on cherche tout d'abord à construire des segments audio homogènes afin d'initialiser un modèle de Markov caché (HMM). Ce modèle permet d'apprendre de manière non supervisée un modèle gaussien pour chaque symbole, et un modèle markovien pour l'enchaînement des symboles.

Une fois le dictionnaire de symboles appris, pour chaque signal à représenter (signatures et flux), on obtient l'enchaînement le plus probable de symboles (au niveau de la trame de descripteurs) grâce à l'algorithme de Viterbi.

Pour les flux à annoter, un algorithme de mise en correspondance fournit le candidat éventuel le plus probable, au niveau du symbole grâce à la comparaison des clés du flux et des éléments de la base de données. On regroupe enfin les détections voisines semblables, et le candidat le plus fréquent est choisi sur des plages temporelles de l'ordre de 1min40 recouvrantes sur 50s.

3.2 Marquage des signaux audio

Le marquage des signaux audio intervient à l'étape de la construction de la base de données et au moment d'analyser les flux inconnus. Avant de marquer les signaux, on apprendre un modèle, grâce à la base de données de signatures :

- extraction des descripteurs
- calcul de similarité, détection de ruptures et formation de segments homogènes,
- clustering par la méthode LBG
- apprentissage du HMM, initialisé à partir des vecteurs issus du clustering LBG

Puis le marquage audio intervient pour les tous les signaux disponibles (signatures et flux) :

- extraction des descripteurs
- décodage des signaux par l'algorithme de Viterbi, sur la base du HMM appris

Cette partie contient alors 3 sections, la première concerne la présentation des descripteurs, la deuxième la présentation des HMM (initialisation, apprentissage et décodage), et enfin la dernière présente la construction de la table de hachage.

3.2.1 Caractéristiques : vecteurs de chromas

Afin d'extraire des segments homogènes, on va décrire le signal grâce aux descripteurs appelés chromas. Ils présentent l'avantage de décrire un signal de musique de manière assez compacte, puisqu'ils se basent sur les hauteurs de notes modulo l'octave. D'autre part, on les calcule sur des trames de l'ordre de la centaine de milliseconde, alors que, pour citer les descripteurs MFCC *Mel Frequency Cepstral Coefficients*, ceux-ci demandent une analyse sur des trames de l'ordre de la dizaine de milliseconde.

Les chromas représentent donc un signal selon la répartition de l'énergie entre les 12 demi-tons de la gamme tempérée. D'après Peeters [Pee06], les chromas sont obtenus pour une trame temporelle de la manière suivante :

1. Le spectrogramme est passé à travers un banc de filtres centrés sur les hauteurs de demi-tons.
2. Après un mapping entre les hauteurs de demi-tons et les classes de demi-tons correspondantes, la valeur du vecteur de chromas est obtenue par addition des valeurs des classes de hauteur équivalentes.

On normalise les chromas pour avoir des valeurs entre 0 et 1 pour chaque signature. Les vecteurs de chromas présentent l'intérêt de représenter l'information harmonique d'un signal, sans considérer sur quelle octave on est placé. Cette représentation est compacte puisqu'on ne considère que les 12 demi-tons de la gamme harmonique. Cependant, elle n'est pas robuste à un décalage en fréquence différent d'une octave.

On choisit comme longueur de fenêtre 300ms, durée qui correspond à la moitié d'une battue à un tempo de 100bpm. On considère que sur une telle durée, l'harmonie est stationnaire. Le recouvrement est d'1/3.

3.2.2 Modèles de Markov cachés

Présentation

Les chaînes de Markov discrètes permettent de modéliser temporellement un processus. Sachant l'ensemble d'états que le processus peut traverser et les probabilités de transition entre les états, on peut en déduire la probabilité d'une séquence d'états. On peut également calculer la séquence d'états la plus probable. Le modèle de Markov caché (HMM *Hidden Markov Model*) permet de modéliser un signal sans avoir accès directement aux états, mais à ce qu'on appelle des observations. Les états sont dit "cachés" car on ne les observe pas, mais on connaît la densité de probabilité des observations associée à chaque état. Pour une observation, on peut donc calculer la probabilité d'être dans chacun des états.

Rabiner, dans [Rab89], présente les modèles de Markov cachés, et en particulier la résolution des trois problèmes principaux de HMM pour une séquence d'observations $\mathbf{O} = (\mathbf{o}_1\mathbf{o}_2...\mathbf{o}_T)$:

- Connaissant les probabilités initiales $\boldsymbol{\pi} = (\pi_i)$ des états et la matrice de transitions $\mathbf{A} = (a_{i,j})$, on peut calculer la probabilité d'une séquence d'états $\mathbf{q} = (q_1, q_2, ..., q_t)$.

- Connaissant les probabilités initiales $\boldsymbol{\pi}$ des états, la matrice de transitions $\mathbf{A}$ et les densités de probabilités des observations $\mathbf{B} = (b_i(\mathbf{o_t}))$, l'algorithme de Viterbi donne la séquence d'états $\mathbf{q}$ la plus probable.

- Connaissant les probabilités initiales $\boldsymbol{\pi}$ des états, la matrice de transitions $\mathbf{A}$ et les densités de probabilités des observations $\mathbf{B}$, on va chercher à maximiser la probabilité d'avoir la séquence d'états la plus probable $\mathbf{q}$. C'est l'algorithme EM (*Expectation Maximization*) qui permet de mettre à jour les paramètres du modèle. On appelle λ l'ensemble des paramètres du modèles : $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$

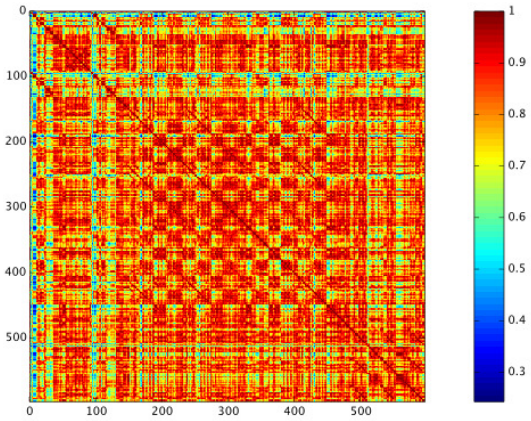
Dans notre travail, nous sommes confrontés aux deux dernières configurations.

Apprentissage du Modèle de Markov caché

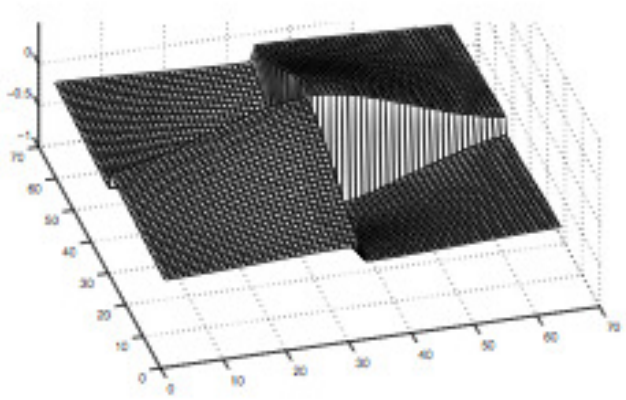
Cherchant à modéliser le signal par un modèle de Markov caché, on n'a aucune connaissance a priori. On utilise donc une approche non supervisée qui nous permet d'apprendre les paramètres du modèle ; leurs valeurs optimales sont déduites de la maximisation d'un critère. Ici le critère est la vraisemblance des données. Cependant, on sait que cette approche, basée sur l'algorithme EM (*Expectation Maximization*), est très sensible à l'initialisation des paramètres : une bonne initialisation permet d'atteindre plus sûrement un maximum global du critère à maximiser, plutôt qu'un maximum seulement local dans le cas d'une initialisation non appropriée.

- **Initialisation** : construction de segments homogènes par détection de rupture.
  La première étape consiste à découper les signatures en segments homogènes. Pour cela, on utilise un module de détection de ruptures de similarité. Le calcul de la matrice de similarités des vecteurs descripteurs entre eux s'obtient en calculant la distance cosinusoïdale entre les vecteurs. Un masque gaussien en damier appliqué à cette matrice permet de détecter les ruptures (voir figure 3.1). Cette méthode est tiré de [SBV10].
  La seconde étape est réalisée par clustering, grâce à l'algorithme des k-moyennes LBG (de ses auteurs Linde, Buzo et Gray [LBG03]). À partir de ce clustering, on calcule la distribution gaussienne associée à chaque cluster. Ces clusters sont associés à chaque état, et les distributions gaussiennes obtenues initialisent les distributions de probabilité des observations pour le HMM. On choisit de travailler sur des matrices de covariance diagonales.

- **Apprentissage des paramètres**
  L'apprentissage des paramètres est réalisé grâce à l'algorithme EM. Cet algorithme permet d'estimer de manière itérative les paramètres $\boldsymbol{\Theta} = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$ d'un modèle,



(a) Matrice de similarité pour une signature.



(b) Masque gaussien en damier, extrait de [Foo00]

FIGURE 3.1 – On applique un masque gaussien en damier à la matrice de similarité pour détecter les ruptures, et construire des segments homogènes.

en maximisant à chaque itération la fonction de vraisemblance des données. D'après [DLR77], on peut dériver cet algorithme dans le cas d'un HMM, soit à l'itération n :

Expectation

$$Q_{\mathbf{\Theta}_n}(\mathbf{\Theta}) = E[\log p(\mathbf{O}, \mathbf{q}; \mathbf{\Theta})|\mathbf{O}; \mathbf{\Theta}_n] \tag{3.1}$$
$$= E[\log p(\mathbf{O}|\mathbf{q}; \mathbf{\Theta})|\mathbf{O}; \mathbf{\Theta}_n] + E[\log p(\mathbf{q}; \mathbf{\Theta})|\mathbf{O}; \mathbf{\Theta}_n] \tag{3.2}$$

Maximization

$$\mathbf{\Theta}_{n+1} = arg \max_{\mathbf{\Theta}} Q_{\mathbf{\Theta}_n}(\mathbf{\Theta}) \tag{3.3}$$

Le calcul de la vraisemblance est très complexe, et on utilise en pratique un algorithme basé sur la programmation dynamique pour calculer cette vraisemblance à chaque étape de la séquence. On définit pour cela les grandeurs *forward* et *backward*, α et β pour M états : α est la probabilité d'observer $(\mathbf{o}_1\mathbf{o}_2...\mathbf{o}_t)$ et d'être dans l'état i à l'instant t, sachant le modèle λ, β est la probabilité d'observer $(\mathbf{o}_{t+1}\mathbf{o}_{t+2}...\mathbf{o}_T)$ sachant qu'on est dans l'état i à l'instant t et qu'on connaît le modèle λ

$$\begin{cases} \alpha_t(i) = P(\mathbf{o}_1\mathbf{o}_2...\mathbf{o}_t, q_t = i \mid \mathbf{\Theta}) \\ \alpha_1(i) = \pi_i b_i(\mathbf{o_t}) \\ \alpha_{t+1}(j) = [\sum_{i=1}^{M} \alpha_t(i) a_{i,j}]\ b_j(\mathbf{o}_{t+1}) \end{cases} \tag{3.4}$$

$$\begin{cases} \beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2}...\mathbf{o}_T \mid q_t(i), \mathbf{\Theta}) \\ \beta_T(i) = 1 \\ \beta_t(i) = \sum_{j=1}^{M} a_{i,j} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j) \end{cases} \tag{3.5}$$

Ces deux grandeurs permettent de calculer la probabilité $\gamma_t(i)$ d'être dans l'état i à l'état t sachant l'observation complète $\mathbf{O}$, et également la probabilité $\xi_t(i,j)$ d'être dans l'état i à l'instant t et dans l'état j à l'instant t+1 sachant l'observation complète $\mathbf{O}$:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{M} \alpha_t(j)\beta_t(j)} \tag{3.6}$$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{i,j} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{M} \sum_{i=1}^{M} \alpha_t(i) a_{i,j} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)} \tag{3.7}$$

À l'étape n, on peut alors exprimer $Q^{(n)}(\mathbf{\Theta}) = Q_{\mathbf{\Theta}_n}(\mathbf{\Theta})$ en fonction de γ et ξ :

$$Q^{(n)}(\mathbf{\Theta}) = \sum_{t=1}^{T} \sum_{i=1}^{M} \gamma_t^{(n)}(i) \log b_i(\mathbf{o}_t) + \sum_{t=1}^{T-1} \sum_{i=1}^{M} \sum_{j=1}^{M} \xi_t^{(n)}(i,j) \log a_{i,j} + \sum_{i=1}^{M} \gamma_1^{(n)}(i) \log \pi_i \tag{3.8}$$

Pour l'étape de maximisation, on dérive partiellement par rapport à chacun des paramètres pour obtenir l'estimation des paramètres à l'étape n+1, connaissant les paramètres estimés à l'étape n. Dans notre cas où $b_i(\mathbf{o}_t)$ est une distribution normale, on cherche à estimer également ses paramètres $\boldsymbol{\mu}_i$ et $\mathbf{\Sigma}_i$ pour chaque état i. On obtient finalement les formules de mise à jour :

$$\pi_i^{(n+1)} = \frac{1}{T} \sum_{t=1}^{T} \gamma_t^{(n)}(i) \tag{3.9}$$

$$a_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t^{(n)}(i,j)}{\sum_{t=1}^{T-1} \gamma_t^{(n)}(i)} \tag{3.10}$$

$$\boldsymbol{\mu}_i^{(n+1)} = \frac{\sum_{t=1}^{T} \gamma_t^{(n)}(i) \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t^{(n)}(i)} \tag{3.11}$$

$$\mathbf{\Sigma}_i^{(n+1)} = \frac{\sum_{t=1}^{T} \gamma_t^{(n)}(i) (\mathbf{o}_t - \boldsymbol{\mu}_i^{(n+1)})(\mathbf{o}_t - \boldsymbol{\mu}_i^{(n+1)})^T}{\sum_{t=1}^{T} \gamma_t^{(n)}(i)} \tag{3.12}$$

$$\tag{3.13}$$

où ^{T} désigne la transposition de matrice. On notera que π_i est estimé sous la contrainte que la somme des π_i sur tous les états est égale à 1.

En pratique, l'entrainement est fait sur toutes les signatures mises bout à bout. Ceci introduit artificiellement une distribution de probabilité initiale (due à la première signature de l'ensemble) et des transitions entre symboles (entre les différentes signatures), et on corrige ces artefacts en modifiant l'ordre des signatures aléatoirement à chaque itération de l'algorithme. La distribution de probabilités initiale est ré-initialisée à chaque itération à une distribution équiprobable.

Décodage par l'algorithme de Viterbi

Sachant les paramètres du modèle de Markov caché, on apprend pour chaque signal l'enchainement de symboles le plus probable. On regroupe ensuite les symboles identiques contigus.

L'enchaînement de symboles le plus probable est obtenu grâce à l'algorithme de Viterbi. L'algorithme de Viterbi est basé sur la programmation dynamique : on cherche à maximiser la probabilité de la séquence d'états **q** sachant la séquence d'observations **O**

et le modèle $\mathbf{\Theta}$. À chaque étape du chemin, on calcule le meilleur chemin partiel $\delta_t(i)$ jusqu'au temps t. Ce chemin est celui dont la probabilité est maximale, l'état courant étant l'état i. On a donc :

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P[q_1 q_2 ... q_{t-1}, q_t = i, \mathbf{o}_1 \mathbf{o}_2 ... \mathbf{o}_t | \mathbf{\Theta}] \tag{3.14}$$

La procédure entière pour trouver la meilleure séquence d'états est la suivante :

1. Initialisation

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \ 1 \leq i \leq N \tag{3.15}$$
$$\phi_1(i) = 0 \tag{3.16}$$

2. Récursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{i,j}] b_j(\mathbf{o}_t), \ 2 \leq t \leq T-1; \ 1 \leq j \leq N \tag{3.17}$$
$$\phi_t(j) = arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{i,j}], \ 2 \leq t \leq T-1; \ 1 \leq j \leq N \tag{3.18}$$

3. Arrêt

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{3.19}$$
$$q_T^* = arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{3.20}$$

4. Rétro-propagation

$$q_t^* = \phi_{t+1}(q_{t+1}^*), \ t = T-1, T-2, ..., 1 \tag{3.21}$$

$\mathbf{q}^*$ contient la séquence d'états optimale recherchée.

3.2.3 Construction de la table de hachage

Une fois que l'on a passé toutes les signatures à travers l'algorithme de Viterbi, Cano propose de les introduire tels quels dans une table de hachage. Mais dans ce cas, on ne conserve pas l'information temporelle entre les symboles. Nous proposons donc ici de les rassembler par 3 pour former une clé, comme le font les auteurs de [HKO01]. La générations de 2 clés consécutives se faisant par décalage d'un symbole (il y a donc 2 symboles en commun entre 2 clés voisines), on évite les problèmes de désynchronisation qui se posent pour le système de Philips. On conserve donc de manière pertinente l'information qui concerne l'enchaînement entre les symboles dans les signatures. Ceci permet d'abaisser la complexité de l'algorithme de mise en correspondance des clés dans l'étape d'alignement qui suit.

La table de hachage contient alors deux structures :

- un Index qui range les données par valeur de clé. Pour chaque clé, on a la liste des signatures où apparaît la clé.
- une Table qui range les données par signature. Pour chaque signature, on a la liste des clés qui apparaissent dans la signature.

On notera qu'on ne retient pas l'information temporelle [1], on ne s'attache qu'à l'enchaînement des symboles. D'une part, les signatures sont des extraits des titres réels et on ne connait pas leurs positions dans le signal original. L'identification s'appuie donc sur le fait de produire des clés génériques qui représentent le signal original entier, même si elles sont tirées des signatures uniquement. D'autre part, en ne s'intéressant qu'à l'enchaînement des symboles sans considérer leurs durées apporte plus de souplesse vis à vis d'un étirement ou d'une compression temporelle : on fait face de manière plus robuste au problème de désynchronisation locale.

3.3 Mise en correspondance

La mise en correspondance s'effectue sur les suites de symboles issues du décodage par l'algorithme de Viterbi, mises sous la forme de clés, pour mettre en correspondance des éléments du flux avec des signatures. Une mise en correspondance locale se fait implicitement grâce à la structure des clés. On présente ici la méthode d'alignement que nous avons implémentée pour la suite.

3.3.1 Identification locale

Afin de choisir un candidat éventuel au niveau de chaque clé du flux, on utilise une méthode d'alignement dynamique. Pour cela, après avoir établi la liste des titres de la base de données qui contiennent cette clé, on aligne localement chacun des candidats avec le flux sur une largeur d'au plus 32 symboles, en calculant la distance de Hamming entre les clés. On moyenne ensuite la distance de Hamming sur toute la portion de flux considérée. L'alignement est fait à partir de la clé courant en arrière (*backward*) et en avant (*forward*), comme on peut le voir sur la figure 3.2 pour une largeur de 4 symboles. Si les clés sont codées sur n bits, une distance de Hamming moyenne de n/2 est la valeur limite pour laquelle on retient le candidat.

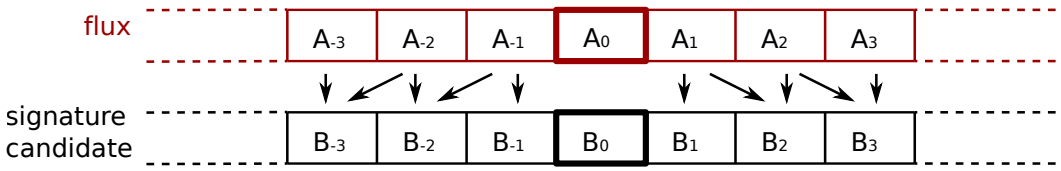


FIGURE 3.2 – Schéma de l'alignement local entre une portion de flux et une portion d'une signature candidate. La signature est candidate si les symboles A_0 et B_0 sont identiques. Les flèches indiquent la distance de Hamming calculée entre les symboles. À chaque fois que l'on calcule deux distances de Hamming pour une clé du flux, on prend la valeur minimale.

3.3.2 Identification globale

L'identification globale sert à lisser les résultats de l'identification locale. Elle se fait sur une plage temporelle de 1min40, avec un décalage de moitié entre chaque plage. Sur cette

1. On a néanmoins observé que les clés durent entre 500ms et 10,5s, et en moyenne 674ms.

plage temporelle, étant donné la liste des candidats résultant de l'identification locale, on choisit le candidat qui apparaît le plus souvent, le nombre d'apparitions étant au moins de n% de la signature. Le seuil est réglé par apprentissage.

Chapitre 4

Performances

Le travail a été intégralement développé sous Octave [1], logiciel de langage interprété haut-niveau (équivalent libre de Matlab) qui permet de traiter des problèmes mathématiques sous forme numérique.

4.1 Base de données

La base de données se divise en 2 parties : la partie "signatures" d'une part, et la partie "flux" d'autre part,. La partie "signatures" permet de construire une table de hachage, et la partie "flux" permet d'entraîner et d'évaluer les systèmes d'identification.

Les signatures constituent tout ce qu'on connaît des titres, au nombre de 30 000 : il s'agit d'extraits de 1 minute environ, échantillonnés à 11025 Hz et codés sur 16 bits signés. On ignore la position de la signature dans le signal original.

Les flux sont des diffusions réelles de 5 radios musicales françaises sur une journée, annotées en termes de titres musicaux. Ils sont découpés en portions de 5 minutes, et sont dans le format .wav en mono. La fréquence d'échantillonnage est de 16 kHz et le débit de 256 kbps.

4.2 Tests préliminaires de robustesse à l'étirement temporel

Ces tests unitaires de robustesse à l'étirement temporel sur un petit ensemble de signatures (11) mettent en évidence les propriétés recherchées du système proposé. Tout d'abord, voici les différents choix expérimentaux que nous avons fait :

- Nous avons choisi la taille de la fenêtre d'analyse de 300ms pour le calcul de chromas, durée basée sur 2 battues, à un tempo de 100 bpm.
- Le recouvrement des fenêtres d'analyse pour les chromas a été fixé à 2/3 de la taille de la fenêtre d'analyse, soit 100 ms.
- Les tests sont faits sur 10 des signatures de la base de données, sur leurs équivalents étirés et compressés temporellement à 1 et 10%, et enfin sur une diffusion réelle de broadcast de ces même signatures.

1. http ://www.gnu.org/software/octave/

4.2.1 Apport de l'initialisation du HMM par segmentation

Afin d'évaluer l'apport de l'initialisation du HMM élaborée pour le système proposé, nous avons évalué comparativement les pourcentages de correspondances entre les signatures originales et leurs équivalents dégradés. Le dictionnaire de symbole a été appris sur 8 éléments, grâce à 11 signatures. Les résultats sont consignés dans le tableau 4.1.

	Taux de compression				**Flux réel**
Segmentation	-10%	-1%	+1%	+10%	
non	22 ± 12	23 ± 11	22 ± 11	24 ± 12	41 ± 15
oui	82 ± 5	88 ± 5	77 ± 13	75 ± 13	2 ± 0.5

TABLE 4.1 – Résultat de correspondances en % (± l'écart-type) entre les signatures originales et leurs équivalents compressés. La dernière colonne le pourcentage de correspondance avec des diffusions réelles du titre. L'apprentissage se fait sur la base de 8 symboles sur 11 signatures, dans le cas d'une initialisation du HMM sans segmentation, et avec segmentation.

La taille de la base de tests utilisée étant très réduite, ces résultats ne nous permettent pas de conclure sur l'importance de la segmentation. Néanmoins, ils nous permettent de considérer la segmentation peut être une piste valable pour la suite. En effet, on peut remarquer qu'à taille de dictionnaire fixée [2], les pourcentages de correspondances seraient plus élevés dans le cas d'une segmentation préalable, et pour un étirement temporel seulement. Dans le cas de données de broadcast, les résultats autour de 3% montrent qu'il serait plus pertinent de ne pas effectuer de segmentation, dans ce cas la correspondance atteint environ 50%.

Dans la suite, on n'applique pas de segmentation pour pouvoir apprendre un dictionnaire avec davantage de symboles, et on va voir que ceci produit de meilleures performances.

4.2.2 Évaluation comparée

Dans le but de comparer le système proposé au système de référence, on a tout d'abord appris un dictionnaire de 16 symboles pour chaque signature. Dans un second temps, on a appris un dictionnaire de 32 symboles pour les 11 signatures ensemble.

Là encore, la petite taille de la base de données de tests ne nous permet pas de conclure de manière générale, mais nous encourage à poursuivre dans la voie empruntée. Les tableaux 4.2 et 4.3 montrent que les performances du système proposé en terme de robustesse au *time shifting* seraient assez élevées. Comparativement, le système de référence montre des résultats très faibles. Les résultats sur les données réellement diffusées montrent aussi que la robustesse aux distorsions appliquées réellement ne serait pas aussi grande que pour un traitement connu de *time shirting* directement sur la signature. Ces résultats sont pourtant encourageants pour le système proposé, car assez élevés en dépit de

2. Le nombre de symboles influe directement sur la taille des clés, et donc sur la taille de la table de hachage qui peut devenir critique dans le cas de très large bases de données. La taille du dictionnaire est également limitée par la taille des données disponibles pour l'apprentissage, ce qui est le cas ici.

	Taux de compression				Flux réel
Système	-10%	-1%	+1%	+10%	
Référence	1 ± 0.3	9 ± 2	10 ± 2	1 ± 0.3	0 ± 0
Proposé	88 ± 3	81 ± 4	77 ± 13	73 ± 13	45 ± 6

TABLE 4.2 – Résultat de correspondances en % (± l'écart-type) entre les signatures originales et leurs équivalents compressés. La dernière colonne le pourcentage de correspondance avec des diffusions réelles du titre. L'apprentissage se fait sur la base de 16 symboles sur 1 signature à la fois. Les résultats sont présentés pour le système de référence et le système proposé.

	Taux de compression				Flux réel
Système	-10%	-1%	+1%	+10%	
Référence	1 ± 0.3	10 ± 2	10 ± 1	1 ± 0.3	0 ± 0
Proposé	78 ± 7	83 ± 4	74 ± 13	70 ± 13	20 ± 10

TABLE 4.3 – Résultat de correspondances en % (± l'écart-type) entre les signatures originales et leurs équivalents compressés. La dernière colonne le pourcentage de correspondance avec des diffusions réelles du titre L'apprentissage se fait sur la base de 32 symboles sur les 11 signatures. Les résultats sont présentés pour le système de référence et le système proposé.

traitements courants et déjà importants que sont l'étirement et la compression temporels à 10%.

On peut noter également qu'on a un gain en termes d'occupation en mémoire : alors que les *fingerprints* pour le système de référence sont au nombre de 1000 éléments en moyenne pour une signature de 1 min, les *fingerprints* générés par le système proposé sont au nombre de 250 en moyenne pour une signature de 1 min. On gagne donc un facteur 4 pour représenter un signal, ce qui est très intéressant si on peut compter sur la robustesse de la représentation.

4.3 Performances en identification

4.3.1 Framework d'évaluation des performances

Afin d'évaluer les performances des systèmes d'identification, nous avons utilisé le système d'évaluation PyAFE[3], élaboré spécifiquement pour l'évaluation des systèmes d'identification audio automatiques pour le projet de recherche européen Quaero.

Le système PyAFE propose dans [RFB+11] des mesures de performances adaptées au cas de l'identification audio dans des flux de broadcast :

– elles ne reposent pas sur la détection de segments, mais sur la détection d'événements : l'instant de la prise de décision. Ceci permet d'être plus tolérant à une

3. http ://perso.limsi.fr/bredin/code/pyafe/

désynchronisation, car ce qui est important, c'est de détecter la présence d'un titre, sans détecter ses bornes. La nature même des signatures oblige à faire ce choix.
- on autorise un cas où aucun titre n'est détecté : le signal de broadcast ne correspond à aucun élément de la base de données.
- elles se limitent aux mesures des fausses alarmes et des faux rejets, sans privilégier l'une ou l'autre erreur.

Le nombre d'occurrences étant le nombre N d'items audio réellement présents, et le nombre de détection étant le nombre D d'items audio détectés, on définit l'*Accuracy* (ACC) comme le nombre de détections correctes (S_{OK}) sur le nombre d'occurrences :

$$S_{OK} = Card\{n \in [1..N], \exists\ d,\ j_d = i_n\ and\ t_n^{sta} \leq \tau_d \leq t_n^{end}\} \tag{4.1}$$

$$ACC = \frac{S_{OK}}{N} \tag{4.2}$$

Le nombre de faux rejets est :

$$S_{FR} = N - S_{OK} \tag{4.3}$$

S_{FA} et S_{FA}^{out} étant les les nombres de fausses alarmes à l'intérieur et à l'extérieur des occurrences, le score global par rapport au nombre d'occurrences est mesuré comme la différence entre le nombre de détections correctes diminué des fausses alarmes, le tout sur le nombre d'occurrences :

$$R = \frac{1}{N}(S_{OK} - [S_{FA} + S_{FA}^{out}]) \tag{4.4}$$

En réalité, on peut définir le nombre de fausses alarmes de trois manières. La première compte chaque fausse alarme comme une erreur, comme le montre la figure 4.1(a). La mesure $S_{FA,1}$ compte le nombre de détection qui sont alignée avec une occurrence, et $S_{FA,1}^{out}$ mesure le nombre de détections qui ne correspondent à aucune occurrence.

$$\begin{cases} S_{FA,1} = Card\{d \in [1..D], \exists\ n, t_n^{sta} \leq \tau_d \leq t_n^{end}\ and\ j_d \neq i_n\} \\ S_{FA,1}^{out} = Card\{d \in [1..D], \nexists\ n, t_n^{sta} \leq \tau_d \leq t_n^{end}\} \end{cases} \tag{4.5}$$

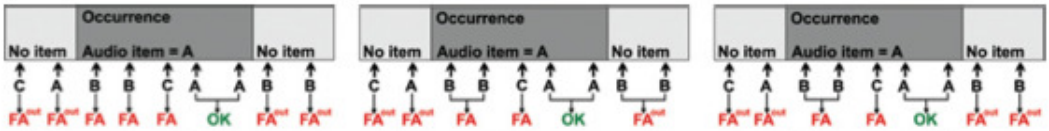
La seconde rassemble en une seule erreur les fausses alarmes d'un même item dans une zone d'occurrence ou dans une zone sans occurrence, comme le montre la figure 4.1(b). Pour ne pas biaiser la mesure en faveur des détections correctes, on considère les zones "sans item" comme des occurrences là où on ne compte que des fausses alarmes. Les fausses alarmes sont donc décomptées dans une mesure $S_{FA,2}$ comme le nombre d'occurrences pour lesquelles on a une détection incorrecte à l'intérieur d'un item, et $S_{FA,2}^{out}$ est le nombre d'occurrences pour lesquelles on n'a aucune détection :

$$\begin{cases} S_{FA,2} = Card\{n \in [1..N], \exists\ d, t_n^{sta} \leq \tau_d \leq t_n^{end}\ and\ j_d \neq i_n\} \\ S_{FA,2}^{out} = Card\{n \in [1..N], \nexists\ d, t_n^{sta} \leq \tau_d \leq t_n^{end}\} \end{cases} \tag{4.6}$$

Enfin, la troisième manière est un compromis entre les deux précédentes. Elle regroupe les fausses alarmes d'un même item, seulement lorsqu'elles apparaissent dans une occurrence (figure 4.1(c)).

Enfin, on en déduit trois mesures de performances globales [4] :

4. Aucune de ces mesures n'est privilégiée a priori.



(a) **Score 1:** each FA = 1 error in or outside the occurrences.

(b) **Score 2:** FAs of same ID in a same area = 1 error.

(c) **Score 1.5:** FAs not grouped in "no-item" areas.

FIGURE 4.1 – Différentes manières de compter les fausses alarmes, extrait de [RFB+11].

$$R_1 = \frac{1}{N}(S - OK - [S_{FA,1} + S^{out}_{FA,1}]) \tag{4.7}$$

$$R_2 = \frac{1}{N}(S - OK - [S_{FA,2} + S^{out}_{FA,2}]) \tag{4.8}$$

$$R_{1.5} = \frac{1}{N}(S - OK - [S_{FA,2} + S^{out}_{FA,1}]) \tag{4.9}$$

4.3.2 Résultats de performances en identification

Afin de régler le seuil de décision dans l'étape d'identification, il est nécessaire de tester plusieurs valeurs, pour lesquelles les résultats sont consignés dans le tableau 4.4 [5]. On utilise pour cela les données de 3 radios sur 4 fois 24h.

Par manque de temps, nous n'avons malheureusement pu réaliser l'évaluation des performances que pour le système de référence sur la base de données de diffusions réelles disponible.

Les résultats obtenus pour système de référence montrent bien la décroissance monotone du nombre de fausses alarmes et du nombre de détections correctes, quand le seuil augmente : le seuil influe directement sur les candidats tolérés, un seuil plus élevé signifiant une contrainte plus forte sur les candidats acceptés. C'est la balance entre ces deux mesures qui nous donne donc la valeur optimale du seuil.

La valeur optimale retenue pour le seuil est 1%. Pour cette valeur, l'accuracy atteint 50%, alors que les fausses alarmes n'atteignent que 24%, ce qui nous donne une valeur de 34% pour R. On a donc 1 titre sur 2 reconnu, et en considérant les fausses alarmes, 1 titre sur 3. Ces valeurs sont en-dessous des résultats annoncés dans la littérature, qui sont plutôt de l'ordre de 80% pour la valeur de R [RFB+11]. Ceci peut être expliqué d'une part par la nature de la base de données utilisée dans ce travail car on regroupe les résultats de plusieurs radios, alors qu'il se pourrait que des réglages spécifiques aux stations génèrent de meilleurs résultats : les résultats de [RFB+11] sont d'ailleurs donnés pour une seule radio, la 548. Les écarts-types assez élevés dans le tableau des résultats nous montrent également la variation des résultats en fonction de la station. D'autre part, la différence des performances peut aussi venir du fait que l'on ne connaît pas l'algorithme de mise en correspondance pour les autres études. Cependant, ces remarques ne sont pas gênantes dans notre cas, dans la mesure où l'on cherche à comparer deux méthodes sur la même base de données.

5. Les valeurs de FA,1 et FA,2 n'étant pas différenciées en pratique, on ne fera pas la distinction.

Seuil(%)	**Radio**	$ACC(\%)$	$FA(\%)$	$FA^{out}(\%)$	$R(\%)$
0.5	1	48	22	10	25
	3	37	32	14	6
	548	67	13	26	25
	Total	**51** $\pm$ 15	**22** $\pm$ 10	**17** $\pm$ 8	**19** $\pm$ 11
1	1	45	15	9	30
	3	39	17	13	20
	548	67	6	14	51
	Total	**50** $\pm$ 15	**13** $\pm$ 6	**11** $\pm$ 2	**34** $\pm$ 16
2	1	43	12	9	32
	3	27	13	11	17
	548	65	5	13	51
	Total	**45** $\pm$ 19	**10** $\pm$ 4	**11** $\pm$ 2	**33** $\pm$ 17
3	1	42	9	11	29
	3	24	10	11	18
	548	59	5	12	47
	Total	**42** $\pm$ 17	**8** $\pm$ 3	**11** $\pm$ 1	**31** $\pm$ 14

TABLE 4.4 – Résultat des performances pour différentes valeurs du seuil de décision pour le système de référence, évaluées pour 3 radios sur 4 fois 24h, soit 288h de diffusion.

Conclusion

Dans ce rapport, nous avons présenté notre travail sur différentes techniques de marquage audio dans l'objectif d'identifier automatiquement des flux de broadcast. En particulier, le système proposé s'appuie sur une représentation symbolique du signal, avant de former des clés de 3 symboles, afin de conserver une information sur l'enchaînement des symboles directement dans le *fingerprint* du signal. La marque obtenue pour chaque signal peut ainsi gagner en robustesse et en concision.

Nous avons ensuite évalué ces techniques de manière comparative en terme de robustesse à la compression et à l'étirement temporel (*time shifting*), ainsi que sur des données réelles de flux de broadcast. Cette évaluation, limitée à un petit nombre de signatures, ne nous permet pas de conclure de manière générale, mais nous permet de poursuivre dans la voie proposée afin de produire des résultats plus généraux par la suite. Une évaluation plus exhaustive en termes de performances d'identification sur des données réelles diffusées sur des radio musicales n'a été menée que sur le système de référence, le temps imparti pour ce stage n'ayant pas permis de tester le système proposé de ce point de vue. Cependant, les tests sur la robustesse au *time shifting* sont très encourageant en comparaison avec le système de référence et poussent à continuer la piste empruntée dans ce travail.

En effet, cette étude a ouvert plusieurs perspectives en ce qui concerne

la taille de la base de données : le temps ayant été un facteur limitant dans notre travail, nous avons dû réduire la base de données de signatures pour une partie des tests. Agrandir la base de données permet de tester la robustesse des modèles à plus données d'entrée, et ainsi généraliser les résultats de performances.

la représentation symbolique : en regardant de plus près l'ensemble des clés attachées aux différentes signatures, on a remarqué que les musiques percussives et à large dynamique sont représentées par un grand nombre de clés, et au contraire, les musiques à variation lente et à faible dynamique sont représentées par nombre restreint de signatures. Cela vient directement du choix des descripteurs, les chromas pris sur des fenêtres de 300ms, qui représentent le plus efficacement les musiques à variation lentes. Afin de prendre en compte les variations rapides, on pourrait envisager d'extraire les chromas sur des fenêtres adaptées au tempo de la musique considérée. C'est ce que propose Shiu dans [SJK06] pour l'analyse de structure musicale par calcul de matrice de similarité.

le traitement de donnée compressées : si on considère la chaîne complète de l'identification audio, la décompression occupe une part non négligeable de l'ordre de 40% du temps de traitement. Éviter cette étape est donc une amélioration pertinente pour accélérer le processus complet. Des travaux ont déjà été effectués par Li [LLX10],

Liu [LC11] dans cette perspective, en extrayant les descripteurs directement des données au format mp3.

la robustesse au décalage en fréquence : on a étudié une modélisation statistique avec une certaine initialisation, basé sur les descripteurs de chromas. Il pourrait être intéressant d'implémenter le même type de système, basé cette fois-ci sur les descripteurs de MFCCs. Les MFCCs présentant l'avantage d'être plus robuste au décalage en fréquence que les chromas, on pourrait évaluer les performances dans le cas de ce type de descripteurs.

le temps de calcul et l'occupation en mémoire : le problème du temps de traitement et de l'occupation mémoire est inhérent à tout travail sur de larges bases de données. Dans notre étude, il a été à l'origine de certains choix d'implémentation et de conditions expérimentales, comme la taille de la base de données et du dictionnaire de symboles. Développer des stratégie de réduction de temps de calcul permettrait d'implémenter des modèles mathématiques plus riches et adaptés.

Bibliographie

[CBMN02] Pedro Cano, Eloi Batlle, Harald Mayer, and Helmut Neuschmied. Robust sound modeling for song detection in broadcast audio. In *Proceedings from AES 112th International Convention*, pages 1–7, 2002.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1) :1–38, 1977.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6) :381–395, 1981.

[Foo00] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, 1 :452–455, 2000.

[Gus97] Dan Gusfield. *Algorithms on strings, trees, and sequences : computer science and computational biology.* Cambridge University Press, New York, NY, USA, 1997.

[HKO01] Jaap Haitsma, Ton Kalker, and Job Oostveen. Robust audio hashing for content identification. In *Content-Based Multimedia Indexing*, 2001.

[KHS05] Yan Ke, Derek Hoiem, and Rahul Sukthankar. Computer vision for music identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 597–604, 2005.

[LBG03] Y. Linde, A. Buzo, and R. Gray. An Algorithm for Vector Quantizer Design. *Communications, IEEE Transactions on*, 28(1) :84–95, January 2003.

[LC11] Chih-Chin Liu and Po-Feng Chang. An efficient audio fingerprint design for mp3 music. In *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*, pages 190–193, 2011.

[LLX10] Wei Li, Yaduo Liu, and Xiangyang Xue. Robust audio identification for mp3 popular music. In *Proceedings of the 33rd international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 627–634, 2010.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60 :91–110, 2004.

[Pee06] Geoffroy Peeters. Chroma-based estimation of musical key from audio-signal analysis. In *Proceedings from ISMIR 06 : International Conference on Music Information Retrieval*, pages 115–120, 2006.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[RFB+11] Mathieu Ramona, Sébastien Fenet, Raphaël Blouet, Hervé Bredin, Thomas Fillon, and Geoffroy Peeters. A public audio identification evaluation framework for broadcast monitoring. *Applied Artificial Intelligence*, 2011.

[SBV10] Gabriel Sargent, Frédéric Bimbot, and Emmanuel Vincent. Un système de détection de rupture de timbre pour la description de la structure des morceaux de musique. In *Journées d'Informatique Musicale 2010*, May 2010.

[SJK06] Yu Shiu, Hong Jeong, and C.-C. Jay Kuo. Similarity matrix processing for music structure analysis. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 69–76, 2006.

[VJ03] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2003.

[Wan03] Avery L. Wang. An industrial-strength audio search algorithm. In *Proceedings of the 4th Symposium Conference on Music Information Retrieval*, pages 7–13, 2003.

[Zhu10] Zhu, Bilei and Li, Wei and Wang, Zhurong and Xue, Xiangyang. A novel audio fingerprinting method robust to time scale modification and pitch shifting. In *Proceedings of the International Conference on Multimedia*, pages 987–990, 2010.