Extraction de descripteurs musicaux: une approche évolutionniste

Thèse de Doctorat de l'Université Paris 6
Spécialité: Informatique

Aymeric ZILS
Sony CSL Paris, 6 rue Amyot 75005 Paris
Laboratoire d'Informatique de Paris 6, 4 Place Jussieu 75005 Paris

Soutenue le
30 septembre 2004

devant le jury composé de

Directeur de thèse entreprise:	*François PACHET*
Directeur de thèse université:	*Patrick GALLINARI*
Rapporteurs:	*Myriam DESAINTE-CATHERINE* *Gaël RICHARD*
Examinateurs:	*Yann ORLAREY* *Xavier RODET* *Thierry ARTIERES*

TABLE DES MATIÈRES

1. Introduction - Objectifs & méthodologie 1
1.1 Introduction 1
1.1.1 L'essor de l'informatique musicale 1
1.1.2 Différents types de descripteurs musicaux 2
Descripteurs acoustiques et culturels 2
Niveaux de description 2
Descripteurs locaux et globaux 2
Descripteurs unaires 3
Subjectivité des descripteurs 3
1.1.3 Extraction et utilisation des descripteurs 4
Etude perceptive des descripteurs 4
Extraction des descripteurs à partir du signal acoustique . 5
Intégration des descripteurs dans des applications musicales 6
1.2 Plan du document 6

Partie 1
Extraction automatique de descripteurs par traitement du signal 11

2. Etat de l'art de l'analyse de signaux musicaux 11
2.1 Évaluation de propriétés sonores 12
2.1.1 Extraction de mélodie 12
2.1.2 Extraction de tempo 13
2.1.3 Evaluation de l'énergie musicale 14
2.2 Identification et classification sonore 14
2.2.1 Classification de sons instrumentaux 15
2.2.2 Caractérisation perceptive des sons 16
2.2.3 Genre et Timbre global d'un extrait musical 17
2.2.4 Reconnaissance des émotions musicales 18
2.3 Recherches complémentaires à la description de signaux musicaux 19
2.3.1 Segmentation et analyse de flux audio 19
2.3.2 Systèmes d'écoute musicale 20
2.4 Conclusion 20

3. Construction empirique de descripteurs musicaux 23
3.1 Extraction du rythme percussif 23
3.1.1 Extraction d'un son percussif caractéristique 24
3.1.2 Extraction de « Drum track » 26
3.1.3 Performances 26
3.2 Modélisation de l'énergie musicale 27

3.2.1 Évaluation par tests perceptifs . . . 27
3.2.2 Modélisation des résultats des tests perceptifs . . . 28
3.3 Construction de descripteurs basiques à partir de fonctions caractéristiques simples . . . 30
3.3.1 Percussivité d'un son polyphonique . . . 30
3.3.2 Détection de voix chantée . . . 31
3.4 Conclusion . . . 31

4. Le système EDS . . . 33
4.1 Présentation des principes généraux du système EDS . . . 34
4.1.1 Justification de l'extraction automatique de descripteurs . 34
Un problème simple : Extraction d'un sinus dans un bruit rose . . . 34
Limites des caractéristiques traditionnelles . . . 34
Ajout d'opérations spécifiques (post et pré-traitements) . 35
Choix des opérations additionnelles . . . 35
Paramétrage des opérations additionnelles . . . 35
Modèle final . . . 36
4.1.2 Intégration des principes dans le système EDS . . . 36
4.2 Définition de problèmes descriptifs pour EDS . . . 37
4.2.1 Types de problèmes descriptifs à résoudre . . . 38
4.2.2 Mode de soumission du problème descriptif . . . 39
4.3 Représentation des fonctions caractéristiques . . . 39
4.3.1 Décomposition en opérations de base . . . 39
4.3.2 Représentation sous forme d'arbre . . . 40
4.3.3 Types des éléments constitutifs d'une fonction . . . 40
4.3.4 Construction automatique de fonctions syntaxiquement correctes . . . 41
4.4 Typage des données . . . 42
4.4.1 Typage physique des données atomiques . . . 42
4.4.2 Typage physique de l'ensemble des données . . . 43
4.5 Typage des opérations . . . 44
4.5.1 Transformation du type de donnée lors d'une opération . 44
4.5.2 Opérateurs opérationnels et opérateurs fonctionnels . . . 46
4.5.3 Table de typage d'un opérateur . . . 46
4.6 Guidage de la construction automatique des fonctions . . . 49
4.6.1 Opérateurs génériques . . . 50
4.6.2 Structuration des fonctions par patterns . . . 51
4.6.3 Guidage par heuristiques . . . 53
Contrôle des opérations . . . 54
Contrôle de la structure . . . 55
Contrôle des valeurs des paramètres . . . 55
4.6.4 Choix des opérations suivant les types et heuristiques . . 55
4.6.5 Instantiation automatique de fonctions à partir de patterns et d'heuristiques . . . 57
4.7 Pertinence des fonctions caractéristiques . . . 58
4.7.1 Application d'une fonction sur un signal . . . 59
4.7.2 Calcul de la pertinence d'une fonction . . . 60
4.8 Recherche de fonctions caractéristiques optimales . . . 60
4.8.1 Algorithme de recherche génétique . . . 61

4.8.2 Création d'une nouvelle population par transformations génétiques . . . 62
Optimisation numérique . . . 62
Optimisation opérationnelle . . . 63
Optimisation structurelle . . . 64
4.8.3 Recherche d'un ensemble de fonctions pertinentes . . . 65
4.8.4 Amélioration des performances de la recherche . . . 66
Calcul des opérations . . . 66
Réécriture des fonctions . . . 67
Mécanisme de cache . . . 70
4.9 Construction et optimisation des modèles descriptifs . . . 70
4.9.1 Dimension des fonctions caractéristiques . . . 70
4.9.2 Sélection de fonctions caractéristiques . . . 71
4.9.3 Synthèse d'un modèle à partir des fonctions caractéristiques 71
4.9.4 Optimisation des modèles . . . 72
4.10 Agrégation temporelle des modèles - Synthèse d'un extracteur final 72
4.10.1 Validité du descripteur « brut » . . . 73
4.10.2 Agrégation temporelle de modèles locaux . . . 73
4.10.3 Segmentations et Agrégations . . . 74
4.10.4 Enregistrement d'un exécutable pour un descripteur agrégé 74
4.11 Conclusion . . . 75

5. Modélisation de descripteurs musicaux avec EDS . . . 77
5.1 Définition de problèmes de description musicale pour EDS . . . 77
5.1.1 Evaluation a priori de la pertinence de descripteurs musicaux . . . 78
Jeux de discrimination d'extraits musicaux . . . 78
5.1.2 Constitution d'une base de signaux . . . 80
Contexte d'utilisation . . . 80
Propriété à évaluer . . . 80
Sélection des signaux . . . 80
5.1.3 Étiquetage de la base de signaux . . . 81
Mise en place de tests perceptifs pour la description musicale 81
Validation des résultats des tests perceptifs . . . 82
5.2 Élaboration d'une méthode générale d'étude des descripteurs musicaux . . . 83
5.2.1 Étude de l'influence des paramètres de recherche . . . 83
Paramètres de référence . . . 84
Résultats de référence . . . 85
Influence de la taille de population . . . 85
Influence du critère d'arrêt . . . 87
Influence du pattern . . . 87
5.2.2 Génération d'un ensemble de fonctions pertinent . . . 88
5.2.3 Étude de problèmes classiques de description musicale . . 89
Contexte des bases de signaux . . . 89
Fonctions de référence . . . 90
Méthodes de classification . . . 90
Évaluation des performances du système . . . 90
5.3 Modélisation de descripteurs musicaux avec EDS . . . 91
5.3.1 Reconnaissance de sons de voix . . . 91

Détection générale de voix chantée 91
Reconnaissance d'une voix particulière 93
Reconnaissance d'une voix particulière dans une chanson précise 96
5.3.2 Perception de la percussivité de sons polyphoniques . . . 98
Modélisation 98
Étapes typiques d'une recherche génétique de fonction . . 101
5.3.3 Classification de sons d'instruments 101
Discrimination de 4 sons d'instruments monophoniques . 101
Discrimination des 3 instruments les plus proches 103
Effet de contexte 104
5.3.4 Reconnaissance du genre d'extraits musicaux 105
Classification en 4 genres musicaux 105
5.3.5 Perception de l'énergie globale d'extraits musicaux 107
5.4 Conclusion 109

Partie 2
Exploitation des descripteurs dans des applications musicales 113

6. Utilisation des descripteurs dans des applications musicales . 113
6.1 Contexte général de la distribution électronique de musique . . . 114
6.1.1 L'information musicale en EMD 114
6.1.2 Le projet Cuidado et le Music Browser 115
Recherche directe 115
Recherche par similarité 115
Modes de recherche originaux 116
6.2 Utilisation des descripteurs musicaux synthétisés par EDS 116
6.2.1 Descripteurs musicaux bruts 116
6.2.2 Descripteurs musicaux dérivés par agrégation temporelle . 116
6.3 Intégration des descripteurs dans les applications musicales . . . 117
6.3.1 Utilisation de descripteurs pour l'exploration de bases d'objets musicaux 118
6.3.2 Suivi de descripteur 120
Segmentation locale 120
Segmentation structurelle 120
6.3.3 Utilisation de descripteurs pour la création musicale par séquençage de sons 120
6.4 Mise en séquence d'objets sonores 120
6.4.1 Séquence d'objets interdépendants 121
6.4.2 Intérêt des applications musicales utilisant des séquences . 121
Recherche d'objets musicaux dans des grandes bases . . . 121
Pertinence de la séquence réalisée 122
6.4.3 Mise en place d'un système général de mise en séquence . 122
Spécification des propriétés de la séquence 122
Recherche des objets et construction de la séquence . . . 123
6.4.4 Présentation détaillée du système « Musaicing » 123
Mécanisme général 124
Définition et résolution d'un problème de satisfaction de contraintes 124

Problème de satisfaction de contraintes d'une mosaïque . 125
Résolution d'un problème de mosaïques musicales 126
Calculs des coûts de segments et de séquence 127
Synthèse sonore de la mosaïque 128
6.4.5 Exemples de génération de mosaïques musicales 128
Construction d'une mosaïque simple : contraintes locales . 129
Effet d'une contrainte globale : continuité 130
Effet de la pondération des contraintes 130
Contrainte globale élaborée : tempo percussif 131
Combinaison de contraintes locales et globales multiples . 131
6.4.6 Génération de programmes musicaux 134
6.5 Conclusion - Extension aux objets multimedia 134

7. Conclusion et Perspectives 137

Annexes 141

A. Glossaire 143

B. Principales fonctions caractéristiques et modèles utilisés en description musicale 147
B.1 Classification de sons 147
B.2 Genre musical 149
B.3 Discrimination parole / musique 151
B.4 Détection et reconnaissance de voix chantée 152
B.5 Emotion 153

C. Présentation des méthodes classiques de modélisation utilisées dans EDS 155
C.1 Méthode 1-R de Holte 155
C.2 Modèle de Bayes Naïf 155
C.3 Plus-proche(s) voisin(s) 156
C.4 Régression Linéaire 157
C.5 Régression Locale Pondérée 157
C.6 Modèle Gaussien 157
C.7 Modèle de Mélange de Gaussiennes 158
C.8 Arbres de Décisions 159
C.9 Réseaux de Neurones 159

D. Mesures de pertinence 161

E. Liste de fonctions trouvées par EDS 163
E.1 Reconnaissance de sons de voix 163
E.2 Reconnaissance de la voix d'une chanteuse 165
E.3 Reconnaissance de la voix d'une chanteuse dans une chanson . . 167
E.4 Reconnaissance de sons percussifs polyphoniques 168
E.5 Classification en 4 instruments 169
E.6 Discrimination en 4 genres musicaux 170
E.7 Évaluation de l'énergie musicale 170

LISTE DES TABLEAUX

4.1 Types de données utilisés dans EDS 45
4.2 Exemples de tables de typage d'opérations utilisées dans EDS . . 49
4.3 Comparaison des temps de calcul d'opérateurs externes et internes, sur un signal numérique de 10 secondes échantillonné à 44100Hz 67

5.1 Comparaison des taux d'utilisation dans le Discrimination Game, et d'erreur dans le Recognition Game, pour 4 descripteurs usuels 79
5.2 Comparaison des performances des modèles classiques et générés par EDS pour la reconnaissance de sons vocaux 92
5.3 Importance de chaque opération pour la performance globale de la meilleure fonction EDS pour la détection de voix chantée . . . 93
5.4 Comparaison des performances des modèles de référence et générés par EDS pour la reconnaissance de la voix d'une chanteuse . . . 94
5.5 Importance de chaque opération pour la performance globale de la meilleure fonction EDS pour la détection de la voix d'une chanteuse 95
5.6 Comparaison des performances des modèles de référence et générés par EDS pour la reconnaissance de la voix d'une chanteuse dans une chanson précise 96
5.7 Importance de chaque opération pour la performance globale de la meilleure fonction EDS pour la détection de la voix d'une chanteuse dans une chanson précise 99
5.8 Comparaison des performances des modèles classiques et générés par EDS pour la perception de la percussivité de sons polyphoniques100
5.9 Comparaison des performances des modèles classiques et générés par EDS pour la classification de sons de 4 instruments acoustiques102
5.10 Comparaison des performances des modèles classiques et générés par EDS pour la classification de sons de 3 instruments acoustiques103
5.11 Comparaison des performances des modèles classiques et générés par EDS pour la reconnaissance d'extraits caractéristiques de genres musicaux 106
5.12 Comparaison des performances des fonctions classiques et générées par EDS pour la modélisation de l'énergie musicale 108

6.1 Applications principales pour les différents types d'objets musicaux118

B.1 Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en classification de sons 149

B.2 Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en classification en genre musical 150
B.3 Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en discrimination parole / musique 151
B.4 Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en détection de voix chantée 152
B.5 Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en émotion dans la musique 153

TABLE DES FIGURES

2.1 Schéma de l'extracteur de tempo de [Scheirer, 1998] 13

3.1 Interface du test d'évaluation de l'Energie Globale d'extraits musicaux 28
3.2 Visualisation de signaux d'extraits musicaux d'énergies diverses . 29
3.3 Graphiques de correspondance entre l'énergie musicale (« E perçue ») et diverses caractéristiques du signal (Tempo, énergie brute « ESignal », fréquence centrale « FMoy », notre fonction Logvardiff « LogDeriveeVar ») 29
3.4 Graphiques de correspondance entre la percussivité (« Percussivity ») et diverses caractéristiques du signal (Fréquence fondamentale « Pitch », énergie brute « energ », fonction d'attaque avec une fenêtre de Chebyshev de 1000 échantillons « attac1000 », fonction d'attaque avec une fenêtre de Chebyshev de 5000 échantillons « attac5000 ») 31
3.5 Profil Percussif d'un extrait de « Darling »(Beatles) 31
3.6 Profil Vocal d'un extrait de « Streets of Philadelphia »(Bruce Springsteen) 32

4.1 Visualisation de spectres du signal acoustique d'un sinus pur (650Hz) noyé dans un bruit rose (1000-2000Hz) de forte amplitude 36
4.2 Architecture globale du système EDS 38
4.3 Représentation de fonctions sous forme d'arbre équivalents 40
4.4 Barème de notation des heuristiques 54
4.5 Choix d'une opération parmi trois par tirage pondéré sur leur score global : espaces de tirage pour différents ordres de pondération 57
4.6 Exemple de clonage d'une fonction 63
4.7 Exemple de substitution d'un opérateur dans une fonction 63
4.8 Exemple d'addition d'un opérateur à une fonction 64
4.9 Exemple de mutation d'une fonction 64
4.10 Exemple de croisement de deux fonctions 65

5.1 Interfaces des jeux descriptifs utilisés pour évaluer la pertinence de descripteurs acoustiques sur des musiques populaires 78
5.2 Applet utilisées pour les tests perceptifs des descripteurs de percussivité sonore (régressif) et de sensualité d'un extrait musical (classe) 81
5.3 Répartition des pertinences sur une base de test, des fonctions obtenues sur 100 recherches génétiques de la fonction F_{ref}, en faisant varier les paramètres par défaut : Taille de population = 20, Critère d'arrêt = 5, Pattern = $!_x(Signal)$ 86

5.4 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la détection de sons vocaux 92
5.5 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la détection de la voix d'une chanteuse 94
5.6 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la détection de la voix d'une chanteuse dans une chanson précise 97
5.7 Représentation sous forme d'arbre d'opérations de la meilleure fonction EDS pour la détection de la voix d'une chanteuse dans une chanson précise 97
5.8 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la caractérisation de sons polyphoniques percussifs 100
5.9 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la classification de 4 instruments 102
5.10 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la classification de 3 instruments 104
5.11 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la classification de 2 sons de guitare 105
5.12 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la discrimination entre 4 genres musicaux 107
5.13 Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS en fonction de leur étiquette perceptive, pour l'évaluation de l'énergie musicale . . . 108

6.1 Groupes de descripteurs musicaux et applications associées . . . 113
6.2 Interface du Music Browser consacrée à la recherche directe de titres musicaux, incluant les descripteurs de Présence de Voix et d'Énergie Globale synthétisés par EDS (dans « Song Properties ») 119
6.3 Interface du Music Browser consacrée à la génération automatique de descripteurs personnels à partir d'une sélection de titres caractéristiques 119
6.4 Traduction des propriétés de la mosaïque sous forme de contraintes locales et globales 125
6.5 Fonctionnement de l'algorithme de « Local Search »appliqué à la résolution de mosaïques musicales 127
6.6 Interface de l'application « Musaicing » 129
6.7 Effet de la contrainte globale de continuité sur la fréquence instantanée de la mosaïque d'un extrait de fréquence ascendante, constituée de sons de fréquence descendante 130
6.8 Effet de la pondération de la contrainte globale de continuité sur la fréquence instantanée de la mosaïque d'un extrait de fréquence ascendante, constituée de sons de fréquence descendante 131

6.9 Visualisation du résultat de l'application d'une contrainte de tempo percussif avec deux paramètres différents sur le signal de la mosaïque132
6.10 Visualisation du signal brut d'une mosaïque défini par deux contraintes de tempo percussif en basse fréquence et en haute fréquence - Observation du résultat de ces contraintes sur le signal filtré . . 133
6.11 Evolution de la fréquence instantanée dans une mosaïque définie par deux contraintes de tempo percussif, et d'une contrainte globale de « montée puis descente »de la fréquence fondamentale le long de la séquence 133
6.12 Interface du Music Browser consacrée à la génération automatique de programmes musicaux - Exemple de programme généré à l'aide de quatre contraintes globales 135
6.13 Interface du Music Browser consacrée à la spécification des propriétés de programmes musicaux en vue de leur génération automatique 135

C.1 Modèle de classification Bayésienne pour un paramètre x et 2 classes 156
C.2 Prédiction de la classe d'un point X parmi 2 classes A et B dans un espace à 2 dimensions, par la méthode des k plus proches voisins : A pour k=1 ; B pour k=3 ; A pour k=5 156
C.3 Modélisation d'un nuage de points dans un espace à 2 dimensions par un mélange de 3 gaussiennes 158
C.4 Schéma d'un neurone élémentaire 160
C.5 Schéma d'un perceptron à 2 couches cachées [N :U :V :S] 160

1. INTRODUCTION - OBJECTIFS & MÉTHODOLOGIE

1.1 Introduction

1.1.1 L'essor de l'informatique musicale

La récente explosion de la quantité de musique accessible sur Internet a créé de nouveaux intérêts dans le domaine de la distribution de musique, aussi bien pour les utilisateurs qui recherchent des moyens pratiques de gérer et d'utiliser leurs collections personnelles, que pour les distributeurs, qui cherchent à donner accès au plus grand nombre à la musique qui les intéresse.

Cette gigantesque quantité d'information ne pouvant être gérée directement, l'utilisation de méta-données, c'est-à-dire de données qui décrivent elles-mêmes les données recherchées, devient incontournable. Dans le cadre de la recherche en informatique musicale, ce besoin se traduit par un regain d'activité dans le domaine des descripteurs musicaux. Les recherches actuelles ont pour but d'extraire des descripteurs pertinents dans le cadre des applications visées (comme par exemple la recherche efficace de musique sur Internet), et surtout de manière automatique car le volume d'information à traiter ne permet pas une extraction au cas par cas.

Chez Sony CSL, cette recherche s'inscrit dans le cadre du projet EMD (« Electronic Music Distribution »), dont l'objectif est de comprendre et modéliser les goûts musicaux des utilisateurs, dans le but de faciliter l'accès à la musique à un public d'utilisateurs non-spécialistes. L'enjeu de ce projet est double :

- Gestion des discothèques personnelles (de sons et de morceaux de musique), et utilisation dans des programmes musicaux originaux ;
- Recherche de nouvelle musique (sur Internet), en fonction des goûts et des aspirations de l'utilisateur, dans le contexte des catalogues musicaux numériques de grande taille et de la distribution en ligne.

Les axes de recherche associés concernent la modélisation musicale au sens large, c'est-à-dire les problèmes de représentation par le contenu de titres musicaux et l'extraction automatique de descripteurs musicaux pertinents (dans le contexte de la norme Mpeg7), et la création d'applications musicales intéressantes, comme la génération automatique de programmes musicaux ou de compositions à partir de ces méta-données.

1.1.2 Différents types de descripteurs musicaux

Les descripteurs musicaux sont extrêmement variés, et on peut les classer suivant divers critères : leur origine, leur niveau d'abstraction, leur subjectivité, etc...

Descripteurs acoustiques et culturels

Les descripteurs culturels sont des données liées au contexte dans lequel a été créé un son ou un extrait musical. Ils sont en général textuels (nom de l'artiste, titre, etc...), et extractibles par recherche dans des bases de données sur internet. Cependant, ces descripteurs ne donnent aucune information sur les caractéristiques sonores intrinsèques à la musique. Dans ce travail, afin de décrire véritablement les propriétés sonores de passages musicaux indépendamment de leur contexte culturel, nous nous intéressons uniquement aux descripteurs extraits à partir de l'étude de signaux acoustiques.

Niveaux de description

L'ensemble des descripteurs acoustiques peut être divisé en plusieurs niveaux de description, mesurant l'abstraction des propriétés qu'ils évaluent par rapport à celles du signal acoustique brut :

- les descripteurs de bas niveau expriment des propriétés sonores significatives directement observables ou calculables à partir du signal audio, mais qui n'ont pas forcément de signification évidente au sens musical, comme par exemple les caractéristiques des enveloppes d'énergie ou du spectre du signal ;
- les descripteurs de haut niveau expriment des propriétés qui ont une signification musicale plus évidente et compréhensible par les utilisateurs, mais nécessitent le calcul de modèles acoustiques plus complexes, éventuellement à partir de données extérieures au signal acoustique lui-même (comme par exemple des sons de référence).

Nous nous intéressons ici à la modélisation de descripteurs de haut-niveau, c'est-à-dire de propriétés musicales utilisables par un public non-spécialiste des signaux acoustiques, dans le cadre d'applications musicales variées.

Descripteurs locaux et globaux

Les critères acoustiques d'un morceau de musique peuvent être définis à différentes échelles :

- Au niveau local, l'étude des sons contenus dans les morceaux (sur des durées de l'ordre de quelques dixièmes de seconde), comme par exemple leurs hauteur, timbre, percussivité, brillance, niveau sonore, origine, etc... fournit des descripteurs utilisables dans le cadre d'applications de composition musicale, ou de gestion de bases de sons.
- Au niveau global, la comparaison globale de chansons entières (sur des macro-durées de l'ordre de plusieurs minutes), comme par exemple leur genre, rythme, humeur, timbre global, énergie globale,...fournit des descripteurs utilisables dans le cadre d'applications originales de recherche de musique dans des grands catalogues.

– Au niveau intermédiaire, l'étude de la structure interne des morceaux, par analyse d'extraits musicaux (sur des durées de l'ordre de quelques secondes), comme par exemple la détection de textures, d'instruments, de structures solo-couplet-refrain,...fournit des descripteurs utilisables dans le cadre d'applications alternatives de traitement d'extraits musicaux, comme par exemple la recherche de solos instrumentaux, ou le résumé d'un morceau pour une écoute rapide des passages pertinents.

Descripteurs unaires

De nombreux travaux sur le sujet des descripteurs sonores ont été réalisés depuis de nombreuses années, visant à définir directement les caractéristiques d'extraits sonores brefs et précis, comme par exemple leur timbre. L'originalité de ce travail réside dans le fait qu'il ouvre un nouvel axe d'étude en considérant le problème des descripteurs musicaux sous un aspect global.

Ainsi, pour représenter les sons, extraits, et titres musicaux, nous avons choisi d'utiliser des descripteurs unaires, c'est-à-dire qui attribuent une valeur perceptive unique pour un objet musical global. Par exemple, pour décrire des sons, on utilisera une hauteur globale perçue, une énergie, une percussivité,...Pour les titres entiers, on cherchera un tempo, ou plus généralement un rythme, une énergie globale du morceau, un timbre global,...Les valeurs associées peuvent être des évaluations numériques (Tempo = 120, Dansabilité = 50%, Tempo = 120, ...), ou bien des classes sonores (rythme = bossa, énergie = [élevée], timbre global = « guitare électrique », ...).

L'utilisation de descripteurs unaires repose sur une hypothèse fondamentale qui suppose qu'on peut décrire un objet aussi complexe que 3 mn de musique avec une seule valeur. Dans le cas de titres de musique très hétérogènes, il peut être intéressant de s'intéresser à l'évolution d'une propriété locale. Dans ce cas, l'hypothèse reste valable localement sur les extraits étudiés. Parfois, même la description globale de titres musicaux nécessite une extraction de descripteurs sonores au niveau local. Par exemple, nous avons réalisé un extracteur rythmique qui utilise l'extraction de sons percussifs au niveau local qui vont être recherchés à un point de vue global pour déterminer un modèle rythmique. Plus généralement, l'approche d'étude de descripteurs que nous présentons peut être qualifiée d' « holistique », dans le mesure où elle considère les fortes relations et interdépendances entre les niveaux de perception local et global.

Subjectivité des descripteurs

Ces descripteurs représentent une propriété particulière d'un son ou d'un extrait musical. Ils sont donc liés à la perception qu'en ont les auditeurs. Cependant, pour assurer la robustesse de la description musicale, il est nécessaire que ces descripteurs soient perçus de manière identiques par l'ensemble des auditeurs. On peut ainsi distinguer différents niveaux de généralisation des descripteurs :

– les descripteurs quasi-objectifs, qui reposent sur des propriétés musicales bien définies et universelles, comme la fréquence fondamentale d'un son, ou le tempo d'un extrait musical ;
– les descripteurs subjectifs, qui reposent sur des propriétés musicales dont l'universalité doit être montrée et évaluée par la réalisation de tests per-

ceptifs sur un ensemble d'auditeurs, comme par exemple la perception de l'énergie musicale d'un extrait ou de la percussivité d'un son polyphonique. L'étude de ces descripteurs repose sur l'hypothèse qu'on peut extraire directement du signal musical des valeurs perceptives universelles ;
- les descripteurs personnels, qui reposent sur une propriété musicale perçue par un auditeur unique, comme par exemple le fait qu'il apprécie ou non un extrait musical ; ces descripteurs ne sont pas utilisables dans des applications musicales générales, mais sont intéressants pour modéliser par exemple les préférences personnelles d'utilisateurs.

1.1.3 Extraction et utilisation des descripteurs

Le travail de recherche présenté dans ce document a pour objet l'extraction automatique de descripteurs par traitement du signal, et son application aux signaux acoustiques dans le cadre d'applications musicales. Il touche simultanément aux domaines :
- perceptif : dans le choix de descripteurs pertinents dans le cadre des applications désirées, et la constitution de bases de valeurs générales utilisables pour la modélisation de ces descripteurs ;
- traitement du signal : dans les phases de recherche et d'extraction de modèles descriptifs à partir des signaux ;
- informatique : dans les phases d'intégration de ces descripteurs dans des applications musicales.

Etude perceptive des descripteurs

Les descripteurs acoustiques étudiés sont destinés à être utilisés dans des applications musicales pour fournir à un public non-musicologue des outils permettant de manipuler des sons, extraits et titres musicaux, à partir de caractéristiques simples, utiles et générales.

La *simplicité* des descripteurs est la certitude que ces derniers pourront être utilisés par une majorité d'utilisateurs. Leur *utilité* correspond à leur faculté à exprimer les goûts/désirs des utilisateurs dans le cadre des applications désirées. On peut associer ces deux notions dans une mesure de la pertinence globale des descripteurs.

Bien qu'il soit possible de se faire une idée à priori de la pertinence de certains descripteurs, celle-ci peut être évaluée plus précisément en réalisant des tests auprès d'un ensemble d'utilisateurs, pour étudier leurs méthodes d'interaction avec la musique, comme par exemple pour savoir de quelle manière recherchent de la musique nouvelle : par instrumentation, par genre, par rythme, par mélodie, par tonalité, par timbre sonore, par ambiance sonore,...

L'autre propriété indispensable des descripteurs utilisés est leur *généralité*, c'est-à-dire le fait que différents utilisateurs auront la même perception d'un même titre musical, pour un axe descriptif donné. Cette propriété soulève le problème de la liaison entre le monde subjectif de la perception musicale et le monde objectif de la description musicale. En effet, les descripteurs sont tous liés à la façon dont est perçue la musique par les auditeurs. Cependant nous distinguerons les descripteurs dont la perception est objective, de ceux qui font appel à la subjectivité de l'auditeur.

D'une part, l'évaluation des descripteurs *quasi-objectifs* se fait en général de manière évidente et identique pour tous les auditeurs. Ce seront le plus souvent des descripteurs booléens du type « Cet extrait musical contient-il de la voix chantée ? », que tous les auditeurs pourront évaluer sans difficulté pour la plupart des extraits. Bien que cette information puisse être extraite facilement à la main par tout auditeur, sa modélisation en vue d'une extraction automatique est indispensable pour pouvoir gérer et étiqueter facilement des grandes bases de l'ordre des centaines de milliers de titres.

D'autre part, de nombreux autres descripteurs sont *subjectifs* et liés à la perception musicale de chaque auditeur, comme par exemple l'évaluation de l'énergie globale d'un extrait musical, ou la modélisation des goûts musicaux. Dans ce cas, il est également nécessaire d'évaluer la dépendance du descripteur par rapport à différents auditeurs. Ainsi, pour valider ce type de descripteurs, il est nécessaire de réaliser des tests perceptifs auprès d'un ensemble d'auditeurs, afin d'évaluer, pour divers titres musicaux, la cohérence des résultats.

Nous avons par exemple réalisé ce genre de tests pour vérifier l'universalité de la notion d'énergie globale d'un extrait musical, qui mesure la différence entre musiques calmes et des musiques énergiques, indépendamment du volume sonore de l'extrait écouté. Ces tests ont montré que l'énergie est un descripteur généralisable car il est globalement indépendant de l'auditeur concerné.

Certains descripteurs peuvent être liés simultanément au contenu musical et à l'auditeur, comme par exemple « Ressentez-vous une émotion quelconque à l'écoute de cet extrait musical ? », qui dépend généralement des souvenirs et de l'état d'esprit de l'auditeur. Des tests perceptifs permettent d'évaluer le caractère personnel des descripteurs. A l'extrême, les descripteurs *personnels* sont liés à un auditeur précis, comme par exemple « Est-ce que vous aimez cet extrait ? ». Ces descripteurs doivent alors être modélisés pour chaque individu.

Extraction des descripteurs à partir du signal acoustique

Différentes techniques existent pour extraire de manière automatique une information perceptive à partir de signaux acoustiques. Ainsi, certains descripteurs simples peuvent être extraits directement à partir du signal audio en utilisant des algorithmes spécifiques, tandis que d'autres descripteurs de haut-niveau nécessitent non seulement l'extraction de caractéristiques primaires à partir du signal (les descripteurs de bas-niveau), mais également la construction d'un modèle permettant de combiner ces paramètres pour obtenir le descripteur désiré.

Ce dernier schéma est classique dans l'état de l'art de la description musicale : on calcule tout d'abord un grand nombre de descripteurs de bas-niveau à partir du signal audio, on choisit ensuite les plus pertinents en utilisant diverses méthodes de sélection de fonctions plus ou moins complexes, avant de les agréger dans un modèle plus complexe afin de produire un descripteur de plus haut-niveau.
L'agrégation des valeurs des caractéristiques sur des fenêtres d'observation locales successives pour obtenir une valeur globale pour le signal entier, peut se faire de diverses façons. Par exemple, cette agrégation peut se faire de manière séquentielle (on obtient un vecteur de dimension le nombre de fenêtres), par extraction de motifs (on obtient un vecteur de dimension la longueur du motif), par analyse statistique (en utilisant moyenne, variance, variables statistiques

d'ordre supérieur, corrélations,...), ou par analyse modale (on obtient alors un vecteur de dimension le nombre de modes - par exemple, pour le volume sonore, un morceau peut être constitué de 2 parties de volumes très différents, la bonne représentation du volume ne sera donc pas la valeur moyenne du volume du morceau, mais un couple de valeurs de volume avec leurs proportions respectives (représentation bi-modale)). La modélisation finale consiste à lier les valeurs des descripteurs aux résultats perceptifs, en utilisant des méthodes d'analyse de données (analyse en composante principale, réseaux de neurones, modèles gaussiens,...).

Intégration des descripteurs dans des applications musicales

Tous ces descripteurs sont modélisés dans le but d'être intégrés dans des applications musicales utilisant du contenu musical original. Ils doivent alors satisfaire à diverses contraintes d'exploitation, principalement : leur généralité, leur modularité, leur qualité, et leur efficacité. La généralité d'un descripteur implique qu'il sera applicable à tout extrait musical concerné. La modularité d'un descripteur implique qu'il puisse être intégré dans diverses applications informatiques, sans problème de compatibilité. La qualité d'un descripteur est sa capacité à donner une valeur descriptive correcte pour tout extrait musical. Enfin, l'efficacité d'un descripteur est la capacité à calculer sa valeur en un temps minimum compatible avec l'utilisation des applications musicales.

1.2 Plan du document

Le travail de recherche est présenté en deux parties : la synthèse automatique de descripteurs suivant notre approche évolutionniste, et l'intégration de ces descripteurs dans des applications musicales.

La première partie, qui constitue l'essentiel de ce document, a abouti à la création d'un système général d'extraction automatique de descripteurs à partir de signaux, appelé EDS (« Extractor Discovery System »). Ce système est utilisé principalement pour extraire des descripteurs musicaux à partir du signal audio, et traiter ainsi les problèmes usuels de description rythmique, mélodique ou timbrale, ainsi que des problèmes originaux, comme l'évaluation de l'énergie ou de la dansabilité d'extraits musicaux. Le système étant général, il est également utilisable pour résoudre tout problème de description à partir de signaux numériques, par exemple à partir d'images.

La seconde partie, présentée en fin de document, a abouti au développement de diverses applications musicales utilisant les descripteurs créés par EDS, principalement leur intégration dans une application de recherche musicale, le « Music Browser », et la création d'un outil de composition musicale original, le « Musaicing ».

Dans la première partie du document, le chapitre 2 présente l'état de l'art des recherches en description musicale à partir du signal acoustique. Après avoir présenté les études classiques de rythme, mélodie, et timbre, nous présentons les travaux plus récents sur la modélisation de perceptions musicales.

Comme dans la majorité des travaux de l'état de l'art, nous avons construit nos premiers descripteurs musicaux de manière empirique : un extracteur de sons

percussifs, un modèle d' « énergie »musicale, et des fonctions de percussivité et de vocalité de sons, présentés dans le chapitre 3.

L'étude de l'état de l'art et l'expérience apportée par nos études empiriques nous ont mené à un constat général : dans la majorité des travaux, la synthèse de descripteurs musicaux suit une méthode classique consistant à chercher des paramètres caractéristiques du signal et à les agréger pour construire l'extracteur du descripteur. De plus, beaucoup de travaux utilisent comme paramètres les mêmes fonctions de traitement du signal, comme par exemple les fonctions proposées par la norme Mpeg7, en plus de fonctions empiriques, et montrent que ces dernières permettent d'améliorer les performances du modèle descriptif, suivant des méthodes classiques de régression ou de classification.

C'est de cette apparente similarité méthodique qu'est née l'idée d'un système réalisant automatiquement la construction de fonctions pertinentes, et leur intégration dans des modèles optimaux, aboutissant à la création d'EDS (« Extractor Discovery System »).

Dans le chapitre 4, après avoir justifié sur un exemple simple l'approche utilisée dans le système, nous présentons les principes généraux d'EDS, notamment les problèmes descriptifs qu'il se propose de résoudre (classification ou régression) et l'architecture permettant de réaliser automatiquement les deux étapes d'extraction de fonctions caractéristiques pertinentes et de synthèse d'un modèle optimal, ainsi que la synthèse d'un programme exécutable permettant de calculer ce modèle.

Nous présentons et justifions ensuite la méthode de construction de fonctions caractéristiques pertinentes, qui est la partie la plus importante d'EDS. Les fonctions sont représentées sous forme d'une combinaison d'opérations mathématiques ou de traitement du signal simples et paramétrables. Le système génère automatiquement des fonctions à partir de règles de construction strictes (typage des données et patterns), et recommandées (heuristiques), qui représentent l'expertise du système en traitement du signal. Afin de trouver les fonctions pertinentes permettant de résoudre un problème descriptif donné, le système génère des fonctions, et fait évoluer les « meilleures »en leur appliquant des transformations, suivant un algorithme de type génétique, jusqu'à obtenir une fonction optimale.

La phase de recherche génétique fournit un ensemble de fonctions pertinentes, qu'EDS utilise, après une éventuelle étape de sélection, pour réaliser la synthèse finale du modèle. Pour cela le système teste les modèles de descripteur obtenus en utilisant différents algorithmes d'apprentissage avec des paramètres variés, et choisit le modèle optimal.

Enfin, en vue de l'utilisation des descripteurs dans des applications externes, le système génère un programme exécutable permettant de calculer le modèle et de lui associer une méthode d'agrégation, c'est-à-dire un post-traitement permettant de le calculer sur un flux audio segmenté.

Dans le chapitre 5, nous évaluons les performances des modèles synthétisés par EDS pour résoudre des problèmes de description musicale, classiques et originaux, à partir du signal acoustique, et les comparons aux modèles synthétisés à partir de fonctions de référence comme celles de la norme Mpeg7.

Dans la deuxième partie du document, le chapitre 6 présente les possibilités d'exploitation des descripteurs dans des applications musicales, la synthèse automatique de descripteurs par EDS permettant d'intégrer facilement dans des applications musicales des critères de recherche ou de classification origi-

naux. Nous présentons plus particulièrement une application générale de gestion de bases musicales, le Music Browser, et une application originale de génération des séquences d'objets associés à des descripteurs, comme par exemple les sons (mosaïques musicales) et les chansons entières (Programmes musicaux), la génération des séquences étant contrôlée par la mise en place de contraintes sur les valeurs des descripteurs des objets.

Pour terminer, le chapitre 7 conclut en synthétisant les perspectives ouvertes par ce travail de recherche.

PARTIE 1
EXTRACTION AUTOMATIQUE DE DESCRIPTEURS PAR TRAITEMENT DU SIGNAL

2. ETAT DE L'ART DE L'ANALYSE DE SIGNAUX MUSICAUX

Nous recevons à chaque instant de nombreux signaux naturels, c'est-à-dire des informations sur les grandeurs physiques de notre environnement et leur évolution dans le temps. Parmi l'ensemble des signaux que nous recevons, ceux que nous sommes capables d'analyser et d'interpréter constituent notre perception du monde, qui est donc essentiellement basée sur nos 5 sens.

Lorsque nous recevons un signal acoustique, notre système auditif est capable de percevoir ses composantes dans une bande de fréquences limitée (20-20000Hz) : les sons audibles. Nous réalisons alors une analyse de ce signal perçu, afin d'en déduire une interprétation physique : comprendre une phrase parlée, suivre la mélodie d'un morceau de musique, reconnaître la sonnerie du téléphone, la voix d'un ami, évaluer le niveau de bruit dans une pièce, etc...Nous réalisons donc sans cesse des évaluations et classifications des signaux sonores perçus.

La recherche en analyse de signaux sonores a pour objectif de comprendre et modéliser ces tâches, que nous réalisons généralement inconsciemment. Les applications associées consistent alors à concevoir des systèmes automatiques capables de reproduire les tâches réalisées par des experts humains, comme par exemple des diagnostics médicaux ou la description de musique.

Dans le domaine récent de la description de signaux musicaux, les travaux menés jusqu'à présent visent principalement deux types d'applications : l'analyse de documents multimédia (transcription automatique de musique, découpage de programmes radiophoniques ou de bandes sonores de cinéma), et la gestion des bases d'objets sonores. Ces vastes domaines d'étude ne sont évidemment abordables que par leur division en une multitude de travaux plus spécifiques, reposant sur un principe de base : la représentation à l'aide de descripteurs musicaux, qui sont ensuite intégrés dans les applications.

Par exemple, la transcription automatique de musique peut être vue comme l'évaluation de la fréquence fondamentale (descripteur mélodique) et la reconnaissance de l'instrument (description de timbre) jouant chaque note d'un flux audio segmenté. Cet exemple présente simplement les trois axes de recherche principaux, suivant lesquels nous abordons cet état de l'art :

- les travaux d'*évaluation/régression*, qui visent à mesurer l'importance de propriétés sonores : la sonie, la fréquence fondamentale d'un son, sa percussivité, l'énergie perçue à l'écoute d'un extrait musical, sa sensualité, etc...
- les travaux d'*identification/classification*, qui visent à classer les sons suivant des propriétés connues : détection de musique dans un flux audio, reconnaissance d'instrument, de chanson, de mélodie, de genre, de voix, d'un type de son, etc...

– la construction d'outils d'analyse efficaces, qui permettent de traiter les signaux sonores en amont et aval de leur description : séparation de sources, effets audio, segmentation de flux sonore, analyse harmonique, etc...

2.1 Évaluation de propriétés sonores

Nous présentons ici les travaux de recherche en évaluation, qui ont porté principalement sur deux propriétés musicales : les hauteurs de sons et le tempo d'extraits musicaux. Nous introduisons également notre évaluation de la notion nouvelle d'énergie musicale.

2.1.1 Extraction de mélodie

Une étude mélodique consiste à extraire les hauteurs caractéristiques des sons successifs d'un extrait musical. Ces sons peuvent être mono- ou poly-phoniques (une note ou un accord de piano), mono- ou poly-instrumentaux (un accord de guitare, ou un extrait de musique populaire).

Cette extraction est utilisée principalement pour la transcription automatique de morceaux, utilisée notamment pour construire des applications musicales de recherche de musique par mélodie, par exemple en fredonnant une chanson (« query by humming »).

Les premières recherches ont porté sur l'évaluation de la fréquence fondamentale de notes monophoniques, notamment à partir d'études psycho-acoustiques visant à comprendre comment nous percevons la hauteur d'une note ([Rabiner *et al.*, 1976]). Les algorithmes de détection basés sur l'autocorrélation ([Brown, 1991]) et la corrélation avec un pattern harmonique ([Brown, 1992]), efficaces pour l'étude de hauteur de voix, se sont révélés peu efficaces pour l'étude de sons musicaux moins harmoniques. D'autres méthodes, comme l'autocorrélation spectrale utilisée en traitement de la voix ([Kunieda *et al.*, 1996]), ou la périodicité de l'enveloppe du signal ([Hartman, 1996]) permettent de mieux gérer ces inharmonicités, et sont plus robustes au bruit. L'utilisation de la transformée de Fourier d'ordre 1 ([Brown, 1993]), d'ordre N ([Marchand, 1998]), ou d'une succession de transformées de Fourier ([Marchand, 2001]) permet d'obtenir une plus grande résolution fréquentielle. [Keiler et Marchand, 2002] présente une revue comparative des différentes techniques utilisées pour l'extraction de hauteur dans les sons stationnaires.

Dans des environnements musicaux plus complexes et polyphoniques, [Klapuri, 2000] présente les techniques utilisées, et notamment un algorithme itératif qui extrait la hauteur la plus vraisemblable sur des bandes de fréquence séparées, et donne des résultats équivalents aux performances humaines (détaillé dans [Klapuri *et al.*, 2000]). Un système multi-agents basé sur la probabilité de trajectoires fréquentielles permet de détecter séparément des instruments différents comme la ligne de basse et la mélodie ([Goto, 2001b]). La recherche d'une combinaison linéaire de notes dans le domaine temporel, à partir d'une base de notes auto-générée, donne d'excellents résultats sur des enregistrements polyphoniques mono-instrumentaux ([Bello *et al.*, 2002]).

Même si ces méthodes sont toutes différentes, on constate que les traitements qu'elles utilisent sont pour la plupart basés sur un passage au domaine fréquentiel par transformée de Fourier, auquel on ajoute un pré-traitement

(fenêtrage, dérivation du signal), un post-traitement (2^{eme}transformée de Fourier, filtrage), et dont on récupère une valeur caractéristique (pic, maximum de vraisemblance).

2.1.2 Extraction de tempo

La description du rythme d'un extrait musical est une notion qui reste difficile à définir précisément, et l'étude des dimensions importantes du rythme fait l'objet d'études perceptives depuis de nombreuses années ([Cooper et Meyer, 1960], [Gabrielsson, 1973], [Handel, 1989], [Desain et Honing, 1999]). Globalement, les études de rythme visent à extraire une structure dynamique des événements musicaux importants d'un extrait sonore.

Dans le domaine du traitement du signal musical, la plupart des travaux concernent l'extraction du tempo et de la métrique d'extraits musicaux. Les méthodes les plus simples utilisent l'autocorrélation du signal audio pour la détection de la mesure ([Brown, 1993]), ou l'excitation d'oscillateurs non-linéaires pour le suivi du tempo (« beat tracking ») ([Large, 1995]). [Scheirer, 1998] propose une méthode complète de suivi de tempo, qui donne de très bons résultats sur des signaux de musique populaire au rythme prononcé, dont la figure 2.1 présente le schéma général.

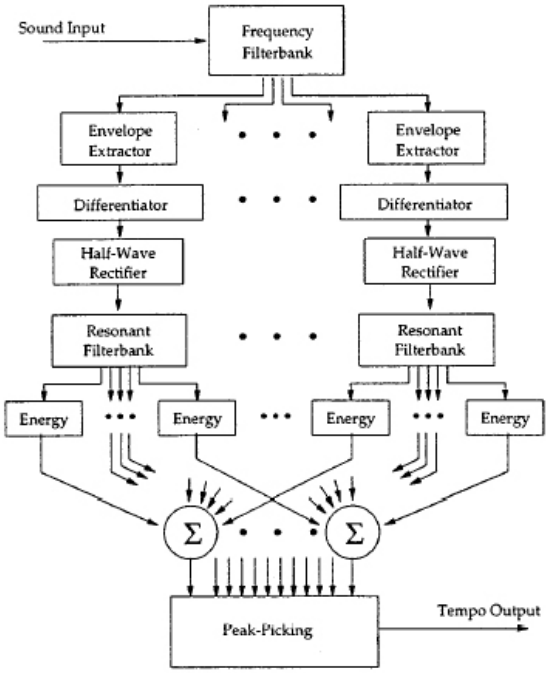


Fig. 2.1: Schéma de l'extracteur de tempo de [Scheirer, 1998]

Cette méthode est basée sur un suivi du tempo dans différentes bandes de fréquence par passage préalable du signal dans un banc de filtres. Pour chaque bande, le signal est passé dans un banc d'oscillateurs couvrant les fréquences de

60 à 240 bpm (beats par minute). Au final, en réalisant le cumul des énergies de chaque oscillateur pour toutes les bandes de fréquences, c'est l'oscillateur le plus excité qui indique la valeur du tempo.

D'autres algorithmes sont ensuite apparus afin de détecter également le rythme de musiques non-percussives, basés notamment sur des détections de ruptures. Ainsi, [Goto, 2001a] présente une méthode complète de suivi de tempo en temps réel sur des signaux percussifs ou non. Cette méthode utilise simultanément les détections :

- de ruptures énergiques (« onsets »), par étude des variations d'énergie spectrale sur plusieurs bandes de fréquences ;
- de ruptures mélodiques (changements d'accords), par étude des variations des fréquences dominantes du spectre ;
- d'éventuels motifs temporels de sons percussifs : les sons proches d'une grosse caisse de batterie sont détectés par les onsets sur les basses fréquences du signal, et ceux proches d'une caisse claire sont détectés à partir de la partie bruitée du signal.

L'ensemble se révèle assez performant, mais sur un petit ensemble de chansons tests.

Le principe de détection de ruptures est également utilisé par [Jensen et Andersen, 2003], dont la méthode offre de bonnes performances pour détecter le tempo à partir d'une sélection limitée de caractéristiques énergétiques et spectrales.

Une revue des principales techniques d'extraction de tempo est présentée dans [Alonso *et al.*, 2003b], qui introduit également une nouvelle technique utilisant la décomposition harmonique-bruit du signal, détaillée dans [Alonso *et al.*, 2003a].

2.1.3 Evaluation de l'énergie musicale

Nous avons réalisé l'étude d'un descripteur perceptif original : l'énergie musicale, notion qui vise à mesurer la sensation d'énergie perçue par un auditeur à l'écoute d'un extrait musical, indépendamment du volume sonore. Elle modélise la différence intuitive entre des extraits musicaux énergiques, comme du Punk-Rock avec des guitares saturées et un chant puissant, et des musiques calmes, comme des ballades folk accompagnées à la guitare acoustique. La construction empirique de ce descripteur est présentée dans la section 3.2, et sa modélisation automatique est présentée dans la section 5.3.5. Les résultats de cette étude sont publiés dans [Zils et Pachet, 2003].

2.2 Identification et classification sonore

Nous présentons ici les travaux de recherche en classification audio, qui ont porté principalement sur la reconnaissance d'instruments, la caractérisation des sons, notamment la voix, associée à la reconnaissance d'artistes, puis la classification en genre, et nouvellement la modélisation d'émotions musicales.

2.2.1 Classification de sons instrumentaux

Le *timbre* est une dimension fondamentale de la musique qui reste difficile à définir : c'est l'ensemble des caractéristiques acoustiques qui permettent de différencier deux sons, indépendamment de leur hauteur et de leur volume. La majorité des méthodes de classification de sons actuelles sont basées sur le calcul de ces caractéristiques timbrales à partir du signal audio et la représentation des sons dans l'espace paramétrique associé.

Le système Musclefish présenté dans [Wold *et al.*, 1996], est la première application utilisant ce principe pour la recherche de sons dans de grandes bases de données. Le timbre y est défini par un ensemble de caractéristiques pertinentes, temporelles comme le niveau sonore et l'attaque, ou spectrales comme la largeur de bande, la brillance(« brightness ») ou l'harmonicité.

Après que [Brown, 1997] a montré l'importance des coefficients cepstraux dans la reconnaissance et la définition du timbre des instruments de musique, des systèmes de segmentation et la classification en temps réel d'un flux sonore sont apparus, utilisant l'évolution temporelle des valeurs de fonctions caractéristiques, comme ceux présentés dans [Wold *et al.*, 1999] et [Rossignol *et al.*, 1999].

La norme MPEG7, qui vise à décrire des extraits multimédias par leur contenu, a permis de générer un ensemble de caractéristiques intéressantes utilisables dans le cadre des applications musicales, qui servent de référence en description musicale. Ces caractéristiques sont présentées dans les travaux de [Philippe, 2000], [Herrera *et al.*, 1999], et [Peeters *et al.*, 2000].

A partir de cette norme, et en utilisant des pré-filtrages et une étude des variations des fonctions caractéristiques, [Eronen, 2001] obtient de bons résultats sur la reconnaissance de six familles d'instruments.

De la même façon, [Brown *et al.*, 2001] a étudié la classification de sons d'instruments à vent, en montrant à nouveau l'importance des coefficients cepstraux.

Des travaux plus spécifiques ont été effectués sur les sons percussifs, avec notamment [Gouyon *et al.*, 2000] qui a montré l'importance des caractéristiques dérivées du ZCR (« Zero Crossing Rate » : taux de passage à zéro) dans la classification de sons percussifs. A partir de ces caractéristiques et d'autres caractéristiques spectrales, le même auteur présente dans [Gouyon et Herrera, 2001] une classification de sons percussifs utilisant une analyse linéaire discriminante et des arbres de décision. Nous avons présenté dans [Zils *et al.*, 2002] une méthode alternative originale basée sur une recherche itérative, qui permet d'extraire les sons percussifs principaux d'un extrait musical à partir de sons synthétiques, et d'en déduire ainsi le rythme « percussif ». Cette méthode est présentée en détails dans la section 3.1. Enfin [Gillet et Richard, 2004] présente une classification de sons percussifs utilisant les MFCC et des caractéristiques spectrales, avec des classifieurs HMM et SVM, fournissant 83,9% de réussite sur une taxonomie simplifiée de 8 types de sons.

On distingue dans les travaux les plus récents deux enjeux principaux dans la caractérisation du timbre musical. Le premier enjeu est l'extraction et la sélection de fonctions caractéristiques pertinentes. En effet, si certains travaux utilisent un ensemble de paramètres spécifiques au problème étudié, d'autres partent d'un très grand nombre de caractéristiques, avant de sélectionner automatiquement les plus pertinentes par des méthodes de sélection de variables. On pourra se référer aux études des performances de diverses méthodes de sélection

de variables appliquée à la musique ([Peeters et Rodet, 2002]) ou en règle générale ([Guyon et Elisseeff, 2003]). Le deuxième enjeu est l'obtention d'une classification performante. Dans cette optique, plusieurs travaux récemment publiés présentent une revue comparative des différentes techniques de sélection de fonctions caractéristiques et modèles de classification de sons, notamment [Herrera *et al.*, 2000], [Tzanetakis, 2002], [Herrera *et al.*, 2002a], et [Herrera *et al.*, 2002b] pour les sons percussifs.

Enfin, [Eggink et Brown, 2003] aborde la classification poly-instrumentale, et utilise la théorie du paramètre manquant (« missing feature theory », utilisée avec de bonnes performances en reconnaissance vocale) afin de reconnaître des instruments différents émettant des sons simultanément, avec des performances qui restent cependant assez moyennes.

2.2.2 Caractérisation perceptive des sons

A partir des caractéristiques utilisées pour la simple classification d'instruments, la définition de caractérisations sonores de plus haut niveau permet de réaliser la segmentation d'un flux audio non plus d'un point de vue uniquement sonore, mais d'un point de vue structurel. Ainsi, [Saunders, 1996] réalise un discriminateur voix parlée / musique applicable sur des programmes radiophoniques, qui utilise les fonctions caractéristiques traditionnelles de la description musicale, dans des modèles gaussiens. Ce système est repris et amélioré par [Scheirer et Slaney, 1997], [Carey et Lloyd-Thomas, 1999] et [Williams et Ellis, 1999] avec des paramètres plus spécifiques.

Puis, d'un point de vue plus musical, [Rossignol *et al.*, 1999] aborde la discrimination entre voix parlée, voix chantée, et passages instrumentaux à partir de fonctions caractéristiques simples. Ce travail est repris dans [Chou et Gu, 2001], qui réalise en premier lieu une détection de voix chantée par l'utilisation de caractéristiques harmoniques, et applique ensuite une deuxième discrimination sur les parties non-chantées pour déterminer si elles sont de la parole ou de la musique. La première étape de ce système est améliorée par [Berenzweig et Ellis, 2001], en utilisant les MFCC pour construire des fonctions caractéristiques de phonèmes de langue anglaise, générées par un réseau de neurones. L'emploi de ces fonctions dans des modèles de Markov cachés afin de réaliser des modèles de voix chantée, voix parlée, et absence de voix basés sur l'excitation de ces phonèmes, fournit 80% de bonnes classifications sur 100 extraits de 15s de programmes radiophoniques.

Au delà de la simple détection de voix, le même auteur présente dans [Berenzweig *et al.*, 2002] un système d'identification d'artistes qui fournit entre 50 et 65% de réussite, en détectant préalablement les passages chantés, et en utilisant ensuite les MFCC avec un perceptron multicouches comme classifieur. De même, [Kim et Whitman, 2002] réalise un système de reconnaissance de chanteur à partir du système de reconnaissance d'artiste de [Whitman *et al.*, 2001], basé sur des fonctions de prédiction linéaire et un classifieur SVM appliqué sur les passages vocaux, qui obtient 45% de réussite sur une base de 250 chansons et 17 artistes. Dans [Kim et Whitman, 2003] un système de reconnaissance de chanteur a capella basé sur la construction de modèles phonétiques à l'aide des MFCC et une classification utilisant des modèles de Markov cachés, donne 90% de réussite. Un autre système de reconnaissance de chanteur, présenté dans [Zhang, 2003], obtient 82% de réussite sur 45 chansons par 8 artistes, à partir de

coefficients de prédiction linéaire et de mélanges de gaussiennes. Dans [Bartsch et Wakefield, 2004] la même tâche est réalisée à l'aide de caractéristiques d'enveloppe spectrale apprises sur 12 chanteurs classiques, et donne entre 70 et 80% de réussite avec un classifieur quadratique, sur des extraits de chant avec un léger accompagnement au piano. Enfin, [Tsai *et al.*, 2003] ajoute la caractérisation de la voix chantée elle-même à partir de coefficients cepstraux, afin de réaliser une classification des types de voix de manière non-supervisée.

En marge des travaux de classification, [Qi *et al.*, 2002] présente une méthode intéressante de recherche de sons par l'exemple dans de grandes bases, à partir de caractéristiques acoustiques, qui permet de réaliser une caractérisation plus sémantique des sons. En effet, ce système réalise une description adaptive du son recherché en ajustant les poids des différentes caractéristiques, à partir des valeurs caractéristiques des sons désignés les plus proches par un ensemble d'auditeurs. Ainsi, à partir d'une taxonomie donnée, le système réalise un apprentissage sémantique des sons, en déterminant la combinaison de fonctions caractéristiques optimale permettant de définir un son par un mot (par exemple « métallique »).

2.2.3 Genre et Timbre global d'un extrait musical

En généralisant ces méthodes au niveau global (un extrait ou un titre musical entier), la définition d'un « timbre global »permet d'approcher la notion de genre et de réaliser ainsi des classifications ou comparaisons entre titres musicaux. [Foote, 1997] présente une première utilisation des coefficients cepstraux pour caractériser et retrouver des extraits musicaux de quelques secondes. Puis [Lambrou *et al.*, 1998] présente une première classification d'extraits musicaux en 3 genres (Rock, Piano, Jazz), utilisant le protocole classique d'extraction, de sélection de fonctions caractéristiques, et de classification, à partir de coefficients de transformation en ondelettes. Il obtient ainsi de bonnes performances, mais sur une base de test de petite taille. La technique des ondelettes est reprise et perfectionnée dans [Li *et al.*, 2003], qui obtient 80% de réussite sur une classification de 1000 extraits musicaux en 10 genres différents.

Après que [Logan, 2000] ait montré l'importance d'utiliser l'échelle Mel et les coefficients cepstraux pour la modélisation des genres musicaux, [Tzanetakis *et al.*, 2001b] utilise ces caractéristiques en plus des caractéristiques usuelles, pour réaliser une classification hiérarchique d'extraits musicaux. Ainsi, il utilise des caractéristiques différentes suivant qu'il discrimine les genres principaux (Classique, Disco, Country, Jazz, Rock, Hiphop), ou les sous-genres de la catégorie « Classique »(Orchestre, Piano, Chorale, Cordes). Les performances en classification vont de 50% à 90% suivant les genres. Il obtient des performances équivalentes en utilisant la transformée en ondelettes dans [Tzanetakis *et al.*, 2001a].

Egalement à partir des coefficients cepstraux, [Aucouturier et Pachet, 2002a] présente une méthode originale utilisant une représentation par mélange de gaussiennes, qui donne d'intéressants résultats en matière de similarité timbrale.

Récemment, [Berenzweig *et al.*, 2003] et [Aucouturier et Pachet, 2003] ont présenté des études comparatives des nombreux travaux en caractérisation des genres musicaux, à partir de diverses caractéristiques et méthodes de classification, confirmant à nouveau l'importance des coefficients cepstraux, et soulignant les bonnes performances de la méthode de classification des plus proches voi-

sins. Ce dernier, en évoquant l'éventuelle utilisation de techniques collaboratives (« datamining »), soulève également les problèmes liés à la représentation même du genre (taxonomie), et à son aspect subjectif.

2.2.4 Reconnaissance des émotions musicales

Jusqu'à présent, très peu de travaux ont porté sur des descripteurs subjectifs dans le domaine musical, et tous les travaux citent comme référence les travaux effectués en reconnaissance vocale, notamment sur la reconnaissance des émotions.

Un des plus récent est [Oudeyer, 2002], qui concerne l'interaction émotionnelle avec les robots. Il présente un état de l'art complet des fonctions et algorithmes de classification utilisées pour la reconnaissance d'émotions dans la voix parlée, et propose une méthode qui fournit plus de 90% de réussite pour la distinction entre quatre états émotionnels : calme, énervement, joie, tristesse. Il a notamment montré que les caractéristiques liées aux basses fréquences sont très informatives quant à l'émotion vocale.

Dans le domaine musical, les principaux travaux ont porté sur la manipulation de représentations symboliques de la musique, comme par exemple les fichiers MIDI. [Katayose *et al.*, 1988] fut le premier à présenter une extraction automatique de sentiments dans la musique pop, mais à partir de caractéristiques musicales transcrites manuellement, concernant la mélodie, le rythme, et l'harmonie.

Récemment, l'explosion de la quantité de musique accessible sur internet, et le besoin de données musicales plus sémantiques pour sa distribution (mis en évidence par des experts comme [Huron, 2000]), a remis ce sujet au goût du jour. [Liu *et al.*, 2003b] a présenté un nouveau système utilisant des caractéristiques extraites automatiquement de fichiers symboliques MIDI (tempo, niveau sonore, changements de hauteur de note, densité de notes, et timbre), pour classer les valses de Strauss en 5 clusters non-supervisés.

Puis [Li et Ogihara, 2003] a tenté une classification d'extraits musicaux de divers genres en 10 adjectifs (les groupes de Farnsworth), à partir de caractéristiques acoustiques timbrales, rythmiques, et mélodiques, extraites du signal audio, dont les performances restent modestes.

En parallèle, le travail le plus intéressant est [Liu *et al.*, 2003a], qui présente le premier véritable extracteur d'émotion musicale à partir du signal audio. Celui-ci est basé sur le modèle émotionnel de Thayer ([Thayer, 1989]), qui représente l'état d'esprit en deux dimensions : le stress et l'énergie, qui mènent à quatre émotions : la joie, la tristesse, l'anxiété, et l'exubérance.

Il classe des extraits de 20 secondes de musique classique dans l'une de ces quatre émotions, en utilisant des caractéristiques de timbre (forme et contraste spectraux), de rythme (corrélation des basses fréquences), et d'intensité sonore du signal. Avec un apprentissage de modèles gaussiens pour chaque émotion à partir de ces caractéristiques, le système fournit, sur une base de 800 extraits, entre 75% et 94% de réussite en validation croisée.

2.3 Recherches complémentaires à la description de signaux musicaux

Nous présentons ici les deux principaux domaines de recherche associés aux descripteurs musicaux : la segmentation de documents audio, et plus globalement la réalisation de systèmes d'écoute musicale.

2.3.1 Segmentation et analyse de flux audio

En appliquant l'étude du timbre à un flux musical, on peut réaliser des segmentations de flux audio basées sur les variations timbrales. Ainsi, la plupart des recherches présentées précédemment en caractérisation de timbre sonore sont applicables à la segmentation de documents audio : discrimination entre parole et musique ([Scheirer et Slaney, 1997], [Williams et Ellis, 1999]), ou détection de voix chantée ([Berenzweig et Ellis, 2001], [Chou et Gu, 2001]).

D'autres travaux sont plus focalisés sur la segmentation proprement dite. Les premières segmentations structurelles de flux audio, par [Kimber et Wilcox, 1996], puis [Liu *et al.*, 1998] et [Zhang et Kuo, 1999], sont réalisées sur des enregistrements radiophoniques à partir des variations de caractéristiques simples. [Laurent *et al.*, 1998] montre qu'on obtient une segmentation musicale robuste à partir des variations brusques du spectrogramme d'un signal audio.

En se focalisant sur la segmentation musicale, [Rossignol *et al.*, 1999] présente une segmentation sur trois niveaux, à partir de caractéristiques simples : reconnaissance de source sonore (voix parlée, chant, instrumentaux), segmentation paramétrique (en fonction de l'harmonicité, de la présence de voix, de l'énergie sonore, etc...), et segmentation finale en notes ou sons stables.

Puis [Tzanetakis et Cook, 1999] aborde la représentation de la structure d'extraits musicaux, à partir d'une segmentation détectant les changements de texture sonore. Il compare les performances de son système à des segmentations réalisées manuellement par des utilisateurs sur des extraits d'une minute de divers genres musicaux, et montre notamment la robustesse d'une segmentation automatique par rapport à des segmentations « humaines ».

Parallèlement, en définissant une mesure de nouveauté audio(« Audio Novelty »), [Foote, 2000] réalise une segmentation par analyse de la similarité locale, qui lui permet d'introduire la notion de résumé sonore à partir d'une matrice de similarité timbrale. [Chu et Logan, 2000] produit de la même façon une segmentation puis un résumé audio en utilisant des modèles de Markov cachés. En utilisant des modèles de Markov avec des caractéristiques d'enveloppe spectrale et les coefficients cepstraux, [Aucouturier et Sandler, 2001]) présente une segmentation structurelle non-supervisée qui extrait des textures sonores représentatives de la présence de voix et d'instruments. Enfin, [Peeters *et al.*, 2002] réutilise ces modèles de Markov avec des fonctions caractéristiques de la dynamique acoustique, afin de réaliser des résumés sonores.

Il existe également des méthodes de segmentation qui n'utilisent pas de descripteurs acoustiques, principalement basées sur la détection d'attaques (« onsets »), comme par exemple celles présentées dans [Klapuri *et al.*, 2001] et [Duxbury *et al.*, 2001].

2.3.2 Systèmes d'écoute musicale

Il est difficile de séparer complètement les différents domaines de recherche en description de signaux musicaux, tant les différents problèmes abordés sont liés les uns aux autres. Par exemple, la description d'objets sonores est étroitement liée à la segmentation d'un flux audio, les deux domaines se servant mutuellement. En effet, un flux audio de longue durée est généralement préalablement segmenté en notes (par exemple par détection de transitoires) pour décrire des sons au niveau local, et éventuellement en déduire une description globale. Inversement, les variations locales des descripteurs sonores pourront permettre de réaliser une segmentation permettant une indexation globale de l'extrait audio par le contenu.

Les travaux les plus récents tentent d'intégrer l'ensemble des travaux réalisés, tant dans le domaine des descripteurs que dans celui des outils de traitement du signal musical, dans des systèmes d'écoute globaux capables d'analyser automatiquement des scènes sonores, simultanément des points de vue rythmique, mélodique, structurel, etc...Ces systèmes, introduits par [Bregman, 1990] et généralisés par [Ellis, 1996] puis [Scheirer, 2000], offrent diverses possibilités d'applications, comme la transcription automatique de mélodies ([Karjalainen et Tolonen, 1999], [Klapuri *et al.*, 2001], ou [Martins et Ferreira, 2002] pour le format MIDI), la performance musicale assistée ([Suzuki *et al.*, 2002], [Pachet, 2002]).

2.4 Conclusion

Bien qu'il n'existe pas de méthode universelle pour extraire des descripteurs musicaux, on peut distinguer deux types de techniques d'extraction automatique de descripteurs musicaux : empirique et paramétrique.

Dans le cas d'une méthode « empirique », on met en oeuvre une chaîne de traitements du signal aboutissant à un système d'extraction complexe (extracteur de tempo [Scheirer, 1998], modèle de timbre global [Aucouturier et Pachet, 2002a], transcription polyphonique [Klapuri, 2000]).

Bien que les fonctions construites par ces méthodes utilisent en général des traitements courants du signal audio, comme des corrélations, spectres, enveloppes (études mélodiques), oscillateurs (études rythmiques), filtrages, etc..., leur difficulté réside à la fois dans le choix de traitements efficaces des signaux, et dans le long travail de mise au point et d'ajustage des paramètres.

Dans le cas d'une méthode « paramétrique », utilisées pour la plupart des études de timbre, on utilise toutes sortes de caractéristiques temporelles, spectrales ou cepstrales, ainsi que leurs statistiques (moyenne, variance, moments, etc...). Les données à traiter sont ensuite projetées dans l'espace des caractéristiques pour être analysées (extraction de tempo [Jensen et Andersen, 2003], classification de sons instrumentaux [Herrera *et al.*, 2002a], classification de titres musicaux [Tzanetakis *et al.*, 2001b], détection de voix [Berenzweig et Ellis, 2001]). La difficulté de ces méthodes est de trouver un espace de caractéristiques pertinent, qui décrit suffisamment bien le problème posé.

Pour cela, on utilise habituellement les fonctions caractéristiques comme celles de la norme MPEG7. Ces fonctions sont robustes, dans la mesure où elles correspondent à des propriétés générales des signaux musicaux. Cepen-

dant, elles sont souvent insuffisantes pour modéliser des problèmes descriptifs plus spécifiques. C'est pourquoi la plupart des travaux utilisent ces fonctions comme ensemble de caractéristiques de base, et leur ajoutent un ou plusieurs descripteurs synthétisés empiriquement, liés spécifiquement au problème descriptif à résoudre. Les tableaux en annexe B récapitulent les principales fonctions caractéristiques et techniques de modélisation utilisées dans les travaux présentés dans ce chapitre.

Pour étudier plus précisément ces différentes techniques d'extraction de descripteurs et la synthèse de fonctions caractéristiques spécifiques, nous avons réalisé de manière empirique plusieurs descripteurs simples de musiques et de sons, présentés en détails dans le prochain chapitre.

3. CONSTRUCTION EMPIRIQUE DE DESCRIPTEURS MUSICAUX

La construction de descripteurs musicaux efficaces repose sur l'utilisation de fonctions robustes. Ces fonctions peuvent être des fonctions d'*extraction* complètes, ou bien seulement des fonctions *caractéristiques* utilisées pour représenter des propriétés musicales spécifiques, combinées lors de l'apprentissage de modèles plus élaborés.

Les fonctions caractéristiques de type Mpeg7 constituent une base de propriétés musicales générales. Pour résoudre des problèmes de description musicale spécifiques, il est nécessaire d'utiliser également des fonctions caractéristiques adaptées, que le chercheur doit construire de manière empirique. C'est donc naturellement que nous avons débuté l'étude de l'extraction de descripteurs musicaux par la synthèse empirique de fonctions de traitement du signal musical adaptées à des problèmes descriptifs spécifiques.

Notre contexte d'étude, la description de musique populaire dans le cadre de la distribution en ligne, nous a donné quelques idées de départ simples :

- les musiques populaires étant en majorité rythmées par l'utilisation d'instruments percussifs, nous avons abordé l'étude du rythme par l'étude des sons percussifs : construction d'une fonction complète d'extraction de sons percussifs caractéristiques d'un extrait musical, et fonction de « percussivité » d'un son ;
- d'autre part, les musiques populaires couvrent de nombreux genres et sous-genres qui dépendent souvent de la taxonomie utilisée ; devant l'impossibilité de trouver une taxonomie universelle, nous avons voulu aborder la discrimination des extraits musicaux par un concept plus général et perceptif : l'énergie musicale ressentie à l'écoute de l'extrait ;
- enfin, le chant étant souvent prépondérant, nous avons abordé l'étude de timbre en cherchant une fonction caractéristique de la présence de voix chantée dans un extrait musical.

3.1 Extraction du rythme percussif

La notion de rythme d'un extrait musical a été principalement abordée par l'extraction du tempo et de la métrique. Cependant, bien que fondamentales, ces caractéristiques ne sont pas vraiment suffisantes pour représenter pleinement les subtilités rythmiques de musiques, notamment dans le cadre d'une recherche de musique populaire. En effet, pour donner un exemple simple, la grande majorité des titres d'un catalogue de musique Rock ont approximativement le même tempo (100bpm) et la même métrique (4/4).

Pour mener une étude rythmique plus précise, nous nous sommes basés sur les travaux de [Pachet *et al.*, 2000], qui reposent sur deux hypothèses de départ concernant les musiques populaires :

- la perception du rythme est corrélée aux occurrences répétées de sons accentués ; nous nous focalisons sur les répétitions à court terme (sons percussifs, notes successives jouées par un instrument), mais cette répétition se retrouve également à moyen terme (phrases, grilles d'accords) et à long terme (couplets, refrains) ;
- dans la grande majorité des titres, le rythme est soutenu par des sons de batterie ou d'autres instruments percussifs, dont on peut distinguer deux groupes principaux : les sons graves (par exemple une grosse caisse de batterie), et les sons aigus (par exemple une caisse claire de batterie).

Ainsi, à partir de ces hypothèses, cette approche consiste à extraire d'un morceau de musique les occurrences de ces deux types de sons percussifs (grave et aigu), afin de reconstituer une représentation rythmique binaire, appelée « Drum track ». Plus précisément, le problème est le suivant : étant donné un extrait de musique polyphonique, nous voulons trouver les positions des sons percussifs significatifs, et identifier les sons percussifs eux-mêmes, sans connaissance préalable sur ces derniers.

3.1.1 Extraction d'un son percussif caractéristique

Pris isolément, les signaux de sons percussifs sont facilement caractérisables, notamment par leur pic d'énergie et leur contenu fréquentiel. Cependant, dès qu'on les étudie dans le contexte d'un extrait de musique populaire polyphonique, la détection et l'extraction de ces sons devient particulièrement difficile à réaliser, du fait de leur brièveté et de leur non-stationnarité, dans un environnement également extrêmement non-stationnaire. Par exemple, [Duxbury *et al.*, 2001] montre que la détection des parties non-stationnaires dans un signal polyphonique extrait aussi bien les sons percussifs que les attaques de notes des autres instruments.

Nous avons perfectionné la méthode initiale d'extraction de [Pachet *et al.*, 2000], qui utilise les résultats de [Gouyon *et al.*, 2000] sur la caractérisation des sons percussifs, et qui est basée sur les seules hypothèses que nous pouvons faire, c'est-à-dire que nous recherchons des sons :

- courts : la durée des sons percussifs est de l'ordre de quelques dizaines de millisecondes ;
- non-stationnaires : les sons percussifs originaux de l'extrait musical étant a priori inconnus, la méthode doit être basée sur un modèle très général ;
- énergiques : la position des sons percussifs est souvent marquée par un pic d'énergie local dans le signal ;
- répétés : nous utilisons la corrélation du signal avec ces modèles percussifs afin de détecter les occurrences répétées.

L'utilisation de la corrélation, appliquée brutalement sur des sons percussifs acoustiques, est inefficace du fait de leur niveau de bruit élevé. Cependant, notre méthode permet d'améliorer son efficacité en supprimant une grande partie de ce bruit, par le moyennage des extraits sonores et l'utilisation de sons très courts, où les premiers modes de vibrations de instruments restent prépondérants.

La méthode consiste à identifier progressivement le son percussif à extraire, de manière itérative pendant le processus d'analyse :

1. Extraction préalable des pics d'énergie du signal
   Afin de réduire la quantité d'information à traiter, nous extrayons préalablement les positions de tous les pics locaux d'énergie à très court terme dans l'extrait musical à traiter, qui sont des positions possibles de sons percussifs. Cette méthode est très utile pour extraire les sons principaux, mais peut être trop restrictive pour extraire les sons secondaires noyés dans du bruit.
2. Son percussif initial
   La méthode débute par le chargement d'un son percussif synthétique simple, du type recherché (grave ou aigu). Afin, d'obtenir le modèle le plus général possible, nous utilisons la réponse impulsionnelle d'un filtre passe-bande.
3. Extraction des occurrences
   On recherche dans le signal de l'extrait musical les occurrences les plus plausibles de ce son synthétique. Pour cela, nous extrayons les pics de corrélation entre l'extrait musical et le son percussif. Cette méthode permet de repérer de manière simple et assez efficace, une majorité des occurrences du son recherché.
4. Qualité des occurrences
   Nous évaluons ensuite la qualité des occurrences extraites, afin de filtrer les éventuelles erreurs de détection, en utilisant des seuils sur plusieurs critères :
   – la proximité du pic de corrélation avec un pic d'énergie du signal
   – l'amplitude du pic de corrélation
   – l'énergie locale relative
   – le taux de passage à zéro ou ZCR (« Zero-crossing rate ») : ce paramètre est performant pour discriminer les différents types de sons percussifs pendant l'extraction (voir [Gouyon *et al.*, 2000])

   Ces critères nous permettent d'éliminer les mauvaises occurrences. Cependant, certaines occurrences correctes peuvent avoir été manquées, et d'autres mauvaises peuvent avoir été trouvées, du fait de la faible spécification du son percussif. Afin de définir plus précisément la recherche à effectuer, un nouveau son percussif à retrouver est synthétisé à partir des occurrences détectées précédemment.
5. Synthèse d'un nouveau son percussif à rechercher
   C'est l'idée principale de cette approche : on affine itérativement le son à rechercher. Le nouveau son est réalisé par un mixage de toutes les occurrences trouvées précédemment, en plus du son de synthèse initial, tous les sons utilisés étant centrés et synchronisés.
6. Itération jusqu'à convergence
   Le schéma précédent (synthèse d'un son à rechercher, recherche des occurrences possibles, filtrage des occurrences, [synthèse d'un nouveau son à partir des occurrences précédentes, etc...]) est répété jusqu'à ce que le système converge vers une solution stable, c'est-à-dire que les occurrences trouvées n'évoluent plus entre deux itérations successives. On considère alors que toutes les occurrences détectables ont été trouvées. Le système fournit alors comme résultat un son percussif (le dernier son synthétisé) et ses positions dans l'extrait musical analysé.

3.1.2 Extraction de « Drum track »

Afin d'obtenir une représentation binaire du rythme (du type « grosse caisse / caisse claire »), le système est lancé deux fois avec des sons initiaux différents : un son grave synthétisé à partir de la réponse impulsionnelle d'un filtre passe-bas, puis un son aigu à partir de celle d'un passe-bande. La première extraction consiste donc à trouver les sons du type « grosse caisse », et la seconde les sons de type « caisse claire », dont les positions ne sont pas conflictuelles avec les premiers, la priorité étant donnée au son grave en cas d'occurrences simultanées.

3.1.3 Performances

Nous avons réalisé une évaluation systématique des performances du système sur une base de 100 extraits musicaux variés, contenant de 10 à 20 secondes de musique populaire percussive de divers genres (pop, rock, dance, jazz, hip-hop), contenant des sons percussifs d'origines diverses (batterie, percussions africaines, sons électroniques).

Tout d'abord, nous avons évalué manuellement la difficulté à extraire les sons percussifs de ces titres :

- 20% des titres contiennent des sons percussifs prédominants au rythme régulier, mixés avec d'autres instruments ou voix calmes, par exemple certaines chansons de pop acoustique : ces titres sont considérés comme faciles à traiter ;
- 60% des titres contiennent des sons percussifs réguliers, au même niveau que les autres instruments : ces extraits, constituant la majorité des titres de musique populaire, sont considérés comme traitables ;
- 20% des titres contiennent des sons percussifs dont les occurrences ne sont pas toujours évidentes (rythme irrégulier, ou noyage derrière les autres instruments), comme dans certaines chansons de jazz ou de rock bruyant : ils sont considérés comme difficiles à traiter.

Nous avons lancé le système sur ces extraits, puis déterminé si le résultat était satisfaisant en le classant dans une de ces catégories :

- parfait : on obtient deux sons très similaires aux deux sons percussifs principaux de l'extrait, ainsi que l'ensemble de leurs occurrences ;
- acceptable : on obtient deux sons très similaires aux deux sons percussifs principaux de l'extrait, et une majorité des occurrences est correcte, la structure rythmique étant conservée ;
- semi-acceptable : on obtient correctement un seul des deux sons percussifs principaux, avec une majorité d'occurrences correctes ;
- mauvaise : on n'obtient aucun des deux sons percussifs, ou les occurrences ne représentent pas la structure rythmique du morceau original.

Ainsi, pour les morceaux non-difficiles, nous obtenons environ 50% de résultats parfaits, et 25% de résultats acceptables, soit environ 75% de résultats utilisables pour une analyse ultérieure. Pour les morceaux difficiles, nous obtenons 40% de résultats corrects, ce qui est une performance acceptable. La plupart des mauvais résultats sont dus à :

- 35% : occurrences dont la présence n'est pas évidente, comme par exemple dans des passages subtils de jazz, dont l'extraction nécessiterait un traitement du signal plus spécifique. Cependant, l'extraction de toutes les occurrences percussives de ces morceaux n'est sans doute pas la meilleure

façon de représenter leur rythme.
- 15% : confusion entre la voix de la chanteuse et du son percussif aigu, dû au moyennage du signal lors de la synthèse du nouveau son percussif, qui en supprimant son bruit de haute-fréquence, le rend plus harmonique et proche d'une voix humaine aigüe. En introduisant un critère de durée du son, on pourrait sans doute distinguer les sons percussifs des syllabes chantées (plus longues).
- 10% : confusion entre les deux sons percussifs, qui arrive souvent lorsque l'extrait contient des occurrences simultanées. L'utilisation d'une troisième catégorie de sons (grave et aigu simultanés) est impossible avec cette méthode, car elle rendrait le critère discriminatif ZCR inutile.
- 10% : le son percussif aigu n'est pas détecté, généralement car il est trop spécifique pour le modèle synthétique initial.
- 10% : le bruit dans le signal est trop élevé, et les sons percussifs sont noyés.

Ce descripteur permet par exemple de calculer des similarités percussives entre différents morceaux polyphoniques, par comparaison des sons percussifs eux-mêmes (similarité timbrale), et de leur occurrences (similarité rythmique). Notre implémentation de la méthode d'extraction et l'ensemble de ces résultats sont présentés dans [Zils *et al.*, 2002].

3.2 Modélisation de l'énergie musicale

Actuellement, la caractérisation d'extraits musicaux est souvent faite en terme de données éditoriales, comme le nom de l'artiste, ou le genre musical. Ces données posent problème dans le cadre de recherche de musique, dans la mesure où elles sont dépendantes de connaissances (nom) ou d'une structuration (taxonomie de genre) par un expert musical. Afin de se libérer de ces contraintes d'expertise, nous avons voulu aborder la caractérisation d'extraits musicaux suivant une dimension perceptive universelle : l'impression d'énergie ressentie à l'écoute de l'extrait, indépendamment du niveau d'écoute.

Les travaux liés à ce descripteur, introduits dans l'état de l'art (section 2.1.3), ont été publiés dans [Zils et Pachet, 2003]. Ils consistent en une modélisation du concept d'énergie musicale à partir de tests perceptifs, puis à la recherche dans le signal musical de propriétés associées.

3.2.1 Évaluation par tests perceptifs

D'après nos premières observations, la notion d'énergie musicale, bien que liée à la perception de chacun, semblait perçue de façon approximativement équivalente par tous les auditeurs. Afin de vérifier la pertinence de cette hypothèse, nous avons réalisé des tests perceptifs auprès d'un groupe d'auditeurs, visant à évaluer l'énergie musicale d'un ensemble d'extraits musicaux. Les tests ont porté sur 2 bases de 200 extraits musicaux normalisés en niveau sonore. Ces extraits, d'une durée de 10s, couvrent a priori tout l'éventail des énergies possibles, dans divers genres musicaux, plus particulièrement axés sur la musique populaire. Le test consiste pour les auditeurs à évaluer l'énergie qu'ils perçoivent à l'écoute de chaque extrait musical, de « Très Faible »à « Très Elevée »(voir l'interface du test sur la figure 3.1).

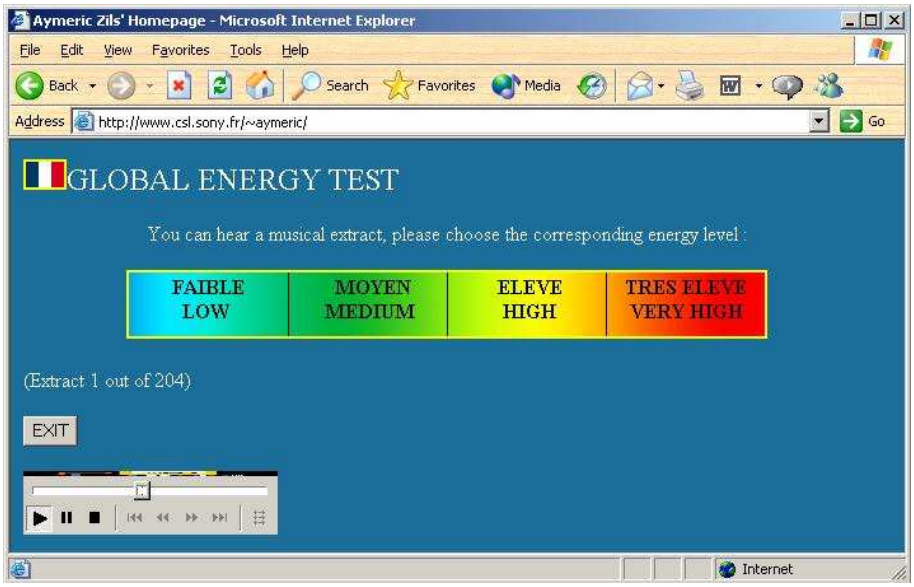


Fig. 3.1: Interface du test d'évaluation de l'Energie Globale d'extraits musicaux

Ce test a été réalisé à l'intérieur de notre laboratoire et était également accessible sur Internet. Nous avons obtenu 4500 réponses, ce qui correspond à une moyenne de 10 à 12 réponses pour chaque extrait musical. Pour chaque extrait, les valeurs extrêmes sont ignorées, et une analyse statistique des résultats montre que, pour 98% des titres, l'écart type des réponses est inférieur à la distance entre 2 catégories d'énergie. Ainsi, l'évaluation de l'énergie globale, tout en étant subjective, reste consensuelle, ce qui justifie le fait de chercher à extraire cette information à partir du signal audio, indépendamment de l'auditeur. Les résultats du test sont ensuite normalisés et conservés afin de servir de base à la construction du modèle de descripteur.

3.2.2 Modélisation des résultats des tests perceptifs

Suite à ces tests, nous disposons de 200 signaux d'extraits musicaux étiquetés par une mesure générale de leur « énergie musicale ». La recherche empirique du modèle consiste à trouver des traitements des signaux fournissant les caractéristiques les plus corrélées aux niveaux d'énergie associés. La visualisation des divers signaux (figure 3.2) montre que l'évaluation n'est pas triviale, et que des extraits dont les signaux sont visiblement équivalents peuvent avoir des énergies musicales très différentes, et inversement.

En l'absence de piste de départ évidente, nous avons utilisé comme caractéristiques initiales les fonctions de la norme MPEG7, couramment utilisées en classification sonore. Ces fonctions mesurent notamment l'énergie du signal, les statistiques spectrales, les variances associées, etc.... Nous avons ajouté une valeur de tempo extraite par la méthode de [Scheirer, 1998], pour obtenir en tout une trentaine de paramètres.

On évalue ensuite la corrélation entre chacune de ces caractéristiques et les

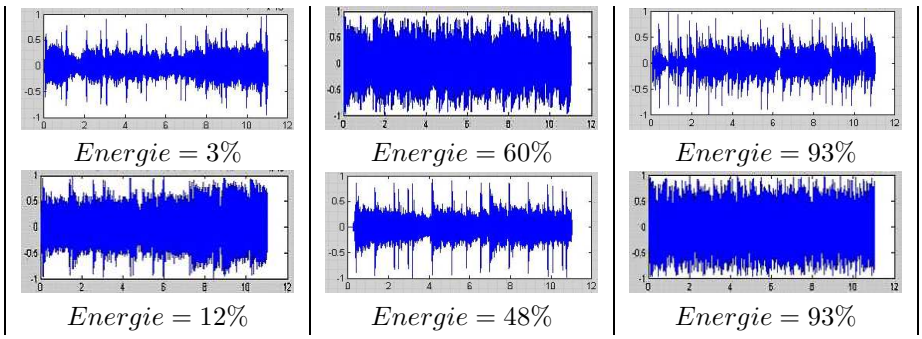


Fig. 3.2: Visualisation de signaux d'extraits musicaux d'énergies diverses

niveaux d'énergie musicale associés. On a ainsi montré que l'énergie perçue est assez corrélée à certaines caractéristiques, comme le tempo, ou l'énergie brute du signal (normalisé en amplitude maximale), qui peut être interprété comme une « densité »sonore. Cependant, la dispersion sur ces seules caractéristiques reste encore trop importante, comme on peut le constater sur la figure 3.3.

Nous avons ensuite testé manuellement divers traitements appliqués sur les fonctions caractéristiques les plus corrélées, et sommes arrivés à une fonction simple et assez pertinente : $Logvardiff = Log(Variance(\frac{dx^2}{dt}))$, dont le modèle linéaire donne une erreur moyenne de 15% par rapport aux valeurs étiquetées. Sur la figure 3.3, qui montre les fonctions des différentes caractéristiques en fonction de l'énergie musicale, on constate visuellement que Logvardiff produit une répartition approximativement linéaire de l'énergie, le tempo et l'énergie du signal également dans une moindre mesure, tandis que la fréquence centrale n'y est pas du tout corrélée (on trouve tous les niveaux d'énergie musicale à toutes les fréquences).

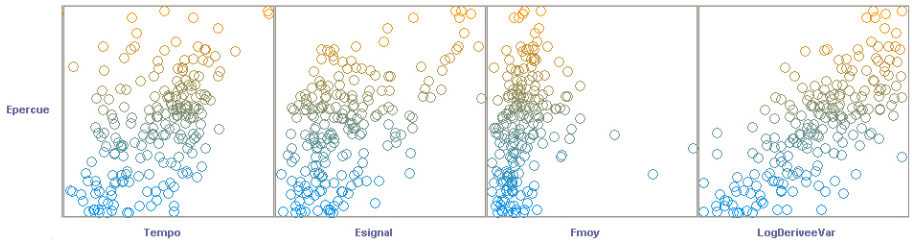


Fig. 3.3: Graphiques de correspondance entre l'énergie musicale (« E perçue ») et diverses caractéristiques du signal (Tempo, énergie brute « ESignal », fréquence centrale « FMoy », notre fonction Logvardiff « LogDeriveeVar »)

On peut interpréter la formule Logvardiff d'un point de vue physique de la façon suivante : l'extrait apparaît d'autant plus énergique que les variations temporelles de son énergie sont variées. Cette formule tend à reconnaître comme énergiques les extraits contenant des occurrences de percussions et des changements d'instrumentation fréquents, importants, et éventuellement irréguliers.

Plus tard, notre système d'extraction automatique de caractéristiques per-

tinentes nous permettra de retrouver cette formule et un ensemble de fonctions dérivées de celle-ci (voir la section 5.3.5).

3.3 Construction de descripteurs basiques à partir de fonctions caractéristiques simples

Dans le cadre de l'application Musaicing (voir section 6.4), nous avons voulu construire rapidement des descripteurs simples et rapides à calculer permettant d'évaluer deux propriétés de brefs extraits sonores polyphoniques : leur percussivité (présence ou ressemblance à des percussions) et leur vocalité (présence ou ressemblance à la voix chantée). Pour cela, nous avons réalisé le même type d'étude que pour l'énergie : étiquetage d'une base de sons grâce à des tests perceptifs, et recherche empirique de caractéristiques pertinentes pour modéliser le descripteur.

3.3.1 Percussivité d'un son polyphonique

Pour constituer notre base de sons polyphoniques étiquetés, nous avons réalisé une segmentation de divers extraits de musique populaire en sons de quelques dizièmes de secondes. Ensuite, nous avons évalué à la main si ils étaient percussifs ou non, en leur attribuant l'étiquette « 0 / non-percussif »ou « 1 / percussif »(discrimination booléenne). Enfin nous avons sélectionné une centaine de sons parmi tous les segments obtenus : les sons les plus homogènes, pour lesquels les tests montraient un consensus sur leur (non-)percussivité.

Puis nous avons étudié de la même manière que précédemment la corrélation de la percussivité des signaux sonores avec des caractéristiques de bas-niveau du type Mpeg7 (voir les graphiques de la figure 3.4), avant de rechercher manuellement une fonction *simple* traduisant la brutalité de l'attaque du son. Nous sommes arrivés à la formule $Attac = Max(\frac{d(Window(x^2))}{dt})$, qui donne des résultats assez corrélés avec la percussivité. Enfin, nous avons optimisé cette formule, en essayant différents types de fenêtres d'observation (Blackman, Hanning, Chebyshev, etc...), avec différentes tailles de fenêtre, avant de retenir la plus corrélée : la fenêtre de Chebyshev à 1000 échantillons, dont la formule complète :
$Attac = Max(\frac{d(Chebyshev(x^2,1000echantillons))}{dt})$, donne 85% de bonnes classifications sur l'ensemble de nos sons.

Sur la figure 3.4, qui montre les fonctions des différentes caractéristiques en fonction de la percussivité des sons, on constate visuellement que la fonction Attac discrimine assez bien les sons non-percussifs (en bas du graphe) des sons percussifs (en haut), et encore mieux avec une fenêtre de 1000 échantillons qu'avec une fenêtre de 5000, tandis que l'énergie la fréquence fondamentale et l'énergie du signal ne permettent pas la moindre discrimination.

Cette formule permet de réaliser simplement le « profil percussif »d'une chanson, comme présenté sur la figure 3.5.

A nouveau, la formule Attac, interprétable comme la brutalité de l'attaque énergique du signal sonore, est un paramètre obtenu par une succession d'opérations simples de traitement du signal, qui offre de bonnes performances pour

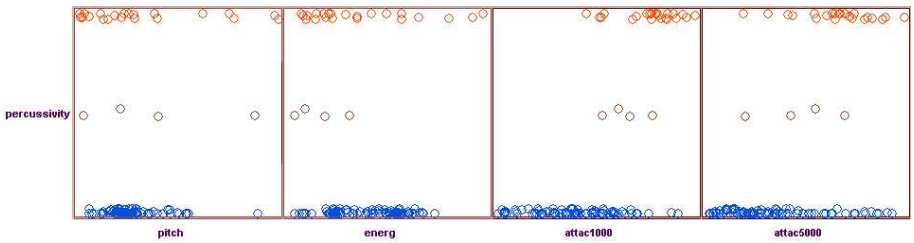


Fig. 3.4: Graphiques de correspondance entre la percussivité (« Percussivity ») et diverses caractéristiques du signal (Fréquence fondamentale « Pitch », énergie brute « energ », fonction d'attaque avec une fenêtre de Chebyshev de 1000 échantillons « attac1000 », fonction d'attaque avec une fenêtre de Chebyshev de 5000 échantillons « attac5000 »)

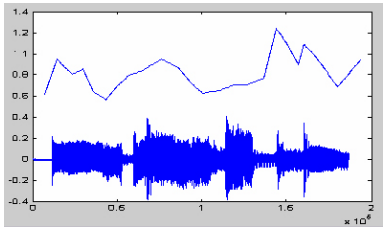


Fig. 3.5: Profil Percussif d'un extrait de « Darling »(Beatles)

résoudre un problème donné, alors que des fonctions générales ne le permettent pas.

3.3.2 Détection de voix chantée

Pour la détection de voix chantée, la construction de la base de sons est réalisée par segmentation de chansons polyphoniques, et étiquetage manuel booléen des échantillons sonores (« contient / ne contient pas de voix chantée »). A nouveau, nos caractéristiques générales ne donnant pas de résultat probant, nous avons construit empiriquement une fonction simple évaluant l'harmonicité du signal dans les fréquences de la voix :

$VOCALITY = Platitude(Autocorrelation(LogSpectre(Hamming(Filtrage(x,<5000Hz)))))$

Comme pour la percussivité, cette formule permet de réaliser assez simplement le « profil vocal »d'une chanson, présenté sur la figure 3.6.

3.4 Conclusion

La construction empirique de descripteurs de signaux musicaux, est une tâche extrêmement coûteuse en temps et en expertise, notamment dans la phase de construction de fonctions pertinentes. Les diverses expériences réalisées nous ont cependant montré que l'extraction de descripteurs variés utilise généralement un ensemble de techniques de base proches, et on a pu ainsi déterminer divers points communs primordiaux à toute construction de descripteurs.

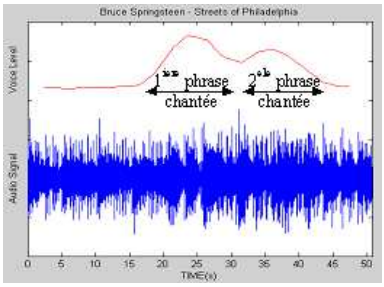


Fig. 3.6: Profil Vocal d'un extrait de « Streets of Philadelphia »(Bruce Springsteen)

Tout d'abord, il est nécessaire de construire des fonctions pertinentes pour résoudre le problème posé. Ces fonctions peuvent être soit une fonction d'extraction traduisant une méthode complète, souvent complexe, qui fournit directement une évaluation du descripteur désiré, soit un ensemble de fonctions caractéristiques plus simples, évaluant chacune des propriétés précises du signal musical, combinées dans des modèles plus complexes.

Ensuite, la plupart de ces fonctions caractéristiques et d'extraction peuvent être décomposées en une succession d'opérations simples appliquées sur le signal audio, comme des filtrages, des transformations temporelles/fréquentielles (spectre), des extractions de statistiques, l'intégration d'informations perceptives (passage à l'échelle Mel), etc...

Enfin, pour construire des fonctions pertinentes, le choix de traitements utilisés et l'ajustement de leurs paramètres nécessitent un long travail de tâtonnement, que seule l'expertise du chercheur permet de réduire.

Ces points communs sont généraux, applicables et suffisants à toute synthèse de descripteurs, et permettent donc théoriquement de constituer un système qui réalise automatiquement l'extraction complète de descripteurs. Le prochain chapitre présente l'étude de l'intégration et de l'automatisation de ces principes de base, qui mène à la réalisation du système « EDS ».

4. LE SYSTÈME EDS

Lors de la découverte de l'état de l'art et lors des premières expériences réalisées, les méthodes employées pour extraire diverses propriétés d'un signal musical nous ont toutes semblé assez proches. Cette proximité se traduit sous forme d'un ensemble de principes de base simples et généraux :

- on cherche à résoudre un problème de description à partir d'un signal numérique, c'est-à-dire on cherche à attribuer une valeur ou une classe à un signal à partir d'un modèle de descripteur ;
- le problème descriptif peut être résolu ou approximé grâce à des caractéristiques pertinentes, qui peuvent être utilisées seules ou combinées entre elles ;
- ces caractéristiques, qu'elles soient simples ou complexes, peuvent généralement être décomposées fonctionnellement en une succession d'opérations de traitements simples appliqués sur le signal numérique ;
- les fonctions caractéristiques peuvent être optimisées par le choix des méthodes utilisées et l'ajustement des valeurs des paramètres de ces méthodes ;
- la synthèse du modèle final de l'extracteur peut également être optimisée, par le choix des méthodes de modélisation utilisées, et l'ajustement de paramètres spécifiques à ces méthodes ;
- l'expertise du chercheur est fondamentale pour optimiser le choix des méthodes et les ajustements de valeurs lors de la construction de fonctions caractéristiques et la synthèse de modèles descriptifs.

L'application de ces principes montre que l'extraction de descripteurs est une tâche longue et fastidieuse, d'autant plus qu'elle est généralement réalisée à la main par les chercheurs. Cependant, l'observation de bases communes à toutes les extractions nous a fait entrevoir la possibilité d'automatiser cette tâche, en la décomposant en trois étapes :

1. Elaboration d'un problème descriptif par construction automatique (segmentation) et étiquetage manuel (tests auprès d'auditeurs) d'une base de signaux
2. Construction automatique de fonctions extrayant des propriétés pertinentes pour résoudre ce problème
3. Optimisation automatique de modèles de classification utilisant ces fonctions

Ce sont les trois étapes de base à partir desquelles nous avons implémenté le système EDS (« Extractor Discovery System »). Nous présentons dans ce chapitre l'architecture globale d'EDS, les choix techniques fondamentaux permettant de réaliser l'automatisation des étapes de l'extraction. Nous nous attacherons plus particulièrement à décrire l'automatisation de l'étape de construction

automatique de fonctions, qui est le point fondamental du système, et notamment l'introduction de connaissances dans le système afin de s'assurer de leur pertinence.

Enfin, nous présenterons le module d'exportation d'EDS, permettant d'utiliser les descripteurs extraits dans des applications extérieures.

NB : Bien que nous nous soyons initialement intéressés à la construction de descripteurs à partir de signaux musicaux, nous présentons dans ce chapitre le système EDS dans toute sa généralité, c'est-à-dire utilisable pour résoudre tout type de problème descriptif à partir de signaux numériques.

4.1 Présentation des principes généraux du système EDS

Nous établissons ici les principes généraux utilisés dans EDS, illustrés par un exemple simple de description d'un signal sonore.

4.1.1 Justification de l'extraction automatique de descripteurs

L'ensemble de l'état de l'art montre que pour résoudre des problèmes difficiles, les caractéristiques générales du type Mpeg7 sont souvent insuffisantes utilisées telles quelles, mais qu'elles restent cependant utiles comme base pour construire des caractéristiques plus spécifiques. Nous présentons ici sous la forme d'un exemple simple, deux hypothèses principales permettant de justifier l'automatisation du processus : (1) pour résoudre un problème descriptif, il est nécessaire de déterminer des caractéristiques spécifiquement adaptées à ce problème, et (2) ces caractéristiques spécifiques peuvent être obtenues par des combinaisons d'opérations simples de traitement du signal.

Un problème simple : Extraction d'un sinus dans un bruit rose

Considérons le problème descriptif simple de détecter, à partir d'un signal acoustique, la fréquence d'un sinus pur dans une bande de fréquences donnée (0-1000Hz), noyé dans un bruit rose de forte puissance dans une autre bande de fréquences (1000-2000Hz).

Limites des caractéristiques traditionnelles

Du fait de sa forte puissance, le bruit rose est la composante principale du son étudié. Ainsi, les fonctions caractéristiques générales se focalisent sur ce bruit rose, et sont incapables de détecter le sinus. Comme on peut l'observer sur la partie gauche de la figure 4.1, représentant le spectre d'un sinus de 650Hz mélangé à un bruit rose entre 1000 et 2000Hz, le pic spectral associé au sinus est bien visible. Cependant, comme il n'est pas prédominant, il reste très difficile à extraire automatiquement.

Ceci illustre notre premier principe de base : Il est nécessaire de construire des fonctions caractéristiques spécifiques à chaque problème.

Ajout d'opérations spécifiques (post et pré-traitements)

Les descripteurs généraux ne couvrant pas un espace de recherche suffisamment vaste pour trouver des extracteurs spécialisés, il est nécessaire de les rendre adaptés aux problèmes descriptifs considérés en permettant l'exploration d'un espace fonctionnel plus vaste et complexe, notamment par ajout de traitements additionnels paramétrés.

Dans notre exemple, le problème est évidemment très facile à résoudre en appliquant un pré-filtrage adéquat, qui permet d'éliminer une grande partie des composantes du bruit rose. La fréquence du sinus étant comprise entre 0 et 1000Hz et celle du bruit rose entre 1000 et 2000Hz, la solution est bien sûr un filtre passe-bas de fréquence de coupure 1000Hz.

Ceci illustre notre second principe : une fonction spécifique à un problème peut être construite par composition d'opérations simples de traitement du signal.

Choix des opérations additionnelles

Le choix de l'opération à ajouter (le filtrage passe-bas) est guidé par un double ensemble de connaissances : celle du problème à résoudre (les fréquences des sinus sont limitées à une bande donnée), et celle des effets des opérations (le filtrage permet de faire émerger les caractéristiques dans une bande de fréquences), c'est-à-dire le savoir-faire des experts en traitement du signal.

Ceci illustre notre troisième principe : la construction d'une fonction caractéristique pertinente nécessite un apport de connaissances heuristiques sur les opérations qui la composent, et le problème qu'elle doit résoudre.

Paramétrage des opérations additionnelles

Ce filtrage est une opération définie par trois paramètres : filtrage (type d'opération), passe-bas (premier paramètre de l'opération), fréquence de coupure (deuxième paramètre de l'opération). Évidemment, le niveau de paramétrage de cette opération peut varier suivant la précision désirée, par exemple en ajoutant un choix du type de fenêtre à utiliser lors du filtrage, etc...

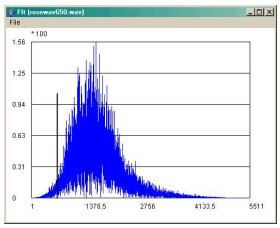
Seul un paramétrage précis permet de résoudre efficacement le problème. Par exemple, si on applique effectivement au signal une opération de pré-filtrage passe-bas, mais de fréquence de coupure 2000Hz, on observe sur le spectre de ce signal pré-traité (partie centrale de la figure 4.1) que le pic associé au sinus est toujours visible, mais que son amplitude, bien que plus importante que sans filtrage, reste plus faible que celle du bruit rose ; il ne peut donc pas être détecté.

L'ajustement de ce paramètre peut être réalisé de deux façons : soit à partir d'une connaissance du problème à résoudre (la bande de fréquence des sinus), soit par tâtonnements et observation de effets des paramètres sur la lisibilité du problème.

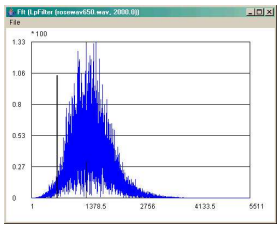
Dans notre exemple, en l'absence de connaissance a priori du problème, on remarque que le fait de baisser la fréquence de coupure du filtre augmente l'amplitude du pic spectral du sinus : cette opération est donc plus *pertinente*. On va alors essayer d'optimiser l'extraction en cherchant un paramétrage encore plus pertinent. Cette optimisation va nous mener, plus ou moins rapidement suivant la méthode utilisée, à la fréquence de coupure optimale de 1000Hz. Comme on peut l'observer sur la partie droite de la figure 4.1, le pic associé au sinus dans

le spectre du signal pré-filtré par un passe-bas à 1000Hz, est non seulement bien visible, mais surtout prédominant par rapport à l'ensemble des composantes du bruit rose, et devient par conséquent très facile à extraire automatiquement.

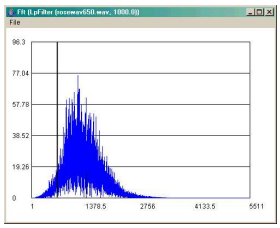
Ceci illustre notre quatrième principe : la pertinence d'une fonction mesure sa capacité à représenter une propriété permettant de résoudre le problème posé ; et notre cinquième principe : pour résoudre un problème descriptif, il est nécessaire de paramétrer les opérations des fonctions caractéristiques afin d'obtenir une pertinence optimale.



Spectre du signal brut



Spectre du signal pré-filtré par un passe-bas de fréquence de coupure Fc=2000Hz



Spectre du signal pré-filtré par un passe-bas de fréquence de coupure Fc=1000Hz

Fig. 4.1: Visualisation de spectres du signal acoustique d'un sinus pur (650Hz) noyé dans un bruit rose (1000-2000Hz) de forte amplitude

Modèle final

Enfin, nous cherchons à extraire la fréquence du sinus. Cependant, la position du pic du spectre du signal filtré, n'est pas une fréquence, mais seulement un index lié à cette fréquence. Il est donc nécessaire de construire une échelle de correspondance entre cet index et la fréquence désirée : dans notre exemple, c'est une relation linéaire simple dépendant de la fréquence d'échantillonnage du signal, et de la taille de la fenêtre d'observation du spectre. La fréquence à extraire est donc un modèle linéaire de la fonction caractéristique trouvée précédemment.

Ceci illustre notre sixième et dernier principe : la synthèse d'un extracteur nécessite l'utilisation d'un modèle combinant des fonctions caractéristiques pertinentes.

4.1.2 Intégration des principes dans le système EDS

Suivant ces principes, l'idée du système EDS est donc, pour un problème descriptif donné, de construire automatiquement un ensemble de fonctions caractéristiques pertinentes, à partir de connaissances générales internes au système et de connaissances spécifiques apportées par l'utilisateur, puis de combiner ces fonctions pour synthétiser un modèle optimal du descripteur recherché.

Ces principes ont posé les bases de la construction d'EDS :

– « Il est nécessaire de construire des fonctions caractéristiques spécifiques à chaque problème » : c'est l'hypothèse de base qui justifie l'intérêt du système ; dans EDS, un problème est défini par un ensemble de signaux

étiquetés suivant les valeurs d'un descripteur, et sa résolution consiste à modéliser cet étiquetage ;
- « une fonction spécifique à un problème peut être construite par composition d'opérations simples de traitement du signal » : les fonctions construites par le système sont représentables sous forme d'arbres d'opérations simples ;
- « la construction d'une fonction caractéristique pertinente nécessite un apport de connaissances heuristiques sur les opérations qui la composent, et le problème qu'elle doit résoudre » : les fonctions sont construites en respectant les types de données traitées, les types d'opérations à réaliser, et des heuristiques de construction traduisant une expertise en traitement du signal ;
- « la pertinence d'une fonction mesure sa capacité à représenter une propriété permettant de résoudre le problème posé » : la pertinence d'une fonction est sa capacité à représenter l'étiquetage de la base de signaux ;
- « pour résoudre un problème descriptif, il est nécessaire de paramétrer les opérations des fonctions caractéristiques afin d'obtenir une pertinence optimale » : la recherche de fonctions caractéristiques optimales est réalisée automatiquement par une méthode génétique ;
- « la synthèse d'un extracteur nécessite l'utilisation d'un modèle combinant des fonctions caractéristiques pertinentes » : finalement, le système cherche automatiquement un modèle de classification optimal intégrant les fonctions construites.

Ceci constitue la partie « recherche » d'EDS : c'est la phase de modélisation du descripteur.

Après cette recherche d'un modèle optimal, une partie « export » permet à EDS d'exporter ce modèle en vue d'une utilisation dans d'autres applications : c'est la phase de synthèse de l'extracteur. Cette phase consiste tout d'abord à évaluer la performance effective du modèle sur une base de test indépendante de la base d'apprentissage, puis à générer et enregistrer un programme exécutable permettant de calculer ce modèle sur tout nouveau signal, indépendamment d'EDS.

La figure 4.2 présente ainsi un aperçu global de l'architecture d'EDS, constituée de deux grands blocs successifs : la modélisation du descripteur, puis la synthèse d'un extracteur.

La suite de ce chapitre présente en détails toutes les grandes parties du système : la définition d'un problème descriptif, l'automatisation des principes de la recherche de fonctions caractéristiques pertinentes présentés ci-dessus, la modélisation du descripteur, et la synthèse de l'extracteur final.

4.2 Définition de problèmes descriptifs pour EDS

Le mode de définition du problème perceptif à résoudre est un choix fondamental qui détermine l'ensemble de l'architecture d'EDS. Ainsi, nous utilisons une représentation générale permettant de traiter l'ensemble des problèmes descriptifs.

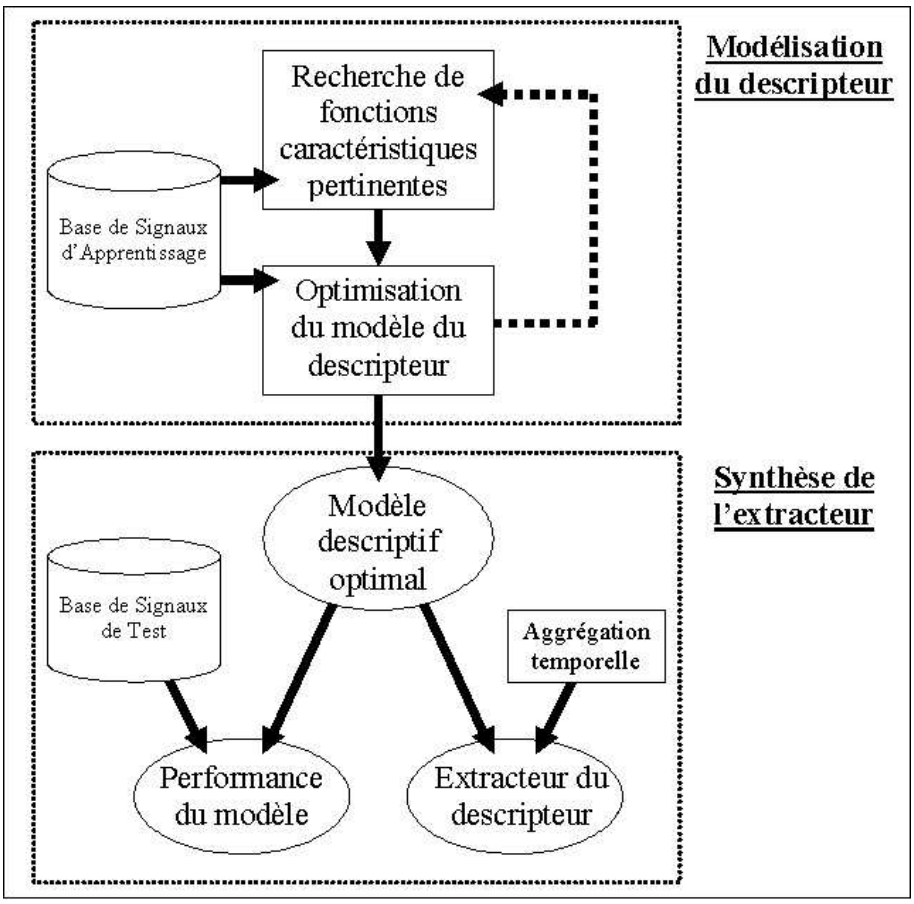


Fig. 4.2: Architecture globale du système EDS

4.2.1 Types de problèmes descriptifs à résoudre

Deux types de problèmes descriptifs sont présentés dans l'état de l'art (chapitre 2) : les évaluations et les identifications.

Les évaluations attribuent à un signal une valeur sur une échelle descriptive. Par exemple l'évaluation de l'énergie musicale d'un extrait est mesurée entre 0% et 100%, et l'extraction de la fréquence fondamentale d'un son polyphonique est indiquée en Hz. L'évaluation d'un descripteur se fait à partir d'un modèle de régression.

Les identifications sont des classifications supervisées, c'est-à-dire qui affectent à un signal une classe donnée parmi un ensemble de classes connu à priori. Or un problème de classification N classes peut généralement être décomposé en N problèmes de classification en deux classes du type « ce signal appartient-il à une classe donnée ou pas ? ». Ainsi, nous nous focaliserons principalement sur les problèmes de discrimination entre deux classes, dites classifications binaires, tout en gardant la possibilité d'étendre le système à plus de deux classes.

4.2.2 Mode de soumission du problème descriptif

Bien que légèrement différents, les deux types de problèmes considérés peuvent être présentés de façon similaire : on cherche à étiqueter des signaux numériques, en choisissant l'étiquette parmi un ensemble discret (classes) ou continu (valeurs).

A partir de ce constat, et afin de contraindre le moins possible l'utilisation du système, nous avons décidé de réduire au strict minimum les entrées nécessaires à son fonctionnement : ainsi, le système doit être capable d'extraire des descripteurs à partir d'une seule base d'exemples du problème à résoudre, c'est-à-dire une base de signaux étiquetés, sur laquelle la pertinence des fonctions caractéristiques créées est vérifiée. Ainsi, le système fonctionne techniquement de manière identique pour les problèmes de classification et de régression ; seules les techniques utilisées sont adaptées au type de problème, comme par exemple le critère d'évaluation de la pertinence d'une fonction caractéristique, ou les méthodes de modélisation du descripteur à partir d'un ensemble de fonctions.

De plus, nous conservons la possibilité d'apporter des données supplémentaires en plus de la base d'exemples, afin d'optimiser le fonctionnement du système. Par exemple, l'apport de règles ou d'heuristiques externes spécifiques à certains problèmes descriptifs permettront d'améliorer la qualité du modèle créé par le système.

4.3 Représentation des fonctions caractéristiques

Le principe d'EDS est de construire automatiquement des fonctions de traitement du signal. Ces fonctions peuvent être des caractéristiques directes comme par exemple « les dix premiers coefficients cepstraux »(utilisés pour les études de timbre de la section 2.2.1), ou le résultat de traitements simples comme « la position du pic maximal de corrélation du signal avec un signal donné »(utilisé dans notre extracteur de « drum tracks »présenté dans la section 3.1), ou plus complexes comme la fonction d'extraction de tempo de [Scheirer, 1998] présentée sur la figure 2.1.

4.3.1 Décomposition en opérations de base

Les fonctions caractéristiques doivent pouvoir être synthétisées de manière simple et modulable par le système. Comme cela a été présenté en introduction, toute fonction de traitement numérique est décomposable en blocs de fonctions de plus bas niveau. Par exemple, la fonction d'énergie d'un signal peut être décomposée en : $Energie = \sum x^2 = Somme(Carré(x))$, c'est-à-dire sous la forme d'une composition de deux fonctions simples « Somme »et « Carré ».

Suivant ce principe, nous avons choisi de représenter les fonctions générées par le système sous la forme d'une combinaison d'opérations basiques réalisant des calculs mathématiques ou des traitements de signaux, c'est-à-dire transformant ensembles de valeurs numériques (valeur unaire, vecteur, séquence, matrice, représentations multi-dimensionnelles, etc...), en d'autres ensembles de valeurs.

Ces opérations peuvent agir à différents niveaux de complexité et de proximité du signal traité. Par exemple, une opération simple « Moyenne »réalise directement la moyenne des valeurs d'un ensemble, tandis qu'une opération complexe « Fréquence fondamentale »réalise un grand nombre de calculs pour fournir une valeur caractéristique.

De plus, les opérations peuvent éventuellement être paramétrées, comme on l'a vu dans l'exemple 4.1.1. Par exemple, une fonction de « Filtre passe bas »peut avoir comme paramètre basique la fréquence de coupure du filtre. Mais le niveau de paramétrage des fonctions peut également varier de manière optionnelle suivant la précision souhaitée pour l'opération ; ainsi, pour cette même opération « Filtre passe bas », on peut également envisager de contrôler d'autres paramètres comme le type de fenêtre utilisé, ou la pente de coupure. L'ajustement du niveau de paramétrisation des opérations utilisées doit être contrôlé, afin de maîtriser la taille de l'espace des fonctions constructibles, notamment pour assurer l'efficacité de la phase de recherche des fonctions pertinentes.

4.3.2 Représentation sous forme d'arbre

La combinaison des opérations dans les fonctions caractéristiques pouvant se faire de diverses façons (séquentielle, récursive, paramétrée, etc...), nous avons choisi de représenter ces dernières sous forme d'arbres opérationnels.

Par exemple, la fonction précédente « $Energie = Somme(Carré(x))$ », est représentée sous forme d'un arbre linéaire comprenant 2 opérations, tandis que les paramètres d'une fonction un peu plus complexe comme « $Fonction = Max(Corrélation(Filtre\ Passe-bas(Signal, 500Hz), Signal\ de\ référence))$ » constitueront de nouvelles branches (voir la figure 4.3).

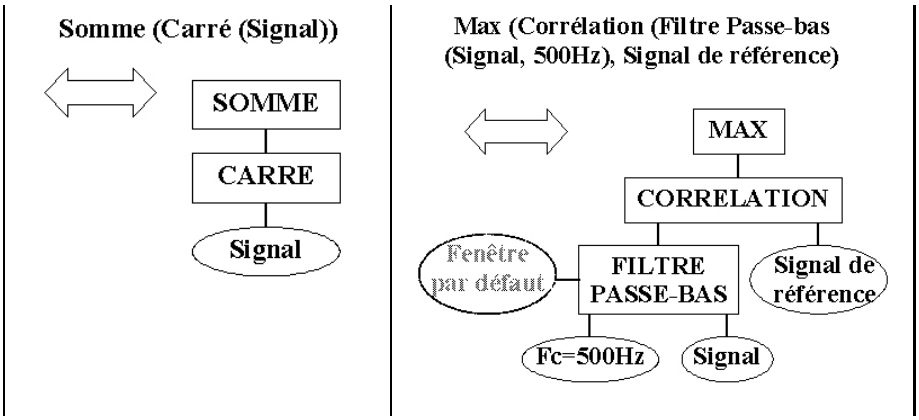


Fig. 4.3: Représentation de fonctions sous forme d'arbre équivalents

4.3.3 Types des éléments constitutifs d'une fonction

Ainsi, dans EDS, une fonction complète pouvant être représentée sous forme d'arbre, est définie comme une composition d'opérateurs de base, de paramètres, et d'arguments d'entrée.

Les *opérateurs* peuvent être :

– des opérateurs mathématiques : moyenne, variance, etc...(« Max »sur la figure 4.3) ;
– des opérateurs de traitement du signal dans le domaine temporel : Fenêtrage, Durées caractéristiques, etc...(« Corrélation ») ;
– des opérateurs de traitement du signal dans le domaine spectral : Spectre, Filtrage, etc...(« Filtre passe-bas »).

Les *paramètres* de ces opérateurs peuvent être :
– des valeurs numériques : fréquences caractéristiques, nombre d'éléments désirés, tailles de fenêtres, etc...(« Fc ») ;
– des labels : choix d'un type de fenêtre, d'une méthode de traitement, etc...
– le résultat d'un traitement plus complexe : par exemple la fréquence de coupure « Fc »pourrait être la fréquence fondamentale du signal étudié, calculée par l'opération « $Fc = Fréquence\ fondamentale(Signal)$ ».

Les *arguments* d'entrée peuvent être :
– le signal à étudier (« Signal ») : c'est l'argument d'entrée principal ;
– des constantes numériques (« Fc ») ;
– des signaux externes constants (« Signal de référence »).

4.3.4 Construction automatique de fonctions syntaxiquement correctes

La construction de fonctions à partir des éléments de base *arguments* d'entrée, *opérations* possibles, et *paramètres* des opérations, est exprimable sous la forme :

« une fonction consiste à appliquer une succession d'*opérations* sur un *argument* d'entrée principal, le signal étudié, chacune de ces *opérations* pouvant être contrôlée par des *paramètres*, qui peuvent être soit des *arguments* d'entrée secondaires, soit le résultat d'autres *opérations* appliquées sur les *arguments* d'entrée principal ou secondaires »

A partir de cette définition, nous avons élaboré une méthode simple de génération automatique de nouvelles fonctions non-contrainte, en construisant l'arbre fonctionnel (voir section 4.3) en partant de l'élément terminal principal : le signal étudié. Ainsi, la méthode est la suivante :

1. on part du signal étudié (argument d'entrée principal), appelé *fils*
2. on choisit une opération à appliquer sur le *fils*
3. si l'opération nécessite d'autres arguments que le *fils*, le système essaye de les créer automatiquement en utilisant cette même méthode, ou choisit des arguments par défaut
4. le résultat de l'opération est un nouveau *fils*
   – si le système ne trouve plus d'opération à appliquer au *fils*, ou si on dépasse un nombre limite d'itérations, la construction s'arrête, et le système renvoie la fonction obtenue
   – sinon, on recommence la méthode à partir du (2)

Cette méthode de synthèse permet de construire l'ensemble des fonctions de l'espace pour un ensemble d'opérations possibles donné. Cependant, parmi ces fonctions, beaucoup seront inutilisables car syntaxiquement incorrectes, comme par exemple la fonction « $Filtre\ passe - haut(Moyenne(Signal), 1000Hz)$ »,

qui est physiquement fausse et incalculable, car elle essaye de filtrer une valeur unique.

De plus, pour les fonctions physiquement correctes, le type de résultat obtenu en sortie reste incontrôlable. En effet, par exemple, la fonction « $Max(Signal)$ » renvoie une valeur d'amplitude unique, tandis que « $Spectre(Signal)$ »renvoie un vecteur d'amplitude rangé suivant un vecteur de fréquences.

Afin d'assurer le validité physique et de contrôler le résultat en sortie des fonctions générées, il est nécessaire d'ajouter à cette méthode de synthèse un ensemble de règles de construction fondamentales : le typage des données et des opérations.

4.4 Typage des données

La définition d'un système de typage est fondamentale, dans la mesure où il va permettre de représenter physiquement l'évolution des données dans une fonction, une opération après l'autre, chaque donnée utilisée dans une fonction étant caractérisée par un type spécifique et unique. Le typage permet ainsi de s'assurer que les fonctions créées sont syntaxiquement correctes.

4.4.1 Typage physique des données atomiques

Plusieurs systèmes de typage existent, notamment le « strong typing »([Montana, 1995]), qui permettent de différencier les données suivant leur dimension numérique de programmation, comme par exemple les entiers, les flottants, les vecteurs et matrices associés, etc...Cependant, dans le contexte de traitement de données physiques, la différence entre ces types de programmation est superficielle.

En effet, par exemple, un signal acoustique et son spectre fréquentiel sont tous deux, d'un point de vue numérique, des vecteurs de valeurs flottantes. Cependant, d'un point de vue physique, ce sont des données fondamentalement différentes : le signal acoustique est une représentation d'amplitude en fonction du temps, tandis que son spectre est une représentation d'amplitude en fonction de la fréquence. Et par conséquent, les traitements applicables à ces données sont également complètement différents. Cette différence physique doit donc absolument être transcrite dans le système pour s'assurer que les opérations appliquées sur les données ont du sens physiquement.

De plus, comme on l'a présenté en introduction de ce chapitre, une des fonctionnalités les plus importantes du système EDS est la possibilité d'y introduire des connaissances a priori sur la pertinence physique des fonctions synthétisées. La traduction de cette expertise pour le système, sous forme de règles physiques de construction des fonctions, nécessite également l'utilisation des types physiques des données traitées.

Ainsi, nous utilisons dans EDS un système de typage des données basé sur trois dimensions physiques de base : l'amplitude (sans dimension), le temps, et la fréquence, qui permettent de représenter l'ensemble de types de données traitées en traitement du signal acoustique. On définit pour ces trois dimensions, trois types atomiques de données notés « a », « t », « f », qui vont caractériser des valeurs numériques uniques de dimension physique donnée. Par exemple, la

hauteur d'un pic spectral est de type « a », la position d'un pic temporel est de type « t », et la fréquence fondamentale d'un signal est de type « f ».

4.4.2 Typage physique de l'ensemble des données

Bien entendu, les données traitées sont rarement constituées d'une seule valeur. Les données constituées d'un ensemble de valeurs numériques peuvent généralement être considérées comme *fonctionnelles*, dans la mesure où elles représentent l'évolution de valeurs d'une dimension physique donnée (temps, fréquence, ou amplitude), dans d'autres dimensions physiques (temps, fréquence). Nous avons choisi comme notation des représentations fonctionnelles le symbole « : », sous la forme « type de l'axe d'evolution : type des valeurs ». Par exemple, un signal acoustique est une fonction de données de type Amplitude « a », qui évoluent dans le temps de type « t ». Son type est donc noté « $t : a$ ».

De plus, du fait de l'utilisation de signaux numériques échantillonnés, nous avons dû introduire la notion de fréquence d'échantillonnage suivant chaque axe d'évolution. Ainsi, pour un signal acoustique échantillonné à 44100Hz, l'axe d'évolution échantillonné à 44100Hz a un pas temporel $\delta t = 1/44100s$, et son type est plus précisément « $t(\delta t) : a$ ».

Dans le cas d'un ensemble de valeurs d'un type précis qui ne suivent pas un axe d'évolution, les données ne sont plus considérées comme fonctionnelles, mais deviennent vectorielles. Nous avons alors introduit la notation « V »(de « V »ecteur) pour définir ces ensembles de valeurs. Par exemple, l'ensemble des hauteurs des 5 plus grands pics d'amplitude du spectre d'un signal contient 5 valeurs d'amplitude qui ne représentent aucune évolution. Son type est donc noté « Va ». De même, le vecteur contenant les positions temporelles des onsets percussifs d'un signal audio est noté « Vt ».

Enfin, certains descripteurs plus complexes peuvent traiter des données qui évoluent dans plusieurs dimensions simultanément. La notation fonctionnelle « : » est alors conservée, et les dimensions supplémentaires sont notées séquentiellement, sous la forme « type de l'axe d'évolution 1,type de l'axe d'évolution 2,... : type des valeurs ». Par exemple, l'observation sur 10 fenêtres temporelles successives d'un signal acoustique échantillonné à la fréquence f_e (de type « $t(\delta t_e) : a$ », avec $\delta t_e = 1/f_e$), produit une séquence de 10 signaux acoustiques échantillonnés à f_e, observés régulièrement dans le Temps tous les δt_f, dépendant de la taille de la fenêtre. Son type est donc noté « $t(\delta t_f)t(\delta t_e) : a$ », les deux axes d'évolution étant notés séquentiellement.

A nouveau, on introduira la notation « V » pour un ensemble de données sans axe d'évolution. Par exemple, si on observe un signal sur 10 fenêtres d'observation choisies de manière aléatoire, on obtient un ensemble de 10 signaux qui ne suivent aucune évolution. On notera cette donnée « $Vt : a$ », ou « $Vt(\delta t_e) : a$ ».

Dans le cas de dimensions d'évolution multiples, la représentation des données sous forme vectorielle ne suffit plus à distinguer ces dimensions. On introduit alors une représentation matricielle (pour une évolution dans 2 dimensions), voire, à partir de 3 dimensions simultanées (utilisées pour le calcul de certains descripteurs dynamiques), une représentation sous forme d'arbre multidimensionnel dont les niveaux de profondeur correspondent aux dimensions d'évolution. La permutation des dimensions d'évolution dans la notation du type ne change alors pas la nature des données, mais correspond simplement à une réorganisation de celles-ci. Par exemple, pour le signal précédent découpé en

fenêtres d'observation (notation matricielle à deux dimensions évolutives), les types de données « $t(\delta t_1)t(\delta t_2) : a$ » et « $t(\delta t_2)t(\delta t_1) : a$ » sont équivalents, et correspondent simplement à une transposition de la matrice de représentation des données : dans un cas, on observe l'évolution d'un extrait de signal dans une fenêtre d'observation donnée (δt_e), et dans l'autre on observe l'évolution d'une composante du signal sur les fenêtres d'observation successives (δt_f).

Le tableau 4.1 résume les différents types utilisés dans le système.

4.5 Typage des opérations

A partir de ce système de typage des données traitées dans EDS, on peut caractériser les traitements appliqués à ces données en terme de changements de valeurs ET de types.

Ainsi, dans une fonction complète composée d'une succession d'opérateurs, les données vont être transformées lors de chaque opération pour arriver au résultat final. En d'autres termes, toute opération appliquée sur une donnée d'un certain type consiste à la transformer en une nouvelle donnée d'un type précis.

4.5.1 Transformation du type de donnée lors d'une opération

Par exemple, une opération de *Filtrage* appliquée sur un signal acoustique, vecteur d'Amplitudes fonction du Temps de type « $t : a$ », va produire un signal acoustique différent, mais de même type « $t : a$ ». Par contre, une opération d'*Extraction de pic* appliquée sur un signal acoustique (de type « $t : a$ ») va produire une valeur unique d'amplitude (de type « a »).

Dans le cas des données fonctionnelles (voir le tableau 4.1), le changement de type peut également s'appliquer sur le pas associé à une dimension d'évolution. Par exemple, une opération de ré-échantillonnage à 11025Hz, appliquée sur un signal de fréquence d'échantillonnage 44100Hz, donc de type « $t(1/44100) : a$ », fournit un signal proche, mais de fréquence d'échantillonnage 11025Hz, donc de type « $t(1/11025) : a$ ».

Enfin, le type du résultat de l'opération peut dépendre du type de donnée en entrée. Par exemple, une opération de Moyennage appliquée sur un ensemble de durées de type « Vt », produira une durée moyenne unique de type « t », tandis que la même opération appliquée sur un signal de type « $t : a$ » produira une amplitude moyenne unique de type « a ».

Afin de pouvoir construire automatiquement des fonctions physiquement correctes, il est nécessaire que toutes les opérations utilisées dans le système soient typées en terme de types de données d'entrée et de sortie. Or ces exemples montrent qu'on ne peut pas trouver de règle de typage commune à toutes les opérations, et qu'il faut donc trouver des règles spécifiques suivant les données traitées. Ainsi, on associera à chaque opérateur une *table de typage*, qui précise de manière dynamique le type du résultat en sortie en fonction des types des données d'entrée.

CATEGORIE	TYPE DE DONNEES	NOTATION	EXEMPLE
Atomique	Temps	t	durée d'un signal
	Fréquence	f	fréquence de coupure d'un filtre
	Amplitude	a	hauteur d'un pic spectral
Vectoriel	Ensemble d'Amplitudes	$V : a$ noté Va	signal acoustique
	Ensemble de Temps	Vt	positions de pics temporels
	Ensemble de Fréquences	Vf	positions de pics fréquentiels
Fonctionnel	Amplitude fonction de Temps	$t : a$ ou $t(\delta t) : a$	signal acoustique
	Amplitude fonction de Fréquences	$f : a$ ou $f(\delta f) : a$	spectre d'un signal
	Données de type T_{data} évoluant suivant une dimension de type T_{dim}	$T_{dim} : T_{data}$	
Fonctionnel Multi-Dimensionnel	Amplitude fonction de 2 axes Temporels	$tt : a$ ou $t(\delta t_1)t(\delta t_2) : a$	Observations successives d'un signal sur une fenêtre glissante
	Données de type T_{data} évoluant suivant deux dimensions de type T_{dim1} et T_{dim2}	$T_{dim1}T_{dim2} : T_{data}$	
	Données de type T_{data} évoluant dans N dimensions de type T_{dim1}, ..., T_{dimN}	$T_{dim1}...T_{dimN} : T_{data}$	
Vectoriel et Fonctionnel Multi-Dimensionnel	Ensemble d'Amplitudes fonctions d'un même axe Temporel de pas δt	$Vt : a$ ou $Vt(\delta t) : a$	Observations d'un signal sur des fenêtres aléatoires
	Ensembles de données de type T_{data} évoluant dans N dimensions de types T_{d1}, ..., T_{dN}	$V...VT_{d1}...T_{dN} : T_{data}$	

Tab. 4.1: Types de données utilisés dans EDS

4.5.2 Opérateurs opérationnels et opérateurs fonctionnels

Afin de générer ces *tables de typage* de manière simple et optimisée, il est important de noter que tous les opérateurs n'agissent pas de la même façon sur les données. En effet, du fait de l'existence dans notre système de typage de données fonctionnelles, c'est-à-dire de données caractérisées par plusieurs dimensions physiques (dimension de donnée et dimension(s) d'évolution(s)), on peut distinguer deux familles d'opérateurs :

- les opérateurs s'appliquant sur les valeurs, comme par exemple des opérations mathématiques (Mise au carré, log, ...), appelés *opérateurs opérationnels*, qui sont en général applicables sur tous les types de données ;
- les opérateurs s'appliquant sur l'évolution des valeurs dans une dimension donnée (filtrage fréquentiel, extraction de pics temporels, ...), appelés *opérateurs fonctionnels*, qui ne peuvent être appliqués que sur des données fonctionnelles compatibles, et qui peuvent changer les types de données.

Tous les opérateurs fonctionnels utilisés actuellement dans le système sont mono-dimensionnels, c'est-à-dire qu'ils s'appliquent toujours dans UNE dimension physique donnée. Par conséquent :

- ils ne s'appliquent pas sur des données vectorielles (de types « a », « Vt », « VVf », ...) ;
- dans le cas de données fonctionnelles mono-dimensionnelles (de types « $t : a$ », « $t : f$ », « $Vt : a$ », ...), ils s'appliquent sur la dimension d'évolution, si cette dernière est compatible avec l'opération (filtrage d'un signal acoustique de type « $t : a$ » suivant la direction « t ») ;
- dans le cas de données fonctionnelles multi-dimensionnelles (« $tt : a$ », « $ft : f$ », « $Vtt : a$ », ...), il est nécessaire de choisir la dimension d'évolution sur laquelle l'opération s'applique : par exemple « t » ou « f » pour une donnée de type « $ft : a$ », ou « $t(\delta t_1)$ » ou « $t(\delta t_2)$ » pour une donnée de type « $t(\delta t_1)t(\delta t_2) : a$ ».

Dans le cas de ces opérateurs multi-dimensionnels, dans le système EDS, l'opération s'applique par défaut sur la première dimension suivant laquelle est stockée la donnée. D'autre part, le choix de la dimension d'application peut être évident du fait de la compatibilité de l'opération réalisée ; par exemple, le filtrage d'un signal acoustique ne pouvant se faire que suivant un axe Temporel « t », l'application d'une opération de Filtrage sur une donnée de type « $ft : a$ » se fera forcément suivant l'axe « t ». Il est également envisageable d'ajouter au système des opérateurs multi-dimensionnels, c'est-à-dire qui s'appliquent simultanément dans PLUSIEURS dimensions physiques. Par exemple, on peut imaginer une fonction extrayant une mesure caractéristique à partir d'un spectrogramme (de type « $tf : a$ »), et fournissant en sortie une valeur numérique non-dimensionnelle de type « a ».

Enfin, certaines opérations fonctionnelles peuvent faire varier le nombre de dimensions des données : réduction (par exemple dans le cas d'un moyennage) ou expansion (par exemple les observations multiples d'un même signal).

4.5.3 Table de typage d'un opérateur

L'ensemble des remarques précédentes permet de générer de manière simple une table de typage pour chaque opérateur. La table doit présenter de la manière

la plus générale possible les règles de typage associant à tous les types de données en entrée, un type de sortie précis. On réalise ainsi une première intégration dans le système de connaissances en traitement du signal.

Une table de typage peut être très simple, comme par exemple pour l'opération « Valeur Absolue ». C'est une opération non-fonctionnelle, qui agit sur les dimensions des valeurs sans toucher aux dimensions d'évolution. De plus, sachant que la valeur absolue d'un Temps est également un Temps, de même pour les Fréquences et les Amplitudes, on en déduit que cette opération ne change jamais le type des données sur lesquelles elle est appliquée. Ainsi, la table de typage de « Valeur Absolue »se résume à deux conditions :

- fonctionnalité : non-fonctionnelle (le nombre de dimensions d'évolution peut être quelconque) ;
- transformation : le type de valeur des données est transformée en sa valeur absolue.

Ces conditions permettent de déduire l'ensemble des règles de typage associées à cet opérateur ($\forall(x, y, z) \in \{Types\ Atomiques\}$, avec $|t| = t$, $|f| = f$, $|a| = a$) :

- $x \rightarrow |x|$
- $Vx \rightarrow V|x|$
- $x : y \rightarrow x : |y|$
- $Vx : y \rightarrow Vx : |y|$
- $xz : y \rightarrow xz : |y|$
- etc ...

Afin d'obtenir une table de typage simple, on peut résumer toutes ces règles en deux conditions générales desquelles on peut déduire les autres :

- $Nombre\ de\ dimensions \geq 0$
- $\forall x \in \{Types\ Atomiques\}, x \rightarrow |x|$

En effet, on sait que l'ajout d'une dimension évolutive à la donnée d'entrée ne change pas son type de valeur, donc il n'y aura pas de modification de ces dimensions.

L'opération mathématique d'« Inversion »des valeurs est similaire, dans la mesure où elle transforme uniquement le type des valeurs, mais cette fois, ce type est transformé en son type inverse. Par exemple, un temps « t » devient l'inverse d'un temps, soit une fréquence « f ». La table de typage générale pour cette opération est la suivante :

- $Nombre\ de\ dimensions \geq 0$
- $\forall x \in \{Types\ Atomiques\}, x \rightarrow \frac{1}{x}$ (avec $\frac{1}{t} = f$, $1\frac{1}{f} = t$, $\frac{1}{a} = a$)

On peut en déduire de la même façon les règles pour tous les autres types d'entrée.

D'autres opérations sont fonctionnelles, c'est-à-dire qu'elles agissent sur une dimension d'évolution donnée. Par exemple, le « Moyennage »est une opération fonctionnelle adimensionnelle, car elle est valable pour tout type d'évolution, y compris les groupes non-structurés de type « V ». Elle suit la table de typage suivante :

- $Nombre\ de\ dimensions \geq 1$, V inclus
- $\forall x \in \{Types\ Atomiques, V\}, y \in \{Types\ Atomiques\}, x : y \rightarrow y$

On en déduit l'ensemble des règles de typage
($\forall x \in \{Types\ Atomiques, V\}, (y, z) \in \{Types\ Atomiques\}$) :

- $x \rightarrow \varnothing$ (opération impossible)
- $Vy \rightarrow y$
- $x : y \rightarrow y$

- $Vx : y \rightarrow Vy$ (opération suivant x)
- $xz : y \rightarrow z : y$ (opération suivant x)

Par ailleurs, l'opération de moyennage est *réductrice*, dans la mesure ou elle réduit la dimensionnalité évolutive des données. En effet, on élimine l'axe suivant lequel on réalise la moyenne.

L'opération de passage au « Spectre »est quant à elle fonctionnelle dimensionnelle, c'est-à-dire qu'elle agit sur une dimension d'évolution d'un type précis, en transformant le type d'une dimension d'évolution en son inverse (par exemple une évolution temporelle en évolution fréquentielle). On en déduit ainsi la table de typage générale suivante :

- $Nombre\ de\ dimensions \geq 1$
- $\forall (x, y) \in \{Types\ Atomiques\}, x : y \rightarrow \frac{1}{x} : y$ (avec $\frac{1}{t} = f$, $\frac{1}{f} = t$, $\frac{1}{a} = a$)

On retrouve bien la transformation d'un signal (« $t : a$ ») en spectre (« $f : a$ »).

La table de typage peut également restreindre les opérations à certains types de données. Par exemple, l'opération de « Filtrage », restreinte aux données fonctionnelles possédant une dimension d'évolution temporelle, est associée à la table de typage générale suivante :

- $Nombre\ de\ dimensions \geq 1$
- $\forall y \in \{Types\ Atomiques\}, t : y \rightarrow t : y$

L'axe d'évolution est ainsi limité au type « t ».

Il existe également des opérations fonctionnelles multidimensionnelles, qui agissent sur plusieurs dimensions évolutives simultanément. Par exemple une « Moyenne Bidimensionnelle », qui calcule la moyenne d'une moyenne, n'est applicable que sur des données possédant au moins deux dimensions évolutives. Elle réduit la dimensionnalité des données suivant la table de typage générale :

- $Nombre\ de\ dimensions \geq 2$, V inclus
- $\forall (x, y, z) \in \{Types\ Atomiques, V\}, xy : z \rightarrow z$

On en déduit notamment que l'application de l'opération sur x, Vx ou x :y, est impossible, leur dimensionnalité étant insuffisante.

Enfin, certaines opérations peuvent prendre plusieurs arguments en entrée, de dimensions précises. Par exemple un « Filtrage Passe-bas »nécessite une donnée d'entrée fonctionnelle temporelle de type « $t : x$ », et l'indication d'une fréquence de coupure de type « f », ce qui est retranscrit dans la table de typage générale suivante :

- $Nombre\ de\ dimensions \geq 1$
- $\forall x \in \{Types\ Atomiques\}, (t : x, f) \rightarrow t : x$

Le tableau 4.2 présente différentes tables de typage d'opérations utilisées dans le système.

Grâce à notre système de typage des opérateurs, le système est dorénavant capable de limiter les possibilités d'opérations applicables à certains types de données. Nous avons ainsi assuré l'homogénéité physique des fonctions construites automatiquement par ajout d'opérations successives sur le signal initial (méthode présentée au début de 4.3.4).

CATEGORIE	OPERATION	NOMBRE DE DIMENSIONS	REGLES DE TYPAGE
Non-Fonctionnelle	Valeur absolue	≥ 0	$\forall x \in \{Types\ Atomiques\}$, $x \rightarrow \|x\| \equiv x$ $avec \|t\| = t, \|f\| = f, \|a\| = a$
	Mise au carré	≥ 0	$\forall x \in \{Types\ Atomiques\}$, $x \rightarrow x^2 \equiv x$ avec $t^2 \equiv t, f^2 \equiv f, a^2 \equiv a$
	Inverse	≥ 0	$\forall x \in \{Types\ Atomiques\}$, $x \rightarrow \frac{1}{x}$ $avec\ \frac{1}{t} = f, \frac{1}{f} = t, \frac{1}{a} = a$
Fonctionnelle	Moyenne	≥ 1 ou « V »	$\forall x \in \{Types\ Atomiques, V\}$, $\forall y \in \{Types\ Atomiques\}$, $x : y \rightarrow y$
	Max	≥ 1 ou « V »	$\forall x \in \{Types\ Atomiques, V\}$, $\forall y \in \{Types\ Atomiques\}$, $x : y \rightarrow y$
Fonctionnelle Typée	Position du Max	≥ 1	$\forall (x, y) \in \{Types\ Atomiques\}$, $x : y \rightarrow x$
	Spectre	≥ 1	$\forall (x, y) \in \{Types\ Atomiques\}$, $x : y \rightarrow \frac{1}{x} : y$
	Dérivation	≥ 1	$\forall y \in \{Types\ Atomiques\}$, $x : y \rightarrow \frac{x}{y} : y$ $avec \frac{t}{t} = t, \frac{t}{f} = t^2 \equiv t, \frac{a}{t} = f, \ldots$
	Filtrage temporel	≥ 1	$\forall y \in \{Types\ Atomiques\}$, $t : y \rightarrow t : y$
	Décroissance spectrale	≥ 1	$t : a \rightarrow a$
	Fenêtrage (Hanning)	≥ 1	$\forall (x, y) \in \{Types\ Atomiques\}$, $x : y \rightarrow x : y$
Multi-Dimensionnelle	Moyenne Double	≥ 2 ou « V »	$\forall (x, y, z) \in \{Types\ Atomiques, V\}$, $xy : z \rightarrow z$
Arguments Multiples	Extraction de pics ($Donnee$, N_{pics})	≥ 1 ou « V »	$\forall x \in \{Types\ Atomiques, V\}$, $\forall (y, z) \in \{Types\ Atomiques\}$, $(x : y, z) \rightarrow Vy$
	Corrélation ($Donnee_1$, $Donnee_2$)	≥ 1	$\forall (x, y) \in \{Types\ Atomiques\}$, $(x : y, x : y) \rightarrow \Delta x : y^2 \equiv x : y$ $avec \Delta t = t, \Delta f = f, \Delta a = a$
	Filtrage Passe-bas ($Donnee$, $F_{coupure}$)	≥ 1	$\forall y \in \{Types\ Atomiques\}$, $(t : y, f) \rightarrow t : y$
	Multi-Fenêtrage aléatoire ($Donnee$, $Taille_{fenetre}$)	≥ 1	$\forall (x, y, z) \in \{Types\ Atomiques\}$, $(x : y, z) \rightarrow Vx : y$
	Multi-Fenêtrage régulier ($Donnee$, $Taille_{fenetre}$)	≥ 1	$\forall (x, y, z) \in \{Types\ Atomiques\}$, $(x_1 : y, z) \rightarrow x_2 x_1 : y$

Tab. 4.2: Exemples de tables de typage d'opérations utilisées dans EDS

4.6 Guidage de la construction automatique des fonctions

En l'état actuel, lors de la synthèse d'une fonction, le système de typage permet de vérifier le type de donnée obtenu après chaque opération effectuée,

sans pour autant le contrôler, et de s'assurer du type de résultat final obtenu en sortie de la fonction. Par exemple, si on veut obtenir une fréquence en sortie, il suffit de préciser que la construction s'arrête si la donnée obtenue après la dernière opération est de type « f ».

A partir de ce système de typage, le choix des opérations intermédiaires peut également être guidé par l'introduction de deux types de contrôles supplémentaires sur les fonctions à synthétiser :

- le contrôle structurel, qui consiste à préciser de manière globale les méthodes et traitements qu'on désire appliquer au signal, est réalisé par la spécification des fonctions à construire sous forme d'expressions régulières utilisant des opérateurs génériques ;
- le contrôle opérationnel, qui consiste à orienter le choix local des opérations et de leurs paramètres, nécessite l'utilisation de connaissances apportées au système sous forme d'heuristiques.

4.6.1 Opérateurs génériques

Les opérateurs génériques sont une notation spécifiant une combinaison quelconque d'opérations permettant d'obtenir une donnée d'un type précis. Il peuvent donc remplacer un ou plusieurs opérateurs qui traitent des données de types variés. Par exemple, un *opérateur générique unaire de type* « t » remplace un SEUL opérateur qui permet d'obtenir un temps (« t ») à partir d'une donnée quelconque. Un *opérateur générique multiple de type* « f » remplace une COMPOSITION d'opérateurs qui permet d'obtenir une fréquence (« f ») à partir d'une donnée quelconque.

L'*instantiation* d'un opérateur générique consiste à remplacer celui-ci par un ou des opérateurs réels. En général, un opérateur générique peut être instantié de différentes façons, suivant la palette d'opérateurs dont dispose le système. Par exemple, pour obtenir une valeur d'amplitude (type « a ») à partir d'un signal acoustique (type « $t : a$ »), on peut aussi bien utiliser la moyenne des valeurs d'échantillons que leur variance (opérateurs « Moyenne »ou « Variance ») ; ce sont deux instantiations possibles de cet opérateur générique unaire.

Dans le système EDS, nous avons introduit trois types d'opérateurs génériques notés « ? $_{type}$ » , « * $_{type}$ » , et « ! $_{type}$ » , qui présentent différentes fonctionnalités suivant leur type de donnée de sortie, le nombre d'opérations qu'ils remplacent, et le nombre d'entrées qu'ils peuvent accepter.

Le premier est l'opérateur générique *unaire*, noté « ? $_{type}$ » , qui remplace *un seul* opérateur de type de sortie « *type* ». Par exemple, à partir d'un signal acoustique de type « $t : a$ », « ? $_a$ (Signal) » peut être instantié sous :

- un calcul de moyenne : « Moyenne (Signal) » ;
- un calcul de variance : « Variance (Signal) ».

La moyenne et la variance transformant bien une donnée de type « $t : a$ » en donnée de type « a ». Dorénavant, on pourra noter le type de sortie de chaque opérateur en indice de celui-ci, afin de suivre les types de données le long d'un calcul de fonction. Ainsi, les fonctions précédentes peuvent être notées : « Moyenne$_a$ (Signal$_{t:a}$) »et « Variance$_a$ (Signal$_{t:a}$) ».

Le second est l'opérateur générique *multiple contraint*, noté « * $_{type}$ » , qui remplace *un ou plusieurs* opérateur(s) qui doivent tous avoir « *type* » comme

de type de sortie. Par exemple, à partir d'un signal acoustique de type « $t : a$ », « $*_a$ (Signal) » peut être instantié par :
- une combinaison de 2 opérateurs : « RacineCarrée$_a$ (Moyenne$_a$ (Signal$_{t:a}$)) » ;
- une combinaison de 3 opérateurs : « Log$_a$ (Carré$_a$ (Variance$_a$ (Signal$_{t:a}$))) ».

On constate que chacun des opérateurs utilisés fournit bien une donnée de type « a » en sortie : ainsi, dans le premier exemple, la moyenne transforme le signal en amplitude (transformation de « $t : a$ » vers « a »), puis la racine carrée transforme cette moyenne en une nouvelle valeur d'amplitude (transformation de « a » vers « $\sqrt{a}$ » ≡ « a »).

Enfin, le troisième est l'opérateur générique *multiple non-contraint*, noté « $!_{type}$ » , qui remplace *un ou plusieurs* opérateur(s), qui doivent simplement fournir en sortie une donnée de type « *type* ». Par exemple, à partir d'un signal acoustique de type « $t : a$ », « $!_f$ (Signal) » peut être instantié par :
- une combinaison de deux opérateurs, plus un argument : « Fréquence Centrale$_f$ (Filtrage Passe-Bas$_{t:a}$ (Signal$_{t:a}$, 1000Hz$_f$)) » ;
- une combinaison de trois opérateurs : « Carré$_f$ (Position du Pic Maximal$_f$ (Spectre$_{f:a}$ (Signal$_{t:a}$))) ».

Cette fois, les opérateurs utilisés fournissent des types de données différents en sortie : « $f : a$ », « $t : a$ », « f », etc.... La seule condition est que le type de résultat final soit bien « f ».

Parfois, on peut chercher à obtenir une donnée d'une dimensionnalité précise (en général 0, c'est-à-dire une valeur simple), sans forcément savoir le type de donnée que l'on désire. Nous avons donc introduit dans le système un type de donnée atomique *régulier*, noté « x », qui peut remplacer tous les types de données atomiques réels. Ainsi, l'opérateur générique « $?_x$ » remplace un opérateur fournissant en sortie une donnée de type atomique quelconque, « $*_{Vx}$ » remplace une séquence d'opérateurs fournissant tous en sortie une donnée vectorielle de dimension 1 dont les valeurs sont de type quelconque, et « $!_{Vx:a}$ » remplace une séquence d'opérateurs fournissant en sortie une donnée fonctionnelle de dimension 2, le première dimension évolutive étant non-typée, la seconde étant de type quelconque, et les valeurs étant de type « a ».
Par exemple, l'opérateur générique « $?_x$ (Signal) » peut être instantié en « Max$_a$ (Signal$_{t:a}$) » ou « Fréquence Centrale$_f$ (Signal$_{t:a}$) », les types atomiques « a » et « f » étant tous deux autorisés en sortie.

4.6.2 Structuration des fonctions par patterns

L'introduction des opérateurs génériques permet de spécifier plus précisément la structure des fonctions qu'on veut que le système construise automatiquement, en détaillant les types de données intermédiaires dans des expressions régulières. Ainsi, par combinaison d'opérateurs génériques et d'opérateurs réels, il est possible de spécifier des *schémas de construction* des fonctions, ou « Patterns ». Cette spécification revient alors à restreindre l'espace des fonctions que le système peut construire automatiquement, en leur donnant toutes une structure de base identique.

Par exemple, le pattern simple : « $?_a$ ($?_{t:a}$ (SIGNAL$_{t:a}$)) » restreint l'espace de construction à 2 opérateurs (chaque « ? » remplace un seul opérateur) : le premier transforme le signal d'entrée en un autre signal de type « $t : a$ », puis

le second transforme le signal intermédiaire en une donnée de type « a ». Ce pattern peut être instantié par diverses compositions de 2 opérateurs :

- « Max_a ($\text{Autocorrélation}_{t:a}$ ($\text{Signal}_{t:a}$)) : l'autocorrélation conserve bien le type « $t : a$ », puis l'extraction de l'amplitude du plus grand pic fournit une valeur caractéristique de type « a » ;
- « RMS_a (Filtrage Passe-Bas$_{t:a}$ (Signal_a, 234Hz_f)) : dans ce cas, le premier opérateur générique « ? $_{t:a}$ » est instantié par un opérateur réel de filtrage possédant un argument, la fréquence de coupure, qui est calculée automatiquement lors de la construction des fonctions.

L'utilisation d'opérateurs génériques unaires dans ce pattern restreint énormément l'espace de construction des fonctions. Ainsi, ces opérateurs unaires permettent de contrôler la synthèse des fonctions au niveau local des opérations.

C'est l'utilisation d'opérateurs génériques multiples et de types de données réguliers qui va permettre de couvrir des espaces plus vastes, et de contrôler la synthèse de manière plus générale, au niveau des méthodes globales de traitement du signal initial.

Par exemple, en utilisant le pattern :

$$\text{« ? }_x \text{ (! }_{Vx} \text{ (Split}_{Vt:a} \text{ (* }_{t:a} \text{ (Signal}_{t:a}\text{), 100 :10000ms}_t\text{))) »}$$

on va spécifier un espace de construction dans lequel toutes les fonctions synthétisées respecteront la méthode générale suivante :

- « * $_{t:a}$ (Signal) » : Transformations du signal dans le domaine temporel, c'est-à-dire une ou plusieurs opérations qui vont modifier le signal sans en changer la nature ;
- « Split (Découpage en fenêtres d'observation régulières de 100 à 10000 ms) » : Une opération d'observation du signal transformé sur une suite de fenêtres régulières, de taille quelconque spécifiée par des bornes temporelles 100ms et 10000ms ;
- « ! $_{Vx}$ » : Une ou plusieurs opérations aboutissant à un vecteur de valeurs caractéristiques de type dimension physique quelconque (le type atomique générique « x »pouvant être un temps, une fréquence ou une amplitude) ; on obtiendra en général une valeur caractéristique pour chaque fenêtre d'observation ;
- « ? $_x$ » : Une opération finale réduisant le vecteur de valeurs caractéristiques en une seule valeur caractéristique pour l'ensemble du signal, qui sera le résultat de la fonction synthétisée.

On a ainsi défini par ce pattern une méthode complète de traitement d'un signal acoustique permettant d'obtenir une valeur caractéristique unaire de dimension physique quelconque. Ce pattern permet notamment de construire des fonctions suivant une méthode récurrente en extraction de descripteurs musicaux à partir d'un signal acoustique, consistant à filtrer le signal (traitement dans le domaine temporel), à le découper en fenêtres, à appliquer des traitements spécifiques à chaque fenêtre, puis à agréger ces résultats afin d'obtenir une valeur unique pour l'ensemble du signal. Ce schéma général d'expansion et réduction de la dimensionnalité des données est typiquement utilisé pour l'extraction de divers descripteurs de timbre ([Aucouturier et Pachet, 2002a], [Tzanetakis *et al.*, 2001b]).

Ce pattern général permet de construire virtuellement une infinité de fonctions, et peut par exemple être instantié en :

- Sum_a (Square_{Va} (Mean_{Va} ($\mathrm{Split}_{Vt:a}$ ($\mathrm{HpFilter}_{t:a}$ ($\mathrm{SIGNAL}_{t:a}$, $1000\mathrm{Hz}_f$), $1000\mathrm{ms}_t$))))
- $\mathrm{Log10}_a$ ($\mathrm{Variance}_a$ (NPeaks_{Va} ($\mathrm{Split}_{Vt:a}$ ($\mathrm{Autocorrélation}_{t:a}$ ($\mathrm{Signal}_{t:a}$), $100\mathrm{Hz}_f$), $200\mathrm{ms}_t$)))

Ces deux fonctions respectent la structure générale de traitement du signal, tout en restant indépendantes.

Le pattern présenté précédemment est linéaire, dans la mesure où il ne représente qu'une séquence d'opérations typées. Cependant, il est possible de spécifier des patterns non-séquentiels en y introduisant de nouvelles branches fonctionnelles. Par exemple, dans le pattern précédent, l'introduction de opérateur réel « Split »permet de forcer cette opération avec une taille de fenêtre spécifiée numériquement par des bornes temporelles. Mais cette taille de fenêtre pourrait également résulter du calcul d'une valeur temporelle caractéristique du signal, par l'opération générique « $!_t$ (Signal) » . Le pattern alors spécifié de la façon suivante : « $?_x$ ($!_{Vx}$ ($\mathrm{Split}_{Vt:a}$ ($*_{t:a}$ ($\mathrm{Signal}_{t:a}$), $!_t$ ($\mathrm{Signal}_{t:a}$)))) », permet de construire des fonctions à deux branches, l'une calculant la taille de fenêtre et l'autre réalisant des traitements temporels du signal.

4.6.3 Guidage par heuristiques

L'utilisation des règles de typage assure la cohérence syntaxique et l'homogénéité physique des fonctions. La spécification de patterns de construction permet d'imposer des types d'opérations précis, et de traduire ainsi une méthode générale de synthèse de fonction en limitant l'espace des opérations possibles.

Cependant, le système ne possède toujours aucune information quant au choix effectif des opérations à utiliser dans ces fonctions et de leurs paramètres. Autrement dit, le typage seul ne permet de créer des fonctions structurées précisément, mais dont les opérations sont choisies uniquement de manière aléatoire.

Afin de guider la synthèse des fonctions, dans le but d'optimiser leur pertinence à priori, il est donc nécessaire d'introduire de nouvelles règles : les heuristiques de construction, qui vont orienter les choix de certaines opérations et de certains paramètres, qui sembleront à priori plus pertinents pour résoudre le problème posé. Elles peuvent être inhérentes au système, ou apportées pour résoudre un problème spécifique (par exemple en précisant la bande de fréquences caractéristique de la voix humaine, afin de résoudre un problème de caractérisation de la voix chantée dans des signaux acoustiques). Elles sont donc la transcription d'une expertise en traitement du signal, utilisable de manière positive, en favorisant des fonctions considérées comme intéressantes, ou négative, en rejetant des fonctions qui sont de manière évidente inintéressantes.

Du fait de la représentation des fonctions dans EDS sous forme d'arbres d'opérateurs, il est nécessaire d'exprimer les heuristiques sous forme de conditions sur les opérateurs contenus dans ces arbres. Ainsi, une heuristique permet d'associer un *score* à un opérateur qu'on désire appliquer sur un *fils* donné (ou sur un ensemble de *fils* dans le cas d'opérateurs à plusieurs entrées), suivant les opérateurs et paramètres composant le fils. Ces scores sont ensuite utilisés par le système pour choisir les meilleurs candidats lors de la construction de nouvelles fonctions.

Dans EDS, nous avons choisi un barème simple de notation des heuristiques associant un score entre 0 et 10 à un opérateur (voir la figure 4.4), applicable

sur un fils donné, permettant de rendre compte de la pertinence de l'opération.

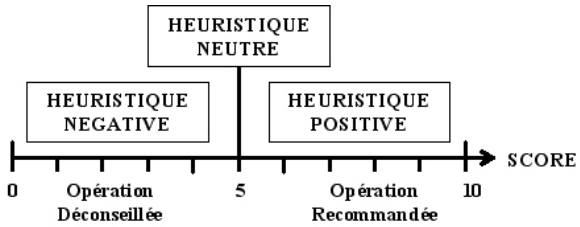


Fig. 4.4: Barème de notation des heuristiques

Ainsi, une heuristique s'exprime toujours par rapport aux opérations déjà réalisées par le système, et s'utilise donc de manière ascendante lors de la création d'une fonction. Par exemple, considérons l'heuristique suivante : « Un *filtrage passe-bas* est conseillé si aucun filtrage *général* n'a été fait sur le signal jusqu'à présent ($score = 7$), et interdit si un autre filtrage *passe-bas* a déjà été réalisé ($score = 0$) »Elle permet de donner des scores différents pour l'opération « Filtrage passe-bas », suivant les opérations déjà réalisées, par exemple :

- après le signal initial brut « $\text{Signal}_{t:a}$ », l'heuristique donne le score 7/10, donc la création de la fonction « Filtrage $\text{passe-bas}_{t:a}$ ($\text{Signal}_{t:a}$) »est favorisée ;
- après le signal filtré passe-bas « Filtrage $\text{passe-bas}_{t:a}$ ($\text{Signal}_{t:a}$) », l'heuristique donne le score 0/10, donc la création de la fonction « Filtrage $\text{passe-bas}_{t:a}$ (Filtrage $\text{passe-bas}_{t:a}$ ($\text{Signal}_{t:a}$)) »est interdite ;
- après le signal filtré passe-haut « Filtrage $\text{passe-haut}_{t:a}$ ($\text{Signal}_{t:a}$) », l'heuristique ne précise rien, donc donne le score neutre par défaut 5/10, et n'intervient pas dans la création de la fonction « Filtrage $\text{passe-haut}_{t:a}$ (Filtrage $\text{passe-bas}_{t:a}$ ($\text{Signal}_{t:a}$)) », qui n'est ni favorisée, ni déconseillée.

L'utilisation des heuristiques dans EDS offre différents types de contôle sur les fonctions créées : opérationnel, structurel, ou numérique.

Contrôle des opérations

Le *contrôle opérationnel* permet de s'assurer de la pertinence des opérations choisies dans une fonction. Par exemple, on peut exprimer le fait que « *Il peut être intéressant de réaliser 2 calculs de spectre consécutifs, mais en général inutile d'en réaliser un troisième* » , en spécifiant une heuristique du type « *Pour une opération de* calcul de spectre*, si la dernière opération du fils est un calcul de spectre, score* = 2 *si l'avant-dernière opération du fils est également un calcul de spectre, et score* = 7 *sinon* » . Cette heuristique de favoriser la création de fonctions originales se rapprochant des coefficients cepstraux (2 calculs de spectre), tout en maîtrisant le nombre d'opérations originales. On peut également favoriser des opérations, par exemple « *Favoriser le découpage du signal initial en fenêtres d'observation régulières* » , en spécifiant une heuristique du type « *Pour une opération de* découpage régulier*, score* = 7 *si aucune opération n'a été réalisée sur le fils, c'est-à-dire si le fils est le signal initial* » . Cette dernière heuristique peut être associée à la « *Favorisation de l'observation d'un signal découpé suivant des fenêtres de Hanning* » , en spécifiant une

heuristique du type « *Pour une opération de* fenêtrage de Hanning*, score* = 9 *si la dernière opération du fils est un découpage en fenêtres (Split)* » .

Contrôle de la structure

Le *contrôle structurel* permet de s'assurer de validité de la taille des fonctions créées. Par exemple, on peut exprimer le fait que « *La valeur de la fréquence de coupure d'un filtrage quelconque doit résulter d'un nombre minimal d'opérations, si possible aucune* » , en spécifiant une heuristique du type « *Le score d'une opération de* filtrage *dépend du nombre d'opérations utilisées pour calculer sa fréquence de coupure, suivant la formule score* $= Max(0, 5 - N)$*, N étant le nombre d'opérations de la branche calculant la fréquence de coupure* » .
Ainsi, si la fréquence de coupure est une valeur brute ne provenant d'aucun calcul, alors $N = 0$ et $score = 5$, l'heuristique n'intervient pas dans le choix de l'opération. Par contre, si la fréquence de coupure est provient d'une opération, par exemple « Fréquence Centrale (Signal) », alors $N = 1$ et $score = 4$, l'opération est légèrement déconseillée, etc. . .Si le calcul de la valeur de la fréquence résulte de plus de 5 opérations, alors $score = 0$, et l'opération de filtrage est interdite.
Cette heuristique permet d'éviter la création de fonctions complexes pour calculer un paramètre simple comme la fréquence de coupure.

Contrôle des valeurs des paramètres

Enfin, le *contrôle numérique* permet de s'assurer de la validité des valeurs des paramètres des opérations utilisées. Par exemple, on peut exprimer le fait que « *La taille d'une fenêtre d'observation doit être obligatoirement supérieure à 1ms* » ou « *La fréquence de coupure d'un filtre passe-haut doit être obligatoirement supérieure à 100Hz* » , en spécifiant des heuristiques du type « *Pour une opération de* fenêtrage*, score* $= 0$ *si la valeur de l'argument de taille de fenêtre est inférieure à 1ms* » ou « *Pour une opération de* filtrage passe-haut*, score* $= 0$ *si la valeur de l'argument de fréquence de coupure est inférieure à 100Hz* » . Ces heuristiques permettent d'éviter la création d'opérations avec des paramètres non-pertinents.

Ainsi, chaque heuristique du système permet d'évaluer la pertinence d'un couple (opération à effectuer, fils existant) suivant une règle précise, sous forme d'un score borné. L'ensemble des heuristiques forme un groupe de connaissances utilisé lors de la construction d'une fonction, pour la décision finale des opérations utilisées et de leurs paramètres.

4.6.4 Choix des opérations suivant les types et heuristiques

La construction d'une fonction nouvelle se fait séquentiellement, par ajout d'une opération à une fonction pré-existante appelée *fils* (voir en section 4.3.4). A chaque choix d'opération à ajouter, le système doit prendre en compte l'ensemble des règles comprises dans le système : le typage et les heuristiques, suivant l'algorithme général suivant :

1. on considère une fonction initiale, appelée *fils*, sur laquelle on désire rajouter une nouvelle opération

2. Prise en compte des règles de typage : le système sélectionne parmi l'ensemble des opérations, celles qui acceptent en entrée le type de donnée fourni par le fils
3. Prise en compte des heuristiques : pour chaque opérateur OP sélectionné, le système :
   - liste l'ensemble des heuristiques concernant OP
   - évalue le score donné par chaque heuristique listée pour le couple (OP,*fils*)
   - on calcule un *score global* pour le couple (OP,*fils*), en agrégeant tous les scores donnés par l'ensemble des heuristiques
4. Le système choisit une opération parmi les possibles, par *tirage au sort pondéré* par le score de chaque opération (voir la figure 4.5)
5. Finalement, l'opération choisie est appliquée sur le *fils*

Le calcul du score global d'un opérateur se fait par agrégation des scores de toutes les heuristiques concernant cet opérateur. Si une heuristique fournit le score 0 pour cet opérateur, alors le score global vaut 0 (opération interdite). Sinon, le calcul typique du score global consiste à prendre la moyenne de tous les scores, sans tenir compte des scores neutres ($score = 5$) qui n'apportent aucune information sur la pertinence de l'opération :

$$Score\ Global = \begin{cases} 0, \text{si } \exists h \in Heuristiques \mid Score(h) = 0 \\ \\ MOYENNE(Score(h)), sinon \\ {\scriptstyle h \in Heuristiques} \\ {\scriptstyle Score(h) \notin \{0,5\}} \end{cases}$$

Il est également possible d'utiliser une méthode d'agrégation plus stricte, qui éliminera toutes les opérations déconseillées par une quelconque heuristique, en choisissant comme score global le plus mauvais score non-neutre de l'ensemble des heuristiques, c'est-à-dire utiliser le MIN à la place de la MOYENNE.

C'est ce score global qu'on utilise ensuite comme pondération dans le tirage au sort, pour le choix final de l'opérateur. On réalise un tirage au sort de type Monte-Carlo : on alloue à chaque opérateur une région de l'espace dont la taille dépend de son score global, puis on tire aléatoirement un point dans l'espace, et on choisit l'opérateur dont la région contient ce point. Comme le montre la figure 4.5, le choix d'un ordre de pondération va déterminer l'importance qu'on donne aux bonnes heuristiques par rapport aux mauvaises, en proportion de l'espace total.
A l'ordre 1, la taille de la région d'un opérateur est directement proportionnelle à son score. A l'ordre N, la taille de la région d'un opérateur est proportionnelle à son score puissance N, la taille de référence des heuristiques neutres étant préservée par une normalisation par la note neutre 5, favorisant d'autant plus les opérations aux scores supérieurs à 5 par rapport à celles de scores inférieurs à 5.

Dans le système EDS, une pondération d'ordre 2 offre un bon compromis entre souplesse de choix des opérations et respect des heuristiques.

Nous disposons maintenant de tous les outils pour réaliser une construction automatique et intelligente des fonctions, le paragraphe suivant présente donc comment ces outils sont intégrés dans le système.

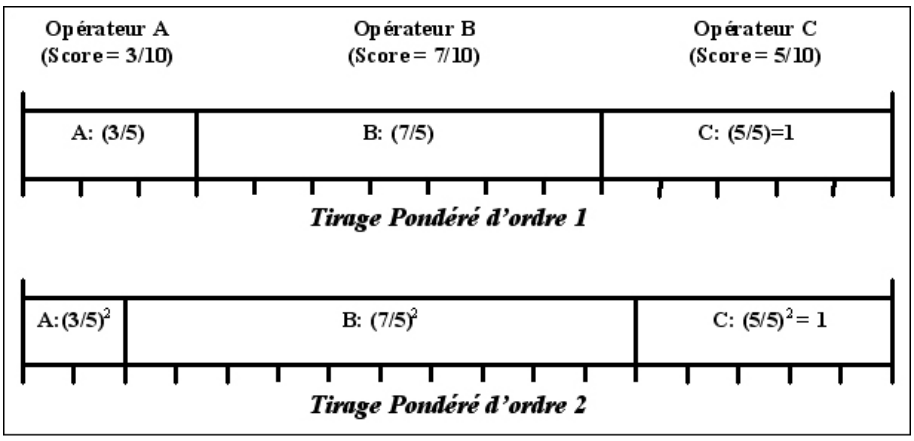


Fig. 4.5: Choix d'une opération parmi trois par tirage pondéré sur leur score global : espaces de tirage pour différents ordres de pondération

4.6.5 Instantiation automatique de fonctions à partir de patterns et d'heuristiques

Pour créer une fonction, il suffit de spécifier une structure générale sous forme de pattern. Le système instantie automatiquement ce pattern en remplaçant ses opérateurs génériques par des opérations choisies en utilisant les règles de typage et les heuristiques.

On peut alors détailler la méthode générale de construction automatique de fonctions, présentée en section 4.3.4, pour la génération de fonctions spécifiées par des patterns.

Un pattern est composé de quatre types d'éléments : le signal initial, les opérateurs génériques, les opérateurs réels, et les arguments externes. L'instantiation d'un pattern revient à instantier successivement dans chaque branche, chaque opérateur générique, en partant du signal initial, et en vérifiant les heuristiques à chaque nouvelle opération, suivant l'algorithme suivant :

1. on part du signal étudié (argument d'entrée principal), appelé *fils*
2. si l'opérateur à appliquer sur le *fils* est générique, on l'*instantie (voir en dessous)*
   - si l'instantiation réussit, on l'applique sur le *fils*
   - sinon, le pattern n'est pas instantiable : STOP
3. sinon, si l'opérateur à appliquer sur le *fils* est réel, on évalue son score global sur l'ensemble des heuristiques
   - si le score global est supérieur à un seuil (en général 0), on l'applique sur le *fils*
   - sinon, le pattern n'est pas instantiable : STOP
4. le résultat de l'opération est un nouveau *fils*
   - si le système a encore une opération à appliquer au *fils*, on recommence la méthode à partir du (2)
   - sinon, le système renvoie la fonction obtenue : STOP

L'algorithme d'instantiation d'un pattern repose ainsi sur l'algorithme suivant, qui permet d'instantier tous les types d'opérateurs génériques (unaire,

multiple contraint ou non-contraint) :

1. on part d'un *fils* qui est un ensemble d'opérations réelles (non-génériques) appliquées sur le signal initial
2. on considère la première opération de l'opérateur générique ($op = 1$)
3. le système choisit une opération réelle à appliquer sur le *fils*, suivant l'algorithme de *Choix des opérations en fonction des types et des heuristiques* présenté précédemment
4. si le système ne trouve pas d'opérateur convenable,
   - si la fonction précédente est une instantiation acceptable de l'opérateur générique, on renvoie cette fonction : STOP
   - sinon, le pattern n'est pas instantiable : STOP
5. on instantie la op^{eme} opération en l'appliquant sur le *fils*
6. le système décide si l'instantiation de l'opérateur nécessite l'ajout d'une nouvelle opération (voir ci-dessous) ; si oui, la fonction courante est alors un nouveau *fils*, sur lequel on essaye d'appliquer une nouvelle opération ($op = op + 1$), en recommençant à partir du (3)
7. construction terminée : on renvoie la fonction obtenue : STOP

En (6), la décision de l'ajout d'une nouvelle opération dépend du type d'opérateur générique considéré.
Dans le cas d'un opérateur générique unaire, $op \leq 1$, donc l'algorithme réalise une seule itération et on n'ajoute pas d'opération.
Sinon, dans le cas d'un opérateur générique multiple, le nombre d'opérations n'étant pas défini à priori, la décision dépend du type de donnée obtenu jusqu'à présent :

- si le type obtenu est le type attendu en sortie, on réalise un tirage au sort pour déterminer si on arrête la construction à ce point ou si on ajoute une opération supplémentaire ;
- si le type obtenu n'est le type désiré en sortie de l'opérateur générique, on continue la construction.

Cette section a présenté l'ensemble des règles et algorithmes utilisés dans EDS afin de réaliser la synthèse automatique et intelligente de fonctions à partir d'un ensemble d'opérateurs et de heuristiques de construction. La section suivante montre comment le système va évaluer la capacité de ces fonctions à résoudre un problème descriptif donné.

4.7 Pertinence des fonctions caractéristiques

La méthode de création automatique de fonctions dans le système EDS permet de s'assurer que ces dernières sont mathématiquement justes, physiquement censées, et *a priori* pertinentes pour décrire des signaux, grâce à l'utilisation d'heuristiques générales de construction.

Cependant, les heuristiques générales seules, aussi précises soient elles, sont insuffisantes pour trouver directement les meilleures fonctions caractéristiques pour un problème descriptif donné. Ainsi, il est nécessaire d'évaluer pour chaque problème descriptif considéré, si les fonctions créées sont réellement pertinentes.

Pour cela, il est nécessaire d'observer les résultats de l'application de cette fonction sur l'ensemble des signaux de la base d'apprentissage étiquetée, et d'évaluer si cette représentation facilite la résolution du problème. Ainsi, on va calculer la fonction sur chaque signal de la base, avant de calculer la pertinence des valeurs obtenues par rapport aux étiquettes des signaux.

4.7.1 Application d'une fonction sur un signal

Dans EDS, les fonctions sont représentées sous forme d'arbres d'opérations appliquées sur un signal initial. Le calcul d'une fonction sur un signal donné se fait alors facilement par application récursive des différentes opérations, à partir de l'opération finale, en descendant l'arbre jusqu'au signal d'entrée, suivant l'*algorithme de calcul récursif d'une fonction « f »* :

1. considérer la fonction f comme la fonction courante f_c
2. compter le nombre d'entrées N de la fonction courante f_c :
3. si $N > 0$:
   - la fonction courante est constituée d'une opération finale *Operation* appliquée sur N entrées (ou fils), chaque fils étant lui-même un arbre d'opérations : $f_c = Operation(fils_1, \ldots, fils_N)$
   - calculer toutes les fonctions d'entrée ($fils_1$, ..., $fils_N$) de la fonction courante f_c, en utilisant cet algorithme
   - appliquer l'opération finale *Operation* sur les fonctions d'entrée ($fils_1$, ..., $fils_N$), et retourner le résultat : STOP.
4. si $N = 0$:
   - la fonction courante f_c est une donnée fixée : soit le signal d'entrée, soit un signal constant, soit un paramètre constant
   - décoder la donnée f_c dans un format compréhensible par le système EDS

Afin de permettre au système de manipuler les données et de leur appliquer les différentes opérations, nous utilisons un format homogène de représentation compréhensible par le système EDS, suivant la dimensionnalité des données considérées (voir section 4.4 : sous forme unaire, vectorielle, matricielle, ou d'arbre pour les dimensions d'évolution supérieures à 3). C'est pourquoi il est nécessaire en (4) de décoder les entrées (signal principal, signaux externes, paramètres fixes) vers ce format.

Le calcul effectif du résultat de chaque opération est réalisé par un programme exécutable « $Operation_{Exe}(input_1, \ldots, input_N)$ », qui renvoie un résultat au format EDS, à partir de N données d'entrée (les résultats du calcul des N fils) également au format EDS.

Par exemple, pour le calcul de la fonction
« $f(Signal.wav) = Max(Autocorrélation(Signal.wav))$ »
à partir d'un signal acoustique enregistré au format .wav :

- $Fonction\ courante = Max(Autocorrélation(Signal.wav)) = Max(fils_1)$, avec $fils_1 = Autocorrélation(Signal.wav)$.
- On calcule $fils_1$ récursivement :
  - $Fonction\ courante_1 = Autocorrélation(Signal.wav)$
    $= Autocorrélation(fils_{11})$, avec $fils_{11} = Signal.wav$.
  - On calcule $fils_{11}$ récursivement :
    . $Fonction\ courante_{11} = Signal.wav$: c'est le signal d'entrée

. On décode $Fonction\ courante_{11}$, en enregistrant les amplitudes des échantillons du fichier .wav au format EDS
. On renvoie la valeur de $fils_{11} = Fonction\ courante_{11}$
* On lance l'exécutable $Autocorrélation_{Exe}$, pour obtenir la valeur intermédiaire : $Fonction\ courante_1 = Autocorrélation_{Exe}(fils_{11})$
* On renvoie la valeur de $fils_1 = Fonction\ courante_1$

– On lance l'exécutable Max_{Exe}, pour obtenir la valeur finale : $Fonction\ courante = Max_{Exe}(fils_1)$
– On renvoie la valeur de $f(signal.wav) = Fonction\ courante$

Le fonctionnement est bien entendu valable pour des fonctions plus complexes à branches et entrées multiples, du genre : « $f(Signal.wav) = Max(Corrélation(Filtre\ passe\text{-}bas(Signal.wav, 400Hz), Signal_{ref}.wav))$ »

4.7.2 Calcul de la pertinence d'une fonction

L'évaluation de la pertinence d'une fonction sur un problème donné, défini par une base de N signaux étiquetés, consiste à comparer les résultats du calcul de la fonction sur l'ensemble des signaux de la base, avec les étiquettes de ces signaux. Il suffit alors de construire une *fonction de pertinence* à partir des deux séries de N valeurs : les N résultats de la fonction, et les N étiquettes des signaux de la base.

La fonction de pertinence utilisée dépend du type de problème (présentés en 4.2.1), et de la dimensionnalité de la fonction (unaire ou vectorielle) considérés.

Nous présentons en annexe D deux mesures simples utilisées dans EDS comme critères d'optimisation pour les fonctions créées :

– pour les problèmes de *régression*, le coefficient de corrélation de Pearson évalue la qualité de la relation linéaire entre les fonctions et les étiquettes ;
– pour les problèmes de *classification*, le ratio discriminant de Fisher évalue le pouvoir discriminant de la fonction entre les différentes classes du problème.

Plus généralement, pour tous les types de problèmes, et tous les types de fonctions (c'est-à-dire également les fonctions multi-dimensionnelles, dont le résultat est non pas une seule mais un ensemble de valeurs numériques), il est possible de calculer la pertinence sous forme d'une performance d'apprentissage d'un modèle, à partir d'une méthode de classification/régression, et d'un critère de performance. Ceci consiste à réaliser l'apprentissage d'un modèle du descripteur étudié, en utilisant uniquement la fonction à évaluer. Ce modèle peut être une simple transformation linéaire de la fonction, ou un modèle plus évolué utilisant par exemple des réseaux de neurones (voir section 4.9). Une fois l'apprentissage réalisé, la qualité du modèle est évaluée par validation croisée sur la base d'apprentissage, pour un critère donné comme le taux d'éléments bien classés, ou le coefficient de corrélation avec les étiquettes.

4.8 Recherche de fonctions caractéristiques optimales

La mesure de pertinence définie précédemment est le critère à optimiser pour obtenir des fonctions caractéristiques pertinentes pour la résolution d'un problème donné.

Pour cela, il doit parcourir l'espace des fonctions possibles afin de trouver toutes les fonctions optimales, c'est-à-dire dont la pertinence est localement maximale. Cependant, du fait du nombre d'opérateurs et du nombre de paramètres possibles pour chaque opérateur, cet espace contient des millions de fonctions potentielles, et il n'est pas envisageable de le parcourir de manière exhaustive.

Il faut donc réduire l'espace de recherche en n'explorant que des zones potentiellement intéressantes. Le parcours aléatoire de l'espace étant trop incertain quant à la pertinence des fonctions obtenues, il est nécessaire de définir des règles pour optimiser le parcours de cet espace de fonctions, en orientant la recherche sur des zones précises de l'espace.

Pour cela, nous avons élaboré une méthode d'optimisation locale des fonctions basée sur une technique de programmation génétique, c'est-à-dire l'application de la recherche génétique ([Holland, 1975], dont une présentation simple est faite dans [Beasley *et al.*, 1993a] et [Beasley *et al.*, 1993b]), à l'exploration d'un univers de fonctions ([Goldberg, 1989]), réputée pour l'exploration de larges espaces de fonctions ([Koza, 1992]).

4.8.1 Algorithme de recherche génétique

L'algorithme de recherche génétique utilisé dans EDS permet de construire une population de fonctions à partir d'un pattern donné, dont les meilleures sont optimales pour le problème descriptif considéré. Il fonctionne globalement en 5 étapes, à partir d'une base de signaux numériques étiquetés par leurs valeurs perceptives pour un descripteur donné, et d'un pattern P :

1. Genèse d'une population
   Création d'une population initiale P_0 de N fonctions aléatoires à partir du pattern P ($i = 0$)
2. Évaluation de chaque individu
   Calcul de la mesure de pertinence de chaque fonction de la population P_i pour le problème considéré (voir section 4.7.2)
3. Si (Test d'arrêt de l'algorithme) : STOP et on renvoie la population P_i.
4. Sinon : Sélection et diversification de la population
   Création d'une nouvelle population de fonctions P_{i+1}
5. Itération en (2)

Le « test d'arrêt de l'algorithme » est accepté si une des conditions suivantes est vérifiée :

- Un nombre d'itérations maximum est atteint ; typiquement, la recherche est arrêtée à la population P_{1000} ;
- La population courante contient une fonction suffisamment pertinente, c'est-à-dire dont la mesure de pertinence est supérieure à un certain seuil ; la recherche est notamment arrêtée si la corrélation des valeurs d'une fonction avec les étiquettes de la base vaut 1 : la fonction est alors un modèle linéaire parfait du descripteur ;
- La pertinence des fonctions de la population n'augmente plus pendant plusieurs itérations ; on peut par exemple arrêter la recherche si la pertinence de la meilleure fonction de la population P_i est égale à la pertinence de la meilleure fonction de la population P_{i-5} (5 populations successives sans amélioration).

Les règles de création d'une nouvelle population utilisées en (4), sont présentées dans le paragraphe suivant.

4.8.2 Création d'une nouvelle population par transformations génétiques

Afin de s'assurer que les populations successives sont de plus en plus pertinentes lors de la recherche génétique, la création d'une nouvelle population est basée sur les meilleures fonctions de la population précédente : ce sont les étapes 3 et 4 de l'algorithme génétique, sélection et diversification.

En général, les algorithmes génétiques utilisent des méthodes de sélection diverses suivant les résultats recherchés : ordonnancement (« N/2 élitisme »), tournoi, Monte-Carlo « roulette wheel » ou Monte-Carlo modifié « Stochastic remainder without replacement »([Goldberg, 1989]), ...

Ces méthodes permettent de sélectionner les individus d'une population en favorisant plus ou moins ceux qui ont les meilleurs résultats sur la fonction d'adaptation. Des méthodes additionnelles, comme la mise à l'échelle, ou le partage et le nichage des individus, permettent de contrôler la sélection plus précisément, notamment pour faciliter l'émergence de solutions multimodales ([Mahfoud, 1995]).

Ainsi, à partir de ces diverses méthodes, nous avons élaboré une méthode de génération de population spécifique à EDS, une nouvelle population P_{i+1} étant composée :

- des meilleures fonctions de la population P_i : afin de toujours conserver les meilleures fonctions obtenues jusqu'à présent, et de pouvoir tester si leur pertinence s'améliore (test d'arrêt de l'algorithme) ; typiquement, on conserve les 20% de fonctions les plus pertinentes, et on assure l'indépendance des fonctions dans cette sélection en éliminant les fonctions trop corrélées à une fonction plus pertinente ;
- de nouvelles fonctions obtenues par transformations génétiques des meilleures fonctions de P_i (transformations présentées plus loin) : afin de recomposer les opérations effectuées localement dans diverses fonctions pertinentes pour construire des fonctions encore plus pertinentes ; typiquement 60% des fonctions de la nouvelle population sont créées ainsi ;
- de nouvelles fonctions obtenues aléatoirement à partir du pattern de base P : afin de constituer un « vivier »de fonctions originales dans lequel l'algorithme peut trouver de nouvelles compositions pertinentes d'opérations, à intégrer dans les fonctions courantes par transformation génétique ; typiquement, 20% de la population est ainsi créée aléatoirement.

En recherche génétique, les transformations usuelles sont le croisement et la mutation des individus. Dans EDS, nous avons élaboré des transformations génétiques supplémentaires, permettant de modifier les fonctions à divers niveaux de profondeur :

Optimisation numérique

C'est une optimisation locale permettant d'ajuster les valeurs des paramètres des opérations. Elle est associée à une transformation génétique de fonction précise : le *clonage structurel*, qui consiste à conserver l'ensemble de la structure

des opérations constituant une fonction, et à ne modifier que les valeurs des paramètres numériques de ces opérations.

Par exemple, la fonction « Rms (Filtre Passe-haut (Signal, 1000Hz)) »possède un paramètre numérique : la fréquence de coupure du filtre passe-haut à 1000Hz.

Son clonage va donc modifier cette valeur de fréquence, et pourra par exemple donner les fonctions « Rms (Filtre Passe-haut (Signal, 800Hz)) »(voir figure 4.6), ou « Rms (Filtre Passe-haut (Signal, 1212Hz)) ».

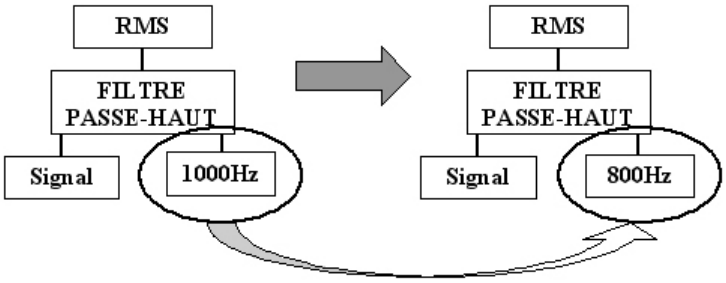


Fig. 4.6: Exemple de clonage d'une fonction

Optimisation opérationnelle

C'est une optimisation locale permettant d'ajuster les choix d'opérations pour obtenir une donnée d'un type précis. Elle est associée à deux types de transformations génétiques des fonctions : la *substitution* et l'*addition*.

La *substitution* consiste à choisir une opération précise dans une fonction, et à la remplacer par une opération équivalente, c'est-à-dire fournissant une donnée de type équivalent.

Par exemple, la substitution de l'opération de filtrage passe-haut dans la fonction « Rms_a ($\mathit{Filtre\ Passe\text{-}haut}_{t:a}$ ($\mathrm{Signal}_{t:a}$, $1000\mathrm{Hz}_f$)) », pourra donner les fonctions « Rms_a ($\mathit{Filtre\ Passe\text{-}bas}_{t:a}$ ($\mathrm{Signal}_{t:a}$, $340\mathrm{Hz}_f$)) », ou « Rms_a ($\mathit{Autocorrélation}_{t:a}$ ($\mathrm{Signal}_{t:a}$)) »(voir figure 4.7).

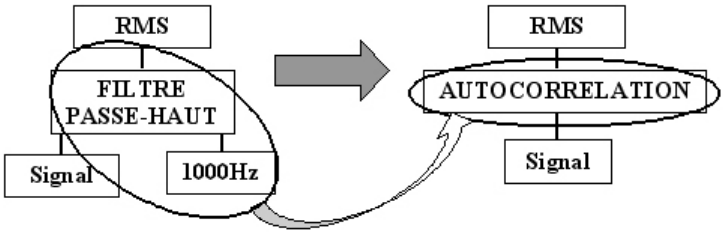


Fig. 4.7: Exemple de substitution d'un opérateur dans une fonction

L'*addition* consiste à insérer une nouvelle opération à un endroit précis dans une fonction, qui ne modifie pas le type de donnée, afin de tester de nouveaux pré- ou post-traitements.

Par exemple, l'addition d'un opérateur à la fonction « Rms_a ($\text{Spectre}_{f:a}$ ($\text{Signal}_{t:a}$)) », pourra donner les fonctions « Rms_a ($\text{Spectre}_{f:a}$ (*Filtre Passe-bas*$_{t:a}$ ($\text{Signal}_{t:a}$, 1000Hz_f))) »(voir figure 4.8), ou « *Log*$_a$ (Rms_a ($\text{Spectre}_{f:a}$ ($\text{Signal}_{t:a}$))) ».

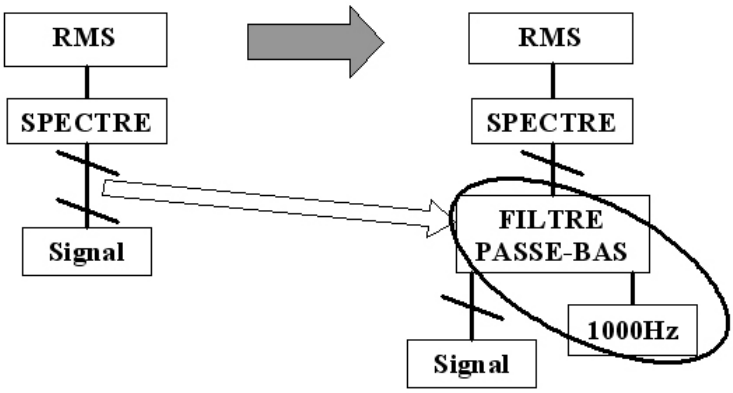


Fig. 4.8: Exemple d'addition d'un opérateur à une fonction

Optimisation structurelle

C'est une optimisation globale permettant d'ajuster les choix de méthodes complètes de traitement, par modification d'un ensemble d'opérations. Elle est associée à deux types de transformations génétiques des fonctions : la *mutation* et le *croisement*.

La *mutation* consiste à choisir un ensemble d'opérations dans une fonction, et à le remplacer par un ensemble d'opérations fournissant une donnée de type équivalent.
Par exemple, la mutation de « Position du Pic_f ($\text{Spectre}_{f:a}$ ($\text{Signal}_{t:a}$)) »pourra donner les fonctions « Position du Pic_f (*Carré*$_{f:a}$ (*Spectre*$_{f:a}$ (*Autocorrélation*$_{t:a}$ ($\text{Signal}_{t:a}$)))) »(voir figure 4.9), ou « *Max*$_f$ (*Position du Pic*$_{Vf}$ (*Spectre*$_{Vf:a}$ (*Fenêtrage*$_{Vt:a}$ ($\text{Signal}_{t:a}$)))) ».

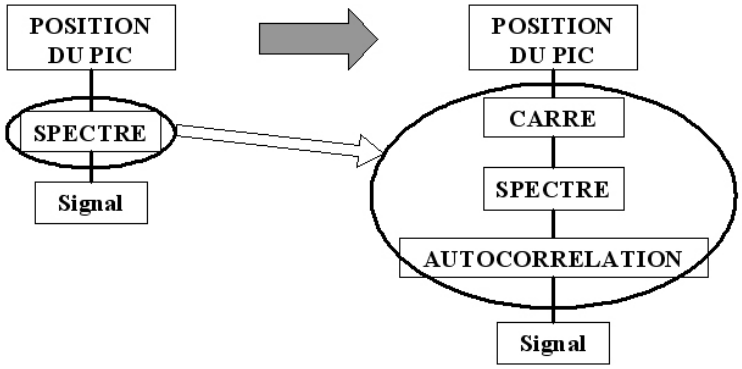


Fig. 4.9: Exemple de mutation d'une fonction

Le *croisement* de deux fonctions consiste à choisir un ensemble d'opérations dans chaque fonction, et à les combiner afin d'obtenir une nouvelle fonction. Par exemple, le croisement des fonctions « *Position du Pic (Spectre (Signal))* » et « Rms (Filtre passe-haut (Signal, 1000Hz)) », pourra donner les fonctions « *Position du Pic (Spectre* (Filtre passe-haut (Signal, 1000Hz))) »(voir figure 4.10), ou « Rms (*Spectre (Signal)*) ».

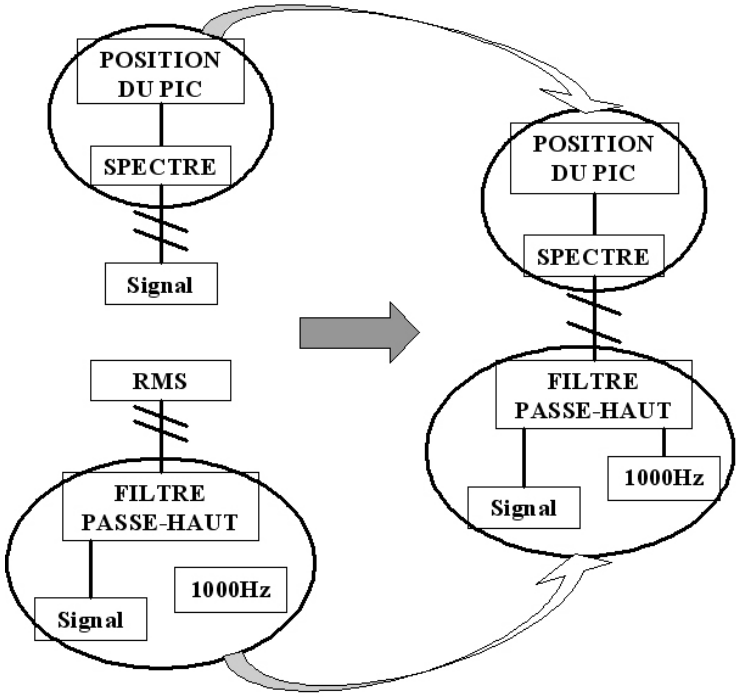


Fig. 4.10: Exemple de croisement de deux fonctions

4.8.3 Recherche d'un ensemble de fonctions pertinentes

L'algorithme génétique présenté ci-dessus permet de générer une population optimisée pour résoudre un problème descriptif donné.

Cependant, cette population finale ne peut pas être considérée comme un ensemble de fonctions à utiliser directement comme caractéristiques pour la modélisation.

En effet, du fait des règles de création des nouvelles populations, la population finale obtenue à l'arrêt de l'algorithme de recherche contient :

- les meilleures fonctions obtenues jusqu'à présent ;
- des fonctions dérivées des meilleures fonctions par transformations génétiques ;
- des fonctions construites aléatoirement.

Par définition du critère d'arrêt de la recherche (voir ci-dessus), soit la meilleure fonction est un modèle parfait du descripteur, soit l'algorithme n'arrive plus à construire de meilleure fonction par des transformations locales

(numériques ou opérationnelles). On considère donc que la meilleure fonction de la population est un optimum local de l'espace des fonctions pour résoudre le problème.

Les fonctions suivantes, par construction indépendantes de cette meilleure fonction (voir ci-dessus : création d'une nouvelle population), correspondent à des zones pertinentes de l'espace des fonctions qui restent à optimiser. En effet, l'algorithme s'est arrêté dès la découverte de l'optimum de la meilleure fonction, et ne garantit pas que l'optimisation des fonctions suivantes était terminée.

Ainsi, on ne conserve pour la modélisation du descripteur que la meilleure fonction d'une population. Puis, afin de trouver de nouvelles fonctions, on relance l'algorithme de recherche génétique à partir d'une nouvelle population initiale constituée des meilleures fonctions de la population finale de la recherche génétique précédente, à l'exception de la meilleure fonction et des fonctions qui y sont trop corrélées (typiquement à plus de 0.5), et complétée par des nouvelles fonctions construites aléatoirement.

4.8.4 Amélioration des performances de la recherche

D'un point de vue exploratoire, la recherche de fonctions pertinentes est optimisée par :

- la prise en compte des règles de typage qui assurent la construction de fonctions physiquement correctes ;
- la spécification de patterns, qui orientent la recherche dans des régions pertinentes de l'espace des fonctions ;
- la prise en compte d'heuristiques de traitement qui apportent une expertise supplémentaire lors de la construction des fonctions ;
- l'utilisation d'un algorithme de type génétique qui permet de parcourir l'espace de manière intelligente.

Cependant, bien que l'exploration de l'espace des fonctions soit optimisée, l'évaluation de la pertinence des fonctions créées reste la partie la plus coûteuse pour le fonctionnement du système, car elle nécessite le calcul de nombreuses opérations sur de grandes bases de signaux.

Ainsi, afin d'améliorer l'efficacité de la recherche d'un point de vue computationnel, nous avons optimisé le temps de calcul effectif des opérations, et avons introduit deux techniques supplémentaires permettant de réduire le nombre de calculs opérationnels réalisés par le système : la réécriture des fonctions, et la mise en cache des résultats des calculs intermédiaires.

Calcul des opérations

Le calcul de chaque opération à partir de données EDS peut être réalisé de manière :

- externe : par enregistrement des données d'entrée dans des fichiers, exécution d'un programme externe qui transforme les données d'entrée et écrit le résultat dans un fichier de sortie, et lecture du fichier de sortie par le système. Cette méthode est très modulable, car les exécutables externes peuvent être modifiés facilement ; cependant, du fait des appels aux fichiers, elle est assez coûteuse en temps de calcul.
- interne : calcul direct par des opérateurs programmés dans le langage d'EDS (Java) ; à l'inverse, les temps de calcul sont diminués, mais les

opérateurs ne sont modifiables qu'en modifiant le code du système.
L'internalisation des fonctions les plus courantes dans EDS a permis des gains importants en temps de calcul, comme le montre la table 4.3.

OPERATEUR	Temps de calcul EXTERNE (ms)	Temps de calcul INTERNE (ms)	GAIN (%)
Encodage d'un fichier .wav	851	420	51
Corrélation	6369	1960	69
Autocorrélation	4566	1480	68
Enveloppe	3234	2220	31
Racine carrée	1182	360	70
Fréquence centrale	1041	650	38
Filtrage	902	330	63
Taux de passage à zéro	801	90	89
Fenêtrage	700	320	54
Dérivation	591	320	46
Variance	511	90	82
Décodage d'un fichier .mat	4316	inutile	

Tab. 4.3: Comparaison des temps de calcul d'opérateurs externes et internes, sur un signal numérique de 10 secondes échantillonné à 44100Hz

Par exemple, le calcul de la fonction « Autocorrélation (Signal) » sur un fichier .wav de 10 secondes échantillonné à 44100Hz nécessite :

- en externe : l'encodage d'un fichier .wav (850ms), le calcul de l'opération (4500ms), et le décodage du fichier résultat final (4300ms), soit environ 10 secondes pour un signal, et 2000s pour une base de 200 signaux.
- en interne : l'encodage d'un fichier .wav (420ms), et le calcul de l'opération (1500ms), soit environ 2 secondes pour un signal, et 400s pour une base de 200 signaux.

Au prix de travaux d'optimisation plus poussés, nous pourrions gagner encore quelques ms sur le calcul de certaines opérations. Cependant il est plus intéressant de réaliser de nouveaux gains plus importants en contrôlant le nombre d'opérations effectivement calculées, à partir de techniques de réécriture et de cache de données.

Réécriture des fonctions

La réécriture des fonctions est une technique permettant de simplifier les fonctions avant leur évaluation, diminuant ainsi le nombre d'opérations à effectuer par le système.

Ainsi, chaque fonction créée par EDS est simplifiée suivant un ensemble de règles de réécriture sur les opérations qui la constituent. Contrairement aux heuristiques, ces règles ne sont pas utilisées pour favoriser certaines compositions d'opérateurs, mais permettent :

- d'éviter de calculer plusieurs fois la même fonction écrite sous différentes formes :
  Par exemple, la fonction « Max (Corrélation (Signal, Signal)) »est réécrite

en « Max (Autocorrélation (Signal)) », grâce à la règle de réécriture suivante :
. « $Corrélation(x, x) \Rightarrow Autocorrélation(x)$ »

– de calculer directement les valeurs numériques constantes :
Par exemple, la fonction « Moyenne (Carré (Puissance (Signal, 2))) »est réécrite en « Moyenne (Puissance (Signal, 4)) », grâce aux règles de réécriture suivantes :
. « $Carré(x) \Rightarrow Puissance(x, 2)$ »
(on obtient « Moyenne (Puissance (Puissance (Signal, 2), 2)) »)
. « $Puissance(Puissance(x), N), M)$
$\Rightarrow \begin{cases} Puissance(x, N * M), si(N\ entier\ impair) et(N * M\ entier) \\ Puissance(Valeur\ absolue(x), N * M), sinon \end{cases}$ »
(on obtient « Moyenne (Puissance (Abs (Signal), 4)) »)
. « $Puissance(Valeur\ absolue(x), N)$
$\Rightarrow Puissance(x, N), si(N\ entier\ pair)$ »
(on obtient « Moyenne (Puissance (Signal), 4) »)

– d'éliminer des opérations inutiles :
Par exemple, la fonction « Variance (Carré (Valeur absolue (Signal))) »est réécrite en « Variance (Puissance (Signal, 2)) », grâce aux règles de réécriture suivantes :
. « $Carré(x) \Rightarrow Puissance(x, 2)$ »
(on obtient « Variance (Puissance (Valeur absolue (Signal), 2)) »)
. « $Puissance(Valeur\ absolue(x), N)$
$\Rightarrow Puissance(x, N), si(N\ entier\ pair)$ »
(on obtient « Variance (Puissance (Signal, 2)) »)

– d'éliminer des opérations antagonistes :
Par exemple, la fonction « Variance (Racine carrée (Puissance (Signal, 2))) »est réécrite en « Variance (Abs (Signal)) », grâce aux règles de réécriture suivantes :
. « $Racinecarrée(x) \Rightarrow Puissance(x, 1/2)$ »
(on obtient « Variance (Puissance (Puissance (Signal, 2), 1/2)) »)
. « $Puissance(Puissance(x), N), M)$
$\Rightarrow \begin{cases} Puissance(x, N * M), si(N\ entier\ impair) et(N * M\ entier) \\ Puissance(Valeur\ absolue(x), N * M), sinon \end{cases}$ »
(on obtient « Variance (Puissance (Valeur absolue (Signal), 1)) »)
. « $Puissance(x, 1) \Rightarrow x$ »
(on obtient « Variance (Valeur absolue (Signal)) »)

– de corriger des fonctions non-calculables :
Par exemple, la fonction « Variance (Puissance (Racine carrée (Signal), 2)) »est incalculable car l'opération Racine carrée n'est pas applicable sur les valeurs négatives de Signal ; cependant, elle est réécrite en « Variance (Valeur absolue (Signal)) », grâce aux règles de réécriture suivantes :
. « $Racine\ carrée(x) \Rightarrow Puissance(x, 1/2)$ »
(on obtient « Variance (Puissance (Puissance (Signal, 1/2), 2)) »)
. « $Puissance(Puissance(x), N), M)$

$\Rightarrow \begin{cases} Puissance(x, N*M), si(N\ entier\ impair) et(N*M\ entier) \\ Puissance(Valeur\ absolue(x), N*M), sinon \end{cases}$ »
(on obtient « Variance (Puissance (Valeur absolue (Signal), 1)) »)
. « $Puissance(x, 1) \Rightarrow x$ »
(on obtient « Variance (Valeur absolue (Signal)) »)

– de combiner des opérations complémentaires :
Par exemple, la fonction « RMS (Filtre passe-haut (Filtre passe-bas (Signal, 2000Hz), 1000Hz)) »est réécrite en « RMS(Filtre passe-bande (Signal, 1000Hz, 2000Hz)) », grâce à la règle de réécriture suivante :
. « $Filtre\ passe-haut(Filtre\ passe-bas(x, f_{max}), f_{min})$
$\Rightarrow Filtre\ passe-bande(x, min(f_{min}, f_{max}), max(f_{min}, f_{max}))$ »
Note : Cette règle permet également de corriger une expression incorrecte dans laquelle $f_{min} > f_{max}$.

– de réduire le temps de calcul, en utilisant des opérations moins coûteuses :
Par exemple, la fonction « Max (Autocorrélation (Filtre passe-haut (Signal, 2000Hz))) »est réécrite en « Somme (Carré (Filtre passe-haut (Signal, 2000Hz))) », grâce à la règle de réécriture suivante, traduisant la formule de l'énergie totale d'un signal quelconque :
. « $Max(Autocorrélation(x)) \Rightarrow Somme(Carré(x)$ »
Cette règle permet de remplacer le calcul de corrélation, coûteux, par un simple calcul de somme.
Un autre exemple, la règle de réécriture suivante :
. « $Moyenne(Spectre(Carré(Signal))) \Rightarrow Somme(Carré(Signal))$ »
traduit l'égalité de Perseval entre l'énergie d'un signal échantillonné quelconque et de son spectre, et permet d'éviter une opération de calcul de Spectre.

– d'éliminer les opérations linéaires :
Par exemple, la fonction « Multiplication (Variance (Signal), 100) »est réécrite en « Variance (Signal) », grâce à la règle de réécriture suivante :
. « $Multiplication(x, A), en\ fin\ de\ fonction \Rightarrow \begin{cases} 0, si(A = 0) \\ x, sinon \end{cases}$ »
Cette règle permet de simplifier toutes les fonctions linéairement équivalentes, afin de s'assurer de leur indépendance.

La simplification d'une fonction est réalisée par applications successives de l'ensemble des règles incluses dans EDS. La terminaison de la simplification est assurée par l'utilisation de règles de réduction stricte du nombre d'opérateurs (du type « $A(B \rightarrow C$ ») qui se terminent par définition, de permutations d'opérations non-réversibles (pas de règles concurrentes « $AB \rightarrow BA$ »et « $BA \rightarrow AB$ »), et de remplacements d'opérations non-circulaires (pas de règles bouclantes « $A \rightarrow B$ », « $B \rightarrow C$ », et « $C \rightarrow A$ »).

Les règles utilisées montrent généralement la confluence de la simplification vers des fonctions fixes, bien que celle-ci n'ait pas été démontrée rigoureusement, l'objectif principal de la simplification étant d'obtenir des fonctions plus simples et plus rapidement calculables.

Mécanisme de cache

Du fait des transformations appliquées aux fonctions, la recherche génétique doit évaluer de nombreuses fonctions différentes, mais contenant souvent des groupes d'opérations similaires.

Afin de réduire les temps de recherche en évitant de calculer plusieurs fois les mêmes opérations, nous avons introduit un mécanisme de cache des calculs intermédiaires des fonctions : ainsi, tout ensemble d'opérations coûteux en temps de calcul, est calculé une fois sur chaque signal de la base, et les résultats sont conservés en mémoire et réutilisés pour accélérer le calcul de nouvelles fonctions.

Le principe du système de cache est que, chaque fois qu'une nouvelle fonction est calculée, tous les résultats intermédiaires sont enregistrés. Par exemple : le calcul de « Max (Envelope (Spectre (Signal)) »enregistrera les résultats du décodage de « Signal », et des calculs de « Spectre (Signal) », « Envelope (Spectre (Signal)) », et « Max (Envelope (Spectre (Signal)) », pour chaque signal de la base de calcul.

Ces résultats intermédiaires sont ensuite réutilisés lors du calcul de nouvelles fonctions contenant des compositions d'opérations identiques. Par exemple, suite au calcul de la fonction précédente « Max (Envelope (Spectre (Signal)) », le calcul d'une nouvelle fonction « Moyenne (Max (Fenêtrage (Spectre (Signal)))) »réutilisera sans le recalculer le résultat de « Spectre (Signal) », évitant ainsi le décodage du signal initial et le calcul de son spectre.

Cependant, l'espace de stockage des résultats étant limité, on ne peut pas conserver l'ensemble des résultats intermédiaires de l'ensemble des fonctions calculées lors de la recherche. Il est donc nécessaire de ne conserver que les résultats les plus utiles. Nous avons ainsi défini l'utilité d'un résultat, en fonction de :

- leur fréquence d'utilisation : les résultats utilisés fréquemment sont conservés en priorité (par exemple le résultat du décodage du signal d'entrée, utilisé dans chaque fonction, est toujours conservé) ;
- leur temps de calcul : les résultats nécessitant un long temps de calcul sont conservés en priorité ;
- leur taille : à fréquence et temps de calcul égaux, priorité est donnée aux résultats contenant le plus faible volume de données.

4.9 Construction et optimisation des modèles descriptifs

Nous avons présenté dans la section précédente la partie principale d'EDS, consistant à construire un ensemble de fonctions caractéristiques pertinentes pour modéliser un descripteur précis à partir du signal. Ces fonctions sont ensuite utilisées par EDS pour réaliser l'apprentissage du modèle optimal fournissant la représentation finale du descripteur.

4.9.1 Dimension des fonctions caractéristiques

La recherche génétique permet de générer des fonctions caractéristiques unaires, c'est-à-dire qui renvoient une donnée unique décrivant le signal dans

sa globalité. Cependant, cette donnée unique peut aussi bien être une valeur simple qu'un vecteur de valeurs multi-dimensionnel.

Dans ce cas, afin de conserver la généralité du système, nous considérerons une caractéristique multi-dimensionnelle comme un ensemble de caractéristiques uni-dimensionnelles, plus ou moins dépendantes. Par exemple, une fonction caractéristique extrayant les dix premiers coefficients cepstraux d'un signal, est représentée comme un ensemble de dix fonctions $\{f_i\}$ extrayant chacune le i^{eme} coefficient cepstral.

Ainsi, nous disposons pour notre modélisation d'un ensemble de fonctions caractéristiques unaires uni-dimensionnelles, qu'il nous faut combiner de manière optimale.

4.9.2 Sélection de fonctions caractéristiques

La recherche automatique dans EDS fournit un ensemble de fonctions caractéristiques pertinentes. Cependant, une sélection de fonctions préalable à la modélisation présente divers intérêts.

D'une part, elle assure la constitution d'un ensemble de fonctions indépendantes, complémentaires, et pertinentes. D'autre part, en limitant le nombre de paramètres (les fonctions caractéristiques), elle permet :

- d'assurer la généralité du modèle, en évitant un ajustement trop précis aux données d'apprentissage ;
- de limiter le nombre d'opérations à réaliser pour les calculer, et de diminuer ainsi leur temps de calcul. En effet, les descripteurs modélisés par EDS étant destinés à produire des exécutables utilisables rapidement dans des applications informatiques, il est indispensable d'obtenir les modèles les plus simples possibles ;
- de favoriser la compréhension des modèles obtenus.

De nombreuses méthodes de sélection existent dans la littérature générale (voir [Guyon et Elisseeff, 2003], [Leray et Gallinari, 1999]). Dans le domaine spécifique de la description de signaux musicaux, la sélection de variable a généralement eu pour objectif de nettoyer les grands ensembles de fonctions classiques (du type MPEG7 et leurs dérivées) suivant le problème considéré. Les techniques utilisées, présentées dans [Peeters et Rodet, 2002], vont de l'Analyse en Composante Principale ([Kaminskyj et Materka, 1995]) à l'Analyse Discriminante ([Martin et Kim, 1998]), en passant par les algorithmes génétiques ([Fujinaga, 1998]), l'information mutuelle ([Ellis et Bilmes, 2000]).

Nous avons introduit dans EDS les principales méthodes de la littérature (sélection exhaustive, ascendante, génétique, ...), appliquées préalablement à la modélisation, en utilisant la librairie « Weka »([Witten *et al.*, 2000]). Par défaut, en recherche automatique, une sélection ascendante est appliquée sur les fonctions avant de lancer l'apprentissage du modèle.

4.9.3 Synthèse d'un modèle à partir des fonctions caractéristiques

L'ensemble des fonctions de la sélection obtenue est combiné dans un modèle final du descripteur. De nombreux mécanismes d'apprentissage permettent d'agréger un ensemble de fonctions, de la simple combinaison linéaire aux réseaux de neurones, dont on peut obtenir une vision globale dans de nombreux ouvrages

(notamment [Duda et Hart, 1973], [Fukunaga, 1990], [Theodoridis et Koutroumbas, 1999], ou [Bishop, 1995] plus détaillé sur les réseaux de neurones).

Nous avons introduit dans EDS les principaux mécanismes d'apprentissages, en utilisant la librairie « Weka »([Witten *et al.*, 2000]). Weka est un système Open Source, programmé en Java, permettant d'apprendre et de tester les principales méthodes de modélisation sur un ensemble de données. EDS étant également programmé en Java, nous avons ainsi pu inclure facilement les classes de Weka dans la partie modélisation du système.

Nous proposons ainsi pour chaque descripteur à modéliser, différentes méthodes d'apprentissage dépendant du type de problème à résoudre :

- Méthodes générales : k-Plus-Proches-Voisins (« kNN »), Réseaux de Neurones (« Neural Nets ») ;
- Méthodes spécifiques aux régressions : Régression Linéaire, Arbres de Modèles Linéaires, Régression Localement Pondérée ;
- Méthodes spécifiques aux classifications : Arbres de Décision, Bayes Naïf, 1-R de Holte, « Kernel Density Classifier », Modèles de mélange gaussiens (« GMM ») ;
- Méthodes spécifiques aux classifications booléennes : « Support Vector Machine »([Burges, 1998]), Régression logistique (sachant que tout problème multiclasse peut être réduit à un ensemble de problèmes booléens, [Allwein *et al.*, 2000]).

Une présentation des méthodes principales est faite en annexe C.

4.9.4 Optimisation des modèles

En fonction de la méthode utilisée et de ses paramètres (comme par exemple le nombre de voisins k pour la méthode des Plus-Proches Voisins), le modèle peut être plus ou moins pertinent. EDS réalise donc une optimisation du modèle, en cherchant la méthode la plus pertinente pour l'extraction du descripteur, et en optimisant ses paramètres.

Pour cela, le système actuel réalise une recherche complète, en testant l'ensemble des méthodes (par validation croisée sur la base d'apprentissage). A nouveau, différentes mesures de pertinence sont utilisables :

- en régression : Coefficient de corrélation, erreur moyenne, relative, ...
- en classification : Pourcentage de classifications correctes, coefficient de Kappa, erreur moyenne, relative, ...

Finalement, EDS réalise une évaluation du meilleur modèle trouvé sur une base de test indépendante de la base d'apprentissage, et peut alors synthétiser un extracteur final exécutable de ce modèle brut.

Cependant, il est également possible de dériver plusieurs descripteurs variés à partir de ce même modèle, en utilisant diverses « agrégations temporelles », comme cela est présenté dans la section suivante.

4.10 Agrégation temporelle des modèles - Synthèse d'un extracteur final

A partir du modèle obtenu précédemment, EDS est capable de synthétiser un descripteur « brut »appliquant directement le modèle, mais également toute

une variété d'autres descripteurs, obtenus à partir de l'évolution de ce modèle, par « agrégation temporelle ».

4.10.1 Validité du descripteur « brut »

En créant des modèles de descripteurs à partir d'une base de signaux d'apprentissage, EDS en limite le domaine d'application pratique. En effet, bien que les modèles créés ainsi soient théoriquement applicables sur tout nouveau signal, ils ne sont cependant validés et applicables de manière robuste que sur des signaux homogènes à ceux de la base d'apprentissage, c'est-à-dire essentiellement de durée équivalente.

Par exemple, dans le cas des signaux musicaux, supposons qu'on dispose d'un modèle décrivant la « percussivité »d'un son, qui évalue si un son polyphonique a un timbre proche de celui d'un instrument percussif.

Si on applique ce modèle sur un solo de batterie d'une minute, il est peu probable qu'on obtienne effectivement une évaluation de la « percussivité »du solo de batterie.

En fait, ces deux notions de « percussivité »correspondent à deux descripteurs différents suivant la durée des signaux étudiés : pour un son d'une durée de l'ordre de la seconde, on veut savoir si le son provient principalement d'un instrument percussif, tandis que pour un passage musical d'une durée de l'ordre de la minute, on veut par exemple évaluer la prédominance des instruments percussifs sur les autres.

Or le modèle basique de percussivité synthétisé par EDS correspond uniquement au premier descripteur, et ne peut pas être appliqué à une échelle temporelle différente.

Ainsi, si on désire générer un nouveau descripteur de percussivité d'un passage musical, deux solutions s'offrent à nous :

- soit la notion de « percussivité »d'un passage musical est indépendante de celle d'un son ; dans ce cas, il est nécessaire de réaliser des tests perceptifs sur un ensemble d'auditeurs afin d'évaluer et de valider ce concept sur des passages musicaux variés, puis de construire une base d'apprentissage associée, et de synthétiser un nouveau modèle du descripteur, indépendant du modèle du descripteur de « percussivité sonore » ;
- soit on est capable de trouver une relation entre les deux concepts de percussivité, du type « la percussivité d'un passage musical est la proportion de sons percussifs dans le passage » ; dans ce cas, il est possible de dériver le descripteur de percussivité à partir du modèle de percussivité sonore, par une « agrégation temporelle ».

4.10.2 Agrégation temporelle de modèles locaux

L' »agrégation temporelle »permet de créer un descripteur « global »par généralisation d'un modèle « local »sur une durée plus importante, en utilisant son évolution dans un signal plus long que sa durée spécifique.

Le descripteur agrégé est alors défini par :

- un modèle de descripteur local ;
- un mode de parcours du signal global afin d'analyser l'ensemble de ses modèles locaux : c'est la « segmentation ». Typiquement, la segmentation

utilisée est la même que celle qui a servi à créer la base d'apprentissage du descripteur local ;
- un formule de traitement de l'ensemble des résultats locaux : c'est l' « agrégation ».

Dans notre exemple, nous avons défini le concept de percussivité d'un passage musical par la proportion de sons percussifs dans le passage (signal global), c'est-à-dire la proportion de sons (signaux locaux) dans le passage que le modèle de « percussivité sonore »décrit comme percussifs. Le descripteur de « percussivité globale »peut alors être défini ainsi :
- modèle local : « percussivité locale »(classe 0 : non-percussif, ou 1 : percussif) ;
- segmentation : signaux de 100ms sans chevauchement (par exemple) ;
- agrégation : moyenne des résultats sur l'ensemble des segments (valeur entre 0 et 1).

4.10.3 Segmentations et Agrégations

Dans sa version actuelle, pour représenter les descripteurs agrégés, EDS intègre une segmentation basique (en segments de durée constante), et les agrégations temporelles les plus communes (moyenne, variance, classe majoritaire).

De plus, une « segmentation »spéciale prend en compte l'ensemble du signal global comme un seul segment. En utilisant cette segmentation, on peut alors définir tout descripteur « local »simple comme l'agrégation temporelle de son propre modèle, retournant l'unique valeur obtenue. Cette propriété est utilisée lors de la synthèse de l'exécutable final (voir section 4.10.4).

Enfin, une « agrégation »spéciale retourne la série des valeurs locales obtenues sans réaliser de calcul d'agrégation (comme la moyenne). Cette agrégation est utilisée pour des applications spécifiques comme le suivi (« tracking ») de descripteurs le long d'un signal. Elle permet également de réaliser des post-traitements externes à EDS, comme par exemple le filtrage de la série de valeurs, ou la détection de motifs.

La segmentation et l'agrégation n'étant pas utilisées dans EDS, mais uniquement lors de l'exécution de l'extracteur final, il est possible d'ajouter de nouvelles méthodes sous forme de modules externes, utilisant les paramètres d'agrégation temporelle écrits automatiquement par EDS lors de la synthèse de l'exécutable final (voir section 4.10.4).

4.10.4 Enregistrement d'un exécutable pour un descripteur agrégé

A partir d'un descripteur final agrégé, EDS enregistre automatiquement un programme exécutable permettant d'évaluer ce descripteur sur tout signal externe.

L'exécutable est généré en fonction des paramètres d'agrégation temporelle du descripteur : modèle local, segmentation, et agrégation. La segmentation « globale », prenant l'ensemble du signal comme un segment, permet la génération du descripteur local associé au modèle brut.

Les exécutables sont enregistrés automatiquement sous la forme d'un groupe de fichier contenant :

- un fichier global de calcul du descripteur (.bat), qui contient les paramètres (fonctions caractéristiques, modèle utilisé, paramètres de l'agrégation temporelle (segmentation + agrégation)), et qui appelle les classes réalisant le calcul effectif du descripteur agrégé ;
- un fichier contenant le modèle local ;
- un fichier .jar contenant les classes de calcul des opérateurs EDS ;
- un fichier .jar contenant les classes de calcul des modèles Weka ;
- un fichier .jar contenant les classes de calcul du descripteur agrégé.

Nécessitant uniquement une installation de Java, cet ensemble de fichiers exécutable permet donc de réaliser le calcul du descripteur sur tout nouveau signal, indépendamment d'EDS. Cet exécutable est donc utilisé pour l'intégration des descripteurs EDS dans les applications externes.

4.11 Conclusion

Nous avons donc réalisé EDS, un système complet, automatique, de modélisation et d'extraction de descripteurs perceptifs à partir de signaux. Cette modélisation se fait en trois étapes principales :

- Recherche d'un ensemble de fonctions caractéristiques pertinentes ;
- Apprentissage du modèle à partir de ces fonctions caractéristiques ;
- Synthèse de l'extracteur final à partir du modèle optimal.

Ce système est général, et utilisable pour décrire tout type de donnée représentable par un signal. Le chapitre suivant présente les utilisations possibles du système, en se focalisant sur le domaine de la description musicale à partir du signal acoustique.

5. MODÉLISATION DE DESCRIPTEURS MUSICAUX AVEC EDS

EDS est un système général permettant de réaliser automatiquement la modélisation et l'extraction de descripteurs de signaux quelconques. Dans le cadre de nos applications, nous nous sommes intéressés plus précisément aux descripteurs musicaux construits à partir de signaux acoustiques.

Pour résoudre ces problèmes de description musicale, il est tout d'abord nécessaire de les transcrire dans un format compréhensible par EDS, c'est-à-dire générer une base de signaux étiquetés (voir section 4.2.2) ; la génération d'un ensemble de signaux homogènes et d'un étiquetage pertinent est l'objet de la première partie de ce chapitre.

Nous définissons alors une méthode générale, un paramétrage du système et des fonctions de référence, permettant de réaliser l'étude des performances d'EDS sur ces descripteurs.

Enfin, nous présenterons les résultats obtenus sur des problèmes de description musicale classiques, pour lesquels nous comparerons les performances des fonctions générées par EDS à celles des descripteurs de l'état de l'art synthétisés manuellement, et sur des problèmes musicaux originaux, pour lesquels nous comparerons les performances des fonctions générées par EDS à celles des fonctions générales de type Mpeg7.

5.1 Définition de problèmes de description musicale pour EDS

Nous abordons ici uniquement les problèmes de description de sons et d'extraits musicaux. Nous n'avons pas cherché à modéliser directement des descripteurs globaux pour des titres entiers, bien que cela soit théoriquement réalisable avec EDS, pour deux raisons principales.

La première raison est le coût du calcul. En effet, le principe de recherche génétique induit de nombreux calculs sur les signaux, qui sont extrêmement lourds dans le cas de signaux des titres entiers, d'une longueur de plusieurs centaines de milliers d'échantillons.

La seconde raison est qu'en général, les propriétés d'un morceau de musique évoluent avec le temps : variations de tempo, d'instrumentation, de style, etc. ... Plutôt que d'assigner à ces propriétés une valeur globale pour l'ensemble du morceau, il est donc beaucoup plus intéressant de suivre leur évolution à moyen terme, c'est-à-dire sur des segments plus courts ; la valeur globale pouvant être

obtenue par une agrégation temporelle de ces valeurs (typiquement la moyenne).

Ces problèmes de description de sons et d'extraits musicaux doivent alors être soumis à EDS, sous forme d'une base de signaux étiquetés, comme on l'a présenté en section 4.2.2.

La définition d'un problème de description musicale pour EDS repose donc sur le choix d'un ensemble de signaux représentatifs et leur étiquetage en fonction de critères perceptifs universels. La réalisation de l'étiquetage permet de vérifier la validité du problème a posteriori. Il est également possible d'évaluer la pertinence d'un descripteur musical a priori lors de tests utilisateurs, présentés ci-après.

5.1.1 Evaluation a priori de la pertinence de descripteurs musicaux

L'intégration d'un descripteur dans une application musicale générale n'est possible que si ce dernier définit un critère acoustique universel, ce qui garantit sa pertinence pour l'exploration musicale par l'ensemble des utilisateurs. Cette universalité est d'autant plus importante pour la description de sons simples, dans la mesure où il n'existe pas de donnée culturelle permettant de les classer.

Il est possible d'évaluer la pertinence des descripteurs *a posteriori*, à partir des résultats des tests perceptifs réalisés pour l'étiquetage (voir section 5.1.3). Nous présentons ici une méthode *a priori* réalisant une étude plus ciblée des habitudes des utilisateurs, sous forme de jeux perceptifs.

Jeux de discrimination d'extraits musicaux

Cette méthode consiste à imaginer un ensemble de descripteurs musicaux, et à tester leur utilité *a priori*. Pour réaliser cette évaluation, nous avons créé des tests, présentés sous forme de deux jeux complémentaires accessibles sur internet : « Discrimination Game »et « Recognition Game »(dont les interfaces sont présentées sur la figure 5.1).

Discrimination Game

Recognition Game

Fig. 5.1: Interfaces des jeux descriptifs utilisés pour évaluer la pertinence de descripteurs acoustiques sur des musiques populaires

Ce sont deux jeux complémentaires, qui permettent d'une part de déterminer quels sont les descripteurs musicaux les plus utilisés (Discrimination Game), et d'autre part de vérifier si la perception de ces descripteurs est bien universelle (Recognition Game).

Dans le « Discrimination Game », on fait écouter à l'utilisateur deux extraits musicaux, et celui-ci choisit un descripteur et une valeur, permettant de discriminer sans ambiguïté le premier morceau du second. Par exemple, si le premier extrait est un solo de guitare rapide, et le second un chant a capela lent, l'utilisateur peut choisir un descripteur :

- « vocal » : en indiquant que le premier extrait est « Non-Vocal », ou bien
- « rythmique » : en indiquant que le premier extrait a un « Tempo rapide ».

Les résultats sont enregistrés, le but du jeu étant de déterminer si le descripteur le plus utilisé est le descripteur de voix, ou celui de tempo.

Le « Recognition Game »est le jeu complémentaire au Discrimination Game, permettant de valider les évaluations réalisées par les utilisateurs dans ce dernier ; ainsi, on fait écouter à l'utilisateur deux extraits musicaux et on lui présente une description provenant du Discrimination Game, du type « l'extrait au tempo élevé ». L'utilisateur choisit alors celui des deux extraits qui, selon lui, correspond à la description. Si pour un descripteur donné, l'utilisateur trouve systématiquement l'extrait correspondant à la description, cela prouve que le descripteur choisi est pertinent.

Nous avons testé ces jeux en utilisant 200 extraits de musique populaire variés. Dans le Discrimination Game, les utilisateurs avaient le choix entre quatre descripteurs : « Tempo »(Lent, Moyen, Rapide), « Type de voix »(Homme, Femme, Aucune, Grave, Aiguë), « Genre »(Rock, Jazz, Disco, Pop, Classique), et « Énergie Musicale »(Basse, Moyenne, Élevée).

Les résultats des premiers tests simples, portant sur une centaine de couples de chansons, sont présentés dans le tableau 5.1.

DESCRIPTEUR	Genre	Tempo	Voix	Énergie
% Utilisation	19	15	37	29
% Erreur	0	8	18	10

Tab. 5.1: Comparaison des taux d'utilisation dans le Discrimination Game, et d'erreur dans le Recognition Game, pour 4 descripteurs usuels

Ces premiers tests, bien que rapides, ont cependant pu montrer des tendances marquées :

- le descripteur original d'énergie globale est assez utilisé, avec une erreur limitée, ce qui en fait un descripteur pertinent ;
- malgré les nombreux travaux qui ont porté sur son extraction, le tempo est le moins utilisé des descripteurs ;
- le genre est utilisé avec précaution, c'est-à-dire uniquement dans les cas évidents, comme par exemple pour distinguer les morceaux classiques, et jamais pour distinguer des morceaux de pop et rock ;
- la voix est le plus utilisé des descripteurs, principalement pour caractériser les passages instrumentaux (absence de voix), et distinguer entre les voix féminines et masculines, ce qui est source de plusieurs erreurs en reconnaissance.

Une nouvelle série de tests avec une palette de descripteurs plus large permettra de mener une étude de pertinence plus complète.

5.1.2 Constitution d'une base de signaux

La résolution d'un problème perceptif consiste à modéliser une *propriété* dans un *contexte* donné, par exemple « l'énergie musicale des chansons de ma base de musique personnelle », ou « les passages vocaux dans une chanson donnée ».

La base de signaux soumise à EDS doit donc être représentative de ceux deux aspects du problème.

Contexte d'utilisation

D'une part, le contexte, représentatif du domaine d'application du descripteur, est défini par l'ensemble des signaux de la base. Il peut être très spécifique (une chanson donnée) ou plus général (l'ensemble des chansons d'une base de musique personnelle). Afin d'assurer la robustesse du descripteur, il est nécessaire de spécifier son contexte de manière précise : origine des sons et extraits sonores, méthode d'obtention (typiquement une méthode de segmentation précise), domaine musical (par exemple la variété internationale). Pour être représentative du contexte, la base doit donc contenir un ensemble varié de signaux provenant de celui-ci.

Propriété à évaluer

D'autre part, la propriété à modéliser peut être définie de deux façons. Pour les problèmes de classification booléenne, on cherche à distinguer les signaux possédant cette propriété de ceux ne la possédant pas (par exemple des sons percussifs ou pas) ; la base doit donc contenir ces deux types de signaux, typiquement en proportions égales (50% de sons percussif, et 50% non-percussifs). Pour les problèmes de régression, on évalue la propriété pour tous les signaux (par exemple leur fréquence fondamentale), pour en faire un modèle général ; la base doit donc contenir un ensemble de signaux représentatif de l'ensemble des valeurs possibles pour la propriété considérée (par exemple des signaux dont les fréquences fondamentales couvrent toutes les fréquences de 0 à 5000Hz).

Sélection des signaux

Les signaux utilisés dans les bases musicales proviennent d'extraits musicaux variés extraits du contexte, comme par exemple des bandes sonores de films pour des descripteurs pour le cinéma (dialogues, ambiance, etc...), ou une base personnelle de musique pour la description de sons polyphoniques dans des applications comme le Musaicing (voir section 6.4.4).

A partir de ce matériau musical spécifique, on réalise une première extraction de signaux par une segmentation automatique. De nombreuses méthodes existent pour réaliser cette segmentation, et le choix d'une méthode dépend de l'application visée : par exemple extraction de fenêtres de longueur constante pour le suivi de bande sonore de film, segmentation plus élaborée par variations d'énergie ou de timbre pour les sons utilisés dans le Musaicing, etc...

On obtient alors un premier ensemble de signaux bruts. On réalise alors une présélection manuelle parmi ces signaux, qui permet d'éliminer les signaux qui ne sont pas pertinents, principalement les signaux de durée inadaptée ou qui ne sont pas musicalement homogènes par rapport au problème à résoudre. Par

exemple, pour l'évaluation de la percussivité, on élimine les signaux où les sons percussifs sont tronqués.

On obtient alors un ensemble de signaux globalement homogènes qui serviront à constituer la base. Il reste alors à étiqueter ces signaux avec les valeurs perceptives, ce qui est réalisé à l'aide de tests perceptifs, présentés dans le prochain paragraphe.

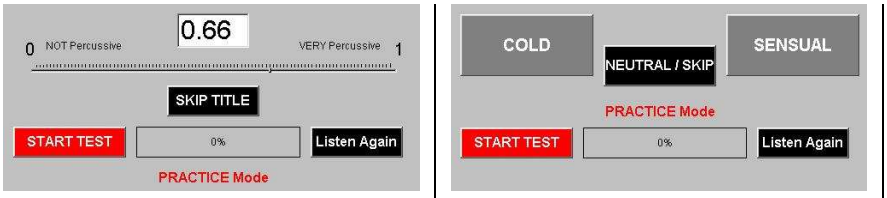
5.1.3 Étiquetage de la base de signaux

L'étiquetage de la base de signaux permet de valider la pertinence du problème descriptif considéré, et de s'assurer que l'ensemble des signaux de la base couvre tout le domaine de valeurs de la propriété étudiée. Pour réaliser cet étiquetage, nous utilisons les résultats de *tests perceptifs*, dans lesquels un groupe d'auditeurs évalue la propriété étudiée pour l'ensemble des signaux de la base, comme par exemple les tests d'évaluation de l'énergie musicale présentés section 3.2.1.

Mise en place de tests perceptifs pour la description musicale

Afin de pouvoir tester facilement une grande variété de descripteurs sonores et musicaux, nous avons mis en place une batterie de tests perceptifs, accessibles sur Internet, qui permettent d'évaluer les propriétés de bases de sons concrètes.

Deux types de test sont possibles, selon qu'on considère un problème de régression ou de classification. Dans tous les cas, on fait écouter à l'auditeur un son ou un extrait musical, puis on lui demande d'évaluer cet objet musical suivant un critère descriptif précis. Pour les régressions (par exemple « énergie globale d'un extrait musical »), l'auditeur évalue le descripteur entre 0 et 1 ; pour les classifications (par exemple « son percussif OU pas »), il/elle choisit une classe parmi les propositions (dans ce cas « Percussif »ou « Non-Percussif »). Les interfaces des applets utilisées sont présentées sur la figure 5.2.



Percussivité d'un son (Régression)

Sensualité d'un extrait musical (Classification)

Fig. 5.2: Applet utilisées pour les tests perceptifs des descripteurs de percussivité sonore (régressif) et de sensualité d'un extrait musical (classe)

On obtient ainsi grâce à ces applets l'ensemble des évaluations de chaque auditeur pour chaque descripteur. Il est alors nécessaire d'analyser cette grande quantité d'information brute, afin de réaliser et de valider l'étiquetage de bases de signaux.

Validation des résultats des tests perceptifs

L'analyse des résultats obtenus lors des tests perceptifs, fournit les informations nécessaires à l'étiquetage de la base de signaux. En effet, elle permet de :

- Vérifier la pertinence du problème descriptif étudié
  Les statistiques d'évaluation sur l'ensemble des utilisateurs permettent d'évaluer *a posteriori* deux notions fondamentales pour la pertinence du descripteur :
  * l'universalité du descripteur, c'est-à-dire la constance de ses évaluations pour des utilisateurs différents : mesurée par la variance des réponses sur l'ensemble des utilisateurs ; si cette variance est faible, le descripteur est considéré comme universel ; si elle est trop grande, la généralisation est impossible.
  * son utilité ou sa facilité d'utilisation : si la proportion d'évaluations avortées (« skip ») est élevée, on considère que le descripteur est difficile à évaluer.

  Ainsi, par exemple, nous avons pu constater d'après le nombre d'évaluation avortées, que le concept de « Complexité Musicale »était difficile à utiliser. Au contraire, la percussivité d'un son polyphonique semble très intéressante, dans la mesure où d'une part peu d'évaluations sont évitées, et d'autre part les réponses sont homogènes sur l'ensemble des utilisateurs/testeurs. Pour les descripteurs non-universels, il reste possible d'envisager une modélisation dépendant de l'utilisateur (« user dependent »), si les différentes évaluations provenant d'un même auditeur sont stables. C'est évidemment le cas de descripteurs personnels du type « Aimez-vous cet extrait ? », pour lequel les réponses dépendent des goûts des auditeurs, qui sont stables à moyen terme.
- Nettoyer à nouveau la base de signaux
  Après cette validation globale du problème, si les évaluations de l'ensemble des signaux sont homogènes, certains signaux peuvent cependant perturber localement les résultats des tests. Typiquement, on repère ces « mauvais »signaux de deux façons :
  * Évaluations avortées : le mode de fonctionnement des tests permet aux utilisateurs de ne pas répondre à une évaluation en cas de difficulté (bouton « Skip »sur l'interface des applets utilisées pour les tests, voir figure 5.2) ; si pour un signal donné, la proportion d'évaluations avortées est élevée, ce signal est considéré comme non-évaluable, et est éliminé de la base
  * Variance trop élevée : de même, un signal sera éliminé de la base si l'ensemble de ses évaluations présente une variance élevée entre les différents auditeurs
- Réaliser l'étiquetage
  Après le nettoyage final, on obtient l'ensemble définitif des signaux utilisables pour la modélisation. Il reste à attribuer à chacun de ces signaux une valeur unique et définitive : son étiquette perceptive. Cela consiste à agréger l'ensemble des résultats des tests perceptifs d'un descripteur pour chaque signal de la base, afin d'obtenir une valeur considérée comme sa meilleure évaluation (« Grounded truth »). Typiquement, les nettoyages successifs ayant assuré l'homogénéité des évaluations, la valeur finale uti-

lisée est la moyenne de l'ensemble des évaluations.

On dispose ainsi d'un ensemble de signaux étiquetés pouvant être utilisés pour modéliser le descripteur étudié. Il est possible de mesurer l'imprécision de l'étiquetage en calculant la variance globale des évaluations lors du test perceptif. Elle mesure l'incertitude « humaine »de la perception du descripteur. C'est donc une limite naturelle dont on doit tenir compte lors de la construction d'un modèle perceptif : le modèle ne peut pas être plus précis que la perception. Par exemple, dans le cas de l'énergie musicale, on obtient une imprécision de 10% sur les évaluations perceptives, il est donc inutile de chercher un modèle dépassant cette précision. Par contre, pour la détection de voix, l'imprécision est quasiment nulle (la classe de chaque son est reconnue sans ambiguïté par l'ensemble des auditeurs), il est donc nécessaire d'obtenir un modèle parfait. C'est un critère important pour évaluer les performances du système sur des descripteurs *perceptifs*, présentées dans les sections suivantes.

5.2 Élaboration d'une méthode générale d'étude des descripteurs musicaux

A partir de la base de signaux associée, la modélisation automatique d'un descripteur avec EDS est réalisée en deux étapes : la construction de fonctions caractéristiques pertinentes, puis la synthèse du modèle optimal.

Divers paramètres, utilisés pour la définition et la résolution du problème descriptif, peuvent influer sur la qualité du modèle final, par exemple :

- paramètres de la base : taille de la base, qualité de l'étiquetage ;
- paramètres de la recherche génétique de fonctions caractéristiques : taille de population, choix du critère d'arrêt, et choix d'un éventuel pattern de recherche ;
- paramètres de la synthèse du modèle : paramètres des classifieurs, nombre de fonctions caractéristiques prises en compte.

Les paramètres de la base sont fixés lors de sa construction (voir section 5.1.2). Les paramètres de synthèse du modèle sont également connus, car EDS utilise des techniques classiques d'apprentissage (voir en annexe C), et réalise une optimisation par recherche complète. Ainsi, les performances globales du modèle, et en conséquence de tout le système EDS, vont dépendre principalement de la qualité des fonctions caractéristiques trouvées lors de la recherche génétique.

Nous étudions ici l'influence des paramètres de la recherche génétique sur la qualité des fonctions caractéristiques trouvées, le réglage de ces paramètres nous permettant ensuite de déterminer une méthode générale pour l'extraction complète de descripteurs avec EDS.

5.2.1 Étude de l'influence des paramètres de recherche

Afin de pouvoir évaluer facilement les performances des fonctions caractéristiques trouvées, nous étudions l'influence des paramètres (taille des populations, critère d'arrêt, et pattern de recherche), sur les résultats de la recherche génétique d'une fonction caractéristique.

Notre test consiste à retrouver une fonction de traitement du signal constituée d'opérateurs inclus dans EDS : « F_{ref}(Signal) = Mean (Peaks (Autocorrelation (HpFilter (Signal, 1230Hz)))) »

EDS est théoriquement capable de retrouver exactement cette fonction, on peut ainsi étudier à la fois le temps de convergence du système pour obtenir chaque fonction, ainsi que la précision des fonctions obtenues. Pour mesurer la précision d'une fonction, nous calculons son coefficient de corrélation avec F_{ref}. A titre de comparaison, la meilleure fonction Mpeg7 modélisant F_{ref} est le taux de passage à zéro (ZCR), corrélée à 0,75.

Nous construisons tout d'abord une base d'apprentissage, contenant 100 signaux variés s_i de 0,1 seconde, échantillonnés à 11025Hz, étiquetées par les valeurs perceptives de la fonction F_{ref} : $F_{ref}(s_i)$. Ces signaux sont obtenus par segmentation constante de signaux de musiques populaires variées. Nous vérifions que l'ensemble des étiquettes couvre uniformément un large domaine de valeurs. 100 signaux est la taille moyenne de bases pour l'utilisation du système dans nos applications musicales, permettant d'obtenir rapidement un modèle de descripteur à partir d'un nombre d'exemples limité.

Paramètres de référence

Les paramètres de référence d'une recherche génétique sont les suivants :

- Taille de chaque population : 20 fonctions.
  Ainsi, pour chaque nouvelle population, les 5 meilleures fonctions seront retenues pour fabriquer environ 10 nouvelles fonctions par transformation génétique, et la population est complétée par des fonctions aléatoires (voir section 4.8.2).
- Critère d'arrêt : 5 populations consécutives sans amélioration
  La recherche s'arrête si la fonction optimale est trouvée, où si la meilleure fonction n'évolue plus pendant un nombre donné de générations, par défaut 5.
- Pattern de recherche : $!_x(Signal)$
  $!_x(signal)$ est le pattern le plus général pour modéliser une fonction en EDS : une suite quelconque d'opérations quelconques avec des paramètres quelconques fournissant une valeur de sortie unique à partir du signal d'entrée. En spécifiant un pattern aussi général, l'espace de recherche est extrêmement vaste, de l'ordre de 50^N fonctions, pour des opérations brutes sans paramètres ; la fonction F_{ref} contenant N=4 opérations, on obtient un espace de 6 millions de fonctions. Si en plus on tient compte des paramètres, comme les fréquences (entre 0 et 5000Hz), les tailles de fenêtres (entre 0 et 10000 échantillons), les puissances (entre -1 et 10), les constantes diverses (pourcentages, nombre de bandes, etc. . .entre 0 et 100), sachant que certaines fonctions utilisent deux de ces paramètres, on obtient plus de 20 millions d'opérations possibles, soit 20000000^N fonctions de N opérations. . .. Pour F_{ref}, N=4, donc l'espace est de l'ordre de 10^{29}, la recherche complète n'est donc pas envisageable.

Nous réalisons ici une série d'expériences en faisant varier chacun de ces paramètres de recherche. Afin d'obtenir des statistiques suffisamment pertinentes, chaque expérience consiste à lancer 100 fois la recherche génétique. On obtient donc de manière indépendante 100 fonctions pour chaque paramètre. La pertinence de chaque fonction obtenue est évaluée par calcul du coefficient de

corrélation sur une base de test indépendante. On évalue donc la qualité de chaque fonction indépendamment des autres, sans tenir compte des éventuels apports de combinaisons de fonctions.

On réalise une série d'expériences sur la fonction F_{ref}, en faisant varier à chaque fois un seul des paramètres (taille de chaque population, critère d'arrêt, pattern de recherche).

Les statistiques des résultats sont présentées sur la figure 5.3, où on peut observer le nombre de fonctions obtenues, leur pertinence sur une base de test indépendante, et le nombre de populations calculées avant convergence.

Elles permettent de valider certains comportements du système utiles pour proposer une recherche complète performante.

Résultats de référence

Les statistiques de recherche génétique avec les paramètres par défaut sont pris comme résultats de référence (graphe (1) de la figure 5.3) :

- 31% de fonctions de pertinence supérieure à 0.7, obtenues majoritairement après la 15^{eme}population, dont seulement 7% supérieure à 0.9 ;
- 39% de fonctions de pertinence entre 0.5 et 0.7, obtenues également entre 1 et 30 populations ;
- 30% de fonctions de pertinence inférieure 0.5, dont les deux-tiers sont obtenues avant la 15^{eme}population.

La première conclusion est qu'EDS ne retrouve pas la solution exacte, et peu de très bonnes fonctions. La meilleure fonction obtenue est : « Sqrt (Median (HFC (Derivation (Derivation (Hamming (Split (Autocorrelation (Testwav), 189.0))))))) », corrélée à 0.99 en apprentissage, et 0.98 en test. On y retrouve deux composantes de la fonction F_{ref} : l'autocorrélation et la prise en compte des hautes-fréquences (HFC : « High-Frequency Content »). « F_{ref}(Signal) = Mean (Peaks (Autocorrelation (HpFilter (Signal, 1230Hz)))) »

De plus, les bonnes fonctions obtenues le sont majoritairement après plus de 15 populations (plus de 30 pour les meilleures). Ce sont donc des *convergences naturelles* dans des minima locaux après une recherche approfondie.

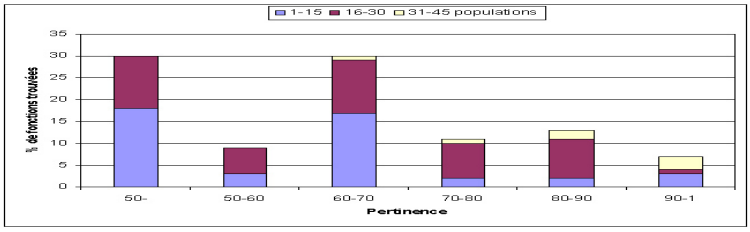
Les fonctions restantes, soit deux-tiers des résultats, sont peu pertinentes (corrélation inférieure à 0.70). La moitié d'entre elles sont obtenues avant la 15^{eme}population. Elles ont en général une pertinence moyenne en apprentissage, et faible en test. On les appelle des *convergences prématurées* dans des minima locaux. L'autre moitié résulte d'une recherche plus approfondie. Elles ont une pertinence assez bonne en apprentissage, mais faible en test. Ce sont des *convergences spécifiques* à la base d'apprentissage.

Influence de la taille de population

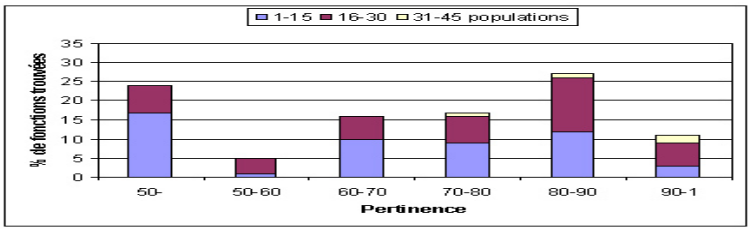
En réalisant l'expérience avec une population doublée à 40 fonctions, nous obtenons les statistiques suivantes (graphe (2) de la figure 5.3) :

- 55% de fonctions de pertinence supérieure à 0.7, dont la moitié entre 0.8 et 0.9, et 11% de très bonnes fonctions supérieures à 0.9 ;
- 45% de fonctions de pertinence inférieure à 0.7, dont 24% inférieures à 0.5, obtenues aux deux-tiers avant la 15^{eme}population.

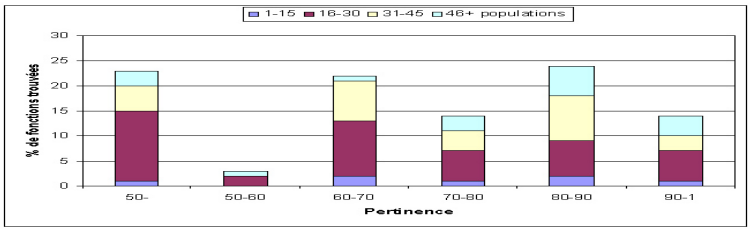
On observe donc une amélioration de l'ensemble des fonctions obtenues : diminution de moitié des fonctions entre 0.5 et 0.7, et doublement du nombre



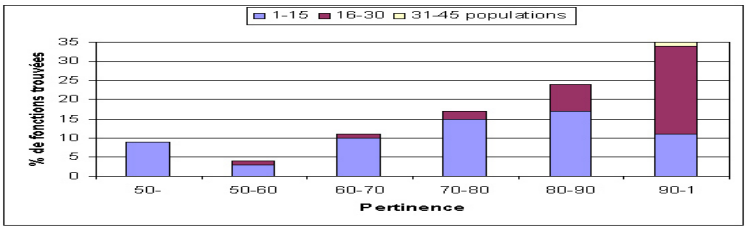
(1) Paramètres par défaut



(2) Taille de Population = 40



(3) Critère d'arrêt = 10



(4) Pattern = $!_a(*_{t:a}(Signal))$

Fig. 5.3: Répartition des pertinences sur une base de test, des fonctions obtenues sur 100 recherches génétiques de la fonction F_{ref}, en faisant varier les paramètres par défaut : Taille de population = 20, Critère d'arrêt = 5, Pattern = $!_x(Signal)$

de fonctions de plus de 0.8. De plus, une plus grande proportion de bonnes fonctions sont obtenues avant la 15^{eme}population.

Ainsi, l'augmentation de la taille de population semble globalement accélérer la convergence (augmentation générale du nombre de fonctions obtenues avant la 15^{eme}population). En effet, le nombre plus important de fonctions dans chaque population offre une variété qui permet d'arriver plus rapidement dans des mi-

nima locaux. Pour chaque nouvelle population de 40 fonctions, les 10 meilleures fonctions sont retenues pour fabriquer environ 20 nouvelles fonctions par transformation génétique, la population étant complétée par des fonctions aléatoires. Cependant, même si la population de convergence arrive plus rapidement, chaque génération contient deux fois plus de fonctions, et le nombre total de fonctions calculées augmente énormément, en passant à 600 en moyenne, contre 350 pour une recherche classique. Ainsi, le temps de recherche est quasiment doublé.

Enfin, bien qu'on observe une légère augmentation du nombre de très bonnes fonctions, on note que l'amélioration porte essentiellement sur les fonctions « assez »bonnes, de pertinence entre 0.7 et 0.9. Ainsi, l'augmentation de la taille de la population semble assurer une plus grande robustesse des fonctions trouvées, mais est insuffisante pour rechercher de manière approfondie les meilleures fonctions.

Influence du critère d'arrêt

En changeant maintenant le critère d'arrêt de 5 à 10 générations sans amélioration, nous obtenons les statistiques suivantes (graphe (3) de la figure 5.3) :

- 38% de bonnes fonctions de pertinence supérieure à 0.8, obtenues en majorité après la 30^{eme}population ;
- 36% de fonctions moyennes de pertinence entre 0.6 et 0.8, obtenues également entre 15 et 50 populations ;
- 26% de fonctions mauvaises de pertinence inférieure à 0.6, obtenues majoritairement entre 15 et 30 populations.

On note une amélioration du nombre de très bonnes fonctions au détriment des très mauvaises. De plus, le temps de convergence est globalement doublé, les convergences étant globalement plus rapides pour les mauvaises fonctions. Ainsi, l'augmentation du critère d'arrêt, en limitant les convergences prématurées dans des minima locaux, permet d'obtenir de meilleurs résultats.

Cependant, on remarque que les meilleures fonctions obtenues sont assez proches et corrélées les unes aux autres, comme par exemple « Sqrt (Iqr (Square (Autocorrelation (HpFilter (Testwav, 1009.0))))) »ou « Rms (Autocorrelation (HpFilter (Testwav, 890.0))) ». On peut donc reprocher un léger manque de variété des solutions obtenues.

Le réglage du critère d'arrêt est donc un équilibre subtil entre :

- l'exploration d'une grande variété de minima locaux, au risque de manquer la meilleure fonction si elle existe (valeur basse) ;
- l'optimisation des fonctions obtenues, permettant de se rapprocher de la meilleure fonction, mais au détriment d'une grande variété de minima locaux qui peuvent être intéressants (valeur élevée).

Influence du pattern

La spécification d'un pattern de recherche permet de limiter la recherche à des domaines a priori pertinents, comme les méthodes classiques de traitement du signal ou une idée de méthode que l'utilisateur désire explorer.

Par exemple, si nous spécifions que nous désirons construire une fonction réalisant une suite d'opérations dans le domaine temporel, puis une agrégation des valeurs (pattern $!_a(*_{t:a}(Signal))$), comme la fait la fonction F_{ref}, on obtient les statistiques suivantes (graphe (4) de la figure 5.3) :

- 59% de bonnes fonctions, de pertinence supérieure à 0.8, les meilleures étant obtenues majoritairement entre la 15^{eme}et la 30^{eme}population ;
- 28% de fonctions moyennes entre 0.6 et 0.8, obtenues majoritairement avant la 15^{eme}population ;
- 13% de mauvaises fonctions inférieures à 0.6, obtenues essentiellement avant la 15^{eme}population.

Les résultats sont donc très bons, la proportion de bonnes fonctions a triplé, et la proportion de mauvaises est divisée par 3. De plus, les convergences sont globalement plus rapides, les meilleures fonctions étant le résultat de recherches un peu plus approfondies (entre 15 et 30 générations), tandis que les mauvaises sont quasiment toutes obtenues par convergence prématurée.

Cependant, à nouveau, on constate que les meilleures fonctions sont très proches et corrélées entre elles, comme « Iqr (Derivation (Correlation (Testwav, BpFilter (Testwav, 4035.0, 1386.0)))) », « Iqr (LpFilter (Autocorrelation (BpFilter (Testwav, 1120.0, 4190.0)), 1716.0)) », ou « Sqrt (Min (Autocorrelation (Derivation (BpFilter (Derivation (Correlation (Normalize (Testwav), Testwav)), 692.0, 2512.0))))) ».

Ainsi, l'utilisation d'un pattern adéquat permet d'obtenir de très bonnes solutions très rapidement, mais limite cependant la variété des résultats obtenus. Mais le pattern doit être spécifié avec précaution, car s'il ne correspond pas aux meilleures solutions, les performances peuvent être très diminuées. La spécification d'un pattern nécessite donc une bonne connaissance du problème, et des intuitions pertinentes sur la manière de le résoudre.

5.2.2 Génération d'un ensemble de fonctions pertinent

Pour récapituler, les effets généraux des paramètres sont les suivants :

- Taille de population : une plus grande taille de population permet d'obtenir des fonctions globalement plus robustes, mais pas forcément les plus performantes, et allonge le temps de recherche ;
- Critère d'arrêt : il gère l'équilibre entre la variété des solutions et l'optimisation des meilleures fonctions ;
- Pattern de recherche : il diminue la variété, mais permet de trouver rapidement les meilleures fonctions s'il est bien spécifié, sinon permet d'explorer de nouvelles portions de l'espace sans assurer de résultats.

L'étude des effets des paramètres permet de déduire une méthode de construction d'un ensemble de fonctions pertinent pour la modélisation finale.

Pour cela, il est important de garder à l'esprit que la fonction optimale seule est rarement suffisante pour réaliser un modèle pertinent, et que c'est l'ajout d'une grande variété de fonctions approchées indépendantes qui permet généralement d'améliorer celui-ci.

Dans tous les cas, on cherchera à obtenir les fonctions les plus robustes possible à la généralisation. On utilisera donc les bases d'apprentissage les plus grandes et variées possible. De même, on utilisera une taille de population la plus grande possible, compatible avec le temps de modélisation désiré.

On cherchera ensuite à obtenir tout d'abord des fonctions optimales, en spécifiant un critère d'arrêt élevé et un éventuel pattern de recherche, puis des fonctions peut-être moins pertinentes, mais plus variées et indépendantes, en éliminant les fonctions trop corrélées avec les optimales et en spécifiant un critère d'arrêt plus court.

Lors de la résolution d'un problème complet, nous constituons donc notre panel de fonctions caractéristiques en 4 étapes parcourant l'ensemble de l'espace de recherche :

1. Génération d'une nouvelle population et critère d'arrêt élevé AVEC prise en compte du pattern : afin d'obtenir les fonctions optimales dans l'espace spécifié par le pattern
2. Elimination des fonctions corrélées avec les optimales et critère d'arrêt bas AVEC prise en compte du pattern : afin d'obtenir des fonctions sous-optimales indépendantes dans l'espace du pattern
3. Génération d'une nouvelle population et critère d'arrêt élevé SANS prise en compte du pattern (pattern général) : afin d'obtenir les éventuelles fonctions optimales dans le reste de l'espace
4. Elimination des fonctions corrélées avec les précédentes et critère d'arrêt bas SANS prise en compte du pattern : afin d'obtenir des fonctions sous-optimales indépendantes dans le reste de l'espace

Cette méthode permet de constituer les ensembles de fonctions caractéristiques utilisés dans les modélisations présentées par la suite.

5.2.3 Étude de problèmes classiques de description musicale

Les problèmes de description musicaux classiques touchent des domaines très variés : extraits de musique, de radio, de télévision, de cinéma, etc...Bien que les sons traités soient d'origines très diverses, la majorité des problèmes abordés peuvent être considérés comme des études de timbre, c'est-à-dire, par définition, de l'ensemble des paramètres qui font la « couleur »d'un son, indépendamment de son niveau sonore et de sa fréquence fondamentale perçue, que ce soit en reconnaissance de sons d'instruments, en détection de sons percussifs, de voix, de sons d'ambiance, ou en caractérisation de l'environnement sonore (par exemple intérieur/extérieur).

Contexte des bases de signaux

Cependant, même dans un domaine précis, il est difficile de comparer les différents travaux présentés dans l'état de l'art, car leurs résultats sont en général liés à des problèmes particuliers et des contextes différents. Par exemple, en reconnaissance d'instruments, alors que [Peeters *et al.*, 2000] étudie l'ensemble des sons naturels d'instruments acoustiques, [Eronen, 2001] considère 29 instruments et 6 familles, [Brown *et al.*, 2001] se focalise sur la famille des bois, et [Eggink et Brown, 2003] utilise 5 instruments (flute, clarinette, hautbois, violon, violoncelle), etc...Dans ces travaux, le contexte d'étude est en général une base de signaux sonores connue. Le problème est identique pour la voix, certains travaux visant à la détecter dans une chanson ([Chou et Gu, 2001]), à la discriminer de la musique ([Scheirer et Slaney, 1997]), ou à la caractériser ([Tsai *et al.*, 2003]), ou pour le genre musical, chaque travail présenté utilisant sa propre taxonomie et sa propre base de signaux sonores.

Sachant qu'EDS est capable de traiter l'ensemble de ces problèmes, et afin de pouvoir comparer ses performances à l'ensemble de l'état de l'art, nous avons

construit pour chaque grande catégorie de problème descriptif une base de signaux générale, sur laquelle nous testons toutes les méthodes connues avant de lancer la modélisation automatique avec EDS.

Fonctions de référence

Enfin, nous avons constaté que de nombreux travaux utilisaient les mêmes fonctions caractéristiques de base (typiquement les fonctions Mpeg7), en plus de quelques fonctions originales. Plus précisément, ces caractéristiques sont généralement utilisées brutes pour décrire des sons courts (reconnaissance d'instrument, de voix, etc...), et également dynamiquement pour la description d'extraits musicaux et de sons longs.

Ainsi, nous avons construit deux ensembles de fonctions caractéristiques de référence, par rapport auxquels nous comparerons les fonctions synthétisées par EDS. Le premier constitue la référence pour la description de sons ; c'est l'ensemble des principales fonctions utilisées dans tous les travaux de l'état de l'art sur la description de sons d'ambiance, d'instruments, et de voix. A noter que certains travaux utilisent des sons « purs »(comme la base « Studio online »de l'IRCAM), et extraient ainsi des caractéristiques temporelles spécifiques aux instruments étudiés ; comme nous travaillons plus particulièrement sur des échantillons de sons dans un contexte de musique populaire polyphonique, nous n'utilisons pas les paramètres d'enveloppe temporelle (ADSR, centroïde temporel, etc...). De plus, ces sons ayant une durée très courte, nous nous limitons à l'utilisation de caractéristiques statiques. Enfin, nous n'avons pas sélectionné de caractéristiques perceptives, comme la douceur (« smoothness ») d'un son, qui ne sont elles-mêmes que des modèles dépendant d'autres caractéristiques de plus bas-niveau. Nous obtenons alors un ensemble de 35 fonctions caractéristiques de référence évaluant les moyenne, variance, dérivée, corrélation, de caractéristiques énergétiques (RMS), temporelles (ZCR), spectrales (statistiques), ou perceptives (MFCC).

Le second ensemble contient les fonctions de référence pour la description des extraits ; c'est l'ensemble des principales fonctions utilisées dans tous les travaux de l'état de l'art sur la description de genre, similarité, chant, et émotion dans la musique. Cet ensemble est constitué de 95 fonctions de référence, basées sur les fonctions de description de sons présentées précédemment, dont on étudie les valeurs brutes et les statistiques d'évolution sur des fenêtres d'observation de 23ms (de 20ms à 30ms dans la plupart des travaux de l'état de l'art).

Méthodes de classification

Enfin, la plupart des méthodes de classification/régression utilisées dans l'état de l'art sont des méthodes usuelles intégrées dans EDS (voir en annexe C), seuls les modèles Auto-Régressifs et les statistiques d'ordre élevé n'étant pas inclus dans le système à l'heure actuelle.

Évaluation des performances du système

Le but de ces tests est de comparer les performances d'EDS par rapport à celles de fonctions et méthodes de références tirées de l'état de l'art.

Pour chaque problème descriptif, on réalise tout d'abord un modèle optimal à partir du groupe de fonctions de référence (« REF »). On lance ensuite EDS

afin d'obtenir une centaine de fonctions suivant la méthode générale présentée en 5.2.2, puis on réalise un modèle optimal à partir de ces fonctions générées automatiquement (« EDS »).

Nous présentons pour chaque problème descriptif les performances :

- des modèles optimaux REF et EDS : erreur moyenne pour les problèmes régressifs, et pourcentage de réussite pour les problèmes de classification ;
- des meilleures fonctions REF et EDS : nous avons réalisé la recherche génétique de ces fonctions par optimisation d'un critère (corrélation pour les problèmes régressifs, coefficient de Fisher pour les problèmes de classification). Cependant, afin de pouvoir évaluer l'apport de chaque fonction dans le modèle global, nous indiquons leur performance de la même façon que les modèles : erreur, et pourcentage de réussite, pour des modèles linéaires simples. Ainsi, pour les problèmes régressifs, nous utilisons l'erreur minimale obtenue par régression linéaire (coefficients linéaires optimaux appliqués sur les valeurs de la fonction considérée) ; pour les problèmes de classification, nous utilisons le pourcentage de réussite obtenue par une classification bayésienne (recherche d'un seuil sur les valeurs de la fonction considérée).

Enfin, les performances sont présentées de deux façons : en apprentissage (calculée par validation croisée sur la base d'apprentissage), et en test (calculées sur une base de signaux différente de la base d'apprentissage).

5.3 Modélisation de descripteurs musicaux avec EDS

En utilisant les méthodes d'étude présentées précédemment, nous présentons ici les performances du système sur plusieurs problèmes variés de description musicale : la détection de voix chantée, la classification de sons d'instruments, la reconnaissance de genres musicaux, et l'évaluation de l'énergie musicale d'un extrait. Ces descripteurs sont étudiés dans divers contextes, spécifiques ou plus généraux.

5.3.1 Reconnaissance de sons de voix

La détection de voix chantée est un problème difficile, qui peut en outre être considéré à plusieurs échelles : syllabes (par opposition aux silences), phrases (par opposition aux mesures non-chantées), ou couplet entier (par opposition aux passages instrumentaux, solos, etc...). Nous traiterons ici la reconnaissance de sons vocaux, c'est-à-dire au niveau de la syllabe.

Nous étudions ce problème à différents niveaux de spécificité, de la détection générale de voix chantée à l'identification d'une voix particulière.

Détection générale de voix chantée

PROBLÈME : Nous cherchons à reconnaître les sons contenant une syllabe chantée.

BASE DE SIGNAUX : Pour cela, nous disposons de bases de 200 signaux d'une durée moyenne d'environ 200ms, extraits de musiques polyphoniques populaires variées, et contenant en parts égales des extraits vocaux et instrumen-

taux. Les types de voix contenus dans les bases sont également variés : homme, femme, grave, aigu, etc...On recherche donc une propriété très générale dans un contexte très général.

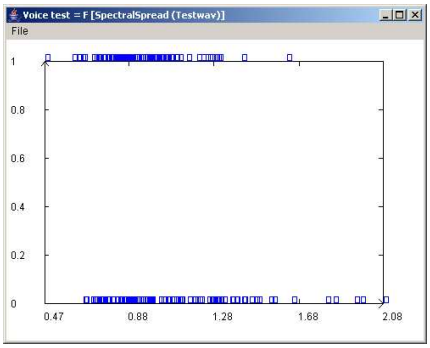
ÉTIQUETAGE : Les signaux sont étiquetés « Voix »et « Pas de voix ». Les signaux proviennent de la segmentation de passages vocaux et instrumentaux dont l'origine est certaine, et ont été validés à nouveau par une écoute individuelle. L'étiquetage est donc supposé certain.

ÉTAT DE L'ART : Les principaux travaux dans ce domaine sont récapitulés en annexe dans le tableau B.4, et inclus dans l'ensemble de fonctions de référence pour la description de sons. Notons que les modèles probabilistes de phonèmes anglais (utilisés par exemple dans [Berenzweig et Ellis, 2001]) ne font pas partie de ces fonctions de référence, car ils nécessitent un lien à une base externe d'exemples sonores.

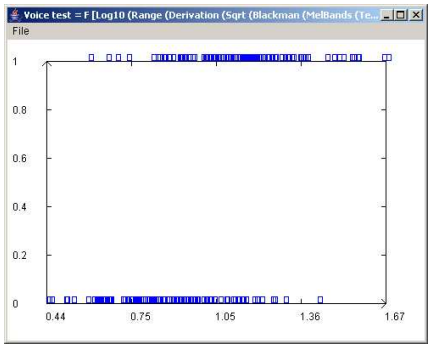
Les principales fonctions trouvées par EDS sont présentées en annexe E.1. Les meilleurs résultats sont présentés dans le tableau 5.2. La figure 5.4 présente la projection des données de test sur les axes des meilleures fonctions REF et EDS.

		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	64.5%	63%	Spectral Spread
	EDS	81%	76%	Log10 (Range (Derivation (Sqrt (Blackman (MelBands (Testwav, 24.0))))))
Meilleurs Modèles	REF	72%	69,5%	Naïve Bayes Classifier (8 fonctions)
	EDS	86,5%	78,5%	kNN Classifier (12 fonctions, 8 voisins)

Tab. 5.2: Comparaison des performances des modèles classiques et générés par EDS pour la reconnaissance de sons vocaux



Meilleure fonction de référence



Meilleure fonction EDS

Fig. 5.4: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la détection de sons vocaux

La meilleure fonction de référence, « Spectral Spread », mesure l'étalement

du spectre du signal. Cependant, sa pertinence est très faible, et les fonctions de référence ne permettent pas de réaliser un modèle pertinent des sons vocaux courts (69,5% de réussite).

La meilleure fonction trouvée par EDS est liée aux variations de l'énergie du signal dans les différentes bandes Mel. On note également quelques fonctions traitant le signal filtré dans des fréquences entre 300 et 3000Hz, qui sont proches des fréquences de la voix. L'ensemble des fonctions trouvées permet de construire un modèle plus performant : 78,5% de réussite sur la base de test. On peut montrer que la meilleure fonction trouvée par EDS est une caractéristique optimale locale, dont chacune des opérations est optimisée par rapport au critère de recherche, le Ratio Discriminant de Fisher (FDR). En effet, en observant le tableau 5.3, on constate que toute modification d'une seule opération de cette fonction entraîne une chute du FDR, que ce soit le fenêtrage, le nombre de bandes Mel, ou le reconstructeur Range, ou les mises à l'échelle Log et Sqrt.

Opérateur étudié	Fonction étudiée	Perf learn	Perf test
FONCTION ORIGINALE	Log10 (Range (Derivation (Sqrt (Blackman (MelBands (Testwav, 24.0))))))	1.20	0.83
SANS LOG	Range (Derivation (Sqrt (Blackman (MelBands (Testwav, 24.0)))))	0.79	0.60
RECONSTRUCTEUR RANGE REMPLACE PAR MEAN	Mean (Derivation (Sqrt (Blackman (MelBands (Testwav, 24.0)))))	0.00	0.04
SANS DERIVATION	Log10 (Range (Sqrt (Blackman (MelBands (Testwav, 24.0)))))	0.65	0.43
SANS SQRT	Log10 (Range (Derivation (Blackman (MelBands (Testwav, 24.0)))))	0.97	0.70
SANS FENETRE DE BLACKMAN	Log10 (Range (Derivation (Sqrt (MelBands (Testwav, 24.0)))))	0.49	0.33
20 BANDES MEL A LA PLACE DE 24	Log10 (Range (Derivation (Sqrt (Blackman (MelBands (Testwav, 20.0))))))	1.06	0.65
25 BANDES MEL A LA PLACE DE 24	Log10 (Range (Derivation (Sqrt (Blackman (MelBands (Testwav, 25.0))))))	1.18	0.80

Tab. 5.3: Importance de chaque opération pour la performance globale de la meilleure fonction EDS pour la détection de voix chantée

Reconnaissance d'une voix particulière

Nous pouvons étudier les performances du système sur un problème plus spécifique : la détection de la voix d'une chanteuse dans ses propres chansons.

PROBLÈME : Dans ce problème, nous cherchons à détecter la voix d'une chanteuse dans des extraits musicaux variés de sa composition. Nous traitons toujours le problème au niveau local de la reconnaissance de syllabes chantées.

BASE DE SIGNAUX : Pour cela, nous disposons de bases de 200 signaux

d'une durée moyenne d'environ 200ms, extraits de plusieurs chansons de la chanteuse, et contenant en parts égales des sons contenant sa voix ou pas.

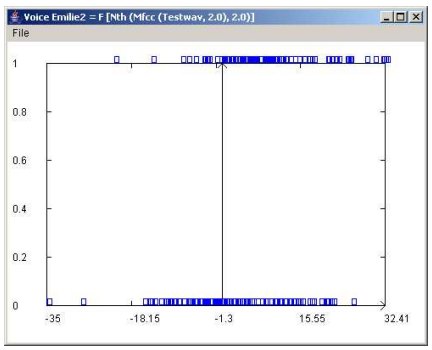
ÉTIQUETAGE : Les signaux sont étiquetés « Voix de la chanteuse »ou pas. Les signaux proviennent de la segmentation de passages chantés par elle ou non dont l'origine est certaine, et ont été validés à nouveau par une écoute individuelle. L'étiquetage est donc supposé certain.

ÉTAT DE L'ART : Bien sûr, aucun travail de recherche n'a porté sur la détection de la voix de cette chanteuse, et c'est un des avantages d'EDS de pouvoir modéliser automatiquement des descripteurs originaux. Nous comparons les résultats d'EDS à ceux obtenus avec les fonctions de référence pour la description de sons, contenant notamment les fonctions utilisées dans les travaux en description de voix chantée présentés en annexe dans le tableau B.4.

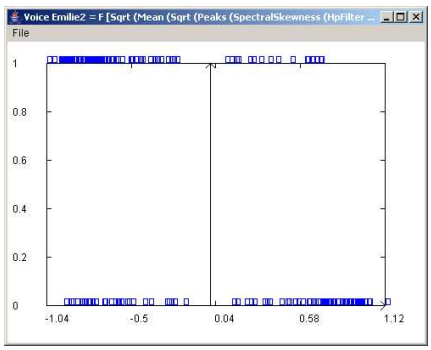
Les principales fonctions trouvées par EDS sont présentées en annexe E.2. Les meilleurs résultats sont présentés dans le tableau 5.4. La figure 5.5 présente la projection des données de test sur les axes des meilleures fonctions REF et EDS.

		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	66,5%	65,5%	2^{eme}MFCC
	EDS	80,5%	76,5%	Sqrt (Mean (Sqrt (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300.0)), 3122.0))))))
Meilleurs Modèles	REF	78,50%	72.5%	SVM Classifier (5 fonctions)
	EDS	85,5%	82,50%	SVM Classifier (10 fonctions)

Tab. 5.4: Comparaison des performances des modèles de référence et générés par EDS pour la reconnaissance de la voix d'une chanteuse



Meilleure fonction de référence



Meilleure fonction EDS

Fig. 5.5: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la détection de la voix d'une chanteuse

A nouveau, la meilleure fonction trouvée par EDS est plus performante que la meilleure combinaison de fonctions de référence (76,5% de réussite contre

72,5%). Cependant, les performances du modèle final sont équivalentes à celles de la détection de voix chantée en général.

A nouveau, la meilleure fonction trouvée par EDS est une caractéristique optimale locale, dont chacune des opérations est optimisée par rapport au critère de recherche, le Ratio Discriminant de Fisher (FDR). En observant le tableau 5.5, on constate que toute modification d'une seule opération de cette fonction entraîne une chute du FDR, que ce soit le fenêtrage et la taille des fenêtres, la fréquence de coupure du filtre, la caractéristique spectrale « skewness », ou les mises à l'échelle Sqrt. Seule la dernière opération « Sqrt »n'améliore pas la performance sur la base de test (FDR=1.09 contre FDR=1.08 sans Sqrt), alors qu'elle donne un meilleur résultat sur la base d'apprentissage.

Opérateur étudié	Fonction étudiée	Perf learn	Perf test
FONCTION ORIGINALE	Sqrt (Mean (Sqrt (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300)), 3122))))))	1.17	1.08
SANS SQRT	Mean (Sqrt (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300)), 3122)))))	1.01	1.09
RECONSTRUCTEUR MEAN REMPLACE PAR VARIANCE	Sqrt (Variance (Sqrt (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300)), 3122))))))	0.03	0.00
SANS 2^{eme}SQRT	Sqrt (Mean (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300)), 3122)))))	1.13	1.02
RECONSTRUCTEUR PEAKS REMPLACE PAR HOLES	Sqrt (Mean (Sqrt (Holes (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300)), 3122))))))	0.64	0.83
OPERATION SKEWNESS REMPLACEE PAR SPREAD	Sqrt (Mean (Sqrt (Peaks (SpectralFlatness (HpFilter (Triangle (Split (Testwav, 300)), 3122))))))	0.17	0.17
SANS FILTRAGE	Sqrt (Mean (Sqrt (Peaks (SpectralSkewness (Triangle (Split (Testwav, 300)))))))	0.01	0.04
FREQUENCE DE COUPURE DU FILTRE 3122Hz REMPLACE PAR 3000Hz	Sqrt (Mean (Sqrt (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300)), 3000))))))	1.00	0.98
SANS FENETRE TRIANGLE	Sqrt (Mean (Sqrt (Peaks (SpectralSkewness (HpFilter (Split (Testwav, 300), 3122))))))	0.56	0.84
TAILLE DE FENETRE 250 A LA PLACE DE 300	Sqrt (Mean (Sqrt (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 250)), 3122))))))	0.81	1.00

Tab. 5.5: Importance de chaque opération pour la performance globale de la meilleure fonction EDS pour la détection de la voix d'une chanteuse

Reconnaissance d'une voix particulière dans une chanson précise

Enfin, EDS est également utilisable dans un contexte extrêmement spécifique : la détection de la voix d'une chanteuse dans une de ses chansons.

PROBLÈME : Nous cherchons à localiser la voix d'une chanteuse dans une des ses chansons à partir d'un nombre restreint d'exemples.

BASE DE SIGNAUX : Pour cela, nous disposons d'une bases de 50 signaux d'une durée moyenne d'environ 200ms extraits de la chanson étudiée : 25 contenant sa voix et 25 sans. Le test est réalisé sur 2 minutes de la chanson, soit environ 500 samples.

ÉTIQUETAGE : Les signaux sont étiquetés « Voix de la chanteuse »ou pas. Les signaux proviennent de la segmentation de passages chantés par elle ou non dont l'origine est certaine, et ont été validés à nouveau par une écoute individuelle. L'étiquetage est donc supposé certain.

ÉTAT DE L'ART : Comme pour le problème précédent, nous comparons les résultats d'EDS à ceux obtenus avec les fonctions de référence pour la description de sons, contenant notamment les fonctions utilisées dans les travaux en description de voix chantée présentés en annexe dans le tableau B.4.

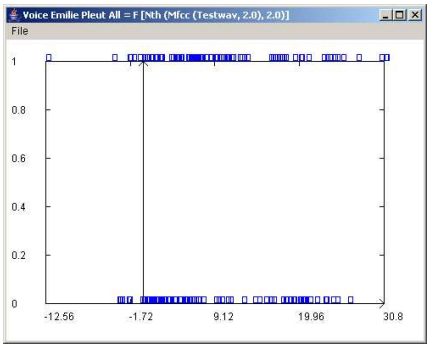
Les principales fonctions trouvées par EDS sont présentées en annexe E.3. Les meilleurs résultats sont présentés dans le tableau 5.6. La figure 5.6 présente la projection des données de test sur les axes des meilleures fonctions REF et EDS.

		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	74,4%	57,9%	2^{eme}MFCC
	EDS	90,7%	79,5%	Square (Log10 (Abs (SpectralFlatness (BpFilter (Normalize (BpFilter (Blackman (Correlation (Abs (BpFilter (Normalize (Blackman (Correlation (BpFilter (Normalize (Testwav), 308, 965), Testwav))), 232, 1596)), Testwav)), 1256, 244)), 326, 1848)))))
Meilleurs Modèles	REF	81,48%	54.14%	Neural Nets (3 fonctions, 3 couches)
	EDS	90,7%	79,5%	Naive Bayes (1 fonction)

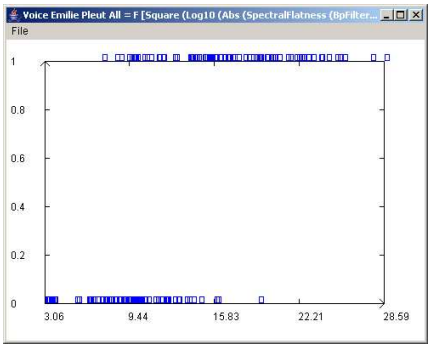
Tab. 5.6: Comparaison des performances des modèles de référence et générés par EDS pour la reconnaissance de la voix d'une chanteuse dans une chanson précise

La difficulté de cette modélisation est la taille limitée de la base qui pose un problème de généralisation. La meilleure fonction trouvée par EDS est beaucoup plus performante que la meilleure combinaison de fonctions de référence (79% de réussite contre 54%). Et le meilleur modèle obtenu avec des fonctions EDS est un simple classifieur bayésien utilisant uniquement cette fonction, ce qui montre la pertinence de celle-ci.

Cette fonction « Square (Log10 (Abs (SpectralFlatness (BpFilter (Normalize (BpFilter (Blackman (Correlation (Abs (BpFilter (Normalize (Blackman



Meilleure fonction de référence



Meilleure fonction EDS

Fig. 5.6: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la détection de la voix d'une chanteuse dans une chanson précise

(Correlation (BpFilter (Normalize (Testwav), 308, 965), Testwav))), 232, 1596)), Testwav)), 1256, 244)), 326, 1848))))) »semble compliquée, mais reste fondamentalement proche de « SpectralFlatness (Correlation (Abs (BpFilter (Correlation (BpFilter (Testwav), Testwav)), Testwav) », c'est-à-dire une mesure de l'harmonicité (Spectral Flatness) de la corrélation de la corrélation absolue et lissée (BpFilter) du signal avec sa partie filtrée par un passe-bande. On peut observer sa représentation sous forme d'arbre d'opérations sur la figure 5.7.

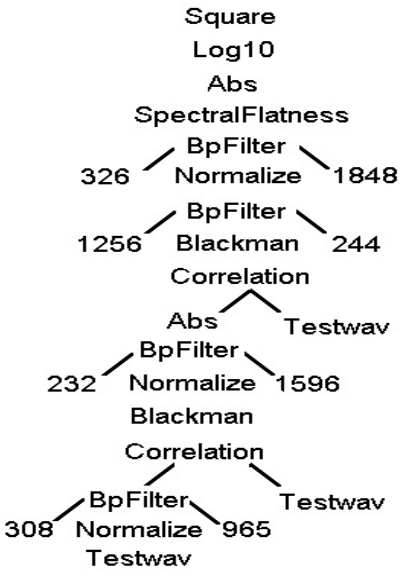


Fig. 5.7: Représentation sous forme d'arbre d'opérations de la meilleure fonction EDS pour la détection de la voix d'une chanteuse dans une chanson précise

Cette fonction reste difficile à interpréter ; le tableau 5.7 présente l'étude

de l'influence des diverses opérations de la fonction sur ses performances. Dans ce tableau, la fonction a été simplifiée en « Square (Log10 (SpectralFlatness (BpFilter (BpFilter (Blackman (Correlation (Abs (BpFilter (Normalize (Blackman (Correlation (BpFilter (Testwav, 308.0, 965.0), Testwav))), 232.0, 1596.0)), Testwav)), 1256.0, 244.0), 326.0, 1848.0)))) », les simplifications « Abs (SpectralFlatness) = Spectral Flatness »et « Correlation (Filter (Normalize(...))) = Correlation (Filter (...)) »ayant été rajoutées dans le système a posteriori.

On constate à nouveau que cette fonction est optimale, et que toute modification d'une de ses opérations entraîne une chute du FDR en apprentissage, que ce soit les fenêtrages, les filtrages, les bandes passantes de filtres, ou la caractéristique spectrale « flatness ». Par contre, certains paramétrages permettent d'obtenir une légère amélioration du FDR en test, qui reste cependant insignifiante : FDR=1.72 (contre 1.69 pour la fonction originale) pour la mise au carré et une légère variation de la bande passante du filtre, FDR=1.78 en supprimant la 2^{eme}fenêtre de Blackman. Ainsi, cette fenêtre semble améliorer spécifiquement les résultats pour la base d'apprentissage, sans rien apporter pour des bases générales.

5.3.2 Perception de la percussivité de sons polyphoniques

En marge des problèmes de description classiques, on voudra définir des caractéristiques sonores originales utiles dans le cadre d'applications musicales (voir le chapitre 6). Ces caractéristiques peuvent être les jugements perceptifs d'un auditeur (un son « harmonique », « bruité », etc...) qui doit être certain de ses choix, ou d'un groupe d'auditeurs pour lequel il est nécessaire de réaliser une évaluation statistique.

Nous présentons ici un problème de perception de sons polyphoniques : la percussivité. Ce problème est différent de celui de classification d'instruments percussifs, qui se rapproche de problème général présenté en 5.3.3. Ici, la percussivité d'un son quelconque (extrait de musique, bruit d'ambiance, de voix, etc...) est une notion subjective qui mesure sa « proximité de timbre »avec un instrument percussif.

Modélisation

PROBLÈME : Nous abordons ici le problème booléen, consistant à classifier un son polyphonique en « percussif »ou « non-percussif ».

BASES DE SIGNAUX : Nous disposons pour cela de bases de 200 sons polyphoniques d'environ 200ms, associés à un étiquetage manuel binaire « percussif / non-percussif ». C'est donc un problème de classification booléenne.

ÉTIQUETAGE : Chaque étiquette est réalisée et validée par un auditeur ; l'étiquetage est donc considéré comme certain dans le contexte de la perception de cet auditeur, mais sa généralisation n'a pas été testée.

ÉTAT DE L'ART : Ce descripteur original n'est pas abordé dans l'état de l'art. Cependant, l'ensemble de fonctions de référence pour la description de sons contient les travaux sur la classification de sons percussifs purs présentés dans le tableau B.1, qui peuvent être utiles pour cette modélisation.

Les principales fonctions trouvées par EDS sont présentées en annexe E.4. Les meilleurs résultats sont présentés dans le tableau 5.8. La figure 5.8 présente

Opérateur étudié	Fonction étudiée	Perf learn	Perf test
FONCTION ORIGINALE	Square (Log10 (SpectralFlatness (BpFilter (BpFilter (Blackman (Correlation (Abs (BpFilter (Normalize (Blackman (Correlation (BpFilter (Testwav, 308.0, 965.0), Testwav))), 232.0, 1596.0)), Testwav)), 1256.0, 244.0), 326.0, 1848.0))))	2.72	1.69
SANS SQUARE	Log10 (SpectralFlatness ...	2.23	1.70
SANS LOG	Square (SpectralFlatness ...	0.05	0.21
OPERATION FLATNESS REMPLACEE PAR SPREAD	Square (Log10 (SpectralSpread (BpFilter ...	0.16	0.49
SIMPLIFICATION DES 2 FILTRES PASSE-BANDE EN 1	...(SpectralFlatness (BpFilter (Blackman ...	2.44	1.13
SANS FENETRE BLACKMAN	...(BpFilter (BpFilter (Correlation (Abs ...	1.28	0.78
FENETRE BLACKMAN REMPLACE PAR HANNING	...(BpFilter (BpFilter (Hanning (Correlation (Abs ...	1.26	0.84
SANS 2^{eme}FILTRAGE	...(Correlation (Abs (Normalize (Blackman (Correlation ...	1.00	0.74
SANS 2^{eme}FENETRE DE BLACKMAN	...(Correlation (Abs (BpFilter (Normalize (Correlation ...	1.11	1.78
2^{eme}FENETRE DE BLACKMAN REMPLACE PAR HANNING	...(Correlation (Abs (BpFilter (Normalize (Hanning (Correlation ...	1.95	1.72
SANS 3^{eme}FILTRAGE	...(BpFilter (Normalize (Blackman (Correlation (Testwav, Testwav))), 232.0, 1596.0)) ...	0.77	0.67
BANDE PASSANTE 500-1000Hz AU LIEU DE 308-965Hz	...(Blackman (Correlation (BpFilter (Testwav, 500, 1000), Testwav))), ...	2.22	1.72

Tab. 5.7: Importance de chaque opération pour la performance globale de la meilleure fonction EDS pour la détection de la voix d'une chanteuse dans une chanson précise

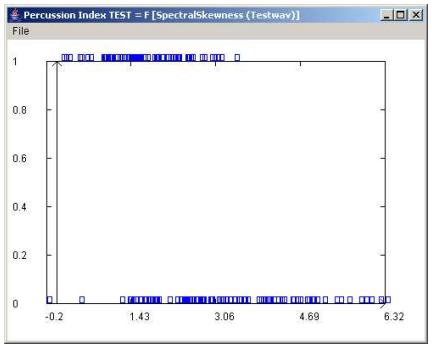
la projection des données de test sur les axes des meilleures fonctions REF et EDS.

La meilleure fonction de référence, « Spectral Skewness », mesure l'asymétrie de la répartition des fréquences autour du pic spectral. La meilleure classification, utilisant les SVM, donne 79,7% de bonnes classifications.

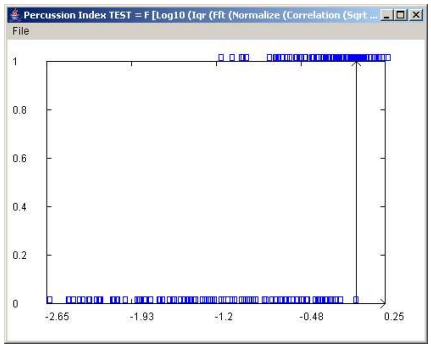
Les fonctions trouvées par EDS sont très variées, et seule une fonction utilise le Spectral Skewness. La meilleure fonction est liée à l'étalement du spectre de l'autocorrélation du signal brut avec le signal filtré par un passe-haut à 2131Hz,

		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	78,5%	73,08%	Spectral skewness
	EDS	86,5%	79,12%	Log10 (Iqr (Fft (Normalize (Correlation (Sqrt (HpFilter (Testwav, 2131.0)), Testwav)))))
Meilleurs Modèles	REF	87%	79,7%	SVM Classifier (10 fonctions)
	EDS	92.5%	80.7692%	Kernel Density Classifier (16 fonctions)

Tab. 5.8: Comparaison des performances des modèles classiques et générés par EDS pour la perception de la percussivité de sons polyphoniques



Meilleure fonction de référence



Meilleure fonction EDS

Fig. 5.8: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la caractérisation de sons polyphoniques percussifs

et compressé à la puissance 1/2. Assez difficile à interpréter, elle mesure une caractéristique harmonique (spectre, autocorrélation) des hautes-fréquences du signal, dans lesquelles se situe une part importante du bruit des sons percussifs. Cette fonction donne à elle seule avec un classifieur bayésien simple, un modèle aussi performant qu'un modèle SVM utilisant toutes les fonctions de référence.

Cependant, bien qu'on puisse obtenir des performances en apprentissage bien meilleures, l'évaluation du modèle sur la base de test fournit toujours une performance d'environ 80%. Ceci signifie qu'EDS a su modéliser en une fonction une dimension significative de la percussivité sonore. Les 20% d'erreurs sont principalement des sons étiquetés « non-percussifs »classifiés comme percussifs. En écoutant ces sons, on remarque que beaucoup sont un mélange d'un son de type percussif avec un son ayant une forte composante harmonique ; la présence de sons légèrement hétérogènes dans la base, et leur étiquetage booléen validé par un seul auditeur, sont donc en partie responsables des limites du modèle.

Étapes typiques d'une recherche génétique de fonction

Une des fonctions trouvées par EDS pour discriminer les sons percussifs est « Log10 (Abs (SpectralCentroid (BpFilter (Blackman (Testwav), 3216.0, 9179.0)))) », qui évalue simplement la fréquence fondamentale du spectre entre environ 3200Hz et 9200Hz.

Cette fonction a été validée à la 22^{eme}population d'une recherche génétique, chaque population comptant 20 fonctions. Les grandes étapes de la recherche sont les suivantes :

- 1^{ere} population construite aléatoirement
  la *meilleure fonction initiale* est une fonction de référence, le centroïde spectral (FDR=0.38) ;
- 2^{eme}population construite par transformations génétiques et combinaisons des meilleures fonctions de la 1^{ere} population
  les *premières améliorations* fournissent les fonctions « Power (SpectralCentroid (Testwav), -1.0) »(FDR=0.48), et « Log10 (SpectralSkewness (BpFilter (Normalize (Testwav), 4195.0, 1023.0))) »(FDR=0.48) ;
- les populations suivantes, construites par transformations génétiques et combinaisons des meilleures fonctions, permettent d'*améliorer progressivement les performances* des fonctions ;
- à la 14^{eme}population, on obtient une fonction EDS meilleure que la meilleure fonction de référence Spectral Skewness (FDR=0.93) : « Log10 (Abs (SpectralCentroid (BpFilter (Testwav, 13028.0, 1823.0)))) »(FDR=1.00) ;
- dans les populations suivantes, le système *ajuste les paramètres des opérations* de filtrage et de fenêtrage ;
- à la 18^{eme}population, le système *trouve la solution* « Log10 (Abs (SpectralCentroid (BpFilter (Blackman (Testwav), 3216.0, 9179.0)))) »(FDR=1.13) ;
- les dernières populations vérifient que la solution trouvée est un *optimum local* ;
  à la 22^{eme}population, la *recherche est arrêtée* après 4 populations sans amélioration, et la meilleure fonction est validée.

5.3.3 Classification de sons d'instruments

Notre dernier exemple de description de sons courts est un problème multi-classes de classification d'instruments. Nous abordons ici ce problème dans sa globalité, en cherchant des fonctions générales permettant de discriminer simultanément toutes les classes. L'autre approche possible, consistant à décomposer le problème multi-classe en un ensemble de problèmes de discrimination booléens traités séparément (du type « 1 instrument ou les autres »), n'est pas présentée ici.

Discrimination de 4 sons d'instruments monophoniques

PROBLÈME : Nous cherchons à classifier les sons purs monophoniques provenant de 4 instruments acoustiques : piano, guitare, violon, et violoncelle.

CONTEXTE DES BASES DE SIGNAUX : Les bases d'apprentissage et de test sont constituées de 400 signaux de durée approximative 200ms, composés en parties égales de violon, violoncelle, guitare, et piano. Les sons sont monophoniques, obtenus par segmentation par variations d'énergie sur des pièces monophoniques.

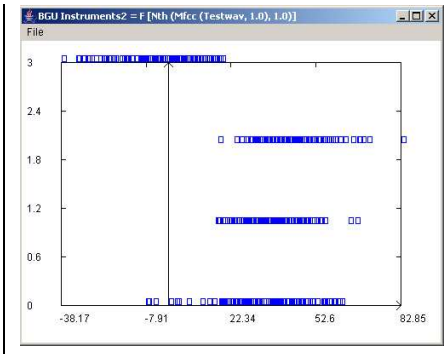
ÉTIQUETAGE : L'étiquetage est parfait, l'origine de chaque signal étant connue.

ÉTAT DE L'ART : Les principaux travaux dans ce domaine sont récapitulés en annexe dans le tableau B.1, et inclus dans l'ensemble de fonctions de référence pour la description de sons.

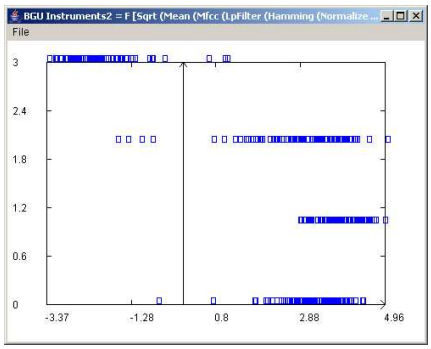
Les principales fonctions trouvées par EDS sont présentées en annexe E.5. Les meilleurs résultats sont présentés dans le tableau 5.9. La figure 5.9 présente la projection des données de test sur les axes des meilleures fonctions REF et EDS.

		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	56,25%	50,75%	1^{er} MFCC
	EDS	56,7%	59%	Sqrt (Mean (Mfcc (LpFilter (Hamming (Normalize (Testwav)), 1394), 5)))
Meilleurs Modèles	REF	90%	89%	kNN Classifier (11 fonctions, 6 voisins)
	EDS	91%	90,5%	Neural Nets Classifier (16 fonctions, 3 couches)

Tab. 5.9: Comparaison des performances des modèles classiques et générés par EDS pour la classification de sons de 4 instruments acoustiques



Fonctions de référence



Fonctions EDS

Fig. 5.9: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la classification de 4 instruments

Les fonctions de référence permettant le mieux de discriminer nos 4 instruments sont le 1^{er} coefficient MFCC, lié au logarithme de l'énergie du signal, et 4 paramètres liés au spectre du signal (centroïde, spread, skewness, roll-off). Ils permettent d'obtenir un modèle offrant 89% de bonnes classifications sur les 400 signaux de la base de test.

Les meilleures fonctions trouvées par EDS sont principalement des fonctions utilisant les MFCC et le centroïde spectral du signal, après des filtrages variés.

Le modèle donne 90,5% de bonnes classifications sur la base de test.

Les fonctions EDS permettent d'améliorer le modèle, mais pas de manière significative. Cependant, le fait que les fonctions trouvées soient plus robustes (59% de réussite en test pour la meilleure), permet d'obtenir un modèle également plus robuste. Ceci est confirmé par la légère baisse de l'erreur relative des prédictions, de 21,6% à 14,5%.

Discrimination des 3 instruments les plus proches

Cependant, nous avons noté que les deux modèles REF et EDS permettaient de reconnaître un instrument particulier à 100% (le violon), et que la confusion portait entre les 3 autres instruments. Comme on peut le constater sur la figure 5.9, les fonctions EDS sont précisément focalisées sur la discrimination de cet instrument. Nous avons donc réalisé une nouvelle base limitée aux sons des trois autres intruments (violoncelle, piano, guitare).

Si nous utilisons les fonctions trouvées précédemment par EDS pour résoudre le problème à 4 instruments, la meilleure fonction a seulement une performance de 66,3%, et le modèle résultant un taux de réussite de 84,3% sur la base de test. Ces performances sont moins bonnes que celles obtenues par les fonctions de référence. Ainsi, les meilleures fonctions trouvées précédemment par EDS permettent spécifiquement de reconnaître le violon des autres instruments, et ne sont pas optimisées pour la discrimination entre les trois instruments restants.

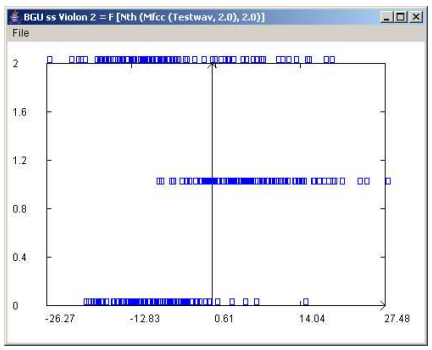
Nous avons donc lancé une nouvelle série de recherches génétiques afin de trouver des fonctions spécifiques au problème à 3 instruments, dans les mêmes conditions que précédemment.

Les résultats sont présentés dans le tableau 5.10. La figure 5.10 présente la projection des données de test sur les axes des meilleures fonctions REF et EDS.

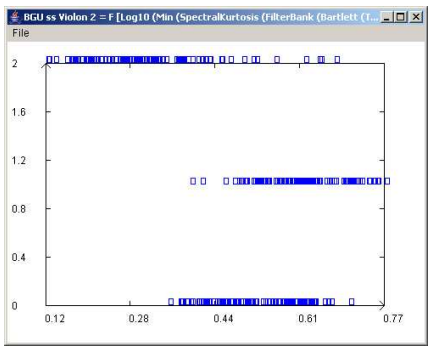
		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	55,6%	58%	2^{eme}MFCC
	EDS	70,3%	68,6%	Log10 (Min (SpectralKurtosis (FilterBank (Bartlett (Testwav), 10))))
Meilleurs Modèles	REF	86,3%	87,6%	kNN Classifier (9 fonctions, 9 voisins)
	EDS	91,6%	95%	Neural Nets Classifier (14 fonctions, 3 couches)

Tab. 5.10: Comparaison des performances des modèles classiques et générés par EDS pour la classification de sons de 3 instruments acoustiques

Cette fois, la meilleure fonction trouvée par EDS est bien meilleure que la meilleure fonction de référence, et l'utilisation des fonctions pour la modélisation par un réseau de neurones à 3 couches cachées fournit une performance de 95% en test. En effet, les fonctions de référence sont des caractéristiques générales pouvant être utilisées pour décrire les sons de toutes origines, mais sont moins efficaces pour résoudre des problèmes descriptifs plus précis. A l'inverse, en construisant des fonctions plus spécifiques, EDS est capable de s'adapter à la résolution de ces problèmes plus précis. Le fait que les performances du modèle



Fonctions de référence



Fonctions EDS

Fig. 5.10: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la classification de 3 instruments

soient meilleures en test, laisse penser que l'étiquetage de la base de test est plus précis que celui de la base d'apprentissage.

Effet de contexte

Enfin, nous avons réalisé un modèle dans un contexte très spécifique, en cherchant à reconnaître chacune des deux guitares utilisées pour l'enregistrement des signaux de la base initiale, à partir d'une petite base de 50 signaux pour chaque son de guitare.

La meilleure fonction de référence est le « Spectral Decrease »avec 98% de réussite.

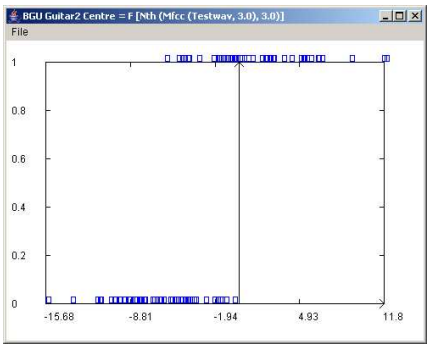
En lançant EDS, le système trouva très rapidement que la fonction « Mean (Signal) », qui fait simplement la moyenne des échantillons du signal, donne les mêmes performances. Nous avons ainsi remarqué que le signal de l'enregistrements d'une des deux guitares n'était pas centré, et EDS s'est focalisé sur cette propriété simple sans chercher à modéliser le son lui-même.

Cet exemple simple montre que les propriétés découvertes par EDS sont dépendantes du contexte, et ne concernent pas forcément une propriété sonore. Ainsi, EDS peut découvrir certaines propriétés simples que les fonctions générales de traitement audio ne peuvent pas détecter. De même, la méthode de segmentation utilisée pour générer les signaux de la base peut avoir des effets différents suivant le type de son extrait (par exemple des signaux plus longs pour des notes tenues). EDS pouvant modéliser aussi bien les propriétés sonores que les propriétés contextuelles, il est important de s'assurer de la justesse du contexte défini dans la base de signaux.

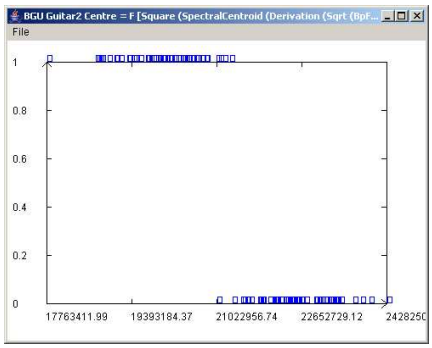
Après avoir centré tous les signaux de la base pour éliminer cette propriété, la meilleure fonction de référence est le 3^{eme}coefficient MFCC, avec 89% de bonnes classifications en apprentissage (87% en test), tandis que la meilleure fonction trouvée par EDS, « Square (SpectralCentroid (Derivation (Sqrt (Bp-Filter (Normalize (Testwav), 5766.0, 3842.0))))) », fournit 95% de réussite en apprentissage, et 98% en test.

La figure 5.11 présente la projection des données de test sur les axes de chacune de ces fonctions, et on constate bien que la séparation engendrée par

la fonction EDS est plus marquée.



Meilleure fonction de référence



Meilleure fonction EDS

Fig. 5.11: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la classification de 2 sons de guitare

5.3.4 Reconnaissance du genre d'extraits musicaux

Après la description locale de sons courts, nous abordons maintenant la description d'extraits musicaux de durée plus importante, de l'ordre de plusieurs secondes. Peu de travaux ont été réalisés jusqu'à présent dans ce domaine à partir du signal acoustique seul. Les principaux concernent l'extraction du tempo et la classification en genre musical, et quelques travaux récents étudient la détection de voix chantée, et l'émotion musicale. Nous présentons ici les performances d'EDS sur un problème de reconnaissance de 4 genres musicaux.

Classification en 4 genres musicaux

Comme pour la classification de sons par instrument, tous les travaux de l'état de l'art en classification en genre musical (voir le tableau B.2 en annexe) portent sur des problèmes différents, quant aux genres utilisés (rock, classique, pop, hip-hop, piano, quatuor à cordes, dance, etc...) et à la durée des échantillons traités (extraits courts, titres entiers).

De plus, le choix des genres à classifier repose sur la taxonomie utilisée (voir [Aucouturier et Pachet, 2003]). Typiquement, une taxonomie trop précise est inutilisable car elle reflète la perception musicale de l'expert qui l'a construite. Par exemple, la distinction entre « Pop »et « Rock », qui sont pourtant deux grands genres musicaux, n'est pas forcément perçue de la même façon par l'ensemble des auditeurs. Souvent, c'est la différence de structure globale (à l'échelle de la minute) qui va faire de deux titres possédant des caractéristiques similaires (en termes d'instrumentation, de rythme, etc...), sont de genres différents.

De notre point de vue acoustique, nous n'abordons pas la structure à long terme des titres musicaux. Ainsi, plutôt que le genre, il est plus intéressant d'utiliser le notion de « timbre global »d'un extrait musical, qui ne nécessite pas l'établissement d'une taxonomie, mais simplement le regroupement par similarité auditive.

PROBLÈME : Nous avons donc choisi de reconnaître des extraits possédant des caractéristiques acoustiques spécifiques de 4 genres : « Dance »(rythme fort régulier, sons électroniques, chant fort), « Hip-Hop »(rythme fort saccadé, chant prédominant), « Symphonie classique »(sons orchestraux), et « Pop/Rock »(caractéristiques moyennes). Comme pour la classification de sons d'instruments, nous abordons le problème directement

BASE DE SIGNAUX : Les tests réalisés ont montré que quelques secondes étaient suffisantes pour reconnaître le genre de ces extraits caractéristiques. Ainsi, nous disposons de deux bases de 100 signaux d'une durée de 5s, extraits de musiques variées, homogènes et dont le timbre est représentatif de leur genre musical. Le contexte est donc assez précis, dans la mesure où il représente un ensemble de titres considérés comme très représentatifs d'un genre précis dans une taxonomie limitée à 4 classes. C'est pourquoi nous pouvons réaliser cette étude sur des bases définies par un petit nombre d'exemples.

ÉTIQUETAGE : Les extraits étant caractéristiques, les signaux sont étiquetés de manière certaine dans une des 4 classes de genres.

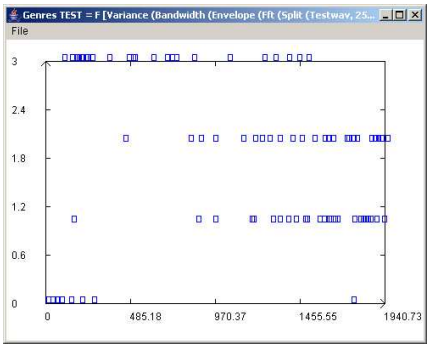
ÉTAT DE L'ART : Les principaux travaux dans ce domaine sont récapitulés en annexe dans le tableau B.2, et inclus dans l'ensemble de fonctions de référence pour la description d'extraits musicaux.

Les principales fonctions trouvées par EDS sont présentées en annexe E.6. Les meilleurs résultats sont présentés dans le tableau 5.11. La figure 5.12 présente la projection des données de test sur les axes des meilleures fonctions REF et EDS.

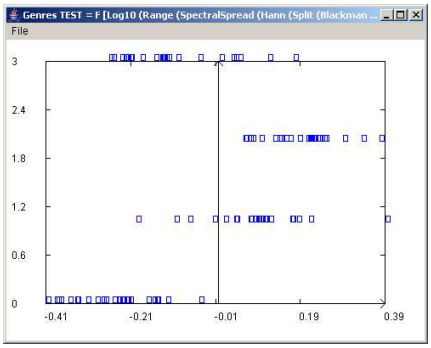
		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	58%	53%	Variance (Bandwidth (Envelope (Fft (Split (Testwav, 250.0)), 50.0), 10.0))
	EDS	61%	64%	Log10 (Range (SpectralSpread (Hann (Split (Blackman (Testwav), 153.0)))))
Meilleurs Modèles	REF	79%	80%	Neural Nets Classifier (23 fonctions, 3 couches)
	EDS	82%	80%	kNN Classifier (25 fonctions, 8 voisins)

Tab. 5.11: Comparaison des performances des modèles classiques et générés par EDS pour la reconnaissance d'extraits caractéristiques de genres musicaux

Ainsi, même si EDS est capable de construire des fonctions plus pertinentes que les fonctions de référence suivant le critère du coefficient de Fisher, les performances des modèles finals sont équivalentes. Le problème étant défini par un petit nombre de signaux (25 par genre), la généralisation du modèle n'est pas robuste, et ni les fonctions EDS, ni les fonctions de référence ne peuvent assurer celle-ci simultanément sur les 4 classes. Ceci montre l'importance de constituer des bases de signaux variés et suffisamment larges pour permettre au système de trouver des fonctions généralisables.



Meilleure fonction de référence



Meilleure fonction EDS

Fig. 5.12: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS, pour la discrimination entre 4 genres musicaux

5.3.5 Perception de l'énergie globale d'extraits musicaux

Dans le domaine de la parole, de nombreux travaux se sont attachés à la reconnaissance des émotions, notamment la joie, la tristesse, l'énervement, et la crainte (voir [Oudeyer, 2002]).

L'extension de ces notions au caractère émotionnel véhiculé par des extraits musicaux (« music mood »), fait l'objet de quelques travaux récents, notamment [Liu *et al.*, 2003b], qui applique le modèle émotionnel de Thayer ([Thayer, 1989]) à la musique classique, en prenant en considération deux axes émotionnels : l'énergie et la tension.

Cependant, le modèle d'énergie musicale de [Liu *et al.*, 2003b] ne prend en compte que le niveau sonore RMS de l'extrait considéré, supposant que l'énergie ne dépend que du niveau d'écoute. Nous présentons ici une modélisation de l'énergie musicale indépendante du niveau sonore, partant de l'hypothèse qu'un extrait « énergique »écouté à un faible volume reste plus énergique qu'un extrait « calme »écouté à un niveau élevé. Nous cherchons ainsi une modélisation de l'énergie musicale sur des critères acoustiques.

PROBLÈME : Nous cherchons ici à modéliser l'énergie subjective perçue à l'écoute d'un extrait musical, présentée en section 3.2. C'est un problème de régression, dans lequel on associe à chaque extrait un niveau d'énergie entre 0 et 1.

BASE DE SIGNAUX : Nous disposons de 2 bases de 200 signaux d'extraits musicaux de 5 secondes.

PRÉCISION DE L'ÉTIQUETAGE : Chaque signal est étiqueté par un niveau d'énergie entre 0 et 1, résultant de l'étude des résultats de tests perceptifs spécifiques (voir section 3.2.1). La validation statistique de ces tests perceptifs (voir section 5.1.3) montre que les évaluations de divers auditeurs présentent des variations d'environ 10%, c'est la précision de notre étiquetage.

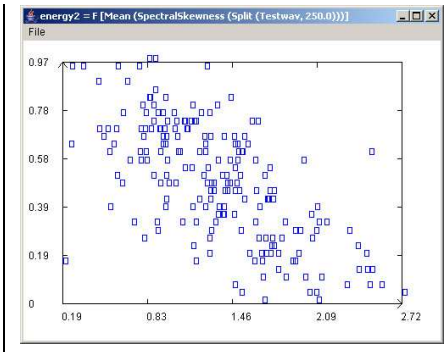
ÉTAT DE L'ART : Le seul travail sur l'énergie musicale est celui que nous avons réalisé empiriquement, présenté dans la section 3.2.

Les principales fonctions trouvées par EDS sont présentées en annexe E.7. Les meilleurs résultats sont présentés dans le tableau 5.12. La figure 5.13 présente la projection des données de test sur les axes des meilleures fonctions REF et

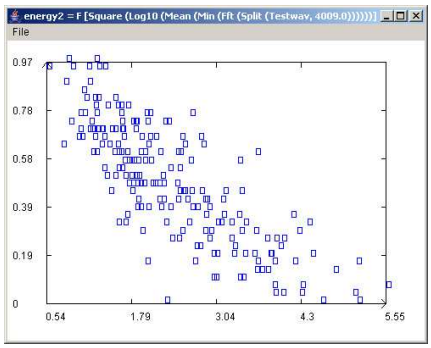
EDS. Les performances données correspondent à la « précision »des fonctions et modèles, c'est-à-dire le complément à 100% de l'erreur relative moyenne pour la reconstruction de l'étiquetage (81% signifie que l'erreur relative moyenne sur les valeurs d'étiquettes est 19%).

		Perf learn	Perf test	Fonction
Meilleures Fonctions	REF	54,5%	68,1%	Mean (SpectralSkewness (Split (Testwav, 250.0)))
	EDS	73,9%	81%	Square (Log10 (Mean (Min (Fft (Split (Testwav, 4009.0))))))
Meilleurs Modèles	REF	69,8%	81,1%	Model Trees Classifier (25 fonctions)
	EDS	78,2%	85,1%	kNN Classifier (24 fonctions, 10 voisins)

Tab. 5.12: Comparaison des performances des fonctions classiques et générées par EDS pour la modélisation de l'énergie musicale



Meilleure fonction de référence



Meilleure fonction EDS

Fig. 5.13: Visualisation de la projection des signaux de test sur les axes des meilleures fonctions de référence et EDS en fonction de leur étiquette perceptive, pour l'évaluation de l'énergie musicale

La meilleure fonction trouvée par EDS fournit à elle seule un modèle linéaire simple aussi performant que le meilleur modèle combinant les fonctions de référence (81% de précision). C'est pourtant une fonction simple liée à la succession de minima d'énergie dans le spectre, sans distinction de bandes de fréquences.

On constate que les performances d'une fonction ou d'un modèle sur la base de test sont systématiquement meilleurs que sur la base d'apprentissage. Ceci signifie que l'étiquetage de la base de test est plus « proche »de caractéristiques du signal. En analysant les tests perceptifs, on remarque que la dispersion des résultats est plus faible en moyenne sur les signaux de la base de test, c'est-à-dire que l'étiquetage de cette dernière est plus certain. Le fait que les fonctions trouvées par EDS, comme les fonctions de référence, ont des meilleures performances en test sur une base mieux étiquetée, indique que ces fonctions sont bien des caractéristiques générales corrélées à la perception de l'énergie musicale.

Le meilleur modèle obtenu est un classifieur des 10 plus-proches voisins construit à partir des fonctions EDS, dont la précision (85.1%) est proche de la précision de la base (90%), limite naturelle de la précision des modèles.

5.4 Conclusion

Le système EDS permet de synthétiser automatiquement des extracteurs de descripteurs à partir de signaux, en construisant des fonctions caractéristiques pertinentes spécifiques. Dans le domaine de la description de signaux musicaux, les fonctions construites par EDS sont généralement plus performantes que les fonctions de référence comme celles de la norme MPEG7, suivant un critère de pertinence donné (corrélation de Pearson pour les régressions, coefficient de Fisher pour les classifications). Cependant, ces fonctions sont spécifiques à un contexte précis défini par l'ensemble des signaux contenus dans la base d'apprentissage, et la capacité de généralisation des fonctions dépend de la qualité de cette base, en termes de variété de signaux et de précision de l'étiquetage de la propriété à modéliser.

De nombreux modèles descriptifs peuvent ainsi être générés facilement et automatiquement par EDS, notamment dans le cadre d'applications musicales. Dans les applications de composition, comme par exemple les mosaïques musicales (section 6.4.4), il est intéressant d'utiliser des propriétés des sons comme leur vocalité, percussivité, harmonicité, électronicité, bruit, rugosité, etc...De même, il est possible de réaliser une recherche de titres musicaux sur des critères instrumentaux, vocaux, ou perceptifs comme le type de voix, l'émotion, la gaieté, la sensualité, l'énergie, la dansabilité, la légèreté, l'agressivité, la simplicité, l'excitation, etc...Toutes ces propriétés sont potentiellement modélisables par EDS à partir d'une sélection manuelle d'exemples sonores caractéristiques. Le chapitre suivant présente l'intégration de descripteurs générés par EDS dans diverses applications musicales.

PARTIE 2
EXPLOITATION DES DESCRIPTEURS DANS DES APPLICATIONS MUSICALES

6. UTILISATION DES DESCRIPTEURS DANS DES APPLICATIONS MUSICALES

EDS permet de construire de manière automatique des descripteurs pertinents pour toute donnée représentable sous forme d'un signal.

Dans le domaine musical, on s'attachera principalement à décrire trois types de données : les sons (durée de l'ordre de quelques dixièmes de seconde), les passages musicaux (quelques secondes), et les titres entiers (quelques minutes).

Ces objets musicaux peuvent être décrits séparément (recherche directe de sons ou de titres), ou dans un ensemble (séquence de sons, suivi des caractéristiques locales d'un titre, « playlist »). Le schéma 6.1 récapitule les groupes de descripteurs utilisables pour les applications présentées dans ce chapitre.

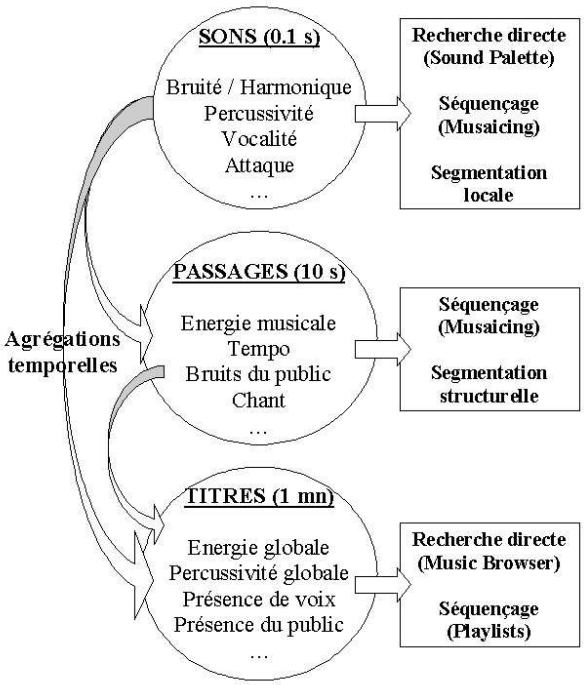


Fig. 6.1: Groupes de descripteurs musicaux et applications associées

Dans ce chapitre , après avoir justifié l'utilisation de ces applications dans

le cadre de la distribution électronique de musique (« Electronic Music Distribution », ou EMD), nous montrerons comment s'assurer de la pertinence des descripteurs musicaux utilisés. Nous présenterons ensuite plus particulièrement l'intégration de ces descripteurs dans une application de gestion de titres musicaux (le « Music Browser », [Pachet *et al.*, 2003], [La Burthe *et al.*, 2003], et [Pachet *et al.*, 2004]), ainsi que dans une application originale de séquençage de sons, que nous avons appelée Musaicing (pour « MUSical moSAICING », [Zils et Pachet, 2001]), et dans une application de séquençage de titres musicaux basée sur la méthode de Musaicing (« Playlists », [Aucouturier et Pachet, 2002b]).

6.1 Contexte général de la distribution électronique de musique

On parle habituellement de distribution électronique de musique (EMD ou Electronic Music Distribution) en termes de méthodes de diffusion (les systèmes peer-to-peer), de compression de fichiers musicaux (le format MP3), ou de gestion des droits d'auteurs. Si bien qu'on semble oublier le problème de fond que pose la quantité de titres potentiellement disponibles en version électronique (10 millions pour la seule musique populaire) : le problème « humain »de la mise en place de moyens d'accès et de recherche de musique intéressants pour les utilisateurs, qui sont rarement de vrais musicologues, mais plutôt des passionnés de musique, avides de découvertes, et prêts à passer du temps à parcourir des catalogues musicaux ([Pachet, 2003]).

6.1.1 L'information musicale en EMD

Bien que la plupart des systèmes actuels d'EMD utilisent uniquement les titres des chansons et les noms des artistes, nous disposons de nombreuses autres informations sur les titres musicaux, sous forme de données culturelles et acoustiques.

Ainsi, certains sites internet proposent un certain nombre de données culturelles en plus des titres et noms d'artistes, comme le contenu des albums ([CDDB, 2004]), ou des informations complémentaires sur les artistes et les chansons ([AllMusicGuide, 2004]). Ces données sont élaborés par des experts musicaux, ou de manière collaborative ([CDDB, 2004]). Cependant ces sites proposent uniquement des méta-données et ne traitent pas les fichiers musicaux eux-mêmes. Certains systèmes créent dorénavant une liaison entre ces données et les fichiers musicaux réels, en proposant des « Browser « (outils de gestion de bases de données), s'appliquant sur les collections musicales personnelles stockées sur disque dur ([MoodLogic, 2004], [MusicBrainz, 2004]).

Enfin, les données acoustiques constituent la source d'information musicale la plus susceptible de décrire la musique de manière générale et universelle, du fait de leur extraction directe à partir du signal audio. Cependant, bien qu'il existe actuellement des descripteurs portant sur le rythme, la mélodie, et le timbre des titres musicaux, ceux-ci n'avaient pas encore été intégrés dans des applications musicales, avant le « Music Browser », conçu pour intégrer facilement des descripteurs acoustiques créés par EDS.

6.1.2 Le projet Cuidado et le Music Browser

La plupart des applications que nous avons réalisées ont été conçues dans le cadre du projet européen IST « Cuidado »([Vinet *et al.*, 2002]), visant à produire, dans le cadre de la distribution électronique de musique, deux applications proposant des moyens de recherche et d'exploration de grandes bases de sons (« Sound Palette »), et de titres musicaux (« Music Browser », [Pachet *et al.*, 2003]).

Nous présenterons plus particulièrement dans ce chapitre l'intégration des descripteurs musicaux dans le « Music Browser », qui propose notamment une recherche de titres musicaux, directe, par similarité, ou par construction de programmes musicaux (« playlists »).

Le Music Browser fournit divers types d'exploration et de recherche de musique dans des grandes bases de titres musicaux : recherche directe (utilisée dans la plupart des applications existantes, qui n'utilisent cependant pas de descripteurs acoustiques), exploration par similarité acoustique ou culturelle, et exploration par génération de programmes musicaux.

Recherche directe

Actuellement, la plupart des systèmes de recherche de musique proposent essentiellement une recherche directe sur des données simples : le titre et le nom de l'artiste.

Cependant, et plus généralement, la recherche directe sur ce type de données éditoriales présente deux inconvénients majeurs. D'une part, ce type de recherche présuppose que l'utilisateur sait à priori ce qu'il cherche, et bloque toute exploration non-aléatoire d'un catalogue musical. D'autre part, les données éditoriales sont habituellement dépendantes d'une taxonomie qui en limite la généralisation. Par exemple, il est quasiment impossible de construire une taxonomie universelle pour définir les genres musicaux ([Aucouturier et Pachet, 2003]).

Ainsi, la recherche directe n'est envisageable que pour des critères universels, comme les critères acoustiques. Dans le Music Browser, la recherche directe sur des critères acoustiques générés par EDS, comme par exemple l'énergie perçue, un type d'instrument ou de voix, permet ainsi l'exploration et la découverte de musiques inconnues, correspondant cependant à des goûts précis.

Recherche par similarité

Dans la perspective de découverte de musiques inconnues, le Music Browser intègre une recherche par similarité qui permet de comparer des titres sans établir d'échelle de valeur. Tous les descripteurs acoustiques peuvent être ainsi utilisés, sous réserve de construire une fonction de distance entre titres (différence des tempos, distance entre gaussiennes pour un modèle de timbre global dans l'espace paramétrique).

L'intégration de techniques de recherche par similarité culturelle, utilisant notamment des mesures de co-occurrence sur internet ([La Burthe *et al.*, 2003]), permet d'apporter également un nouvel intérêt exploratoire aux données culturelles.

Modes de recherche originaux

Enfin, le Music Browser intègre une dernière manière, plus originale, d'explorer des bases de données consiste à générer automatiquement des programmes musicaux (listes de titres, ou « playlists »), en spécifiant uniquement certaines contraintes (comme par exemple un titre de départ et un titre d'arrivée, ou la continuité des valeurs des descripteurs ([Pachet *et al.*, 1999])), et en laissant le système générer automatiquement le programme complet suivant ces contraintes.

6.2 Utilisation des descripteurs musicaux synthétisés par EDS

Comme on l'a vu dans la présentation du système, les descripteurs synthétisés par EDS sont fournis aux applications sous forme d'un programme exécutable calculant l'agrégation d'un modèle local (voir section 4.10).

Ainsi, EDS fournit deux types de descripteurs musicaux : les descripteurs bruts, et les descripteurs dérivés obtenus par diverses agrégations temporelles.

6.2.1 Descripteurs musicaux bruts

Nous avons synthétisé avec EDS des descripteurs bruts pour les sons (comme la percussivité ou la vocalité), et les extraits sonores (comme l'énergie globale, le tempo, ou la reconnaissance de voix).

Pour les titres entiers, bien qu'on puisse en théorie utiliser dans EDS des bases d'apprentissages contenant les signaux complets des titres, en pratique il est beaucoup plus intéressant d'utiliser l'agrégation temporelle de modèle locaux, notamment car :

- la taille et la complexité de ces signaux entraînent des calculs extrêmement lourds peu compatibles avec la synthèse rapide de descripteur ;
- les titres musicaux étant rarement homogènes, la calcul direct d'une valeur descriptive pour l'ensemble d'un titre n'a pas forcément de sens (par exemple, le tempo d'un titre peut évoluer au cours du morceau) ;
- l'évolution des descripteurs locaux contient une grande quantité d'information exploitable de diverses manières par les différentes agrégations temporelles.

Ainsi, dans les applications musicales, seuls les descripteurs de sons et d'extraits sonores sont utilisés de manière brute, ou pour construire des descripteurs globaux par agrégation temporelle.

6.2.2 Descripteurs musicaux dérivés par agrégation temporelle

L'agrégation de descripteurs musicaux bruts pour produire des descripteurs plus globaux peut se faire de diverses façons :

- agrégation de descripteurs de sons pour décrire les extraits musicaux, par exemple :

  - o classification d'instruments : l'agrégation du descripteur local de « note de guitare électrique »donne un descripteur d'extraits de « solo de guitare électrique »
  - o reconnaissance de timbre : l'agrégation du descripteur local de « voix »donne un descripteur d'extraits « chantés »

  Cette agrégation permet principalement de modéliser une prédominance sonore dans l'extrait.
- agrégation de descripteurs de sons pour décrire les titres entiers, par exemple :
  - o classification d'instruments : l'agrégation du descripteur local de « son percussif »donne un descripteur de « percussivité »globale du titre
  - o reconnaissance de timbre : l'agrégation du descripteur local de « voix »donne un descripteur de « présence de voix »globale dans le titre

  Cette agrégation permet principalement de modéliser une ambiance générale du titre.
- agrégation de descripteurs d'extraits musicaux pour décrire les titres entiers, par exemple :
  - o critère de globalité : l'agrégation de la *moyenne* du descripteur local d'« énergie musicale »d'un extrait, donne un descripteur d'« énergie globale »du titre
  - o critère d'hétérogénéité : l'agrégation de la *variance* de ce même descripteur local d'« énergie musicale »d'un extrait, donne un descripteur de la « complexité énergique »du titre
- quelques utilisations alternatives sont également possibles, comme la description de « sons élaborés », par agrégation de modèles de sons simples. On peut ainsi par exemple définir un descripteur de « roulement percussion »par l'agrégation de quelques sons percussifs successifs.

L'utilisation de l'agrégation spécifique retournant la série des valeurs des descripteurs locaux (voir section 4.10.3), ne produit pas réellement de nouveau descripteur finalisé, mais une base pour les applications de suivi (« tracking »), souvent associé à une segmentation réalisée en post-traitement.

6.3 Intégration des descripteurs dans les applications musicales

Nous présentons ici l'utilisation des descripteurs musicaux dans trois grands types d'applications, résumés dans la table 6.1 :

- Recherche / exploration dans des bases d'objets sonores : « Sound Palette »pour les sons, et « Music Browser »pour les titres entiers ;
- Suivi : observation simple, segmentation locale par suivi de descripteurs de sons, ou segmentation structurelle par suivi de descripteurs d'extraits musicaux sur des titres entiers ;
- Mise en séquence : programmes musicaux (« Playlists ») pour les titres, mosaïques musicales (« Musaicing ») pour les sons et éventuellement de courts extraits musicaux. Ces applications originales sont présentées en détails dans la section. 6.4.

<table>
<tr><th></th><th>Description directe</th><th>Suivi (tracking)</th><th>Mise en séquence</th></tr>
<tr><td>Sons</td><td rowspan="3">Exploration de bases d'objets musicaux</td><td>Description d'extraits / Segmentation locale par agrégation de sons</td><td rowspan="2">Mosaïques musicales</td></tr>
<tr><td>Extraits</td><td rowspan="2">Description de titres / Segmentation structurelle par agrégation d'extraits</td></tr>
<tr><td>Titres</td><td>Programmes musicaux (Playlists)</td></tr>
</table>

Tab. 6.1: Applications principales pour les différents types d'objets musicaux

6.3.1 Utilisation de descripteurs pour l'exploration de bases d'objets musicaux

L'application évidente et directe des descripteurs acoustiques synthétisés par EDS est l'exploration de grandes bases d'objets musicaux sur des critères pertinents (voir section 5.1.1). On applique pour cela tous les descripteurs créés, bruts ou agrégés, sur des bases de titres de sons.

Nous présentons ici l'intégration des descripteurs modélisés par EDS dans l'application de gestion de titres, le Music Browser (MB), l'intégration étant similaire dans l'application de gestion de bases de sons Sound Palette.

Les descripteurs générés par EDS sont enregistrés sous la forme d'un programme exécutable indépendant d'EDS nécessitant l'utilisation de Java. Ce programme permet d'extraire la valeur du descripteur sur tout signal acoustique externe.

L'interface du MB, programmée en Java, intègre dynamiquement tout nouveau descripteur dans les différents champs de recherche.

L'intégration d'un descripteur EDS se fait donc très simplement en créant un nouveau champ dans le MB, et en l'associant au programme exécutable du descripteur. Les valeurs du descripteur sont alors calculées sur la base de titres du MB en utilisant cet exécutable, et stockés dans une table de valeurs.

Les valeurs contenues dans cette table sont la référence pour les différents types d'exploration possibles :

- recherche directe : les titres sont recherchés par la valeur brute des descripteurs, par exemple « les titres dont l'énergie musicale est maximale » ;
- recherche par similarité : les titres sont recherchés relativement à un titre de référence, par exemple « les titres dont la percussivité globale est la plus proche d'un titre donné » ;
- génération de programmes musicaux : la succession des titres est contrôlée par les valeurs des descripteurs, par exemple « un ensemble de titres instrumentaux »(voir section 6.4).

La figure 6.2 montre un exemple d'intégration des descripteurs de « Présence de Voix »et d' »Énergie Globale »dans l'interface de recherche de musique du MB.

La gestion de bases de sons peut se faire de manière similaire : recherche directe (« les sons percussifs à 80% »), ou par similarité (« les sons dont le niveau de voix est le plus proche d'un son donné »).

Récemment, grâce au développement du langage MCM (Music Content Management), permettant de manipuler des objets multimedia et des méta-données associées et de les partager entre plusieurs applications, une extension d'EDS a

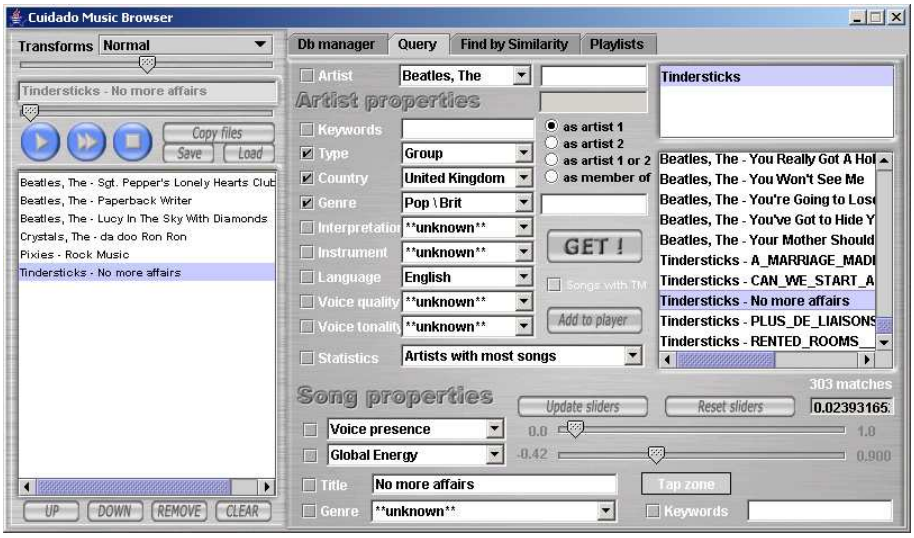


Fig. 6.2: Interface du Music Browser consacrée à la recherche directe de titres musicaux, incluant les descripteurs de Présence de Voix et d'Énergie Globale synthétisés par EDS (dans « Song Properties »)

pu être intégrée dans le Music Browser. La figure 6.3 présente ainsi l'interface de construction automatique d'un descripteur personnel « genreFP_auto », à partir d'une sélection d'exemples musicaux caractéristiques (les titres affichés dans la colonne de droite).

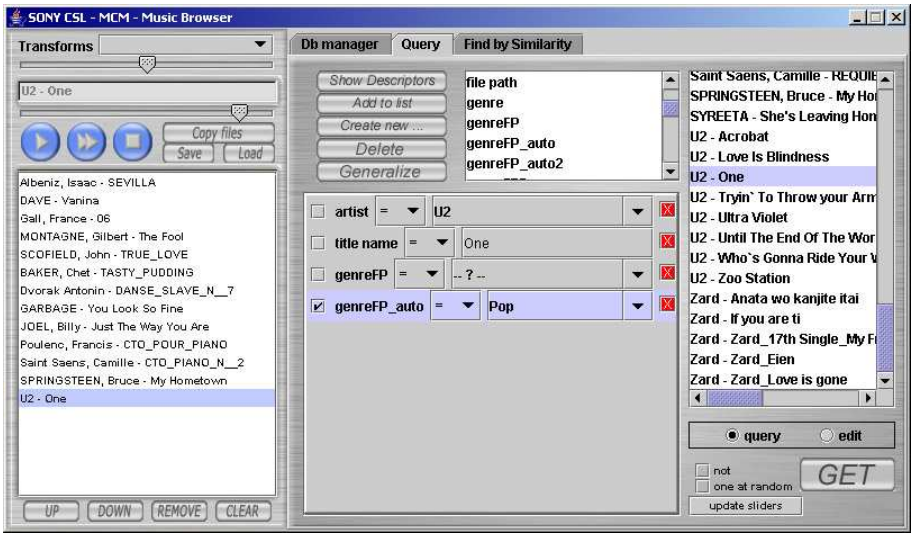


Fig. 6.3: Interface du Music Browser consacrée à la génération automatique de descripteurs personnels à partir d'une sélection de titres caractéristiques

6.3.2 Suivi de descripteur

Le suivi (ou « tracking ») consiste à suivre l'évolution d'un descripteur local le long d'un objet de durée plus importante.

Au delà de l'observation simple, permettant de visualiser les propriétés d'un signal, il est surtout utilisé pour la segmentation, locale ou structurelle.

Segmentation locale

D'un point de vue musical, la segmentation locale est un problème réputé compliqué (voir section 2.3.1), consistant à obtenir des sons homogènes à partir d'un flux sonore continu.

En général, c'est la détection de ruptures caractéristiques lors du suivi d'un descripteur sonore local (harmonicité, attaque, bruit), qui permet de réaliser la segmentation. Cette segmentation est alors dite spécifique, dans la mesure où les sons obtenus sont homogènes relativement à ce descripteur précis.

Segmentation structurelle

La segmentation structurelle, c'est-à-dire l'obtention de passages musicaux homogènes, est équivalente à la segmentation sonore, à une échelle temporelle plus importante.

A nouveau, ce sont les ruptures caractéristiques des valeurs de descripteurs d'extraits musicaux, comme l'énergie musicale, ou la présence de voix ou de guitare, qui délimitent la structure globale d'un titre complet. Par exemple, cette segmentation permet de repérer les passages chantés et instrumentaux dans un titre, ou les différentes scènes sur la bande sonore d'un film.

6.3.3 Utilisation de descripteurs pour la création musicale par séquençage de sons

La mise en séquence de sons utilise le même principe que la mise en séquence de titres pour générer des programmes musicaux : les propriétés de la séquence générée sont contrôlées en utilisant les valeurs des descripteurs.

Cependant, contrairement au séquençage de titres musicaux, qui permet d'explorer de manière originale une grande base musicale, le séquençage de sons est plus proche de la création musicale.

La section suivante 6.4 présente de manière détaillée les principes de mise en séquence d'objets sonores.

6.4 Mise en séquence d'objets sonores

Nous présentons ici une méthode générale de création automatique de séquences d'objets sonores, par spécification de propriétés sur les valeurs de leurs descripteurs. Cette méthode, présentée initialement dans [Zils et Pachet, 2001], a permis le développement de deux applications originales :

- les « Mosaïques musicales »(séquences de sons) : introduites dans [Zils et Pachet, 2001], une version plus perfectionnée est actuellement en développement ;

– les « Programmes musicaux »(séquences de titres) : une méthode basée sur notre système de séquençage, intégrée dans le Music Browser, est présentée dans [Aucouturier et Pachet, 2002b].

6.4.1 Séquence d'objets interdépendants

Les applications classiques présentées précédemment permettent de rechercher un objet sonore précis, par comparaison à d'autres objets, voire à l'ensemble de la base.

Il est même possible de constituer un groupe d'objets de cette manière, en recherchant séparément chaque élément de ce groupe, qui sera par définition indépendant des autres.

La seule manière de décrire le groupe ainsi constitué consiste donc à décrire séparément chacun de ses éléments : c'est une énumération de *propriétés locales*.

Par exemple, un utilisateur peut constituer manuellement un programme musical, en recherchant les dix chansons les plus énergiques des Beatles.

L'utilisation de séquences d'objets sonores permet de considérer en plus des propriétés locales de chaque élément, les relations entre les différents objets de la séquence, appelées *propriétés globales*.

Ainsi, on peut constituer diverses séquences de titres à partir du groupe précédent (les dix chansons les plus énergiques des Beatles) : par exemple en les ordonnant par tempo croissant, ou en alternant régulièrement les titres au piano et les titres à la guitare. Les propriétés globales du programme sont alors définies par des relations entre ses éléments.

6.4.2 Intérêt des applications musicales utilisant des séquences

En utilisant les propriétés des séquences, on peut créer des applications musicales originales, proches de la création artistique, utilisant les sons ou les titres.

En effet, de nombreuses compilations de titres actuelles disponibles dans le commerce sont réalisées par des experts musicaux qui cherchent à créer une atmosphère spécifique.

De même, de nombreuses compositions musicales modernes sont réalisées par assemblage de sons existants (les « samples »), plutôt qu'à partir de sons entièrement originaux. Cette tendance est particulièrement forte dans la communauté de la musique techno, mais s'étend également à d'autres genres musicaux, la plupart des musiciens utilisant désormais des séquenceurs informatiques (Cubase, ProTools) pour réaliser leurs compositions.

Dans les deux cas, la composition de séquences musicales à la main est une tâche de création comprenant deux étapes longues et complexes : la recherche d'objets musicaux, et la construction d'une séquence intéressante.

L'idée générale de combiner la recherche et génération de séquence a été introduite par [Pachet *et al.*, 1999], dans le contexte des séquences de titres musicaux.

Recherche d'objets musicaux dans des grandes bases

Il est extrêmement difficile de retrouver des objets spécifiques dans une grande base sans une connaissance parfaite du contenu de cette dernière.

Dans le cas des titres, l'utilisation d'informations éditoriales (nom de l'artiste, titre de la chanson) permet d'aider le programmeur, mais uniquement sur les titres qu'il connait. L'utilisation de critères acoustiques (comme les descripteurs synthétisés par EDS) en plus des informations éditoriales, bien qu'elle lui rende cette tâche beaucoup plus facile, ne le dispense cependant pas de rechercher chacun des titres séparément.

La recherche de sons est encore plus complexe, dans la mesure où aucune information éditoriale n'est disponible sur ces derniers, et où leur classification est toujours très subjective, les séquenceurs offrant peu de moyens pour retrouver des sons de manière efficace.

Souvent, les sons bruts n'existent même pas dans une base, et le compositeur doit les localiser manuellement dans un extrait musical, les découper et les recopier manuellement. C'est alors la segmentation automatique qui va permettre d'éviter ce travail considérable.

Pertinence de la séquence réalisée

La construction d'une séquence musicale *intéressante* nécessite une forte intuition musicale, la pertinence des séquences composées reposant sur le respect de contraintes de composition, souvent difficiles à mettre en oeuvre.

Ainsi, les sons ou les titres successifs sont très dépendants les uns des autres, et ne peuvent pas être pris en compte de manière isolée : ils doivent satisfaire à des propriétés diverses entre eux, afin d'être intégrés harmonieusement dans la séquence.

Par exemple, un programmateur musical doit choisir précisément l'ordre des titres d'une compilation pour produire une atmosphère précise. De même, un compositeur qui veut produire une continuité de timbre entre les sons successifs de sa séquence, ou produire une mélodie précise, doit résoudre ces contraintes manuellement.

6.4.3 Mise en place d'un système général de mise en séquence

Nous avons créé un système qui propose de résoudre de manière automatique la mise en séquence d'objets en utilisant les valeurs de leurs descripteurs. Pour fonctionner automatiquement, le système doit être capable de gérer simultanément les deux problèmes présentés précédemment : la spécification de propriétés pertinentes de la séquence, et la recherche efficace d'objets dans de grandes bases.

Spécification des propriétés de la séquence

Afin de définir une séquence, on peut distinguer deux types de propriétés, qu'on peut en général exprimer en termes descriptifs, c'est-à-dire sous forme de descripteurs spécifiques des objets et de leur évolution le long de la séquence :

- propriété locale : propriété d'un objet précis dans la séquence ; elle peut être traduite sous la forme d'une valeur préconisée pour un descripteur de cet objet ;

– propriété globale : relation entre plusieurs objets de la séquence ; elle peut être traduite sous la forme d'une relation entre les valeurs de descripteurs de ces objets.

Ainsi, de manière générale, les propriétés de la séquence sont exprimables sous forme de contraintes sur les valeurs des descripteurs des objets qui la composent. A partir d'une palette de contraintes pertinente, il est alors possible de spécifier au système des propriétés de haut-niveau de la séquence à générer, assurant l'intérêt de cette dernière.

Recherche des objets et construction de la séquence

A partir de ces spécifications précises, le système recherche alors de manière automatique les objets de la base qui leur satisfont le mieux, en utilisant leurs valeurs perceptives. La construction de la séquence revient alors à un problème de résolution de contraintes.

Afin d'assurer l'efficacité de la recherche d'objets dans des grandes bases, nous utilisons pour résoudre les contraintes un algorithme de recherche locale, évitant ainsi une recherche complète impossible à réaliser.

6.4.4 Présentation détaillée du système « Musaicing »

Le système de séquençage présenté précédemment est un système universel applicable sur tout groupe d'objets associés à des descripteurs.

Pratiquement, nous avons utilisé ce système principalement pour réaliser des séquences de sons, que nous avons appelées « mosaïques musicales ».

Le système, alors appelé spécifiquement « Musaicing »(pour « MUSical mosAICING »), est un outil de Composition Assistée par Ordinateur (CAO) permettant aux musiciens de créer des compositions originales de manière simple et efficace ([Zils et Pachet, 2001]).

L'idée de base des mosaïques musicales est de reconstituer un extrait musical existant, appelé « cible », en utilisant des sons ne provenant pas de cet extrait. Cette idée a été inspirée de l'analogie avec les mosaïques d'images, consistant à reconstituer une grande image par collage d'un grand nombre d'images plus petites (voir les photo-mosaïques de [Silvers, 2004], avec l'exemple célèbre de la photo de Lady Diana reconstituée à partir de milliers de petites photos de fleurs). Cette approche exploite notre capacité à synthétiser une perception au niveau macroscopique d'une image, à partir de la perception cumulée d'objets au niveau microscopique. Les mosaïques images prouvent que cette propriété est vraie au niveau visuel. L'hypothèse du Musaicing est qu'elle est également vraie au niveau sonore, c'est-à-dire que la perception globale d'une mélodie peut être reconstituée à partir de la suite de perceptions locale générée par une séquence de sons.

Par exemple, on peut imaginer créer une mosaïque musicale d'une chanson des Beatles, à partir de sons extraits des tubes rock des années 60. Cette approche fournit alors deux niveaux d'écoute : une écoute distante qui « sonne »comme la chanson originale des Beatles, et une écoute proche (ou attentive) qui révèle les différents extraits sonores des titres des années 60 qui composent la mosaïque.

Ce principe de composer de la musique sous forme de séquences de sons a été utilisé par le compositeur John Oswald dans son album Plunderphonics ([Os-

wald, 1989]), qui a réalisé manuellement la lourde tâche de repérer, découper, classer, et recoller les sons dans des séquences musicales. Le Musaicing est une généralisation de cette approche, qui propose de gérer de manière automatique les propriétés de la séquence, sous forme de contraintes sur les caractéristiques des sons.

Bien que l'utilisation des contraintes en CAO soit courante ([Truchet *et al.*, 2001]), la notion de séquences de sons contrôlées par des contraintes a été introduite par [Schwarz, 2000], qui propose un système qui gère les contraintes de continuité sur les sons successifs (le système complet est présenté dans [Schwarz, 2004]). Dans le Musaicing, nous avons introduit de nombreuses contraintes supplémentaires, comme la cardinalité, la différence, ou la distribution, indispensables pour spécifier des propriétés intéressantes de la composition à réaliser.

Mécanisme général

Pour construire une mosaïque image, l'image originale est divisée en petites régions, homogènes suivant des paramètres visuels comme la couleur moyenne ou la luminosité. Puis chaque petite région est remplacée par une nouvelle image dont les paramètres sont proches, dont l'assemblage final fournit la mosaïque.

Pour construire une mosaïque musicale, le principe est similaire : l'extrait sonore initial (la « cible ») est découpé en sons homogènes (ou « segments ») par segmentation, on obtient ainsi une « séquence cible », constituée d'une suite de « segments cibles ». Chaque *segment cible* est ensuite décrit à l'aide de paramètres musicaux comme sa fréquence fondamentale ou son niveau sonore. Enfin, chaque *segment cible* est remplacé par un son (ou « segment ») dont les paramètres sont proches, et la séquence des segments trouvés constitue la mosaïque finale.

De manière plus générale, on peut introduire une notion plus élaborée de « plus proche segment », qui doit non seulement avoir des caractéristiques proches du *segment cible*, mais également être cohérent vis-à-vis des autres segments de la mosaïque. Ainsi, la *séquence cible* peut être définie par un ensemble de « propriétés locales »sur chaque *segment cible*, et de « propriétés globales »sur l'ensemble de la *séquence cible*, comme par exemple l'interdiction d'utiliser deux fois le même son dans la mosaïque.

Ce principe permet alors de construire des mosaïques classiques à partir d'un extrait *cible*, ou originales en définissant uniquement des propriétés globales, comme par exemple « une séquence dont la note fondamentale monte progressivement et régulièrement de Do à Mi ».

En traduisant ces propriétés *locales* et *globales* sous forme de contraintes, la construction de la mosaïque revient à trouver la séquence qui satisfait au mieux l'ensemble des contraintes.

Définition et résolution d'un problème de satisfaction de contraintes

Typiquement un problème de satisfaction de contraintes est défini par un ensemble de variables, dont les valeurs possibles sont contenues dans un domaine fini, et un ensemble de contraintes, c'est-à-dire d'informations sur les valeurs des variables qu'on veut obtenir. La résolution du problème consiste à trouver les valeurs des variables qui « satisfont »le mieux à l'ensemble des contraintes.

Pour résoudre le problème, on introduit une fonction de « coût »mesurant la « non-satisfaction »des variables à l'ensemble des contraintes, et on cherche les valeurs des variables qui permettent de minimiser cette fonction. Deux types de méthodes permettent de réaliser cette optimisation :

- complètes : reposant sur une exploration systématique de tout l'espace de recherche (back-tracking, arc-consistency, forward-checking) ; elles assurent la découverte de la solution optimale, mais nécessitent le parcours de l'ensemble de l'espace pour la valider, ce qui est extrêmement coûteux en temps de calcul ;
- incomplètes : reposant sur une amélioration itérative des solutions trouvées (tabu search, recuit simulé) ; elles peuvent trouver rapidement des solutions localement optimales, mais ne garantissent pas la découverte de la meilleure solution.

Problème de satisfaction de contraintes d'une mosaïque

Dans le cas des mosaïques, on définit les différents éléments du problème :

- variables : les sons (« segments ») constituant la séquence ;
- domaine : l'ensemble des sons contenus dans la base ;
- contraintes : les propriétés locales et globales de la séquence (récapitulées sur la figure 6.4).

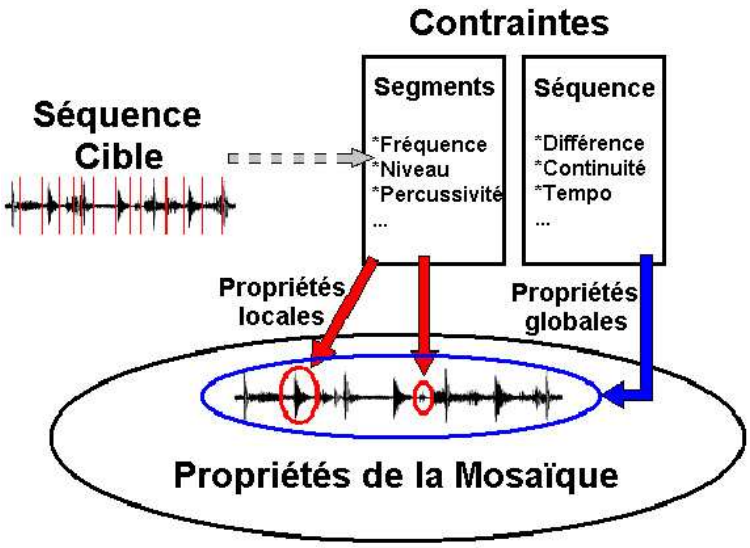


Fig. 6.4: Traduction des propriétés de la mosaïque sous forme de contraintes locales et globales

Les propriétés locales sont exprimées sous forme de contraintes sur un segment, chaque segment étant défini par les valeurs de ses descripteurs. Typiquement, les descripteurs utilisés pour spécifier les contraintes sont les descripteurs acoustiques traditionnels : durée, fréquence fondamentale, niveau sonore, niveau de bruit, harmonicité, vocalité, ...

Une contrainte locale consiste donc à imposer dans un segment donné, une valeur précise pour un descripteur précis, par exemple : « le premier segment

doit avoir une fréquence fondamentale de 400Hz ».

La spécification d'une *séquence cible* permet de créer automatiquement un ensemble de contraintes locales permettant de recréer la cible. En effet, cette dernière est segmentée en sons homogènes, pour lesquels on calcule les valeurs des descripteurs acoustiques. Chaque valeur de descripteur obtenue est alors traduite sous forme d'une contrainte locale sur son segment.

Les propriétés globales sont exprimées sous forme de contraintes sur un groupe de segments (voire sur l'ensemble des segments de la séquence). Elle permettent notamment de contrôler :

- la variété des sons : contraintes de différence (all-Difference, au moins 5 sons différents, ...) ;
- les transitions entre sons : contraintes de continuité (de fréquence, de position du son dans l'extrait d'origine, ...) ;
- la répartition des sons : contraintes de distribution (tempo de 100 bpm, fréquence suivant une répartition gaussienne autour de 440Hz) ;
- les variations des sons : contraintes d'évolution (augmentation progressive du niveau sonore).

Résolution d'un problème de mosaïques musicales

La résolution de la mosaïque consiste à trouver dans la base les sons qui permettent de satisfaire au mieux l'ensemble des contraintes locales et globales.

Or, contrairement aux contraintes locales, limitées par le nombre de segments et de descripteurs, les contraintes globales n'ont pas de restriction, et peuvent être aussi nombreuses que nécessaire. Par conséquent, souvent, les problèmes de mosaïques obtenus sont sur-contraints (les propriétés ne peuvent pas être toutes vérifiées simultanément), et il n'existe pas de solution parfaite au problème. Il peut alors exister une grande variété de solutions approximatives, qui sont toutes potentiellement intéressantes.

De plus, la taille des bases utilisées pour la construction des séquences, de l'ordre de centaines de milliers de sons, entraîne une explosion combinatoire qui ne permet pas de tester l'ensemble des configurations possibles. En effet, pour construire une séquence de seulement 10 sons à partir d'une base de seulement 10000 sons, le nombre de configurations possibles est 10000^{10}. Il est donc impossible d'utiliser une méthode de recherche complète.

Afin de gérer ces deux problèmes, nous avons choisi une méthode de résolution incomplète, qui permet en outre d'obtenir rapidement des solutions intermédiaires variées sur les problèmes sur-contraints : l'algorithme itératif de recherche locale « Adaptive Search »([Codognet et Diaz, 2001]). [Truchet *et al.*, 2001] a montré que cet algorithme est particulièrement bien adapté à la résolution de problèmes musicaux, pour lesquels les approximations présentent souvent un grand intérêt créatif.

L'algorithme permet de résoudre le problème, en cherchant les valeurs des variables qui permettent de minimiser la « fonction de coût »globale mesurant la satisfaction de l'ensemble des contraintes. Basé sur le « Tabu Search »([Glover *et al.*, 1993]), il fonctionne de manière itérative, en modifiant la valeur de la variable qui contribue le plus à la fonction de coût. Son fonctionnement détaillé est expliqué sur la figure 6.5 :

1. Construction d'une séquence initiale, aléatoirement, ou en choisissant les segments qui satisfont le mieux les contraintes locales

2. Calcul du coût global de la séquence, et des coûts locaux de chaque segment
3. On considère le segment S dont le coût est le plus élevé
4. Recherche dans la base du segment de remplacement de S, qui permet de diminuer le plus possible le coût global de la séquence. Si aucun changement ne permet de diminuer le coût, le segment est marqué « Tabu », c'est-à-dire qu'il ne pourra plus être remplacé pendant un certain nombre d'itérations
5. Itération de l'algorithme en 2, jusqu'à l'obtention d'une solution, ou l'atteinte d'un nombre maximal d'itérations

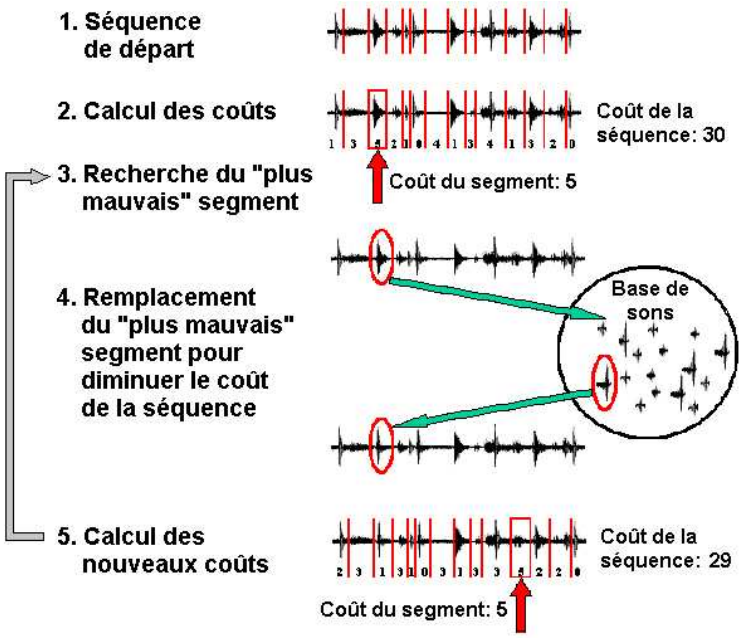


Fig. 6.5: Fonctionnement de l'algorithme de « Local Search »appliqué à la résolution de mosaïques musicales

Calculs des coûts de segments et de séquence

Le fonctionnement de l'algorithme repose sur la définition de fonctions de « coût »mesurant la satisfaction des contraintes à deux niveaux :

- coût local d'un segment : c'est la fonction qui indique le « plus mauvais »segment ;
- coût global de la séquence : c'est la fonction à minimiser pour résoudre le problème.

Pour calculer le coût local d'un segment, il est nécessaire d'associer à chaque contrainte une fonction de « coût pour une variable », mesurant la contribution de la variable à sa satisfaction, ce coût valant par définition 0 pour les variables sur lesquelles la contrainte ne s'applique pas.

Par exemple, la fonction de coût pour la contrainte locale de niveau sonore d'un segment est définie ainsi :

$$coutLocal_{niveau}(segment) = \frac{|niveau_{segment} - niveau_{reference}|}{max_{base}(|niveau|)}$$

Pour les contraintes globales, portant sur plusieurs variables, voire la totalité de la séquence, on définit le coût local est une projection du coût de la contrainte sur une variable. Par exemple, la fonction de coût pour la contrainte globale « all-difference », est définie ainsi pour un segment :

$$coutLocal_{allDiff}(segment) = \frac{\sum_{s \in sequence} egal(segment, s) - 1}{\sum_{s \in sequence} -1}$$

avec $egal(s_1, s_2) = \begin{cases} 1, s_1 = s_2 \\ 0, sinon \end{cases}$

Ensuite, à partir des fonctions de coût local pour chaque contrainte, on peut facilement définir la fonction de coût global d'un segment, qui mesure la satisfaction d'une variable à l'ensemble des contraintes qui s'appliquent sur elle :

$$coutGlobal(segment) = \sum_{c \in contraintes} coutLocal(c, segment) * poids(c)$$

Le fonction « poids »d'une contrainte permet d'attribuer plus d'importance à une fonction. Par défaut, toutes les contraintes ont la même importance, soit un poids de 1.

Cette fonction de coût global d'un segment est calculée dans l'étape 2 pour chaque élément de la séquence, et le segment pour lequel elle est la plus élevée est marqué comme « plus mauvais ».

Enfin, à partir des fonctions de coût global pour chaque segment, on peut facilement définir la fonction de coût global de la séquence, qui mesure la satisfaction d'une séquence à l'ensemble des contraintes :

$$coutGlobal = \sum_{s \in segments} coutGlobal(s)$$

Dans l'étape 2, cette fonction est calculée pour chaque segment de la séquence, et le segment pour lequel elle est la plus élevée est marquée comme « plus mauvais »segment.

C'est la fonction globale à minimiser pour résoudre le problème, lorsqu'on cherche une variable de remplacement à l'étape 4.

Synthèse sonore de la mosaïque

Les segments trouvés finalement par l'algorithme constituent la séquence de sons qui sera utilisée pour construire la mosaïque.

La mosaïque peut alors être synthétisée par un collage simple de ces sons les uns à la suite des autres. Cependant, il est également possible d'appliquer des post-traitements sur les sons choisis afin d'améliorer la sonorité de la mosaïque, sans en changer le contenu. Par exemple, afin d'adoucir des transitions trop brutales entre sons successifs, on peut réaliser un fondu enchaîné, ou filtrer légèrement la séquence dans les zones de transition ; on peut également compresser le niveau de la séquence finale afin de réduire les écarts de volume, etc...

La figure 6.6 présente l'interface actuelle du Musaicing et un ensemble de contraintes applicables sur les séquences.

6.4.5 Exemples de génération de mosaïques musicales

Nous présentons ici divers exemples montrant le fonctionnement et l'intérêt des résultats du Musaicing.

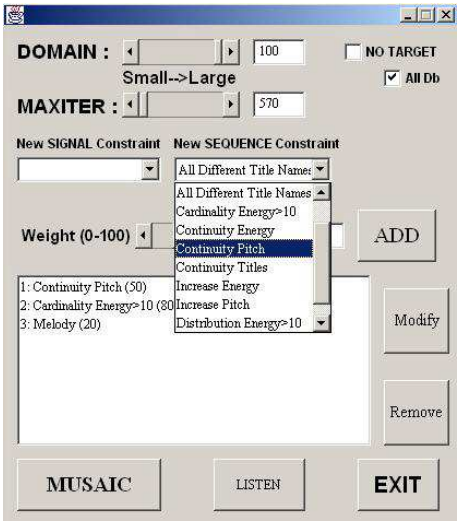


Fig. 6.6: Interface de l'application « Musaicing »

Construction d'une mosaïque simple : contraintes locales

Cet exemple simple montre clairement le fonctionnement de l'algorithme, et la façon dont il prend en compte les différentes contraintes et leur poids respectif.

Son principe consiste à construire une mosaïque ayant les propriétés suivantes :

- elle doit reproduire un extrait musical cible, caractérisé par une fréquence fondamentale qui *augmente* de manière continue et régulière de 0 à 1000 Hertz ;
- la base de sons utilisée contient une centaine de sons extraits de chansons populaires, ainsi qu'un groupe de segments sonores provenant d'un extrait synthétisé de la même façon que l'extrait cible, mais dont la fréquence fondamentale *diminue* de manière continue et régulière de 1000 à 0 Hertz ;
- l'application de contraintes variées montre les différents comportements du système.

On utilise tout d'abord le Musaicing dans sa forme basique : la reconstitution simple de l'extrait cible. Celui-ci est traduit automatiquement par le système, sous forme d'un ensemble de contraintes locales, après une segmentation et le calcul d'un descripteur de fréquence fondamentale sur chacun des segments.

On lance ensuite la résolution des contraintes, qui converge rapidement vers une séquence optimale, dont on peut observer la fréquence instantanée à gauche sur la figure 6.7.

On observe bien une fréquence qui augmente progressivement de 0 à 1000 Hz, mais avec des petites oscillations autour de la valeur moyenne.

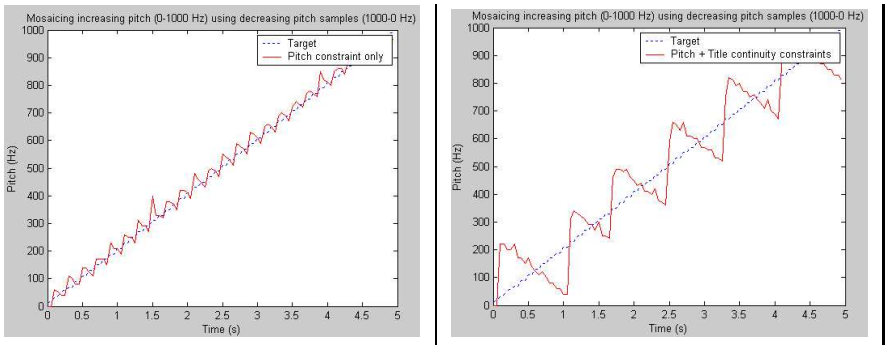
En effet, par définition de l'extrait cible, les segments cibles sont des sons brefs dont la fréquence fondamentale *diminue*. Le système va alors chercher les sons les plus proches, dans la base qui contient des sons dont la fréquence *augmente*. La convergence rapide est due au fait qu'en absence de contrainte globale, une fois que l'algorithme a trouvé pour chaque segment ciblé le son de la base dont la fréquence moyenne est la plus proche, la mosaïque est optimale.

Le résultat est donc une séquence dont la fréquence globale augmente, mais dont chaque son a une fréquence locale qui diminue légèrement, d'où cet effet d'évolution en légères dents de scie.

Effet d'une contrainte globale : continuité

On ajoute maintenant en plus des contraintes locales de fréquence, une contrainte globale de continuité fréquentielle, afin que les segments successifs de la mosaïque proviennent dans la mesure du possible de segments contigus dans la base.

Le résultat obtenu est une mosaïque dont on peut observer la fréquence instantanée à droite sur la figure 6.7.



Contraintes locales de fréquences

Contraintes locales de fréquences + Contrainte globale de continuité

Fig. 6.7: Effet de la contrainte globale de continuité sur la fréquence instantanée de la mosaïque d'un extrait de fréquence ascendante, constituée de sons de fréquence descendante

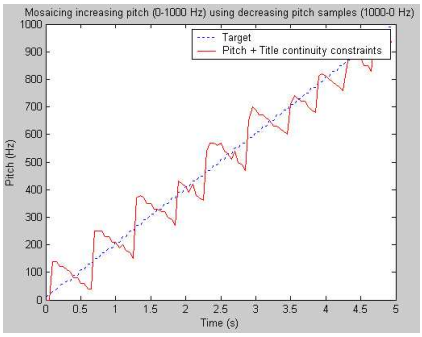
On observe toujours bien une fréquence qui augmente progressivement de 0 à 1000 Hz, mais avec des oscillations plus importantes autour de la valeur moyenne. En effet, la contrainte de la continuité des segments va à l'encontre de celle de respect de la fréquence globale ascendante, les segments continus dans la base ayant une fréquence descendante.

Ainsi, ne pouvant respecter simultanément toutes les contraintes, le système va alors chercher le meilleur compromis, qui est donc de conserver la tendance globale d'augmentation de la fréquence (contrainte de fréquence), tout en utilisant localement plusieurs segments successifs dont la fréquence diminue (contrainte de continuité). On obtient donc cette évolution de la fréquence en dents de scie d'amplitude plus importante.

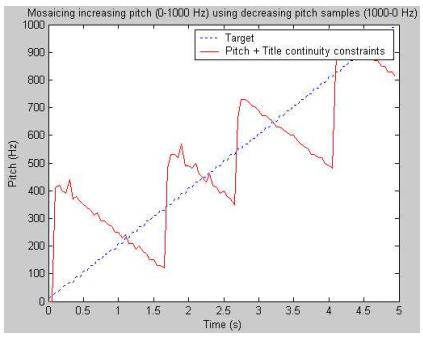
Effet de la pondération des contraintes

Il est possible de contrôler l'importance des contraintes en jouant sur leur pondération. On recommence l'expérience précédente (contraintes locales de fréquence + contrainte globale de continuité), en appliquant deux pondérations

différentes sur la contrainte de continuité : 10% et 100%. La figure 6.8 présente les courbes de fréquence instantanée des mosaïques obtenues.



Contraintes locales
de fréquences
+ Contrainte globale
de continuité (poids 10%)



Contraintes locales
de fréquences
+ Contrainte globale
de continuité (poids 100%)

Fig. 6.8: Effet de la pondération de la contrainte globale de continuité sur la fréquence instantanée de la mosaïque d'un extrait de fréquence ascendante, constituée de sons de fréquence descendante

On observe que l'augmentation de la pondération entraîne une augmentation de l'amplitude des oscillations fréquentielles.

En effet, avec une contrainte de continuité de poids faible, l'algorithme va résoudre en priorité les contraintes de fréquence, d'où une mosaïque plus proche de l'extrait cible. Inversement, une contrainte de continuité de poids fort sera prioritaire, et contiendra plus de segments successifs de la base, d'où ces longs passages de fréquences descendantes.

En conclusion, le système essaye de résoudre simultanément au mieux un ensemble de contraintes conflictuelles en cherchant un équilibre optimal, que l'utilisateur peut contrôler en ajustant les poids relatifs des contraintes.

Contrainte globale élaborée : tempo percussif

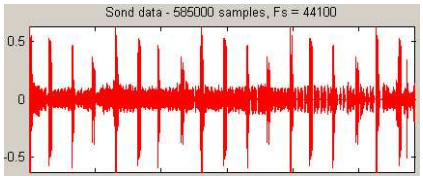
Le tempo percussif est une contrainte globale qui vise à placer de manière régulière des sons percussifs le long de la séquence. Elle utilise donc les propriétés locales des sons (leur percussivité) afin de les répartir de manière globale (distribution régulière).

En spécifiant uniquement une contrainte de tempo percussif, on obtient des séquences dont les signaux montrent de manière évidente les tempos obtenus (figure 6.9).

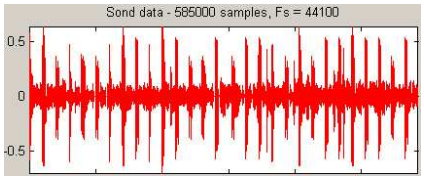
On peut ainsi imposer simplement une structure rythmique à la mosaïque, en combinant propriétés locales et globales.

Combinaison de contraintes locales et globales multiples

Ce dernier exemple montre qu'on peut construire automatiquement des séquences rythmiquement et mélodiquement complexes, à partir d'une spécification simple de contraintes globales avec des paramètres variés :



Contrainte globale de tempo percussif à 80 bpm



Contrainte globale de tempo percussif à 130 bpm

Fig. 6.9: Visualisation du résultat de l'application d'une contrainte de tempo percussif avec deux paramètres différents sur le signal de la mosaïque

- deux contraintes globales d'évolution : la fréquence fondamentale augmente jusqu'en milieu de séquence, puis diminue ensuite ;
- deux contraintes globales de tempo percussif, portant sur des sons aux caractéristiques différentes : les sons proches d'une grosse caisse doivent respecter un tempo élevé de 132, et les sons proches d'une caisse claire un tempo bas de 88.

Le descripteur utilisé pour distinguer les deux types de sons de batterie est le taux de passage à zéro (ZCR) : comme cela est montré dans [Gouyon *et al.*, 2000], les sons proches d'une caisse claire ont un ZCR élevé, et les sons proches d'une grosse caisse un ZCR bas.

De plus, les deux tempos sont dans un rapport de deux-tiers, ce qui implique qu'un son de caisse claire sur trois sera en conflit avec un son de grosse caisse.

La résolution de la mosaïque fournit une séquence de sons variés, avec un rythme complexe en contre-temps, suivant bien une mélodie de notes croissantes puis décroissantes. On peut observer en haut de la figure 6.10 le signal de cette mosaïque. Cependant, bien qu'on puisse deviner une structure assez régulière, ce dernier est trop complexe pour être visualisé directement.

Nous utilisons donc deux filtrages préalables afin d'observer les différents composants de la mosaïque : un passe-bas à 500Hz, et un passe-haut à 1000Hz. Les deux signaux filtrés correspondants sont observables au milieu (basses fréquences) et en bas (hautes fréquences) de la même figure 6.10.

Sur la partie filtrée par un passe-bas, on observe une succession de pics de percussion réguliers, dont le tempo est évalué à 131. Ces pics correspondent donc bien au tempo percussif des sons de grosse caisse, qui ont une forte composante dans les basses fréquences. De même, sur la partie filtrée par un passe-haut, on observe une succession de pics de percussion réguliers dont le tempo est évalué à 87, correspondant au tempo percussif des sons de caisse claire, qui ont eux une forte composante en hautes fréquences. En comparant les positions temporelles des pics pour les deux tempos percussifs, on note que les pics du signal en hautes fréquences sont légèrement décalés par rapport à ceux du signal en basses fréquences. On en déduit que le système a donc résolu le conflit entre les deux types de sons percussifs (un son de caisse claire tous les deux sons de grosse caisse) en introduisant un déphasage rythmique entre les deux tempo, d'où la perception de contre-temps à l'écoute de la mosaïque.

Enfin, on vérifie que la contrainte de fréquence est bien respectée en observant l'évolution de la fréquence instantanée de la mosaïque (figure 6.11).

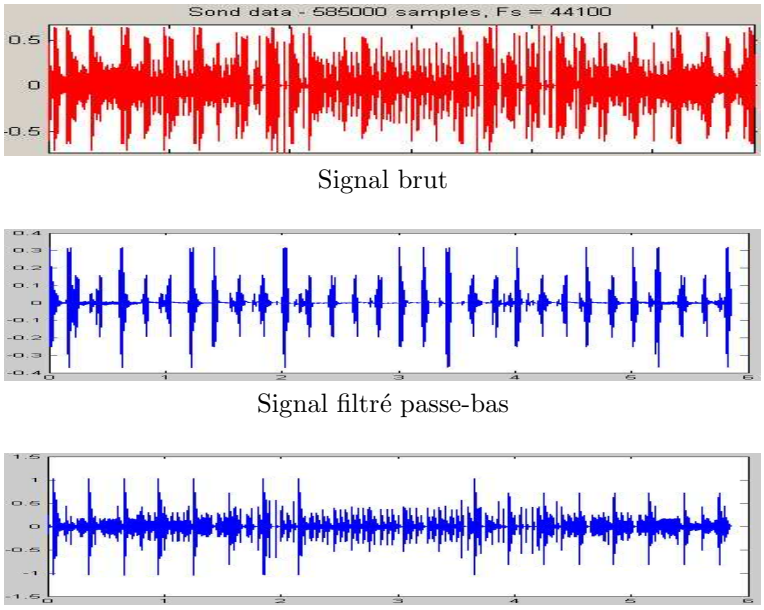


Fig. 6.10: Visualisation du signal brut d'une mosaïque défini par deux contraintes de tempo percussif en basse fréquence et en haute fréquence - Observation du résultat de ces contraintes sur le signal filtré

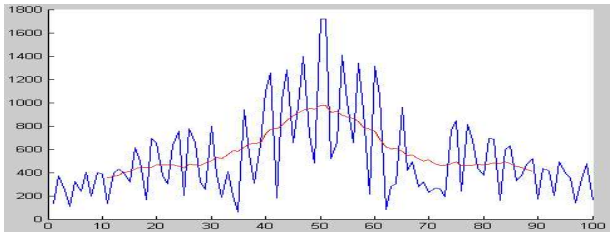


Fig. 6.11: Evolution de la fréquence instantanée dans une mosaïque définie par deux contraintes de tempo percussif, et d'une contrainte globale de « montée puis descente »de la fréquence fondamentale le long de la séquence

L'évolution respecte bien globalement les deux contraintes d'augmentation puis de diminution de la fréquence. On observe cependant localement de nombreuses irrégularités, sous forme de variations de la fréquence fortes et très rapides. Celles-ci sont dues en fait à l'occurence de sons percussifs, dont la fréquence n'est pas réellement significative.

Cependant, cette dernière est prise en compte par l'algorithme, et entre en conflit avec la contrainte de tempo percussif en basses fréquences. En effet, le tempo élevé pour les sons de grosse caisse entraîne un nombre important de sons en basses fréquences tout le long de la séquence. Lorsqu'on arrive au milieu de la séquence, la contrainte de fréquence impose un maximum de fréquence que

les sons de grosse caisse ne peuvent pas fournir. C'est pourquoi on observe une plus forte amplitude des variations de fréquence vers le milieu de la séquence : l'algorithme a donc du trouver un compromis entre les deux contraintes. Comme on l'a vu, il est possible de contrôler les caractéristiques de ce compromis en assignant des poids différents à chacune des contraintes.

En conclusion, ces exemples montrent comment la combinaison de contraintes simples permet d'obtenir des contrôles efficaces sur la composition de séquences musicales complexes.

6.4.6 Génération de programmes musicaux

En appliquant le système général de séquençage du Musaicing sur les titres musicaux, nous obtenons une deuxième application musicale : la génération automatique de programmes musicaux (playlists). La construction automatique d'un programme musical intéressant est un problème délicat, abordé par [Pachet *et al.*, 1999] dans l'application « Recital Composer ». Quelques applications musicales actuelles proposent une génération de playlists (MoodLogic, I-tunes), mais celles-ci restent construites aléatoirement, ou à partir de propriétés éditoriales simples.

La spécification du programme sous forme de propriétés locales (sur une propriété d'un titre précis) et globales (sur le programme en général), permet d'utiliser notre algorithme de résolution de séquences par satisfaction de contraintes. Les contraintes locales sont alors liées aux descripteurs spécifiques aux titres (artiste, genre, vocalité, énergie, etc...), comme par exemple « tous les titres doivent avoir une énergie de moins de 50% »ou « le premier et le dernier titre doivent être des instrumentaux ». Les contraintes globales générales utilisées dans le Musaicing (continuité, distribution, cardinalité, ...), sont toujours utilisables pour la conception de programmes musicaux, comme par exemple :

- différence de genre : tous les titres doivent être d'un genre différent ;
- continuité de timbre : les titres successifs doivent avoir une sonorité proche ;
- cardinalité : « au moins 10% des titres doivent être instrumentaux » ;
- distribution énergétique : concentration de titres énergiques en milieu de programme ;
- évolution vocale : présence vocale de plus en plus importante, etc...

Cette application a été intégrée dans le « Music Browser » : la figure 6.12 montre le résultat d'une génération de playlist à partir de quatre contraintes, spécifiées simplement dans l'interface de la figure 6.13.

6.5 Conclusion - Extension aux objets multimedia

Nous avons présenté dans ce chapitre un ensemble d'utilisations possibles des descripteurs musicaux dans des applications musicales.

Cependant, toutes ces applications sont basées non sur le type d'objet traité, mais sur un ensemble de descripteurs spécifiques de sa représentation sous forme d'un signal. Ainsi, elles donc sont adaptables à l'exploration de toute base d'objets pouvant être représentés sous forme d'un signal quelconque : recherche directe, segmentation, ou séquençage (le système de séquençage utilisé pour le Musaicing étant un système général, comme on l'a vu dans la section 6.4.3).

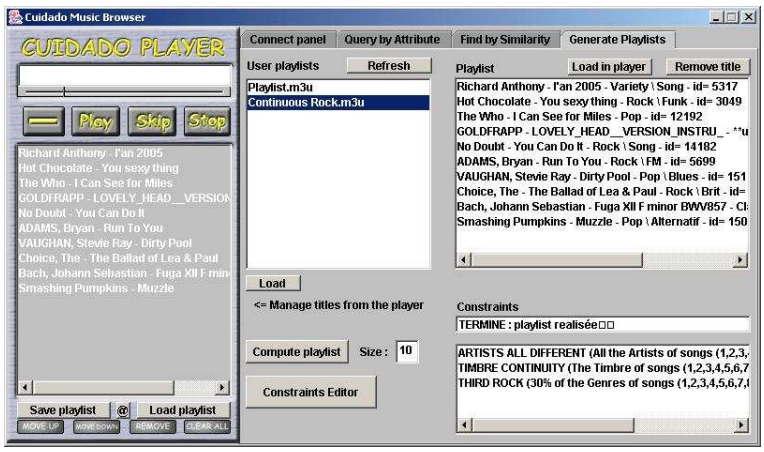


Fig. 6.12: Interface du Music Browser consacrée à la génération automatique de programmes musicaux - Exemple de programme généré à l'aide de quatre contraintes globales

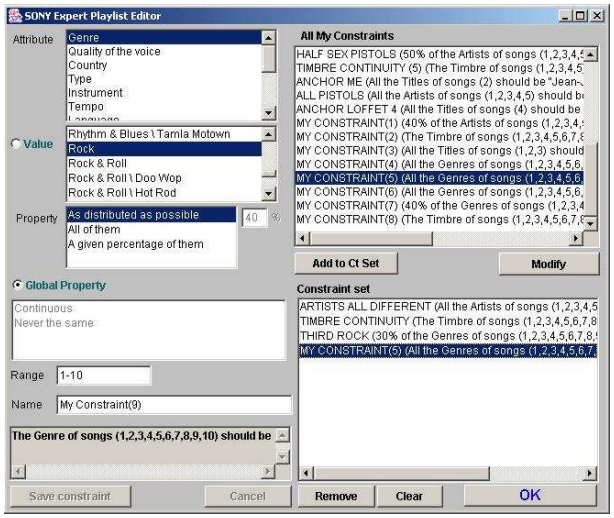


Fig. 6.13: Interface du Music Browser consacrée à la spécification des propriétés de programmes musicaux en vue de leur génération automatique

Les ouvertures les plus évidentes concernent l'ensemble des objets multimédia, principalement les images (recherche à partir de descripteurs perceptifs personnalisés synthétisés avec EDS, séquençage sous forme d'un diaporama), et les videos (repérage de scènes dans un flux vidéo par segmentation structurelle des bandes sonores et des images, création de films expérimentaux par séquençage d'extraits de films).

7. CONCLUSION ET PERSPECTIVES

Synthèse

L'explosion de la distribution informatique de documents multimedia est le catalyseur de nouvelles recherches visant à donner aux utilisateurs les moyens de gérer, rechercher, traiter des bases de documents de plus en plus gigantesques.

Dans le domaine musical, devant l'impossibilité de connaître parfaitement le contenu de dizaines de milliers de sons ou de titres, la gestion de ces bases énormes passe forcément par l'utilisation de descripteurs traduisant des propriétés sonores pertinentes. De nombreuses recherches récentes sur l'extraction de ces propriétés à partir du signal acoustique visent à réaliser cette tâche automatiquement.

Après avoir constaté que la plupart de ces recherches mettaient en oeuvre des techniques proches pour modéliser des propriétés diverses, nous nous sommes intéressés à créer un système général permettant de générer automatiquement tout type de descripteur à partir d'une base d'exemples.

Nous avons ainsi créé le système EDS (Extractor Discovery System), qui est capable, à partir d'un ensemble de signaux étiquetés suivant une propriété donnée, de générer automatiquement un modèle général de cette propriété, en construisant des fonctions de traitement du signal pertinentes suivant une technique inspirée par la programmation génétique.

Les évaluations ont montré que le système actuel est capable en général de construire automatiquement des fonctions extrêmement performantes suivant un critère de pertinence donné, aussi bien dans des contextes généraux que spécifiques.

Nous avons ainsi pu vérifier sur plusieurs problèmes de description musicale l'hypothèse fondamentale que l'ajout des fonctions caractéristiques spécifiques créées par EDS permettait d'améliorer considérablement les performances de modèles descriptifs constitués uniquement de fonctions générales.

Enfin, la possibilité de générer des modèles exécutables indépendants d'EDS permet d'intégrer les descripteurs obtenus dans des applications musicales, comme des outils de recherche d'objets musicaux, ou de composition musicale.

Perspectives

Les fonctions présentées dans ce document ont été obtenues avec un nombre limité d'opérations de base et d'heuristiques, principalement adaptées au traitement de signaux audio.

Or EDS est un système modulaire en constante extension. Il est en effet extrêmement facile d'ajouter de nouveaux opérateurs, des paramétrages plus

détaillés, de nouvelles heuristiques de construction et règles de simplifications, ou de nouveaux types de données. L'extension du vocabulaire opérationnel d'EDS permet d'ouvrir l'espace de recherche et de trouver des fonctions toujours plus pertinentes.

Cependant, l'augmentation du nombre d'opérations possibles dans le cadre d'une recherche génétique, entraîne une augmentation des possibilités combinatoires. L'espace à explorer étant plus large, il est nécessaire d'optimiser également les opérations de traitement du signal afin de les maintenir dans limites compatibles avec les applications musicales. Il est notamment intéressant d'envisager un rapprochement avec le projet Faust ([Orlarey *et al.*, 2002]), visant à développer des techniques et outils pour la synthèse de code de traitement du signal efficace, qui utilise une sémantique formelle adaptée aux opérateurs EDS.

D'autre part, la phase de modélisation est actuellement divisée en deux étapes successives : la construction de fonctions caractéristiques pertinentes, utilisées ensuite dans des modèles de classification. Or nous envisageons actuellement la réunion de ces deux étapes par une intégration des modèles et de leurs paramètres dans la recherche génétique. Pour cela, il est nécessaire d'adapter la recherche génétique actuelle, qui permet de synthétiser une seule fonction optimale, à la construction d'un ensemble de fonctions pertinentes complémentaires.

Enfin, nous avons vu que la qualité du modèle dépend de celle de la base de signaux utilisée, dont le contexte doit être bien défini. Une solution pour faciliter la création de descripteurs musicaux serait le développement d'un module de création de bases de signaux à partir des fichiers musicaux d'un utilisateur. Plus fondamentalement, la gestion de bases mal étiquetées peut être prise en compte en adaptant EDS à un apprentissage semi-supervisé, par exemple en lançant plusieurs fois le système et en utilisant les modèles intermédiaires pour ré-étiqueter les signaux.

Diffusion

Publications

- Zils A., Pachet F., « Musical Mosaicing », Proceedings of the 4th COST-G6 Conference on Digital Audio Effects (DAFX01), University of Limerick, Ireland, December 2001.
- Zils A., Pachet F., Delerue O., Gouyon F., « Automatic Extraction of Drum Tracks from Polyphonic Music Signals », Proceedings of the International Conference on Web Delivering of music (WEDELMUSIC 2002), Darmstadt, Germany, 8th-11th December 2002.
- Pachet F., Zils A. « Evolving automatically high-level music descriptors from acoustic signals », 1st International Symposium on Computer Music Modeling and Retrieval (CMMR03), Springer Verlag LNCS, 2771, 2003.
- Pachet F., La Burthe A., Zils A. & Aucouturier JJ., « Popular Music Access The Sony music browser », in JASIS Journal of the American Society for Information, 2003.
- Zils A., Pachet F., « Automatic Extraction of High-Level Musical Descriptors from Acoustic Signals », Proceedings of the 6th COST-G6 Conference on Digital Audio Effects (DAFX03), Queen Mary University, London, September 2003.

- Zils A., Pachet F., « Automatic extraction of music descriptors from acoustic signals using EDS », AES 116th Convention, Berlin, Germany, May 8-11, 2004.
- Pachet, F., Aucouturier, J.-J., La Burthe, A., Zils, A. and Beurive, A., « The Cuidado Music Browser : an end-to-end Electronic Music Distribution System », Multimedia Tools and Applications, Special Issue on the CBMI03 Conference, 2004.
- Zils A., Pachet F., « Automatic extraction of music descriptors from acoustic signals », submitted to ISMIR04.

Présentations

- JJCAM (Journees Jeunes chercheurs en Acoustique Musicale) : « Query by drumming », IRCAM Paris, France, Juin 2001.
- AFPLC Seminary (Association Francaise de Programmation Logique et par Contraintes) : « Musical Mosaicing », IRCAM Paris, France, Janvier 2002.
- Sony CSL Paris Open House, « Synthesizing Holistic Music Extractors », Octobre 2002.
- Lip6, « Synthesizing Holistic Music Extractors », Paris, France, Décembre 2002.
- JJCAAS, « Automatic Extraction of High-Level Musical Descriptors using EDS », Telecom Paris, France, Octobre 2003.

Brevet

- Sony Patent I-02-138 : « Method and apparatus for automatically generating a general extraction function calculable on an input signal, e.g. an audio signal to extract therefrom a predetermined global characteristic value of its contents, e.g. a descriptor »

ANNEXES

A. GLOSSAIRE

ADSR : Attack Decay Sustain Release	Caractéristiques de l'enveloppe temporelle d'un son : Attaque Descente Soutien Relache
Bark	Echelle de fréquences tenant compte de la perception auditive humaine
Brightness	Brillance
Bandwidth	Largeur de bande : gamme des fréquences du signal
Harmonicité	Caractérise le fait qu'un signal possède des pics de résonance forts à des intervalles de fréquence réguliers
Log attack time	Log du temps d'attaque (voir ADSR)
Loudness	Sonie (volume perçu)
Mel	Echelle de fréquences tenant compte de la perception auditive humaine
MFCC : Mel Frequency Cepstral Coefficient	Coefficients cepstraux : ils représentent la forme globale du spectre avec un faible nombre de coefficients pertinents d'un point de vue perceptif
Modulation	Filtrage du signal à une (basse) fréquence donnée : typiquement, la modulation à 4Hz est utilisée en reconnaissance de parole car c'est la fréquence des syllabes dans un discours
Pitch	Fréquence perçue
RMS : Root Mean Square	Racine de la moyenne des carrés : mesure de l'énergie moyenne du signal
Spectral Centroid (Brightness)	Centroïde spectral : fréquence barycentre de l'énergie du spectre (moment d'ordre 1), elle mesure la « brillance » d'un son
Spectral Decrease	Décroissance spectrale : mesure la décroissance de l'amplitude spectrale (voir Spectral slope), suivant des critères perceptifs
Spectral Flatness (« whiteness »)	Platitude du spectre (« blancheur ») : mesure la proximité du son avec un bruit blanc
Spectral Flux (delta spectral magnitude)	Flux spectral : mesure le taux de variation du contenu spectral

Spectral Kurtosis	Aplatissement spectral : mesure l'aplatissement du spectre par rapport à une fonction gaussienne (moment d'ordre 4)
Spectral Roll-off	Point de roulement du spectre : fréquence sous laquelle on trouve la majorité de l'énergie spectrale (typiquement 95%) ; c'est une approximation de la fréquence de coupure entre les parties sinusoïdale et bruit du signal.
Spectral Sharpness	Equivalent du centroïde spectral, mais calculé en utilisant l'échelle de Bark.
Spectral Skewness	Dissymétrie du spectre : mesure la dissymétrie du spectre par rapport à une distribution gaussienne (moment d'ordre 3)
Spectral Smoothness ([McAdams, 1999])	« Douceur » du spectre : elle mesure la continuité de l'enveloppe spectrale, et est liée à la perception de différentes sources sonores
Spectral Slope	Pente spectrale : mesure la décroissance de l'amplitude spectrale avec la fréquence
Spectral Spread	Étendue spectrale : mesure l'étendue de l'énergie spectrale autour du centroïde (moment d'ordre 2)
Spectral Variation	Variation spectrale : mesure la variation du spectre d'une trame d'analyse à la suivante, à partir de la corrélation de leurs fonctions d'amplitude
ZCR : Zero-Crossing Rate	Taux de passage à zéro : sur des signaux non-bruités, donne une approximation de la fréquence fondamentale ; sur des signaux quelconques, donne une approximation de la quantité de bruit présente dans le signal
GMM : Gaussian Mixture Model	Mélange de gaussiennes

HMM : Hidden Markov Models	Modèles de Markov Cachés (MMC) : Modélisation de l'évolution d'un système par la succession de plusieurs états, auxquels sont associés des probabilités de transition ; les états sont « cachés »dans la mesure où ils ne sont pas connus, mais seulement modélisés par une probabilité d'observation de certains paramètres (par exemple les valeurs de fonctions caractéristiques). Typiquement, son apprentissage est réalisé par l'algorithme de Baum-Welch, et son application sur une séquence est réalisée par l'algorithme de Viterbi.
kNN : k-nearest neighbors	Plus proches voisins
k-Means	Algorithme d'agrégation par nuées dynamiques (classification non-supervisée), regroupant les individus en constituant itérativement des groupes autour d'individus « moyens »
LDA : Linear discriminant analysis	Analyse discriminante linéaire
LWR : Locally weighted regression	Régression localement pondérée
MDR : Minimum distance classifier	Classifieur plus proche voisin
NN : Neural Network	Réseau de neurones
SVM : Support vector machine	Machine à support de vecteur ([Vapnik, 1998])
Iterative ranking	Sélection de fonctions par calcul de la distance de Bhattacharyya entre classes, et sélection de l'erreur la plus basse

B. PRINCIPALES FONCTIONS CARACTÉRISTIQUES ET MODÈLES UTILISÉS EN DESCRIPTION MUSICALE

B.1 Classification de sons

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Wold *et al.*, 1996]	Énergie, fréquence fondamentale, brillance, largeur de bande, harmonicité (moyenne, variance, autocorrélation)	Distance euclidienne
[Brown, 1997]	MFCC	GMM
[Wold *et al.*, 1999]	Énergie, fréquence fondamentale, brillance, largeur de bande, coefficients cepstraux bruts et dérivés	Distance euclidienne, Modèle gaussien simple, GMM, corrélation
[Rossignol *et al.*, 1999] (Segmentation)	Moyenne et logarithme de la variance de : flux spectral, spectral centroïd, ZCR, CRRM (« Cepstrum Resynthesis Residual Magnitude ») ; fréquence fondamentale, énergie, harmonicité, vocalité : dérivée	GMM, kNN (7), réseaux de neurones (3 couches), modèle gaussien, modèle AR (changements abrupts)
[Tzanetakis et Cook, 1999] (Segmentation)	Moyenne et variance de : flux spectral, spectral centroïd, spectral roll-off, ZCR, RMS	dérivée de la distance de Mahalanobis
[Peeters *et al.*, 2000] (Instruments)	Log-attack time, harmonic spectral centroïd, harmonic spectral spread, harmonic spectral variation, harmonic spectral deviation	Distance de Mahalanobis
[Herrera *et al.*, 2000]	Mpeg7	kNN, Bayes naïf, analyse discriminante, SVM, arbres binaires, réseaux de neurones, statistiques d'ordre élevé, rough sets

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Gouyon *et al.*, 2000] (Sons percussifs)	Modèle de Prony, temps d'attaque, temps de decay, ZCR, énergie	Groupement par agglomération (agglomerative clustering)
[Gouyon et Herrera, 2001] (Sons percussifs)	Kurtosis spectral, centroïd temporel, Strong decay (energy + temporal centroid), ZCR, Centroïd spectral, énergie, Strong peak (spectral peak width)	Clustering (hiérarchique, K-Means, Fuzzy C-Means [Zadeh, 1965]), arbres de décision, analyse linéaire discriminante
[Aucouturier et Sandler, 2001] (segmentation)	MFCC	GMM
[Brown *et al.*, 2001] (instruments à vent)	MFCC, spectral smoothness [McAdams, 1999], spectral centroïd, énergie, autocorrelation, moment temporels du signal (3 :skew, 4 :kurtosis, 5 - dubnov 97)	GMM
[Eronen, 2001] (instruments)	Enveloppe d'amplitude, modulation 4-8Hz et 10-40Hz, centroïd spectral, fréquence fondamentale, MFCC	kNN
[Peeters et Rodet, 2002]	Valeur brute, moyenne, variance, corrélation, pente, modulations de : Log-attack time, temporal decrease, temporal centroïd, durée, ZCR, corrélation, énergie, énergie harmonique, énergie du bruit, spectral centroïd, spectral skewness, spectral spread, spectral kurtosis, spectral slope, spectral decrease, spectral rolloff, spectral variation, fréquence fondamentale, niveau de bruit, ratio harmonique, MFCC, sonie, sharpness, spread, rugosité	Sélection (Analyse discriminante, information mutuelle) + GMM
[Herrera *et al.*, 2002a]	Descripteurs bruts, obtenus par traitement du signal, perceptifs, MFCC	kNN, Bayes naïf, analyse discriminante, arbre binaire, statistiques d'ordre élevé, réseaux de neurones, SVM, rough sets, HMM
[Qi *et al.*, 2002] (Reconnaissance de sons)	ADSR : Durée, ratio, MFCC, ratio de puissance de 20 harmoniques sur la fréquence fondamentale	Distance paramétrique pondérée

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Herrera *et al.*, 2002b] (Sons percussifs)	Attaque et decay, moyenne et variance, par bande : Énergie, centroïd temporel, log attack time, ZCR, duréee, spectral flatness, spectral centroïd, spectral kurtosis, strong peak, strong decay, skewness, 13 MFCC	Sélection de fonctions (CFS, ReliefF, wrapping ([Blum et Langley, 1997]), stepwise(=backwards)), puis (k)NN, analyse discriminante ([Huberty, 1994]), arbres de décision (C4.5)
[Eggink et Brown, 2003] (instruments)	Énergie par bande spectrale	GMM avec « missing feature theory »
[Gillet et Richard, 2004] (Sons percussifs)	13 MFCC, centroïde spectral, largeur du spectre, spectral skewness, spectral flatness, log-énergie sur 6 bandes	HMM, SVM

Tab. B.1: Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en classification de sons

B.2 Genre musical

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Foote, 1997]	MFCC	arbre-Q
[Lambrou *et al.*, 1998]	Statistiques du 1er ordre (moyenne, variance, skewness, kurtosis), du 2eme ordre (second moment angulaire, corrélation, entropie), ZCR	MDC, kNN, MDC moindres carrés, MDC quadratique, meilleur taux de classification
[Logan, 2000]	MFCC	GMM
[Tzanetakis *et al.*, 2001b]	Spectral centroid, spectral rolloff, flux spectral, ZCR, énergie, MFCC, fonction de rythme (passage en ondelettes, enveloppe d'amplitude par bande, somme par bande, autocorrélation, positions et amplitudes des 5 pics principaux)	Modèle gaussien
[Tzanetakis *et al.*, 2001a] (Musique classique (chorale, orchestre, piano, cordes))	Fonctions spectrales ([Wold *et al.*, 1996]), MFCC, transformée en ondelettes discrète (moyenne, variance, et ratio par bande de fréquence)	Modèle gaussien

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Li *et al.*, 2003] (10 genres : 1000 extraits de 30s, 5 genres : 756 extraits de 30s)	MFCC, spectral centroid, spectral roll-off, flux spectral, ZCR, énergie, pics d'autocorrélation (pitch), coefficients d'ondelettes (énergie et 3 premiers moments de l'histogramme par bande)	Analyse discriminante, kNN, régression linéaire, arbres de décision transformation de problème multiple en problèmes booléens (1 contre le reste, comparaison par paires, codage de sortie avec correction d'erreurs, fonctions objectives multiclasses) avec LDA, SVM, kNN, et GMM
[McKinney et Breebaart, 2003] (5 classes (musique classique, pop, discours, bruit de foule, bruits d'ambiance) ou 7 genres musicaux)	4 groupes de fonctions : ([Li *et al.*, 2003]) RMS, Spectral centroid, largeur de bande, ZCR, spectral roll-off, ratio par bandes d'énergie, flux spectral, pitch (par autocorrélation), pitch strength ; MFCC ; (Psychoacoustiques) rugosité, loudness, sharpness, modulation d'énergie sur 3 bandes 1-2, 3-15, et 20-43Hz ; (Auditory filterbank - Gammatone) enveloppe moyenne, enveloppe de modulation d'énergie	Sélection de fonctions par « iterative ranking », et GMM
[Allamanche *et al.*, 2003] (Similarité musicale)	MFCC, sonie (+ delta log), spectral flatness, largeur de pic spectral, RCC (« real cepstral coeffs »), spectral tilt, spectral sharpness, ZCR	kNN
[Berenzweig *et al.*, 2003] (Similarité musicale)	MFCC et variations	GMM

Tab. B.2: Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en classification en genre musical

B.3 Discrimination parole / musique

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Saunders, 1996] (98% de réussite sur 150 samples de 2.4s)	Tonalité, largeur de bande, énergie, ZCR, durée	Modèle gaussien
[Scheirer et Slaney, 1997] (6% d'erreur +-2% sur 40mn, par tranche de 15s)	Modulation d'énergie à 4Hz (+), pourcentage de fenêtres de faible énergie, spectral rolloff, spectral centroïd, flux spectral (+), ZCR, CRRM (« Cepstrum Resynthesis Residual Magnitude »), métrique par autocorrélation par bandes (original)(+)	Estimateur gaussien multidimensionnel MAP (Maximum A Posteriori), GMM, classification spatiale k-d spatial, kNN
[Williams et Ellis, 1999] (98.7% de réussite sur 480 segments de 2.5s)	Fonctions de probabilité a posteriori (HMM NN sur 50 phonemes [Morgan et Bourlard, 1995]) : entropie moyenne par fenêtre, dynamisme moyen, ratio d'énergie, distribution des phonèmes	Modèle gaussien
[Carey et Lloyd-Thomas, 1999] (1.2% d'erreur sur 1000 samples de 10s)	Pitch, amplitude, coefficients cepstraux, ZCR	GMM
[Tzanetakis *et al.*, 2001a] (Parole/Musique, Voix (homme, femme, commentaire sportif))	Fonctions spectrales ([Wold *et al.*, 1996]), MFCC, transformée en ondelettes discrète (moyenne, variance, et ratio par bande de fréquence)	Modèle gaussien

Tab. B.3: Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en discrimination parole / musique

B.4 Détection et reconnaissance de voix chantée

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Chou et Gu, 2001] (26 mn de dialogue, musique et chant, par fenêtres de 500ms)	Valeurs brutes, et dérivées 1ère et 2nde : MFCC, Log Énergie, modulation d'énergie à 4Hz, ZCR, coefficient harmonique (original)	GMM : 2 gaussiennes
[Berenzweig et Ellis, 2001] (100 extraits musicaux de 15s, par fenêtres de 16ms : 26,1% d'erreur)	54 fonctions de probabilité a posteriori (sorties d'un NN) : moyenne et statistiques dérivées (entropie)	HMM
[Whitman *et al.*, 2001] (Reconnaissance d'artiste)	Coefficients de prédiction linéaire	SVM
[Berenzweig *et al.*, 2002] (Identification d'artiste)	MFCC	Perceptron multicouches
[Maddage *et al.*, 2003]	MFCC et dérivée, énergie , ZCR	SVM
[Tsai *et al.*, 2003] (Reconnaissance de voix chantée)	MFCC	GMM
[Kim et Whitman, 2003] (Reconnaissance de voix chantée)	MFCC	HMM
[Zhang, 2003] (Reconnaissance de chanteur)	Coefficients de prédiction linéaire	GMM
[Bartsch et Wakefield, 2004] (Reconnaissance de voix chantée)	Caractéristiques de l'enveloppe spectrale	Classifieur quadratique

Tab. B.4: Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en détection de voix chantée

B.5 Emotion

REFERENCE	FONCTIONS CARACTERISTIQUES	MODELE
[Oudeyer, 2002] (Émotion dans la parole)	Pitch, énergie, énergie basses-fréquences (250Hz), énergie hautes-fréquences (250Hz), MFCC : séries temporelles (brute, minima, maxima, intervalle entre extreme filtré à 10Hz)	kNN (1, 5, 10), arbres de décision C4.5, PART, densité du kernel, Kstar, régression linéaire, régression localement pondérée, perceptron, SVM, VFI (« voted feature interval »), régression M5prime, modèle de Bayes naïf
[Liu *et al.*, 2003b] (800 extraits de 20s de musique classique annotés à la main)	spectral centroïd, largeur de bande, spectral rolloff, spectral flux, contraste spectral (maxima, minima, moyenne par bande), RMS, enveloppe d'amplitude basses-fréquences (énergie, pic de corrélation, tempo)	GMM
[Li et Ogihara, 2003]	Fonctions de [Tzanetakis *et al.*, 2001b]	SVM

Tab. B.5: Récapitulatif des fonctions caractéristiques et modèles utilisés dans les principaux travaux de recherche en émotion dans la musique

C. PRÉSENTATION DES MÉTHODES CLASSIQUES DE MODÉLISATION UTILISÉES DANS EDS

C.1 Méthode 1-R de Holte

C'est un modèle de classification extrêmement simple consistant à diviser l'espace des valeurs de chaque fonction caractéristique en intervalles correspondant chacun à une classe donnée ([Holte, 1993]).

Apprentissage

Pour chaque fonction, l'espace des valeurs est segmenté afin de déterminer les limites des intervalles pour chaque classe. Seule la fonction f_i offrant la meilleure répartition des classes est conservée.

Classification

Pour classifier un nouvel individu, on lui attribue la classe correspondant à l'intervalle dans lequel se situe la valeur de sa fonction caractéristique f_i.

Paramètres principaux

Le nombre minimum d'individus dans un intervalle

C.2 Modèle de Bayes Naïf

Le modèle de Bayes Naïf est un modèle de classification qui associe une fonction de probabilité de classe sur chaque axe de l'espace paramétrique, c'est-à-dire pour chaque fonction caractéristique.

Apprentissage

Habituellement, les probabilités de classe pour chaque fonction sont modélisées par une gaussienne sur l'axe correspondant à cette fonction.

Classification

Pour classifier un nouvel individu, on calcule le produit des probabilités de chaque classe pour chaque fonction, et on choisit la classe la plus probable (voir figure C.1). Cependant, la validité du modèle repose sur l'hypothèse

d'indépendance des fonctions caractéristiques, qui est difficile à obtenir pratiquement (l'utilisation de « Réseaux Bayesiens »permet de contourner cette difficulté en tenant compte des relations entre les fonctions).

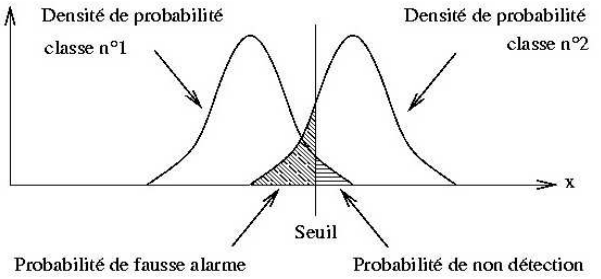


Fig. C.1: Modèle de classification Bayésienne pour un paramètre x et 2 classes

C.3 Plus-proche(s) voisin(s)

La méthode des Plus-Proches Voisins (« (k) Nearest Neighbor ») est une méthode de prédiction non-paramétrique, c'est-à-dire qui conserve l'ensemble des points d'apprentissage en mémoire, et les utilise à chaque prédiction.

Classification / Prédiction

Pour classifier un nouveau point, on examine le voisinage local de ce point dans l'espace paramétrique, et on lui attribue la classe du point d'apprentissage le plus proche (1 voisin). Dans l'algorithme des k Plus-Proches Voisins (« k Nearest Neighbors », ou kNN), on considère non pas le point mais les k points les plus proches, et on réalise la décision finale de la classe par un vote sur les classes de ces k points les plus proches (voir figure C.2). Afin de déterminer les points les plus proches de manière efficace, il est également possible de réaliser un partitionnement préalable de l'espace (« Spatial Partitioning »), par exemple sous forme d'arbre « k-d tree »([Bentley, 1975]).

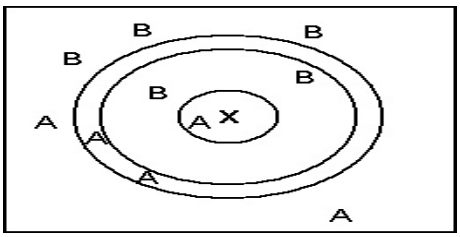


Fig. C.2: Prédiction de la classe d'un point X parmi 2 classes A et B dans un espace à 2 dimensions, par la méthode des k plus proches voisins : A pour k=1 ; B pour k=3 ; A pour k=5

Paramètres principaux

Le nombre de voisins pris en compte, la pondération des voisins (constante, en fonction de la distance)

C.4 Régression Linéaire

La Régression Linéaire est une méthode de prédiction qui consiste à trouver un modèle des valeurs perceptives sous forme d'une combinaison linéaire des fonctions caractéristiques.

Apprentissage

L'apprentissage consiste à trouver l'axe de l'espace des paramètres qui minimise l'erreur de modélisation, en utilisant un critère simple, comme par exemple le critère des « moindres carrés »(équivalent au critère du « maximum de vraisemblance »dans le cas particulier des régressions), qui minimise la somme des carrés des erreurs sur l'ensemble des points d'apprentissage. L'axe ainsi trouvé est une droite dont l'équation est appelée fonction de régression linéaire.

Prédiction

Pour prédire la valeur d'un nouveau point, il suffit d'appliquer la fonction de régression linéaire sur les paramètres du point.

La vérification d'hypothèses assez restrictives(absence d'erreur d'étiquetage des données, indépendance des fonctions caractéristiques, distribution normale de l'erreur de modélisation), permet d'assurer la stabilité et le pouvoir de généralisation du modèle.

C.5 Régression Locale Pondérée

Comme la méthode des k Plus-Proches Voisins, la Régression Locale Pondérée est une méthode de prédiction non-paramétrique conservant l'ensemble des points d'apprentissage en mémoire et les utilisant à chaque prédiction.

Prédiction

Pour prédire la valeur d'un nouvel individu, l'algorithme réalise un apprentissage local par régression (linéaire, gaussienne, ...) dans une partie de l'espace paramétrique située autour de cet individu, et lui applique ensuite le modèle appris pour obtenir sa valeur.

C.6 Modèle Gaussien

Le Modèle Gaussien est une méthode de classification qui consiste à modéliser chaque classe de données sous forme d'une région gaussienne de l'espace des paramètres.

Apprentissage

Les moyennes et les matrices de covariance de chaque gaussienne sont calculées par apprentissage supervisé.

Classification

La classification de nouvelles données est réalisée par calcul de la distance de Mahalanobis avec chaque classe.

C.7 Modèle de Mélange de Gaussiennes

Un Modèle de Mélange de Gaussiennes (« Gaussian Mixture Model », ou GMM) est une méthode de classification, qui consiste à modéliser chaque classe de données comme la réunion de plusieurs régions gaussiennes de l'espace des paramètres (voir figure C.3).

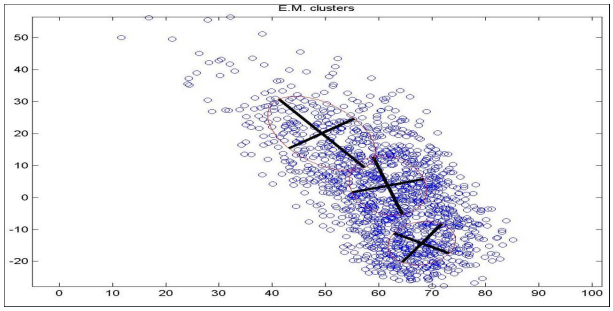


Fig. C.3: Modélisation d'un nuage de points dans un espace à 2 dimensions par un mélange de 3 gaussiennes

Apprentissage

Habituellement, l'apprentissage des gaussiennes est réalisé en utilisant un algorithme de recherche du paramètre réalisant le Maximum de Vraisemblance (« Expectation Maximization », ou EM). C'est un algorithme itératif en deux étapes : calcul d'Espérance E et Maximisation M ([Dempster *et al.*, 1977]). Contrairement aux modèles gaussiens simples, les gaussiennes ne sont pas représentées par des matrices de covariance complètes, mais seulement par leur approximation diagonale. Ainsi, les axes des gaussiennes sont orientés parallèlement aux axes de l'espace paramétrique.

Classification

On utilise pour classifier de nouvelles données un estimateur de Vraisemblance, mesurant la qualité de la représentation des données par chaque groupe de gaussiennes. On affecte finalement à cette donnée la classe qui est son meilleur modèle, c'est-à-dire son origine la plus vraisemblable.

Paramètres principaux

Nombre de gaussiennes

C.8 Arbres de Décisions

Les Arbres de Décision consistent à définir différentes familles d'individus en fonction de leurs propriétés (valeurs de leurs fonctions caractéristiques), et à générer un modèle spécifique pour chaque famille (régression), ou à attribuer une classe spécifique à chaque famille (classification).

Apprentissage

L'apprentissage d'un arbre de décision fournit une arborescence inversée dont chaque noeud terminal (ou « feuille ») contient un modèle (régression) ou correspond une classe (classification) associés à une famille d'individus. Chaque noeud de l'arbre correspond à un test sur les caractéristiques de l'individu permettant de lui attribuer une famille. En régression, les modèles des familles peuvent être variés, le plus courant étant le modèle de régression linéaire présenté précédemment. Les types d'apprentissage d'arbres de décision les plus courant sont CART et C5.

Classification / Prédiction

Pour prédire la valeur d'un nouvel individu, on détermine tout d'abord sa famille en descendant l'arbre depuis la racine jusqu'à une famille unique, le trajet dans l'arbre étant entièrement déterminé par les valeurs des fonctions caractéristiques (ou « prédicteurs »). En régression, on lui applique alors le modèle associé, afin de calculer la valeur du modèle. En classification, on lui attribue directement la classe associée.

Paramètres principaux

Le nombre minimal d'individus par famille (« feuille »), le nombre de possibilités par noeud (pour les prédicteurs)

C.9 Réseaux de Neurones

Les modèles de Réseaux de Neurones (« Neural Nets », ou NN) sont dérivés du modèle biologique du cerveau, consistant à reproduire l'architecture d'un ensemble de neurones connectés ensemble. Chaque neurone est une unité de calcul élémentaire qui applique une fonction de transfert donnée (typiquement la fonction sigmoïde) sur une ou plusieurs entrées pondérées, pour donner une valeur de sortie (voir figure C.4).

Le réseau le plus utilisé est le perceptron multicouche (PMC), qui est constitué d'une première couche de neurones (la « couche d'entrée ») reproduisant les valeurs des fonctions d'entrée, d'un nombre variable de « couches cachées »effectuant des opérations entre ces valeurs, tous les neurones d'une couche étant les uniques entrées des neurones de la couche suivante, et d'une « couche de sortie »fournissant les résultats finals (voir figure C.5).

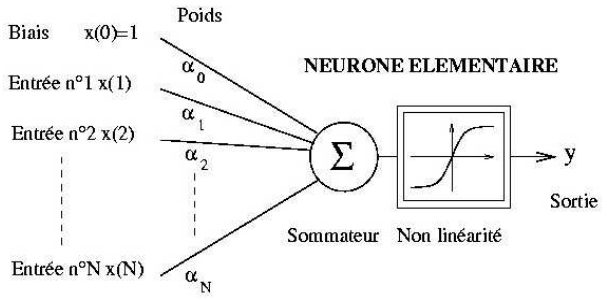


Fig. C.4: Schéma d'un neurone élémentaire

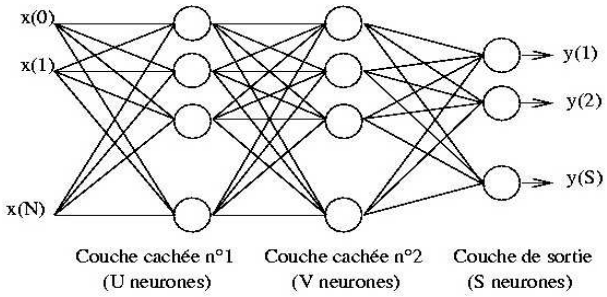


Fig. C.5: Schéma d'un perceptron à 2 couches cachées [N :U :V :S]

Apprentissage

Les neurones constituant un réseau sont définis par les pondérations de leurs entrées, et leur fonction de transfert. La fonction de transfert fixée (sigmoïde), l'apprentissage d'un réseau de neurones revient alors à déterminer les poids des entrées de chaque neurone permettant d'obtenir les valeurs perceptives en sortie, lorsqu'on fournit les valeurs des fonctions caractéristiques en entrée.

Classification / Prédiction

Pour prédire la classe ou la valeur d'un nouvel individu à partir d'un réseau de neurones, il suffit d'entrer les valeurs de ses fonctions caractéristiques dans le réseau, et de récupérer la valeur de sortie.

Paramètres principaux

Le nombre et la taille des couches cachées, les paramètres d'apprentissage (nombre d'époques, taux d'apprentissage)

D. MESURES DE PERTINENCE

Coefficient de corrélation de Pearson

Pour les problèmes de régression, la pertinence d'une fonction unaire peut être évaluée simplement par le calcul du coefficient de corrélation de Pearson entre la série des valeurs de la fonction et celle des étiquettes. Ce coefficient évalue la qualité de la relation linéaire entre les deux séries.

Il est calculé suivant la formule :

$$\text{Pearson}(fonction\ f, etiquettes\ e) = \frac{\sum_{i=1}^{N_{Signaux}} (f_i - \overline{f})(e_i - \overline{e})}{\left[\sum_{i=1}^{N_{Signaux}} (f_i - \overline{f})^2 \sum_{i=1}^{N_{Signaux}} (e_i - \overline{e})^2 \right]^{1/2}}$$

Ce coefficient de corrélation prend ses valeurs dans l'intervalle [-1 1] :

- une valeur proche de 0 indique une faible corrélation : les deux séries sont indépendantes, leurs variations n'étant pas liées ;
- une valeur proche de 1 indique une forte corrélation positive : les deux séries sont liées, et leurs variations se font dans le même sens ;
- une valeur proche de -1 indique une forte corrélation négative : les deux séries sont liées, et leurs variations se font en sens opposé ; on considère alors la fonction -f, dont le coefficient de corrélation a la valeur opposée, et qui est donc corrélée positivement.

Dans EDS, nous considérons qu'une fonction est pertinente si ses valeurs sur la base de signaux sont liées à celles des étiquettes, c'est-à-dire si le coefficient de Pearson est proche de 1 ou de -1. Nous utilisons donc comme fonction de pertinence la *valeur absolue du coefficient de corrélation de Pearson*, qui prend ses valeurs dans l'intervalle [0 1].

Ratio Discriminant de Fisher

Pour les problèmes de classification, la pertinence d'une fonction unaire peut être calculée simplement par un calcul de la fonction discriminante de Fisher ([Fisher, 1923]) pour l'ensemble des classes considérées. Cette fonction évalue le pouvoir discriminant de la fonction pour séparer les différentes classes de problème.

Elle est calculée suivant la formule :

$$\text{Fisher}(fonction\ f, classes\ c) = \sum_{i=1}^{N_{Classes}} \sum_{j=1, j\neq i}^{N_{Classes}} \frac{(\mu_i - \mu_j)^2}{(\sigma_i^2 + \sigma_j^2)},$$

avec : $\begin{cases} \mu_i = Moyenne\{f\}_{c=i} \\ \sigma_i = Variance\{f\}_{c=i} \end{cases}$

Le ratio discriminant de Fisher n'est pas calculable si la variance est nulle pour toutes les classes, c'est-à-dire que la fonction f renvoie une valeur identique pour tous les éléments d'une classe donnée. Dans ce cas, soit la fonction f est constante pour toutes les classes et ne présente donc aucun intérêt, soit elle a une valeur différente pour une des classes et est donc un critère parfait de reconnaissance de cette classe.

Dans le cas d'une discrimination booléenne entre 2 classes A et B, la formule se simplifie en :

$$\text{Fisher}(fonction\ f) = \frac{2 * (\mu_A - \mu_B)^2}{(\sigma_A^2 + \sigma_B^2)},$$

E. LISTE DE FONCTIONS TROUVÉES PAR EDS

Cette annexe présente une liste des principales fonctions trouvées par EDS pour les problèmes de description musicale présentés dans le chapitre 5 : Reconnaissance d'instruments, de voix, de sons percussifs, et de genre, et évaluation de l'énergie musicale.

Chaque tableau indique les performances des fonctions pour reproduire l'étiquetage des signaux, suivant le coefficient de corrélation absolue de Pearson pour l'énergie musicale (régression), et le critère de Fisher pour les autres problèmes (classification) (voir annexe D). Les performances sont présentées sur la base d'apprentissage (« Perf Learn ») et la base de test indépendante (« Perf Test »).

E.1 Reconnaissance de sons de voix

(Etude présentée section 5.3.1)

Fonction	Perf Learn	Perf Test
Square (Log10 (Abs (Max (Variance (Hanning (Derivation (BarkBands (Hamming (Split (Testwav, 1336.0)), 22.0))))))))	1,132066528	0,865689686
Log10 (Range (Derivation (Sqrt (Blackman (MelBands (Testwav, 24.0))))))	1,208870139	0,83137767
Log10 (Rms (Envelope (BpFilter (Testwav, 447.0, 1066.0), 225.0)))	0,724875886	0,739070286
Log10 (Abs (Log10 (Median (Iqr (Square (BpFilter (Blackman (Split (Testwav, 6873.0)), 354.0, 984.0)))))))	0,711270073	0,73710555
Log10 (Abs (Mean (Square (Rms (BpFilter (Hann (Split (Testwav, 410.0)), 975.0, 481.0))))))	0,74190902	0,726540421
Square (Log10 (Log10 (Max (Derivation (Derivation (Blackman (Blackman (MelBands (Testwav, 24.0)))))))))	1,156632661	0,725414027
Log10 (Rms (HFC (BpFilter (Triangle (Split (Sqrt (LpFilter (Testwav, 599.0)), 9013.0)), 645.0, 3509.0))))	0,755659339	0,677182166
Log10 (Abs (Mean (Holes (Rms (FilterBank (Abs (Hann (MelBands (Testwav, 28.0))), 23.0))))))	1,086347037	0,67390094
Square (Log10 (Multiplication (HFC (BpFilter (Testwav, 939.0, 462.0)), 9189.68042661452)))	0,734474144	0,673375106
Log10 (Rms (Rms (BpFilter (Hamming (Split (Testwav, 261.0)), 1146.0, 493.0))))	0,692525698	0,663087955
Log10 (Rms (HpFilter (Sqrt (Triangle (Testwav)), 334.0)))	0,685416181	0,654750955
Square (Log10 (Median (BarkBands (Correlation (Triangle (Testwav), Normalize (LpFilter (Normalize (LpFilter (Testwav, 4151.0)), 475.0))), 4.0))))	0,702938576	0,64852393
Log10 (HFC (BpFilter (Testwav, 497.0, 794.0)))	0,717857287	0,64622467
Log10 (HFC (BpFilter (Testwav, 499.0, 877.0)))	0,710775449	0,645076007
Sqrt (Rms (Max (BpFilter (Triangle (Split (Derivation (Testwav), 352.0)), 791.0, 236.0))))	0,655670249	0,64255673
Abs (Log10 (Multiplication (Percentile (HFC (BpFilter (Integration (Split (Testwav, 9746.0)), 445.0, 959.0)), 25.0), 8760.676801929989)))	0,692671565	0,638814465
Square (Log10 (Range (Hanning (BarkBands (Bartlett (Testwav), 45.0)))))	0,941077039	0,632123903
Abs (Log10 (Abs (Min (Correlation (BpFilter (Hann (Normalize (Testwav)), 4877.0, 704.0), Testwav)))))	0,664069196	0,629294729
Log10 (Abs (Log10 (Rms (Abs (Hanning (HpFilter (Testwav, 416.0)))))))	0,649486361	0,623489601
Log10 (Variance (Holes (Hanning (Fft (LpFilter (Testwav, 477.0))))))	1,215809863	0,617643463
Power (Log10 (Max (Sum (MelBands (Triangle (FilterBank (HpFilter (Testwav, 486.0), 89.0)), 4.0)))), 3.0)	1,01922987	0,603900952
Log10 (Max (Bartlett (BarkBands (Normalize (Testwav), 45.0))))	0,712466811	0,60311225
Log10 (Variance (BpFilter (Testwav, 375.0, 2436.0)))	0,64240869	0,600245769
Log10 (Rms (Derivation (Derivation (Blackman (BarkBands (Testwav, 15.0))))))	1,1267451	0,59488738
Log10 (Abs (Arcsin (Median (Rms (Square (BpFilter (Split (Sqrt (Testwav), 9038.0), 410.0, 4426.0)))))))	0,712471286	0,594818759
Sqrt (Mean (Fft (Derivation (BpFilter (Testwav, 482.0, 784.0)))))	0,806275583	0,584951636
Log10 (Rms (HFC (Hann (Split (Integration (HpFilter (Integration (HpFilter (Derivation (Testwav), 532.0)), 267.0)), 5072.0)))))	0,655884764	0,583640623
Sqrt (Arcsin (Sqrt (Median (Min (BpFilter (Split (Testwav, 264.0), 290.0, 2768.0))))))	0,664628933	0,583303784
Log10 (Abs (Max (Variance (Triangle (Split (Correlation (BpFilter (Testwav, 615.0, 3495.0), Testwav), 94.0))))))	0,599555204	0,575323944
Square (Log10 (Median (Envelope (Variance (Integration (BarkBands (Blackman (Split (Testwav, 62.0)), 10.0))), 8032.0))))	0,710661539	0,572012706
Log10 (Abs (Max (HFC (Triangle (Split (Sqrt (BpFilter (Sqrt (Testwav), 875.0, 181.0)), 875.0))))))	0,757314205	0,562059347
Log10 (Max (Max (Derivation (Blackman (Split (BarkBands (Testwav, 15.0), 1123.0))))))	1,102316629	0,559774085
Log10 (Variance (Abs (Derivation (BpFilter (Testwav, 1198.0, 11.0)))))	0,638839213	0,558594827

Fonction	Perf Learn	Perf Test
Power (Log10 (Abs (Median (Sqrt (Max (BarkBands (BpFilter (Triangle (Split (Testwav, 3175.0)), 1320.0, 365.0), 21.0)))))), 3.0)	0,842752877	0,558377507
Square (Log10 (Max (BarkBands (BpFilter (Normalize (Testwav), 1160.0, 370.0), 16.0))))	0,683583739	0,537211186
Log10 (Rms (HpFilter (Bartlett (Testwav), 265.0)))	0,565572901	0,529475799
Sqrt (Max (Max (Blackman (BarkBands (Split (Testwav, 390.0), 12.0)))))	0,862600323	0,52699444
Log10 (Abs (Iqr (HFC (Triangle (Split (BpFilter (Testwav, 4191.0, 344.0), 140.0))))))	0,904582011	0,520358535
Log10 (Variance (Holes (Hanning (Fft (LpFilter (Normalize (LpFilter (Normalize (BpFilter (Normalize (Testwav), 694.0, 280.0)), 498.0)), 2596.0))))))	1,625828071	0,519621335
Log10 (Rms (Range (MelBands (Sqrt (Autocorrelation (Hanning (Split (Testwav, 25.0)))), 5.0))))	0,605915225	0,505275393
Log10 (Rms (Iqr (BpFilter (Hamming (Split (Hanning (Testwav), 1398.0)), 477.0, 1039.0))))	0,71739879	0,502047816
Square (Log10 (Log10 (Rms (Hann (Derivation (BarkBands (Testwav, 15.0)))))))	1,018740159	0,499367284
Square (Log10 (Abs (Percentile (MelBands (Correlation (Normalize (Abs (Normalize (LpFilter (Normalize (Testwav), 461.0)))), Testwav), 6.0), 3.0))))	0,90583011	0,498354558
Log10 (Abs (Iqr (Range (HpFilter (Bartlett (Split (Testwav, 103.0)), 592.0)))))	1,07826965	0,497258839
Log10 (Median (HFC (Sqrt (Blackman (Split (Testwav, 353.0))))))	0,573380615	0,4962983
Sqrt (Variance (Sqrt (Blackman (BarkBands (Normalize (BpFilter (Normalize (Testwav), 997.0, 8.0)), 12.0)))))	0,738316628	0,495000441
Sqrt (HFC (Bartlett (BpFilter (Testwav, 181.0, 3162.0))))	0,588929466	0,47893825
Log10 (Median (HFC (Hann (Split (Testwav, 258.0)))))	0,512765145	0,478808364
Arcsin (Sqrt (Abs (Mean (Hann (Fft (BpFilter (Testwav, 744.0, 237.0)))))))	0,713251493	0,478569578
Abs (Log10 (HFC (BpFilter (Testwav, 2565.0, 215.0))))	0,641083927	0,475547223
Square (Log10 (Abs (Sum (Peaks (Blackman (Blackman (BarkBands (Testwav, 17.0))))))))	1,066292967	0,47087069
Square (Log10 (HFC (Sqrt (Triangle (LpFilter (Triangle (Testwav), 2730.0))))))	0,587687542	0,468621776
Sqrt (HFC (Bartlett (Sqrt (LpFilter (Testwav, 2549.0)))))	0,579709612	0,46623214
Sqrt (Max (HFC (Sqrt (Split (Hamming (Testwav), 1354.0)))))	0,567360754	0,459240622
Square (Log10 (Sum (HFC (Sqrt (LpFilter (Bartlett (Split (Testwav, 5680.0)), 930.0))))))	0,562398804	0,457251036
Log10 (Mean (Rms (Abs (MelBands (Split (Testwav, 7.0), 3.0)))))	0,498170506	0,455926959
Square (Log10 (Abs (Min (Power (Bartlett (MelBands (BpFilter (Testwav, 3334.0, 181.0), 21.0)), -2.0)))))	0,732964925	0,44617584
Square (Log10 (Abs (Mean (Fft (Sqrt (Normalize (LpFilter (Normalize (Testwav), 550.0))))))))	0,866189111	0,442879159
Log10 (HFC (Normalize (Testwav)))	0,415761149	0,440215342
Log10 (Min (HFC (SplitOverlap (BpFilter (Testwav, 3336.0, 212.0), 27193.0, 0.0))))	0,587033281	0,435225802
Sqrt (Percentile (HFC (Bartlett (FilterBank (Testwav, 9.0))), 85.0))	0,584858556	0,418308207
Log10 (Abs (HFC (BpFilter (Testwav, 126.0, 3189.0))))	0,553980662	0,408534467
Square (Log10 (Abs (Iqr (BarkBands (LpFilter (Normalize (LpFilter (Testwav, 2740.0)), 2740.0), 11.0)))))	0,791322448	0,399034882
Sqrt (Rms (HFC (Sqrt (Hamming (Split (Testwav, 4121.0))))))	0,551067535	0,384710675
Abs (Log10 (Rms (HFC (BpFilter (Hann (Split (Testwav, 6484.0)), 3095.0, 190.0)))))	0,588889929	0,378485207
Square (Log10 (Max (BarkBands (BpFilter (Testwav, 454.0, 2207.0), 11.0))))	0,783088335	0,377434968
Log10 (Rms (Power (HFC (LpFilter (Hamming (Split (Testwav, 1449.0)), 2852.0)), 5.0)))	0,468550933	0,369713552
Square (Log10 (Mean (Triangle (Abs (MelBands (Normalize (LpFilter (Testwav, 542.0)), 10.0))))))	0,617597959	0,369618674
Square (Log10 (Abs (Max (HFC (Hamming (Split (Testwav, 3910.0)))))))	0,523445301	0,3612194
Sqrt (Rms (HFC (Triangle (Split (Testwav, 6254.0)))))	0,489130639	0,360078932
Sqrt (Rms (HFC (Bartlett (Split (Testwav, 5053.0)))))	0,489095301	0,359965558
Square (Log10 (HFC (LpFilter (Testwav, 4183.0))))	0,555004054	0,358431134
Abs (Log10 (HFC (Triangle (LpFilter (Testwav, 4837.0)))))	0,567515312	0,357446758
Sqrt (Mean (Fft (Normalize (LpFilter (Normalize (Testwav), 546.0)))))	0,687346612	0,356215475
Square (Log10 (Log10 (Sum (Iqr (MelBands (Split (Testwav, 2224.0), 26.0))))))	0,619719969	0,353119211
Square (Log10 (Rms (HFC (Triangle (Split (Testwav, 6512.0))))))	0,521973797	0,351769668
Abs (Log10 (Abs (Min (HFC (Bartlett (Split (Testwav, 4963.0)))))))	0,550149545	0,349067119
Sqrt (Median (BarkBands (LpFilter (Testwav, 436.0), 4.0)))	0,597825901	0,343996019
Abs (Log10 (Abs (Max (Variance (BarkBands (Blackman (Split (HpFilter (Testwav, 346.0), 72.0)), 1.0))))))	0,68114202	0,343481723
Abs (Log10 (Abs (HFC (Blackman (Testwav)))))	0,524645429	0,34258447
Abs (Log10 (HFC (Blackman (Testwav))))	0,524645429	0,34258447
Square (Log10 (HFC (Testwav)))	0,537609687	0,342410043
Square (Log10 (Abs (HFC (Testwav))))	0,537609687	0,342410043
Square (Log10 (Rms (HFC (Split (Testwav, 7737.0)))))	0,537609687	0,342410043
Square (Log10 (Median (HFC (Split (Testwav, 4960.0)))))	0,537609687	0,342410043
Square (Log10 (Abs (Min (HFC (Split (Testwav, 8837.0))))))	0,537609687	0,342410043
Square (Log10 (Sum (Sqrt (HFC (Split (Testwav, 5831.0))))))	0,537609687	0,342410043
Log10 (Abs (Mean (Fft (HpFilter (Testwav, 473.0)))))	0,584428826	0,342090264
Power (Log10 (HFC (Sqrt (Testwav))), 3.0)	0,521269853	0,339077386
Square (Log10 (SpectralCentroid (LpFilter (Normalize (Testwav), 858.0))))	0,37861238	0,335648171
Sqrt (Abs (Min (HFC (Split (Testwav, 6967.0)))))	0,509909236	0,332906251
Power (Log10 (Mean (BarkBands (Normalize (LpFilter (Normalize (LpFilter (Normalize (Testwav), 576.0)), 3065.0)), 2.0))), 4.0)	0,601991311	0,332040002
Power (Log10 (Median (Square (BarkBands (Normalize (Testwav), 9.0)))), 3.0)	0,506398325	0,324243449
Sqrt (Abs (Log10 (HFC (Triangle (Testwav)))))	0,517381491	0,322954783
Power (Log10 (HFC (Sqrt (Testwav))), 4.0)	0,513104805	0,318884873
Sqrt (Iqr (Fft (Testwav)))	0,557502412	0,318599557
Log10 (HFC (Testwav))	0,48671529	0,317826703
Square (Log10 (Abs (Mean (Fft (HpFilter (Testwav, 372.0))))))	0,626808331	0,311768727
Sqrt (Sum (Median (BarkBands (SplitOverlap (Testwav, 3635.0, 0.8809635911708378), 7.0))))	0,622995665	0,306994468
Log10 (Abs (Iqr (Square (HFC (Split (Testwav, 258.0))))))	0,597988278	0,306901905
Sqrt (Percentile (Median (BarkBands (Split (Testwav, 2946.0), 10.0)), 80.0))	0,589115391	0,306521672
Square (Log10 (Abs (Median (Max (Derivation (Abs (BarkBands (Triangle (Split (Testwav, 9983.0)), 18.0))))))))	0,735713331	0,294048979
Power (Log10 (Abs (Iqr (BarkBands (Normalize (LpFilter (Normalize (Testwav), 1999.0)), 9.0)))), 8.0)	0,907123105	0,286521451
Square (Log10 (Iqr (BarkBands (Hamming (LpFilter (Testwav, 2387.0)), 9.0))))	0,690519805	0,284278165
Abs (Log10 (Abs (Sum (Abs (HFC (Hanning (Split (Testwav, 627.0))))))))	0,580356856	0,28062105
Square (Log10 (Mean (Square (Min (Sqrt (SplitOverlap (Iqr (BarkBands (Triangle (Split (LpFilter (Testwav, 3002.0), 7026.0)), 16.0)), 1514.0, 0.0)))))))	0,699694554	0,273939996
SpectralSpread (Abs (Normalize (Testwav)))	0,306189431	0,27353631
Min (MelBands (HpFilter (Testwav, 2368.0), 4.0))	0,541613477	0,265648355
Log10 (Rms (Square (MelBands (Normalize (Testwav), 2.0))))	0,443061165	0,265379623
Power (Log10 (Abs (Percentile (Hann (BarkBands (Normalize (Testwav), 7.0)), 85.0))), -1.0)	0,304573901	0,260160858
Square (Log10 (Mean (Fft (Sqrt (Testwav)))))	0,508131512	0,259918983
Rms (Iqr (Fft (Split (Testwav, 6214.0))))	0,554248896	0,258621825
Min (MelBands (Derivation (Testwav), 2.0))	0,559054495	0,256172474
Power (Log10 (Percentile (BarkBands (Testwav, 8.0), 45.0)), 5.0)	0,570719268	0,250624092
Max (Percentile (BarkBands (SplitOverlap (Testwav, 3302.0, 0.6827903793953651), 1.0), 31.0))	0,566493271	0,232068006
Power (Log10 (Abs (Mean (Integration (Integration (Abs (Iqr (MelBands (Split (Testwav, 3333.0), 224.0)))))))), 3.0)	0,617711873	0,227974093
Median (BarkBands (Testwav, 9.0))	0,567895008	0,223544327

Fonction	Perf Learn	Perf Test
Abs (Sum (Hamming (BarkBands (Testwav, 7.0))))	0,549587637	0,218312987
Median (BarkBands (Testwav, 8.0))	0,586495622	0,218089203
Sqrt (Median (MelBands (Testwav, 9.0)))	0,522083825	0,21771663
Log10 (Iqr (HFC (Blackman (Split (Testwav, 252.0)))))	0,595793315	0,216936756
Sum (BarkBands (Testwav, 2.0))	0,520160946	0,191663721
Percentile (MelBands (Testwav, 1.0), 11.0)	0,506560344	0,184052609
Mean (Max (Abs (MelBands (Split (Testwav, 4150.0), 2.0))))	0,538875408	0,181286373
Median (MelBands (Testwav, 9.0))	0,524611892	0,177097714
Max (BarkBands (Testwav, 2.0))	0,541734493	0,161491136
Percentile (MaxPos (Derivation (Autocorrelation (Hamming (Split (Testwav, 1592.0))))), 8.0)	0,545887661	0,103143706

E.2 Reconnaissance de la voix d'une chanteuse

(Etude présentée section 5.3.1)

Fonction	Perf Learn	Perf Test
Log10 (Median (SpectralKurtosis (HpFilter (Hamming (Split (Normalize (Testwav), 942.0)), 4457.0))))	0,923547917	1,087231394
Sqrt (Mean (Sqrt (Peaks (SpectralSkewness (HpFilter (Triangle (Split (Testwav, 300)), 3122.0))))))	1,175681647	1,085234432
Log10 (Log10 (Percentile (SpectralKurtosis (Integration (BpFilter (Hanning (Split (HpFilter (Testwav, 4404.0), 84.0)), 5418.0, 2583.0))), 46.0)))	0,843080036	1,012194856
Power (Log10 (Median (Sqrt (SpectralKurtosis (Derivation (Hamming (Split (Testwav, 455.0)))))), -1.0)	0,781821267	0,894019648
Square (Zcr (Derivation (BpFilter (Normalize (Testwav), 4886.0, 2719.0))))	0,671043726	0,887018276
Power (Arcsin (Square (Zcr (Normalize (BpFilter (Normalize (LpFilter (Blackman (Testwav), 3369.0)), 3505.0, 4105.0))))), 4.0)	0,728929122	0,859407182
Square (Max (Zcr (LpFilter (Hann (Split (BpFilter (Testwav, 4406.0, 4951.0), 3077.0)), 3609.0))))	0,752790941	0,842434733
Square (Zcr (HpFilter (Normalize (Derivation (Normalize (Testwav))), 3102.0)))	0,689095328	0,818253587
Log10 (SpectralSpread (HpFilter (Normalize (HpFilter (Hann (LpFilter (Sqrt (HpFilter (Testwav, 3221.0)), 1785.0)), 4184.0)), 4399.0)))	0,69049591	0,813442272
Square (Max (Zcr (LpFilter (Hann (FilterBank (Testwav, 9.0)), 3213.0))))	0,578443766	0,810027214
Arcsin (Square (Percentile (Zcr (LpFilter (Blackman (FilterBank (Testwav, 10.0)), 3112.0)), 86.0)))	0,574977952	0,799087462
Log10 (Abs (Max (SpectralKurtosis (BpFilter (Hann (Split (Normalize (Testwav), 1711.0)), 3633.0, 4837.0)))))	0,823572943	0,799080229
Square (Zcr (Normalize (Hanning (HpFilter (Triangle (Normalize (Blackman (Normalize (Triangle (Normalize (Testwav)))))), 3692.0)))))	0,710602937	0,798185195
Square (Zcr (HpFilter (Testwav, 3719.0)))	0,694559985	0,788780024
Square (Zcr (HpFilter (Normalize (Testwav), 3703.0)))	0,692229141	0,785343448
Square (Median (Zcr (SplitOverlap (HpFilter (Normalize (Testwav), 3832.0), 9333.0, 0.560519832359025))))	0,691521388	0,783806165
Square (Log10 (SpectralKurtosis (BpFilter (Normalize (Testwav), 4212.0, 3803.0))))	0,681065312	0,777891693
Square (Zcr (HpFilter (Normalize (Testwav), 3922.0)))	0,685379335	0,771142027
Abs (Arcsin (Sqrt (Zcr (HpFilter (Hanning (Testwav), 3821.0)))))	0,66124505	0,757537373
Square (Log10 (Range (Correlation (BpFilter (Bartlett (Square (BpFilter (Correlation (Normalize (Testwav), Correlation (BpFilter (Testwav, 175.0, 319.0), Testwav)), 517.0, 245.0))), 652.0, 287.0), Testwav))))	0,795838002	0,754734241
Log10 (SpectralKurtosis (BpFilter (Testwav, 4552.0, 3585.0)))	0,707950429	0,75016293
Log10 (Abs (Log10 (Percentile (SpectralKurtosis (HpFilter (Bartlett (Split (Testwav, 34.0)), 2112.0)), 0.0))))	0,842532009	0,745809469
Power (Min (SpectralKurtosis (Derivation (Triangle (Split (Testwav, 596.0))))), -2.0)	0,765982128	0,739135764
Log10 (Median (SpectralKurtosis (Derivation (Blackman (Split (Testwav, 133.0))))))	0,728887401	0,736618291
Square (Median (Hanning (Zcr (FilterBank (Normalize (Testwav), 3.0)))))	0,607005586	0,733098778
Zcr (HpFilter (Hanning (Testwav), 3420.0))	0,641185226	0,723213797
Power (Arcsin (Median (RHF (HpFilter (Hann (Split (Normalize (Testwav), 186.0)), 4836.0)))), 3.0)	0,65065534	0,722947632
Square (Mean (Zcr (Hann (SplitOverlap (Normalize (HpFilter (Testwav, 3320.0)), 2723.0, 0.13259602717792018)))))	0,667082598	0,711973838
Power (SpectralKurtosis (BpFilter (Normalize (Testwav), 3944.0, 5388.0)), -2.0)	0,637425993	0,71147484
Arcsin (Square (Zcr (HpFilter (Testwav, 4219.0))))	0,606471645	0,705608764
Log10 (SpectralSpread (Normalize (BpFilter (Normalize (BpFilter (Normalize (Blackman (Normalize (Testwav))), 4434.0, 3087.0)), 4434.0, 3572.0))))	0,661873641	0,70473037
Arcsin (Power (Percentile (Zcr (FilterBank (Triangle (Normalize (Testwav)), 6.0)), 89.0), 6.0))	0,604530545	0,701588363
Zcr (Normalize (Power (HpFilter (Normalize (Testwav), 3336.0), -1.0)))	0,624251362	0,699698549
Log10 (SpectralKurtosis (HpFilter (Normalize (Testwav), 4220.0)))	0,513966075	0,694324785
Log10 (Abs (Max (Zcr (Triangle (Split (HpFilter (Normalize (Testwav), 3944.0), 19238.0))))))	0,565715071	0,676877149
Sqrt (Percentile (SpectralRolloff (Derivation (Hamming (Split (HpFilter (Normalize (Testwav), 1399.0), 35.0)))), 43.0))	0,601354924	0,671014052
Log10 (SpectralKurtosis (Normalize (Triangle (Derivation (BpFilter (Testwav, 3707.0, 4891.0))))))	0,592449042	0,669216931
Arcsin (Square (Zcr (Power (Bartlett (Power (HpFilter (Normalize (Testwav), 3207.0), -1.0)), -1.0))))	0,606721207	0,654080725
Power (Log10 (Rms (SpectralDecrease (HpFilter (Bartlett (Split (Derivation (Normalize (Testwav)), 188.0)), 4678.0)))), 5.0)	0,56311732	0,645116012
SpectralKurtosis (Derivation (HpFilter (Normalize (Testwav), 4310.0)))	0,295534521	0,619642376
Power (Log10 (Abs (Range (BarkBands (Correlation (Abs (Hanning (Hanning (Normalize (Testwav)))), Testwav), 2.0)))), 4.0)	0,426872942	0,612030328
Sqrt (HFC (BpFilter (Sqrt (BpFilter (Testwav, 125.0, 3079.0)), 125.0, 313.0)))	0,688324411	0,602708935
Variance (Derivation (RHF (FilterBank (Triangle (Testwav), 5.0))))	0,391334279	0,588771051
Power (Variance (Power (RHF (FilterBank (Triangle (Normalize (Testwav)), 10.0)), 5.0)), 3.0)	0,50178541	0,587945772
Power (SpectralRolloff (BpFilter (Normalize (BpFilter (Normalize (Testwav), 712.0, 4533.0)), 5222.0, 2303.0)), 3.0)	0,441679481	0,576200504
Power (Log10 (Abs (Min (SpectralKurtosis (Derivation (Hanning (Split (Testwav, 1499.0))))))), -1.0)	0,770839183	0,576070606
Power (Log10 (Abs (Percentile (Iqr (BpFilter (Hanning (Split (Testwav, 5775.0)), 419.0, 237.0)), 0.0))), -1.0)	0,569655124	0,55875649
Power (Log10 (Zcr (HpFilter (Normalize (Testwav), 3106.0))), -1.0)	0,456233375	0,556102078
Power (Log10 (Min (Peaks (Abs (SpectralRolloff (Sqrt (Split (Integration (HpFilter (Sqrt (BpFilter (Sqrt (Testwav), 4910.0, 1988.0)), 1519.0)), 1988.0))))))), 3.0)	0,617236629	0,552432166

Fonction	Perf Learn	Perf Test
Square (Log10 (SpectralSpread (HpFilter (Normalize (HpFilter (Normalize (Testwav), 4184.0)), 2932.0))))	0,601269655	0,544067311
Max (SpectralCentroid (HpFilter (Bartlett (Split (BpFilter (Normalize (Testwav), 2547.0, 4564.0), 6551.0)), 1763.0)))	0,464184946	0,543023509
Abs (SpectralRolloff (Autocorrelation (BpFilter (Normalize (Testwav), 184.0, 2513.0))))	0,344991804	0,537387043
Power (Min (SpectralKurtosis (Derivation (Hann (Split (Normalize (Testwav), 1520.0))))), -4.0)	0,76299596	0,535571436
Power (SpectralRolloff (BpFilter (Testwav, 2132.0, 3736.0)), 3.0)	0,635522762	0,531019118
Power (Abs (SpectralRolloff (Derivation (BpFilter (Testwav, 3513.0, 1761.0)))), 3.0)	0,608635458	0,528458136
Square (Max (MaxPos (MelBands (Bartlett (Split (HpFilter (Blackman (Testwav), 2767.0), 2582.0)), 7.0))))	0,497572631	0,524259313
Power (Abs (SpectralRolloff (Derivation (BpFilter (Testwav, 3568.0, 1906.0)))), 5.0)	0,619823632	0,52395355
Square (Log10 (SpectralSpread (Sqrt (HpFilter (Normalize (Testwav), 4628.0)))))	0,56945893	0,521288865
Log10 (Min (SpectralSpread (BpFilter (Split (HpFilter (Testwav, 1635.0), 8971.0), 4458.0, 4994.0))))	0,565717105	0,517832735
Power (Log10 (Abs (Range (SpectralCentroid (FilterBank (Normalize (Testwav), 3.0))))), 4.0)	0,345749203	0,51441546
Sqrt (Rms (BpFilter (Autocorrelation (Testwav), 249.0, 370.0)))	0,586186978	0,486117237
Log10 (Min (SpectralSpread (Hann (Split (Normalize (Testwav), 37.0)))))	0,523845197	0,486071705
Power (SpectralRolloff (BpFilter (Normalize (Hamming (Normalize (Testwav))), 2788.0, 3549.0)), 3.0)	0,470199721	0,482126015
MaxPos (Mfcc (Blackman (HpFilter (Testwav, 3567.0)), 3.0))	0,331223355	0,481325967
Power (Log10 (Median (SpectralKurtosis (Derivation (Split (Normalize (Testwav), 1796.0))))), -1.0)	0,543574429	0,468039817
Power (SpectralKurtosis (Normalize (Derivation (Normalize (Testwav)))), -2.0)	0,546461326	0,462928322
Power (SpectralKurtosis (Derivation (Testwav)), -2.0)	0,548684182	0,462928322
Square (Log10 (Range (Correlation (BpFilter (Integration (Testwav), 17.0, 385.0), Correlation (BpFilter (Normalize (Testwav), 216.0, 308.0), BpFilter (Testwav, 2706.0, 4918.0))))))	0,748134389	0,457486531
Power (Log10 (Abs (SpectralKurtosis (Derivation (Testwav)))), -1.0)	0,522228674	0,448564766
Log10 (Abs (SpectralKurtosis (HpFilter (Testwav, 1169.0))))	0,426636711	0,443094751
Log10 (SpectralKurtosis (HpFilter (Normalize (Testwav), 1525.0)))	0,454266384	0,43865376
Log10 (SpectralKurtosis (Derivation (Testwav)))	0,439712224	0,434388703
Log10 (Abs (SpectralKurtosis (Derivation (Testwav))))	0,439712224	0,434388703
Square (Log10 (HFC (Correlation (BpFilter (Bartlett (Power (BpFilter (Correlation (Integration (Normalize (Testwav)), Correlation (BpFilter (Testwav, 285.0, 259.0), Testwav)), 377.0, 225.0), 4.0)), 418.0, 217.0), Testwav))))	0,801157055	0,433358114
Power (Min (SpectralKurtosis (Hann (Split (HpFilter (Normalize (HpFilter (Testwav, 1349.0)), 942.0), 1707.0)))), -1.0)	0,722011088	0,430572481
Arcsin (Power (Abs (Min (Zcr (Bartlett (Split (Derivation (Power (Derivation (Normalize (Testwav)), 5.0)), 8310.0))))), 3.0))	0,34542303	0,426942277
Power (SpectralKurtosis (BpFilter (Derivation (Normalize (Testwav)), 4321.0, 160.0)), -1.0)	0,447104725	0,417961669
Log10 (Abs (SpectralKurtosis (HpFilter (Normalize (Testwav), 1645.0))))	0,433099057	0,403451353
Log10 (RHF (HpFilter (Normalize (Testwav), 2585.0)))	0,298939418	0,399705206
Percentile (Sqrt (SpectralRolloff (BpFilter (Bartlett (Split (Derivation (Testwav), 4721.0)), 3932.0, 1293.0))), 98.0)	0,343172334	0,39612409
Power (SpectralKurtosis (HpFilter (Normalize (Testwav), 1645.0)), -1.0)	0,51769491	0,3953087
Log10 (Abs (Mean (Peaks (LpFilter (Testwav, 587.0)))))	0,255601497	0,38803664
Abs (SpectralRolloff (BpFilter (Normalize (Testwav), 4004.0, 1758.0)))	0,439750174	0,387306839
Power (Log10 (SpectralRolloff (BpFilter (Testwav, 4985.0, 2658.0))), 9.0)	0,258034996	0,385364343
SpectralRolloff (Normalize (Abs (Normalize (BpFilter (Derivation (Normalize (Testwav)), 1305.0, 604.0)))))	0,345714795	0,38274627
Log10 (Median (SpectralSpread (HpFilter (Hamming (Split (Testwav, 3847.0)), 4351.0))))	0,420427502	0,374333251
Abs (SpectralRolloff (Normalize (HpFilter (Normalize (Testwav), 1822.0))))	0,388482727	0,359118219
Power (Log10 (Max (SpectralRolloff (Split (HpFilter (Normalize (HpFilter (Normalize (Testwav), 1205.0)), 1272.0), 3187.0)))), 5.0)	0,482233571	0,351377461
Square (Percentile (MaxPos (BarkBands (Abs (Split (Testwav, 1257.0)), 3.0)), 2.0))	0,280846809	0,321188877
Square (SpectralRolloff (HpFilter (Normalize (Testwav), 2104.0)))	0,351679482	0,317293599
Power (Log10 (Abs (SpectralRolloff (HpFilter (Testwav, 1660.0)))), 3.0)	0,358507053	0,314176808
SpectralRolloff (HpFilter (Testwav, 1620.0))	0,388469954	0,309110213
Log10 (Abs (Log10 (Mean (Sqrt (Rms (FilterBank (Hann (Mfcc (Sqrt (Correlation (Normalize (Testwav), Testwav)), 8.0)), 7.0)))))))	0,298003135	0,297228998
Log10 (Zcr (Normalize (HpFilter (Correlation (Normalize (HpFilter (Correlation (Derivation (HpFilter (Normalize (HpFilter (Correlation (Hanning (Normalize (Testwav)), Testwav), 753.0)), 121.0)), Testwav), 662.0)), Testwav), 1088.0))))	0,339828807	0,293412035
Log10 (Abs (Mean (Peaks (Mean (Sqrt (Split (SpectralSkewness (HpFilter (Triangle (Split (HpFilter (Testwav, 2898.0), 55.0)), 3430.0)), 3.0)))))))	0,282771044	0,28204747
Variance (Derivation (BpFilter (Testwav, 339.0, 88.0)))	0,355482133	0,265038373
Log10 (Sum (SpectralFlatness (Derivation (Split (Abs (Normalize (Testwav)), 2006.0)))))	0,242361789	0,231087536
Sqrt (Abs (Log10 (Median (Fft (Hamming (Correlation (Normalize (Testwav), Testwav)))))))	0,457525666	0,229080549
SpectralCentroid (BpFilter (Normalize (Testwav), 2141.0, 3626.0))	0,238591108	0,224761995
Abs (Log10 (Iqr (Fft (Hamming (Hamming (Correlation (Derivation (Testwav), Testwav)))))))	0,460024674	0,213295765
Log10 (SpectralKurtosis (Blackman (Autocorrelation (BpFilter (Testwav, 1879.0, 4403.0)))))	0,296282865	0,200918318
Square (Log10 (Mean (Power (MaxPos (BarkBands (Blackman (Split (Testwav, 26.0)), 2.0)), -1.0))))	0,250932705	0,199587083
Power (Log10 (SpectralRolloff (Normalize (Derivation (Normalize (Testwav))))), 4.0)	0,259237795	0,19467779
Square (Log10 (Abs (SpectralRolloff (Derivation (Testwav)))))	0,254062448	0,19387752
Power (Zcr (BpFilter (Normalize (Testwav), 2351.0, 1717.0)), 5.0)	0,157169741	0,191965433
Abs (SpectralRolloff (Normalize (Derivation (Normalize (Testwav)))))	0,261106874	0,187517056
SpectralRolloff (Derivation (Testwav))	0,260775291	0,187517056
SpectralRolloff (Derivation (Normalize (Testwav)))	0,260775291	0,187517056
Arcsin (Sum (Holes (Min (BpFilter (Hann (Split (Normalize (Testwav), 2534.0)), 3722.0, 1064.0)))))	0,110561408	0,147402676
Arcsin (Square (Percentile (Zcr (LpFilter (Hann (Split (Testwav, 3841.0)), 3112.0)), 86.0)))	0,192990613	0,114357734
Square (Min (Zcr (Bartlett (Split (LpFilter (Hanning (Normalize (Testwav)), 2656.0), 13110.0)))))	0,187468701	0,109987163
Square (Percentile (Abs (Holes (Max (FilterBank (Triangle (Normalize (Testwav)), 8.0)))), 8.0))	0,122048049	0,106318004
Sqrt (Abs (Log10 (Sum (HFC (Derivation (Blackman (Split (Testwav, 282.0))))))))	0,51884065	0,105208678
Square (Rms (Median (Square (Mfcc (Split (Testwav, 2207.0), 1.0)))))	0,181089321	0,084838898
Power (Percentile (Zcr (Bartlett (Split (Testwav, 278.0))), 27.0), 3.0)	0,170297003	0,079468859
Power (Abs (Arcsin (Rms (Zcr (Blackman (Split (Testwav, 9372.0)))))), 3.0)	0,135743323	0,078325726
Square (Max (Holes (HpFilter (Normalize (Testwav), 3784.0))))	0,242296288	0,061545657
Square (Arcsin (Rms (Abs (Holes (Max (FilterBank (Normalize (Testwav), 7.0)))))))	0,135148055	0,040776529

E.3 Reconnaissance de la voix d'une chanteuse dans une chanson

(Etude présentée section 5.3.1)

Fonction	Perf Learn	Perf Test
Square (Log10 (Abs (SpectralFlatness (BpFilter (Normalize (BpFilter (Blackman (Correlation (Abs (BpFilter (Normalize (Blackman (Correlation (BpFilter (Normalize (Testwav), 308.0, 965.0), Testwav))), 232.0, 1596.0)), Testwav)), 1256.0, 244.0)), 326.0, 1848.0)))))	2,725546343	1,696628702
Log10 (Abs (Log10 (Zcr (BpFilter (Normalize (BpFilter (Hann (Normalize (Testwav)), 1623.0, 391.0)), 1623.0, 465.0)))))	1,462439532	0,992591519
Log10 (SpectralRolloff (Autocorrelation (BpFilter (Autocorrelation (BpFilter (Testwav, 550.0, 3116.0)), 1087.0, 341.0))))	1,465082478	0,910658526
Power (Log10 (Multiplication (SpectralRolloff (Derivation (BpFilter (Normalize (BpFilter (Testwav, 650.0, 211.0)), 319.0, 1685.0))), 8493.072424674227)), 13.0)	1,506411512	0,686040607
Square (Log10 (Percentile (SpectralCentroid (Square (Split (BpFilter (Triangle (Testwav), 1183.0, 663.0), 29.0))), 29.0)))	1,52117516	0,639577225
Square (SpectralRolloff (Normalize (BpFilter (Normalize (BpFilter (Normalize (Testwav), 1608.0, 272.0)), 900.0, 285.0))))	1,422629966	0,599556175
Log10 (Zcr (BpFilter (Derivation (BpFilter (Derivation (Derivation (Autocorrelation (Testwav))), 1399.0, 476.0)), 641.0, 88.0)))	1,373465131	0,594030388
Power (Log10 (MaxPos (MelBands (Sqrt (BpFilter (Normalize (Testwav), 367.0, 4062.0)), 17.0))), 4.0)	1,642503488	0,568657606
Log10 (SpectralSpread (Integration (Sqrt (Hamming (HpFilter (Hamming (HpFilter (Testwav, 578.0)), 511.0))))))	1,21954092	0,567861902
Square (Log10 (MaxPos (Triangle (Abs (Range (FilterBank (Bartlett (HpFilter (Correlation (LpFilter (Normalize (Testwav), 1379.0), Bartlett (HpFilter (Testwav, 540.0))), 794.0)), 26.0)))))))	0,973671307	0,514391723
Square (Log10 (Min (SpectralKurtosis (BpFilter (Hamming (Split (Normalize (Testwav), 770.0)), 1340.0, 1674.0)))))	1,028201334	0,512482336
Sqrt (Abs (SpectralRolloff (BpFilter (Testwav, 432.0, 944.0))))	1,278346612	0,442525078
Square (SpectralRolloff (Sqrt (BpFilter (Normalize (BpFilter (Sqrt (BpFilter (Sqrt (Testwav), 161.0, 1540.0)), 161.0, 1540.0)), 161.0, 1540.0))))	1,669676773	0,442075334
Arcsin (Log10 (MaxPos (Square (Range (FilterBank (Bartlett (HpFilter (Correlation (HpFilter (Normalize (Testwav), 1083.0), Testwav), 364.0)), 7.0))))))	1,544554455	0,436867486
Power (Log10 (Abs (Sum (Zcr (Hamming (FilterBank (BpFilter (Normalize (Testwav), 589.0, 1591.0), 1.0))))))), -1.0)	1,235373445	0,43537244
Log10 (Abs (Log10 (Median (SpectralKurtosis (HpFilter (Bartlett (Split (Normalize (Testwav), 189.0)), 1620.0))))))	1,424643977	0,415941928
Power (Log10 (Log10 (SpectralRolloff (BpFilter (Derivation (Testwav), 179.0, 569.0)))), 7.0)	1,564281461	0,39781268
Abs (Sum (Rms (Autocorrelation (Bartlett (Split (Testwav, 822.0))))))	0,449447025	0,392009859
Power (Log10 (Mean (Variance (Mfcc (Hanning (Split (Testwav, 470.0)), 2.0)))), 4.0)	1,864614956	0,379739507
Power (Log10 (SpectralRolloff (Autocorrelation (BpFilter (Integration (Normalize (BpFilter (Integration (Normalize (Testwav)), 671.0, 5434.0))), 671.0, 5434.0)))), 4.0)	1,274404565	0,348701441
Sqrt (Percentile (SpectralSpread (Hamming (Split (Normalize (Testwav), 21.0))), 37.0))	1,037428033	0,326414542
Power (Log10 (SpectralSpread (Integration (Sqrt (Hamming (HpFilter (Normalize (Sqrt (Hamming (HpFilter (Normalize (Testwav), 505.0)))), 158.0))))))), 7.0)	1,527088927	0,316864796
Power (Log10 (Abs (SpectralRolloff (Triangle (Integration (BpFilter (Integration (BpFilter (Normalize (BpFilter (Normalize (Testwav), 61.0, 4444.0)), 532.0, 5373.0)), 454.0, 1268.0))))))), 5.0)	1,677323455	0,30821975
Square (Log10 (Zcr (Autocorrelation (HpFilter (Normalize (Testwav), 814.0)))))	1,149283171	0,308012097
Arcsin (Square (Zcr (BpFilter (Normalize (BpFilter (Sqrt (Hamming (Derivation (Normalize (Hamming (Derivation (Correlation (Hamming (Derivation (Testwav)), Testwav))))))), 384.0, 1140.0)), 384.0, 1140.0))))	1,906606626	0,304459925
SpectralRolloff (BpFilter (Normalize (Testwav), 230.0, 915.0))	1,386474464	0,303074424
Power (Rms (SpectralKurtosis (Derivation (Hamming (Split (Testwav, 468.0))))), -1.0)	1,847915899	0,272589436
Log10 (Max (Sqrt (SpectralSkewness (BpFilter (Hanning (Split (LpFilter (Testwav, 769.0), 359.0)), 386.0, 3219.0)))))	1,612682099	0,222043408
Power (SpectralSpread (BpFilter (Normalize (Testwav), 142.0, 4314.0)), -2.5)	0,439719234	0,21445885
Square (Arcsin (Log10 (Range (Zcr (FilterBank (Hann (Normalize (Testwav)), 2.0))))))	1,102629339	0,210658329
Log10 (Abs (Sum (SpectralSkewness (BpFilter (Bartlett (Split (Testwav, 1514.0)), 1.0, 1324.0)))))	0,631652476	0,208094792
Arcsin (Percentile (SpectralSpread (Triangle (Split (Hann (Normalize (Testwav)), 20.0))), 32.0))	1,249253041	0,202720658
Log10 (Percentile (Holes (SpectralSpread (Triangle (Split (Square (Testwav), 20.0)))), 47.0))	1,497528845	0,188410708
Sqrt (Min (Mfcc (Normalize (Testwav), 3.0)))	1,619146528	0,182037209
Sqrt (Percentile (Mfcc (Normalize (Testwav), 3.0), 22.0))	1,619146528	0,182037209
Square (SpectralRolloff (LpFilter (Testwav, 2277.0)))	0,504628743	0,181767205
Sqrt (Median (Min (Mfcc (Hanning (Split (Normalize (Testwav), 961.0)), 2.0))))	1,375882412	0,177173722
SpectralCentroid (Normalize (LpFilter (Normalize (BpFilter (Normalize (Testwav), 3611.0, 196.0)), 1399.0)))	0,521355537	0,174190553
Zcr (Mean (Hann (Split (LpFilter (Normalize (Testwav), 1450.0), 350.0))))	0,911264671	0,161571163
Power (SpectralSpread (Testwav), -1.0)	0,442067921	0,14667978
SpectralCentroid (Power (Correlation (BpFilter (Correlation (Derivation (Correlation (BpFilter (Normalize (Testwav), 434.0, 1165.0), Testwav)), BpFilter (Normalize (Testwav), 705.0, 1063.0)), 601.0, 1671.0), Testwav), 4.0))	1,73087335	0,144559583
Sqrt (Abs (Mean (SpectralDecrease (Hanning (Split (Hanning (Normalize (Testwav)), 654.0))))))	0,677563478	0,123191898
SpectralRolloff (Normalize (Testwav))	0,38874968	0,121840569
Iqr (Mfcc (Correlation (BpFilter (Correlation (Correlation (BpFilter (Normalize (Testwav), 1395.0, 4445.0), Testwav), BpFilter (Testwav, 1395.0, 4445.0)), 4.0, 665.0), Correlation (BpFilter (Normalize (Testwav), 1395.0, 4445.0), Testwav)), 4.0))	1,112771581	0,119560947
Log10 (Abs (MaxPos (MelBands (Power (Abs (Normalize (Testwav)), 3.0), 4.0))))	0,99226525	0,118545918
Power (Abs (Sum (MaxPos (MelBands (Split (Testwav, 6899.0), 4.0)))), -2.0)	1,002623848	0,118141454
Median (Min (Mfcc (Hann (Split (Testwav, 5691.0)), 4.0)))	0,745079548	0,11375321
Power (Abs (MaxPos (MelBands (Normalize (Testwav), 4.0))), -5.0)	0,989215606	0,107211554
Power (MaxPos (MelBands (Testwav, 4.0)), -5.0)	0,989215606	0,107211554
Sqrt (Arcsin (Median (Zcr (Mfcc (Split (Testwav, 4861.0), 3.0)))))	1,27748752	0,100117672

E.4 Reconnaissance de sons percussifs polyphoniques

(Etude présentée section 5.3.2)

Fonction	Perf Learn	Perf Test
Log10 (Iqr (Fft (Normalize (Correlation (Sqrt (HpFilter (Testwav, 2131.0)), Testwav)))))	2,384159346	1,9213618
Sqrt (Sum (Variance (Mfcc (Bartlett (Split (Correlation (Normalize (Testwav), Normalize (Autocorrelation (Normalize (Testwav)))), 3206.0)), 23.0))))	1,799527875	1,915454346
Sqrt (Mean (Fft (Derivation (Sqrt (Normalize (Sqrt (Bartlett (Derivation (Normalize (Hamming (Normalize (Testwav))))))))))))	1,919052198	1,637138447
Sqrt (SpectralFlatness (BpFilter (Triangle (Normalize (Testwav)), 21978.0, 644.0)))	1,511084061	1,604674132
Power (SpectralCentroid (Derivation (Correlation (Derivation (Testwav), Testwav))), -2.0)	1,979000472	1,60461812
Max (Range (Mfcc (Triangle (Split (Correlation (Autocorrelation (Triangle (Normalize (Testwav))), Testwav), 5090.0)), 20.0)))	1,682273099	1,560828995
Square (Log10 (Variance (BarkBands (Normalize (Bartlett (Sqrt (Derivation (Testwav)))), 7.0))))	2,310239056	1,514882334
Log10 (Abs (Min (Sqrt (Median (Fft (Hann (Split (Testwav, 2297.0))))))))	1,645241507	1,504805487
Abs (Log10 (Iqr (Fft (Normalize (Hann (BpFilter (Correlation (Hann (Normalize (Testwav)), BpFilter (Testwav, 13662.0, 320.0)), 1818.0, 3279.0)))))))	2,333099844	1,481484048
Log10 (Arcsin (Power (Max (Rms (Square (BpFilter (Triangle (Split (Testwav, 1234.0)), 16077.0, 17224.0)))), 3.0)))	1,316545455	1,474132228
Square (Log10 (Length (Peaks (Peaks (Sqrt (Normalize (Testwav)))))))	1,03648246	1,472988094
Log10 (Abs (Min (Power (Range (Integration (MelBands (Derivation (BpFilter (Bartlett (Split (Testwav, 188.0)), 9396.0, 18054.0)), 10.0))), -2.0))))	1,278232345	1,468501909
Sqrt (SpectralSpread (Bartlett (Correlation (Autocorrelation (Normalize (Testwav)), Testwav))))	1,250437874	1,41E+00
Log10 (Abs (SpectralKurtosis (HpFilter (Blackman (Normalize (Testwav)), 303.0))))	1,252481266	1,385497
Log10 (Abs (SpectralKurtosis (Hamming (Testwav))))	1,108947781	1,343904624
Log10 (Abs (SpectralKurtosis (Hamming (Normalize (Testwav)))))	1,108947781	1,343904624
Log10 (Abs (Min (SpectralKurtosis (Hamming (Split (Testwav, 1865.0))))))	1,282994585	1,341494629
Abs (Log10 (SpectralSkewness (Autocorrelation (BpFilter (Derivation (Bartlett (Integration (BpFilter (Derivation (Normalize (Integration (BpFilter (Derivation (Normalize (Testwav)), 9089.0, 733.0)))), 7913.0, 1267.0)))), 8462.0, 733.0)))))	1,828957261	1,311901021
SpectralRolloff (Autocorrelation (BpFilter (Bartlett (Power (Testwav, 3.0)), 3785.0, 17088.0)))	1,556285056	1,307831025
Log10 (Median (Power (Peaks (Range (HpFilter (Bartlett (Split (Testwav, 1508.0)), 12983.0))), 5.0)))	1,364275744	1,282631079
Log10 (Abs (Max (Peaks (Peaks (Mean (Fft (BpFilter (Split (Normalize (Bartlett (Correlation (Blackman (Autocorrelation (HpFilter (Testwav, 12781.0))), Testwav))), 4508.0), 5867.0, 15482.0))))))))	1,275627617	1,277051586
Abs (Median (Peaks (Integration (SpectralSkewness (Hamming (Split (Correlation (HpFilter (Sqrt (Abs (Testwav)), 3028.0), Testwav), 2141.0)))))))	1,721271923	1,265997867
Abs (SpectralSkewness (Autocorrelation (Derivation (Testwav))))	1,515322556	1,253475897
Square (Log10 (Sum (Integration (Sqrt (Variance (HpFilter (Bartlett (Split (Testwav, 1496.0)), 16072.0)))))))	1,347473525	1,210831705
Power (Log10 (Length (Holes (Normalize (Testwav)))), 3.0)	1,330090162	1,200658493
Sqrt (Percentile (SpectralCentroid (Blackman (Split (Derivation (Normalize (Testwav)), 75.0))), 33.0))	1,387365583	1,194801944
Log10 (Max (SpectralCentroid (BpFilter (Hann (SplitOverlap (Testwav, 1891.0, 0.05578164237666061)), 12667.0, 798.0))))	1,504522702	1,183702091
Log10 (Abs (SpectralDecrease (Autocorrelation (HpFilter (Testwav, 5889.0)))))	1,33485301	1,174254651
Power (Log10 (Multiplication (Length (Peaks (Normalize (Testwav))), 7570.0)), 4.0)	1,333608809	1,161176445
Square (Log10 (Min (SpectralKurtosis (Hamming (Split (Testwav, 1739.0))))))	1,373279237	1,159961999
Square (Log10 (Abs (Length (Square (Derivation (Holes (Normalize (Testwav))))))))	1,336418516	1,153739806
Square (Log10 (Length (Holes (Testwav))))	1,336418516	1,153739806
Square (Log10 (Median (SpectralDecrease (HpFilter (Bartlett (Split (Testwav, 1517.0)), 5559.0)))))	1,53962523	1,126797104
Square (Log10 (SpectralKurtosis (LpFilter (Normalize (Testwav), 17473.0))))	0,955878867	1,119576794
SpectralSkewness (Testwav)	0,935747244	1,117739902
Abs (Log10 (Median (Fft (Derivation (Derivation (Testwav))))))	1,655824125	1,116751687
Square (Log10 (Min (SpectralKurtosis (Hamming (Split (Testwav, 1711.0))))))	1,404529186	1,111984494
Square (Log10 (Percentile (SpectralKurtosis (Hamming (Split (Normalize (Testwav), 1655.0))), 7.0)))	1,39971932	1,105497316
Square (Log10 (Rms (SpectralDecrease (HpFilter (Bartlett (Split (Normalize (Testwav), 1427.0)), 6558.0)))))	1,39574032	1,096158254
Log10 (Length (Holes (Testwav)))	1,319251485	1,070700261
Abs (Log10 (Multiplication (Zcr (Derivation (Normalize (Testwav))), 5895.0)))	1,318396905	1,069183234
Log10 (Zcr (Derivation (Normalize (Testwav))))	1,318396905	1,069183234
Square (Log10 (Rms (SpectralDecrease (HpFilter (Triangle (Split (Testwav, 2030.0)), 4524.0)))))	1,390125716	1,066894637
Square (Log10 (Log10 (Sum (SpectralCentroid (Blackman (FilterBank (Autocorrelation (Abs (Normalize (Testwav))), 10.0)))))))	1,507857079	1,059077364
Power (Log10 (Log10 (Mean (Min (Autocorrelation (Power (Fft (Bartlett (Split (Testwav, 1712.0))), -1.0))))))), 4.0)	1,421405738	1,052823353
Abs (SpectralSkewness (Normalize (BpFilter (Normalize (Testwav), 312.0, 13434.0))))	1,410384346	1,049641445
Log10 (Min (Sqrt (Hamming (BarkBands (Normalize (Testwav), 6.0)))))	1,131082865	1,044004884
Abs (SpectralSkewness (BpFilter (Normalize (Testwav), 8863.0, 218.0)))	1,360372408	1,036567699
Log10 (Mean (SpectralKurtosis (Hamming (Split (BpFilter (Testwav, 605.0, 9429.0), 3153.0)))))	2,035227778	1,006684394
Square (Log10 (Rms (SpectralKurtosis (Hamming (Split (Testwav, 2747.0))))))	1,304463531	0,971145685
Rms (Median (Holes (Mfcc (Bartlett (Split (Testwav, 1346.0)), 1.0))))	1,435830816	0,968373227
Abs (SpectralSkewness (Blackman (BpFilter (Blackman (Normalize (Testwav)), 814.0, 9619.0))))	1,337999408	0,966598387
Log10 (RHF (Square (Derivation (Correlation (Derivation (Hanning (Sqrt (BpFilter (Normalize (Sqrt (BpFilter (Normalize (Testwav), 740.0, 14847.0))), 703.0, 13507.0)))), Testwav)))))	1,604426686	0,955289864
Abs (Min (SpectralSkewness (Hamming (Split (BpFilter (Bartlett (Testwav), 556.0, 12493.0), 1523.0)))))	1,62041197	0,936522027
Log10 (Abs (Log10 (Zcr (Derivation (Normalize (Abs (Normalize (Derivation (Power (Testwav, 5.0))))))))))	1,642925308	0,907772729
Sum (Rms (Mfcc (Bartlett (Split (Testwav, 4199.0)), 1.0)))	1,033030441	0,882495938
Square (Log10 (Sum (SpectralKurtosis (Hamming (Split (Normalize (Testwav), 2446.0))))))	1,317927745	0,826257102
Sqrt (Variance (Mfcc (Square (Derivation (Autocorrelation (HpFilter (Normalize (Sqrt (Normalize (Testwav))), 284.0)))), 3.0)))	1,765335139	0,80779452
Log10 (Abs (Log10 (Min (Iqr (Fft (Bartlett (Split (Testwav, 2224.0))))))))	1,411204989	0,785326298
Abs (Percentile (SpectralSkewness (BpFilter (Hamming (Split (Testwav, 2313.0)), 11354.0, 837.0)), 59.0))	1,571768674	0,714389135

Fonction	Perf Learn	Perf Test
Abs (SpectralSkewness (Normalize (BpFilter (Testwav, 11782.0, 1146.0))))	1,317476104	0,70297476
Abs (Mean (Derivation (Mfcc (Testwav, 3.0))))	1,467854663	0,655572036
Abs (Mean (SpectralSkewness (Hamming (Split (Testwav, 2562.0)))))	1,199077213	0,637621866
Square (MaxPos (Power (BarkBands (Normalize (Testwav), 3.0), -1.0)))	1,385662221	0,588121339
Square (Log10 (Abs (Max (SpectralSkewness (Hamming (Split (Normalize (Testwav), 3132.0)))))))	1,27405366	0,268643384

E.5 Classification en 4 instruments

(Etude présentée section 5.3.3)

Fonction	Perf Learn	Perf Test
Max (Abs (Normalize (HpFilter (Derivation (Testwav), 7760.0))))	13,32380867	13,15927324
Arcsin (Square (Max (Peaks (Square (Normalize (Correlation (Square (Normalize (Testwav)), Testwav)))))))	11,21274373	11,0202129
Sqrt (Sum (Mfcc (Normalize (Hamming (LpFilter (Normalize (Testwav), 2148.0))), 4.0)))	7,8684145	8,879231181
Sqrt (Mean (Mfcc (LpFilter (Hamming (Normalize (Testwav)), 1394.0), 5.0)))	8,22769848	8,589425867
Sqrt (Sum (Mfcc (Hanning (BpFilter (Integration (Hann (Integration (Normalize (Testwav)))), 6082.0, 873.0)), 5.0)))	6,593728441	7,773573004
Sqrt (Sum (Sum (Mfcc (Hamming (Split (Testwav, 6079.0)), 2.0))))	5,068429267	5,932727129
Sqrt (Median (Mean (Mfcc (Hanning (Split (Normalize (Testwav), 6032.0)), 2.0))))	5,042510336	5,848810335
Square (Log10 (Abs (Sum (Mfcc (LpFilter (Normalize (Hann (LpFilter (Normalize (Testwav), 6554.0))), 797.0), 4.0)))))	5,324785115	5,355281491
Log10 (Abs (Log10 (Max (Square (Normalize (Testwav))))))	7,240454076	5,242732263
Sqrt (Percentile (Integration (Max (Mfcc (Hamming (FilterBank (Testwav, 2.0)), 2.0))), 1.0))	4,48592248	5,104706644
Sqrt (Mean (Mfcc (Correlation (Normalize (Testwav), Testwav), 4.0)))	4,90790676	4,643898489
Power (Log10 (SpectralCentroid (BpFilter (Hann (LpFilter (Normalize (Testwav), 7386.0)), 31.0, 169.0))), 8.0)	4,026231509	4,340808525
Mean (Mfcc (Hamming (Normalize (Testwav)), 4.0))	3,518488721	3,629812196
Power (Log10 (Mean (SpectralCentroid (Bartlett (SplitOverlap (Derivation (LpFilter (Normalize (LpFilter (Testwav, 94.0)), 291.0)), 5659.0, 0.12374923596951504))))), 6.0)	3,786894882	3,559089956
Power (Percentile (Mfcc (Triangle (Derivation (Testwav)), 5.0), 39.0), 3.0)	3,539208934	3,498583882
Sqrt (Max (Mfcc (Hamming (Normalize (Testwav)), 3.0)))	3,367036178	3,469962257
Sqrt (Max (Mfcc (Hann (Normalize (Testwav)), 3.0)))	3,322757829	3,466337698
Power (Log10 (SpectralCentroid (BpFilter (Hann (Normalize (Hann (Normalize (Testwav)))), 3.0, 159.0))), 6.0)	3,372665502	3,332640244
Log10 (Abs (Percentile (MaxPos (Power (BarkBands (LpFilter (Triangle (Split (Testwav, 8910.0)), 7440.0), 10.0), -0.5)), 92.0)))	2,882704856	2,848598271
Max (Mfcc (Hamming (Normalize (Testwav)), 3.0))	2,664434484	2,633782606
Sum (Mfcc (Normalize (Testwav), 7.0))	2,613715516	2,595095615
Sqrt (Arcsin (Log10 (SpectralSpread (Testwav))))	2,231514627	2,345807639
Abs (Sum (SpectralCentroid (LpFilter (Hanning (Split (Normalize (Testwav), 5906.0)), 283.0))))	2,083999195	2,158575271
Square (Arcsin (Power (Max (MaxPos (Mfcc (Triangle (Split (Testwav, 14683.0)), 19.0))), -1.0)))	2,824389985	2,154145217
Power (Mean (MaxPos (Mfcc (Bartlett (Split (Normalize (Testwav), 9395.0)), 16.0))), -1.0)	2,788594469	2,150264497
Arcsin (Power (SpectralKurtosis (Hamming (LpFilter (Testwav, 4621.0))), -1.0))	2,316581477	2,063144523
SpectralCentroid (Normalize (BpFilter (Hamming (Testwav), 5337.0, 17.0)))	2,090525827	2,062737789
Abs (Median (SpectralCentroid (Triangle (FilterBank (Testwav, 2.0)))))	2,033880956	2,025714826
SpectralCentroid (Bartlett (LpFilter (Testwav, 4720.0)))	2,049689802	2,017955065
Power (Log10 (Max (SpectralCentroid (FilterBank (Hanning (LpFilter (Normalize (Testwav), 3845.0)), 1.0)))), 6.0)	2,021098207	2,012662324
Max (SpectralCentroid (Hamming (Split (LpFilter (Testwav, 5844.0), 3392.0))))	2,077276326	1,973058749
Power (Log10 (Max (SpectralCentroid (Hamming (Split (LpFilter (Normalize (Testwav), 5901.0), 2424.0))))), 5.0)	2,029051081	1,907084373
Square (Rms (Sqrt (SpectralCentroid (Triangle (Split (Testwav, 831.0))))))	1,942796983	1,876761577
Power (Log10 (Abs (Percentile (SpectralCentroid (LpFilter (Hanning (Split (Normalize (Testwav), 1004.0)), 5236.0)), 49.0))), 4.0)	1,97597029	1,875843694
Power (Log10 (Multiplication (SpectralCentroid (Bartlett (LpFilter (Normalize (Testwav), 4922.0))), 2714.0)), 6.0)	1,871046916	1,856537544
Power (Log10 (SpectralKurtosis (Normalize (LpFilter (Normalize (LpFilter (Normalize (LpFilter (Normalize (LpFilter (Hann (Testwav), 6264.0)), 6264.0)), 6264.0)), 6264.0))))), -1.0)	2,187896603	1,823417457
Log10 (Min (SpectralSpread (Hanning (Split (Normalize (Testwav), 2119.0)))))	2,014108942	1,813837795
SpectralCentroid (Hamming (Testwav))	1,884566638	1,81350076
SpectralCentroid (Normalize (Hamming (Normalize (Testwav))))	1,884566638	1,81350076
Abs (Median (SpectralCentroid (Hanning (Split (Testwav, 964.0)))))	1,99960807	1,812846458
SpectralCentroid (Hann (Normalize (Testwav)))	1,883575398	1,808418956
SpectralCentroid (Hann (Testwav))	1,883575398	1,808418956
Sum (SpectralCentroid (Hanning (Split (Testwav, 5482.0))))	1,889084729	1,80465889
Min (SpectralCentroid (Hann (Split (Normalize (Testwav), 938.0))))	2,001502318	1,802529293
SpectralCentroid (Bartlett (Normalize (Testwav)))	1,867778744	1,800420087
Iqr (Bartlett (Mfcc (LpFilter (Normalize (Testwav), 2211.0), 7.0)))	1,828878535	1,791993037
Min (SpectralCentroid (Hann (Split (Normalize (Testwav), 3636.0))))	1,903124988	1,791245523
Abs (SpectralCentroid (Normalize (Testwav)))	1,830959617	1,774553378
SpectralCentroid (Normalize (Testwav))	1,830959617	1,774553378
SpectralCentroid (Normalize (Bartlett (Hamming (Normalize (Testwav)))))	1,871834907	1,767276009
Log10 (Rms (SpectralSpread (Hanning (Split (Normalize (Testwav), 2148.0)))))	1,962356865	1,76715353
Sqrt (SpectralCentroid (Normalize (Testwav)))	1,829631092	1,753078971
Min (SpectralCentroid (Hann (Split (Testwav, 973.0))))	2,008957852	1,752223603
Sqrt (Rms (SpectralSpread (Hanning (SplitOverlap (Testwav, 1707.0, 0.8022577438067602)))))	1,911884167	1,7189113
Log10 (SpectralSkewness (Normalize (Testwav)))	1,782406662	1,704235048
Power (Abs (SpectralKurtosis (Normalize (Testwav))), -1.0)	1,699362366	1,688384975
Power (Abs (SpectralKurtosis (Hann (Testwav))), -1.0)	1,835218743	1,594506119

E.6 Discrimination en 4 genres musicaux

(Etude présentée section 5.3.4)

Fonction	Perf Learn	Perf Test
Log10 (Range (Bandwidth (LpFilter (Split (Bartlett (Normalize (Testwav)), 120.0), 3548.0), 31.0)))	3,138336523	3,462805941
Square (Log10 (Max (Bandwidth (Square (Triangle (Split (Testwav, 487.0))), 6.0))))	2,963798865	3,027260098
Square (Log10 (Rms (Derivation (SpectralDecrease (Hanning (Split (Normalize (Testwav), 194.0)))))))	2,300318326	2,898765672
Power (Min (SpectralRolloff (Hamming (Split (Testwav, 231.0)))), -1.0)	2,814333165	2,802106977
Log10 (Variance (Bandwidth (Autocorrelation (Bartlett (Split (Testwav, 263.0))), 67.0)))	3,350193968	2,757142229
Log10 (Range (SpectralSpread (Hann (Split (Blackman (Testwav), 153.0)))))	3,180408094	2,656121997
Log10 (Variance (Bandwidth (Bartlett (Split (Testwav, 413.0)), 34.0)))	3,180327541	2,531284496
Sqrt (Min (Derivation (Bandwidth (Triangle (Split (Testwav, 709.0)), 37.0))))	2,950860446	2,494062036
Square (Log10 (Sum (Bandwidth (Bartlett (Split (Correlation (Normalize (Testwav), Testwav), 6987.0)), 87.0))))	2,077093576	2,444143212
Log10 (Max (Sum (Split (Testwav, 82.0))))	2,639440577	2,263538521
Log10 (Rms (Bandwidth (Blackman (Split (Power (Normalize (Testwav), -1.0), 65.0)), 39.0)))	2,141299722	2,028671327
Log10 (Range (Range (LpFilter (Bartlett (Split (Testwav, 2065.0)), 58.0))))	2,769855276	1,929633662
Log10 (Min (SpectralRolloff (Hamming (Split (Testwav, 1003.0)))))	2,16007539	1,916506939
Sqrt (Min (Bandwidth (Bartlett (Split (Correlation (Normalize (Testwav), Testwav), 6693.0)), 40.0)))	2,61631392	1,85342726
Multiplication (Multiplication (Zcr (LpFilter (Correlation (Hann (Testwav), Testwav), 587.0)), 8673.0), 7.0)	0,829539245	1,796241588
Power (Zcr (Holes (Triangle (Testwav))), 3.0)	1,81475663	1,753404463
Log10 (SpectralSpread (Sqrt (Autocorrelation (Normalize (Testwav)))))	1,775960283	1,73691377
Arcsin (Zcr (Holes (Hanning (Normalize (Testwav)))))	1,492194158	1,602095868
Log10 (Abs (Iqr (Bandwidth (Hann (Split (Testwav, 759.0)), 18.0))))	1,845226178	1,507530631
Arcsin (Square (Zcr (Peaks (Hann (Hann (Bartlett (Bartlett (Hann (Hann (Bartlett (Hann (Hamming (Hann (Hann (Bartlett (Testwav))))))))))))))))	2,524743837	1,47291353
Rms (SpectralDecrease (Autocorrelation (Split (Hann (Normalize (Testwav)), 3660.0))))	1,544118674	1,461198873
Power (Log10 (Percentile (Envelope (Bandwidth (Triangle (Split (Testwav, 354.0)), 20.0), 1863.0), 18.0)), -1.0)	2,07600258	1,331489058
Log10 (Rms (SpectralDecrease (Hamming (Split (Abs (Derivation (Normalize (Testwav))), 13499.0)))))	1,596465804	1,250524629
Multiplication (Arcsin (Arcsin (Zcr (Holes (Peaks (Testwav))))), 5118.0)	1,278107695	1,097196469
Power (Log10 (Variance (SpectralSpread (Hanning (Split (Testwav, 807.0))))), 5.0)	1,105187097	1,063622362
Range (Bartlett (Autocorrelation (Testwav)))	0,778589074	1,001223345
Log10 (Abs (SpectralRolloff (HpFilter (Triangle (Testwav), 2714.0))))	1,059565094	0,957708592
Power (Abs (Log10 (Max (Bartlett (Fft (Testwav))))), -1.0)	0,681344511	0,680429502
Sqrt (Min (HFC (Split (Testwav, 553.0))))	0,594450703	0,611118664
MaxPos (Derivation (Fft (Testwav)))	0,392525878	0,609588957
SpectralFlatness (Integration (Testwav))	0,633852086	0,559560452
Power (Min (SpectralSpread (Split (Testwav, 4941.0))), 3.0)	0,365806242	0,5234443
dB (Square (Variance (SpectralRolloff (RemoveSilentFrames (Abs (Multiplication (Split (Testwav, 3806.0), 4933.935)))))))	0,321051907	0,493141912
Power (Log10 (Abs (SpectralRolloff (Square (HpFilter (Testwav, 4222.0))))), 3.0)	0,666825599	0,453844037
Power (Zcr (Peaks (Hann (Hann (Bartlett (Hann (Hamming (Bartlett (Hann (Bartlett (Hann (Hann (Bartlett (Hann (Hann (Bartlett (Hann (Hamming (Hann (Hann (Bartlett (Testwav))))))))))))))))))))), -2.0)	0,358267824	0,395566655
Power (Log10 (Abs (SpectralRolloff (Hanning (Envelope (Testwav, 720.0))))), 5.0)	0,334464596	0,337822564
Range (Derivation (Testwav))	0,258746723	0,334049849
Log10 (Abs (Range (Derivation (Testwav))))	0,214098996	0,310414699
dB (Square (Median (Variance (Envelope (Split (LpFilter (Multiplication (Testwav, 161.47273), 737.0), 3035.0), 8834.0)))))	0,361845108	0,301875915
Zcr (Peaks (Hann (BpFilter (Testwav, 1400.0, 632.0))))	0,324311115	0,25170396
Square (SpectralSpread (Triangle (Envelope (Testwav, 9325.0))))	0,408227212	0,232325102
Arcsin (Square (Zcr (Peaks (Hann (Hann (Bartlett (Bartlett (Hann (Hann (Bartlett (Hann (Hamming (Hann (Hann (BpFilter (Testwav, 3928.0, 892.0))))))))))))))))	0,195010769	0,223259781
Multiplication (Zcr (Hann (Hann (Holes (LpFilter (Testwav, 774.0))))), 6416.0)	0,344743888	0,20992268
Abs (Log10 (Abs (SpectralRolloff (Square (Envelope (Testwav, 1891.0))))))	0,203669831	0,19565663
Log10 (Median (Integration (MaxPos (Split (Autocorrelation (Power (Hanning (Testwav), 5.0)), 6783.0)))))	0,159237887	0,174311665
Range (Sum (Square (Triangle (Split (Abs (Testwav), 8377.0)))))	0,281623192	0,139713855

E.7 Évaluation de l'énergie musicale

(Etude présentée section 5.3.5)

Fonction	Perf Learn	Perf Test
Log10 (Iqr (Power (Fft (Hanning (Hamming (Autocorrelation (Blackman (Hamming (Testwav)))))), -1.0)))	0,729610028	0,811512414
Square (Log10 (Mean (Min (Fft (Split (Testwav, 4009.0))))))	0,744522832	0,809878993
Log10 (Abs (Min (SpectralCentroid (Split (Testwav, 10826.0)))))	0,643395171	0,809878993
Square (Max (Min (Split (Testwav, 5459.0))))	0,648160554	0,800011547
Log10 (Range (Median (Abs (FilterBank (Fft (Testwav), 5.0)))))	0,716847832	0,800011547
Log10 (Range (Median (Abs (FilterBank (Fft (Testwav), 5.0)))))	0,716106639	0,76376387
Log10 (Abs (Mean (Range (Sqrt (Triangle (SplitOverlap (Derivation (Derivation (Testwav)), 4611.0, 0.0))))))))	0,728332405	0,76376387
Log10 (Abs (Sum (Abs (Min (Bartlett (Split (Derivation (Derivation (Testwav)), 2574.0)))))))	0,710103332	0,75830769
Sqrt (Rms (Min (Split (BpFilter (Sqrt (Derivation (Normalize (Testwav)))), 4162.0, 1788.0), 1770.0))))	0,70627298	0,75830769
Square (Log10 (Abs (Mean (Peaks (Sum (Square (MelBands (Derivation (Hamming (Split (Testwav, 4300.0))))), 10.0))))))))	0,656977652	0,75585584
Square (Log10 (Rms (Sum (Fft (HpFilter (Triangle (Split (Derivation (Derivation (Testwav)), 1947.0)), 8.0)))))))	0,716269029	0,75585584
Power (Log10 (Median (Sum (BarkBands (Blackman (Split (Normalize (Testwav), 12526.0)), 15.0)))), 5.0)	0,649971192	0,744813432

Fonction	Perf Learn	Perf Test
Square (Log10 (Abs (Min (Variance (Derivation (BarkBands (Triangle (FilterBank (Testwav, 18.0)), 7.0)))))))	0,668136617	0,744813432
Square (Sum (Min (Split (Sqrt (Normalize (Testwav)), 5453.0))))	0,641056128	0,741237773
Log10 (Percentile (Max (Integration (HpFilter (Triangle (Split (Testwav, 10873.0)), 3486.0))), 10.0))	0,714248076	0,741237773
Log10 (Abs (Median (Range (Hanning (Split (Derivation (Derivation (Testwav)), 6605.0))))))	0,699548666	0,736219422
Power (Log10 (Median (Sum (BarkBands (Blackman (Split (Normalize (Testwav), 12526.0)), 15.0)))), 5.0)	0,649971192	0,736219422
Square (Log10 (Mean (Min (Fft (Split (Testwav, 4009.0))))))	0,744522832	0,735219251
Sqrt (Min (Range (Triangle (Split (BpFilter (Testwav, 3135.0, 3020.0), 10728.0)))))	0,688905265	0,735219251
Log10 (Abs (Median (Range (Hanning (Split (Derivation (Derivation (Testwav)), 6605.0))))))	0,699548666	0,733590998
Sqrt (Median (Max (Derivation (Split (Testwav, 5357.0)))))	0,696026264	0,733590998
Log10 (Percentile (Max (Integration (HpFilter (Triangle (Split (Testwav, 10873.0)), 3486.0))), 10.0))	0,714248076	0,730391608
Log10 (Percentile (Abs (Max (Derivation (LpFilter (Hamming (Split (Testwav, 10366.0)), 5241.0)))), 9.0))	0,696654389	0,730391608
Log10 (Mean (Rms (HpFilter (Blackman (Split (Testwav, 8103.0)), 4592.0))))	0,685195736	0,726673111
Power (Mean (Min (Derivation (Hann (Split (Normalize (Testwav), 3757.0))))), -1.0)	0,631205502	0,726673111
Sqrt (Min (Range (Triangle (Split (BpFilter (Testwav, 3135.0, 3020.0), 10728.0)))))	0,688905265	7,22E-01
Sqrt (Median (Max (Derivation (Split (Testwav, 5357.0)))))	0,696026264	0,721686968
Min (Max (Split (Square (Testwav), 6388.0)))	0,646729012	0,714188252
Min (Zcr (Split (Derivation (Square (Testwav)), 5570.0)))	0,554754779	0,712053095
Log10 (Abs (Sum (Abs (Min (Bartlett (Split (Derivation (Derivation (Testwav)), 2574.0)))))))	0,710103332	0,709390682
Sqrt (Rms (Min (Split (BpFilter (Sqrt (Derivation (Normalize (Testwav))), 4162.0, 1788.0), 1770.0))))	0,70627298	0,709390682
Square (Log10 (Rms (Sum (Fft (HpFilter (Triangle (Split (Derivation (Derivation (Testwav)), 1947.0)), 8.0))))))	0,716269029	0,703842069
Square (Log10 (Abs (Mean (Peaks (Sum (Square (MelBands (Derivation (Hamming (Split (Testwav, 4300.0))), 10.0))))))))	0,656977652	0,703842069
Log10 (Rms (BarkBands (Hann (Derivation (BpFilter (Normalize (Testwav), 4127.0, 2845.0))), 2.0)))	0,626177591	0,699067983
Iqr (Derivation (SpectralFlatness (Blackman (Split (Testwav, 771.0)))))	0,548516734	0,699067983
Square (Sum (Min (Split (Sqrt (Normalize (Testwav)), 5453.0))))	0,641056128	0,690533966
Log10 (Percentile (Abs (Max (Derivation (LpFilter (Hamming (Split (Testwav, 10366.0)), 5241.0)))), 9.0))	0,696654389	0,690533966
Square (Max (Min (Split (Testwav, 5459.0))))	0,648160554	0,686596647
Arcsin (Rms (Sqrt (Hamming (Range (SplitOverlap (Normalize (Testwav), 6355.0, 0.45684660737790395))))))	0,608234891	0,686596647
Log10 (Mean (Rms (HpFilter (Blackman (Split (Testwav, 8103.0)), 4592.0))))	0,685195736	0,678203729
Arcsin (Rms (Sqrt (Hamming (Range (SplitOverlap (Normalize (Testwav), 6355.0, 0.4568466073779039))))))	0,608234891	0,678203729
Iqr (Derivation (SpectralFlatness (Blackman (Split (Testwav, 771.0)))))	0,548516734	0,676866702
Log10 (Rms (BarkBands (Hann (Derivation (BpFilter (Normalize (Testwav), 4127.0, 2845.0))), 2.0)))	0,626177591	0,676866702
Log10 (Abs (Mean (Range (Sqrt (Triangle (SplitOverlap (Derivation (Derivation (Testwav)), 4611.0, 0.0)))))))	0,728332405	0,647716849
Power (Mean (Min (Derivation (Hann (Split (Normalize (Testwav), 3757.0))))), -1.0)	0,631205502	0,647716849
Min (Zcr (Split (Derivation (Derivation (Square (Derivation (Square (Autocorrelation (Autocorrelation (Sqrt (Abs (Testwav))))))))), 34164.0)))	0,612368346	0,647714035
Power (Abs (Log10 (Iqr (Derivation (Fft (Blackman (Testwav)))))), 5.0)	0,555323775	0,617837465
Log10 (Abs (Min (SpectralCentroid (Split (Testwav, 10826.0)))))	0,643395171	0,600984451
Square (Log10 (Abs (Min (Variance (Derivation (BarkBands (Triangle (FilterBank (Testwav, 18.0)), 7.0)))))))	0,668136617	0,600984451
Min (Zcr (Split (Derivation (Derivation (Square (Derivation (Derivation (Square (Autocorrelation (Sqrt (Testwav))))))))), 32390.0)))	0,488148568	0,532550735
Sum (Integration (Sqrt (RHF (Split (Envelope (Derivation (Autocorrelation (Testwav)), 6640.0), 2839.0)))))	0,527409933	0,526383057
Sqrt (Min (Hamming (Testwav)))	0,412615088	0,487767944
Sum (SpectralDecrease (Split (Sqrt (Multiplication (Testwav, 19880.11)), 3360.0)))	0,419596932	0,48123192
Rms (SpectralDecrease (Split (Sqrt (Testwav), 4233.0)))	0,422433501	0,480867061
Log10 (SpectralSkewness (Derivation (Sqrt (LpFilter (Testwav, 594.0)))))	0,401362461	0,457529023
Flatness (Rms (Split (Derivation (Sqrt (Autocorrelation (Testwav))), 4781.0)))	0,417433446	0,408331308
Min (Max (Split (Derivation (Derivation (Square (Derivation (Power (Testwav, 8.0))))), 6857.0)))	0,442153759	0,398437055
Min (Zcr (Split (Derivation (Derivation (Square (Derivation (Square (Derivation (Testwav)))))), 34164.0)))	0,443730206	0,376945609
Zcr (Derivation (Max (Power (Split (Hann (Testwav), 741.0), 3.0))))	0,158750665	0,376825609
Min (Min (Fft (Split (Derivation (Testwav), 8800.0))))	0,336517653	0,374050629
Min (Zcr (Split (Derivation (HpFilter (Testwav, 4341.0)), 21262.0)))	0,237988322	0,366175578
Variance (Min (Split (LpFilter (Testwav, 864.0), 6564.0)))	0,455085364	0,340838518
Sum (Integration (Variance (Derivation (Split (Envelope (LpFilter (Testwav, 135.0), 1942.0), 1031.0)))))	0,228791078	0,319745865
Power (Iqr (HpFilter (Testwav, 2216.0)), 4.0)	0,371849517	0,314018399
Zcr (Derivation (Variance (Split (Multiplication (Sqrt (Testwav), 455720.12), 2561.0))))	0,366629826	0,304998353
Min (Max (Derivation (Split (Sqrt (Envelope (Testwav, 1681.0)), 4849.0))))	0,371453607	0,297990906
Flatness (Rms (Split (Derivation (Power (Derivation (Envelope (Autocorrelation (Testwav), 7546.0)), 5.0)), 6164.0)))	0,1424044	0,275044598
Square (Log10 (SpectralRolloff (Envelope (Square (Hann (BpFilter (Testwav, 648.0, 2752.0))), 6662.0))))	0,25356851	0,272996368
Min (Max (Derivation (Split (Sqrt (Envelope (Autocorrelation (Testwav), 1681.0)), 9090.0))))	0,107090228	0,249677451
Max (Multiplication (Integration (Rms (Derivation (Derivation (Split (Envelope (Testwav, 9601.0), 6666.0))))), 111557.0))	0,21938708	0,23811202
Min (Sum (Fft (Split (Derivation (Envelope (Testwav, 9320.0)), 5456.0))))	0,145658801	0,234041816
Min (Max (Power (Split (Testwav, 266.0), 4.0)))	0,381671634	0,225845538
Flatness (Rms (Split (Integration (Square (Testwav)), 9986.0)))	0,185004996	0,211222221
Zcr (Derivation (Square (RHF (Split (Abs (Hann (Testwav)), 4880.0)))))	0,223361497	0,199814293
Log10 (SpectralKurtosis (Blackman (Blackman (BpFilter (Testwav, 2058.0, 974.0)))))	0,320150798	0,179046604
Power (MaxPos (Testwav), -1.0)	0,185160861	0,154344751
Min (Zcr (Split (Derivation (Derivation (Square (Envelope (Testwav, 4936.0)))), 25634.0)))	0,247430445	0,138610516
Min (Zcr (Split (Derivation (Derivation (Envelope (Testwav, 4151.0))), 12215.0)))	0,252587019	0,132397238

BIBLIOGRAPHIE

[Allamanche *et al.*, 2003] E. Allamanche, J. Herre, O. Hellmuth, T. Kastner, et C. Ertel, A multiple feature model for musical similarity retrieval, In *Proc. International Conference on Music Information Retrieval, ISMIR03*, 2003.

[AllMusicGuide, 2004] AllMusicGuide, http ://www.allmusic.com, 2004.

[Allwein *et al.*, 2000] E.L. Allwein, R.E. Schapire, et Y. Singer, Reducing multiclass to binary : a unifying approach for margin classifiers, In *Proc. of the 17th International Conference on Machine Learning*, éditeur CA Morgan Kaufman, San Francisco, pages 9–16, 2000.

[Alonso *et al.*, 2003a] M. Alonso, R. Badeau, B. David, et G. Richard, Musical tempo estimation using noise subspace projections, In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA03)*, New Paltz, New York, Oct 2003.

[Alonso *et al.*, 2003b] M. Alonso, B. David, et G. Richard, A study of tempo tracking algorithms from polyphonic music signals, In *4th COST 276 Workshop*, France, Mar 2003.

[Aucouturier et Pachet, 2002a] J.J Aucouturier et F. Pachet, Music similarity measures : What's the use ?, In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR02)*, éditeur Ircam, 2002.

[Aucouturier et Pachet, 2002b] J.J Aucouturier et F. Pachet, Scaling up music playlist generation, In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME02)*, numéro 1, Lausanne, Switzerland, August 2002.

[Aucouturier et Pachet, 2003] J.J Aucouturier et F. Pachet, Musical genre : a survey, *Journal of New Music Research*, 32(1), 2003.

[Aucouturier et Sandler, 2001] Jean-Julien Aucouturier et Mark Sandler, Segmentation of musical signals using hidden markov models, In *Proceedings of the 110th Audio Engineering Society Convention*, Amsterdam, The Netherlands, 2001.

[Bartsch et Wakefield, 2004] M. Bartsch et G. Wakefield, Singing voice identification using spectral envelope estimation, *IEEE Transactions on Speech and Audio Processing*, 2004.

[Beasley *et al.*, 1993a] D. Beasley, D.R. Bull, et R.R. Martin, An overview of genetic algorithms : Part 1, fundamentals, *University Computing*, 15(2) :58–69, 1993.

[Beasley *et al.*, 1993b] D. Beasley, D.R. Bull, et R.R. Martin, An overview of genetic algorithms : Part 2, research topics, *University Computing*, 15(4) :170–181, 1993.

[Bello *et al.*, 2002] J.P. Bello, L. Daudet, et M. Sandler, Time domain polyphonic transcription using self generating databases, In *Proceedings of the*

112th Audio Engineering Society Convention (AES02), Munich, Germany, May 2002.

[Bentley, 1975] J.L. Bentley, Multi-dimensional binary search trees used for associated searching, *Communications of the ACM*, 18 :509–517, 1975.

[Berenzweig *et al.*, 2002] A.L. Berenzweig, D.P.W. Ellis, et S. Lawrence, Using voice segments to improve artist classification of music, In *Proc. of the AES 22nd International Conference*, Espoo, Finland, June 2002.

[Berenzweig *et al.*, 2003] A. Berenzweig, B. Logan, D.P.W. Ellis, et B. Whitman, A large-scale evalutation of acoustic and subjective music similarity measures, In *Proceedings of the 2003 International Symposium on Music Information Retrieval*, Baltimore, MD, October 2003.

[Berenzweig et Ellis, 2001] A.L. Berenzweig et D.P.W. Ellis, Locating singing voice segments within music signals, In *IEEE Workshop on Applications of Signal Processing to Acoustics and Audio (WASPAA01)*, Mohonk NY, October 2001.

[Bishop, 1995] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford Press, 1995.

[Blum et Langley, 1997] A. Blum et P. Langley, Selection of relevant features and examples in machine learning, *Artificial Intelligence*, 97 :245–271, 1997.

[Bregman, 1990] A.S. Bregman, *Auditory scene analysis*. MIT Press, 1990.

[Brown *et al.*, 2001] J.C. Brown, O. Houix, et S. McAdams, Feature dependence in the automatic identification of musical woodwind instruments, *Journal of the Acoustical Society of America*, 109(3) :1064–1072, 2001.

[Brown, 1991] J.C. Brown, Musical frequency tracking using the methods of conventional and narrowed autocorrelation, *J. Acoust. Soc. Am.*, 89(5), 1991.

[Brown, 1992] J.C. Brown, Musical fundamental frequency tracking using a pattern recognition method, *J. Acoust. Soc. Am.*, pages 1394–1402, 1992.

[Brown, 1993] J.C. Brown, Determination of the meter of musical scores by autocorrelation, *Journal of the Acoustical Society of America*, 94(4) :1953–1957, 1993.

[Brown, 1997] J.C. Brown, Computer identification of musical instruments using pattern recognition, 1997.

[Burges, 1998] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998.

[Carey et Lloyd-Thomas, 1999] E.S. Carey, M.J. ans Parris et H. Lloyd-Thomas, A comparison of features for speech, music discrimination, *Proceedings of International Conference on Acoustics, Speech, Signal Processing, ICASSP99*, pages 1432–1436, 1999.

[CDDB, 2004] CDDB, http ://www.cddb.com, 2004.

[Chou et Gu, 2001] W. Chou et L. Gu, Robust singing detection in speech/music discriminator design, In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP01)*, pages 865–868, Salt Lake City, Utah, USA, May 2001.

[Chu et Logan, 2000] S. Chu et B. Logan, Music summary using key phrases, In *Proceedings IEEE ICASSP*, Istanbul, Turkey, 2000.

[Codognet et Diaz, 2001] P. Codognet et D. Diaz, Yet another local search method for constraint solving, In *Proceedings of the AAAI Fall 2001 Symposium*, Cape Cod, MA, November 2001.

[Cooper et Meyer, 1960] G.W. Cooper et L.B. Meyer, *The rhythmic structure of music*. University of Chicago Press, 1960.

[Dempster *et al.*, 1977] Dempster, Laird, et Rubin, Maximum likelihood from incomplete data via em algorithm, *Journal of the Royal Statistical Society*, 39(1) :1–38, 1977.

[Desain et Honing, 1999] Peter Desain et Henkjan Honing, Computational models of beat induction : The rule-based approach, *Journal of New Music Research*, 28(1) :29–42, 1999.

[Duda et Hart, 1973] R.O. Duda et P.E. Hart, *Pattern classification and scene analysis*. Wiley, New York, 1973.

[Duxbury *et al.*, 2001] C. Duxbury, M. Davies, et M. Sandler, Separation of transient information in musical audio using mutliresolution analysis techniques, In *Proceedings of the 4th COST-G6 Conference on Digital Audio Effects (DAFX01)*. University of Limerick, December 2001.

[Eggink et Brown, 2003] J. Eggink et G.J. Brown, Application of missing feature theory to the recognition of musical instruments in polyphonic audio, In *Proc. International Conference on Music Information Retrieval, ISMIR03*, 2003.

[Ellis et Bilmes, 2000] D.P.W. Ellis et J.A. Bilmes, Using mutual information to design feature combinations, In *Proc. ICSLP00*, Beijing, 2000.

[Ellis, 1996] D. Ellis, Prediction-driven computational auditory scene analysis, Master's thesis, Ph.D. Dissertation, MIT Department of Electrical Engineering and Computer Science, 1996.

[Eronen, 2001] A. Eronen, Comparison of features for musical instrument recognition, In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA01*, 2001.

[Fisher, 1923] A. Fisher, *The mathematical theory of probabilities*, volume 1. Macmillan, New York, 1923.

[Foote, 1997] J. T. Foote, Content-based retrieval of music and audio, In *SPIE*, 1997, pages 138-147.

[Foote, 2000] J. Foote, Automatic audio segmentation using a measure of audio novelty, In *Proceedings of IEEE International Conference on Multimedia and Expo*, 2000.

[Fujinaga, 1998] I. Fujinaga, Machine recognition of timbre using steady-state tone of acoustical musical instruments, In *ICMC, Ann Harbor, USA*, pages 207–210, 1998.

[Fukunaga, 1990] Keinosuke Fukunaga, *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, 1990.

[Gabrielsson, 1973] A. Gabrielsson, Similarity ratings and dimension analyses of auditory rhythm patterns, part i and ii, *Scandinavian Journal of Physiology*, 14, 1973.

[Gillet et Richard, 2004] O. Gillet et G. Richard, Automatic transcription of drum loops, In *International Conference on Acoustics, Speech, and Signal Processing ICASSP04*, Montréal, Québec, May 2004.

[Glover *et al.*, 1993] F. Glover, E. Taillard, et D. de Werra, A users guide to tabu search, *special issues of the Annals of Operations Research, J.C. Baltzer*, 41 :3–28, 1993.

[Goldberg, 1989] David E. Goldberg, Genetic algorithms in search, optimization and machine learning, 1989.

[Goto, 2001a] M. Goto, An audio based real time beat tracking system for music with or without drum sounds, *Journal of New Music Research*, 30(2) :159–171, 2001.

[Goto, 2001b] M. Goto, A predominant f0 estimation method for cd recordings - map estimation using em algorithm for adaptive tone models, In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP01)*, pages 3365–3368, 2001.

[Gouyon *et al.*, 2000] F. Gouyon, F. Pachet, et O. Delerue, On the use of zero crossing rate for an application of classification of percussive sounds, In *Proceedings of DAFX00*, 2000.

[Gouyon et Herrera, 2001] F. Gouyon et P. Herrera, Exploration of techniques for automatic labeling of audio drum tracks instruments, 2001.

[Guyon et Elisseeff, 2003] I. Guyon et A. Elisseeff, An introduction to variable and feature selection, *Journal of Machine Learning Research*, 3 :1157–1182, 2003.

[Handel, 1989] S. Handel, *Listening*. MIT Cambridge, MA, 1989.

[Hartman, 1996] Hartman, Pitch, periodicity, and auditory organization, *J. Acoust. Soc. Am.*, 6(1000) :1394–1402, 1996.

[Herrera *et al.*, 1999] Perfecto Herrera, Xavier Serra, et Geoffroy Peeters, Audio descriptors and descriptors schemes in the context of mpeg-7, In *Proceedings of the International Computer Music Conference (ICMC99)*, Bejing, China, October 1999.

[Herrera *et al.*, 2000] P. Herrera, X. Amatriain, E. Batlle, et X. Serra, Towards instrument segmentation for music content description : a critical review of instrument classification techniques, In *Proceedings of International Symposium of Music Information Retrieval*, 2000.

[Herrera *et al.*, 2002a] P. Herrera, G. Peeters, et S. Dubnov, Automatic classification of musical instrument sounds, *Journal of New Music Research*, 31(3), 2002.

[Herrera *et al.*, 2002b] Perfecto Herrera, Alexandre Yeterian, et Fabien Gouyon, Automatic classification of drum sounds : A comparison of feature selection and classification techniques, 2002.

[Holland, 1975] J.H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan, 1975.

[Holte, 1993] R.C. Holte, Very simple classification rules performing well on most commonly uset datasets, *Machine Learning*, 11 :63–90, 1993.

[Huberty, 1994] C.J. Huberty, *Applied Discriminant Analysis*. John Wiley, NY, 1994.

[Huron, 2000] D. Huron, Perceptual and cognitive applications in music information retrieval, In *Proceedings of International Symposium of Music Information Retrieval*, 2000.

[Jensen et Andersen, 2003] K. Jensen et T.H. Andersen, Real-time beat estimation using feature selection, In *Proceedings of the 1st International Sym-*

posium on Computer Music Modeling and Retrieval (CMMR03), Montpellier, France, May 2003.

[Kaminskyj et Materka, 1995] Kaminskyj et Materka, Automatic source identification of monophonic musical instrument sounds, In *IEEE International Conference on Neural Networks*, 1995.

[Karjalainen et Tolonen, 1999] M. Karjalainen et T. Tolonen, Multi-pitch and periodicity analysis model for sound separation and auditory scene analysis, In *International Conference on Acoustics, Speech and Signal Processing*, éditeur IEEE, volume 2, pages 929–932, 1999.

[Katayose *et al.*, 1988] H. Katayose, M. Imai, et S. Inokuchi, Sentiment extraction in music, In *Proc. of Intl. Conf. on Pattern Recognition*, pages 1083–1087, 1988.

[Keiler et Marchand, 2002] F. Keiler et S. Marchand, Survey on extraction of sinusoids in stationary sounds, In *Proceedings of the Digital Audio Effects (DAFx'02) Conference*, pages 51–58, Hamburg, Germany, September 2002.

[Kim et Whitman, 2002] Y.E. Kim et B. Whitman, Singer identification in popular music recordings using voice coding features, In *Proceedings of the 3rd International Symposium on Music Information Retrieval*, éditeur Ircam, 2002.

[Kim et Whitman, 2003] Y.E. Kim et B. Whitman, *Singing voice analysis-synthesis*, PhD thesis, MIT, 2003.

[Kimber et Wilcox, 1996] D. Kimber et L. Wilcox, Acoustic segmentation for audio browsers, 1996.

[Klapuri *et al.*, 2000] Klapuri, Virtanen, et Holm, Robust multipitch estimation for the analysis and manipulation of polyphonic musical signals, In *Proc. of the 3rd COST-G6 Conference on Digital Audio Effects (DAFx00)*, Verona, Italy, 2000.

[Klapuri *et al.*, 2001] A. Klapuri, T. Virtanen, A. Eronen, et J. Seppänen, Automatic transcription of music, FWO ATOM, 2001.

[Klapuri, 2000] A. Klapuri, Qualitative and quantitative aspects in the design of periodicity estimation algorithms, In *Proceedings of the European Signal Processing Conference EUSIPCO*, 2000.

[Koza, 1992] J.R. Koza, *Genetic Programming : on the programming of computers by means of natural selection.* Cambridge, MA : The MIT Press, 1992.

[Kunieda *et al.*, 1996] Kunieda, Shimamura, et Suzuki, Robust method of measurement of fundamental frequency by aclos - autocorrelation of log spectrum, In *Proc. of the IEEE Trans. on Acoust., Speech and Signal Processing*, 1996.

[La Burthe *et al.*, 2003] A. La Burthe, F. Pachet, et J.J. Aucouturier, Editorial metadata in the cuidado music browser : between universalism and autism, In *Proceedings of the Wedelmusic Conference*, 2003.

[Lambrou *et al.*, 1998] T. Lambrou, P. Kudumakis, R. Speller, et A. Linney, Classification of audio signals using statistical features on time and wavelet transform domains, In *Proc. Int. Conf. Acoustic, Speech, and Signal Processing (ICASSP-98)*, volume 6, pages 3621–3624, 1998.

[Large, 1995] E.W. Large, Beat tracking with a nonlinear oscillator, In *Working Notes of the IJCAI Workshop on AI and Music*, pages 24–31, 1995.

[Laurent *et al.*, 1998] H. Laurent, E. Hitti, et M.F. Lucas, Abrupt changes detection in the time-scale and in the time-frequency planes : a comparative study, In *Proceedings of TFTS 98, IEEE-SP International Symposium on Time-frequency and Time-scale Analysis*, pages 581–584, 1998.

[Leray et Gallinari, 1999] P. Leray et P. Gallinari, Feature selection with neural networks, *Behaviormetrika (special issue on Analysis of Knowledge Representation in Neural Network Models)*, 26(1) :145–166, jan 1999.

[Li *et al.*, 2003] T. Li, M. Ogihara, et Q. Li, A comparative study on content-based music genre classification, In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, 2003.

[Li et Ogihara, 2003] T. Li et M. Ogihara, Detecting emotion in music, In *Proceedings of 3rd International Symposium on Music Information Retrieval (ISMIR03)*, Baltimore, MD, Oct 2003.

[Liu *et al.*, 1998] Z. Liu, Y. Wang, et T. Chen, Audio feature extraction and analysis for scene segmentation and classification, *Journal of VLSI Signal*, June 1998.

[Liu *et al.*, 2003a] D. Liu, L. Lu, et H.J. Zhang, Automatic music mood detection from acoustic music data, In *Proceedings of 3rd International Symposium on Music Information Retrieval (ISMIR03)*, Baltimore, MD, Oct 2003.

[Liu *et al.*, 2003b] D. Liu, N.Y. Zhang, et H.C. Zhu, Form and modd recognition of johann strauss's waltz centos, *Chinese Journal of Electronics*, 2003.

[Logan, 2000] B. Logan, Mel frequency cepstral coefficients for music modeling, 1st International Symposium on Music Information Retrieval (ISMIR00), 2000.

[Maddage *et al.*, 2003] N.C. Maddage, C. Xu, et Y. Wang, A svm-based classification approach to musical audio, In *Proc. International Conference on Music Information Retrieval, ISMIR03*, 2003.

[Mahfoud, 1995] S.W. Mahfoud, *Niching methods for Genetic Algorithms*, PhD thesis, Illinois Genetic Algorithms Laboratory (IlliGAL Report No. 95001), 1995.

[Marchand, 1998] S. Marchand, Improving spectral analysis precision with an enhanced phase vocoder using signal derivatives, In *Proceedings of the Digital Audio Effects (DAFx98) Workshop*, pages 114–18, Barcelona, Spain, November 1998.

[Marchand, 2001] S. Marchand, An efficient pitch-tracking algorithm using a combination of fourier transforms, In *Proceedings of the Digital Audio Effects (DAFx'01) Conference*, pages 170–174, Limerick, Ireland, Dec 2001.

[Martin et Kim, 1998] K. Martin et Y. Kim, Instrument identification : a pattern-recognition approach, In *136th meeting Acoustical Society of America*, 1998.

[Martins et Ferreira, 2002] L.G.P.M. Martins et A.J.S. Ferreira, Pcm to midi transposition, In *Proceedings of the 112th AES Convention*, Munich, Germany, May 2002.

[McKinney et Breebaart, 2003] M.F. McKinney et J. Breebaart, Features for audio and music classification, In *Proceedings of 3rd International Symposium on Music Information Retrieval (ISMIR03)*, Baltimore, MD, Oct 2003.

[Montana, 1995] D. Montana, Strongly typed genetic programming, *Evolutionary computation*, 3(2) :199–230, 1995.

[MoodLogic, 2004] MoodLogic, http ://www.moodlogic.com, 2004.

[Morgan et Bourlard, 1995] N. Morgan et H. Bourlard, Continuous speech recognition : an introduction to the hybrid hmm/connectionist approach, *Signal Processing Magazine*, pages 25–42, Mai 1995.

[MusicBrainz, 2004] MusicBrainz, http ://www.musicbrainz.org, 2004.

[Orlarey *et al.*, 2002] Y. Orlarey, D. Fober, et S. Letz, An algebra for block diagram languages, In *Proceedings of International Computer Music Conference*, éditeur ICMA, pages 542–547, 2002, http ://sourceforge.net/projects/faudiostream.

[Oswald, 1989] J. Oswald, Plunderphonics, http ://www.plunderphonics.com (Fony), 1989.

[Oudeyer, 2002] P-Y. Oudeyer, The production and recognition of emotions in speech : features and algorithms, *International Journal of Human Computer Interaction*, 2002.

[Pachet *et al.*, 1999] F. Pachet, P. Roy, et D. Cazaly, A combinatorial approach to content-based music selection, In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 457–462, Firenze (It), 1999.

[Pachet *et al.*, 2000] F. Pachet, O. Delerue, et F. Gouyon, Extracting rhythm from audio signals, In *Proceedings of SONY Research Forum*, 2000.

[Pachet *et al.*, 2003] F. Pachet, A. La Burthe, J.J. Aucouturier, et A. Zils, Popular music access : the sony music browser, *Journal of American Society for Information Science*, 2003.

[Pachet *et al.*, 2004] F. Pachet, J.-J. Aucouturier, A. La Burthe, A. Zils, et A. Beurive, The cuidado music browser : an end-to-end electronic music distribution system, *Multimedia Tools and Applications*, April 2004, Special Issue on the CBMI03 Conference.

[Pachet, 2002] F. Pachet, The continuator : Musical interaction with style, In *Proceedings of ICMC*, éditeur ICMA, pages 211–218. ICMA, September 2002.

[Pachet, 2003] F. Pachet, Content management for electronic music distribution : the real issues, *Communications of the ACM*, April 2003.

[Peeters *et al.*, 2000] Geoffroy Peeters, Stephen McAdams, et Perfecto Herrera, Instrument sound description in the context of mpeg-7, In *Proceedings of the 2000 International Computer Music Conference (ICMC00)*, Berlin , Germany, Aug 2000.

[Peeters *et al.*, 2002] G. Peeters, A. La Burthe, et X. Rodet, Towards automatic music audio summary generation from signal analysis, In *Proceedings of the International Conference on Music Information Retrieval (ISMIR02)*, éditeur Ircam, 2002.

[Peeters et Rodet, 2002] G. Peeters et X. Rodet, Automatically selecting signal descriptors for sound classification, In *Proceedings of the 2002 Interna-*

tional Computer Music Conference (ICMC02), Goteborg (Sweden), Sept 2002.

[Philippe, 2000] P. Philippe, Low-level musical descriptors for mpeg7, *Signal Processing Image Communication*, 16 :181–191, 2000.

[Qi *et al.*, 2002] H. Qi, P. Hartono, K. Suzuki, et S. Hashimoto, Sound database retrieved by sound, *Acoustical Science and Technology*, 23(6) :293–300, 2002.

[Rabiner *et al.*, 1976] Rabiner, Cheng, Rosenberg, et McGonegal, A comparative study of several pitch detection algorithms, In *IEEE Trans on Acoustics Speech and Signal Processing*, 1976.

[Rossignol *et al.*, 1999] S. Rossignol, X. Rodet, J. Soumagne, J.L. Colette, et P. Depalle, Automatic characterisation of musical signals : feature extraction and temporal segmentation, *Journal of New Music Research*, 28(4) :281–295, 1999.

[Saunders, 1996] J. Saunders, Real time discrimination of broadcast speech/music discriminator, In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 993–996, 1996.

[Scheirer et Slaney, 1997] E. Scheirer et M. Slaney, Construction and evaluation of a robust multifeature speech/music discriminator, In *Proc. ICASSP'97*, 1997.

[Scheirer, 1998] Eric D. Scheirer, Tempo and beat analysis of acoustic musical signals, *J. Acoust. Soc. Am. (JASA)*, 1(103) :588–601, 1998.

[Scheirer, 2000] Eric D. Scheirer, Music listening systems, Master's thesis, PhD thesis. MIT Media Lab, 2000.

[Schwarz, 2000] D. Schwarz, A system for data-driven concatenative sound synthesis, In *Proceedings of DAFX00*, Verona (It), Dec 2000.

[Schwarz, 2004] D. Schwarz, *Data-Driven Concatenative Sound Synthesis*, PhD thesis, Université Paris 6 - Pierre et Marie Curie, 2004.

[Silvers, 2004] R. Silvers, Photomosaics, http ://www.photomosaics.com, 2004.

[Suzuki *et al.*, 2002] K. Suzuki, Y. Taki, H. Konagaya, P. Hartono, et S. Hashimoto, Machine listening for autonomous musical performance systems, In *Proc. of 2002 International Computer Music Conference, ICMA*, pages 61–64, San Francisco, 2002.

[Thayer, 1989] RE Thayer, *The biopsychology of mood and arousal*. Oxford University Press, 1989.

[Theodoridis et Koutroumbas, 1999] S. Theodoridis et K. Koutroumbas, *Pattern recognition*. Academic Press, academic press édition, 1999.

[Truchet *et al.*, 2001] C. Truchet, C. Agon, et G. Assayag, Cao et contraintes, In *Proceedings of 8th Journées d'Informatique Musicale*, Bourges (France), GMEB, 2001.

[Tsai *et al.*, 2003] W.O. Tsai, H.M. Wang, D. Rodgers, S.S. Cheng, et H.M. Yu, Blind clustering of popular music recordings based on singer voice characteristics, In *Proc. International Conference on Music Information Retrieval, ISMIR03*, 2003.

[Tzanetakis *et al.*, 2001a] George Tzanetakis, Georg Essl, et Perry Cook, Audio analysis using the discrete wavelet transform, In *Proc. WSES Int. Conf. Acoustics and Music : Theory and Applications (AMTA 2001)*, pages 205–210, Skiathos, Greece, 2001.

[Tzanetakis *et al.*, 2001b] George Tzanetakis, Georg Essl, et Perry Cook, Automatic musical genre classification of audio signals, In *Proceedings of 2nd International Symposium on Music Information Retrieval (ISMIR01)*, pages 205–210, Bloomington, IN, USA, Oct 2001.

[Tzanetakis et Cook, 1999] George Tzanetakis et Perry Cook, Multi-feature audio segmentation for browsing and annotation, In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA99)*, New Paltz, New York, Oct 1999.

[Tzanetakis, 2002] George Tzanetakis, Manipulation, analysis and retrieval systems for audio signals, 2002.

[Vapnik, 1998] V.N. Vapnik, *Statistical learning theory*. John Wiley and Sons, 1998.

[Vinet *et al.*, 2002] H. Vinet, P. Herrera, et F. Pachet, The cuidado project, In *Proceedings of the 3rd International Symposium on Music Information Retrieval*, éditeur IRCAM, 2002.

[Whitman *et al.*, 2001] B. Whitman, G. Flake, et S. Lawrence, Artist detection in music with minnowmatch, In *IEEE NNSP Workshop on Neural Networks for Signal Processing*, Falmouth, MA, September 2001.

[Williams et Ellis, 1999] G. Williams et D. Ellis, Speech/music discrimination based on posterior probability features, In *Proceedings Eurospeech-99, Budapest*, 1999.

[Witten *et al.*, 2000] I.H. Witten, F. Eibe, et Kaufmann M., *Data mining : Practical machine learning tools with Java implementations*. San Francisco, 2000.

[Wold *et al.*, 1996] E. Wold, T. Blum, D. Keislar, et J. Wheaton, Content-based classification, search, and retrieval of audio, *IEEE Multimedia*, 3(3) :27–36, 1996.

[Wold *et al.*, 1999] E. Wold, T. Blum, D. Keislar, et J. Wheaton, Classification, search, and retrieval of audio, CRC Handbook of Multimedia Computing, 1999.

[Zhang et Kuo, 1999] T. Zhang et C.C.J. Kuo, Heuristic approach for generic audio data segmentation and annotation, *ACM Multimedia, Orlando, USA*, pages 67–76, 1999.

[Zhang, 2003] T. Zhang, System method for automatic singer identification, In *IEEE International Conference on Multimedia and Expo*, Baltimore, MD, July 2003.

[Zils *et al.*, 2002] A. Zils, F. Pachet, O. Delerue, et F. Gouyon, Automatic extraction of drum tracks from polyphonic music signals, In *Proceedings of the International Conference on Web Delivering of music (WEDELMUSIC02)*, pages 179–183, Darmstadt, Germany, Dec 2002.

[Zils et Pachet, 2001] A. Zils et F. Pachet, Musical mosaicing, In *Proceedings of the 4th COST-G6 Conference on Digital Audio Effects (DAFX 01)*. University of Limerick, December 2001.

[Zils et Pachet, 2003] A. Zils et F. Pachet, Extracting automatically the perceived intensity of music titles, In *Proceedings of the 6th COST-G6 Conference on Digital Audio Effects (DAFX03)*. Queen Mary University, London, September 2003.