ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Représentation et modifications de scénes sonores stéréophoniques par localisation et séparation de sources

AUTOR : José Brígido Ruiz Carrillo
DIRECTOR : Rémi Gribonval
CODIRECTOR : Jean-Jacques Fusch

Septiembre / 2005

Autor	José Brígido Ruiz Carrillo
E-mail del Autor	ruizcarrillo@gmail.com
Director(es)	Remi Gribonval
E-mail del Director	
Codirector(es)	Jean-Jacques Fusch
Título del PFC	Représentation et modifications de scènes sonores stéréophoniques par localisation et séparation de sources
Descriptores	Best Basis, Wavelets, Local Cosines, BSS, Separation Sources, Stereo Representation, Meilleur basse, Ondelettes, cosinus locaux, Séparation de sources en aveugle, représentation stéréo, mejor base, ondículas, cosenos localizados, separación de fuentes, separación ciega de fuentes, representación estéreo

Resumen

En este *stage* se presentan las experiencias realizadas con el algoritmo B2S2 (*Best Basis Source Separation*). Con estas experiencias se ha probado la capacidad del algoritmo, basado en el enmascaramiento binario de las componentes de la señal en una base elegida por métodos de *mejor base*. Tras estas experiencias presentamos un camino para la separación ciega de fuentes con este algoritmo.

Para completar el trabajo, proponemos también un sistema de representación del sonido estéreo dentro del dominio transformado, utilizando un código de colores. El sistema se ha incluido en una aplicación *java* para el tratamiento de la señal estéreo.

Titulación	Ingeniería de Telecomunicación
Intensificación	Sistemas y Redes de Telecomunicación
Departamento	
Fecha de Presentación	Septiembre - 2005

abstract

In this paper we present our experiences with the B2S2 (Best Basis Source Separation) algorithm. We have tested the capacity of this algorithm which is based on a binary mask of the signal components obtained from a base selected using the best basis method. Following that, we have outlined an approach for blind source separation using the same algorithm.

To complete the work, we have proposed a stereo sound representation system in the transformation domain using a color code. This system has been integrated into a stereo sound processing java application.

résumé

Dans cette stage nous pressentons des expériences réalisées avec l'algorithme B2S2 (Best Basis Source Separation). Avec ces expériences nous avons testé la capacité de l'algorithme qui est basé en masquage binaire des composants des signaux dans un base choisi pour méthodes de meilleure base. Après les expériences, nous avons présenté un chemin pour faire la séparation en aveugle avec cet algorithme.

Pour compléter le travail nous avons proposé un système de représentation du son stéréo dans le domaine transformée en utilisant un codage de couleur. Le système a été mis dans une application java pour le traitement de la signal stéréo.

Table des matières

Introduction

Dans le cadre Erasmus, avec la collaboration entre l'Université de Rennes I et l'Universidad Politecnica de Cartagena, le stage a été effectué à l'IRISA au sein du projet METISS (Modélisation et Expérimentation pour le Traitement de l'Information et des Signaux Sonores). Les axes de recherche de l'équipe sont la reconnaissance du locuteur, l'indexation d'enregistrements audio et la séparation de sources. Parmi les membres, il faut mentionner le responsable du projet Frédéric BIMBOT et Rémi GRIBONVAL qui a encadré le stage.

Objectifs du stage

Le sujet de stage s'intitule «Représentation et modifications de scènes sonores stéréophoniques par localisation et séparation de sources». Les objectifs suivent principalement deux lignes directives.

Premièrement, dans le cadre de la séparation de sources, l'objectif est de trouver un algorithme de *clustering* pour la séparation de sources en aveugle en menant des expériences avec les algorithmes de séparation de meilleure base, spécialement avec le *B2S2 (Best Basis Separation Sources)*.

Deuxièmement, dans le cadre de la représentation de scènes sonores, l'objectif est de faire une représentation de son stéréo dans le domaine transformé qui nous donne le plus d'information possible en utilisant un codage de couleurs.

Ce qui a été fait dans ce stage

Sous la direction de mon maître de stage Rémi GRIBONVAL, et après l'étude nécessaire de documentation, le travail réalisé pour la séparation de sources a été :

- d'épurer l'algorithme *B2S2*, composé par des fichiers .m de Matlab,
- de faire l'étude des possibilités en ayant une partie des sources,
- de chercher des paramètres pour faire un algorithme de *clustering*, et
- d'essayer la séparation en aveugle.

Quant à la représentation stéréo, le travail a été :

- de chercher un codage de couleur adéquat pour la représentation de scènes sonores,
- de'implémenter sur Matlab une application pour la *représentation partielle* de son stéréo, et
- d'implémenter sur Java la même application.

Organisation du document

Le texte présenté ici est ainsi composé de deux parties.

La première partie du rapport parle de la séparation de sources. Il fait une introduction au problème en s'intéressant d'abord à savoir d'où vient la signal stéréo à étudier (l'enregistrement stéréo), puis à quelques notions de séparation de sources et enfin aux approches qu'on va utiliser. Ensuite nous allons présenter quelques outils pour la séparation dont les transformées de Fourier et ces variants, les décompositions en bases de paquets d'ondelettes ou cosinus localisées et l'algorithme de meilleure base. Pour finir on verra l'algorithme *B2S2*, les différentes expériences réalisées et ses résultats.

Dans la deuxième partie du texte, on présente des systèmes de représentation qui ont permis la construction d'un modèle, et le développement de l'application qui fait usage de ce modèle pour la représentation de signaux stéréo. Dans le chapitre de la représentation on parlera des représentations utilisées, pourquoi on utilise un système de couleurs et quels seront les systèmes résultants.

Première partie

Séparation de Sources Sonores

Chapitre 1

Introduction au Problème de la Séparation Sources

La séparation de sources c'est le processus d'extraction d'une information concrète à partir d'une série de capteurs. Ces capteurs transforment en signaux la série de phénomènes qui leur arrivent. Dans notre cas :

- les sources sont acoustiques,
- les phénomènes sont des variations de pression
- les capteurs sont les microphones.

Ces objets prendront d'autres formes pendant le cours du travail.

Par la suite nous nous interesserons à classifier le problème exposé en présentant toutes les variables qui interviennent. Le cadre de l'étude se place dans l'enregistrement stéréo.

1.1 L'enregistrement stéréo

Les signaux sur lesquels nous travaillons sont le résultat de l'enregistrement stéréo du son. Dans un enregistrement on peut distinguer deux éléments principaux, les sources de son et les microphones. En plus on a le milieu où ont lieu les phénomènes. La relation entre les caracteristiques de ces éléments produit différents signaux à la sortie.

1.1.1 Des microphones aux signaux stéréo

Les sources de son et les capteurs possèdent une directivité et une réponse fréquentielle. Cela veut dire que en fonction de la fréquence et de la position relative entre eux on aura des signaux différents.

Quand on parle d'enregistrement stéréo on pense toujours à des microphones cardioïdes. La raison vient des caractéristiques de la conjonction de deux microphones de ce type là.

Comme on peut voir sur la figure 1.1 la position d'une source S fait que dans un même microphone ait un gain différent selon l'angle.

$$g_{card}(\alpha) = \frac{1 + cos(\alpha)}{2},$$
$$S_{mic} = g_{card} \cdot I_{source}$$

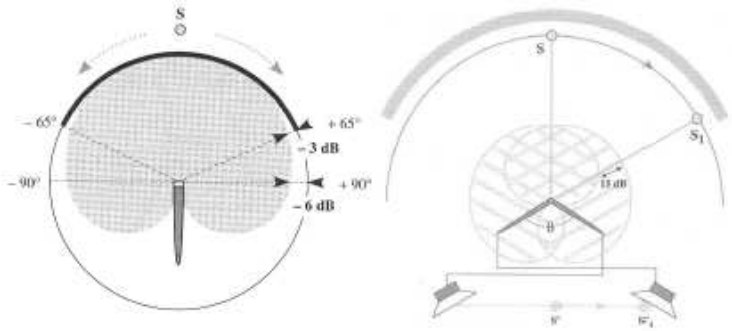

FIG. 1.1 – micros cardioides : directivité et placement pour l'enregistrement stéréo (figure extract de [Reulos,2003])

où g_{card} est un exemple d'une fonction cardioïde, I_{source} est une fonction de l'intensité de son de la source, et S_{mic} est le signal qu'on a après le micro et qui dépends de α et de I_{source}.

Si on a deux cardioïdes le gain du signal qui provient du microphone de gauche est toujours différent de celui qui vient de droite,

$$S_{Lmic}(\theta) = g_{card}(\alpha + \theta/2) \cdot I$$

$$S_{Rmic}(\theta) = g_{card}(\alpha - \theta/2) \cdot I$$

et comme il'est illustré sur la figure 1.1 on a l'effet stéréo qu'on veut.

Par rapport à la position des micros, il y a de nombreuses façon de les configurer. Mais surtout, soit on a les deux micros coincidents, soit ils sont distancts. Dans le premier cas le seul paramètre qui est différent pour les signaux c'est l'angle dans la directivité. Dans le deuxième cas on a aussi une différence de chemin qui va faire un sorte que les deux signaux ne soient pas en phase.

Le cas le plus simple pour voir le décalage de phase entre le signal de gauche et de droite est montré à la figure 1.2 où il y a deux micros omnidirectionels.

Le cas le plus général dans les techniques d'enregistrement est quand on a un décalage de phase et un gain dépendant de la position :

$$S_{Lmic}(\theta, t) = g_{card}(\alpha + \theta/2) \cdot I(t + \tau)$$

$$S_{Rmic}(\theta, t) = g_{card}(\alpha + \theta/2) \cdot I(t + \tau + \Delta t)$$

1.1.2 Le schema de l'enregistrement

Après avoir vu l'aspect le plus physique de l'enregistrement on va essayer de prendre un idée plus général par rapport à l'enregistrement.

À ce moment là on a les sources, le signal stéréo obtenu et le milieu. Le processus d'enregistrement prends les signaux acoustiques des sources et nous donne la signal stéréo.

$$sources \rightarrow salle, position, micros \rightarrow signal$$

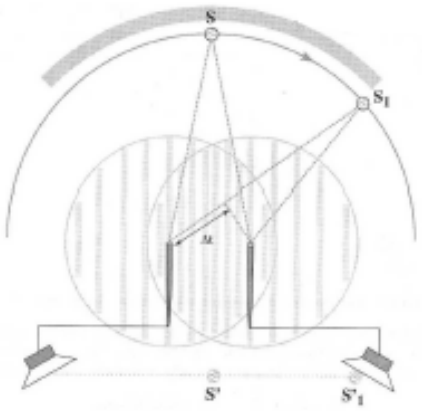

FIG. 1.2 – décalage de phase entre la gauche et la droite (figure extract de [Reulos,2003])

Pour nous abstraire en peu, on va imaginer un ligne d'un table de mixage. À l'entrée on a l'information qui vient d'un source. Cet information monocanal sera traite pendant tout la ligne et finalement elle ira a la sortie stéréo du *master*.

Avec plusieurs lignes, on aurait de différents traitements pour chacune et l'addition final du *master*. L'entrée serait multicanaux et la sortie toujours stéréo.

$$\begin{matrix} entre_1 \rightarrow gain_1, filtre_1, pan_1, father_1 & \\ \vdots & \rightarrow \textbf{Master} \\ entre_i \rightarrow gain_i, filtre_i, pan_i, father_i & \end{matrix}$$

Ce schéma est le même que ce qu'on a pour l'enregistrement :

$$\textbf{sources} \rightarrow \textit{enregistrement} \rightarrow \textbf{signal}$$

Comme dans la table de mixage, l'enregistrement fait des modifications sur les signaux acoustiques des sources, et aprés nous donne l'addition des signaux modifiés. Cet addition est ce qu'on a sur chaque canal de la signal de sortie.

$$\begin{matrix} source_1 \rightarrow f(salle_1, position_1, micros_1) & \\ \vdots & \rightarrow \begin{cases} signal_{gauche} \\ signal_{droite} \end{cases} \\ source_i \rightarrow f(salle_i, position_i, micros_i) & \end{matrix}$$

La signal qu'on enregistre à la fin est l'addition des segnaux des sources modifies pur un facteur a. Ce facteur est différent pour chaque source et pour chaque canal de sortie.

On peut écrir la relation aterieur mathematiquement :

$$x_i = \sum_{j=1}^{N} a_{ij} s_j$$

où x_i pour $i = 1, 2$ est la signal enregistre, N le nombre de sources, s_j les sources et a_{ij} est le facteur qui correspond à chaque source j et pour chaque canal i.

1.2 La separation de sources

La séparation de sources est un problème fondamental dans le domaine du traitement du signal et il apparaît dans une grande quantité d'applications comme trataiment d'antenne, télécommunications ou restauration de voix.

Avant d'aborder le probleme, on va faire quelques anotations que nous aideront à suivre le texto.

1.2.1 Quelques notations

Comme on a vu tout à l'heure on va travailler sur mono et multi-canaux.On parlera aussi de canaux que de signaux. Pour ça on va voir quelques simples notations pour suivre mieux le texte. Pour l'instant on serait en train de parler des signaux multi-canaux quand le texte soit écrit avec caractère gras et des signaux mono-canaux quand le texte soit normal, donc :

$$\mathbf{x}(\mathrm{t}) = (x_1(t), x_2(t), ..., x_L(t))^T$$

où $(\cdot)^T$ est la tranposée de la matrice.

D'autre part, on peut différencier aussi un signal estimé avec un chapeaux « ˆ ». Quand on a une matrice ou un signal on a par exemple $\mathbf{x}$, sinon on doit l'estimer et on aurait $\hat{\mathbf{x}}$ que serait $\mathbf{x}$ estimée.

Ces notations suivent le même systheme de notation que [Gribonval,2003].

1.2.2 le probleme

Le problème peut être formulé comme :

$$\mathbf{x(t)} = \mathbf{A} \cdot \mathbf{s}(\mathrm{t}) + \mathrm{W}$$

Où on suppose qu'on a un réseau de M capteurs qui fournait un vecteur de M observations $\mathbf{x}(\mathrm{t}) = [x_1(t), x_2(t), ..., x_M(t)]^T$ qui sont les résultats de la combinaison de N phénomènes $\mathbf{s}(\mathrm{t}) = [s_1(t), s_2(t), ..., s_N(t)]^T$ dénommé sources. En plus des sources et des observations, on a l'effet du milieu. Il nous fournit un bruit W au mélange, et la matrice de mélange $\mathbf{A}$, qui auraient des caractéristiques données. En général on peut écrire le mélange des sources moyennant une notation matricielle :

$$\mathbf{x}(\mathrm{t}) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_M(t) \end{pmatrix} = \mathbf{A} \begin{pmatrix} s_1(t) \\ \vdots \\ s_N(t) \end{pmatrix} + \begin{pmatrix} w_1(t) \\ \vdots \\ w_M(t) \end{pmatrix} = \mathbf{A} \cdot \mathbf{s}(\mathrm{t}) + \mathbf{w}(\mathrm{t})$$

Pour relier au section de l'enregistrement, on peut dire que $\mathbf{x}$ et $\mathbf{s}$ sont les signaux de sortie et entrée, $\mathbf{w}$ est un bruit qu'on n'a pas compte et $\mathbf{A}$ est la matrice de facteurs a_{ij} pour la signal i et la source j.

En fonction de l'information dont on dispose on a des différentes problèmes de séparation de sources. Dans une enregistrement multicanal les N sources sonores sont mélangées suivant les caractéristiques de position et gain selon la matrice de mélange $\mathbf{A}$ pour obtenir finalement les M canaux d'audio.

Dans un problème donné on peut avoir ou pas l'information sur comment les sources ont été mélangées. On parle de séparation de sources en aveugle quand

on n'a pas la matrice de mélange. Dans ce cas il faut faire une estimation de la matrice de mélange. Généralement on fait l'estimation part l'optimisation de fonctions non linéaires de certains paramètres des signaux observés.

En plus de la matrice de melange, suivant la relation entre le nombre de sources et des observations on distingue la séparation dans le cas non dégénéré (on a au moins autant de capteurs que de sources) ou les cas contraire, cas dégénéré. Quand on est dans le cas non dégénéré on peut trouver les sources en estimant l'inverse de la matrice de mélange (ou la pseudo-inversa) :

$$\mathbf{B} = \mathbf{A}^{\dagger} = (\mathbf{A^T} \cdot \mathbf{A})^{-1} \cdot \mathbf{A^T}$$

Cette matrice inverse (ou pseudo-inverse) est appliquée directement sur la matrice des signaux observées $\mathbf{x}$ pour obtenir directement les sources dans le cas dont il n'y aie pas de bruit : $\mathbf{s}(t) = \mathbf{B}\,\mathbf{x}(t)$.Par contre, quand on se trouve dans le cas dégénéré, la matrice de mélange $\mathbf{A}$ n'est pas inversible et il faut utiliser d'autres techniques plus complexes comme les différentes décompositions et masquages temps/fréquence.

On verra des différents sortes de décompositions dans le chapitre 2.

1.3 L'approche utilisé dans ce rapport

Le problème qui est présenté dans ce rapport de stage est la séparation de sources. Pour cela on va faire usage de quelques outils mathématiques qui sont exprimés dans le chapitre 2 et 3.

L'énoncé du problème est toujours le même. On a un signal $\mathbf{x}$ stéréo (ou multi-canaux en tout cas) qui vient du mélange de N sources s_i. Les signaux sont échantillonnés et on suppose un bruit $\mathbf{w} = 0$. Donc on a

$$\mathbf{x}(t) = \mathbf{A} \begin{pmatrix} s_1(t) \\ \vdots \\ s_N(t) \end{pmatrix} = \mathbf{A} \cdot \mathbf{s}(t).$$

L'objectif est de récupérer les N sources s_i à partir du signal stéréo $\mathbf{x}$. Le nombre des sources N dans notre problème est toujours plus grand que celui des canaux ($M = 2$), donc on est dans le cas dégénéré. Selon les cas on disposera ou non de la matrice de mélange $\mathbf{A}$.

Même si on ne dispose pas de la matrice $\mathbf{A}$, le mode de séparation est toujours le même :

1. décomposition du signal dans un domaine adéquat
2. traitement par masquage
3. reconstruction des signaux traités

Le point 1 fait référence à la décomposition des signaux dans un domaine transformée. C'est dans un domaine temps-fréquence qu'on va traiter nos signaux. Les outiles pour cette transformation sont la transformée de Fourier et les différentes décompositions sur les bases de paquets d'ondeletes ou de cosinus localisées, qui seront mentionnées après. L'idée qui doit être claire déjà est qu'on choisi **une base** B pour représenter nos signaux x_1 et x_2 qui est la plus **adéquate** pour elles, et ensuite on peut travailler avec les composantes de nos signaux sur la base choisie :

si on a $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ et on choisi B, on aurait $B^T\mathbf{x}$ (les composants de $\mathbf{x}$ sur B).

Le traitement par masquage est centré au processus de séparation. On va partir de l'hipothèse de qu'il n'y a qu'une source active dans chaque morceaux temps-fréquence k. Avec les composants $B^T\mathbf{x}$ on fait une **masque binaire** $mask$ qui dit quelle est la source active sur chacune.

Alors pour un échantillon k la source $\mathbf{s}[k]$ peut être estmée en fonction de $masc$ puisque :

$$masc_j[k] = \begin{cases} 1 & \text{si on dis que} s_j \text{est activa} \\ 0 & \text{sinon} \end{cases}$$

$$s[k] = \begin{pmatrix} s_1[k] \\ \vdots \\ s_N[k] \end{pmatrix}, s_j[k] = masc_j[k] \cdot a_j^{\dagger}\mathbf{x}[k]$$

où a_i est la colonne i d'$\mathbf{A}$ qui peut être connue ou doit être estimée ($\widehat{\mathbf{A}}$). Pour avoir des équations plus lisibles, $\mathbf{x}$ et s_j font référence au $B^T\mathbf{x}$ et $B^T s_i$ respectivement puis on est dans le domaine transformé.

Après le processus de séparation on fait la reconstruction des signaux qui ont été estimé avec une masque binaire et dans le domaine d'une base adéquate.

Chapitre 2

Le domaine transformée

Dans le chapitre nous présentons les principaux outiles de représentation de signal que nous sevrot pour la séparation de sources : la transformé de Fourier et ses derivées,

2.1 La Transformée de Fourier

La transformée de Fourier décompose une fonction en sinusoïdes de différentes fréquencee. Elle identifie ou distingue les sinusoïdes des différents fréquences et leurs respectives amplitudes.

La transformée de Fourier peut être définie pour des signaux $x(t)$ que peuvent être : :

1. discrèts ou continus dans le temps,
2. et de durée finie ou infinie.

Dans le premièr cas la transformée est définie comme :

$$X(w) = \int_{-\infty}^{\infty} x(t)e^{-iwt}dt$$

où le temps t est continu, son domaine est infini et $w \in (-\infty, \infty)$. Ici w est la fréquence continue en radian (rad/sec). En plus f dénote la fréquence continue en Hertz, où $w = 2\ \pi\ f$. On a supposé que $x(t)$ est dans l'espace de Hilbert $L_2(\mathbb{R})$.La transformée de Fourier inverse est donnée par :

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(w)e^{iwt}dw$$

Ppour plus d'information sur les différentes définitions et les propriétés de la transformée de Fourier voir [Mallat,1999, Chap. 8] ou [Chui,1992, vol.2, Chap. 3].

2.2 Les transformées de Fourier discrètes

Jusqu'ici, on a parlé du traitement du signal d'une façon «théorique» valable pour les siganux analogiques. Mais pour ne perdre pas le fil de c'est qu'il faut traiter vraiment, des signaux numériques, on va introduire «l'autre théorie» du traitement de signal.

On va plonger dans le domaine du traitement numérique du signal.

2.2.1 DFT

Puisqu'un ordinateur fonctionne seulement avec des données discrètes, le calcul numérique de la transformée de Fourier requières des échantillon de $x(t)$, que nous appellerons x_k. En outre, un ordinateur peut calculer la transformation $X(s)$ seulement aux valeurs discrètes de s, c.-à-d., il peut seulement fournir les échantillons discrets de la transformation, X_r. Si N_0 est le nombre d'échantillons dans le signal dans un période T_0, la transformée discrète de Fourier (DFT) est définie comme

$$X_r = \sum_{k=0}^{N_0-1} x_k e^{-ir\Omega_0 k}$$

où $\Omega_0 = \frac{2\pi}{N_0}$. Son inverse est

$$x_k = \frac{1}{N_0} \sum_{r=0}^{N_0-1} X_r e^{ir\Omega_0 k}$$

Ces équations peuvent être utilisés pour calculer les transformées et les transformées inverses des donnés adéquatement échantillonnés.

2.2.2 FFT

La *Transformé de Fourier Rapide (FFT)* est un algorithme de DFT développé par Tukey et Cooley en 1965 qui ramène le nombre des calculs de l'ordre de N_0^2 à $N_0 log(N_0)$. Il y a fondamentalement deux types d'algorithmes de Tukey-Cooley FFT en service : décimation-en-temps et décimation-en-fréquence. L'algorithme est simplifié si N_0 est choisi pour être une puissance de 2, mais ce n'est pas un requête.

Pour avoir plus d'information de la DFT et de la FFT on peut voir [Mallat,1999, pp. 20–21].

2.2.3 STFT

Dans le traitement du signal, la transformée de Fourier à court terme (*short time Fourier transform* : STFT) joue un rôle fondamental, en particulier dans le domaine de l'analyse du son articulé. Elle est également employée pour représenter, dans le domaine fréquentiel, les propriétés variants dans le temps de la forme d'onde de la parole.

La STFT de $x(n)$ est un ensemble de DFT qui correspondent au différents sections du temps de $x(n)$. La section de temps pour le temps n_0 est obtenue par multiplication de $x(n)$ avec un séquence décalée $w(n_0 - n)$. Cette section du temps n_0 correspond au nombre d'échantillons (N_0 dans la section 2.2.1) du signal qu'on avait pour le cas de la DFT.

Le fait de pouvoir faire la transformée de Fourier de petits morceaux du signal donne l'opportunité de faire la représentation temps-fréquence qu'on appel **spectrogramme**. On a des exemples de spectrogrammes dans la figure 2.1.

Une définition utile de la transformée de Fourier à cour terme est :

$$X(n_0, k) = \sum_{m=-\infty}^{\infty} x(m) w(n_0 - m) e^{-i\frac{2\pi k m}{N}} R_N(k)$$

où N est le facteur de la fréquence d'échantillonnage, $R_N(k)$ est une séquence rectangulaire de N-points utilisée pour isoler la portion du signal d'entrée qui serait analysé. Et $w(n_0 - m)$ est une fenêtre (ou un filtre) qui joue aussi un rôle très important dans l'analyse du signal.

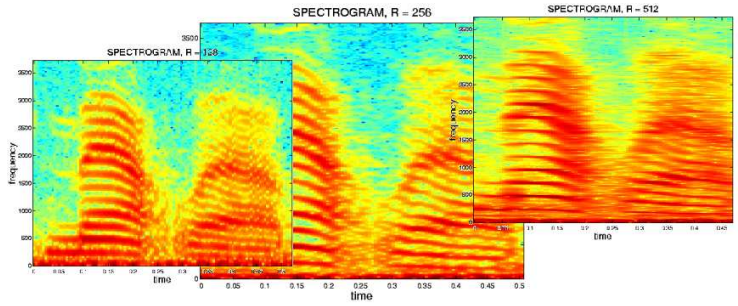

FIG. 2.1 – Le spectre de la droite a la plus longue fenêtre et la résolution fréquentiel le plus grande. Au gauche, le spectre qui a le plus petit longueur, il a la résolution temporel la plus grande.

Pendant l'analyse du son articulé, la forme et la longueur de la fenêtre affecteront la représentation de fréquence de la parole (ou d'autre signal).Un exemple de l'effet de la variation des longueurs de la fenêtre est montrée dans la figure 2.1. Des divers types de fenêtres ont été étudiés en produisant des caractéristiques de formes de fenêtre appropriées pour différentes applications. Certaines de ces fenêtres sont montrées dans la figure 2.2.

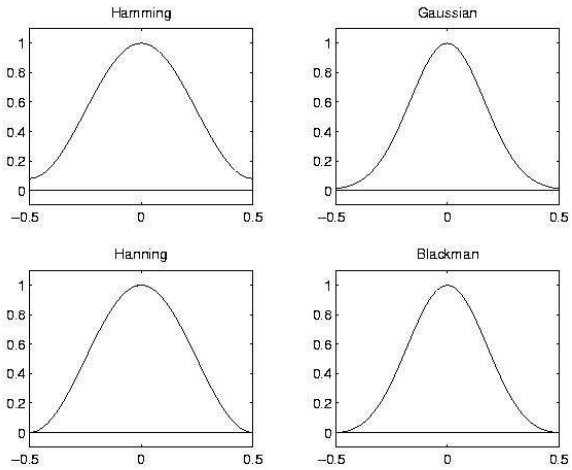

FIG. 2.2 – Exemples des fenêtres utilisées dans la pratique

Pour le choix de la fenêtre adéquate pour différents applications de traitement de signal voir [Mallat,1999, pp. 75–77]. Une extension de ce qu'on a introduit ici on peut le trouver dans [Lim & Oppenheim,1988, pp. 289–337]

2.3 *En 1946 le physicien Gabor ...*

Le Principe d'incertitude déclare que la diffusion d'énergie d'une fonction et de sa transformée de Fourier ne peut pas être simultanément arbitrairement petit (extrait de l'introduction de [Mallat,1999]).

Pour cette raison notre héros définissait les atomes temps-fréquence élémentaires comme formes d'ondes qui ont une diffusion minimale dans un plan temps-fréquence. Pour mesurer le contenu d'*information* temps-fréquence, il proposait de décomposer des signaux en ces formes d'onde atomiques élémentaires.

Les atomes de Gabor sont construits par déplacement en temps et en fréquence d'une fenêtre g :

$$g_{u,\xi}(t) = g(t-u)e^{i\xi t}$$

L'énergie de $g_{u,\xi}$ est concentrée dans le voisinage de u pendant un intervalle de la taille σ_t, mesuré par la déviation standard de $|g|^2$. Sa transformée de Fourier est le déplacement par ξ de la transformée de Fourier $\widehat{g}$ de g :

$$\widehat{g}_{u,\xi}(\omega) = g(\omega-\xi)e^{-iu(\omega-\xi)}$$

L'énergie de $\widehat{g}_{u,\xi}$ est donc localisée près de la fréquence ξ, sur un intervalle de la taille σ_ω, qui mesure le domaine où $\widehat{g}(\omega)$ est non-négligeable. Dans un plane (t,ω) temps-fréquence, la diffusion d'énergie de l'atome $g_{u,\xi}$ est symboliquement représentée par le rectangle de Heisenberg illustré par la figure 2.3. Ce rectangle est centré en (u,ξ) et il a une largeur σ_t de temps et une largeur σ_ω de fréquence.

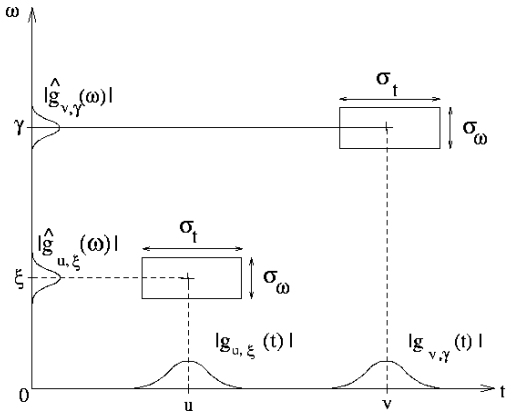

FIG. 2.3 – Rectangles de Heisenberg en représentant l'énergie de dispersion de deux atomes de Gabor (figure 1.1 de [Mallat,1999])

2.4 Les Ondelettes

Les ondelettes permettent de décomposer un signal dans ses différents composants de fréquences et d'étudier chacun de ces composants avec une résolution adéquate à son échelle. Ils ont été développés par des géologues, des ingénieurs intéressés à la vision par ordinateur et le traitement de l'image et par mathématiciens (extrait du résumé de [Hernández,2003]).

On a un introduction aux ondelettes à [Chui,1992, vol.1] et pas mauvais tutoriels dans [Chui,1992, vol.2] et [Mallat,1999].

Une ondelette est une fonction ψ avec une moyenne zéro :

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0$$

qui est dilatée avec un paramètre s, et déplacée par u.

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right).$$

La transformée d'ondelette de f à l'échelle s et la position u est calculée en corrélant f avec un atome d'ondelette

$$Wf(u,s) = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{s}}\psi^*\left(\frac{t-u}{s}\right)dt$$

équation qui peut être aussi écrite comme une intégration en fréquence (en appliquant Parseval)

$$Wf(u,s) = \int_{-\infty}^{+\infty} f(t)\psi^*_{u,s}(t)dt = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \widehat{f}(\omega)\widehat{\psi}^*_{u,s}(\omega)d\omega.$$

Le coefficient de l'ondelette $Wf(u,s)$ dépend ainsi des valeurs $f(t)$ et $\widehat{f}(\omega)$ dans la région de temps-fréquence où l'énergie de $\psi_{u,s}$ $\widehat{\psi}_{u,s}$ est concentré. Des variations harmoniques de temps sont détectées de la position et échelle des coefficients d'ondelettes avec les plus grandes amplitudes.

À temps, $\psi_{u,s}$ est centré à u avec une diffusion proportionnelle à s. Sa transformée de Fourier est calculée a partir de l'expression de $\psi_{u,s}(t)$:

$$\widehat{\psi}_{u,s}(\omega) = e^{-iu\omega}\sqrt{s}\widehat{\psi}(s\omega),$$

où $\widehat{\psi}$ est la transformée de Fourier de ψ.

Dans le plan temps-fréquence, un atome d'ondelette $\psi_{u,s}$ est symboliquement représentée par un rectangle centré en $(u, \eta/s)$. La diffusion temporelle et fréquentiel sont proportionnelles respectivement à s et à $1/s$. Quand s change, la taille et la largeur du rectangle change mais sa surface reste constante. On peut voir ça dans la figure 2.4.

2.5 Bases d'atomes

Dans cette section on ne va pas développer la construction de bases orthonormales de $\mathbf{L}^2(R)$ composées d'atomes temps-fréquence localisées. Pour avoir les détaills des plusieurs types de bases et leurs constructions on peut voir [Mallat,1999, pp. 220–375].

Par contre on va présenter les deux bases qui seront utilisées tout au long du travail : Les paquets d'ondelettes et les bases de cosinus localisés.

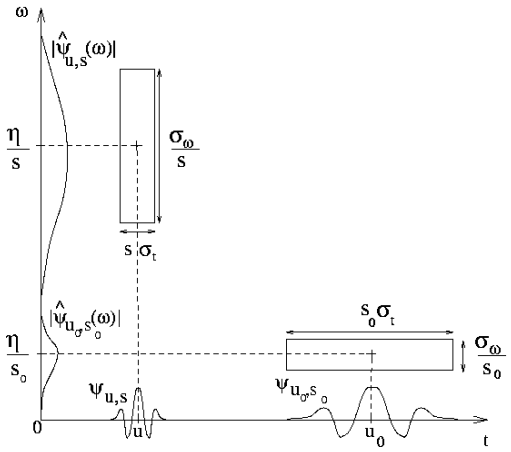

FIG. 2.4 – Boîtes temp-fréquence de deux ondelettes $\psi_{u,s}$ et ψ_{u_0,s_0} (figure 1.2 de [Mallat,1999])

2.5.1 Bases de paquets d'ondelettes

Une base orthonormale d'ondelettes décompose l'axe des fréquence en intervalles dyadiques dont les tailles ont une croissance exponentielle, comme il est montré par la figure 2.5. On va généraliser cette construction dyadique fixe en décomposant la fréquenceen intevalles dont les longueurs de bande peuvent changer. Chaque intervalle de fréquence est couvert par les boîtes temps-fréquence des fonctions du paquet d'ondelettes qui sont uniformément translatées en temps pour couvrir le plan entier comme est montré sur la figure 2.6.

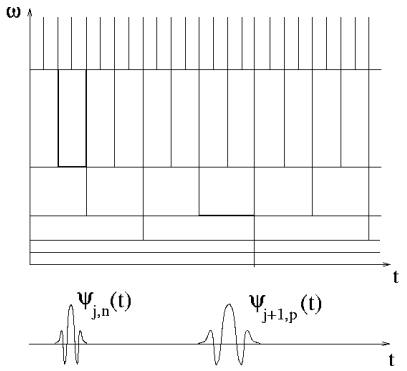

FIG. 2.5 – Les boîtes t/f d'un base d'ondelettes definissent un pavage du plan t/f (figure 1.3 de [Mallat,1999])

2.5.2 Bases de cosinus localisées

Des bases orthonormales de $\mathbf{L}^2(R)$ peuvent également être construites en divisant l'axe de temps au lieu de l'axe de fréquence. L'axe de temps est segmenté

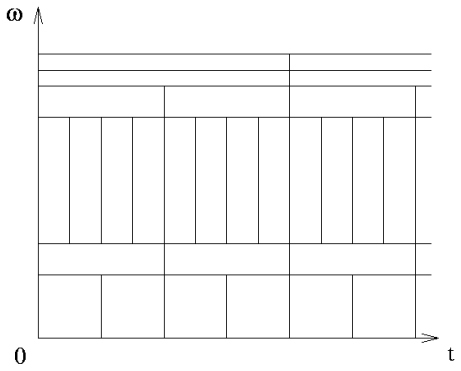

FIG. 2.6 – Les boîtes t/f d'un base de paquets d'ondelettes definissent un pavage du plan t/f (figure 1.4 de [Mallat,1999])

en intervalles finis $[a_p, a_{p+1}]$.

Les bases locales de cosinus sont obtenues en concevant des fenêtres lisses $g_p(t)$ qui ouvrent chaque intervalle $[a_p, a_{p+1}]$, et en les multipliant par des fonctions $\cos(\xi t + \phi)$ à différentes fréquences.

Une multiplication pur $\cos(\xi t + \phi)$ déplace la transformée deFourier $\widehat{g}_p(\omega)$ de $g_p(t)$ pur $\pm\xi$. Au-dessus des fréquences positives, la boîte temps-fréquence de la fenêtre modulée $g_p(t)\cos(\xi t + \phi)$ est donc égale à la boîte temps-fréquence de g_p déplacée de ξ le long des fréquences. Les boîtes temps-fréquence de vecteurs d'un base de cosinus locaux définissent un pavage du plan temps-fréquence illustré par la figure 2.7.

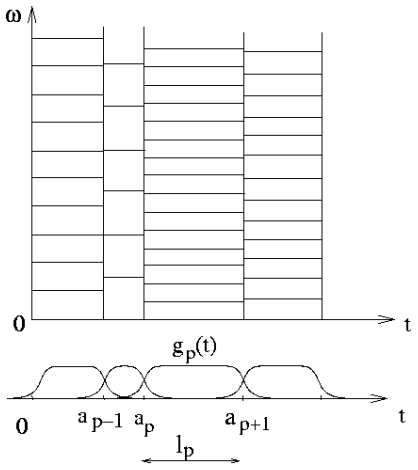

FIG. 2.7 – Les boîtes t/f d'un base d'ondelettes define un carrelage du plan t/f(figure 1.5 de [Mallat,1999])

Chapitre 3

Algorithmes de meilleure base

Selon [labTNSI] au huitième chapitre du [restauration], l'algorithme Best Basis permet de sélectionner la meilleure base pour représenter un signal donné dans un dictionnaire aussi pré-défini, en accord avec un critérie d'optimisation.

Nous pouvons appliquer en particulier le méthode à deux bibliothèques de création récente, *orthogonal wavelet-packets* et les *localized trigonometric functions*, qui ont des propriétés de localisation de l'ensemble temps-fréquence des formes d'onde raisonnablement bien contrôlées. L'idée est d'établir à partir des fonctions d'une bibliothèque une base orthonormale dans laquelle le signal ou la collection des signaux a le plus bas coût d'information.

Chaque bibliothèque de formes d'onde a une structure d'arbre binaire qui tends possible des algorithmes de recherche en $O(N \log N)$ pour obtenir la meilleuree base. En plus chaque forme d'onde est indexée par les paramètres de l'arbre qui ont une interprétation en termes de position, fréquence et échelle.

3.1 Bibliothèques de paquets d'ondelettes et bases de cosinus localisées

Ici on va introduire quelque concepts d'un *bibliothèque de bases orthonormales* sans aller dans trop de détails. On va noter quelles sont les bases utilisées dans l'algorithme et jeter un coup d'oeil à quelques unes de leurs propriétés. On peut voir les explicatins en détail dans [Coifman & Wickerhauser] d'où ce chapitre est basée, et dans [Mallat,1999].

Le deux formes d'onde qui seront utilisées pour la construction des bibliothèque sont les paquets d'ondelettes ou *Wavelets Packets* et les bases de fonctions trigonométriques localisées ou *local trigonometric basis*.

Les bibliothèques des fonctions trigonométriques localisées correspondent à des transformées de sinus localisées associés à un enveloppe par intervalles de $\mathbf{R}$ (section 2.5.2).

Une fonction de sinus localisée a la forme de la figure 3.1.

On parle de sinus localisée parce que dans l'expression de la base du sinus localisée, qui a la forme suivant :

$$S_{i,j}(t) = \sqrt{\frac{2}{|I_i|}} p_i(t) \sin\left[\pi\left(k+\frac{1}{2}\right)\frac{t-c_i}{|I_i|}\right].$$

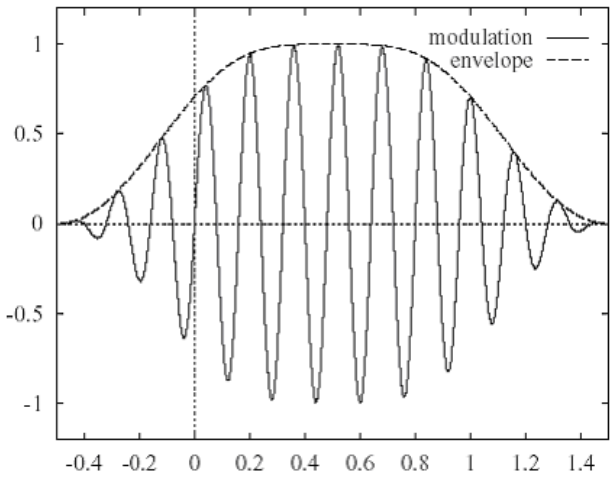

FIG. 3.1 – Example d'un fonction de sinus localisée sur l'intervalle [0,1] (figure extrait de [Coifman & Wickerhauser])

on a un sinus. On peut changer ici le sinus pour un cosinus et on aurait une **base de cosinus localisées**. C'est pour ça qu'on parle de *local trigonometric basis*.

Les index de chaque fonction $S_{i,j}$ indiquent la «position» et la «fréquence» où elle est localisée. Par example, i indique que la fonction es dans l'intervalle $[a_{i-1}, a_i]$, donc la fonction de la figure 3.1 qui est mise dans l'intervalle [0, 1] serait dénotée $S_{1,j}$.

La collection $S_{i,j} : k \in N$ forme une base orthonormale pour un sous-espace de $L^2(\mathbf{R})$ qui se compose de fonctions continues soutenues dedans un intervalle $[a_{i-1}, a_{i+1}]$.

Si on prend deux sous-espaces, leur somme directe nous donne un nouveau sous-espace comme illustré la figure 3.2.

La relation entre les plusieurs sous-espaces longs et les deux petits peut être vue comment la construction d'une base orthonormale pour n'importe quelle partition de $\mathbf{R}$ qui a $\{a_i\}$ comme un raffinement.

Le graphique de l'ordre partiel peut être fait avec un arbre, et cet arbre la peut être cherché efficacement pour trouver la «**meilleure base**».

La bibliothèque de paquets d'ondelettes peut être aussi construite avec la même idée. Pour suivre la construction voir [Coifman & Wickerhauser].

Il faut noter que, comme le *local cosine basis library*, la bibliothèque des bases de paquets d'ondelettes peut être organisée aussi dans un arbre. Celle-ci permet de trouver la meilleur base.

3.2 La meilleure base

Si on fit un vecteur $x \in \mathbf{R}^N$, on peut transformer une fonction de coût sur les multiples des bases orthonormales, cet à dire, le groupe $\mathbf{O}(N)$. Soit $B \in \mathbf{O}(N)$

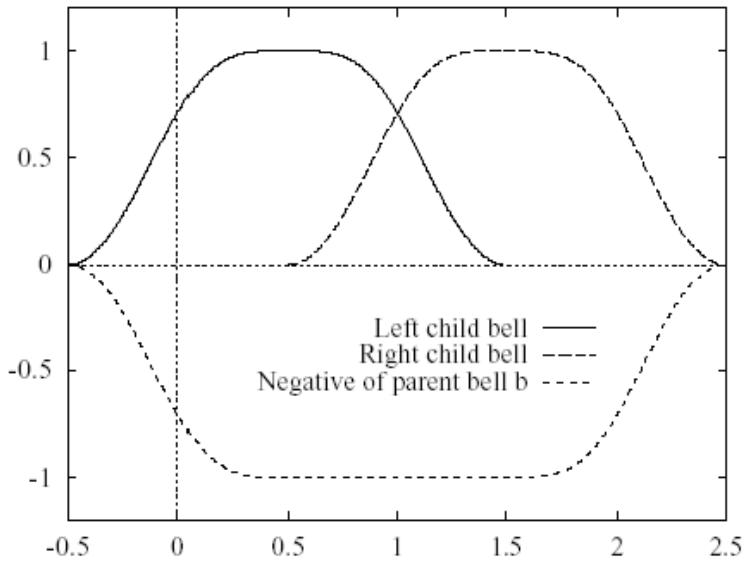

FIG. 3.2 – Le sous-espace long est la somme directe des 2 petits sous-espaces (figure extrait de [Coifman & Wickerhauser])

une base orthonormale, écrite comme matrice des vecteurs de ligne :

$$\textit{le vecteur } x, \qquad x \in \mathbf{R}^N$$

$$\textit{et la base } B, \qquad B \in \mathbf{O}(N)$$

Alors $B^T \cdot x$ est le vecteur de coefficients de x dans la base orthonormale B. On définit aussi $\mathcal{C}(B^T x)$ comme une fonction de coût additif de l'entropie de x dans la base B $(B^T x)$, (voir [restauration, Chap. 8])

$$\textit{le vecteur de coefficients de } x \textit{ dans } B, \qquad B^T \cdot x,$$

$$\textit{et le coût d'information de } x \textit{ dans } B, \qquad \mathcal{C}(B^T \cdot x)$$

On va prendre la bibliothèque $\mathcal{B} \subset bfO(N)$ des bases orthonormales où chacune d'elles a une transformée rapide (d'ordre $O(N \log N)$ ou meilleur) associée. Pour ces bases la recherche d'un minimum obligé de $\mathcal{C}$ converge en$O(N)$ opérations.

On va définir **la meilleure base** relative à $\mathcal{C}$ pour un vecteur x dans un bibliothèque $\mathcal{B}$ de bases comme la base $B_x \in \mathcal{B}$ pour laquelle $\mathcal{C}(B_x^T \cdot x)$ est minimum :

$$\textit{la meilleur base } B_x, \qquad B_x = arg \min_{B \in \mathcal{B}} \mathcal{C}(B^T \cdot x),$$

$$\textit{et le vecteur de coefficients de } x \textit{ dans } B_x, \qquad B_x^T \cdot x.$$

3.3 Critères pour la Recherche de la Meilleur Base

On va commencer par le calcul de l'entropie d'une expansion relative aux intervalles de longueur un, puis on compare l'entropie de chaque couple adjacent d'intervalles à l'entropie d'une expansion sur leur union. On sélectionne l'expansion avec le moins d'entropie et on continue jusqu'à une certaine taille interne maximum.

Le structure de l'arbre est représentée sur la figure 3.3.

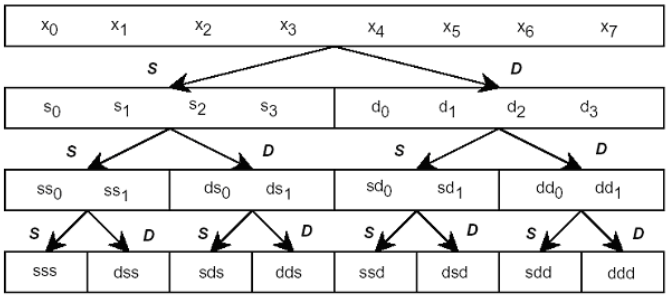

FIG. 3.3 – Paquets d'ondelettes (*WP*) organisées comme un arbre binaire (figure extrait de [Coifman & Wickerhauser])

Chaque noeud représente un sous-espace du signal original. Chaque sous-espace est la somme directe orthogonal de touts ses deux petits noeuds (comme dans la figure 3.2). Les feuilles de chaque sous-arbre nous donnent une base orthonormal. Dans la figure 3.4 on a représenté un exemple de base.

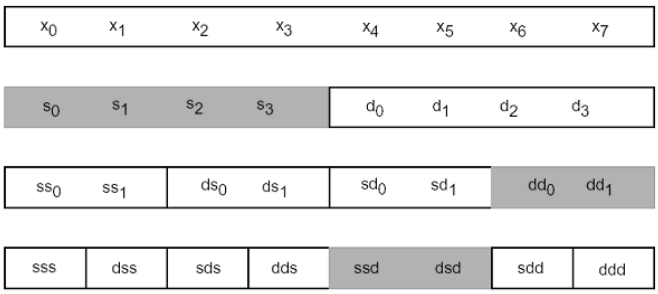

FIG. 3.4 – Une partie de la bibliothèques de bases de *WP* : la base d'ondelettes. La base est formée pour les quatre sous-espaces représentés pour les boîtes en gris (figure extrait de [Coifman & Wickerhauser])

La bibliothèque de *local trigonometric bases* sur un intervalle compact U peut être organisée comme un arbre binaire en prenant des portions localisées à une décomposition dyadique de U. Alors I_{00} corresponde à la base du sinus sur U, et I_{nk} corresponde à la base du sinus localisée dans l'intervalle n des 2^k intervalles

au niveau k de l'arbre. Cette organisation es représenté schématiquement à la figure 3.5.

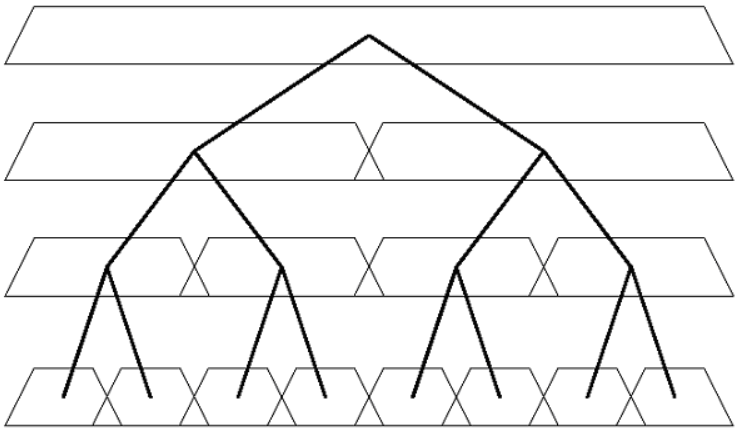

FIG. 3.5 – Organisation des intervalles localisées dans un arbre binaire (figure extrait de [Coifman & Wickerhauser])

Pour finir avec le chapitre on va voir un exemple de segmentation.

Cette procédure permet la segmentation des signaux acoustiques avec des fenêtres dyadiques adaptées au contenu local de fréquence. L'exemple de la figure 3.6 est la segmentation d'une patie du mot «armadillo».

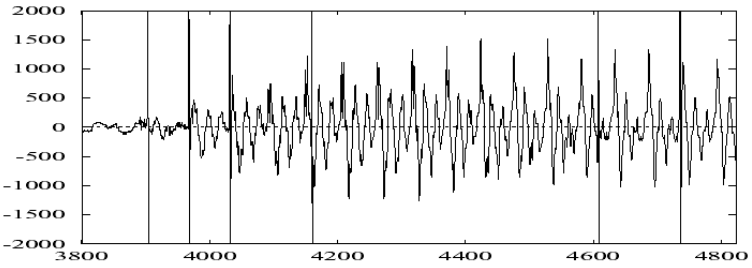

FIG. 3.6 – Segmentation automatique pour la plus petite entropie d'une partie du mot «armadillo» (figure extrait de [Coifman & Wickerhauser])

Chapitre 4

Vers B3S2 (Best Basis Blind Source Separation)

Cette part du travail et ce chapitre en particulier est orienté a l'obtention d'un algorithme B3S2, Best Basis Blind Source Separation, c'est à dire, un algorithme de séparation de sources en aveugle en utilisant les algorithmes Best Basis.

Pour commencer on va utiliser un algorithme qui vient directement de mon maître de stage Rémi Gribonval et qui s'appelle B2S2 (Best Basis Source Separation). En partant de cet algorithme on fera des expériences avec Matlab pour trouver les pistes qui nous emmènent jusqu'à la séparation en aveugle.

On verra les expériences réalisées et les résultats obtenus. Après on verra quels sont lespistes à explorer à partir de ces expériences et ces résultats .

4.1 Les principes

Pour introduire l'algorithme B2S2 on peut réviser l'exposé de notre problème (section 1.3, page 17). À partir de là on va essayer de développer l'algorithme et ses variantes.

D'abord, on va faire un résumé de la section 1.3.

Dans notre problème de séparation de sources on a un signal stéréo $\mathbf{x}$ et la matrice de mélange $\mathbf{A}$ dans l'équation :

$$\mathbf{x} = \mathbf{A} \cdot \mathbf{s},$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [a_1, \ldots, a_N] \cdot \begin{bmatrix} s_1 \\ \vdots \\ s_N \end{bmatrix}$$

où on part déjà de l'hipothése d'un bruit $\mathbf{w} = 0$.

Comme le nombre de sources N n'est pas deux ($N > M = 2$ canaux) on ne peut pas trouver l'inverse $\mathbf{A}^\dagger$ de $\mathbf{A}$ qui ferait :

$$\mathbf{s} = \mathbf{A}^\dagger \cdot \mathbf{x},$$

donc pour venir à bout de ce problème supposons qu'à chaque instant on a un seule source active, et toutes les autre sont nulles. En conséquence la source

active à un instant k est :

$$s_i[k] = a_i^\dagger \cdot \mathbf{x}[k].$$

Les trois sections où se divise l'algorithme sont :

1. décomposition,
2. masquage et
3. reconstruction

où jouent le rôle principal la base adéquate et la masque binaire.

La base est choisi d'un bibliothèque de bases, pour avoir le domaine de représentation le plus adéquate. Le choix est fait par l'algorithme *BestBasis* (chapitre 3).

La masque binaire par contre fait partie de l'algorithme B2S2 et consiste en faire un matrice binaire de la taille de $\mathbf{s}$ où la position de la source active dans un instant k est à 1 et toutes les autres a 0. Pour exemple si on a 3 sources et dans l'instant k on estime que la troisième source s_3 est active, on aura le masque $masc$:

$$s[k] = \begin{pmatrix} s_1[k] \\ s_2[k] \\ \mathbf{s_3[k]} \end{pmatrix}, \qquad masc[k] = \begin{pmatrix} 0 \\ 0 \\ \mathbf{1} \end{pmatrix}$$

Avec cette masque on peut faire la reconstruction des sources $s_i[k]$ puisque pour un instant k donné si la source active est $s_i[k]$, la source estimée $\widehat{s}_j[k]$ est

$$\widehat{s}_j[k] = \begin{cases} a_i^\dagger \mathbf{x}[k] & si \quad j = i \\ 0 & si \quad j \neq i \end{cases}, \qquad j = 1, \ldots, N$$

$$\widehat{s}_j[k] = masc_j[k] \cdot a_i^\dagger \mathbf{x}[k]$$

où $a_j^\dagger$ est la transposé de la psedo-inversa de la colonne a_j de la matrice de mélange $\mathbf{A}$.

Ce masquage est fait dans le domaine de la meilleure base. Pour cette raison on doit noter que l'idée d'avoir une seule source active dans chaque instant du temps devient en peut plus précise. Maintenant l'idée est que dans chaque morceaux temps-fréquence on a une seule source active. Avec ce principe l'algorithme de masquage devient plus complet puisque il est plus vraisembable que les sources qui sont dans le même morceaux de temps seront probablement dans différentes fréquence.

Le problème qu'on a maintenant est décider quelle est la source active pour chaque composante. L'idée la plus intuitive peut être de choisir la source le plus énergétique, mais on doit avoir $\mathbf{s}$ pour faire ça. Comme solution à ce problème on peut faire une premiere estimation $\widehat{\mathbf{s}'}$ de $\mathbf{s}$ et voir quelle est la source le plus énergétique de l'estimation :

$$\widehat{\mathbf{s}'} = \mathbf{A}^\dagger \cdot \mathbf{x}, \qquad \widehat{masc}_j[k] = \begin{cases} 1 & si \quad \widehat{s'}_j[k] = \max_{(j)}(\widehat{s'}_j[k]) \\ 0 & si \quad \widehat{s'}_j[k] \neq \max_{(j)}(\widehat{s'}_j[k]) \end{cases}, \quad j = 1, \ldots, N$$

où $\widehat{masc}_i$ est la masque qui vient de l'estimation des sources $\widehat{s'}$.

Comme pour estimer la source la plus énérgetique on ne doit pas estimer vraiment les sources, les erreurs de cette première estimation des sources $\widehat{s'}$ ne sont pas un grand problème.

4.2 B2S2 (Best Basis Source Separation)

B2S2 c'est un algorithme pour la séparation de sources basé sur [Gribonval,2003].

Cet algorithme de séparation a été développé sur Matlab et divisée en plusieurs fichiers : :

- *B2S2.m*
- MBestBasisAnalysis.m
- BinaryMasks.m
- MaskDecomp.m
- MBestBasisSynthesis.m
- SeparationMatrices.m

En plus, ces fichiers font utilisation de la bibliothèque WaveLab. Cette bibliothèque c'est un collection de fonctions pour Matlab disponible sur Internet [WaveLab] où on peut trouver aussi tout la documentation.

L'algorithme prend un signal multicanaux $\mathbf{x}$, préférablement d'une longueur puissance de deux, et la matrice de mélange $\mathbf{A}$. On peut aussi mettre certains paramètres, comme le choix entre l'utilisation de Wavelets Packets ou Local-Cosinus pour la décomposition ; le choix de la fonction de coût pour trouver la meilleur base ; ou le choix pour avoir dans la sortie les sources estimées monocanal ou les images multicanaux des sources. Dans l'annexe A on trouvera les en-têtes des fichiers *X.m* du paquet.

En retournant à notre paquet, B2S2.m c'est le fichier qui contient vraiment l'algorithme. Les autres fichiers font les différentes fonctions des trois parties de la *Best Basis Source Separation* qu'on va détailler par la suite.

La décomposition des signaux se fait à partir des algorithmes Best Basis. Dans les fichier *MBestBasisAnalysis.m* on a l'option d'utiliser paquete d'ondelettes ou cosinus localisées. . En fonction des paquets choisis et de la fonction de coût $\mathcal{C}$ pour trouver la meilleur base B_x, on obtiens des coefficients $B_x^T x_m$ $(1 \leq m \leq M)$des M signaux observées et l'arbre $tree(B_x)$ de la meilleur base

$$\mathbf{x} \rightarrow \boxed{MBestBasisAnalysis} \rightarrow B_x^T\mathbf{x}, tree(B_x)$$

où $\mathbf{x}$ est le signal multicanal d'entrée.

Maintenant on se trouve dans le domaine transformé, notre signal est projeté sur la base B_x choisie par Best Basis. C'est ici qu'il commencer le traitement pour la séparation de sources.

Le masquage c'est *BinaryMasks.m* qui le fait. Avec la matrice de mélange $\mathbf{A}$ et les coefficients des signaaux $B_x^T\mathbf{x}$ on fait une première estimation de $B_x^T\widehat{\mathbf{s}}'$. Après on trouve le composant le plus énergétique pour chaque k et avec cette information on fait le masque $\widehat{masc}$.

$$B_x^T\mathbf{x}, \mathbf{A} \rightarrow \boxed{BinaryMasks} \rightarrow \widehat{masc}$$

Encore avant de la reconstruction, il faut estimer $B_x^T\widehat{\mathbf{s}}$.

Pour construire les estimations des sources comme expliqué section 4.1 (page 33), *SeparationMatrices.m* fait un tableau F du même longueur que le signal $x_m, (m = 1, 2)$. Dans le tableau F chaque position $F[k]$ est une matrice de la taille de $\mathbf{A}^\dagger$ selon la forme $F[k] = [f_1[k], \ldots, f_N[k]]^T$. Pour chaque instant k et chaque source j, $f[k]$ est

$$f_j[k] = \begin{cases} a_j^\dagger & si \ \widehat{masc}_j = 1 \\ 0 & si \ \widehat{masc}_j = 0 \end{cases}, \qquad j = 1, \ldots, N$$

$$\widehat{masc}, \mathbf{A} \rightarrow \boxed{SeparationMatrices} \rightarrow F$$

et $B_x^T\widehat{\mathbf{s}}$ devient :

$$B_x^T\widehat{\mathbf{s}} = F[k] \cdot B_x^T\widehat{\mathbf{x}}.$$

La reconstruction des signaux des sources estimées fait le procès inverse que pour la décomposition :

$$B_x^T\widehat{\mathbf{s}}, B_x tree \rightarrow \boxed{MBestBasisSynthesis} \rightarrow \widehat{\mathbf{s}}$$

4.3 Analyse des critères

Le problème de choisir les sources qui sont actives dans un première estimation $\widehat{\mathbf{s}}$ de $\mathbf{s}$ c'est qu'on n'a pas le vrai sources pour dire quelle est la source la plus énergétique ni pour dire quelle est la meilleur base pour les représenter.

On peut faire la comparaison de quelle est vraiment la différence entre faire l'algorithme avec la masque et la base qui correspond aux vrai sources (section 4.3.1, page 36).

Si on a les vrai sources on peut aussi modifier nos critères de calcule pour trouver la meilleure base ou un masque plus effectif. Un critère qui semble intéressant c'est la minimisation de la distortion (sections 4.3.2, page 37 et 4.3.3, page 38).

Après *restructurer* les critères pour faire la séparation de sources on doit choisir des nouveaux critères pour faire la séparation en aveugle. L'idée est faire une algorithme de *clustering*. Pour le faire on va utiliser $\mathbf{s}$ et $\mathbf{x}$ pour obtenir un modèle de cluster. On peut définir quelque paramètres $\theta_i \ \ i = 1, 2, \ldots$ selon les propriétés des signaux qu'on est en train de traiter (section 4.3.4, page 39).

Et pour finir on peut faire l'algorithme final avec $\mathbf{x}$ et certains des paramètres θ_i (section 4.3.5, page 40).

On va voir tout ça dans les sections suivants.

4.3.1 Si on connaît un partie de S

En connaissant les sources on peut voir les différents résultats pour la «vraie» meilleure base et le vrai masque.

On appel vraie meilleure base B_s à cela qui minimise la fonction de coût $\mathcal{C}$ pour le vecteurs de sources $\mathbf{s}$ (voir section 3.2)

$$la \ vraie \ meilleure \ base \ B_s, \qquad B_s = arg \min_{B \in \mathcal{B}} \mathcal{C}(B^T \cdot \mathbf{s}),$$

et on appel vrai masque le masque binaire qui vient de l'algorithme *Binary-Masks.m*. Pour le faire on parte des composantes des sources $\mathbf{s}$ dans la meilleure base B_x ou dans la vraie meilleure base B_s

$$B_x^T\mathbf{s}, \mathbf{A} \rightarrow \boxed{BinaryMasks} \rightarrow masc_x$$

$$B_s^T\mathbf{s}, \mathbf{A} \rightarrow \boxed{BinaryMasks} \rightarrow masc_s$$

où $masc_x$ vient du vecteur $B_x^T\mathbf{s}$ et $masc_s$ vient du vecteur $B_s^T\mathbf{s}$.

Il y a alors quatre possibilités pour l'estimation des sources $\widehat{\mathbf{s}}$:

1. les estimations faites sur le domaine de la meilleure base B_x dont les composantes sont :

$$B_x^T\widehat{\mathbf{s}}(\widehat{masc}_x), \text{ ou} \qquad B_x^T\widehat{\mathbf{s}}(masc_x)$$

   où $B_x^T\widehat{\mathbf{s}}(masc_x)$ sont les composantes reconstruites à partir du masque $masc_x$

2. les estimations faites sur le domaine de la vraie meilleure base B_s dont les composantes sont :

$$B_s^T\widehat{\mathbf{s}}(\widehat{masc}_s),\text{ou} \qquad B_s^T\widehat{\mathbf{s}}(masc_s).$$

On verra les comparaisons des résultats avec ces quatre «methodes» dans la section des expériences (section 4.4).

4.3.2 *Ideal Mask*

Après jouer un peu avec les mêmes algorithmes pour les différentes possibilités on va commencer pour changer quelque chose. La premier idée qu'on a c'est de modifié le méthode d'obtention du masque binaire. On disait avant que la source active correspondait à la composante k la plus énergétique d'une première estimation $\mathbf{s}^{'}$ des sources.Nous appellerons le masque correspondant ***MaxEnergy Mask***. Maintenant on va voir quelle est la source avec laquelle on a le moins de distorsion. Pou cela on a besoin de la vraie $B_x^T\mathbf{s}$ et de l'estimation $B_x^T\widehat{\mathbf{s}}$:

$$B_x^T\widehat{\mathbf{s}}[k] = \frac{a_n^\dagger \cdot B_x^T\mathbf{s}[k]}{|a_n^\dagger|}$$

où a_n est la colonne qui correspond à la source n. Et la distortion est :

$$f_{dist\ j}[k] = |B_x^T\widehat{s_j}[k] - B_x^T s_j[k]|^2 - |B_x^T s_j[k]|^2 + \sum_{n\neq j} |B_x^T s_j[k]|^2\ , \qquad j = 1, \ldots, N$$

La source qui donne moins distorsion serait la seule source active pour nous.

$$masc_j[k] = \begin{cases} 1 & si \quad f_{dist\ j}[k] = \min_{(j=1,\ldots,N)}(f_{dist\ j}[k]) \\ 0 & si \quad f_{dist\ j}[k] \neq \min_{(j=1,\ldots,N)}(f_{dist\ j}[k]) \end{cases}$$

Nous appellerons le masque correspondat ***Ideal Mask***.

4.3.3 *Ideal Best Basis*

Avec la même idée d'*Ideal Mask* on a pensée faire une autre méthode de sélection de la meilleure base. On veut chercher maintenant l' ***Ideal Best Basis***.

Pour faire ça on doit entrer dans le détails de l'algorithme de choix de la meilleure base. L'idée est de chercher la base qui donne le moins distorsion. Comme on sait du chapitre 3 (page 27) pour choisir la meilleure base on doit dire si on va utiliser soit les bases d'ondelettes soit de cosinus. En plus, on doit dire aussi quelle fonction de coût utiliser. Pour l'instant on va considérer qu'on veut travailler avec les paquets d'ondelettes et que notre fonction de coût est l'addition de l'entropie (voir [restauration, Chap. 8]).Il faut dire qu'on travaille sur Matlab et on a besoin d'utiliser les paquets de WaveLab. Pour voir les propriétés des fichiers *point m* on peut utiliser le manuel [Buckheit et al. & Scargle, J.,1995].

Bref, pour avoir la meilleure base on propose un algorithme qui pred comme entrées les signaux $\mathbf{x}$ et les sources $\mathbf{s}$ ainsi que $\mathbf{A}$. Pour chaque signal on a un tableau de $D+1$ bases pour les représenter. En étant D le longueur dyadique du signal. Ça veut dire que si on avait le vecteur $\mathbf{x}$ de M canal, maintenant on aurai un tableau $\mathbf{x_{pkt}}$ de $M \times (D+1)$ vecteurs qui correspondent au paquets d'ondelettes. Pour la prise des paquets on utilise le *WPAnalysis.m* :

$$\mathbf{x}, \mathbf{s} \rightarrow \boxed{\textit{WPAnalysis}} \rightarrow \mathbf{x_{pkt}}, \mathbf{s_{pkt}}$$

Comme on avait fait tout a l'heure avec l'*Ideal Mask* on utilise la matrice de mélange $\mathbf{A}$ pour faire les estimations des sources, mais cette fois sur les paquets :

$$\widehat{s}_{pkt_{n,d}}[k] = \frac{A_n^{\dagger} \cdot \mathbf{x_{pkt_d}}[k]}{|A_n^{\dagger}|}$$

où $()_{n,d}[k]$ est l'échantillon k du paquet pour la source n ($n = 1, \ldots, N$) dans la base d ($d = 1, \ldots, D+1$).

La fonction de distortion $f_{dist_{n,d}}$ pour chaque instant k est :

$$f_{dist_{n,d}} = |\widehat{s}_{pkt_{n,d}}[k] - s_{pkt_{n,d}}[k]|^2 - |s_{pkt_{n,d}}[k]|^2 + \sum_{l \neq n} |s_{pkt_{l,d}}[k]|^2$$

Notre élection de la meilleure base passe maintenant pour trouver cela qui fait le moins de distorsion. On peut faire ça avec la fonction de coût *addition de distorsions*. L'élection est donne pur :

$$arg \min_{btree} \sum_{(d) \leftarrow btree} f_{dist_{n,d}}$$

où $(d) \leftarrow btree$ indique les $D+1$ indices des bases possibles avec l'arbre *btree* (voir section 3.3).

Pour trouver la meilleure base la combinaison de *CalcStatTree* et de *BestBasis* de WaveLab font pour la fonction de coût *addition* et pour n'importe quel $f_{n,d}$:

$$B = arg \min_{btree} \sum_{(d) \leftarrow btree} f_{n,d}$$

où B est la meilleure base pour la fonction de coût $\mathcal{C}(f_{n,d})$.

Alors, si on applique $f'_{dist_{dj}}$ qui vient de la posibilité de quiter le valeur constante $\sum_{l=1}^{N} |s_{pkt_{l,d}}[k]|^2$ de $f_{dist_{n,d}}[k]$ dans une minimisation :

$$f_{dist_{dj}} = \sum_{l=1}^{N} |s_{pkt_{l,d}}[k]|^2 + |\widehat{s}_{pkt_{n,d}}[k] - s_{pkt_{n,d}}[k]|^2 - |s_{pkt_{n,d}}[k]|^2.$$

$$f'_{dist_{n,d}}[k] = |\widehat{S}_{pkt_{n,d}}[k] - S_{pkt_{n,d}}[k]|^2 - |S_{pkt_{n,d}}[k]|^2$$

sur la fonction de minimisation de l'algorithme *bestbasis*, on aurait la meilleure base cherché :

$$\textbf{IdealBestBasis} = arg \min_{btree} \sum_{(d) \leftarrow btree} f'_{dist_{n,d}}$$

4.3.4 À la chasse de paramètres

À partir d'ici tout le reste est expérimental. Dans la première partie pour travailler avec B2S2 on connaissait le signal d'entrée et la matrice de mélange. Connaissent une partie des sources on peu chercher des masques et des bases qui doivent amélior les résultats. Maintenant il est temps d'estimer $\mathbf{A}$. Pour faire ça on va essayer de récupérer des propriétés des sources, des mélanges et des estimations pour voir quels sont les pièces qu'on pourrait utiliser pour construire un bon algorithme de *clustering*.

Les propriétés qu'on cherche pour faire un algorithme de *clustering* sont des propretés qui viennent plutôt des caractéristiques de l'enregistrement stéréo commentées dans le premier chapitre.

Ces propiétés sont surtout l'angle et l'énergie qu'ont chaque compossant. Comme vraiment on ne sait pas quelles sont les sources, on va suivre les critères de toujours. On va penser qu'il y a un seule source active pour chaque composant k, donc pour cet instant notre signal

$$\mathbf{x} = \begin{bmatrix} x_{left} \\ x_{right} \end{bmatrix}$$

est le résultat du mélange d'une seule source. Cet-á-dire, que la relation entre l'énergie du signal de droite et du signal de gauche peut nous dire quelle est la direction de la seule source active :

$$\theta = \arctan\left(\frac{x_{right}}{x_{left}}\right).$$

Comme on travaille sur le domaine transformé, dans un base B_x on a différents angles pour chaque composant de temps et de fréquence.

Par rapport à l'énergie de $\mathbf{x}[k]$ on fait directement l'addition des énergies de chaque composant x_i.

4.3.5 Le *clustering*

Pour le *clustering* on ne va travailler qu'avec l'angle θ et l'énergie ρ.

La première idée pour la séparation de sources est prendre les composants par groupes selon ses angles. Si on a la matrice de mélange $\mathbf{A}$ on aurait les différents droites a_i qui disent quel est l'angle des sources :

$$\theta_i = \arctan\left(\frac{a_2}{a_1}\right).$$

Le *clustering* fait la séparation en ayant en compte que pour chaque composant on a un seule source active. On dit que le composant $B_x^T x[k]$ appartient à la source s_i si la droite a_i est la plus proche. Dans le rang des angles la distance la plus petite est équivalent à l'angle plus proche. Ce *clustering* joue le même rôle que le masquage de notre algorithme, et à partir d'ici la séparation suit le même cours.

Si on a pas $\mathbf{A}$ le *clustering* devient plus complexe. Une idée qui peut nous arriver, c'est trouver les régions de la scène où il y a le plus de composantes. Intuitivement dans les différents régions où on a les plus grands nombre de composants on a un source.

Le problème maintenant c'est de choisir quel est l'angle à donner à notre droite a_i. Vraiment il faut pas refaire $\mathbf{A}$ parce que on peut faire directement la séparation des sources en faisant un *clustering* sur les composants qu'on a. Mais pour travailler comme avec *la première idée* on dirait qu'on est en train d'estimer $\mathbf{A}$ pour faire la séparation après.

Comme méthode de choix on va utiliser le plus simple, on va prendre directement les maximums de l'histogramme d'angles (voir figure 4.15). Pour faire ça, il reste encore un problème : on doit savoir quel est le nombre de sources qu'on doit chercher. D'ici jusqu'à la fin on va supposer qu'on connaît les nombre des sources.

Peut être que l'algorithme de *clustering*, qu'on vient d'exposer pour trouver les droites a_i, et les groupes de sources, a quelque problèmes.

En partant d'un erreur angulaire, il est possible que quand un composant a moins d'énergie l'information de l'angle est moins fiable. On peut faire un interprétation graphique de cela. Si le points sont trop proche au l'origine, le minimum erreur il peut provoquer que l'angle soit très différent. Dans la même interprétation graphique c'est aussi intuitive de penser que le points qui sont plus loin du l'origine ajouteraient moins d'erreur par rapport à l'angle. Pour les donnés qu'on a on peut donner le même argument, et dire que les composants avec plus d'énergie ont moins d'erreur angulaire.

À partir de cet idée on va faire un histogramme des angles qui a en compte l'énergie de chaque composant. La relation qu'on va utiliser serait en relation avec le logarithme du ρ. Pour chaque position, dans le rang des angles, on ferait l'addition directe du

$$Histogramme(\theta) \mathrel{+}= \log(\rho)$$

Touts ces idées sont expérimentées dans la section 4.4.

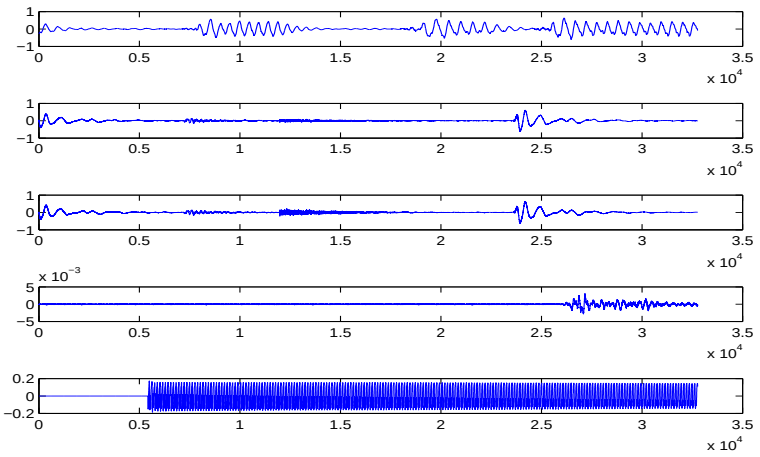

FIG. 4.1 – les cinq sources des expériences

4.4 Expériences et résultats

Le première chose qu'on va montrer ce sont les résultats obtenues avec B2S2 sur certains signaux de son qu'on peut trouver sur [Bass-dB]. Ensuite on va continuer avec les différentes expériences pour trouver un meilleure algorithme de séparation. Pour cela on va essayer des nouveaux masques et trouver des autres meilleures bases. Et pour finir on cherchera des propriétés de l'algorithme et des signaux d'entrée pour les utiliser dans un autre algorithme de *clustering* qui puisse trouver les sources en aveugle.

Les résultats des expériences sont resumées dans des tableaux, quand il convient, et des figures. Les tableaux contiennent le SDR (*Source to Distorsion Ratio*) le SIR (*Source to Interference Ratio*) et le SAR (*Source to Artifacts Ratio*), rapports exprimées das [Reulos,2003, pp. 36–38].

4.4.1 algorithme B2S2

Dans nos expériences on a travaillé dans le cas dégénéré avec un signal d'entrée stéréo ($M = 2$) et plus de deux sources à trouver ($N > M$).On peut voir les résultats de l'algorithme B2S2 sur les cinq sources de la figure 4.1 dans la figure 4.2 et le tableau 4.1.

TAB. 4.1 – SDR, SIR et SAR pour l'estimation de B2S2 sur 5 sources

	s_1	s_2	s_3	s_4	s_5
SDR	5.5800	-2.9768	-2.3954	-42.9599	16.1998
SIR	22.0044	6.3200	12.0633	-29.8529	27.4618
SAR	5.7073	-1.5228	-1.9750	-12.8848	16.5451

Comme on peut voir les sources moins énergetiques ont les pires resultats. Surtout la source s_4, que vraiment a tres peut d'énergie, mais son estimation $\widehat{s}_4$ a plus d'energie. La raison est que dans le masquage on est dit plus fois de ces

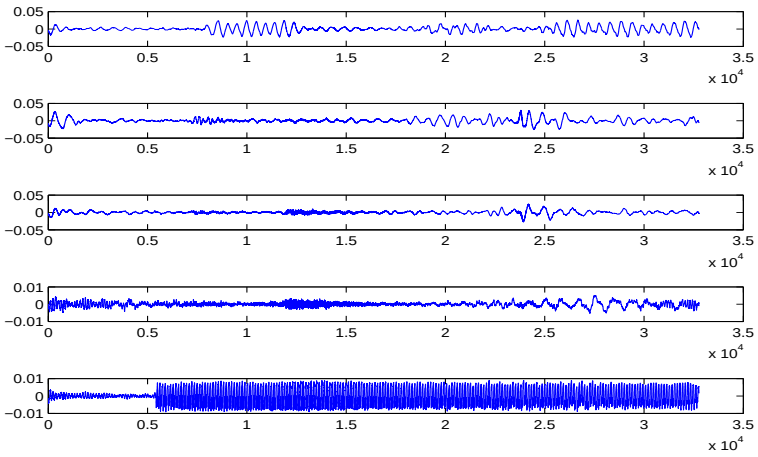

FIG. 4.2 – résultats de l'algorithme B2S2 sur cinq sources

qui sont vrai que la source est active.

4.4.2 avec les vrai sources

Après des résultats sur le signal $\mathbf{x}$ en ayant $\mathbf{A}$ on va commencer l'approche de qu'est ce qui donne l'algorithme si on utilise l'information des sources pour faire les masques et le choix de la meilleure base. On a quatre estimations possibles des sources qui correspondent aux estimations avec les masques $\widehat{masc}$ et $masc$, dans la meilleure base B_x pour le signal $\mathbf{x}$ et aux estimations dans la vrai meilleure base B_s.

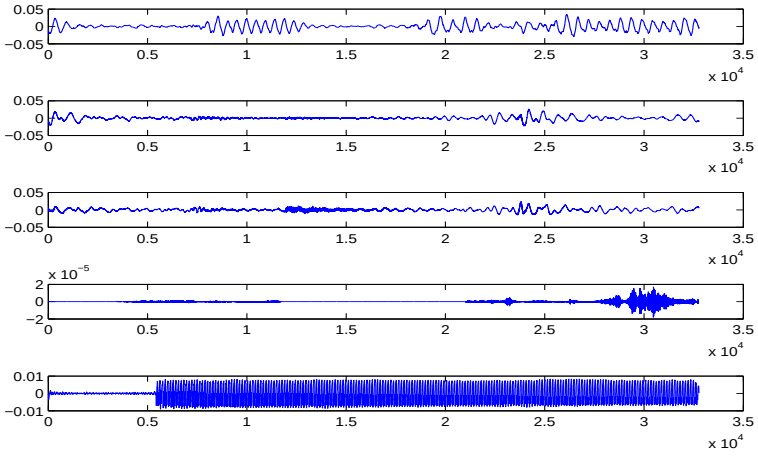

FIG. 4.3 – B2S2 pur le vrai masque $masc$ dans le domaine de la base B_x

Dans la figure 4.3, où le mascage est fait avec le «vrai masque», on peut voire que la forme d'onde de la source estimée $\widehat{s}_4$ devien plus proche par rapport à la forme d'onde de original. On peut dire que la contribution des autre sources dans cela est plus petite qu'avant. En regardant la table 4.2 on peut voire comme le

TAB. 4.2 – SDR, SIR et SAR pour l'estimation de B2S2 qui correspond au vrai masque $masc_x$

	s_1	s_2	s_3	s_4	s_5
SDR	7.6437	-0.4983	-3.7630	-34.1532	22.6108
SIR	13.1694	13.8654	15.3017	18.4619	44.1287
SAR	9.2758	-0.1616	-3.5825	-34.0917	22.6417

le rapport SIR deviens meilleur, puisque les interfernces sont meilleur controlées avec ce masque. On peut voire aussi que les resultats son meilleurs, surtout pur les sources plus energetiques.

Par rapport au «vraie meilleure base» on peut voir déjà sur la figure 4.4 et la table 4.3 que les resultats obtenues ne disent pas grand chose. Peut être que la base adéquate pour travailler sur la siganl $\mathbf{x}$ est la base obtenue à partir d'elle même.

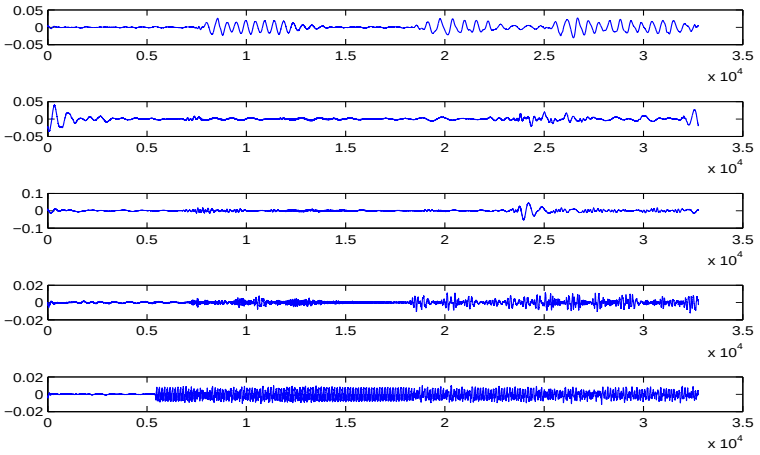

FIG. 4.4 – B2S2 pur le masque $\widehat{masc}$ dans le domaine de la base B_s

TAB. 4.3 – SDR, SIR et SAR pour l'estimation de B2S2 qui correspond au masque $\widehat{masc}_s$

	s_1	s_2	s_3	s_4	s_5
SDR	5.4503	-2.9920	-2.1457	-28.6728	13.4833
SIR	30.4566	5.3188	12.3643	-13.9285	26.7802
SAR	5.4679	-1.1799	-1.7442	-14.4239	13.7006

4.4.3 pour *ideal mask*

Si on a les sources $\mathbf{s}$ on peut modifier les critères pour faire la masque binaire qui dit quelle est la source active à chaque instant (et chaque fréquence), et pour choisir la meilleure base.

Avec cet idée on a construit des algorithmes pour faire un *Ideal Mask*. Le fait d'appliquer ce masque sur les composants $B_x^T\mathbf{x}$ donne les résultats montrés dans la figure 4.5.

TAB. 4.4 – SDR, SIR et SAR pour l'estimation de B2S2 qui correspond au *ideal masque*

	s_1	s_2	s_3	s_4	s_5
SDR	7.9271	-11.7456	3.5448	-26.9529	22.4779
SIR	13.5299	-0.6518	16.5860	18.4542	46.5990
SAR	9.5137	-8.0457	3.8602	-26.8912	22.4948

Le fait d'appliquer ce masque sur les composants $B_x^T\mathbf{x}$ donne les résultats montrés dans la figure 4.5. On peut voir comme les formes d'onde, comme avec la «vrai masque» sont plustôt bonnes. Il y a plus information a commenter sur la table 4.4. On peut voire que les resultats son encore meilleur qu'avant pour notre estimation $\widehat{s}_4$ puis qu'on est estimé la distorsion minimal.

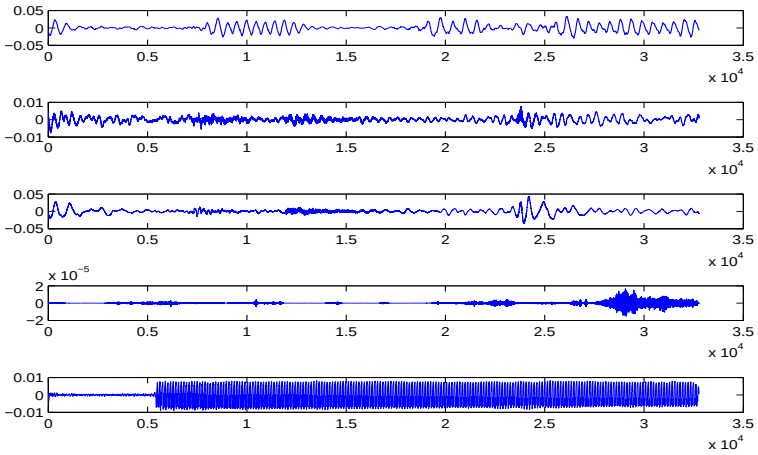

FIG. 4.5 – B2S2 avec ***Ideal Mask***

Un première résultat de touts ces expériences c'est celui qu'on voit en regardant les derniers estimations. On a un très bonne résultat pour une seule source active pour chaque morceaux de temps et de fréquence. Ça savait dire que travailler en cherchant un algorithme de séparation qui considère qu'on a un seule font active à la fois peut donner bonnes résultats en cherchant la bonne masque et la bonne base.

4.4.4 pour *ideal bestbasis*

On a déjà notre masque idéal et on peut voir si en trouvent une meilleure base on aurait des meilleurs résultats. On va travailler sur le domaine de la *Ideal BestBasis*.

Même si les resultats sur la table 4.5 ne sot pas tres mouvais, on dois regarder la figurel 4.7 popur nous rendre compte de que autre fois le choix de l'ideal base ne marche pas assez bien que en trouvant la masque idéal.

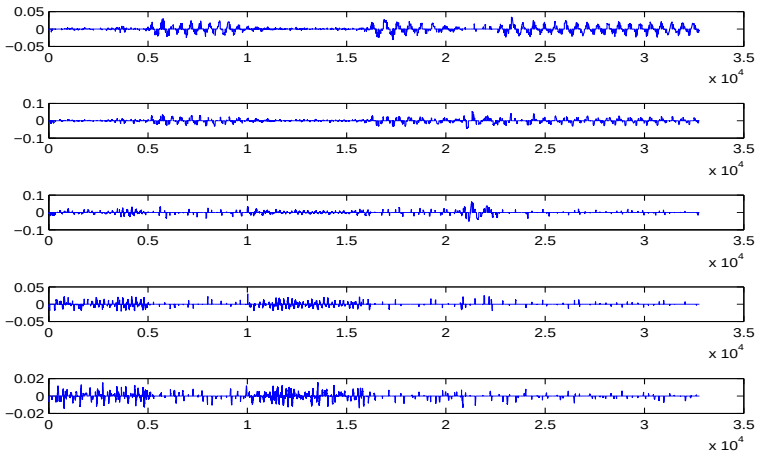

FIG. 4.6 – B2S2 avec ***Ideal BestBasis***

TAB. 4.5 – SDR, SIR et SAR pour l'estimation de B2S2 qui correspond au *ideal best basis*

	s_1	s_2	s_3	s_4	s_5
SDR	9.2678	-4.7403	0.9175	-47.9291	7.1581
SIR	28.6631	7.0490	12.9312	-37.4608	30.2603
SAR	9.3239	-3.6607	1.4153	-10.0589	7.1835

Peut être que la raison soit la même que la comanté au section 4.4.2, mais c'est aussi tout à fait possible qu'il y a des erreurs de calcule ou d'implementation que comme stagiaire n'ai pas reussi à trouver.

4.4.5 la recherche de paramètres

Un fois qu'on a expérimenté avec B2S2 et les différents possibilités de décomposition et de masquage, on va commencer avec des autres expériences pour chercher un algorithme qui travaille en aveugle : le B3S2 (Best Basis Blind Separation Sources).

Pour faire la recherche on veut trouver des paramètres pour nous aider plus tard à faire l'algorithme de *clustering*. On veut connaître les variables qui interviennent pour savoir où est ce que on peut regarder pour faire le *clustering*.

On va voir des différents expériences sur les variables dans les suivants paragraphes.

CloudOb commence pour trouver les variables qui jouent dans la séparation non-dégénéré ; c'est-à-dire, le nombre de sources à trouver sera deux. La recherche de paramètres se fait directement sur les composantes du signal dans le domaine de la base B_x.

Dans la première figure (fig.4.8) on a représenté les composantes qui correspondent à chaque source en faisant le masquage *Max EnergyMask* (en haut) et *IdealMask* (en bas). Les droites a_i nous indiquent quelle est la direction où se

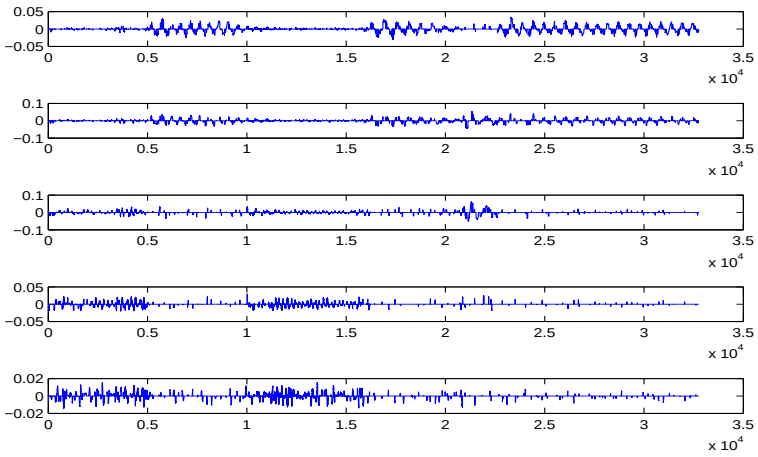

FIG. 4.7 – B2S2 avec ***Ideal BestBasis***

trouve la source.

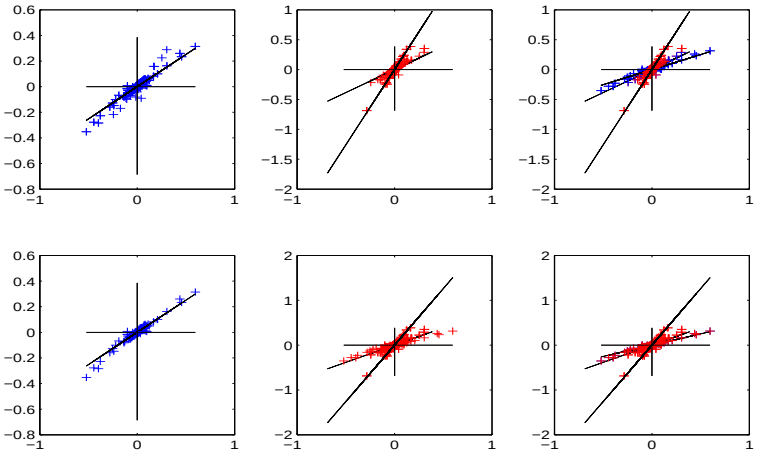

FIG. 4.8 – group X1/X2 pour MaxEnergy et IdealMask

Les représentations de la figure 4.9 ont relation avec l'énergie ρ de chaque $B_x^T x[k]$ et avec leurs projections sur les droites a_i qui correspondent.

Les graphiques de la figure 4.10 mettent en relation les angles et les énergies. En première ligne on a la représentation des énergies par rapport à son angle. Aux autres lignes voit les histogrammes du logarithme de l'énergie ρ et des angles θ.

Les angles sont représentés maintenant pour les histogrammes qu'on peut voir sur la figure 4.11. Chaque courbe représenta l'histogramme des angles des composantes de chaque groupe.

Pour finir, on compare la séparation qu'on peut faire en utilisant la masque *Max EnergyMask* avec la séparation que peut nous donner la minimisation de la distance a la droite a_i. Dans la figure 4.12 on en haut les résultats sur le masque et en bas sur un première *clustering* qui fait la séparation en regardant quelle

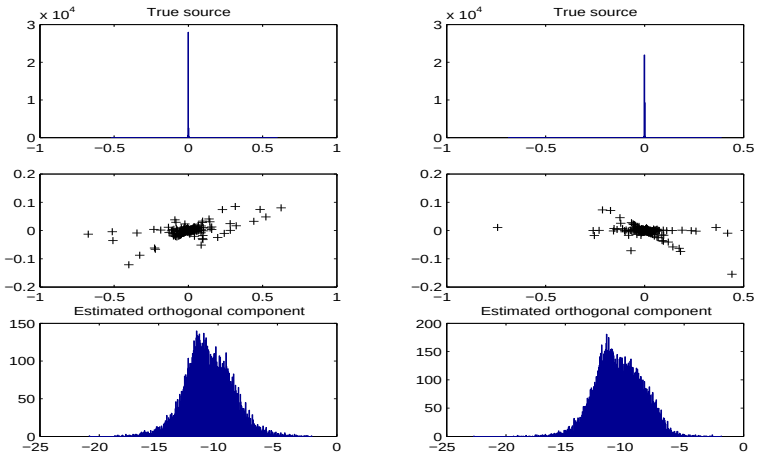

FIG. 4.9 – group X1/X2 pour MaxEnergy et IdealMask

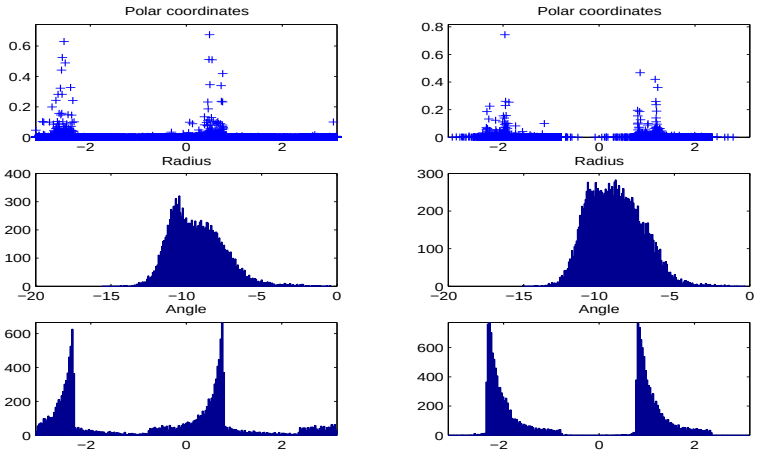

FIG. 4.10 – group X1/X2 pour MaxEnergy et IdealMask

est la droite la plus proche.

ClusterA fait le même que le dernier part de *CloudOb* mais dans le cas dégénéré. Pour l'algorithme on a les composantes qui correspondent à $\mathbf{x}$, la matrice de mélange $\mathbf{A}$ et on doit trouver cinq sources.

Dans la figure 4.13 on voit les estimations des sources et les vraie sources.

Pour comparer avec les masquages qu'on a utilisée pendant l'étude, on a mis aussi les tables 4.6 et 4.7. Dans la première table (table 4.7) on a le nombre de composantes qu'on a pour chaque source. Comme on peut voir, il y a une grande différence pour chaque masquage.

Oú *ClusterA* correspond au minimisation de distance à la droite a_i, *Clouds* est pareil mais en aveugle et estimant les a_i, et *IdealMask* est le masque qui minimise la fonction distortion qu'on a vu en avance.

Dans la deuxième table (table 4.7) on a les relations SDR-SDI-SDA pour

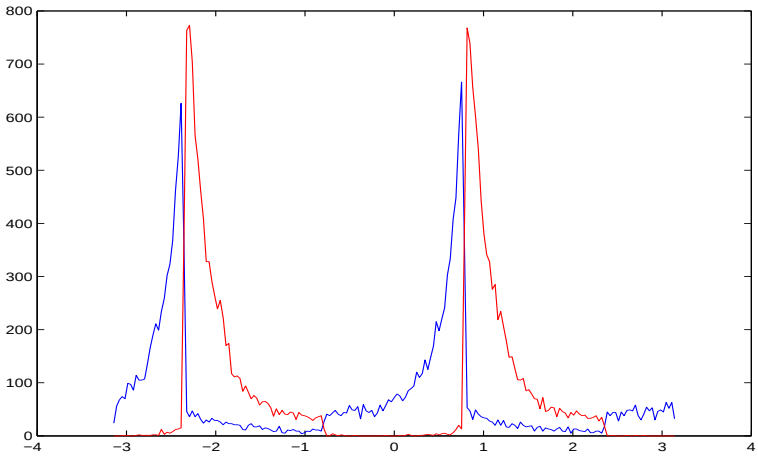

FIG. 4.11 – group X1/X2 pour MaxEnergy et IdealMask

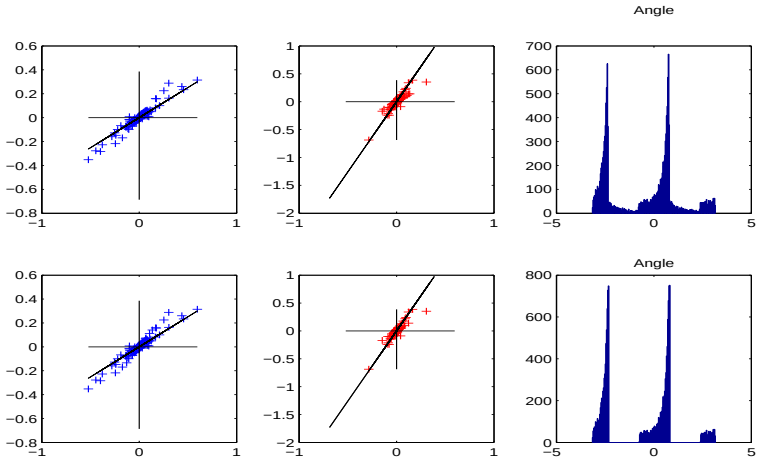

FIG. 4.12 – group X1/X2 pour MaxEnergy et IdealMask

chaque source et chaque masquage.

ClusterA2 compare cette fois le domaines de la meilleure base avec la meilleure base idéale. On fait de comparaisons avec les différentes sortes de séparation dans les deux domaines. Les résultats sont présentés cette fois en fonction des spectrogrommes de chaque source (voir figure 4.14).

4.4.6 le *clustering*

Après ces expériences on ne voit pas trop bien quels sont les paramètres à choisir. Pour l'instant on a seulement l'angle du signal stéréo et l'énergie de chaque composante dans la nouvelle base.

Dans l'histogramme de la figure 4.15 on peut voir les différentes régions.

Si on compare avec les directions des droites a_i on peut savoir si un *clustering*

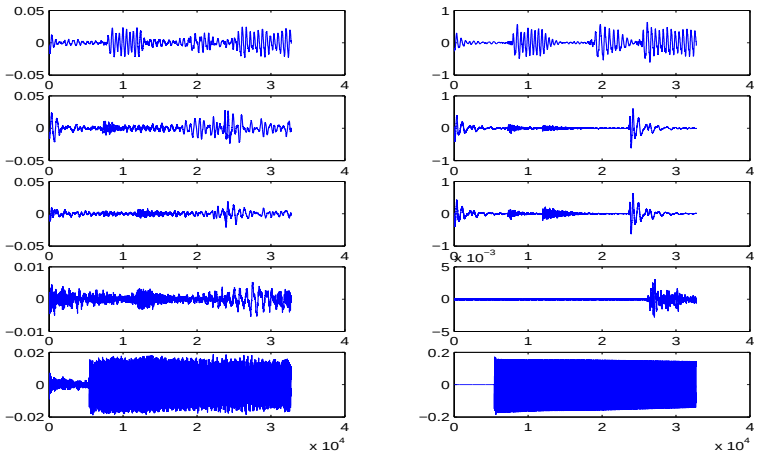

FIG. 4.13 – group X1/X2 pour MaxEnergy et IdealMask

TAB. 4.6 – tab :ClusterA1

Méthode	s_1	s_2	s_3	s_4	s_5
ClusterA	6763	7162	7459	3166	8218
Blind(Clouds)	6326	2409	4496	9219	10318
Ideal Mask	2825	10713	17716	225	1289

sur l'angle ferait vraiment quelque chose (voir fig.4.16).

Déjà c'est un résultat. Mais on sait que cette histogramme ne donne pas un information très efficace sur le vrai angle de la source estimé à partir de chaque composante. Peut être que si on a en compte l'énergie de chaque composante ; c'est à dire, si on fait que si on a un composant plus énergétique l'importance sur l'histogramme soit meilleure, l'histogramme serait plus efficace.

Dans cette idée là, et toujours dans la même idée de chercher les droites a_i on peut voir les résultats à la figure 4.17.

Comme on peut voir les résultats sont clairement meilleures qu'avant. On a réussi à trouver les bonnes droites pour les sources les plus énergétiques, et on a des résultats pas mauvais pour les autres.

4.5 Conclusions et perspectives

À partir des expériences réalisées on a les suivants résultats :

1. L'approche d'avoir un seule source active pour chaque composant $B_x^T\mathbf{x}[k]$ est valide jusque aux termes des tables 4.2 et 4.4. Si on trouve des bonnes masques, comme la vrai masque ou la masque idéal, on peut arriver aux bonnes séparations.
2. Le choix de la meilleure base est faisable directement sur le signal $\mathbf{x}$ puisque on va travailler avec le vecteur $B_x^T\mathbf{x}$.
3. Si on a en compte le énergie qu'il y a dans chaque composant, on peut peut faire un bonne estimation des droites des sources les plus énergétiques en

TAB. 4.7 – ClusterA2

Cluster A :	SDR	SIR	SAR
s_1	(5.58) dB	(289.4614) dB	(5.58) dB.
s_2	(-2.9768) dB	(279.2814) dB	(-2.9768) dB
s_3	(-2.3954) dB	(267.6938) dB	(-2.3954) dB
s_4	(-42.9599) dB	(294.9584) dB	(-42.9599) dB
s_5	(16.1998) dB	(276.7869) dB	(16.1998) dB
Blind(Clouds) :			
s_1	(1.8267) dB	(289.497) dB	(1.8267) dB
s_2	(-10.0414) dB	(279.299) dB	(-10.0414) dB
s_3	(-1.717) dB	(267.6775) dB	(-1.717) dB
s_4	(-33.4704) dB	(294.8077) dB	(-33.4704) dB
s_5	(16.1191) dB	(276.789) dB	(16.1191) dB
Ideal Mask :			
s_1	(7.9271) dB	(289.4797) dB	(7.9271) dB
s_2	(-11.7456) dB	(279.2338) dB	(-11.7456) dB
s_3	(3.5448) dB	(267.6785) dB	(3.5448) dB
s_4	(-26.9529) dB	(295.2257) dB	(-26.9529) dB
s_5	(22.4779) dB	(276.8057) dB	(22.4779) dB

décrément des sources moins énergétiques. Ces droites nous indiqueront quelle est la source assignable à chaque composant. Et le méthode pour les trouver est chercher l'angle où il y a plus de composants et avec plus de énergie.

À partir des résultats on trouverait intéressant faire :

1. Par rapport au choix de la masque :
   (a) faire un algorithme d'apprentissage où on n'ait pas besoin de $\mathbf{s}$ pour exemple en faisant la première estimation $\widehat{\mathbf{s}}$ avec la *MaxEnergie Mask* et faire usage après de cet estimation pour refaire l'algorithme avec *Ideal Mask*.
   (b) Chercher un autre critère pour faire un masque qui ne soit pas binaire. Dans notre cas stéréo, on peut faire un reconstruction en ayant en compte qu'on a deux sources actives, puis que on travaillerait avec un matrice carré :

$$\mathbf{x} = [a_1, \ldots, a_N] \cdot \begin{bmatrix} s_1 \\ \vdots \\ s_N \end{bmatrix} \qquad \overset{mask}{\longrightarrow} \mathbf{x} = [a_P, a_S] \cdot \begin{bmatrix} s_P \\ s_S \end{bmatrix}$$

$$\begin{bmatrix} s_P \\ s_S \end{bmatrix} = \begin{bmatrix} a_P \\ a_S \end{bmatrix}^{-1} \cdot \mathbf{x}$$

   où P et S font référence aux source actives première et deuxième respectivement.
2. Pour faire le *Clustering* :

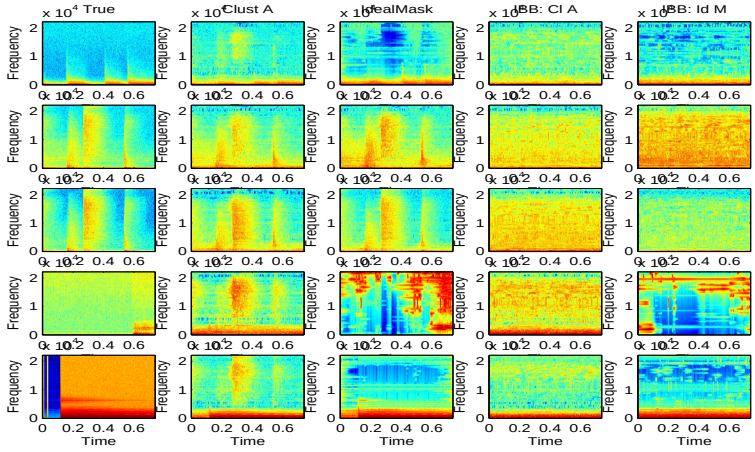

FIG. 4.14 – Histogramme(θ)

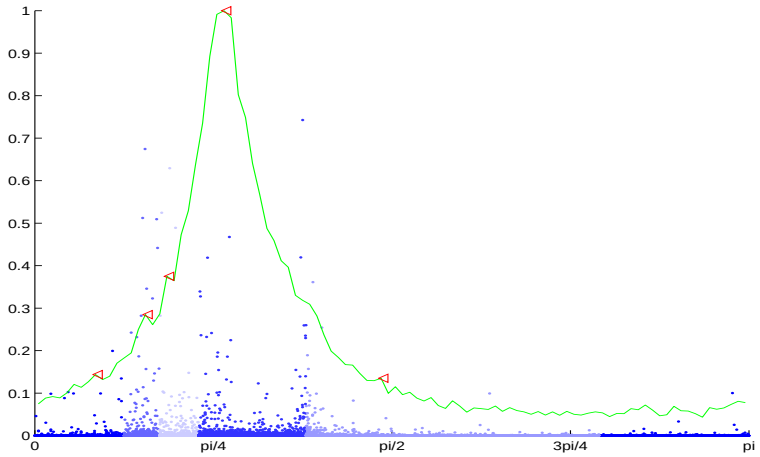

FIG. 4.15 – Histogramme(θ)

(a) choisir les maximums de l'histogramme

(b) faire un algorithme pour trouver combien de sources il y a. Au moins le nombre de sources qui donne le résultats les plus favorables. C'est possible faire l'algorithme pour plusieurs nombres de sources et comparer les résultats. Après on choit le nombre de sources qui correspond au résultat le plus favorable.

(c) une fois qu'on a les droites chercher un autre décision de *clustering* que le plus proche à la droite.

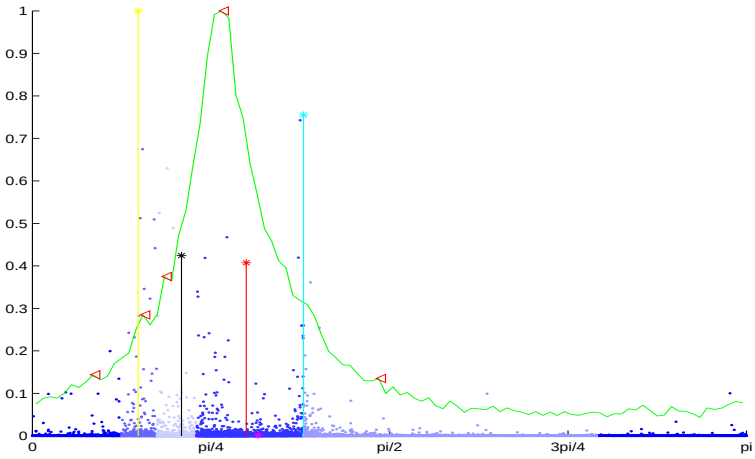

FIG. 4.16 – Histogramme de θ avec les droites a_i

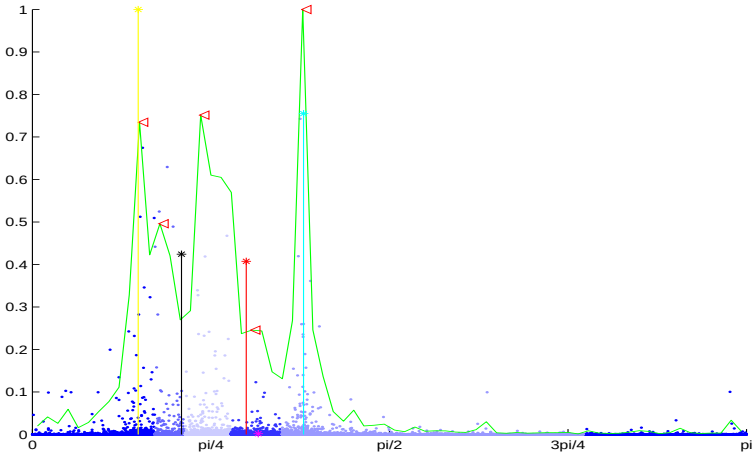

FIG. 4.17 – Histogramme avec $f_{histo}(\theta) += \log(\rho)$ pour 3 sources

Deuxième partie

Représentation du Son Stéréo

Chapitre 5
La représentation

Avec la représentation du son stéréo on veut dire la représentation en guise de graphiques 2D du son qu'on a sur deux canaux, *right* et *left*, de telle sorte qu'on ait tous les deux ensemble sur cette même graphique. L'idée c'est qu'on puisse voir dans un seule représentation les deux canaux et d'un seule coup d'oeil savoir comment est partagée l'information de notre son et dans notre scène stéréo. Ça peut nous aider à traiter le son stéréo, à décider quelles sont les algorithmes qu'on peut utiliser pour les différentes cas qu'on peut se trouver et a priori utiliser les caractéristiques de la représentation pour le traitement même.

Dans les lignes qui suivent on dira quels sont les sortes de représentation qu'on est estimé opportunes et sur la base de quoi, comment ils ont été développées, quels ont été les produits finals et comme conclusion on comparera l'idée base et les résultats obtenues.

5.1 Introduction

Les graphiques qui représente notre son stéréo sont de deux types principalement : graphiques cartésiennes de deux dimensions et les graphiques en guise de spectrogramme temps-fréquence. Les premières graphiques ne sont pas vraiment la représentation qu'on est en train de chercher, plutôt ils sont un pas intermédiaire avant d'arriver à la vrai représentation qui sera la représentation temps-fréquence.

Dans tous les deux cas le principe est toujours le même. On va prendre les deux signaux d'entré correspondant au canal de gauche et de droite, et on va réaliser les passes qu'on a suivi pour la séparation de sources.

D'abord il faut chercher l'espace de représentation adéquat. A priori,le domaine est la base correspondant à cela de l'algorithme BestBasis. Á partir de là on peut chercher l'énergie qu'on a pour chaque composant du signal, voir la relation entre le canal de gauche et droite et trouver l'angle d'où proviens l'énergie en fonction de cette relation.

Une fois qu'on ait l'énergie et la direction, on trouverait une représentation à couleur qui différencie ces deux facteurs la. Par exemple, dans un système de représentation de couleur HSI on peut utiliser une plus haute intensité de couleur pour indiquer qu'il y a une plus haute énergie et une variation du ton du rouge au vert pour indiquer un changement de direction entre la gauche et la droite.

5.2 Système de Cartésiennes

Comme on a expliqué à l'avance, celle la n'est pas la représentation stéréo dont serait mis en relation avec l'idée base d'un image qui doit nous apporter l'information sur l'énergie et direction du signal stéréo à chaque instant. Plutôt il s'agit de chercher les groupes des sources que apparaissaient dans notre enregistrement pour les identifier et essayer la séparation. Ce type de graphique nous a aidé à l'heure de réaliser la séparation de sources, en étant le première moyen de visualisation des composants de la base choisi pour les algorithmes BestBasis des signaux droit et gauche (voir figure 5.1).

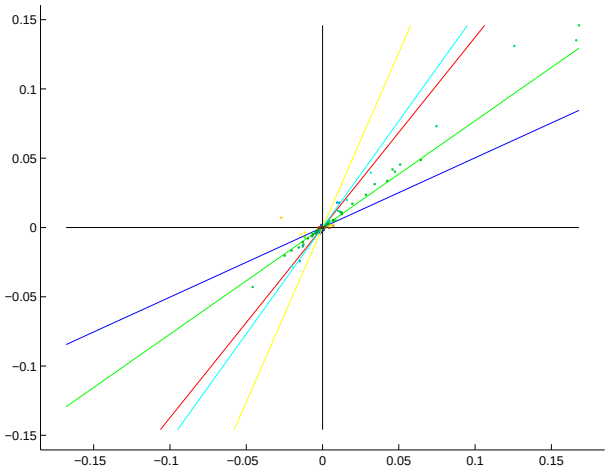

FIG. 5.1 – représentation X_1 vs X_2

Cette simple représentation des signaux dans le domaine de la base choisi nous donnerait déjà presque tout l'information qu'on a besoin, puisque chaque point est situé dans une direction déterminé et il possède une distance au origine proportionnelle á son énergie.

En fonction de la direction et de l'énergie on poudrait définir une fonction qui nous rendait une couleur déterminé avec lequel on colorierait chaque point. L'élection des couleurs qu'on utilise serait décisive pour avoir une représentation plus ou moins «lisible». Dans la section suivant on présenterait les différents formats de représentation en ce qui concerne à la couleur.

5.3 La Couleur

Une possibilité pour représenter les couleurs consiste en utiliser les couleurs primaires pour exprimer une couleur déterminé : en spécifiant quel quantité de chacun d'eux interviens dans le mélange. Ça c'est connaît comme le «système RGB». Son désavantage est que nous est compliqué de travailler avec cette système.

Une forme plus intuitive de représenter les couleurs est le système HSV (HSI ou HLS) qui peut se déduire facilement de l'antérieure. Cette système c'est l'usagé pour notre perception de couleur et il travail avec trois composants

basiques : nuance, saturation et éclat.

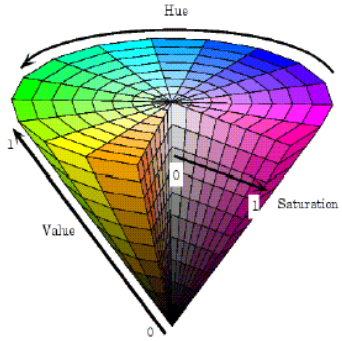

FIG. 5.2 – HSI Cone

La nuance parle de la couleur à soi même (rouge, vert), l'éclat se correspond avec notre appréciation subjective du clarté et obscurité (gris clair ou obscure), et la saturation parle des longueurs d'onde qui s'ajoutent au fréquence fondamental (sur le blanche).

Comme on peut voir sur la figure 5.2 cette système de représentation nous donne la possibilité de réaliser notre objective de façon simple. Vraiment la représentation du HSL est un double cône où l'éclat (value) irait du blanche au noire par tout le périmètre.

Un fois qu'on a le système de représentation de la couleur, il reste seulement définir quelle relation aurait entre la couleur et les entrées angle et énergie.

Pendant la réalisation du studio, ils se faisaient plusieurs essais avec différents relations de couleurs. En principe, pour la représentation de l'angle d'où proviens l'énergie il s'était proposé d'utiliser directement une variation du rouge au vert selon le ton. Pour l'énergie on avait deux option différents, soit en utilisant le noir comme fond où les points peuvent être désignes, soit en utilisant le blanche. Dans le premier cas quand l'énergie augmente il augmentera l'intensité de la couleur (l'éclat ou le valeur) selon :

$$\frac{\log(\rho \ / \ min)}{\log(max \ / \ min)}$$

(où max et min sont les valeurs les plus et les moins énergétiques respectivement)

et on maintiendrait fixe la saturation. Pour la deuxième possibilité il aurait un échange entre l'intensité et la saturation.

Tout le deux représentations ont étés faites avec la même idée, quand on déplace un source par rapport au point d'enregistrement, le ton changerait du rouge (gauche) au vert (droite). En plus la façon où il paraît disparaître le point, l'estompe qu'est au fond de l'image quand il n'y a pas beaucoup d'énergie, soit blanche soit noire, ça nous remet dans l'idée de combien d'énergie a un point par rapport aux autres d'une façon intuitive.

Autre point à voir c'est quels sont les nuances intermédiaires dans le rang du ton et quelle est la courbe que suit la «disparition» d'un point sur le fond de l'image.

Par rapport au rang du ton, la première option, pour être le plus intuitive et la plus simple, l'a fait passé pour le jaune. Cet option il paraissait un peu pauvre, donc on avait testé des autres alternatives. Ces alternatives passaient pour l'utilisation d'une troisième couleur, si on parle de RGB puisque jusqu'à maintenant on avait utilisé, au moins pour le cas avec fond noire, seulement le rouge et le vert. Certains alternatives qui avaient été essayés sont reflètes dans la figure 5.3 où on voit la représentation sur fond noire seulement.

L'autre préoccupation est faire un relation entre l'énergie et la couleur choisi pur chaque point. D'on début on croyait opportun pas représenter les point avec plus de 30 dB de différence par rapport à l'élément le plus énergétique. Même comme ça il restait l'obtention d'un courbe qui aille du maximum de 0 dB à -30 dB. L'élection de la courbe c'était surtout expérimental, à base d'essai-erreur. Finalement la courbe qu'on utilise est matricé pour l'expression :

$$\left(\frac{\log(\rho \ / \ min)}{\log(max \ / \ min)}\right)^{0.7}$$

Pour voir le différentes systèmes de couleur qu'il sont étés utilisés á cette propositions on va représenter cette fois les légendes de chaque système, angle et énergie, selon variation de *left* à *rigth* et de 0(db) à -30 (voir figure 5.3)

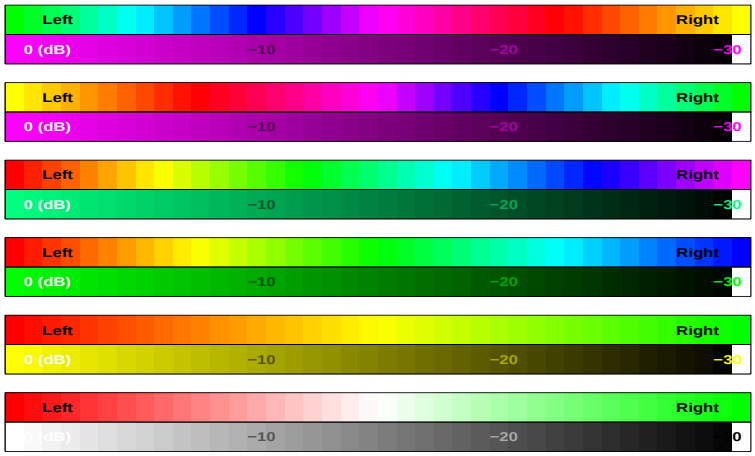

FIG. 5.3 – mythes et légendes

Les différentes systèmes de représentation venaient directement de l'intention de résoudre les différentes problèmes qu'ils ont trouvés dans eux. Ces problèmes son exposées dans la section suivante.

5.4 Représentation Temps/Fréquence

Dans la représentation temps-fréquence on avait pu mettre aussi les représentations cartésiennes, puisque ils sont représentations des composants temps-fréquence. Mais c'est qu'on va appeler représentation temps-fréquence dans cette section est la représentation où on peut parcourir la droite du temps et la droite de la fréquence.

Dans ce type de représentation normalement on représente des signaux mono-canal. Les différents portions du domaine temps-fréquentiel sont représentées selon l'énergie qu'ont.

Il y a aussi plusieurs représentations en fonction du domaine. Par exemple, dans la figure 5.4 on a une représentation spectral selon la STFT, et dans la figure 5.5 on a la représentation dans le plane du phase de la base de paquets de cosinus localisées. Toutes ces graphiques son *représentations mono*.

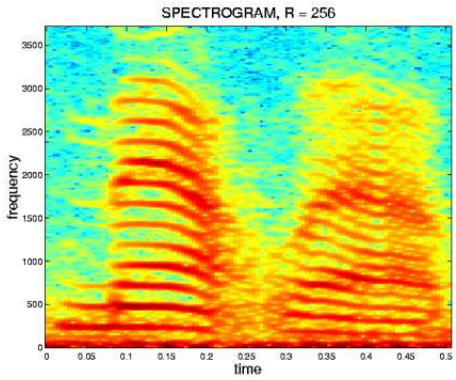

FIG. 5.4 – Représentation dans le domaine de Fourier : spectrogramme

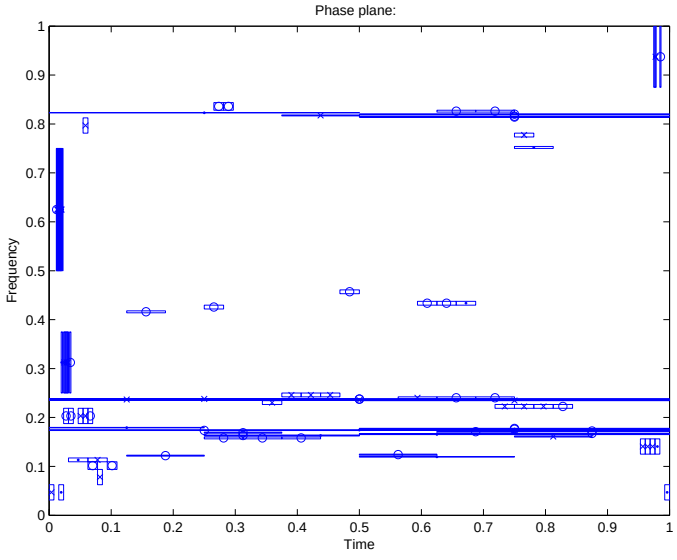

FIG. 5.5 – Représentation *mono* dans un base de *wavelets packets* de la phase du paquets

Avec l'intérêt centrée sur le paquets d'ondelettes, on va faire aussi une représentation de l'énergie des paquets du signal. Dans le plan temps-fréquence on verra en fonction de l'énergie du paquet un intensité du vert différent. Cette représentation nous permettra voir comme est distribue l'énergie des paquets dans le domaine temps-fréquence (voir figure 5.6).

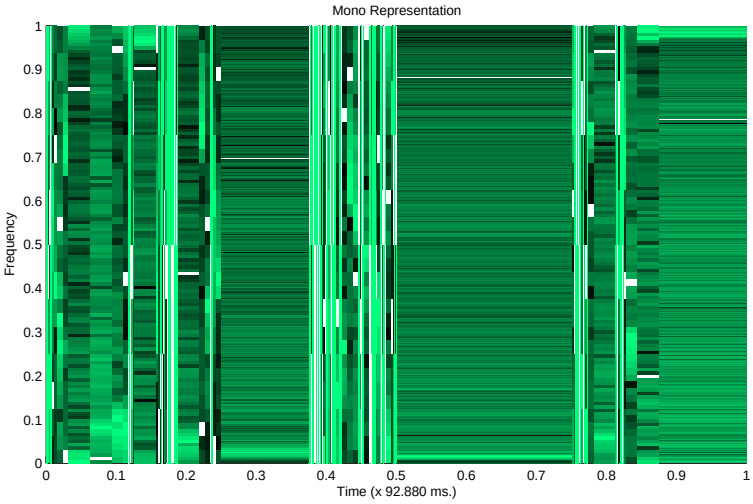

FIG. 5.6 – Représentation *mono* des paquets d'ondelettes

Si on veut voir un signal stéréo, on n'a que faire le même sur les deux signaux et on aurait deux représentations, une pour chaque signal.

D'autre part, ce qui nous intéresse maintenant c'est faire une représentation où on peut voir les deux sources au même temps. Mais on veut avoir de l'information stéréo ; cet à dire, on veut pas voir dans chaque composant l'addition des énergie des composants de la droite et de la gauche. C'est qu'on veut c'est représenter un peu le scène stéréo, avec le plus d'information possible.

De ces idées de représentation ce qu'on parte pour faire notre représentation stéréo.

En prenant les caractéristiques de la représentation de la figure 5.6, et les idées plantées dans ce chapitre sur les couleurs pour le signal stéréo, on arrive à la représentation de la figure 5.7

Dans cette représentation on peut voir les changements de couleur selon les propriétés des composants. On peut voir d'où vient l'énergie des sources qui sont dans chaque instant. Et on peut différencier des régions temporales où le changement du ton est claire. Pour le déplacement d'un source sur le rang angulaire (de gauche au droite) on est utilisé dans l'exemple de la figure 5.7 un codage de couleur qu'on appelle «rainbow». Cela est le codage qui utilise par défaut notre algorithme de représentation.

La premier idée sur la représentation étais changer de rouge à vert sans mettre du bleu. Le codage avec deux couleurs, une pour chaque côte, semble pas trop compréhensible puis que la différence entre les verts (ou rouges) du même côté de la scène n'est pas évident. Pour ce raison on est mis une troisième couleur. L'idée maintenant est avoir plus de différence de ton pour le même incrément d'angle. Le premier codage qu'on essai est un «rgb» qui va du rouge

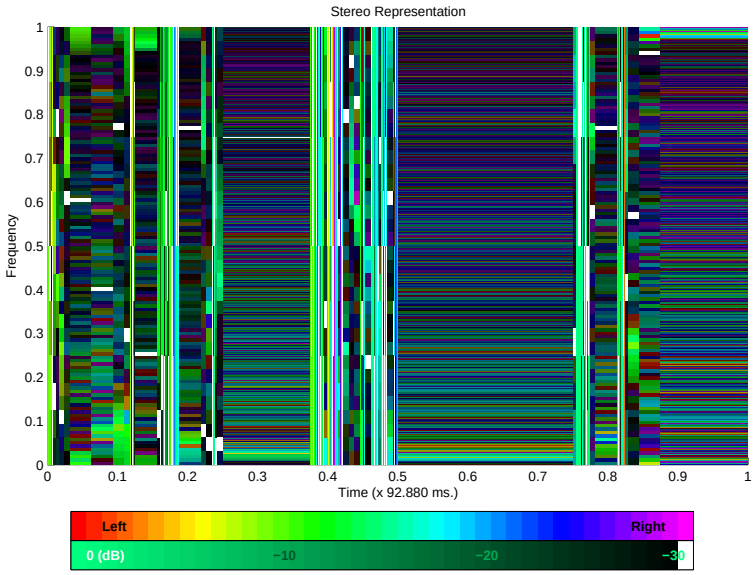

FIG. 5.7 – Représentation stéréo des *wavelets packets*

jusqu'au bleu en utilisant le vert pour le centre. Le problème maintenant est que il y a trop de tons dans notre représentation, et l'interprétation d'où se trouve la source n'est pas intuitif. Pour régler ça, on fait un deuxième essai avec trois couleurs. On cherche un motif qui soit intuitif et facile d'interpréter. On propose de mettre des variations de couleurs froids pour un côté et des couleurs chauds pour l'autre. Finalement les résultats sont très similaires au ces-ci de «rgb».

Après voir les différents résultats qu'on a avec les plusieurs légendes qu'on a vu au section 5.3 de la couleur on doit dire que les résultats, bien que il y a des choses visibles, c'est pas trop claire.

Pour cette raison on veut essayer un dernier chose, une *représentation stéréo partial*. En partant de la représentation stéréo qu'on a vu tout à l'heure on va prendre du rang des angles, qui va de la gauche a la droite, seulement le morceaux qui correspond à un angle θ donné et un accroissement $\Delta\theta = 2\epsilon$ (figure 5.8).

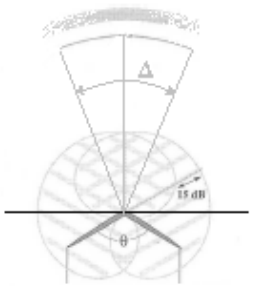

FIG. 5.8 – Rang $[\theta - \epsilon, \theta + \epsilon]$ de la *représentation partial*

Avec ce type de visualisation on peut voir l'énergie qu'il y a dans chaque composant pour un morceaux de la scène stéréo.

Cette fois le déplacement angulaire est représentée pur la *disparition* de la couleur sur le fond, et l'énergie qu'on a est la variation du ton selon un légende de rouge à bleu. Dans la figure 5.9 on peut voir les caractéristiques de ce type de représentation.

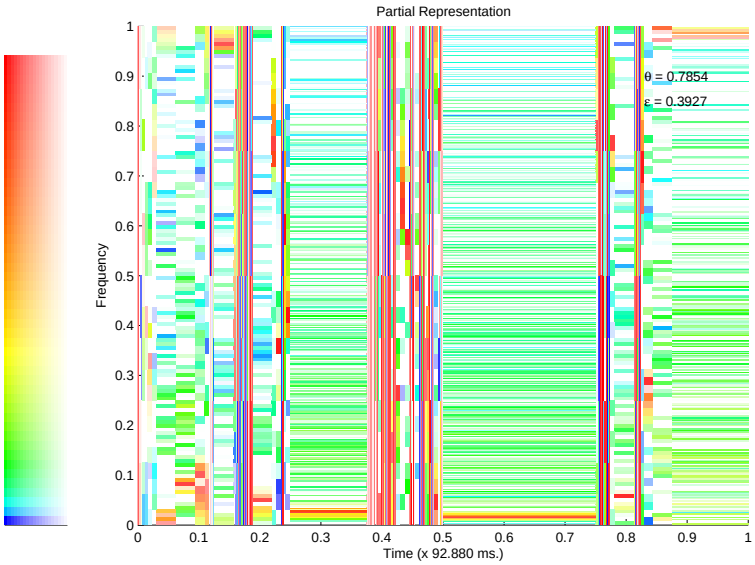

FIG. 5.9 – Représentation *partial* d'un signal stéréo

Chapitre 6
Le logiciel

Le contenu de cette chapitre parle du logiciel crée à partir de l'idée de la représentation stéréo. Les concepts de la représentation ont étés déjà développés dans le chapitre 5. Il y a une relation spécial avec la section 5.4 puisque ce qu'on veut faire est directement en relation avec la *représentation partial*.

6.1 Introduction

L'objectif du logiciel es créer une représentation du son stéréo dans le domaine transformée qui ait en compte l'angle et l'énergie du chaque composant. Après les résultats de la figure 5.7 (page 61) on peut dire qu'on peut pas faire un représentation du son stéréo de la façon qu'on voulait au début. La solution à cette problème peut être profiter le méthode de représentation pour faire la représentation partial de la scène stéréo ; cet à dire, on va faire un graphique temps-fréquence où on va représenter un parti des composants du signal, ces qui sont dans le rang d'angle $[\theta - \epsilon, \theta + \epsilon]$.

Le logiciel qu'on veut créer doit prendre cet methode de représentation et faire un environnement graphique qui permit au utilisateur choisir l'angle θ et l'incrément ϵ sur ce qu'on va faire les représentations. Avec l'information des plusieurs graphiques pur les différents angles sur les qu'on cherche, on peut voir comme est partagé le scène et on peut voir où se trouvent les sources.

Pour créer le logiciel on commence pour Matlab, où la part de traitement du signal sera plutôt simple par rapport à l'application Java. On ferait cet application Java quand on ait clair le structure de ce qu'on veut comme produit final.

Vraiment l'interface qu'on peut faire pour Matlab ne serait pas un logiciel pur lui même, puisque est nécessaire avoir Matlab pour exécuter les *points m*. L'idée est construire une application indépendante qui puisse être exécuté n'importe où. Ce cette idée là que nous pousse à utiliser Java.

6.2 Le développement sur Matlab

En continuant avec les expériences de la *représentation partial*, on veut faire un *m-file* où on puisse visualiser plus clairement qu'est qu'on fait pendant la recherche de sources par le scène stéréo.

L'algorithme de la représentation partial a en compte seulement un partie du rang angulaire. Il fait déjà un graphique où l'énergie suit une variation de ton et l'angle parait disparaître jusqu'à sortir du rang.

Avec l'algorithme déjà fait on dois seulement faire un interface avec l'utilisateur pour faire plus agréable la recherche d'information dans les bases de d'ondelettes ou de cosinus localisées.

Le schéma de l'algorithme est :

1. prendre l'arbre et les paquets de gauche et droite qui correspond à la meilleure base B_x,
2. prendre les valeurs de θ et ϵ,
3. faire le masquage et les calcules pour la représentation et
4. montrer la figure.

Maintenant on va ajouter des passes au esqueme.

Le premier est inclure l'algorithme de transformation pour que l'utilisateur puisse travailler directement sur les signaux temporels. Donc dans l'entrée on a le signal $\mathbf{x}$ et puis on prends les paquets dans la meilleure base. Pour avoir le résultat du masquage qu'on fait avec le rang angulaire, on fait aussi un signal monocanal qui est la sortie de notre application.

Entre le pas du choisir les sources d'entrée et le pas du calcule on mets le choix de θ et ϵ. Le choix vient donné pour des barres glissantes qui peuvent changer la direction et la taille de l'incrément très pratiquement.

Donc le nouveau schéma serait :

1. prendre les signaux d'entrée,
2. choisir les valeurs de θ et ϵ,
3. faire le masquage et les calcules,
4. montrer la figure,
5. faire le signal de sortie.

L'environnement graphique sur Matlab de cet application est divisé dans quatre parts (voir figure 6.1) qui se correspond au schéma de l'application.

L'entrée des donnés, où on peut voir avec deux graphiques le signal stéréo d'entrée. Cet partie est situé en haut de la fenêtre.

La sortie avec le rang angulaire qu'on est choisi avec un autre graphique. Ce signal de sortie varie selon le traitement qu'on le fasse au signal stéréo. Cette partie es situé just'en bais.

La représentation des paquets d'ondicules qui forment notre graphique principal dans l'application. Elle est situé au centre de la fenêtre en indiçant que c'est le pas intermédiaire et elle prends grand part de l'espace qu'il y a pour être vraiment l'objectif de l'application.

Les contrôles interactives lesquels l'utilisateur doit utilisé et qui sont situées à la droite de la fenêtre. Dans les contrôles il y a deux parts aussi, le contrôle du rang angulaire et le contrôle des exécution du programme. Le contrôle du rang se fait avec deux barres glissantes qui peuvent changer l'angle, sur lequel on va

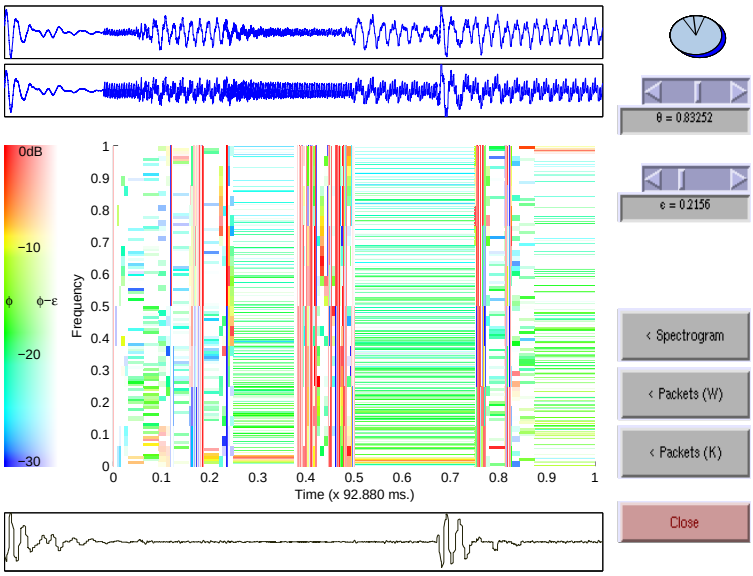

FIG. 6.1 – L'environnement graphique sur Matlab

diriger la masque, et l'intervalle ϵ , qui indique jusqu'à quelle distance angulaire on va considérer. Le contrôle sur les calcules se fait avec les boutons :

- Spectrogram qui montre le spectre du signal stéréo comme l'addition d'énergies du signal de gauche et droite,
- Packets(W) qui fait notre représentation partial du signal par rapport à les variables θ et ϵ qui sont mis avec les contrôles des angles,
- Packets(K) qui fait tout à fait le même que le bouton antérieur sauf qui fait la représentation sur un fond noire (bla**k**), et
- Close qui ferme l'application.

6.3 L'application JAVA

Pour avoir une application qui peut être utilisée pour la plus part du monde on est choisi le langage JAVA.

L'application qu'on est fait a les mêmes caractéristiques que cela de Matlab avec des petites modifications. Par la suite on verra un petit description de l'interface graphique et son mode d'emploi.

6.3.1 Introduction

Les objetives de cette application sont faire la *représentation partial* d'un signal stéréo et la reconstruction du signal àpres le raitement. Pour faire ça on a besoin de :

1. les signaux d'entrée,
2. l'interface pour le reglage des parametres du traitement et
3. les outils mathematics pour le traitement.

Par rapport aux signaux d'entrée, comme maintenant on n'est pas dans un environnement mathematicale comme Matlab, seront pris des fichiers «.wav» et ils seront faites streams lesibles pour java. Pour travailler avec les sounds sur java, on fait de l'API Java Sound.

L'inteface java peut être fait dans deux domaines, l'applicaition java et les applets. La principale différence est que les applets peuvent être ouverts d'un page web directement. On n'a pas besoin de telecharger le logiciel et faire l'instalation. Par contre, ils manquent quelque fonction par rapport aux applications java. La plus important pour nous est que n'est pas possible traiter fichiers avec les applets. Pour l'instant cette caracteristique est imprescindible pour prendre les fichiers .wav qu'on va traiter. Donc notre logiciel est una application java. Pour faire un interface graphique on a fait use du paquet *AWT* de java. On peut trouver la documentation sur java.awt dans [«java.awt»]

En ce qui concerne les outils mathematiques pour travailler avec la segnal, soit on fait usage de differents java files qui sont crées epecialement pour l'application, soit on les prends de la web.

6.3.2 Les streams

Les signaux de son dans Java peuvent être traitées pour l'API JAVA SOUND. On peut trouver la guide pour le programateur sur [«sound guide»].

Le scheme de travail avec les donnés de son est :

1. Du fichier de son, on fait un stream AudioInputStream, qui vient de la class InputStream. On preds aussi le format du fichier d'audio dans un AudioFileFormat.
2. Avec l'information du format, on peut faire un buffer de bytes dont sera utilisé pour lire du stream,
3. ce buffer stockera les morceaux d'information qui vient de l'AudioInputStream dans un ByteArrayOutputStream, et
4. une fois qu'on a lu touts les donnés on peut prendre directement le réreau de bytes du ByteArrayOutputStream.
5. en ayant compte le format du fichier : bits par échantillon, nombre de canaux et le codage, on peut transformé le transformée à floats pour travailler.
6. Pour retourner au fichier on doit creer de nouveau le réseau de bytes en ayant compte le format.
7. Ensuite, on metra le réseau dans un stream ByteArrayInputStream,
8. cela, avec l'information du format d'audio, on le metra dans un AudioInputStream,
9. qui est le chargé de remplir le fichier d'audio en faisant un nouveau fichier du format qui convienne.

file ⇒ **Audio**InputStream ⇒ **ByteArray**OutputStream ⇒ byte ⇒ **float**
file⇐ **Audio**InpoutStream ⇐ **ByteArray**InputSrteam ⇐ byte ⇐ **float**
Le traitement de signal se ferait sur les donnes en forme de floats.

6.3.3 L'environnement graphique

L'interface traitement-utilisteur a le même scheme que cela du Matlab. En plus on doit travailler avec des fichiers et on fait plus attencion au partie de controls pour faire plus detallé le traitement.

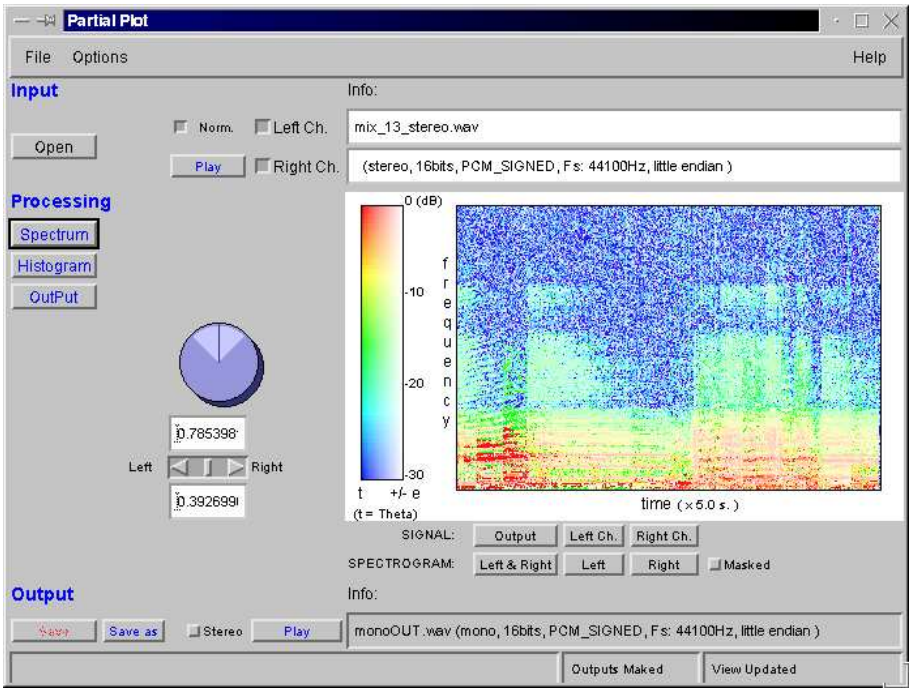

FIG. 6.2 – L'environnement graphique sur Java

On va diviser cette fois l'interface en trois parts.

Input est la premeire partie du processus et de l'interface. Pour le processus on doit faire la conversion *fichier-signal.* donc on dois ouvrir un fichier d'audio avec quelque fenêtre de dialogue.

On a deux possibilités pour prendre notre signal stéréo, soit on prend deux signal mono, soit on prend directament un signal stéréo. Pour cette raison on a deux lignes dans cette partie de l'interface, et un *checkbox* associé pour chaque une. Dans les lignes on afiche le nom de fichier et quelque caracteristique du format d'audio.

Les deux events associés à la partie d'input sont ouvrir et jouer. En dependant des *checkbox* qui soient actives on fait l'action sur un ligne ou sur tout le deux ; cet à dire, on ouvre ou joue soit un signal mono, soit un signal stéréo.

Quand on prend un signal stéréo on mets chaque partie dans la ligne qui correspond.

Un dernier element reste pour être commenté, le *checkbox Norm.* Avec lui on di si on veut écouter le son normalicée ou bien le leisser comme il vient du fichier.

Pour commencer le process il faut d'abort avoir un signal mono sur chaque ligne.

Processing ou partie de traitement, il prend la plus part de la fenêtre. Il est situé au milieu et il est divisé au deux parties.

Le graphique de répresentation, qui est situé à droite, montre les diférents graphiques de qu'on a besoin pour suivre le traitement.

Pour controler le changements de graphiques on a sis boutons de control. La premier ligne de trois boutons sont pour montrer les signaux d'entre et de sortie. Les trois boutons d'en bais montrent les signaux, mais dans le domaine temps-fréquence ; cet à dire, les spectrogrammes des signaux. Il faut noter que on peut voir les différents représentations seulement quan ils sont disponibles. Si on a pas fait le signal de sortie on n'aurait pas le droit d'utiliser le bouton correspondant.

Un autre checkbox aparait cette fois acoté des boutons des spectrogrammes. Avec lui on dis si on veut voir l'espectre du signal *Masqued* (masqué) ou pas. L'option est seulement valide pour eux parce que notre représentation partial touche seulement au graphiques temps-fréquence.

D'autre côté on a la graphique même. Les graphiques temps frequence ils sont fait à partir de l'idée de la représentation partial (section 5.4) et de la STFT (section 2.2.3).

Les graphiques des signals dans le domaine du temps font usage des sets de paquets de Java Ptolemy II, disponible sur [PtolemyII].

Les controls de tout le logiciel se font d'ici. On a un serie de trois boutons qui suivent le processus de l'algorithme :

1. décomponer le signal dans le domaine transformée,
2. faire le masquage,
3. faire la reconstruction des estimations.

Vraiment la premier partie est faite moyent au premier bouton, et les deux suivents sont faites moyent le dernier.

Les boutons sont :

1. Spectrum, qui fait la transformation des signaux au domaine transformée. Il est le première boutons à être active après qu'on a les signaux gauche et droite. Quand on fait la transformation les autres deux boutons devient actives.
2. Histogram, qui fait un histogramme comme celui qui a été proposé dans la section 4.3.5. L'idée est voir l'information de où sont les angles du scène avec la probabilité la plus grande de contenir les sources. Après les voir on peut faire le choix des angles pour le processus de masquage. La position du bouton est le deuxième parce que même s'il ne fait des calcules il nous donne l'information pour faire le masquage. Ce bouton n'est pas active jusqu'à on n'a pas fait la transformation.
3. OutPut, destinée a prendre l'information des angles pour faire le masquage sur les signaux transformés et faire la reconstruction. Après cela on a la signal de sortie et la troisième partie de la fenêtre activée.

Entre les boutons et les graphiques on a dee autre elements de control. Il sont destinées a choisir l'angle θ et l'increment 2ϵ en fonction de deux barres glissantes qui nous donnent le valeur de θ et ϵ. Pour éclaircir le rang angulaire où on se trouve il y un petite figure en forme de bouton *pan*.

Output est finalement la signal de sortie. Une fois qu'on est fait la reconstruction, on arrive au troisième parti de l'environnement. Ici on a le même schéma que sur *Input* mais on a un seule canal. Même si on peut disposer de la signal de sortie en mono ou en stéréo, avec le *checkbox Stéréo* situé accoté du bouton *Play*, on voit tout l'information sur la même ligne. Le raison est que on a qu'un signal à enregistrer. Les boutons *Save* et *Save as* sont qui font ce travail. Dans la ligne d'*Info*rmation on a cela relative à la signal enregistré tout a l'heure, soit stéréo, soit mono.

Dehors des ces parties de l'environnement on a aussi deux éléments caractéristiques des fenêtres : la barre de menu, et la barre d'état.

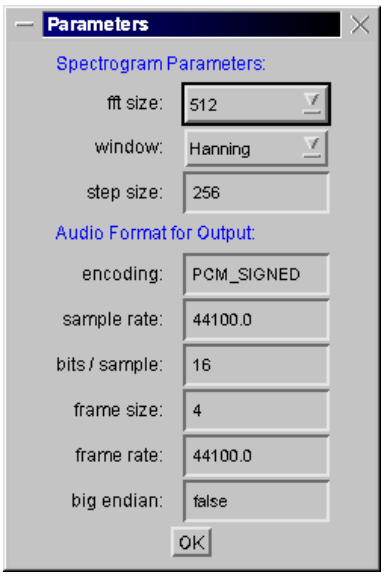

FIG. 6.3 – fenêtre avec l'information des paramètres du spectrogramme et du format d'audio

Dans la barre de menu il apparaît les options :
- *File*, d'où on peut voir le format d'audio d'un fichier .wav (voir figure 6.4) avec l'option *Audio Format ...*, et sortir de l'application avec *Exit*.
- *Options*, que par moyen de *Parameters* nous montre avec un autre fenêtre les paramètres du spectrogramme et du format d'audio à la sortie (voir figure 6.3).
- *Help*, la classique option d'aide que nous parlerait de l'application moyen à *About Partial Plot*.

De la fenêtre de parametres on peut choisir les parametres qui correspond à la transformée de Fourier :
- la taille de la fenêtre de la STFT (section 2.2.3) d'un groupe de tailles puissances de deux, en ayant 512 comme predefinie,
- la taille du pas dont on fait le chaque STFT, que sera la motié de la taille de la fenêtre en principe.
- le type de fenêtre qu'on peut outilisser à choix :

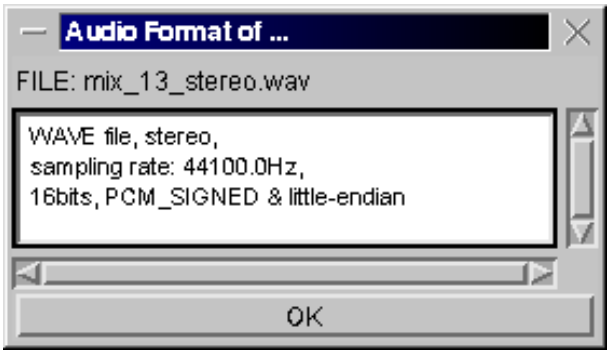

FIG. 6.4 – fenêtre avec l'information du foramt d'audio

- Rectangular
- Welch
- Barlett
- Hanning
- Hamming

De l'autre coté, just'en bas, on a la barre d'état, qui est divisé en trois boîtes. Dans la premier on a l'état du processus qui est en marche. Dans la deuxième indiquée quel partie du processus est déjà fait. Après faire la signal de sortie on a «Output Maked» et ça veut dire qu'on a déjà notre signal de sortie. Pour finir, dans le dernier coin de la fenêtre on a une petite indication du processus de représentation, s'il est en course ou il est actualisé.

6.3.4 Le paquage

Dans la tête du texte du fichier principal de l'application on peut lire :

```
package pkt_aplication;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;
import java.io.*;
import javax.sound.sampled.*;
import ptolemy.plot.*;
```

La liste de fichiers qui forment *pkt_aplication* sont montrés dans l'anexe B.

En plus, on a fair usage de paquets pour la représentation de graphiques **Ptolomy II** disponible sur [PtolemyII].

De la liste de fichiers, Complejo.java est disponible sur [Franco,2004]. Et une partie des *classes : FastFourierTransform.class, FFTProcess.class* et *Windowing.class* sont inspirées en [Steeb] ainsi que aux forums [«Developer Forums»,2004].

6.3.5 Travail à faire

Au travaille de la consturction de l'application il reste encore quelques points sans finir. Il sont :

- faire apparaître l'histogramme des angles des composants de la signal pour aider avec le choix de l'anglede masquage.

- faire apparaître l'aide sur l'application de bouton qui correspond dans le menu.
- Optimiser l'algorithme de la STFT utilisée pour faire le spectrogramme. La function qu'il y a maintenant a été developpe à partir de l'algorithme pour C++ disponible au [Steeb, ch. 5]. Il mets quelque bruit au signal que dois être effacé.
- Mettre les parametres de l'option *parameters* dans l'algorithme. C'est-à-dire, faire usage des options de choix pour le type et le longueur de la fenêtre.

Conclusions

Conclusions

Conclusion

Après la présentation de l'algorithme B2S2 pour la séparation de sources et les expériences réalises sur lui, nous pouvons dire que l'approche d'un seule source active pour chaque morceaux temps-fréquence est valable.

Nous avons vu aussi qu'il y a meilleurs résultats en ayant des bonnes masques binaires qu'en ayant meilleures bases.

Ensuite nous avons cherché des paramètres pour faire un algorithme de *clustering* pour la séparation en aveugle. Dans ce sens, nous avons présenté un méthode avec l'angle et l'énergie des composants.

Par rapport à la représentation stéréo nous avons vu que le codage présenté est imprécis, mais nous avons profité la même idée pour faire un système de représentation partial que nous délimite le scène stéréo dans un rang angulaire.

Finalement nous avons faite une application java pour la représentation des signaux stéréo qui permets la séparation de sources en faisant un masquage angulaire.

Perspectives

La première suite à ce stage est la comparaison des plusieurs méthodes de clustering pour trouver finalement un algorithme de séparation en aveugle.

Un autre travail en parallèle est faire un étude de la prise des résultats de l'histogramme présenté comme premier outil de clustering. Ainsi comme l'étude du masquage non-binaire en ayant en compte deux sources actives.

Par rapport à l'application crée, il faut épurer l'algorithme de STFT pour java, permettre la total paramétrisation des algorithmes et inclure autres outils de séparation.

Annexes

Annexe A

Les têtes des fichiers du paquet B2S2

A.1 B2S2.m

```
   B2S2 : Best Basis Source Separation
   -------------------------
   version 1.0
   Source separation based on
   R. Gribonval, "Piecewise linear separation"
   to appear in Proc. SPIE 03, San Diego, CA, USA.
   Usage :
   S = B2S2(X,A)
   S = B2S2(X,A,rep)
   S = B2S2(X,A,rep,'CP')
   S = B2S2(X,A,rep,'CP',bell)
   S = B2S2(X,A,rep,'CP',bell,ent)
   S = B2S2(X,A,rep,'WP')
   S = B2S2(X,A,rep,'WP',mf,ent)
   Inputs :
   -X : M x T matrix containing the M observed linear signals of length
T
   (rows represent signals) ;
   -A : M x N (estimated) mixing matrix corresponding to the model
X=AS
   -rep : It indicates that type of representation we are going to
use :
   'E' : to (E)stimated Mono Signal (Default)
   'M' : to (M)ultichannel Images
   Outputs :
   -S : matrix containing the estimated sources.
   S : [N x T] for the Mono Signals
   S : [M x N x T] for the Multichannel Images
   See also :
   related functions in B2S2 (package for Best Basis Source Separa-
tion) :
   MBestBasisAnalysis, BinaryMasking, MaskDecomp,
```

```
MBestBasisSynthesis, SeparationMatrices.
```

A.2 BinaryMasks.m

```
BinaryMasks :
-----------
Version : 1.0
Description :
Computes a Binary Mask that esteem that source is the active one.
Usage :
mask = BinaryMasks(A,Y)
Inputs :
-A : M x N (estimated) mixing matrix corresponding to the model
X=AS
-Y : M x T matrix containing the coefficients of the M signals in
the best basis
Outputs :
-mask : N x J matrix containing the activity of the N sources on
the J components
Reference : R. Gribonval, "Piecewise linear separation" to appear
in Proc. SPIE 03, San Diego, CA, USA.
```

A.3 MaskDecomp.m

```
MaskDecomp :
----------
Version : 1.0
Description :
Computes the estimated components of the images of the sources on
each sensor. Usage :
mask = BinaryMasks(A,Y)
Inputs :
-Y : M x J matrix containing the J components of the M decomposed
oberved linear signals
-mask : N x J matrix containing the activity of the N sources on
the J components
Outputs :
-Ymasked : M x J x N matrix containing the estimated components
of the images of the N sources on M sensors.
Ymasked( :, :,n) is an M x J matrix containing the estimated com-
ponents of the image of the source 'n' on the sensors.
Reference : R. Gribonval, "Piecewise linear separation" to appear
in Proc. SPIE 03, San Diego, CA, USA.
```

A.4 MBestBasisAnalysis

```
MBestBasisAnalysis
-------------
```

```
   Description :
   Computes a best basis for simultaneous approximation of several
channels.
   Usage :
   [Y,type,shape,btree,D]=MBestBasisAnalysis(X)
   [Y,type,shape,btree,D]=MBestBasisAnalysis(X,'CP')
   [Y,type,shape,btree,D]=MBestBasisAnalysis(X,'CP',bell)
   [Y,type,shape,btree,D]=MBestBasisAnalysis(X,'CP',bell,ent)
   [Y,type,shape,btree,D]=MBestBasisAnalysis(X,'WP')
   [Y,type,shape,btree,D]=MBestBasisAnalysis(X,'WP',qmf)
   [Y,type,shape,btree,D]=MBestBasisAnalysis(X,'WP',qmf,ent)
   Inputs :
   X : M x T matrix containing the M observed signals of length T rows
represent signals whose length has to be a power of two (T=2^n) ;
   type : either 'CP' (cosine packets) or 'WP' (wavelet packets), de-
faults to 'WP'
   bell : name of bell to use for cosine packets, defaults to 'Sine'
(see also CPAnalysis)
   qmf : orthonormal quadrature mirror filter to use for wavelet pack-
ets, defaults to 'Vaidyanathan' (see also WPAnalysis)
   ent : type of entropy to use, defaults to 'Entropy' (see also Calc-
StatTree)
   Outputs :
   Y : M x T matrix containing the coefficients of the M signals in
the best basis
   type : either 'CP' (cosine packets) or 'WP' (wavelet packets)
   shape : the qmf or bell used for wave/cosine packets decomposi-
tion
   btree : basis-tree of the best basis (see function BestBasis)
   D : maximum depth of tree-search, i.e. degree of finest frequency
partition (wavelet packets) or depth of finest time splitting (co-
sine packets)
   Version : 1.0
   See also :
   related functions in Wavelab : WPAnalysis, MakeONFilter, CPAnal-
ysis, CalcStatTree, BestBasis related functions in 3B2S (package for
Best Basis Blind Source Separation) : MBestBasisSynthesis
   Reference : R. Gribonval, "Piecewise linear separation" to appear
in Proc. SPIE 03, San Diego, CA, USA.
   Copyright (C) 26 May 2003 R.GRIBONVAL
   Inquiries, bug report : remi.gribonval@irisa.fr
   This program is free software ; you can redistribute it and/or mod-
ify
   it under the terms of the GNU General Public License as published
by
   the Free Software Foundation ; either version 2 of the License, or
   (at your option) any later version.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY ; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

```
   GNU General Public License for more details.
   You should have received a copy of the GNU General Public License
   along with this program ; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-
1307 USA
```

A.5 MBestBasisSynthesis

```
   MBestBasisSynthesis
   --------------
   Description :
   Reconstructs several channels from their simultaneous best basis
coefficients.
   Usage :
   X = MBestBasisSynthesis(Y,type,shape,btree,D)
   Inputs :
   Y : M x T matrix containing the coefficients of the M signals in
the best basis
   type : either 'CP' (cosine packets) or 'WP' (wavelet packets)
   shape : the qmf or bell used for wave/cosine packets decomposi-
tion
   btree : basis-tree of the best basis (see function BestBasis)
   D : maximum depth of tree-search, i.e. degree of finest frequency
partition (wavelet packets) or depth of finest time splitting (co-
sine packets)
   Outputs :
   X : M x T matrix containing the M reconstructed signals of length
T
   Example :
   qmf = MakeONFilter('Vaidyanathan') ;
   [Y,type,shape,btree,D] = MBestBasisAnalysis(X,'WP',qmf) ;
   Y = Y.*(Y>0.1) ;
   Xe = MBestBasisSynthesis(Y,type,shape,btree,D) ;
   Version : 1.0
   See also :
   related functions in 3B2S (package for Best Basis Blind Source Sep-
aration) : MBestBasisAnalysis
   Reference : R. Gribonval, "Piecewise linear separation" to appear
in Proc. SPIE 03, San Diego, CA, USA.
   Copyright (C) 26 May 2003 R.GRIBONVAL
   Inquiries, bug report : remi.gribonval@irisa.fr
   This program is free software ; you can redistribute it and/or mod-
ify
   it under the terms of the GNU General Public License as published
by
   the Free Software Foundation ; either version 2 of the License, or
   (at your option) any later version.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY ; without even the implied warranty of
```

```
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
   GNU General Public License for more details.
   You should have received a copy of the GNU General Public License
   along with this program ; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-
1307 USA
```

A.6 SeparationMatrices

```
   SeparationMatrices :
   ----------------
   Version : 1.0
   Description :
   Computes the separation matrix of each moment (for each component
j)
   Usage :
   B = SeparationMatrices(A,mask)
   Inputs :
   -A : (estimate of) the M x N mixing matrix
   -mask : N x J matrix containing the activity of the N sources on
J components
   Outputs :
   -B : N x M x J matrix containing the separation matrix for each
of the J components
   B( :, :,j) is the N x M separation matrix Bj such that YSj = Bj
YXj is the vector of the N components of the estimated sources for
index j It is computed using the pseudo-inverse of A restricted to
the 'active' columns
   Reference : R. Gribonval, "Piecewise linear separation" to appear
in Proc. SPIE 03, San Diego, CA, USA.
```

Annexe B

Les fichiers du paquet de l'application Java

Les fichiers du paquet sont :

Adjust.class	FFTProcessNEW.java
AudioDataBuffer.class	legenda.class
AudioDataBuffer.java*	legenda.java
CaraPAN.class	Listen2File.class
CaraPAN.java*	OyenteMenu.class
Check.class	OyenteMenu.java*
Checking.class	PartialPlot.class
CloseERROR.class	PartialPlot.java*
Complejo.class	PartialPlot.old*
Complejo.java	Playing.class
Dhillable1.class	Ploting.class
Dhillable.class	Spectr.class
DotWavFilter.class	Spectr.java
escuchaChoice.class	StreamIn.class
ExcepcionDivideCero.class	StreamOut.class
FastFourierTransform.class	Streams.class
FFTProcess.class	TwoBytes2Short.class
FFTProcess.java*	Windowing.class

Références

Table des figures

Liste des tableaux

Bibliographie

[Lim & Oppenheim,1988] Lim, J.S. & Oppenheim, A.V. (1988). Avanced Topics in Signal Processing. New Jersey : Prentice Hall Signal Processing Series.

[Mallat,1999] Mallat, S (1999). A wavelet tour of signal processing.(2ème ed.) London :Academic Press.

[Chui,1992, vol.2] Chui, Charles K. (1992). Wavelet analysis and its applications Volume 2. Wavelets : a Tutorial in Theory and Applications. San Diego :Academic Press.

[Chui,1992, vol.1] Chui, Charles K. (1992). Wavelet analysis and its applications Volume 1. An Introduction to Wavelets. San Diego :Academic Press.

[Reulos,2003] Reulos, T. (2003). Mise en æuvre d'algorithmes pour la séparation de sources. Rapport de stage effectué à l'IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires) au sein de l'équipe METISS (Modelisation et Expérimentations pour le Traitement des Inmations et des Signaux Sonores).

[Coifman & Wickerhauser] Coifman, R.R & Wickerhauser, M.V. Entropy-Based Algorthms For Best Basis Selection. Yale University, New Haven, Connecticut, USA.

[Gribonval,2003] Gribonval, R. (2003). Piecewise linear source separation. To appear in Proc. SPIE 03, San Diego, CA, USA.

[Buckheit et al. & Scargle, J.,1995] Buckheit, J., Chen, S., Donoho, D., Joshn-Stone, I. & Scargle, J.(december, 1995) WaveLab Reference Manual.(Version 0.700) Stanford University & Nasa-Armes Research Center.

[Hernández,2003] Hernández, E. Ondículas y tecnología. Boletin de la Sociedad Española de Matemática Aplicada, no. 25, (2003), 39-54.

[labTNSI] Laboratorio de Tratamiento Numérico de la Señal y de la Imagen (Département de Mathématiques, Universidad de Oviedo), au mai 2004. http ://coco.ccu.uniovi.es

[restauration] Proceso de señales y aplicaciones en la restauración de registros sonoros (du labTNSI, Universidad de Oviedo), au mai 2004. http ://coco.ccu.uniovi.es/brahms/sumario.htm

[WaveLab] La page principal de WaveLab au mai, 2004. http ://www-stat.stanford.edu/~wavelab/

[Bass-dB] La page du Blind Audio Source Separation evaluation database. E. Vincent, R. Gribonval and C. Févotte. Funded by GdR ISIS (CNRS,France), au mai 2004. http ://www.irisa.fr/metiss/BASS-dB

[«java.awt»] Package java.awt, Contains all of the classes for creating user interfaces and for painting graphics and images. Copyright 2003, Sun Microsystems Inc. All rights reserved, au june 2004. http ://java.sun.com/j2se/1.3/docs/api/java/awt/package-summary.html

[«sound guide»] Java Sound API Programmer's Guide,© 2000, Sun Microsystems Inc. All rights reserved, au june 2004. http ://java.sun.com/j2se/1.3/docs/guide/sound/prog_guide/title.fm.html

[PtolemyII] Ptolemy II is a set of Java packages supporting heterogeneous, concurrent modeling and design. Department of EECS, UC Berkeley, au june 2004. http ://ptolemy.eecs.berkeley.edu/ptolemyII/index.htm

[Steeb] Mathematical Tools in Signal Processing with C++ and Java Simulation. International School for Scientific Computing.

[Franco,2004] Procedimientos numéricos en lenguaje Java, Ángel Franco García -1998-2004. Universidad del Pais Vasco au june 2004. http ://www.sc.ehu.es/sbweb/fisica/ curso-Java/numerico/complejo/codigo/Complejo.java

[«Developer Forums»,2004] Developer Forums, Java Technology Forums, Sun Microsistems, au june 2004. http ://forum.java.sun.com/index.jsp