Ircam AudioPrint : Calcul des Empreintes Sonores et Choix des Paramètres

Rémi Mignot

Ircam AudioPrint:

Calcul des Empreintes Sonores et Choix des Paramètres

Rémi Mignot

IRCAM -CNRS, rapport interne, projet BeeMusic

13 janvier 2016

Résumé

Dans le contexte de l'indexation audio, le travail présenté par ce document concerne le calcul de nouvelles empreintes sonores. L'objectif est d'une part la robustesse naturelle à certains types de dégradations sonores, et d'autre part de contenir de l'information musicale pertinente, contrairement à d'autres approches de la littérature. En effet, nous verrons que les données obtenues renseignent d'une certaine manière sur la variation temporelle du timbre. Même si l'application visée est ici l'identification audio, notons que ces nouvelles empreintes sonores peuvent être utilisées comme descripteurs audio pour d'autres tâches. Dans ce rapport, la méthode de calcul est présentée en détail, ainsi que les différentes propriétés de robustesse à certaines altérations ciblées. La liste des paramètres est aussi donnée et leurs valeurs seront déterminées par une procédure d'évaluation de la robustesse.

1 Introduction

1.1 Indexation audio, et dégradations sonores

Pour certaines applications audio, il peut être utile d'identifier un signal sonore donné, soit pour en connaître le titre ou l'artiste d'un morceau de musique, ou bien pour éviter les doublons lors de la constitution d'une base de données sonores. L'indexation audio a alors pour but d'identifier chaque signal par une séquence d'empreintes sonores, ou de codes, et la comparaison de leurs valeurs permet ensuite de détecter des possibles répétitions. Ainsi, après avoir construit une base d'items de référence, des morceaux de musique par exemple, il est possible de reconnaître un extrait donné si celui-ci est inclu dans la base de référence.

Ces empreintes, dont le terme anglophone usuel est *Audio FingerPrints*, sont chacune une représentation condensée d'une partie du son, et sont le plus souvent basées sur des propriétés spécifiques des signaux audio. Cependant, dans le cas où l'extrait à identifier est perturbé par une dégradation, la modification des empreintes peut alors provoquer une détérioration significative de la reconnaissance. Ces altérations sonores sont par exemple : ajout de bruit synthétique ou environnemental, distorsion ou saturation, encodage/décodage aux formats MP3 ou GSM, changement de hauteurs, dilatation/compression du temps, égalisation ou filtrage, réverbération, compression des dynamiques. Le travail présenté dans ce papier traite de la robustesse de la reconnaissance d'extraits sonores en présence d'altérations.

1.2 État de l'art

La plupart des techniques mises en œuvres sont divisées en deux étapes : d'une part, l'analyse des signaux conduit à l'obtention des séquences d'empreintes sonores qui décrivent le son de façon unique. D'autre part, la phase de recherche consiste à retrouver dans une base de référence le ou les items les plus similaires par comparaison des empreintes.

Parmi les travaux pionniers du domaine, Fraunhoffer, cf. [1], a proposé l'utilisation de descripteurs audio de la norme MPEG7. Ainsi, à l'aide d'une quantification vectorielle, par clustering, une approche standard de classification permet la sélection de l'item le plus proche, au sens d'une erreur de reconstruction des codes. Pour la méthode de Philips [10], la recherche est basée sur un code binaire de grande dimension calculé sur le spectrogramme filtré par un masque dérivateur. Après une pré-sélection d'items candidats, obtenus par une recherche de sous-codes dans une table de hachage, la distance de Hamming est utilisée pour obtenir un indice de similarité. L'application la plus populaire est probablement celle de Shazam, dont la technique repose sur la recherche par table de hachage des occurrences d'empreintes sonores qui y représentent des paires de maxima du spectrogramme, cf. e.g. [24]. Après une pré-sélection des items candidats ayant le plus grand nombre de codes reconnus, une sélection plus précise est faite en étudiant la cohérence temporelle des codes détectés par un histogramme.

Pour la plupart de techniques présentées ci-dessus, la question de la robustesse est partiellement prise en compte. Par exemple dans [10], une version *étendue* de la table de hachage est faite, de sorte à continuer la recherche tant qu'aucun item sélectionné ne donne une similarité satisfaisante. Ce principe est basé sur un calcul de fiabilité des bits des codes. Par ailleurs, la méthode de Shazam [24] tire partie d'une part de la résistance naturelle au bruit des maxima du spectre, et d'autre part sur un nombre très grand de codes de hachage.

Cependant, pour répondre plus spécifiquement à ce problème d'altérations sonores, d'autres travaux ont été mis en œuvre. Par exemple dans [11], une transformation du spectre a permis d'améliorer la robustesse de la méthode de [11], Philips, au changement d'échelle fréquentielle. Par ailleurs, plus récemment

dans [19], cette même méthode est encore été améliorée en introduisant le principe de *Hachage Approximatif* qui permet dans ce cas une généralisation de l'approche et une amélioration de la tolérance aux erreurs, pour tout type de dégradations. Enfin, nous pouvons citer [8] qui améliore la robustesse de [24], Shazam, au changement de hauteur via l'utilisation de la Transformée à Q-Constant (CQT), et [23] qui utilise des *quadruplets* de maxima dont les positions en temps et en fréquence sont codés de façon relative, améliorant de la sorte la robustesse au changement d'échelle temporelle et fréquentielle.

1.3 IrcamAudioID

De précédents travaux réalisés à l'IRCAM ont permis de mettre en œuvre un système complet d'indexation audio. Dans cette méthode, voir [20, 21], les empreintes sonores sont basées sur le calcul d'un banc de 6 filtres. En considérant des fenêtres d'analyse de 2 secondes démarrant chacune d'un point donné, une transformée de Fourier discrète fournit une représentation de l'évolution temporelle des sorties de chaque filtre. Grâce à un regroupement des coefficients obtenus, cette approche fournit pour chaque temps 6 vecteurs de dimension 6.

Malheureusement cette méthode est peu robuste aux dégradations du signal pouvant survenir. Pour un petit ensemble de morceaux à référencer, une méthode de recherche approximative, basée sur l'approche des "k-plus proches voisins", peut être utilisée pour compenser cette faiblesse. Cependant pour de très grandes bases de données, le coût de calcul de cette méthode est trop grand et empêche son utilisation. C'est la raison pour laquelle un hachage a été mis en place. Ce hachage améliore le temps de la recherche, mais ne permet plus de corriger suffisamment la sensibilité des empreintes aux dégradations.

1.4 Objectif et plan du document

Les empreintes sonores développées dans ce travail sont par leur conception intrinsèquement plus robustes à plus de types de dégradations, telle que la dilatation/compression du temps, et d'autre part, contiennent suffisamment d'information pertinente sur le plan musical pour pouvoir être utilisées comme descripteurs sonores, pour d'autres applications audio.

Remarquons que dans le cadre de l'indexation audio, deux autres étapes sont nécessaires après le calcul des empreintes : une réduction de dimension et la phase de recherche. Ce document ne traite que du calcul des empreintes sonores originales, qui sont de grandes dimensions.

Ce document est organisé de la manière suivante : en section 2, un résumé du calcul des empreintes est donné. Nous y donnons aussi les raisons qui les rendent robustes à certaines dégradations. Puis les détails de chaque étapes des calculs sont présentés en section 3. En section 4 nous faisons la liste des paramètres dont il faudra déterminer les valeurs, et nous donnons quelques idées supplémentaires testées. Finalement en section 5 nous présentons les différents tests qui ont permis par la suite de choisir les valeurs, et une conclusion est donnée en section 6.

2 Nouvelles empreintes sonores robustes

La nouvelle approche mise en œuvre ici permet de définir des empreintes sonores naturellement robuste à certains types de dégradations qui sont : changement d'échelles temporelle et fréquentielle (time stretching et pitch shifting selon les termes anglophones habituels), ajout de bruit, égalisation, et variation du gain. La définition des empreintes utilisées est données en sec. 2.1 via un résumé de chaque étape de la méthode de calcul, voir fig. 1 pour une illustration, puis les propriétés de robustesse sont présentées en sec. 2.2.

2.1 Résumé des empreintes

Voici un résumé de chaque étape du calcul :

Spectrogramme : Dans un premier temps, le spectrogramme complet du signal est calculé par une Transformée de Fourier à Court-Terme. Il s'agit soit du signal d'un morceau de la base de référence, soit du signal complet de l'extrait à reconnaître. Ici le spectre est limité à 5500 Hz environ pour ne conserver que les informations pertinentes.

Fenêtre d'observation : Puis, une fenêtre d'observation long-terme d'environ 2 secondes est déplacée par pas de 250 ms environ afin d'analyser des portions limitées en temps du spectrogramme complet du signal.

Conversion Log–Log : Chaque sous-matrice obtenue fournit alors une représentation temps-fréquences d'une portion du signal, en échelles linéaires (secondes-hertz). De là une conversion en échelles logarithmiques donne une nouvelle représentation temps-fréquence (Log temps – Log fréquences).

Division en sous-bandes de fréquences : Par la suite, l'axe des fréquences est séparé en 5 bandes. En pratique, nous définissons 5 bandes de largeurs égales et de centres uniformément répartis en échelle logarithmique, et avec un recouvrement. Pour plus de simplicité, nous nommerons : "carreaux" ces représentations et "pixels" chacun de leur coefficient.

Modification des amplitudes : C'est alors qu'un traitement des amplitudes est réalisé pour chaque "carreau". Premièrement, un redressement est fait par rapport à une valeur dépendant du maximum. Deuxièmement, une pondération 2D est faite pour ramener à 0 les pixels aux bords. Ensuite une normalisation L_∞ est réalisée, et une changement d'échelle quasi-logarithmique des amplitudes est fait.

Transformée de Fourier à 2 dimensions : Enfin, une transformée de Fourier discrète à 2 dimensions est réalisée sur chaque carreau modifié précédemment, et le module en est extrait. Aussi, la moitié de la matrice obtenue est supprimée de sorte à éviter la symétrie. Nous nommerons cette opération : DFT2D.

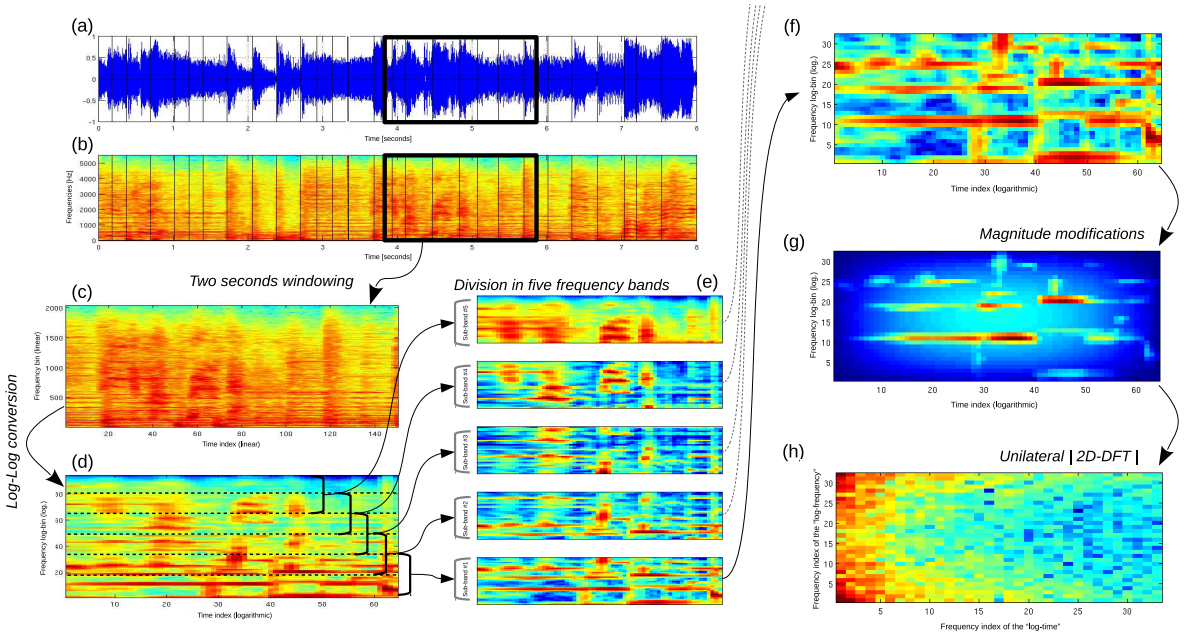

FIGURE 1 – Illustration du calcul des empreintes sonores : (a) Signal temporel. (b) Spectrogramme complet. (c) Sélection d'une sous-fenêtre de 2 secondes environ. (d) Conversion du temps et de la fréquence en échelle logarithmique. (e) Découpage en 5 bandes de fréquences, avec recouvrement. (f) Sélection des bandes une-à-une. (g) Modification des amplitudes : redressement, conversion, normalisation, fenêtrage. (h) Calcul de l'amplitude de la DFT2D.

2.2 Propriétés de robustesse

Cette nouvelle méthode de calcul des empreintes sonores a été initialement conçue pour améliorer la robustesse à certains types de dégradations. Nous en donnons ici les propriétés.

Robustesse au changement d'échelles : Cette altération consiste en la modification des échelles du temps et/ou des fréquences. On parle souvent de : time stretching ou pitch shifting en utilisant les termes anglophones usuels. Comme nous avons vu précédemment, le calcul des empreintes est basé sur

une représentation logarithmique du temps et des fréquences. Alors qu'en linéaire, un time stretching et un pitch shifting produisent une dilatation ou une compression, en échelle logarithmique ces mêmes altérations se manifestent par une transposition. Or, l'une des propriétés remarquables de la transformée de Fourier est que le module est invariant à toute transposition. Ainsi, l'utilisation du module de la transformée de Fourier 2D, permet la robustesse à cette altération. Remarquons que cette propriétés n'est pas valide au sens strict dans le cas d'une transformée de Fourier discrète, en raison des effets de bords. C'est l'une des raison pour laquelle une pondération 2D est réalisée.

Robustesse au bruit additif : La robustesse au bruit additif est prise en compte deux fois dans ce travail. Premièrement, en considérant un bruit stationnaire, sur un horizon de deux secondes, et relativement localisé en fréquences, la séparation en cinq bandes de fréquences différentes permet dans la plupart des cas d'isoler le bruit et de laisser un certain nombre de bandes intactes. Secondement, puisque, le bruit se manifeste le plus souvent par l'ajout de composantes plus faibles que le signal d'intérêt, le redressement des pixels à un niveau minimal, permet d'annihiler l'effet du bruit dans une certaine mesure.

Robustesse à l'égalisation : Cette modification du son consiste en un filtrage spécifique, et est donc représentable par la multiplication des spectres par une fonction de transfert donnée invariante en temps. Le comportement logarithmique du changement d'échelle, y=f(x), permet alors de changer le produit en addition invariante dans le temps. Alors, la transformée de Fourier 2D rejette l'effet de l'égalisation sur l'axe vertical, axe des fréquences des log-fréquences. Notons qu'un changement d'échelle purement logarithmique aurait pour conséquence d'amplifier l'effet résiduel du bruit et de provoquer des valeurs extrêmement faibles. Ainsi, le comportement linéaire pour les faibles amplitudes permet d'éviter ces phénomènes indésirables.

Robustesse au gain : Enfin, la normalisation de la valeur maximale pour chaque carreau permet l'invariance des empreintes sonores à tout changement de volume du signal initial.

Remarque sur le décalage temporel : Nous n'avons pas détaillé le choix des instants p_k d'analyse, mais dans [14] nous expliquons qu'ils sont choisis par une sorte de détections d'événements sonores, *onsets* en anglais. Le problème est qu'en présence d'altérations, la position de ces points peut changer. Même si l'algorithme de décision a été conçu de sorte à réduire au maximum les décalages, de petites erreurs peuvent persister. Pour compenser cela dans le calcul des empreintes, l'échelle logarithmique du temps débute après la trame p_k, considérée comme origine des temps pour la conversion. Nous constatons en effet que plus cet instant est tardif, moins le décalage de l'instant d'analyse influe dans la conversion.

3 Détails des calculs

Voici maintenant quelques détails techniques pour chaque étape du calcul. Il s'agit ici de la méthode finalement retenue pour définir les empreintes sonores utilisées pour l'identification audio du projet. En section 4 nous verrons quelques variantes qui auraient pu améliorer les performances, mais elles ont été rejetées suite aux tests faits en section 5.

3.1 Spectrogramme

Dans un tout premier temps, le spectrogramme complet du signal audio-numérique est calculé par une *Transformée de Fourier à Court-Terme*, dont l'acronyme anglais est STFT, cf. e.g. [25, chap7]. La valeur du spectrogramme du signal discret x_n à la trame d'indice p et à la fréquence d'indice m est résumée par l'équation suivante :

$$\mathrm{STFT}\{x_n\}(m,p) = \sum_{n=0}^{M_s-1} x[n - \lfloor \frac{W_s-1}{2} \rfloor + pH_s]\ w[n]\ \mathrm{e}^{-j2\pi nm/M_s}, \tag{1}$$

où $\lfloor . \rfloor$ représente la partie entière, W_s est la taille en échantillons de la fenêtre d'analyse, H_s le pas d'avancement de la fenêtre glissante, M_s la taille de la transformée de Fourier en bins, et w_n la fenêtre de pondération, avec $w_n = 0\ \forall n \geq W_s$ pour réaliser le remplissage de 0, *0-padding* en anglais. Nous choisissons w_n la fenêtre de Hann périodique entre 0 et $W_s - 1$. De plus, la valeur de M_s est deux fois la

puissance de 2 suivant W_s, supérieure ou égale. Ce choix permet l'utilisation de l'algorithme rapide de transformée de Fourier discrète, FFT en anglais cf. [5], et assure un bon échantillonnage du spectre.

Enfin, seul le module du spectrogramme est utilisé pour les fréquences inférieures à $F_n = F_s/2$. Alors si x_n est le signal d'un morceau de référence de la base ou de l'extrait à reconnaître, avec $M = M_s/2+1$ la taille du spectre unilatéral, la représentation temps-fréquences du signal complet est donnée par

$$X[m,p] = \Big|\ \mathrm{STFT}\{x_n\}(m,p)\ \Big|,\ \forall m \in [0, M-1] \text{ et } p \geq 0. \tag{2}$$

Notons que la fréquence d'échantillonnage choisie est $F_s = 11025$ Hz, ainsi les spectres unilatéraux sont bornés en fréquences à $F_n = 5512.5$ Hz. Même si les fréquences au delà des 5500Hz sont nécessaires à l'écoute pour la qualité audio, elles sont peu informatives sur le contenu musical. Ainsi l'information pertinente se concentre principalement dans le premier quart du spectre audible, et la limitation des fréquences permettra en plus de réduire le temps de calcul et l'utilisation de mémoire. Par ailleurs dans le cas de l'identification audio, les fréquences très élevées auront en générale une sensibilité plus forte au bruit en raison de la pente négative du spectre.

Souvent, une rotation circulaire du segment d'analyse est effectuée afin de placer l'origine des phases au centre de la fenêtre de pondération, en $n = \lfloor (W_s - 1)/2 \rfloor$. Cependant puisque nous n'utilisons ici que le module du spectrogramme, cette opération n'a aucun effet.

3.2 Fenêtre d'observation

Par la suite, une fenêtre d'observation à *long-terme* d'environ 2 secondes est déplacée le long de l'axe du temps sur le spectrogramme X, et chaque empreinte sonore est alors donnée pour un instant correspondant au point de départ de l'intervalle d'analyse. Avec Q la taille de ces intervalles d'observation, en nombre de trames, une empreinte est associée à l'instant p_k d'indice k, et est calculée à partir de $X_{m,p}$ pour $m \in [0, M-1]$ et $p \in [p_k, p_k + Q - 1]$. Notons alors X_k la matrice de taille $(M \times Q)$ pour le calcul de l'empreinte à p_k, et dont les éléments valent :

$$X_k[m,q] = X[m-1, q-1+p_k],\ \forall q \in [1, Q] \text{ et } \forall m \in [1, M]. \tag{3}$$

Après plusieurs transformations de X_k, nous obtiendrons une représentation du signal sonore sur l'intervalle d'analyse de taille Q trames, ou $T = QH_s/F_s = Q/F_r$ en secondes, avec $F_r = F_s/H_s$ le nombre de trames par seconde (*frame rate*).

Remarquons que dans la méthode développée ici pour l'indexation audio, ces temps d'analyse p_k ne sont pas uniformément placés dans le temps. Sans entrer dans les détails, ces instants sont déterminés par une sorte de détection d'événements sonores, *onsets* en anglais, avec une fréquence d'environ 4 par seconde. Pour une illustration, voir les barres noires verticales sur les figures 1a et 1b, ou voir [14, 17] pour plus de précisions.

3.3 Conversion logarithmique temps-fréquence

Les indices m et q de X_k représentent respectivement la fréquence $f_m = F_n(m-1)/(M-1)$ en herzts et le temps $t_q = (q-1)/F_r$ en secondes, avec pour origine la trame p_k. Il s'agit de relations linéaires, et pour des raisons expliquées en sec. 2.2, nous cherchons une représentation avec des échelles logarithmiques. Par conséquent, à partir de la matrice X_k qui fournit une représentation temps-fréquence en échelle linéaire (secondes-herzts) de l'intervalle d'observation, une interpolation produit une nouvelle représentation logarithmique (log temps – log fréquences). Les nouvelles variables sont alors :

— Avec f_{min} la fréquence minimale en [Hz], f_{max} la fréquence maximale, et M_f le nombre de points dans l'échelle logarithmique des fréquences, log-bins, alors pour $m \in [1, M_f]$, la fréquence $\widetilde{f}_m$ est :

$$\widetilde{f}_m = \exp\Big(f_{min} + (m-1)\delta_f\Big), \quad \text{avec} \quad \delta_f = \frac{\log(f_{max}/f_{min})}{M_f - 1}. \tag{4}$$

— Avec t_{min} le temps de départ en [s] où p_k/F_r est l'instant d'origine, t_{max} le dernier instant de l'échelle, et M_t le nombre de points, alors pour $q \in [1, M_t]$, le temps $\widetilde{t}_q$ est :

$$\widetilde{t}_q = \exp\Big(t_{min} + (q-1)\delta_t\Big), \quad \text{avec} \quad \delta_t = \frac{\log(t_{max}/t_{min})}{M_t - 1}. \tag{5}$$

Remarquons que comme nous l'avons expliqué en sec. 2.2, un t_{min} élevé permet en partie d'améliorer la robustesse à de possibles erreurs dans la décision du temps d'analyse p_k. Nous choisirons dans la pratique t_{min} de l'ordre de la demi-seconde.

Pour convertir les échelles linéaires t_q et f_m vers les échelles logarithmiques $\widetilde{t}_q$ et $\widetilde{f}_m$, nous utilisons deux interpolations successives du temps et des fréquences. Techniquement, la conversion mise en place est basée sur les règles de Simpson, souvent utilisée comme schéma d'intégration numérique, cf. [2].

Pour le temps par exemple, l'idée est d'interpoler $X_k[q,m]$ par rapport à $\log(t_q)$ via une approximation $P(\log(t))$ quadratique par morceaux. Pour m fixe et pour tout $q \in]1, Q[$, avec $v = \log(t)$ et $v_q = \log(t_q)$, $P(\log(t))$ s'écrit sur chaque intervalle $t \in [\, t_{q-1},\, t_{q+1}\, [$:

$$P(\log(t)) \;=\; P(v) \;=\; C_{q,2}v^2 + C_{q,1}v + C_{q,0} \tag{6}$$

$$= \sum_{i=-1}^{1} \frac{(v - v_{q-1})(v - v_q)(v - v_{q+1})}{(v_{q+i} - v_{q-1})(v_{q+i} - v_q)(v_{q+i} - v_{q+1})}\; X_k[q+i, m]. \tag{7}$$

Un simple développement de l'équation (7) en la variable v permet alors de trouver par identification les coefficients $C_{q,0}$, $C_{q,1}$ et $C_{q,2}$ qui sont les seules inconnues.

Ensuite, la fonction continue $P(\log(t))$ est échantillonnée aux points $\widetilde{t}_q$ de (4) qui sont uniformément répartis en échelle logarithmique. Cet échantillonnage ne pose pas de problème pour les petites valeurs de $\widetilde{t}_q$ puisque $P(\log(t))$ y est lentement variable en raison de la faible densité des points t_q. En revanche pour les grandes valeurs de $\widetilde{t}_q$, il y a un risque certain d'effet semblable à du repliement, puisqu'il s'agit cette fois-ci de sous-échantillonnage. Pour éviter cela, nous appliquons une intégration de $P(\log(t))$ pondérée par une fenêtre $h(v)$, de sorte à réaliser un filtrage anti-repliement :

$$\delta_t \int_{v_{q-1}}^{v_{q+1}} P\big(v\big) h\big((v - v_q)/\delta_t\big) dv = \int_{-1}^{1} P\big(\delta_t w + v_q\big) h\big(w\big) dw. \tag{8}$$

Connaissant la formule analytique de $h(w)$, avec eq. (6) nous obtenons l'expression explicite de l'intégrale de (8) pour chaque intervalle $[v_{q-1}, v_{q+1}]$. En *recollant* les morceaux, nous aboutissons à une simple combinaison linéaire des éléments de X_k, dont les coefficients sont indépendant de m et de X_k. La difficulté de ce travail comparé à un schéma standard d'intégration numérique de Simpson, cf. e.g. [2], est qu'ici les points d'origines ne sont pas uniformément répartis dans l'échelle logarithmique et qu'une pondération par $h(v)$ est ajoutée dans l'intégrale. De plus nous avons réalisé un découpage différent des intervalles pour une meilleure généralisation.

Pour les fréquences nous opérons exactement de la même manière. Par conséquent les deux conversions sont réalisées par deux combinaisons linéaires successives des $X_k[m,q]$. Ainsi, la nouvelle matrice $\mathcal{X}_k$ représentant X_k en échelles logarithmiques est donnée par le produit matriciel suivant :

$$\mathcal{X}_k = C_f X_k C_t, \tag{9}$$

où C_t et C_f sont les matrices des conversions vers les échelles logarithmiques du temps et de la fréquence respectivement. Remarquons que ces deux matrices ont la même structure parcimonieuse que les noyaux spectraux de la transformée à Q constant, cf. [3]. Ainsi ce produit matriciel peut être accéléré en adoptant l'algorithme adéquat. Notons que C_f est de taille $(M_f \times M)$ et C_t est de taille $(Q \times M_t)$.

L'implémentation de cette conversion se trouve dans le fichier `C_dblLFB.m` du sous-répertoire de l'adresse `beemusic_folder/src_matlab/toolboxes/BeeTools` et utilise les sous-fonctions suivantes : `F_swimpsoncoeff.m` et `F_swimpsonkernel.m` pour le calcul des noyaux de l'interpolation. La résolution symbolique des coefficients est données dans le fichier Maple : `Maple/GetInteg.mw`.

3.4 Division en sous-bandes de fréquences

Pour éviter que du bruit localisé en fréquence n'affecte tous les coefficients des empreintes sonores, nous réalisons une sous-division de l'intervalle des fréquences $[f_{min}, f_{max}]$. Pour se faire, 5 bandes de fréquences sont définies : avec des fréquences centrales uniformément réparties et des largeurs constantes dans cette échelle logarithmique. Elles permettent alors la séparation du spectre en 5 parties, avec recouvrement.

Soient $n_b = 5$ le nombre de bandes et α_b le taux de recouvrement tel que $\Delta_b = \alpha_b \delta_b$, où Δ_b est la largeur des bandes et δ_b l'écartement des centres de 2 bandes consécutives en échelle logarithmique. Avec f_c en herzts la fréquence centrale de la bande d'indice c, et avec $\nu_c = \log(f_c)$, nous avons l'expression :

$$\log(f_c) = \nu_c = (c-1)\delta_b + \nu_1,\ \forall c \in [1, n_b]. \tag{10}$$

Pour que cette sous-division recouvre entièrement l'intervalle [$\log(f_{min}), \log(f_{max})$], la première et la dernière fréquences centrales doivent être : $\nu_1 = \log(f_{min}) + \Delta_b/2$ et $\nu_{n_b} = \log(f_{max}) - \Delta_b/2$, et l'égalité suivante doit être vérifiée :

$$\log(f_{max}) - \log(f_{min}) = (n_b - 1)\delta_b + \Delta_b = (n_b - 1)\delta_b + \alpha_b\delta_b, \tag{11}$$

on en déduit alors δ_b :

$$\delta_b = \frac{1}{n_b - 1 + \alpha_b} \log\left(\frac{f_{max}}{f_{min}}\right). \tag{12}$$

Par exemple, avec un taux $\alpha_b = 2$, et des fréquences aux extrémités $f_{min} = 150$ Hz et $f_{max} = 5000$ Hz, les 5 fréquences centrales sont environ : [269, 483, 866, 1554, 2787] Hz, et leur largeur de bande représente environ 20 demi-tons en musique. La figure 2 illustre cette sous-division.

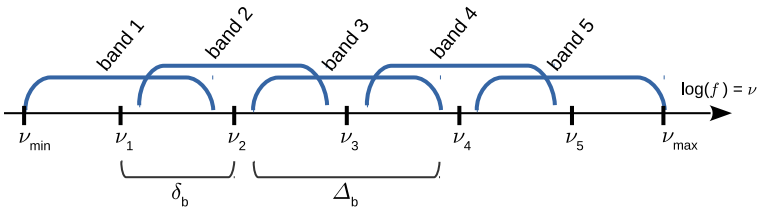

FIGURE 2 – Illustration de la sous-division en 5 bandes de fréquences.

Définissons alors pour une fenêtre k et pour une bande c, la matrice $\mathcal{X}_{k,c}$ représentant un "carreau", c'est-à-dire une portion du spectrogramme en temps et en fréquences. Avec M_f le nombre de log-bins entre f_{min} et f_{max}, chacun des 5 carreaux ainsi définis possède environ $m_f = \Delta_b\ (M_f - 1)\ /\ \log(f_{max}/f_{min}) + 1$ log-bins. Alors les matrices $\mathcal{X}_{k,c}$ sont toutes de taille $(m_f \times M_t)$.

Remarque : contrairement à la méthode de Shazam où les empreintes sont données par des informations localisées en temps et en fréquence, cf. [24], il s'agit ici d'une description distribuée en temps sur [$t_{min},\ t_{max}$], et distribuée en fréquence sur [$\log(\nu_c - \Delta_b/2),\ \log(\nu_c + \Delta_b/2)$], pour chacune des fenêtres k prises à la trame p_k, et chacune des bandes c centrées en $f_c = \exp(\nu_c)$.

3.5 Modification des amplitudes

La modification des amplitudes de $\mathcal{X}_{k,c}[q,m]$ en $\mathcal{Y}_{k,c}[q,m]$ a été résumée en section 2.1. Voici maintenant quelques détails des 4 étapes qui la composent.

Redressement : Cette opération consiste à ramener à une valeur donnée tous les "pixels" de valeur inférieure. Ici le seuil dépend de la valeur maximale du carreau considéré, avec la relation suivante :

$$\text{avec} \quad s_r = g \max_{m,q} \{\ \mathcal{X}_{k,c}[m,q]\ \}, \tag{13}$$

$$\begin{aligned} \mathcal{X}_{k,c}[m,q] \quad &\leftarrow \quad s_r, \qquad\qquad \text{si} \ \ \mathcal{X}_{k,c}[m,q] < s_r, \\ &\leftarrow \quad \mathcal{X}_{k,c}[m,q], \quad \text{si} \ \ \mathcal{X}_{k,c}[m,q] \geq s_r, \end{aligned} \tag{14}$$

où g est un paramètre positif et inférieur à 1. Si la valeur de $\mathcal{X}_{k,c}[m,q]$ est inférieure à s_r, alors elle est ramenée à s_r, et si elle est supérieure, elle reste inchangée : on parle alors de *redressement*.

Comme il a été dit en section 2.2, le but de cette opération est de réduire l'effet d'un bruit additif sur la valeur finale de l'empreinte. En effet, si le spectre du bruit additif reste à un niveau globalement inférieur aux maxima du signal original, alors le redressement a la capacité de *masquer* le spectre du bruit comme l'explique l'exemple suivant : prenons le cas d'un signal original pour lequel $\mathcal{X}_{k,c}[m,q] = 0$, alors le redressement ramène ce pixel à s_r pour le calcul des empreintes de la base. Si en présence de bruit, la valeur du pixel change mais reste inférieure à s_r, alors le redressement la ramène à s_r et le pixel n'a alors aucune différence entre la version originale et la version altérée. Si un bruit plus fort change la valeur du pixel en $s_r + \epsilon$, avec $\epsilon > 0$, alors le redressement le laisse à cette valeur. Cependant, dans ce cas l'effet du bruit est malgré tout réduit car la différence entre la version originale et la version altérée n'est plus que de ϵ, et non $s_r + \epsilon$.

Notons que la mise à 0 des pixels inférieurs à s_r aurait un bon effet pour des bruits faibles, mais très mauvais pour des bruits plus forts, puisque la différence serait cette fois-ci de $s_r + \epsilon$. Cette opération appelée *noise gate* n'est donc pas souhaitable comparée au redressement proposé ici.

Enfin, remarquons qu'une valeur élevée de g permet de réduire efficacement l'effet du fond bruité du spectrogramme, mais supprime par la même l'information nécessaire pour la reconnaissance. Il faut donc trouver le bon compromis ; cf. la section 5.

Pondération 2D : Afin d'éviter les effets de bords lors de la transformée de Fourier 2D qui suit, une pondération est faite pour amener doucement à 0 les pixels aux 4 bords du carreau.

En pratique il suffit de multiplier terme-à-terme la matrice $\mathcal{X}_{k,c}$ du carreau, après redressement, par un masque de pondération H de même taille $(m_t \times M_f)$. Pour simplifier H, nous faisons une séparation des variables m et q, c'est-à-dire : $H[m,q] = h_f[m]\, h_t[q]$, où h_f et h_t sont des fenêtre de pondération 1D de taille m_f et M_t respectivement. La pondération consiste donc à remplacer $\mathcal{X}_{k,c}[m,q]$ par :

$$\mathcal{X}_{k,c}[m,q] \;\leftarrow\; \mathcal{X}_{k,c}[m,q]\, H[m,q] = \mathcal{X}_{k,c}[m,q]\, h_f[m]\, h_t[q]. \tag{15}$$

Nous testerons par la suite la fenêtre de Hann 2D et la fenêtre cosinus 2D définies pour tout $m \in [1, M_f]$ et pour tout $q \in [1, M_t]$ par :

$$\text{Fenêtre de Hann :} \quad H_{m,q} \;=\; \frac{1}{4}\left(1 - \cos\left(\frac{2\pi\, m}{M_f + 1}\right)\right)\left(1 - \cos\left(\frac{2\pi\, q}{M_t + 1}\right)\right), \tag{16}$$

$$\text{Fenêtre cosinus :} \quad H_{m,q} \;=\; \cos\left(\frac{\pi\, m}{M_f + 1} - \frac{\pi}{2}\right) \;\; \cos\left(\frac{\pi\, q}{M_t + 1} - \frac{\pi}{2}\right), \tag{17}$$

Remarquons qu'un autre intérêt de cette pondération est la robustesse au changement d'échelles. En effet, comme il a été résumé, une dilatation/compression des échelles de temps ou de fréquences, en linéaire, devient une translation dans les échelles logarithmiques. Cependant, au sens strict le module de la transformée de Fourier discrète est invariant aux translations *circulaires*, c'est-à-dire que les données qui sortent d'un bord à cause de la translation, reviennent par le bord opposé. Or dans notre cas il s'agit d'une translation simple. Cette pondération aux bords permet alors de limiter cette différence.

Normalisation L_∞ : Ensuite, une normalisation de la norme L_∞ de $\mathcal{X}_{k,c}$ est effectuée. Il s'agit simplement de diviser la valeur de chaque pixel par la valeur maximale sur le carreau. Puisqu'à cette étape chaque pixel a une valeur positive, cette opération garantit d'avoir des valeurs entre 0 et 1. Cette normalisation consiste donc à remplacer $\mathcal{X}_{k,c}[m,q]$ par :

$$\mathcal{X}_{k,c}[m,q] \;\leftarrow\; \frac{\mathcal{X}_{k,c}[m,q]}{\max\limits_{m,q}\{\mathcal{X}_{k,c}[m,q]\}}. \tag{18}$$

Comme nous l'avons précisé plus tôt, cette normalisation permet d'être invariant à toute multiplication du signal original. Remarquons qu'il aurait été possible de supprimer l'effet d'un gain en éliminant certaines valeurs de la transformée de Fourier discrète 2D, mais cette normalisation est un moyen plus simple et plus efficace. De plus, de la sorte chaque valeur de la DFT2D reste potentiellement informative.

Conversion *log/lin* : Enfin, un changement d'échelle est réalisé. Ce changement du type $y = f(x)$ est donné par l'équation suivante :

$$\mathcal{X}_{k,c}[m,q] \;\leftarrow\; f_b\left(\mathcal{X}_{k,c}[m,q]\right) = \frac{\log(1 + b\, \mathcal{X}_{k,c}[m,q])}{\log(1+b)}, \tag{19}$$

avec b un paramètre strictement positif. Cette fonction se comporte comme un changement linéaire pour les valeurs proches de 0, et se comporte comme un changement logarithmique pour les valeurs proches de 1. Aussi un paramètre b petit permet de favoriser le comportement linéaire, et un b grand favorise au contraire le comportement logarithmique.

Comme nous l'avons vu en section 2.2, le comportement logarithmique permet de rejeter l'effet d'une possible égalisation ou d'un filtrage sur l'axe vertical de la DFT2D. L'effet d'un filtrage se manifeste par le produit du spectre original $S(f,t)$ avec une fonction fréquentielle $G(f)$, qui est la réponse du filtre. Etant donné que le logarithmique change le produit en addition, on a :

$$X(f,t) = S(f,t)G(f) \quad \Rightarrow \quad \log(X(f,t)) = \log(S(f,t)) + \log(G(f)). \tag{20}$$

Alors, puisque $\log(G(f))$ est indépendant du temps, sa contribution se retrouve isolée dans la moyenne temporelle de $\log(X(f,t))$, et laisse les autres valeurs de la DFT2D inchangées.

Cependant, ce logarithme a aussi pour effet de "tasser" les amplitudes du spectrogramme, et donc il rehausse d'éventuelles contributions résiduelles du bruit additif. Ainsi, avec un choix judicieux de b, on peut espérer éviter cet effet indésirable, tout en conservant le premier effet souhaité.

Remarquons que pour tout $b > 0$, cette fonction est bijective et strictement croissante sur $[0, 1]$. Ceci signifie que $f_b(0) = 0$ et $f_b(1) = 1$, et donc que chaque pixel a une valeur positive et inférieure à 1.

3.6 Transformée de Fourier à 2 dimensions

Enfin, la transformée de Fourier discrète à 2 dimensions (DFT2D) est réalisée sur la matrice $\mathcal{Y}_{k,c}$ obtenue après les changements d'amplitude de $\mathcal{X}_{k,c}$ précédemment données ; et le module en est extrait pour la robustesse au changement d'échelle.

Sans remplissage de 0, *0-padding* en anglais, cette opération fournit une nouvelle représentation 2D de chaque carreau, de même dimension $(m_f \times M_t)$, pour laquelle les variables sont relatives à la fréquence des log-fréquences d'une part et à la fréquence des log-temps, d'autre part. Avec $\tilde{m} \in [1, m_f]$ et $\tilde{q} \in [1, M_t]$ les indices des fréquences des log-fréquences et des log-temps respectivement, la transformée de Fourier discrète à 2 dimensions donne :

$$Y_{k,c}[\tilde{m}, \tilde{q}] = \Big| \mathrm{DFT}_{2D}\big\{\mathcal{Y}_{k,c}\big\}(\tilde{m}, \tilde{q}) \Big| = \Big| \sum_{m=1}^{m_f} \sum_{q=1}^{M_t} \mathcal{Y}_{k,c}[m, q] \ \exp\left(-j2\pi \left(\frac{\tilde{m}m}{m_f} + \frac{\tilde{q}q}{M_t}\right)\right) \Big|. \tag{21}$$

En raison de la symétrie de la transformée de Fourier, les fréquences des log-temps d'indices supérieurs à $m_t = \lfloor (M_t/2) + 1 \rfloor$ sont supprimés afin d'éviter les redondances d'information. Nous obtenons alors une représentation de taille ($m_f \times m_t$), qui une fois vectorisée fournit une empreinte sonore de haute dimension : $m_f m_t$. Notons qu'il reste de la redondance sur l'axe vertical en $\tilde{q} = 1$ et en $\tilde{q} = m_t$ si M_t est impair. Nous avons choisi de la laissée car elle sera retirée automatiquement lors de la réduction de dimension, non expliquée dans ce document, cf. [16].

Remarque : si m_f et M_t sont des puissances de 2, alors l'algorithme de Fourier rapide peut-être appliqué, cf. [5]. Dans le cas contraire l'algorithme FFTW peut être utilisé, cf. [9]. Il s'agit d'un algorithme rapide pour des dimensions qui ne sont pas des puissances de 2, et qui n'utilise pas de remplissage de 0, 0-padding. Les fonctions `fft` et `fft2` du logiciel Matlab implémentent l'algorithme.

4 Récapitulatif des paramètres et autres tests

Dans cette partie nous faisons un récapitulatif complet des paramètres pour le calcul des empreintes sonores. Enfin quelques idées supplémentaires sont données pour une amélioration possible de la robustesse. Mais avant tout, nous donnons quelques changements de paramètres utilisés par la suite.

4.1 Changements de paramètres

Pour faciliter le paramétrage de b nous faisons le changement de paramètres : $b = 10^a$ ou $a = \log_{10}(b)$. Ainsi, $b > 0$ pour tout $a \in \mathbb{R}$, et $a = 0$ correspond à $b = 1$.

Nous avons dans la plupart des tests voulu paramétrer M_f, et m_f, via la résolution souhaitée entre 2 log-bins consécutifs. Nous définissons alors le paramètre r_h égal à cette résolution en demi-tons. Alors :

$$M_f \leftarrow 1 + \lfloor \ \frac{12}{r_h} \log_2 \left(\frac{f_{max}}{f_{min}}\right) + 0.5 \ \rfloor. \tag{22}$$

Ici $\lfloor x + 0.5 \rfloor$ représente la partie entière la plus proche de x, fonction `round` sous Matlab par exemple. Et nous avons les conversions suivantes, selon que l'on souhaite m_f en fonction de M_f, ou l'inverse :

$$m_f \leftarrow \lfloor \ (M_f - 1) \ \frac{\Delta_b}{\log(f_{max}/f_{min})} + 0.5 \rfloor + 1, \tag{23}$$

$$M_f \leftarrow \lceil \ (m_f - 1) \ \frac{\log(f_{max}/f_{min})}{\Delta_b} \ \rceil + 1, \tag{24}$$

où cette fois $\lceil x \rceil$ représente la partie entière supérieure ou égale à x, fonction `ceil` sous Matlab. Remarquons que nous utilisons ici l'opérateur d'affection "$\leftarrow$" en raison des parties entières qui n'autorisent pas la réciproque au sens strict.

Enfin, les tailles de fenêtres d'analyse et le pas d'avancement seront données en milli-secondes : $w_s = 10^3 W_s/F_s$ et $h_s = 10^3 H_s/F_s$, pour le calcul du spectrogramme, cf. sec. 3.1.

4.2 Liste des paramètres

La liste complète des paramètres modifiables pour le calcul des empreintes est donnée en tab. 1. Cependant un petit nombre d'autres paramètres sont fixes ou dépendent d'autres paramètres, les voici :

— F_s : Taux d'échantillonnage, fixé à 11050 Hz, voir section 3.1.

— M_s : Taille de la transformée de Fourier discrète en bins, qui dépend de la valeur de W_s. Sa valeur est deux fois la puissance de 2 suivant W_s, supérieure ou égale : $M_s = 2^{\lceil \log_2(W_s)+1 \rceil}$. Il vient alors M la taille du spectre unilatéral limité à la fréquence de Nyquist : $M = M_s/2 + 1$.

— Fenêtre de pondération du spectrogramme : fenêtre de Hann, cf. [12].

Param.	Rôle
w_s	Taille de la fenêtre d'analyse en milli-secondes $W_s = w_s F_s 10^{-3}$
h_s	Pas d'avancement en ms, $H_s = h_s F_s 10^{-3}$
f_{min}	Fréquence minimale de l'échelle logarithmique des fréquences en Hz, cf. sec. 3.3
f_{max}	Fréquence maximale de l'échelle logarithmique des fréquences en Hz
t_{min}	Premier temps de l'échelle logarithmique du temps en secondes, cf. sec. 3.3
t_{max}	Dernier temps de l'échelle logarithmique du temps en secondes
M_t	Nombre de *log-samples* d'une fenêtre d'observation en échelle log., cf. secs. 3.2 et 3.3
n_b	Nombre de bandes de fréquences
α_b	Taux de recouvrement pour la sous-division des fréquences en bandes, cf. sec. 3.4
m_f	Nombre de *log-bins* d'un carreau, pour une bande de fréquence, cf. sec. 3.4. Ou bien on peut choisir de paramétrer par r_h ou M_f, en utilisant les conversions des équations (22), (23) et (24)
`win`	Forme de la fenêtre de pondération avant la `DFT2D`. Fenêtre `cos` ou `hann`, cf. sec. 3.5
g	Paramètre du redressement des amplitudes
a	Paramètre de la conversion *log/lin* des amplitudes de la section 3.5

TABLE 1 – Listes des paramètres modifiables.

4.3 Autres tentatives

En section 5 nous testons plusieurs combinaisons de valeurs possibles de paramètres afin de nous aider dans le choix difficile des valeurs. Mais nous y avons aussi testé d'autres variantes pour le calcul des empreintes. Voici un aperçu de ces idées :

Blanchiment L'idée du *blanchiment* est d'aplatir l'enveloppe spectrale du spectrogramme afin d'éliminer l'effet d'une égalisation, et donc augmenter la robustesse.

En pratique, la méthode développée consiste dans un premier temps à déterminer la fonction s_{max} qui donne pour chaque fréquence m, log-bin, la valeur maximale de $\mathcal{X}_k$ au long du temps q :

$$s_{max}[m] = \max_q \mathcal{X}_k[m, q]. \tag{25}$$

Puis les maxima locaux de s_{max} sont cherchés. Nous utilisons ici une sorte de *filtre maximum*, semblable à un filtre médian, qui assure un écart minimal entre deux pics estimés. Remarquons qu'ici s_{max} est dans une échelle logarithmique des fréquences, comme $\mathcal{X}_k$. Ensuite, l'enveloppe spectrale est estimée par une interpolation en fonctions *splines* des maxima locaux obtenus. Cela produit l'estimée s_{env} de l'enveloppe spectrale, et le blanchiment s'opère par la division :

$$\mathcal{X}_k[m, q] \leftarrow \frac{\mathcal{X}_k[m, q]}{s_{env}[m]}. \tag{26}$$

Notons que si cette opération est effectuée, elle est réalisée avant la sous-division en 5 bandes de fréquences, et le redressement n'est alors pas réalisé. Aussi, elle comprend un paramètres correspondant à la taille du filtre maximum, c'est-à-dire l'écart minimal entre 2 pics détectés en log-bins. Nous noterons ce paramètre σ_f, en nombre de log-bins.

Remarquons que nous aurions pu utiliser une méthode connue pour la détection de l'enveloppe spectrale à partir de s_{max}, voir e.g. [22, 4, 18], mais nous avons préféré cette méthode en raison de sa simplicité.

Ordre des modifications d'amplitude Nous avons décrit en section 3.5 quatre modifications des amplitudes : *redressement, pondération 2D, normalisation* L_∞ et *conversion log/lin*.

L'ordre d'exécution de ces opérations peut avoir une influence sur le résultat de l'identification audio, parce que deux ordres différents peuvent produire des résultats différents. Par exemple : placer la pondération 2D après la normalisation peut changer les choses car la valeur maximale peut changer si le maximum n'est pas vers le centre de la fenêtre.

Par conséquent, en section 5, nous testons également plusieurs agencements possibles de ces changements d'amplitudes.

5 Choix des valeurs de paramètres

Le processus de calcul précédemment décrit comporte un grand nombre de paramètres différents, tels que : les paramètres du spectrogramme, paramètres pour le fenêtrage 2D, pour la conversion en échelle logarithmique, et pour la modification d'amplitude, par exemple a et g. Nous avons du décider de leurs valeurs. Le critère choisi pour cette sélection est la *robustesse* aux dégradations sonores, mais même si le lien entre valeurs et performance peut parfois être intuité, il n'est pas facile dans la plupart des cas. De plus, il existe très probablement une interaction entre paramètres, et un compromis doit être trouvé pour certains. Par exemple, pour le niveau s_r du redressement, une valeur trop faible de g ne produit aucun bénéfice par rapport au bruit, et une valeur trop forte a pour effet de masquer une partie importante de l'information, et donc de décroître les performances.

5.1 Procédure d'évaluation

Plutôt que de réaliser une *optimisation* de paramètres standard sur le système complet d'identification audio pour l'évaluation, nous avons simplifier l'approche pour la rendre plus facile à calculer. Premièrement pour éviter de calculer le système complet, nous avons dérivé deux fonctions coût intermédiaires à minimiser, qui sont plus simple à évaluer et qui sont fortement liées au problème de reconnaissance. Deuxièmement l'approche retenue consiste en une approche *essai/erreur* : plusieurs tests ont été réalisés, pour plusieurs ensembles de valeurs de paramètres, et une interface graphique permet de visualiser facilement les résultats, et de choisir manuellement les paramètres testés aux prochains tests.

5.1.1 Dégradations

Premièrement, plusieurs fichiers audio sont sélectionnés pour le test, et chacun d'entre eux est dégradé par un certain nombre de dégradations différentes. Les empreintes sonores sont alors calculées sur les signaux originaux et dégradés, et ce, successivement pour plusieurs ensembles de valeurs de paramètres prédéfinies.

Pour le test de robustesse aux altérations, nous avons réalisé plusieurs dégradations différentes des sons originaux. Ces dégradations sont réalisées par la boîte à outils : *BeeAlter Toolbox*, cf. [13]. Pour obtenir un certain réalisme, nous avons mis à la chaîne plusieurs types de dégradations, avec à chaque fois une dégradation dominante. Ici sept types de dégradations ont été testés. En outre, un coefficient α permet aussi de régler le niveau d'altération. Chaque dégradation a été réglée de sorte que $\alpha = 1$ produit une altération marquée mais raisonnable, et $\alpha = 2$ produit une altération très forte, voir exagérée. Cette section présente la liste des dégradations testées et l'effet de α.

Bruits ambiants : Il s'agit d'un bruit additif enregistré dans un restaurant à un moment d'affluence. Ici le coefficient α agit directement sur le rapport signal-à-bruit, dont l'acronyme anglais est SNR. Par exemple il vaut environ 3dB pour $\alpha = 1$.

Bruit rose synthétique : Ce bruit synthétique a un spectre de pente $-10dB$ par décade. Il est bien connu puisqu'il donne une impression perceptive de bruit blanc. De même, ici α agit sur le SNR.

Distorsion : Ici l'altération est donnée par un écrêtage des valeurs extrêmes, par rapport à une valeur de seuil donné par α. Par exemple pour $\alpha = 1$, le seuil est réglé de sorte à ce que 30 % des échantillons soient écrêtés.

Egalisation : Un égaliseur graphique est utilisé, pour lequel la courbe de gain est modifiée par α de la manière suivant : la réponse varie entre $\alpha[-15, +15]$ décibels.

Transposition fréquentielle : La transposition consiste à déplacer le contenu fréquentiel sans modifier l'échelle du temps comme le ferait un changement de vitesse de lecture. Ici les fréquences sont transposées vers les basses fréquences, avec une valeur de -2α demi-tons.

Effet *wow* : Cette altération est simulée par un filtre à retard variable pouvant représenter par exemple l'effet d'un désaccordage, *detuning*, ou un effet Doppler *cyclique*. Ici α agit directement sur la fréquence de modulation et l'amplitude du retard.

Décalage temporel : Pour simuler une possible erreur dans la décision des instants d'analyse, cf. secs. 2.2 et 3.3 et [14], nous avons réalisé un décalage manuel entre le temps d'analyse pour le signal original et le signal dégradé. Ici le décalage vaut 80α ms.

Comme il a été dit, une chaîne de dégradations a été mise en œuvre, avec une dégradation dominante. Les dégradations listées précédemment sont tour à tour dominantes. Dans la suite, lorsque nous parlons du test d'une altération, il s'agit en réalité d'une altérations dominante, testée simultanément avec d'autres altérations faibles. Par exemple, pour l'ajout du bruit ambiant, le son est au préalable altéré par une légère égalisation, un petit effet *wow*, et une faible transposition, puis un fort ajout de bruit est fait, qui est dans ce cas dominant. Cela permet de prendre en compte plusieurs dégradations simultanément, pour plus de réalisme, mais avec une dégradation plus marquée.

5.1.2 Regroupement des coefficients des empreintes

Les critères donnés en section 5.1.3 sont basés sur un calcul de distances euclidiennes entre les empreintes originales et les empreintes du son altéré. Cependant au lieu de réaliser les distances sur les $m_f m_t$ coefficients des $Y_{k,c}$, nous réalisons un regroupement via un filtrage en 2 dimensions.

Ce regroupement se comporte en fait comme un sous-échantillonnage de la matrice $Y_{k,c}$, avec filtrage anti-repliement, d'une taille $(m_f \times m_t)$ vers une taille $(p_f \times p_t)$, où p_f et p_t sont les tailles selon les axes des fréquences des log-fréquences et des log-temps respectivement. Nous avons fixer $p_f = p_t = 5$ de sorte à aboutir à des vecteurs de dimensions 25 pour le calcul des distances euclidiennes. Notons Z_i les vecteurs ainsi obtenus par regroupement des $Y_{k,c}[\tilde{m}, \tilde{q}]$.

Ici, les facteurs de sous-échantillonnage κ_f et κ_t sont choisis en fonction de (m_f, p_f) et (m_t, p_t) respectivement, et de sorte à éviter les bords, notamment l'axes en $\tilde{m} = 1$ où se concentre en principe l'effet de l'égalisation, cf. sec. 2.2 et sec. 3.6. Le filtre anti-repliement utilisé est un filtre binomial, parfois utilisé en filtrage d'images. Le masque de convolution M de dimension $(r_f \times r_t)$ vaut :

$$H_{\tilde{m},\tilde{q}} = h_{\tilde{m}}\, h_{\tilde{q}}, \tag{27}$$
$$\text{avec} \;\; h_{\tilde{m}} = C_{r_f-1}^{\tilde{m}-1} = \frac{(r_f-1)!}{(r_f-\tilde{m})!\;(\tilde{m}-1)!} \;\; \text{et} \;\; h_{\tilde{q}} = C_{r_t-1}^{\tilde{q}-1} = \frac{(r_t-1)!}{(r_t-\tilde{q})!\;(\tilde{q}-1)!},$$

où C représente les coefficients binomiaux. Ici les dimensions du filtre r_f et r_t sont choisies par :

$$r_f = 2\lfloor \frac{\kappa_f + 1}{2} \rfloor + 1, \;\; \text{et} \;\; r_t = 2\lfloor \frac{\kappa_t + 1}{2} \rfloor + 1, \tag{28}$$

ce qui d'une part assure des nombres impairs, de sorte à ce que la convolution soit centrée aux points de sous-échantillonnage, et de sorte à assurer une bonne fréquence de coupure.

Remarquons que ce filtrage anti-repliement n'est pas réellement nécessaire, parce que chaque coefficient porte une information propre et il ne semble donc pas utile en principe de la combiner avec d'autres. Mais en pratique nous observons une meilleure stabilité des résultats, ce qui rend probablement les tests plus fiables. De plus, cette combinaison n'est pas motivée par une robustesse accrue aux dégradations ; un travail ultérieur permet d'estimer les meilleurs combinaisons possibles au sens d'un critère donné, cf. [16]. Ici nous nous contentons de ce regroupement pour la réalisation des tests.

5.1.3 Critères

Pour évaluer la performance des valeurs de paramètres choisies, nous devons définir un critère d'évaluation. Pour éviter de calculer tout le processus d'indexation, ce critère intermédiaire a pour but de prédire le taux de performance final. Pour cela, nous créons deux distributions, pour chaque ensemble de valeurs et chaque dégradation testée :

1. La première distribution correspond aux distances des positifs, pour laquelle chaque point $\mathcal{P}_i$ est la distance euclidienne entre une empreinte dégradée $\widetilde{Z}_j$ et son empreinte originale associée Z_i^*, obtenue sur le même signal, au même instant d'analyse, et avec les mêmes valeurs de paramètres,

$$\mathcal{P}_i = \|\widetilde{Z}_i - Z_i^*\|_2. \tag{29}$$

2. La seconde distribution correspond aux distances des négatifs, où chaque point $\mathcal{N}_i$ est cette fois-ci la distance euclidienne entre une empreinte dégradée Z_i et une empreinte originale, mais ne correspondant pas, Z_j^*.

$$\mathcal{N}_i = \|\widetilde{Z}_i - Z_{j_i}^*\|_2, \text{ avec } j_i \neq i. \tag{30}$$

La robustesse aux dégradations se traduit alors par des distances $\mathcal{P}_i$ faibles et des distances $\mathcal{N}_i$ élevées. Pour résumer cela dans un coefficient unique, nous utilisons ici l'information de Fisher permettant de décrire combien ces deux distributions sont séparées, cf. par exemple fig. 3. Cette information de Fisher constitue le premier critère utilisé. Elle est calculée séparément pour chaque bande, et aussi une version commune est faite.

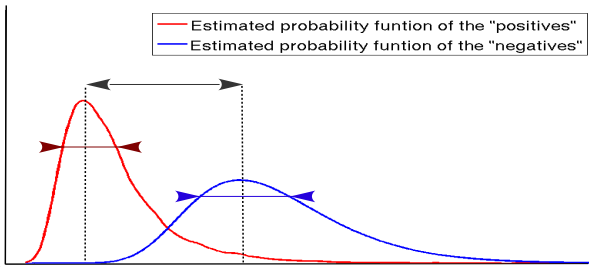

FIGURE 3 – Discrimination des deux distributions : $\mathcal{P}$ et $\mathcal{N}$. Une bonne discrimination est donnée par une variance faible (variance intra-classe) et un fort écart des moyennes (variance inter-classe). Aussi, la moyenne de la distribution $\mathcal{P}$, des positifs, doit être inférieure à celle de la distribution $\mathcal{N}$, des négatifs.

Un second critère est utilisé pour prédire les performances de reconnaissance. Dans le travail initial Ircam-AudioID de [20, 21], la reconnaissance était basée sur un calcul des *k-plus proches voisins* (*k*-nn), alors nous prédisons ici le taux de réussite du problème. Pour se faire, normalisons $\mathcal{P}$ et $\mathcal{N}$ pour obtenir les densités de probabilité de la distance euclidienne pour deux empreintes correspondante au même signal, $\mathcal{P}$, et ne correspondant pas, $\mathcal{N}$. Ainsi, ce nouveau critère donne la probabilité de réussite, c'est-à-dire la probabilité que l'empreinte du signal original Z_i^* est parmi les k-plus proche voisins de $\widetilde{Z}_i$, sachant une base contenant V empreintes originales.

Soient $P_{\mathcal{P}}(\delta)$, et $P_{\mathcal{N}}(\delta)$ respectivement, la densité de probabilité que la distance d'un positif, resp. négatif, vale δ. Ces densités sont obtenues par estimation et normalisation sur les distributions $\mathcal{P}$ et $\mathcal{N}$, cf. fig. 3. Alors on a P_V^k la probabilité que parmi V tirages de négatifs, exactement k ont une distance inférieure à la distance du positif :

$$P_V^k = C_V^k \int_{-\infty}^{\infty} P_{\mathcal{P}}(\delta) Q_{\mathcal{N}}(\delta)^k \left(1 - Q_{\mathcal{N}}(\delta)\right)^{V-k} d\delta, \quad \text{avec} \quad Q_{\mathcal{N}}(\delta) = \int_{-\infty}^{\delta} P_{\mathcal{N}}(x) dx. \tag{31}$$

Alors le critère associé vaut : $\bar{Q}_V^K = \sum_{k=0}^{K} P_V^k$, il s'agit de la probabilité que parmi V tirages de négatifs, moins de K ont une distance inférieure à la distance du positif. Ce critère traduit donc la probabilité de succès pour un problème K-nn avec une base de V empreintes.

Remarquons que pour V et K relativement grand, le calcul du coefficient du binôme C_V^k pose des problèmes de résolution numérique avec l'implémentation de Matlab, fonction `nchoosek`. Même si nous avons pu résoudre ce problème d'implémentation, la lenteur du calcul impose de les choisir petits. Dans les tests nous choisissons alors $V \approx 1000$ et K de l'ordre de 0 à 2.

5.1.4 Interface graphique

Pour chaque test, un long calcul est fait : calcul de toutes les dégradations, par type et par force α, pour chaque morceau sélectionné (environ un millier) ; et calcul des empreintes avec chaque ensemble de valeurs de paramètres. Les résultats sont stockés sur le disque, et une interface graphique permet alors de visualiser aisément les résultats des deux critères. Remarquons que ce calcul dure près de 24 heures.

Cette interface graphique, illustrée en fig. 4, rassemblent sur 2 graphiques toutes les évaluations faites des 2 critères : information de Fisher et probabilité de succès $k$nn. L'abscisse correspond à un ensemble de valeurs de paramètres testés. Une partie inférieure permet d'afficher les valeurs numériques des paramètres à la position du pointeur de souris, et les valeurs numériques des critères correspondant, donnés bande par bande et en commun. Un premier menu déroulant, en haut à gauche, permet de choisir le type de dégradation, un deuxième permet de choisir α, et un dernier permet d'ordonner les ensembles de valeurs de paramètres par ordre croissant d'un critère.

Nous cherchons à maximiser ces deux critères simultanément. Après plusieurs lots de tests, plusieurs élimination d'ensemble, présélections, et raffinement des valeurs de paramètres, nous avons finalement pu choisir les valeurs définitives.

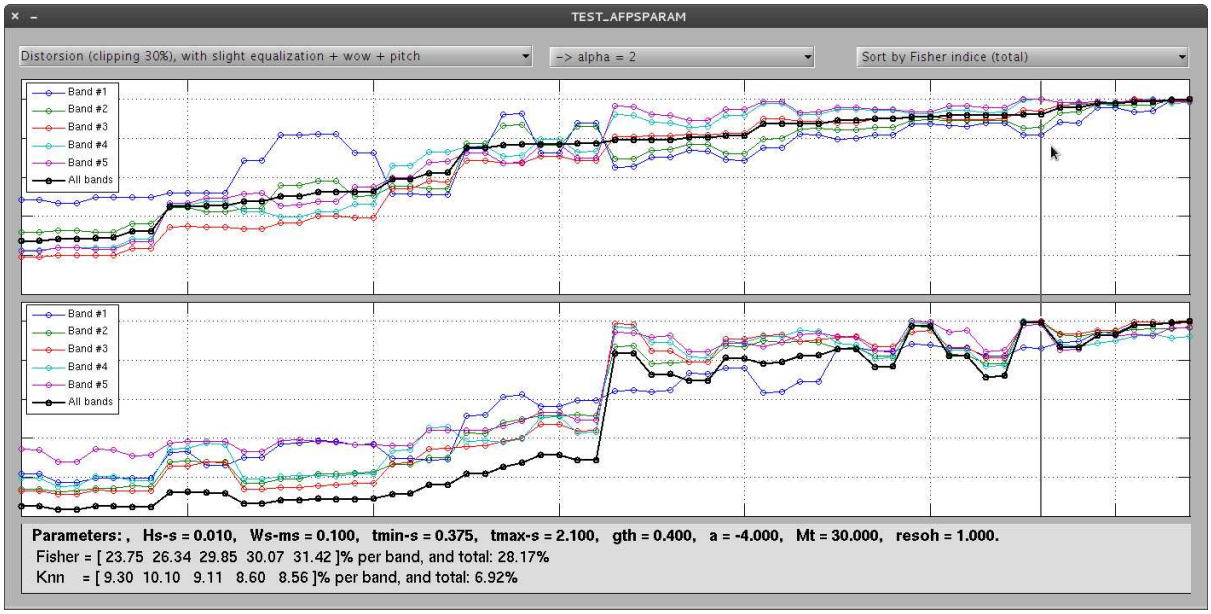

FIGURE 4 – Interface pour la sélection des paramètres des empreintes sonores. Pour chaque ensemble de valeurs testées, en abscisses, les deux graphiques donnent respectivement le critère relatif à Fisher, et le critère basé sur les $k$nn. La partie grisée inférieure donne le détail des valeurs des paramètres et des évaluations, correspondant au test survolé par le pointeur. Précisons que chaque courbe est normalisée séparément.

5.1.5 Quelques notes sur le code source

Nous donnons ici quelques informations sur le code source Matlab utilisé pour les tests. Voir le répertoire : `beemusic_folder/src_matlab/divers/dft2d_ChooseParams`. Les résultats de chaque test est stocké sur l'ordinateur literal de l'équipe analyse/synthèse de l'IRCAM, dans le répertoire : `/data/.../mignot/outputdata/afpeval/analysis`n, où n est le numéro du test. Notons que le code source utilisé pour chaque test est sauvegardé dans le sous-répertoire : `src_used`. Voici maintenant une explication des fichiers qui s'y trouvent :

- ✓ `MAKE_AFPEVAL.m` : Ce script est le premier à lancer. Il sélectionne successivement plusieurs morceaux, il calcule les différentes dégradations et les empreintes sonores avec les différents ensembles de valeurs à tester. Les empreintes sont alors stockées provisoirement sur le disque.
- ✓ `MAKE_POSTEVAL.m` : Une fois toutes les empreintes calculées et stockées, ce script les recharge morceau par morceau, et fait l'estimation des distributions $P_{\mathcal{P}}(\delta)$ et $P_{\mathcal{N}}(\delta)$ utilisées pour l'évaluation des critères pour chaque dégradations et valeurs. Les distributions sont alors stockées sur le disque.
- ✓ `TEST_AFPSPARAM.m` : Ce script lance l'interface graphique représentée en figure 4, avant quoi les valeurs des critères sont calculées à partir des distributions chargées.
- ✓ `TEST_AFPSPARAM.fig` : Il s'agit d'un fichier de ressources utilisé par l'utilitaire Matlab Guide pour la création de l'interface graphique.
- ✓ `PARAMS_FIX.m` : Cette fonction contient toutes les valeurs des paramètres fixes.
- ✓ `PARAMS_VAR.m` : Cette fonction contient toutes les valeurs des paramètres testés. Ces valeurs remplacent celles de `PARAMS_FIX.m`.
- ✓ `PARAMS_AFPEVAL.m` : Contient différents paramètres globaux, pas reliés aux empreintes.
- ✓ `F_Get_Alterations.m` : Cette fonction contient tous les paramètres des dégradations sonores de la bibliothèque BeeAlter.
- ✓ `COMMON_NAMES.m` : Cette fonction contient les noms de dossiers, et fichiers, partagés entre les différents scripts.
- ✓ `C_evalafp.m` : Il s'agit de la classe pour le calcul des empreintes. L'implémentation des empreintes décrite en sec. 3.5 s'y trouve. Notons que plusieurs versions peuvent exister selon le test

(C_evalafp2.m, C_evalafp3.m, etc.). Cela est du aux différentes alternatives possibles.

✓ F_afpdist.m : Calcule de distance eucliennes des empreintes.

✓ F_fisherc.m : Calcul du premier critère relatif à l'information de Fisher.

✓ F_Pnk.m : Calcul du second relatif à la probabilité de réussite du problème k-nn.

✓ handleData.m : Classe spécifique pour l'interface graphique.

D'autres fonctions sont utilisées et se trouvent dans d'autres répertoires correspondant à des librairies. Par exemple la librairie du répertoire beemusic_folder/src_matlab/toolboxes/BeeAlter contient le code source de la boîte à outils BeeAlter, et la librairie du sous-répertoire à l'adresse beemusic_folder/src_matlab/toolboxes/BeeTools contient les fonctions principales pour l'indexation Audio.

Cette dernière librairie contient entre autres le fichier C_dblLFB.m pour la conversion logarithmique des échelles du temps et de la fréquence, cf. sec. 3.3. Ainsi que la classe C_IAP la classe pour le calcul final des empreintes. Cette classe n'est pas utilisée dans les tests qui suivent puisqu'elle a été implémentée après les résultats.

Remarquons que l'implémentation initiale de C_evalafp.m contient aussi une version binarisée des empreintes. Cette méthode a finalement été abandonnée dans les tests ci-dessous. Une autre binarisation plus efficace est faite après la réduction des empreintes pour le calcul des codes de hachage, cf. [17].

5.2 Tests et résultats

Nous donnons ici une description des tests successifs que nous avons réalisés. Comme dit plus tôt, il s'agit d'une approche *essai/erreur* : après avoir observé les résultats d'un test, nous avons pu fixer certains paramètres, et d'autres ont été retestés avec un raffinement des valeurs. De plus de nouvelles idées sont venues en cours de tests, tels que le blanchiment et l'ordre des modifications d'amplitudes, si bien que l'arrangement des tests n'est pas proprement rigoureux. Enfin, remarquons que la plupart de ces tests ont été très long a réaliser, de l'ordre de 24 heures.

5.2.1 Test 1

Dans ce premier test nous avons voulu voir l'effet d'un grande nombre de paramètres sur les performances. Seul un petit nombre d'entre eux ont été fixé : f_{min}, f_{max}, t_{min}, t_{max}, n_b, α_b, et win. Les autres du tableau 1 prendrons deux valeurs distinctes. La procédure de calcul correspond de point en point à celle décrite par les sections 2.1 et 3, à l'exception de l'ordre des modifications qui sont dans ce premier test : *redressement* → *normalisation* L_∞ → *conversion* log/lin → *pondération 2D.*

Paramètres testés : Voici la liste des paramètres variables et leur valeurs successivement testées : $h_s \in \{10, 15\}$ ms, $w_s \in \{100, 150\}$ ms, $g \in \{0.15, 0.4\}$, $a \in \{-4, 4\}$, $M_t \in \{30, 50\}$, et $r_h \in \{0.5, 1\}$. Remarquons qu'on a de façon équivalente : $b = 10^a \in \{10^{-4}, 10^4\}$, $m_f \in \{21, 41\}$, cf. eqs. (22) et (23), et $m_t = \lfloor (M_t + 1)/2 + 1 \rfloor \in \{16, 26\}$. Avec 6 paramètres testés sur 2 valeurs différentes chacun, nous avons alors $64{=}2^6$ combinaisons de valeurs possibles de paramètres, correspondant chacune à un ensemble testé de valeurs.

Paramètres fixés : Voici maintenant les valeurs utilisées pour les paramètres fixes : $f_{min} = 150$ Hz, $f_{max} = 5000$ Hz, $t_{min} = 0.375$ s, $t_{max} = 2.1$ s, $n_b = 5$, $\alpha_b = 2$, win= hann.

Les résultats montrent clairement l'effet prédit de a (ou de b) sur les performances avec du bruit ou de l'égalisation. En effet, un a négatif favorisant le comportement linéaire, donne d'assez mauvais résultats avec de l'égalisation mais meilleur pour l'ajout de bruit, qu'avec $a > 0$ qui favorise la robustesse au filtrage, cf. sec. 3.5. Cependant, pour l'égalisation maximale avec $\alpha = 2$, on constate que $a < -4$ donne les meilleurs résultats. Cela contredit notre analyse, mais nous le mettons sur le compte d'un très mauvais SNR aux creux de la réponse du filtre. Remarquons aussi que la robustesse de la distorsion est facilitée avec $a = -4$.

Pour les autres paramètres, nous ne remarquons pas d'effet significatif, excepté pour g qui semble donner de meilleurs résultats avec $g = 0.4$. A ce stade, la perte d'information due au redressement ne semble pas importante avec $g = 0.4$.

5.2.2 Test 2

Les calculs des empreintes sont les mêmes que précédemment. Dans ce test nous cherchons à évaluer l'effet des temps t_{min} et t_{max} sur la robustesse des empreintes.

Paramètres testés : Voici la liste des paramètres variables et leur valeurs successivement testées : $t_{min} \in \{0.125, 0.25, 0.5\}$ s, $t_{min} \in \{1.5, 2, 2.5\}$ s. Avec 2 paramètres testés sur 3 valeurs chacun, nous avons 9=3^2 ensembles de valeurs testés.

Paramètres fixés : Les paramètres précédemment variables sont fixés aux valeurs : $h_s = 10$ ms, $w_s = 100$ ms, $g = 0.25$, $a = -4$, $M_t = 60$ (donc $m_t = 31$), et $r_h = 0.7$ (donc $m_f = 30$). Les autres paramètres fixes ont les mêmes valeurs qu'en test 1.

Les résultats montrent une meilleure robustesse avec les plus fortes valeurs de t_{min} et t_{max}, c'est-à-dire 0.5 et 2.5 secondes, et ce, quelque soit le type de dégradation considéré. Nous prendrons donc dans la suite ces valeurs ci. Remarquons que nous n'avons pas testé des valeurs plus grandes parce que cela ne ferait pas sens avec l'application visée, il est malgré tout préférable de ne pas avoir t_{min} et t_{max} trop élevés, les valeurs choisies semblent être un bon compromis.

5.2.3 Test 3

Nous testons ici le blanchiment combiné avec d'autres ordres de changements d'amplitudes. Dans un premier temps, soit le blanchiment soit le redressement est réalisé, puis les autres sont réalisés avec trois ordres possibles :

1 : *conversion* log/lin → *pondération 2D*,
2 : *normalisation* L_∞ → *conversion* log/lin → *pondération 2D*,
3 : *conversion* log/lin → *pondération 2D* → *normalisation* L_∞.

Paramètres testés : Ce test a principalement lieu sur la présence du blanchiment ou du redressement, et sur l'ordre des trois autres modifications. Cependant le paramètre a est aussi testé avec les valeurs : $\{-\infty, -4\}$, avec $a = -\infty$ correspondant à l'identité, purement linéaire donc. Il y a en tout 12 combinaisons possibles.

Paramètres fixés : Voici maintenant les valeurs utilisées pour les paramètres fixes : $h_s = 10$ ms, $w_s = 150$ ms, $f_{min} = 150$ Hz, $f_{max} = 5000$ Hz, $t_{min} = 0.5$ s, $t_{max} = 2.5$ s, $n_b = 5$, $\alpha_b = 2$, $g = 0.2$, $a = -4$, $M_t = 60$ (donc $m_t = 31$), $r_h = 0.7$ (donc $m_f = 30$), et `win`= hann. Aussi la taille du filtre maximum pour le blanchiment vaut $\sigma_f = 3$.

Premièrement on constate aucune différence entre $a = -\infty$ et $a = -4$. Deuxièmement on remarque comme attendu un certain intérêt du blanchiment pour la robustesse à l'égalisation. Cependant pour les autres dégradations, l'effet du blanchiment n'est pas clair, tantôt favorable et tantôt pas. Troisièmement, on remarque parmi les ordres testés que le second ordre (*normalisation* → *conversion* → *pondération*) donne toujours les meilleurs résultats.

5.2.4 Test 4

Dans ce test, le choix du blanchiment ou du redressement est encore testé, ainsi que trois ordres différents de changement d'amplitudes :

1 : *pondération 2D* → *normalisation* L_∞ → *conversion* log/lin,
2 : *conversion* log/lin → *pondération 2D* → *normalisation* L_∞,
3 : *conversion* log/lin seule.

Ce test porte encore principalement la présence du blanchiment ou du redressement, et sur l'ordre des trois autres modifications. Ici les valeurs testées de a sont différentes : $a \in \{-4, 0\}$. Les valeurs des paramètres fixes sont les mêmes qu'au test précédent.

Nous observons encore le même effet du blanchiment sur la robustesse des empreintes à l'égalisation ce qui confirme une fois de plus notre analyse. Par contre, l'ajout du bruit et la distorsion semble perturber d'avantage les empreintes avec blanchiment. Enfin la seule chose plus ou moins visible sur l'ordre des changements d'amplitudes est que la normalisation L_∞ semble nécessaire.

5.2.5 Test 5

Cette fois-ci un seul ordre est testé : *blanchiment* → *conversion* log/lin → *pondération 2D* → *normalisation* L_∞. L'intérêt de ce test est dans la variation de la taille du filtre maximum du blanchiment pour voir si celui-ci peut avoir un rôle bénéfique. Les valeurs testées sont : $\sigma_f \in \{4, 6, 9, 12\}$. Les valeurs des autres paramètres fixes sont les mêmes qu'au test précédent, et $a = 0$.

Ce test montre que si le blanchiment est finalement utilisé, la taille du filtre devrait plutôt être de 4 car les petites valeurs de σ_f donnent les meilleurs résultats. Cependant le bruit rose est la seule dégradation qui préfère des valeurs élevées.

5.2.6 Test 6

Il s'agit d'à peu près le même test : blanchiment avec $\sigma_f \in \{6, 9\}$, ou bien redressement. Dans les 3 cas $a = -4$.

Pour des raisons que nous ignorons, les 2 critères utilisés donnent cette fois des résultats parfois inverses. Cela ne facilite pas l'interprétation des résultats. Cependant, on constate qu'il n'y a pas d'effet clairs des trois combinaisons testées sur les performances.

5.2.7 Test 7

Cinq tests ont été réalisés :

1,2,3 : Les trois premiers avec l'ordre :
blanchiment → *pondération 2D* → *normalisation* L_∞ → *conversion* log/lin,
et pour paramètre de blanchiment $\sigma_f \in \{4, 6, 8\}$.

4 : Avec l'ordre : *redressement* → *pondération 2D* → *normalisation* → L_∞ *conversion* log/lin.

5 : Avec l'ordre : *redressement* → *conversion* log/lin → *pondération 2D* → *normalisation* L_∞.

Cette fois-ci, il apparaît clairement que les ordres 4 et 5, sans blanchiment donc, donnent de bien meilleurs résultats pour toutes les dégradations, excepté l'égalisation.

A cause de la pauvre performance du blanchiment, et de sa relative complexité de réalisation, nous avons choisi de ne pas retenir cette technique. Le manque de performance avec l'égalisation pourra être éventuellement compensé en ajustant a.

5.2.8 Test 8

Cette fois-ci l'ordre est fixe : *redressement* → *pondération 2D* → *normalisation* L_∞ *conversion* log/lin. Il correspond à l'ordre 4 du test précédent. Seul le paramètre a est testé : $a \in \{-4, -2, 0, 2, 4, 6\}$. Ici donc c'est surtout l'effet de a qui est testé.

En résultat, nous voyons encore que a petit favorise la robustesse au bruit, et a élevé favorise la robustesse à l'égalisation. Pour les autres dégradations, l'effet n'est pas toujours clair, mais on remarque malgré tout une petite préférence vers un a de l'ordre de 2. Malheureusement il n'apparaît de bon compromis de a entre bruit et égalisation.

5.2.9 Test 9

Nous testons à nouveau deux ordres différents :

1 : *redressement* → *pondération 2D* → *normalisation* L_∞ *conversion* log/lin.

2 : *pondération 2D* → *redressement* → *normalisation* L_∞ *conversion* log/lin.

De plus les paramètre g et a sont testés : $g \in \{0.15, 0.25, 0.5\}$ et $a \in \{-2, 0, 2\}$. Notons que cette fois $M_t = 64$ (donc $m_t = 33$) et $m_f = 32$.

Dans ce dernier test, il s'agit premièrement de tester un nouvel ordre dans lequel le redressement est réalisé après la pondération. De plus l'effet de a est encore étudié en même temps que plusieurs valeurs de g. En effet même si $g = 0.4$ pour le redressement semblait donner de meilleurs résultats au test 1, nous craignons que la perte d'information soit trop importante.

Les résultats de ce dernier test confirment encore une fois les autres résultats sur la valeur de a. Ce que nous remarquons cette fois, c'est que l'ordre 1 semble meilleur, et que $g = 0.15$ donne en général de meilleurs résultats. Cette dernière remarque semble contredire le test 1, mais peut-être cela est du à l'ordre qui ici n'est pas le même.

5.2.10 Résultats définitifs

Le tableau 2 récapitule les valeurs finalement utilisées des paramètres dans l'implémentation finale des empreintes sonores.

Remarquons d'une part que finalement nous choisissons $a = 1$ pour favoriser légèrement l'égalisation. Malheureusement selon les résultats du test 8, on constate que cette valeur ne donne pas réellement de bons résultats ni pour le bruit ni pour l'égalisation. Vu les résultats finaux de l'identification audio, cf. [15], peut-être faudrait-il favoriser le bruit avec $a = -2$.

Aussi, la fenêtre de pondération est finalement une fenêtre `cos` bien que celle-ci ne soit pas testée dans ce document. Ce choix vient d'un autre test fait après la réduction de dimension des empreintes, cf. [16], qui semble donner de meilleurs résultats avec une fenêtre cosinus. Cela vient probablement du fait que la fenêtre cosinus est plus large et donc conserve plus d'information. Sur les résultats finaux de l'identification, cf. [15], on remarque de très bon résultats au changement d'échelles temporelle, *time stretching*, donc cette fenêtre semble un bon choix pour le temps, mais le changement d'échelle fréquentielle donne d'assez mauvais résultats, donc peut-être faudrait-il utiliser une fenêtre de Hann pour les fréquences, qui descend plus progressivement en 0. Remarque : tous ces tests ultérieurs ne sont pas décrits dans ce document.

Pour ce qui concerne les autres paramètres, ils sont conformes aux 9 tests faits ici, ainsi que l'ordre des changements d'amplitudes. Seuls m_f et m_t ont été légèrement modifiés de sorte à avoir $M_t = 64$ et $m_f = 32$, des puissances de 2 qui facilitent le calcul de la DFT2D. L'ordre des modifications d'amplitude est celui de sec. 3.5 : *redressement* $\rightarrow$ *pondération 2D* $\rightarrow$ *normalisation* L_∞ *conversion* log/lin.

Param.	w_s	h_s	f_{min}	f_{max}	t_{min}	t_{max}	m_t	m_f	n_b	α_b	g	a	`cos`
Valeur	150ms	20ms	150Hz	5000Hz	0.5s	2.5s	33	32	5	2	0.15	1	hann

TABLE 2 – Valeurs finales des paramètres.

6 Conclusion

Ce document commence par présenter la procédure de calcul des empreintes sonores utilisées pour l'identification audio du projet BeeMusic, cf. secs. 2.1 et 3. Ces détails pourront être utilisés comme référence pour une réimplémentation du calcul. Aussi, le choix des valeurs des nombreux paramètres de la méthode est donné. A titre informatif, l'approche utilisée pour le choix des paramètres est ici détaillée. Les calculs et les paramètres ont par la suite été utilisés dans l'implémentation faite des empreintes sonores, voir le code de la classe : `C_IAP`.

Non seulement la méthode est conçue de sorte à obtenir des empreintes intrinsèquement robustes à certaines altération sonores, mais en plus le choix des valeurs de paramètres a été guidé par le même but.

Contrairement à beaucoup d'autres travaux en indexation audio, les empreintes développées ici portent de l'information pertinente au sens musical et perceptif. Il peuvent être vus comme la représentation de la variation temporelle du cepstre sonore. En effet, la DFT2D est équivalente à deux DFT successives, l'une selon l'axe des log-fréquences, la seconde selon l'axe du log-temps. Ainsi, avec la conversion quasi-logarithmique des amplitudes et la conversion logarithmique des fréquences, la première DFT donne une représentation similaire aux log-cepstre ou aux MFCCs, cf. [7, 6]. Ensuite la seconde DFT selon l'axe du temps, donne une représentation de l'évolution des coefficients cepstraux au cours du temps. Nous pourrions dans ce cas appeler la représentation donnée par les empreintes : *Modulation Cepstrum*.

Pour cette raison, il est envisageable d'utiliser ces empreintes pas seulement pour obtenir des codes pour l'indexation audio, mais aussi en temps que descripteurs audio pour des tâches telles que la classification. Notons que l'interprétation données précédemment, modulation cepstrum, rappelle ce qui est parfois utilisé en classification ou en reconnaissance de la parole : les *delta-MFCCs*.

Cependant, la grande dimensionnalité des empreintes, $m_t m_f = 33 \times 32 = 1056$, rend leur utilisation difficile. Mais, dans le contexte de l'indexation audio, nous avons développé une approche permettant de réduire la dimension par une simple projection sur un sous-espace. Cette projection est apprise de sorte à réduite une fois de plus la sensibilité des empreintes *réduite* aux dégradations sonores, cf. [16]. Notons que cette approche peut éventuellement être appliquée pour d'autres applications. Dans ce cas il serait nécessaire d'adapter l'apprentissage de la projection.

Références

[1] E. Allamanche, J. Herre, O. Hellmuth, B. Fröba, T. Kastner, and M. Cremer. Content-based identification of audio material using MPEG-7 low level description. In *Int. Symposium on Music Information Retrieval (ISMIR'01)*, October 2001.

[2] K. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, 1990. 2cd edition.

[3] J. Brown and M.S. Puckette. An efficient algorithm for the calculation of a constant Q transform. *J. Audio Eng. Soc.*, 92(5) :2698–2701, 1992.

[4] O. Cappé, J. Laroche, and E. Moulines. Regularized estimation of cepstrum envelope from discrete frequency points. In *Proc. IEEE Workshop on Applicat. of Signal Process. Audio Acoust. (WASPAA'95)*, pages 213–216, New Paltz, NY, USA, October 1995.

[5] J.W. Cooley and J.W. Tukey. An algorithm for the machine computation of complex fourier series. *Mathematics of Computation*, 19, 1965.

[6] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoustics, Speech and Signal Process.*, 28(4) :357–366, 1980.

[7] W. D'haes and X. Rodet. Discrete cepstrum coefficients as perceptual features. In *International Computer Music Conf. (ICMC'03)*, Singapore, 2003.

[8] S. Fenet, G. Richard, and Y. Grenier. A scalable audio fingerprint method with robustness to pitch-shifting. In *Int. Symposium on Music Information Retrieval (ISMIR'11)*, pages 121–126, October 2011.

[9] M. Frigo and S.G. Johnson. Fftw : An adaptive software architecture for the fft. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, volume 3, pages 1381–1384. IEEE, 1998.

[10] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *Int. Symposium on Music Information Retrieval (ISMIR'02)*, volume 2002, pages 107–115, Octobre 2002.

[11] J. Haitsma and T. Kalker. Speed-change resistant audio fingerprinting using auto-correlation. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'03)*, volume 4, pages IV–728, 2003.

[12] F.J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1) :51–83, Jan 1978.

[13] R. Mignot. BeeAlter Toolbox : Boîte à outils de dégradations sonores, pour tests. Technical report, 2015. Rapport interne IRCAM – CNRS, projet BeeMusic.

[14] R. Mignot. Evaluation de flux spectraux pour la sélection de temps d'ancrage robustes aux dégradations sonores. Technical report, 2015. Rapport interne IRCAM – CNRS, projet BeeMusic.

[15] R. Mignot. Ircam AudioID : Evaluation et résultats de la reconnaissance d'extraits audio dégradés. Technical report, 2015. Rapport interne IRCAM – CNRS, projet BeeMusic.

[16] R. Mignot. Ircam AudioPrint : Réduction de dimension des empreintes sonores par analyse discrimante. Technical report, 2015. Rapport interne IRCAM – CNRS, projet BeeMusic.

[17] R. Mignot. RAPPORT GLOBAL – Ircam AudioID : Indexation audio avec robustesse aux dégradations sonores. Technical report, 2015. Rapport interne IRCAM – CNRS, projet BeeMusic.

[18] R. Mignot and V. Välimäki. True discrete cepstrum : an accurate and smooth spectral envelope estimation for music processing. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'14)*, pages 7515–7519, Florence, Italy, May 2014.

[19] K. Moravec and I.J. Cox. A comparison of extended fingerprint hashing and locality sensitive hashing for binary audio fingerprints. In *Proc. of the 1st ACM Int. Conf. on Multimedia Retrieval (ICMR'11)*, page 31, April 2011.

[20] M. Ramona and G. Peeters. Audio identification based on spectral modeling of bark-bands energy and synchronization through onset detection. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'11)*, pages 477–480, 2011.

[21] M. Ramona and G. Peeters. Audioprint : An efficient audio fingerprint system based on a novel cost-less synchronization scheme. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'13)*, pages 818–822, May 2013.

[22] A. Röbel and X. Rodet. Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation. In *Proc. Int. Conf. on Digital Audio Effects (DAFx'05)*, pages 30–35, Madrid, Spain, September 2005.

[23] R. Sonnleitner and G. Widmer. Quad-based audio fingerprinting robust to time and frequency scaling. In *Proc. Int. Conf. on Digital Audio Effects (DAFx'14)*, September 2014.

[24] A. Wang. An industrial strength audio search algorithm. In *Int. Symposium on Music Information Retrieval (ISMIR'03)*, pages 7–13, October 2003.

[25] U. Zölzer. *DAFX : Digital Audio Effects.* Wiley, 2nd edition, 2011. 624 pages.